Université Paris VII – Denis Diderot
UFR Informatique

CRYPTOLOGIE SYMÉTRIQUE

Mémoire présenté et soutenu publiquement le 7 juillet 2008
à l'École normale supérieure, Paris

pour l'obtention de

l'Habilitation à Diriger des Recherches
de l'Université Paris VII – Denis Diderot

par

Henri GILBERT
Orange Labs

Composition du jury :

Rapporteurs	Eli BIHAM	Technion Haifa, Israel
	Kaisa NYBERG	H.U.T. Helsinki et Nokia, Finlande
	Bart PRENEEL	K.U.Leuven, Belgique
Examinateurs	Arnaud DURAND	Université Paris 7
	Marc GIRAULT	Orange Labs
	Jacques PATARIN	Université de Versailles
Directeur	Jacques STERN	ENS et Ingenico

Je remercie Jacques Stern, à qui la recherche française en cryptographie doit pour une large part sa remarquable vitalité actuelle, de m'avoir encouragé à entreprendre ce travail et de m'avoir fait l'honneur de le diriger, après avoir été le directeur de mon stage de DEA et le rapporteur de ma thèse.

Je suis très reconnaissant à Eli Biham, Kaisa Nyberg et Bart Preneel, cryptologues éminents dont le nom est associé à une grande partie des avancées accomplies au cours des dernières années dans le domaine de la cryptographie symétrique, d'avoir accepté d'être mes rapporteurs.

Je remercie Arnaud Durand d'avoir eu l'aimable obligeance de représenter l'Université Paris 7 dans mon jury. Je suis sensible à l'amitié que me font mon collègue Marc Girault, figure charismatique de la recherche à France Télécom, à qui je sais tout ce que je dois, et Jacques Patarin, chercheur profond et inventif avec qui j'ai eu un grand plaisir à coopérer depuis quelques années, d'avoir accepté d'en faire partie.

Dans les laboratoires de recherche et développement de France Télécom[1] *où j'ai rejoint puis dirigé depuis plus de dix ans la recherche en cryptographie, j'ai bénéficié au sein du laboratoire « sécurité » d'un entourage professionnel exceptionnellement sympathique et stimulant. Je remercie les responsables successifs de ce laboratoire : Mireille Campana, David Arditti et Thierry Baritaud, ainsi que Sébastien Nguyen Ngoc et Fabrice Clerc, responsables dans ce laboratoire des unités de recherche isséenne et caennaise qui hébergent la recherche en cryptographie, pour l'enthousiasme et la bonne humeur qu'ils ont su communiquer à leurs équipes.*

Le fleuve de la recherche en entreprise n'est pas aussi tranquille qu'il pourrait y paraître : je remercie Gérard Barberye, Jean-Marc Pitié, Pierre Rolin et Paul Friedel, directeurs de divisions de la R&D et directeurs scientifiques, d'avoir su comprendre l'utilité de la recherche sur les techniques de sécurité et la cryptographie et la préserver des années durant des turbulences inévitables. Fluctuat nec mergitur.

La majorité des résultats présentés dans ce mémoire résultent d'études réalisées en coopération avec les quatre doctorants de France Télécom dont j'ai eu le plaisir d'encadrer la thèse. Je tiens donc à remercier chaleureusement Marine Minier, Olivier Billet, Côme Berbain et Thomas Peyrin. Je remercie non moins chaleureusement tous les autres coauteurs de mes publications : David Arditti, Thierry Baritaud, Mireille Campana, Guy Chassé, Pascal Chauvaud, Carlos Cid, Christophe Debaert, Charaf Ech-Chatbi, Marc Girault, Dipankar Gupta, Helena Handschuh, Thomas Johansson, Antoine Joux, Alexander Maximov, Gérald Mazziotto, Frédéric Muller, Andrew Odlyzko, Jacques Patarin, Jean-Jacques Quisquater, Matt Robshaw, Yannick Seurin, Hervé Sibert, Anne Tardy-Cordir, Rémi Thomas et Serge Vaudenay, ainsi que les auteurs des articles ou rapports collectifs que j'ai cosignés.[2]

[1] successivement appelés le CNET, FT R&D et Orange Labs

[2] Ces articles portent sur l'analyse de candidats AES, la sécurité de l'AES, et les propositions d'algorithmes symétriques DFC, Sosemanuk et Decim.

Merci également aux membres anciens ou actuels du laboratoire « sécurité » et plus généralement aux collègues de France Télécom dont je n'ai pas encore cité le nom et avec qui j'ai eu la chance de travailler depuis que j'exerce le métier de cryptologue : François Allègre, Ryad Benadjila, Florent Bersani, Françoise Broussaud, Jacques Burger, Laurent Butti, Sylvie Camus, Sébastien Canard, Frédéric Cherbonnier, Jean-Michel Combes, Pierre Crégut, Evelyne Czubak, Hervé Debar, Jonathan Etrog, Sylvie Fouquet, Stanislas Francfort, Laurent Frisch, Eric Gauthier, Aline Gouget, Louis Guillou, Loïc Houssier, Francis Klay, Bruno Labbé, Gilles Macario-Rat, Eric Malville, Daniel Migault, Michel Milhau, Jean-François Misarsky, Eric Mora, Dimitri Mouton, Daniel Mustaki, Adam Ouorou, Jean-Claude Paillès, Hieu Phan, Murielle Philippe, Anne-Sophie Pignol, Paul Richy, Nathalie Rogier, Gérard Sabelete, Alain Scheiwe, Jacques Traoré et Franck Veysset, ainsi que les étudiants dont j'ai encadré le stage de DEA. Je salue au passage mes collègues et amis du groupe d'experts en cryptographie ETSI/SAGE, Miklos Santha qui a été mon directeur de thèse, Helena Handschuh avec qui j'ai coorganisé à Paris la conférence FSE 2005, et les nombreux chercheurs et ingénieurs dont j'ai fait au fil des ans la connaissance à l'occasion des innombrables conférences, réunions de normalisation et réunions de projets coopératifs qui ont rythmé ma vie professionnelle. Je remercie Côme Berbain, Olivier Billet et Yannick Seurin pour l'aide qu'il m'ont apportée dans la préparation de ce mémoire[3] *et Marie-Claude Thill pour m'avoir patiemment guidé dans le labyrinthe de la procédure d'habilitation de l'Université Paris 7. Last but not least, merci à ma famille pour tout ce que l'auteur de ce mémoire lui doit.*

Puissent les personnes que j'ai involontairement omis de citer dans ces remerciements bien vouloir m'en excuser.

[3]Ils ne sauraient être tenus pour responsables des coquilles et imprécisions que faute de temps je n'ai pas réussi à corriger.

Table des matières

Introduction

Cryptographie

La **cryptologie** traite des techniques algorithmiques de protection de l'authenticité, de l'intégrité et de la confidentialité de l'information. L'objet de la **cryptographie**, au sens restreint du terme, est la conception de telles techniques, celui de la **cryptanalyse** l'étude ou la mise en œuvre d'attaques susceptibles de mettre leur sécurité en défaut. Dans son acception la plus large qui est aussi la plus courante, le terme de cryptographie recouvre ces deux aspects complémentaires et est synonyme de cryptologie. La cryptographie s'inscrit dans une histoire multimillénaire. Mais elle s'est radicalement transformée avec l'avènement de l'informatique et des technologies de l'information, avec lesquelles elle a partie liée. A la suite de la découverte, à la fin des années 70, des concepts fondateurs de la cryptographie à clé publique [80], elle s'est subdivisée en deux branches : la **cryptographie symétrique**, qui traite des algorithmes de sécurité fondés sur le partage d'un secret entre deux entités et d'algorithmes apparentés comme les fonctions de hachage, et la **cryptographie asymétrique**, qui traite des algorithmes de sécurité fondés sur la détention par une entité d'une clé privée de déchiffrement, de signature, d'authentification ou d'établissement de clé non partagée. Le Data Encryption Standard, que l'on peut considérer comme l'un des tout premiers algorithmes de chiffrement symétrique modernes, n'a été publié qu'en 1977 [174], un an seulement avant la découverte de l'algorithme de chiffrement asymétrique RSA [180].

Cryptographie symétrique et asymétrique sont complémentaires et ont l'une et l'autre connu un développement considérable au cours des trente dernières années. Les propriétés que les algorithmes symétriques doivent satisfaire, comme la dépendance à sens unique en le secret des valeurs de sortie qu'ils produisent, ne sont pas tout à fait aussi paradoxales et contraignantes que celles des algorithmes asymétriques. De ce fait, les algorithmes symétriques actuels reposent sur des opérations élémentaires moins structurées, et sont irremplaçables pour le traitement de données en masse, avec des débits de chiffrement ou d'authentification de messages cent à mille fois supérieurs. Mais la nécessité de disposer de secrets partagés rend la cryptographie symétrique insuffisante pour protéger à elle seule les communications entre des entités reliées à de grands réseaux non centralisés comme l'Internet. En pratique, dans un grand nombre de systèmes de communication, l'usage de la cryptographie asymétrique est réservé à l'authentification mutuelle et à l'établissement d'une clé partagée entre des entités ne possédant pas de lien préalable tandis que la cryptologie symétrique est mise en œuvre

pour protéger le canal sécurisé ainsi établi.

L'évolution de la cryptographie est gouvernée par trois principaux moteurs[4] : (1) la **conception** de nouveaux algorithmes et de nouvelles constructions applicables à la sélection de leur structure ou leurs composantes (2) les progrès de la **cryptanalyse**, c'est-à-dire la découverte de nouvelles attaques ou familles d'attaques permettant d'invalider certains de ces algorithmes ou de ces constructions et (3) la découverte de **preuves de sécurité**, permettant à l'inverse de valider certains aspects de leur résistance aux attaques dans un modèle de sécurité comportant nécessairement certaines limitations.[5] Si comme l'affirme K. Popper ce qui caractérise une théorie scientifique est que son exactitude est potentiellement réfutable, et non prouvable de manière absolue, alors la cryptographie, qui produit des algorithmes dont la sécurité est potentiellement réfutable par la cryptanalyse, est à n'en pas douter une discipline scientifique.

Dans le domaine de la cryptographie symétrique, la dialectique entre conception et cryptanalyse a jusqu'à maintenant largement prévalu, la cryptanalyse opérant une sorte de sélection darwinienne parmi les nombreuses propositions des concepteurs d'algorithmes. Certains aspects de la sécurité des algorithmes symétriques peuvent cependant être validés à l'aide de preuves partielles de sécurité, et l'on peut s'attendre à ce qu'à l'avenir de telles preuves soient de plus en plus systématiquement exigées. Dans la branche de la cryptographie asymétrique actuellement dominante qui traite des algorithmes fondés sur l'arithmétique et la théorie des nombres, il est depuis quelques années usuel d'exiger que la sécurité des algorithmes soit fondée sur des preuves réductionnistes. Cette exigence opère une présélection forte parmi les nouvelles propositions d'algorithmes. Elle aboutit à déplacer la charge d'invalider les algorithmes offrant une sécurité insuffisante de la cryptanalyse vers des domaines des mathématiques et de l'algorithmique susceptibles d'intéresser non seulement les cryptanalystes, mais également une communauté scientifique plus large.

Cryptographie symétrique

Les algorithmes symétriques peuvent être regroupés en un nombre restreint de catégories : algorithmes de **chiffrement**, algorithmes d'**authentification d'entité**,

[4] auxquels il convient d'ajouter les facteurs déterminants sur le long terme que sont les progrès des mathématiques et de l'informatique

[5] Il n'existe dans l'état actuel des connaissances, du moins en dehors du domaine très restreint des algorithmes dits inconditionnellement sûrs, aucun algorithme cryptographique dont on puisse établir une preuve absolue de sécurité. Aucune technique connue ne permet en effet d'établir les bornes inférieures requises sur la complexité de l'algorithme le plus efficace permettant de résoudre le problème calculatoire auquel un adversaire est confronté. L'on ne parvient de ce fait, dans le meilleur des cas, qu'à obtenir (a) des **preuves de sécurité réductionnistes** établissant que s'il existait une attaque efficace contre un algorithme donné, on pourrait en déduire un algorithme efficace de résolution d'un problème mathématique connu et conjecturé difficile ou (b) des **preuves partielles de sécurité** ne permettant d'établir sa solidité que dans un modèle de sécurité restreint où seules certaines classes particulières d'attaques sont considérées.

algorithmes d'**authentification de messages**[6], **fonctions de hachage**. Parmi ces familles d'algorithmes, celle des algorithmes de chiffrement occupe une place centrale : leur conception et leur analyse ont de loin été les plus étudiées, et les algorithmes des autres catégories leur sont le plus souvent très directement reliés. Les algorithmes de chiffrement symétrique se subdivisent en deux principales classes, dont une caractérisation informelle sera proposée dans la suite : les **algorithmes de chiffrement par blocs** et les **algorithmes de chiffrement à flot**. La différence essentielle entre ces deux types d'algorithmes me semble être la suivante : un algorithme par blocs est un **générateur de fonctions** qui à une clé secrète associe une permutation de l'ensemble des blocs de n symboles d'un alphabet fini, alors qu'un algorithme à flot est un **générateur de nombres** qui à une clé secrète associe une suite de symboles d'un alphabet fini appelée suite chiffrante.

La cryptographie symétrique est une discipline vivante, dont l'une des spécificités les plus attachantes est que l'on y est confronté d'emblée, d'une manière plus immédiate et provocatrice encore me semble-t-il qu'en cryptographie asymétrique, aux limites des connaissances actuelles dans les domaines de l'algorithmique et de la théorie de la complexité.[7]

Notions de sécurité employées en cryptographie symétrique

Quatre notions relevant de la théorie de la complexité et fréquemment utilisées dans la suite pour caractériser les propriétés de sécurité attendues des algorithmes symétriques méritent me semble-t-il d'être définies ici d'emblée de manière informelle. Il s'agit des notions de générateur pseudoaléatoire de nombres ou **PRNG** (de l'anglais Pseudo Random Number Generator) et de générateur pseudoaléatoire de fonctions ou **PRF** (de l'anglais Pseudo Random Function Generator), et de deux variantes de cette dernière : les notions de générateur pseudoaléatoire de permutations ou **PRP** (de l'anglais Pseudo Random Permutation Generator) et de générateur super-pseudoaléatoire de permutations ou **SPRP**. Les définitions qui suivent sont formulées dans ce qu'il est convenu d'appeler le modèle de la **sécurité concrète** et non dans le modèle dit de la **sécurité asymptotique**. Les objets élémentaires considérés dans le modèle de la sécurité asymptotique sont des familles infinies de fonctions indexées par un paramètre de sécurité comme la taille de leur entrée, ce qui permet de définir les problèmes de

[6] également appelés codes d'authentification de messages ou MAC, de l'anglais « Message Authentication Code »

[7] Ainsi par exemple, l'on est incapable dans l'état actuel des connaissances de décrire explicitement une fonction f de 128 bits vers 128 bits dont on sache prouver qu'elle est à sens unique en ce sens qu'elle est calculable avec des ressources de calcul très modestes, mais qu'il n'existe aucun algorithme A nécessitant des ressources de calcul accessibles dans un avenir prévisible et permettant d'inverser f en moins d'un an avec une probabilité de succès d'au moins $\frac{1}{2}$, i.e. satisfaisant $\Pr_{x\in\{0,1\}^{128}}(f(A(f(x))) = f(x)) \geq \frac{1}{2}$. Cependant, la composition de quelques dizaines d'opérations linéaires modulo 2 sur l'alphabet $\{0,1\}^8$ des octets et de quelques dizaines d'appels à une permutation fixe bien choisie de ce même alphabet permet de définir une fonction dont on peut raisonnablement conjecturer qu'elle possède ces propriétés : tel est notamment le cas de la fonction f induite par la version de l'AES de taille de clé 128 bits et qui à une clé associe le chiffré du bloc nul.

sécurité qui leur sont associés comme faciles ou difficiles selon qu'il existe ou non un algorithme de temps de calcul polynomial (modèle de sécurité uniforme) ou une famille de circuits de taille polynomiale (modèle de sécurité non uniforme) pour les résoudre. Dans le modèle de la sécurité concrète, les objets élémentaires considérés sont des fonctions ou des familles de fonctions de taille d'entrée et de sortie fixée, et un problème de sécurité associé est considéré comme difficile lorsque la complexité en temps de tout algorithme permettant de le résoudre est supérieure à un seuil choisi pour refléter dans un contexte déterminé ce qui est considéré comme un niveau de sécurité acceptable, par exemple 2^{80} opérations élémentaires.[8] Dans l'un et l'autre de ces modèles, aucune preuve absolue de sécurité ne peut être établie dans l'état actuel des connaissances.

PRNG : Soit n et $m > n$ deux entiers. On définit un **générateur de nombres** comme une fonction $f : \{0,1\}^n \longrightarrow \{0,1\}^m$ qui à un germe secret x de n bits associe un mot binaire $f(x)$ de m bits. On désigne par **distingueur** d'un générateur de nombres de n bits vers m bits tout algorithme probabiliste de test A prenant pour valeur d'entrée un mot de m bits et produisant une sortie binaire (0 ou 1). On définit l'avantage de A (ou avantage de A au sens d'un PRNG) pour distinguer f d'un générateur de nombres parfaitement aléatoire comme la quantité $\mathbf{Adv}_f^{PRNG}(A) = \left|\Pr_{x\in\{0,1\}^n}(A(f(x)) = 1) - \Pr_{y\in\{0,1\}^m}(A(y) = 1)\right|$, où les probabilités sont prises non seulement sur les valeurs de x et y sélectionnées dans $\{0,1\}^n$ et $\{0,1\}^m$ selon la loi uniforme comme indiqué explicitement, mais également sur les choix aléatoires utilisés par l'algorithme probabiliste A. On définit l'**avantage** (au sens d'un PRNG) pour distinguer f en temps t comme la quantité $\mathbf{Adv}_f^{PRNG}(t) = \max_A\{\mathbf{Adv}_f^A\}$, où le maximum est pris sur l'ensemble des distingueurs A acceptant une valeur d'entrée de m bits et de temps de calcul inférieur ou égal à t. Un générateur de nombres f est un **PRNG** si $\mathbf{Adv}_f^{PRNG}(t)$ est négligeable (i.e. dans le contexte considéré ici inférieur à une quantité ε fixée, par exemple 2^{-40}) pour des valeurs de t strictement inférieures à un seuil fixé, par exemple $t = 2^{80}$ opérations élémentaires.

PRF : On définit un **générateur de fonctions** de n bits vers m bits comme une famille $F = \{f_K\}$ de fonctions $f_K : \{0,1\}^n \longrightarrow \{0,1\}^m$ indexées par un paramètre K sélectionné dans un ensemble de clés $\mathcal{K}$ muni de la loi uniforme. On désigne par **distingueur** d'un générateur de fonctions de n bits vers m bits tout algorithme probabiliste de test $A^{\mathcal{O}}$ capable d'interroger un oracle $\mathcal{O} : \{0,1\}^n \longrightarrow \{0,1\}^m$, et pro-

[8] Dans le modèle de la sécurité concrète, afin d'échapper aux paradoxes qu'entraînerait l'absence de toute limitation de la taille de programme des algorithmes considérés (comme toute fonction fixée de n bits vers m bits peut être inversée en une seule opération de lecture d'une table à l'aide d'un programme de taille suffisante, la notion de fonction à sens unique, sur laquelle la cryptographie symétrique est fondée, perdrait toute signification), il est d'usage d'incorporer dans l'expresssion du temps de calcul d'un algorithme un terme représentant la taille de sa description dans un langage arbitraire fixé, exprimée par exemple en bits. Cette convention sera implicitement utilisée dans la suite. Une autre manière, non retenue ici, de résoudre cette difficulté est de représenter un algorithme dont les tailles des paramètres d'entrée et de sortie sont fixées par un circuit, et de mesurer la complexité d'un tel algorithme non par son temps de calcul, mais par la taille du circuit associé.

duisant une réponse binaire. Un tel algorithme de test permet de distinguer une instance aléatoire f de F d'une instance aléatoire de la famille $F^*_{n,m}$ de toutes les fonctions de n bits vers m bits munie de la loi uniforme. On définit l'avantage de A pour distinguer F d'un générateur de fonctions parfaitement aléatoire comme la quantité $\mathbf{Adv}_F^{PRF}(A) = \left|\Pr(A^{f_K} = 1) - \Pr(A^{f^*} = 1)\right|$, où les probabilités sont prises sur $K \in \mathcal{K}$ (resp $f^* \in F^*_{n,m}$) et sur les choix aléatoires utilisés par l'algorithme probabiliste A. On définit l'**avantage** (au sens d'une PRF) pour distinguer F d'un générateur de fonctions parfaitement aléatoire en temps t à l'aide de q questions comme la quantité $\mathbf{Adv}_F^{PRF}(t,q) = \max_A\{\mathbf{Adv}_F^{PRF}(A)\}$, où le maximum est pris sur l'ensemble des algorithmes de test $A^{\mathcal{O}}$ de temps de calcul inférieur ou égal à t et posant au plus q questions à un oracle de n bits vers m bits. Une famille de fonctions $F = \{f_K\}$ est une **PRF** si $\mathbf{Adv}_F^{PRF}(t,q)$ est négligeable (i.e. dans le contexte considéré ici inférieur à une quantité ε fixée, par exemple 2^{-40}) pour toutes valeurs de t et q strictement inférieures à des seuils fixés, par exemple $t = 2^{80}$ et $q = 2^{80}$.

PRP et SPRP : La définition d'une PRP et d'une SPRP de n bits vers n bits se déduit simplement de la définition d'une PRF rappelée ci-dessus. On définit un **générateur de permutations** comme une famille $F = \{f_K\}$ de permutations $f_K : \{0,1\}^n \longrightarrow \{0,1\}^n$ indexées par un paramètre K sélectionné dans un ensemble de clés $\mathcal{K}$ muni de la loi uniforme. Un distingueur au sens d'une PRP (respectivement d'une SPRP) d'un générateur de permutations est un algorithme de test probabiliste $A^{\mathcal{O}}$ (respectivement $A^{\mathcal{O},\mathcal{O}^{-1}}$) capable d'interroger un oracle $\mathcal{O}$ représentant une permutation de n bits vers n bits (respectivement l'oracle $\mathcal{O}$ et l'oracle $\mathcal{O}^{-1}$ représentant la permutation inverse). Un tel algorithme permet de distinguer une instance aléatoire f de F d'une instance aléatoire f^* de la famille P^*_n de toutes les permutations de n bits vers n bits munie de la loi uniforme. On définit l'avantage de A au sens d'une PRP (respectivement d'une SPRP) pour distinguer F d'un générateur de permutations parfaitement aléatoire comme la quantité $\mathbf{Adv}_F^{PRP}(A) = \left|\Pr(A^{f_K} = 1) - \Pr(A^{f^*} = 1)\right|$ (respectivement $\mathbf{Adv}_F^{SPRP}(A) = \left|\Pr(A^{f_K, f_K^{-1}} = 1) - \Pr(A^{f^*, f^{*-1}} = 1)\right|$). On définit l'avantage au sens d'une permutation pseudoaléatoire (resp. au sens d'une permutation super pseudoaléatoire) pour distinguer F d'un générateur de permutations parfaitement aléatoire en temps t à l'aide de q questions comme la quantité $\mathbf{Adv}_F^{PRP}(t,q) = \max_A\{\mathbf{Adv}_F^{PRP}(A)\}$ (resp. $\mathbf{Adv}_F^{SPRP}(t,q) = \max_A\{\mathbf{Adv}_F^{SPRP}(A)\}$), où le maximum est pris sur l'ensemble des algorithmes de test A de temps de calcul inférieur ou égal à t et posant au plus q questions en tout à l'oracle $\mathcal{O}$ (respectivement à l'un des oracles $\mathcal{O}$ et $\mathcal{O}^{-1}$). La famille de permutations $F = \{f_K\}$ est une **PRP** (respectivement une **SPRP**) si $\mathbf{Adv}_F^{PRP}(t,q)$ (respectivement $\mathbf{Adv}_F^{SPRP}(t,q)$) est négligeable, i.e. dans le contexte considéré ici inférieur à une quantité ε fixée, par exemple 2^{-40}, pour toutes valeurs de t et q strictement inférieures à des seuils fixés, par exemple $t = 2^{80}$ et $q = 2^{80}$.

Objet de ce mémoire

La majeure partie de mes travaux en cryptographie depuis 1990, date de mes débuts dans cette discipline, relève de la cryptographie symétrique. Les circonstances ont à l'évidence joué un rôle dans cette spécialisation ; mais sans doute mes préférences dans le domaine des mathématiques et de l'algorithmique ont-elles également influé. Je me suis, le plus souvent, attaché davantage à l'étude des propriétés structurelles globales des algorithmes qu'à l'analyse de leurs composantes élémentaires.

Le présent mémoire est structuré en deux parties, qui récapitulent mes principales contributions à l'étude des algorithmes de chiffrement par blocs et des algorithmes de chiffrement à flot et les situent dans le contexte des évolutions récentes de ces deux branches de la cryptographie symétrique. Mes contributions à d'autres aspects de la recherche en cryptographie comme la cryptographie multivariable, les fonctions de hachage et la cryptanalyse de certains algorithmes asymétriques ne sont pas présentées ici. Dans chacune des deux parties, les principaux aspects abordés sont d'une part la cryptanalyse, et d'autre part la conception de nouveaux algorithmes ou l'investigation de nouveaux paradigmes, notamment à travers la recherche de constructions à sécurité partiellement prouvée.

Première partie

Algorithmes par blocs

1 Introduction

Les algorithmes de chiffrement par blocs représentent la primitive la plus universelle de la cryptographie symétrique. L'existence de modes opératoires variés permettant de les employer non seulement à des fins de chiffrement, mais aussi d'authentification d'entité ou de messages et même de hachage leur confère en effet une très grande souplesse d'utilisation. Une grande partie de l'effort de recherche considérable consacré au cours des trente dernières années à la cryptographie symétrique a porté sur leur conception et leur analyse.

Un algorithme de chiffrement par blocs peut être défini comme un **générateur de permutations** : il associe à une clé de chiffrement[9] K une permutation E_K de l'ensemble Σ^n des **blocs** de n symboles d'un alphabet fini Σ. Cet alphabet est en pratique presque toujours l'alphabet binaire $\{0,1\}$, et tel sera le cas sauf mention explicite du contraire de tous les algorithmes par blocs qui seront considérés dans la suite. Pour chiffrer un message M de longueur quelconque, on peut découper M en une suite de blocs de n symboles (après ajout éventuel d'un suffixe pour se ramener à un nombre entier de blocs) et enchaîner les appels à E_K selon différents **modes opératoires**.[10]

Pour être considéré comme cryptographiquement sûr, un algorithme par blocs doit satisfaire les conditions suivantes, énoncées ici informellement. Il doit être informatiquement impossible à un adversaire capable d'accéder, pour un grand nombre de blocs de clair ou de chiffré de son choix éventuellement choisis de façon adaptative, aux fonctions de chiffrement E_K et de déchiffrement D_K associées à une clé inconnue K, d'en déduire la valeur de K ou une description équivalente de E_K ou D_K, ou de prédire avec une probabilité de succès non négligeable l'image par E_K ou D_K de nouveaux blocs de clair ou de chiffré. Ces conditions imposent que la taille de la clé soit suffisante, par exemple 80 ou 128 bits. Elles imposent également que l'algorithme possède une taille de bloc n suffisante, par exemple 128 bits, pour empêcher qu'un adversaire ne soit en mesure de mémoriser une fraction importante des 2^n valeurs de la fonction E_K et de la fonction inverse.[11] Il doit plus généralement être informatiquement impossible à un tel adversaire de distinguer la permutation E_K d'une permutation aléatoire parfaite à l'aide des couples (entrée, sortie) observés. Cette dernière condition peut être formalisée en définissant un algorithme par blocs sûr $(E_K)_{K\in\mathcal{K}}$ comme un **générateur super-pseudoaléatoire de permutations** de l'ensemble des blocs de n bits ou **SPRP**, notion introduite au chapitre précédent. L'observation « en boîte

[9] L'espace des clés dans lequel la clé K est sélectionnée est généralement l'ensemble $\mathcal{K} = \{0,1\}^k$ des clés de longueur k bits muni de la loi de probabilité uniforme.

[10] Les modes opératoires les plus usuels sont les modes « dictionnaire » ou Electronic Codebook (ECB), « chaînage des blocs chiffrés » ou Cipher Block Chaining (CBC), « rétroaction de la sortie » ou Output Feedback (OFB), « compteur » (CTR), et « authentification de messages par chaînage des blocs chiffrés » ou Cipher Block Chaining Message Authentication Code (CBC-MAC), mais un grand nombre d'autres modes opératoires ont été proposés au cours des dernières années.

[11] Il n'est pas impossible de donner un sens à la notion d'algorithme par blocs sûr pour des valeurs faibles de n, par exemple n=32, à condition cependant de reformuler légèrement certaines des conditions de sécurité énoncées ici afin de ne considérer que la résistance à des attaques qui seraient inapplicables à une permutation aléatoire de $\{0,1\}^n$.

noire » d'un algorithme de chiffrement par blocs sûr et de l'algorithme de déchiffrement associé ne doit donc pas même permettre d'identifier cet algorithme.[12]

algorithme	**taille de clé**	**taille de bloc**	**proposé par**
DES	56	64	IBM [174]
TDES	112, 168	64	IBM [174]
IDEA	128	64	J. Massey et X. Lai [150]
MISTY	128	64	M. Matsui [158]
KASUMI	128	64	3GPP [86]
SERPENT	128, 192, 256	128	E. Biham, R. Anderson et L. Knudsen [22]
AES	128, 192, 256	128	J. Daemen et V. Rijmen [173, 76]

TAB. 1 – Caractéristiques de quelques algorithmes de chiffrement par blocs

Presque tous les algorithmes par blocs proposés à ce jour sont **itératifs** : la fonction de chiffrement E_K est la composée de r fonctions élémentaires bijectives appelées **tours**, paramétrées par des **sous-clés** K_1 à K_r. La **clé étendue** $(K_1, \ldots, K_r)$ est déduite de la clé K à l'aide d'un schéma de génération de sous-clés (en anglais key schedule). Les algorithmes par blocs existants peuvent être regroupés en un petit nombre de classes selon la structure de la transformation appliquée à chaque tour : **schémas de Feistel** lorsque le bloc courant X_i ($i = 0$ à r) est constitué d'un registre (L_i, R_i) de deux mots de w bits, et évolue sous l'action d'une fonction f_{K_i} de w bits vers w bits selon les relations $L_{i+1} = R_i$ et $R_{i+1} = f_i(R_i) \oplus L_i$, ou plus généralement $R_{i+1} = f_i(R_i) * L_i$, où $*$ désigne une loi de groupe, par exemple l'addition modulo 2^w ; **schémas de Feistel généralisés** lorsque le bloc courant X_i est un registre constitué de $m > 2$ mots de w bits et lorsqu'en outre X_{i+1} ($i = 0$ à $r-1$) est une fonction bijective de X_i dont l'un au moins des k mots (par exemple le mot de gauche) est égal à l'un des k mots de X_i (par exemple le mot de droite) ; **schémas S/P** ou réseaux de substitutions-permutations lorsque le bloc courant est constitué de m mots de w bits, et lorsque la fonction de tour est la composée (1) de la transformation consistant à appliquer à chacun de ces m mots une **boîte S** (i.e. une bijection de w bits vers w bits) précédée d'un ou exclusif avec un mot de la sous-clé courante ou dépendant de la clé courante, et (2) d'une transformation linéaire ou non linéaire bijective assurant la diffusion de l'influence de chaque mot du

[12] Aux exigences précédentes qui sont résumées par la notion de SPRP, on ajoute généralement que l'algorithme doit résister aux **attaques à clés reliées**, notion introduite par E. Biham dans [21]. Une notion d'attaque similaire a été proposée indépendamment par L. Knudsen dans un article consacré à la cryptanalyse de LOKI'91 [143]. Dans un attaque à clés reliées, un adversaire est capable d'accéder non seulement à des clairs et chiffrés liés par une clé inconnnue K, mais également à des clairs et chiffrés liés par une ou plusieurs clés déduites de K par une transformation bijective, par exemple un « ou exclusif » avec une quantité choisie par l'adversaire. Une exigence de sécurité supplémentaire, celle de résistance à des attaques à l'aide d'un **distingueur à clés connues**, a été proposée récemment par L. Knudsen et V. Rijmen [144]. Elle est motivée par l'observation que si l'on souhaite utiliser un algorithme par blocs à d'autres fins que le chiffrement comme par exemple l'obtention d'une fonction de hachage, les conditions de sécurité énoncées précédemment ne suffisent pas. Il n'existe pas actuellement de formalisation rigoureuse de la notion de distingueur à clés connues évitant l'écueil des résultats d'impossibilité auxquels conduit la notion apparentée de « correlation intractability » [50].

bloc d'entrée vers plusieurs mots du bloc de sortie.[13] Le tableau 1 rappelle les longueurs de clé et de bloc de quelques algorithmes par blocs de référence.

2 Les débuts de la cryptanalyse différentielle et de la cryptanalyse linéaire

Mes deux premières publications en cryptographie [105, 195], qui résultaient d'études menées en coopération avec mes collègues G. Chassé et A. Tardy-Corfdir, décrivent respectivement une cryptanalyse statistique à clair choisi et à clair connu de versions réduites de l'algorithme par blocs japonais FEAL [190, 165]. Elles peuvent être considérées rétrospectivement comme une cryptanalyse différentielle et une cryptanalyse linéaire, bien qu'elles ne se réfèrent pas, et pour cause, à ces deux méthodes : la première n'avait pas encore été divulguée par E. Biham et A. Shamir[14] [32, 33, 34] et la seconde n'avait pas encore été découverte par M. Matsui [156, 157]. Notre attaque [105] se fondait sur une analyse des caractéristiques différentielles de FEAL moins complète que la cryptanalyse de FEAL qu'E. Biham et A. Shamir avaient découverte [33]. Mais cette limitation était dans une certaine mesure compensée par la technique d'estimation de la clé que nous mettions en œuvre, qui reposait sur des approximations de combinaisons linéaires de bits de différence.[15] Notre attaque de [105] a semble-t-il partiellement influé sur la découverte par M. Matsui de la cryptanalyse linéaire [156, 157]. Paradoxalement, tel n'est pas le cas de notre attaque de [195], qui relève pourtant beaucoup plus directement de la cryptanalyse linéaire.

Dans les années qui ont suivi leur découverte, la cryptanalyse différentielle et la cryptanalyse linéaire ont été appliquées à l'analyse de versions complètes ou réduites d'un grand nombre d'algorithmes par blocs ainsi qu'à l'attaque de fonctions de hachage. L'attaque différentielle de la fonction de hachage FFT de C. Schnorr que j'ai étudiée en collaboration avec T. Baritaud et M. Girault [11] et l'attaque différentielle de l'algorithme par blocs KHUFU que j'ai étudiée en collaboration avec P. Chauvaud [106] s'inscrivent dans ce contexte.[16] Ces premiers travaux sur la cryptanalyse statistique des algorithmes par blocs sont décrits en détail dans ma thèse [101], ainsi qu'une variante

[13] Bien que l'on oppose ici les schémas de Feistel aux schémas S/P comme deux méthodes différentes pour produire une fonction de tour bijective, les transformations élémentaires utilisées dans les tours d'un schéma S/P peuvent être également utilisées dans la fonction f d'un schéma de Feistel.

[14] Le mystère qui entourait l'annonce, faite par A. Shamir lors de la conférence Sécuricom 89, de la découverte par E Biham et lui-même d'une nouveau type d'attaque applicable aux algorithmes DES et FEAL avait partiellement inspiré notre recherche. Le principe de la cryptanalyse différentielle et le détail de l'attaque du DES ne furent révélés par E. Biham et A. Shamir qu'en 1990, lors de la conférence Crypto'90, et le détail de leur attaque de FEAL ne le fut qu'un an plus tard, lors de la conférence Eurocrypt'91.

[15] Cette méthode peut rétrospectivement être rattachée à la cryptanalyse différentielle-linéaire, notion proposée par S. Langford et M. Hellman en 1994 [151].

[16] Une autre attaque de FFT reposant sur des techniques similaires et conduisant également à des collisions fut découverte indépendamment à la même époque par J. Daemen *et al.* [73]. Une version consolidée de FFT, intitulée FFT2, fut également cassée quelques mois plus tard par S. Vaudenay [196].

de l'attaque de D. Davies contre le DES [78] que j'ai mise au point en 1990 et 1991. Cette dernière attaque nécessite, pour la version standard du DES, un nombre de blocs de clair et un nombre d'opérations voisins de 2^{52}. Une variante similaire de l'attaque de Davies fut découverte indépendamment par E. Biham et A. Biryukov en 1994 [23, 24].

Au cours de cette même période, les notions de transition différentielle et linéaire[17] (en anglais « **differential** » et « **linear hull** ») furent introduites. Des bornes reliant les probabilités maximales des transitions différentielles ou linéaires d'un schéma de Feistel à r tours paramétré par des clés indépendantes à celles des fonctions ou permutations $f_1, f_2, \ldots, f_r$ intervenant à chaque tour furent établies par K. Nyberg et L. Knudsen [171, 170] et K. Aoki et K. Ohta [4], et réutilisées par M. Matsui aux différents niveaux d'une construction imbriquée pour aboutir à un algorithme par blocs complet, offrant une résistance prouvée contre les attaques linéaires et différentielles, l'algorithme par blocs MISTY [158].

3 La compétition AES et le développement de la cryptanalyse statistique

Dans le domaine de la cryptographie symétrique, la fin de la décennie 1990–2000 fut marquée par l'appel à propositions du NIST américain[18] en vue de la sélection de l'Advanced Encryption Standard (AES). Cette initiative était motivée par la nécessité de remplacer graduellement le Data Encryption Standard (DES), dont la longueur de clé (56 bits) était devenue nettement insuffisante et dont les performances pour le chiffrement en logiciel étaient sous-optimales. Le cahier des charges imposé par le NIST était très simple : le nouvel algorithme devait posséder une taille de bloc unique de 128 bits, trois longueurs de clé (128, 192 et 256 bits), une sécurité suffisante pour que l'on puisse raisonnablement conjecturer l'algorithme hors d'atteinte des progrès de la cryptanalyse pendant plusieurs décennies, et les meilleures performances logicielles et matérielles possibles. Parmi les quinze candidatures reçues, qui émanaient de quatre continents, près d'une dizaine, parmi lesquels les cinq algorithmes retenus comme finalistes, étaient hors de portée des méthodes cryptanalytiques connues et plus performants en logiciel que l'algorithme de référence de l'époque, 3-DES.

La compétition AES fit progresser l'état de l'art de la conception des algorithmes par blocs. Bien qu'une certaine consanguinité soit perceptible entre les différentes propositions, la palette des techniques de conception mises en œuvre était relativement large. Je fus associé à l'une des candidatures, celle de l'algorithme DFC, présentée par l'ENS et France Télécom [107, 12]. Mais ma participation au travail de spécification fut à vrai dire des plus modeste puisqu'elle se limita essentiellement à une contribution à

[17] Une transition différentielle (resp. linéaire) à r tours désigne un triplet constitué d'une différence (resp. d'un masque) d'entrée, d'une différence (resp. d'un masque) de sortie, et d'une probabilité moyenne de transition, la moyenne étant prise sur les clés et les entrées. Ces notions permettent de quantifier la résistance d'un algorithme aux attaques différentielles et linéaires et sont complémentaires de celle de caractéristique.

[18] National Institute for Standards and Technology

l'analyse de la sécurité de versions réduites de l'algorithme. Conçu principalement par S. Vaudenay, DFC reposait d'une manière très directe sur la théorie de la décorrélation que ce dernier venait d'élaborer [198, 199]. Il possédait une sécurité prouvée[19] contre une large classe d'attaques statistiques et se prêtait à des implantations d'une rapidité exceptionnelle sur un processeur 64 bits. Son adaptatation légèrement moindre à des implantations sur un processeur 32 bits et la simplicité quelque peu provocatrice de sa composante non linéaire lui coûtèrent probablement la place de finaliste que de par ses autres caractéristiques il aurait méritée.

Vers la fin des années 90, la plupart des cryptologues avaient probablement le sentiment que la cryptanalyse différentielle et la cryptanalyse linéaire n'étaient que la partie émergée d'un ensemble d'attaques statistiques beaucoup plus vaste qui restait à découvrir, et s'attendaient à ce que l'évalution des candidats AES en fournisse l'occasion. Cette attente fut en partie déçue. Les perfectionnements qui ont été apportés jusqu'à aujourd'hui à la cryptanalyse différentielle et à la cryptanalyse linéaire n'ont pas entièrement bouleversé le paysage de la cryptanalyse statistique. Pourtant, certains d'entre eux sont loin d'être négligeables et ont conduit à des gains de performance substantiels. Les deux familles suivantes d'améliorations ou d'extensions des méthodes de cryptanalyse statistique mises au point durant cette période me semblent pouvoir être distinguées :

1°) La découverte de méthodes différentielles permettant dans certains cas d'augmenter le nombre de tours attaqués d'un facteur presque égal à deux grâce à une utilisation combinée de propriétés différentielles des premiers et des derniers tours. Ces méthodes sont la **cryptanalyse par différentielles impossibles** proposée par A. Biryukov, E. Biham et A. Shamir [25, 26], **l'attaque par boomerang** [200] proposée par D. Wagner et leurs extensions : « amplified boomerang attack » [138], « rectangle attack » [27, 28].

2°) La généralisation des attaques différentielles et linéaires, qui exploitent des statistiques binomiales, à des attaques exploitant des **statistiques multinomiales**. L'idée d'une telle généralisation, est à vrai dire à peu près contemporaine de la cryptanalyse différentielle et linéaire.[20] On la voit apparaître dans [168] et dans la cryptanalyse différentielle de l'algorithme IPES, ancêtre d'IDEA, découverte par S. Murphy [150]. On la retrouve dans la formalisation proposée par C. Harpes et J. Massey sous le nom de **cryptanalyse par partition** [119] et dans l'analyse très détaillée (reposant sur des partitions des valeurs intermédiaires, mais également des clés) des statistiques induites par les paires ou triplets de boîtes S du DES proposée par D. Davies et S. Murphy dans [79], ou encore (sous une forme légèrement différente) dans la **cryptanalyse Chi-2** proposée par S. Vaudenay et appliquée par ce dernier au DES [197].

[19] La preuve de sécurité concerne une version idéalisée de l'algorithme paramétrée par des sous-clés indépendantes.

[20] Cette idée consiste à partitionner des ensembles de clairs, de chiffrés, ou de valeurs intermédiaires bien choisis non plus en deux classes, mais en strictement plus de deux classes, et à calculer ou mesurer les probabilités d'occurrence des classes d'une partition ou la matrice des probabilités de transition entre les classes de deux partitions.

La technique de filtrage de paires de chiffrés utilisée dans notre attaque de KHUFU de [106] présente une certaine parenté avec la première famille d'attaques puisqu'elle repose sur une utilisation combinée des propriétés différentielles des premiers et des derniers tours. Pour ce qui concerne les premiers tours, cette attaque exploite une propriété différentielle rencontrée dans un nombre restreint d'algorithmes par blocs : la croissance relativement lente avec le nombre r de tours du nombre de valeurs de différence distinctes susceptibles de se présenter à la sortie de r tours.

Deux des trois cryptanalyses de versions réduites de candidats AES que j'ai publiées en 2000, à savoir la cryptanalyse de CRYPTON [164] étudiée en collaboration avec M. Minier et celle RC6 étudiée en collaboration avec H. Handschuh, A. Joux et S. Vaudenay [108] relèvent de la seconde famille d'attaques puisqu'elles exploitent l'une et l'autre des statistiques multinomiales.

Cryptanalyse de RC6 et de CRYPTON

Notre cryptanalyse de RC6 de [108] présente une certaine parenté avec l'attaque de Davies contre le DES : elle exploite la structure de schéma de Feistel de RC6, et le fait que la fonction f soit non bijective et présente des biais locaux relativement forts. Ces biais sont dus à une interaction entre les rotations dépendant des données et les « ou exclusif » entre données réalisés par la fonction f. Une attaque similaire fut découverte indépendamment par L. Knudsen et W. Meier, et publiée lors de la même conférence [145]. Les deux attaques exploitent le même type de propriétés, et ne diffèrent que par le détail des statistiques utilisées et la méthode de dérivation de clé.
Depuis le début des années 2000, les progrès de la cryptanalyse statistique des algorithmes par blocs se sont ralentis. Le nombre des publications consacrées à ce sujet, qui représentait il n'y a pas si longtemps environ la moitié des articles de la principale conférence consacrée à la cryptographie symétrique, Fast Software Encryption, a d'ailleurs très fortement diminué. Les dernières années ont été moins marquées par le renouvellement des méthodes de la cryptanalyse statistique que par leur diffusion à d'autres types d'algorithmes que les algorithmes par blocs, notamment les fonctions de hachage et les algorithmes à flot, contre lesquels elles se sont révélées d'une efficacité spectaculaire.

Quant à notre cryptanalyse de CRYPTON de [164], elle repose sur une partition des blocs de données et des blocs de différence rencontrés à chaque tour du chiffrement en respectivement 16 et 255 classes. Chaque tour est représenté par une matrice stochastique 16×16 (resp 255×255) respectivement dépendante et indépendante de la clé, dont les coefficients sont les probabilités de transition d'une classe d'entrée vers une classe de sortie. Une bonne approximation du comportement de la composition de plusieurs tours est fournie par le produit des matrices de transition associées. Nous avons proposé la notion de **cryptanalyse stochastique** pour qualifier cette méthode et montré en quoi elle généralise les attaques basées sur des caractéristiques, qui reposent sur une partition en seulement deux classes des blocs de données (cryptanalyse linéaire) ou des

blocs de différence (cryptanalyse différentielle).[21]

4 Cryptanalyse structurelle

La cryptanalyse statistique des algorithmes par blocs peut être vue comme une généralisation de techniques non statistiques très anciennes telles que les attaques dans le milieu, traduction approximative de l'anglais « meet in the middle ». En considérant non plus des équations exactes reliant les données d'entrée et les données intermédiaires calculées au cours du chiffrement, mais des équations approchées vérifiées avec une probabilité anormalement forte ou faible[22], on obtient des relations dépendant d'un plus petit nombre de bits de clé, voire indépendantes de la clé[23], ce qui permet de diminuer la complexité des attaques. Mais la contrepartie de cette moindre dépendance en la clé est qu'un très grand nombre de couples (clair, chiffré) est en général requis pour que les biais statistiques des relations approchées ainsi construites soient détectables. Cela vaut particulièrement pour les meilleurs algorithmes par blocs récents, dont l'un des principaux critères de conception est leur résistance à la cryptanalyse statistique. La cryptanalyse statistique ne fournit alors que des attaques « certificationnelles » irréalistes, portant sur un nombre très réduit de tours, et requérant pour une version non réduite de l'algorithme un nombre de couples (clair, chiffré) supérieur au nombre 2^n de blocs d'entrée distincts.

Paradoxalement, la protection de ces mêmes algorithmes contre des attaques non statistiques ne semble pas une question aussi systématiquement explorée. Un retour à des méthodes d'attaque de nature non statistique s'est de ce fait amorcé depuis quelques années, et il n'est pas déraisonnable de conjecturer que les prochaines avancées dans le domaine de la cryptanalyse des algorithmes par blocs pourraient concerner la cryptanalyse non statistique plutôt que la cryptanalyse statistique. C'est la raison pour laquelle j'ai accordé dans mes recherches récentes une attention particulière à ce type d'attaques.

Rappelons brièvement quelques-unes des principales méthodes non statistiques de cryptanalyse découvertes au cours des quinze dernières années. Ces attaques exploitent des propriétés structurelles ou algébriques de l'algorithme cible.

1°) **Les attaques différentielles d'ordre supérieur**, concept proposé par X. Lai [149]. Les blocs de clair, de chiffré partiel et de chiffré d'un algorithme par blocs peuvent généralement être représentés comme des n-uplets de mots appartenant au corps fini $\mathrm{GF}(q)$. Les attaques différentielles d'ordre supérieur reposent sur l'observation que si,

[21] Dans le cas de la cryptanalyse linéaire, les données sont partitionnées en deux classes selon la valeur (0 ou 1) d'une forme linéaire ; dans le cas de la cryptanalyse différentielle, les différences sont partitionnées en deux classes : la première est constituée d'une différence attendue Δ (ou dans le cas de différentielles tronquées d'un ensemble de différences attendues) et la seconde de toutes les autres différences.

[22] c'est-à-dire une probabilité distincte de celle que l'on rencontrerait si les tours de l'algorithme étaient remplacés par des permutations aléatoires

[23] Des relations indépendantes de la clé sont également appelées **distingueurs**.

pour toute clé et tout ou partie des entrées, le degré algébrique de l'expression en fonction des mots d'entrée de certains mots des chiffrés partiels sur r tours est strictement inférieur à un entier d, alors toute différentielle de degré d de cette expression est nulle. Toute somme des valeurs d'une fonction multivariable f sur un sous-espace affine de dimension d de l'ensemble des n-uplets d'entrée est une différentielle de degré d de f, et est donc nulle. Cette propriété fournit un grand nombre de relations exactes faisant intervenir q^d clairs et vérifiées pour toute clé. De telles relations sont exploitables lorsque d est suffisamment petit, et permettent d'attaquer un peu plus de r tours.

2°) **Les attaques par interpolation** proposées par T. Jakobsen et L. Knudsen [130]. Elles s'appliquent à des algorithmes dont les chiffrements et/ou déchiffrements partiels s'expriment à l'aide d'expressions polynomiales de bas degré ou à l'aide de fractions rationnelles dont le numérateur et le dénominateur sont de bas degré. La connaissance d'un nombre suffisant de couples (clair, chiffré) permet de reconstituer les coefficients inconnus de ces expressions.

3°) **Les attaques par saturation** ou « **square attacks** » : cette famille d'attaques, également désignée sous le nom d'« **integral cryptanalysis** » [146], a été découverte par J. Daemen, L. Knudsen et V. Rijmen, qui l'ont initialement appliquée à leur algorithme SQUARE [74] puis transposée à une version réduite à 6 tours de RIJNDAEL [75]. Elle s'est avérée applicable à de nombreux autres algorithmes dont les opérations respectent une structuration des blocs en octets ou en mots de w bits. Il s'agit d'attaques à clair choisi utilisant des ensembles de $m = 2^w$ clairs obtenus en fixant l'ensemble des mots d'entrée sauf un d'entre eux, et en faisant parcourir à ce dernier l'ensemble des 2^w valeurs possibles. Comme elles reposent sur des propriétés d'équilibre des chiffrés partiels qui peuvent être exprimées à l'aide de différentielles d'ordre w, elles peuvent être considérées comme un cas particulier d'attaques différentielles d'ordre supérieur. Une amélioration de la « square attack » due à N. Ferguson *et al.* [95], et dans laquelle on considère non plus des ensembles de 2^8 clairs, mais des ensembles de 2^{32} clairs constitue la meilleure attaque connue contre une version réduite à 6 tours d'AES-128. Cette attaque met en œuvre une technique de comptage efficace, pour les différentes hypothèses de clé, du nombre d'occurrences d'octets auxiliaires intervenant dans l'expression de la propriété d'équilibre testée, connue sous le nom de **partial-sum technique**.

4°) **L'attaque structurelle de SASAS**[24] découverte par A. Biryukov et A. Shamir [41] possède de nombreuses similitudes avec les attaques par saturation. Elle utilise des multiplets de clairs choisis, et étudie la propagation d'un tour au suivant des propriétés des multiplets de chiffrés partiels qu'ils induisent. Ces propriétés sont en grande partie les mêmes que celles utilisées dans les attaques par saturation mais ne s'y réduisent cependant pas toutes. Elles concernent non seulement la constance et l'équilibre, mais aussi

[24] On désigne par SASAS tout algorithme par blocs à cinq tours constitué d'une alternance d'appels à des boîtes S secrètes (tours S) et à des bijections affines secrètes (tours A). L'attaque [41] est structurelle en ce sens qu'elle n'exploite pas les spécificités des composants de l'algorithme (boîtes S et bijections affines), mais seulement la structure générale de ce dernier.

la parité du nombre d'occurrences des valeurs des octets ou des mots de w bits des blocs de clair ou de chiffré partiel.

5°) **La cryptanalyse algébrique** : l'appellation de cryptanalyse algébrique s'est progressivement imposée, dans les domaines de la cryptographie symétrique et de la cryptographie asymétrique, pour désigner des attaques fondées sur la résolution d'un système d'équations algébriques multivariées de bas degré induites par la connaissance d'entrées et de sorties de l'algorithme. Le système considéré comporte généralement des variables de clé et/ou des variables représentant des données intermédiaires. Diverses méthodes peuvent être mises en œuvre pour résoudre ces systèmes d'équations : linéarisation, relinéarisation [142], XL [68, 70], algorithmes de bases de Gröbner (algorithme de Buchberger, algorithmes F4 et F5 dûs à J.-C. Faugère [89, 90]). Dans le cas de systèmes aléatoires d'équations de degré $d > 1$ fixé, si l'on exclut le cas de systèmes fortement sur-déterminés ou sous-déterminés, les méthodes connues sont inopérantes en pratique au-delà d'un nombre restreint de variables. En particulier, lorsque le nombre d'équations et d'inconnues sont égaux, la complexité des meilleures méthodes connues est exponentielle et celles-ci deviennent inopérantes au-delà de quelques dizaines de variables y compris dans le cas de systèmes d'équations quadratiques sur GF(2). De ce fait, la cryptanalyse algébrique ne s'est à ce jour révélée applicable qu'à un ensemble réduit de cryptosystèmes, qui induisent des équations algébriques possédant des caractéristiques très différentes d'un système aléatoire de même degré. Ses succès les plus spectaculaires concernent deux principales classes d'algorithmes : d'une part la cryptanalyse d'algorithmes asymétriques comme le schéma de Matsumoto et Imai [176] et certaines instances du système HFE [142, 65, 92], et d'autre part des algorithmes symétriques de chiffrement à flot fondés sur le filtrage non linéaire de suites récurrentes linéaires [69, 91, 66, 6]. Le cas des algorithmes par blocs reste une question ouverte[25] : l'existence d'attaques algébriques plus efficaces que la recherche exhaustive contre un algorithme par blocs tel qu'AES-256 n'a pu être à ce jour ni confirmée, ni entièrement infirmée.

[25]Pour des algorithmes par blocs définis à l'aide d'opérations sur un corps fini tels qu'AES, MISTY et KASUMI ou faisant appel à des boîtes S de taille très réduite comme SERPENT ou KHAZAD, le chiffrement d'un ou plusieurs clairs peut être décrit à l'aide d'un système sur-déterminé relativement creux et structuré de quelques milliers d'équations de très bas degré sur un corps fini comme GF(2) ou GF(256) [71, 169, 40]. N. Courtois et J. Pieprzyk ont proposé, sous le nom de XSL [71, 72], plusieurs variantes d'une stratégie de résolution de tels systèmes fondée sur la préservation de la structure des équations induites par les boîtes S. Cependant, les premières estimations heuristiques du degré auquel il est nécessaire d'élever les équations de ces systèmes pour les linéariser, qui conduisaient dans le cas d'AES-256 à une complexité estimée légèrement inférieure à celle de la recherche exhaustive, étaient semble-t-il trop faibles. De plus, pour l'une des variantes de XSL [72], la mise en équations ne conduit pas à un système déterminé [62].

4.1 Cryptanalyse de versions réduites de l'AES

J'ai mis au point en 2000, en collaboration avec M. Minier [109], une attaque à clairs choisis de versions réduites de l'AES pour laquelle nous avons proposé la dénomination de « **bottleneck attack**[26] ». Cette attaque peut être qualifiée de cryptanalyse structurelle. De même que les attaques par saturation et la cryptanalyse structurelle de SASAS, elle exploite en effet des propriétés de la fonction de chiffrement indépendantes du choix de composantes de l'algorithme telles que les boîtes S. Mais le distingueur entre quatre tours d'AES et une permutation aléatoire qui sert de point de départ à la « bottleneck attack » diffère substantiellement des distingueurs fondés sur des propriétés d'équilibre dont découlent ces autres attaques.

Notre attaque [109] repose sur l'étude des fonctions partielles d'un octet vers un octet induites par des versions réduites de l'algorithme de chiffrement. Nous analysons la propagation de l'influence d'un octet d'entrée y à travers r tours internes de l'AES lorsque l'on fixe les quinze autres octets du bloc d'entrée. Douze de ces quinze octets sont des constantes arbitraires, et le triplet $c = (c_0, c_1, c_2)$ constitué des trois octets provenant de la même colonne du bloc d'entrée que y est considéré comme un paramètre des fonctions partielles étudiées. La propagation de l'influence de l'octet d'entrée y peut être représentée par un arbre dont la racine est l'octet y et les sommets au niveau r sont les octets des blocs de chiffré partiel influencés par y à l'issue de r tours. Chaque octet au niveau r est relié à quatre octets du niveau $r+1$ par la succession d'opérations suivante : une boîte S suivie d'une multiplication par une constante α égale à l'un des coefficients de MixColumns, elle-même suivie d'un ou exclusif avec un octet β. La valeur de β est une fonction affine des images par la boîte S des octets du niveau r et d'un octet de la sous-clé du tour $r+1$. Du fait des propriétés de diffusion de l'AES, il n'y a de repliements dans l'arbre de propagation ainsi défini qu'à partir du troisième tour, et chaque octet de la sortie du troisième tour est une fonction $f^c(y)$ de l'octet d'entrée y entièrement déterminée par seulement neuf octets inconnus : cinq octets dépendant uniquement de la clé et un quadruplet d'octets $\beta^c = (\beta_0^c, \ldots, \beta_3^c)$ dépendant de la clé et du paramètre c. L'hypothèse heuristique que le comportement du quadruplet d'octets β^c ne diffère pas notablement de celui d'une valeur de 32 bits tirée au sort, et donc que le paradoxe des anniversaires peut lui être appliqué, suggère que si l'on considère un peu plus de 2^{16} valeurs de c, une collision sur la valeur du quadruplet β^c se produit avec une forte probabilité pour au moins deux de ces valeurs, c' et c''. Les fonctions partielles $f^{c'}(y)$ et $f^{c''}(y)$ sont donc égales pour chacune des 256 valeurs de l'octet d'entrée y. Cette propriété de **collision entre fonctions partielles**, confirmée expérimentalement, constitue un distingueur efficace entre trois tours de l'AES et une permutation aléatoire. Elle peut être facilement convertie en un distingueur de même efficacité entre quatre tours de l'AES et une permutation aléatoire en remarquant que le déchiffrement d'un quatrième tour permet de relier tout octet de sortie du troisième tour à une combinaison linéaire t à coefficients connus de quatre des octets de sortie du quatrième tour. Si l'on note $t^c(y)$ la valeur de la combinaison linéaire t correspondant à

[26] Cette attaque est également parfois désignée sous le nom de « collision attack », utilisé dans notre article [109], ou celui de « Gilbert-Minier attack », utilisé par les auteurs de l'AES [76].

la valeur c du paramètre et à l'octet d'entrée y, il existe donc avec une grande probabilité deux triplets d'octets c' et c'' pour lesquels il se produit une collision entre les 256 octets $t^{c'}(y)_{y=0...255}$ et les 256 octets $t^{c''}(y)_{y=0...255}$.

Le distingueur qui précède peut être converti en une attaque contre 7 tours d'AES en ajoutant aux quatre tours sur lesquels il porte un tour initial et deux tours finaux. Notre attaque, dont on trouvera dans [77] une description complétant celle de [109], requiert 2^{32} clairs choisis ; sa complexité est nettement inférieure à celle de la recherche exhaustive dans le cas d'AES-192 et AES-256, mais n'est que très légèrement inférieure à celle de la recherche exhaustive dans le cas d'AES-128. Il s'agit, si l'on excepte les attaques à clés reliées [21, 129, 141, 125, 29, 30], de l'attaque de moindre complexité actuellement connue contre sept tours de l'AES (cf table 2).

Dans un article publié en 2008 [126], H. Demirci et A. Selçuk ont montré que la propriété sur trois tours exploitée dans notre attaque peut être prolongée à quatre tours : l'expression de chaque octet de sortie du quatrième tour fait intervenir 5 octets dépendant de la clé seule et 20 octets dépendant de la clé et du paramètre c (au lieu de 4 dans le cas de trois tour). En raison de la complexité du distingueur sur 5 tours qui en résulte, cette propriété ne peut pas être exploitée pour améliorer les performance de notre attaque sur 7 tours. Mais elle entraîne l'existence d'une attaque plus rapide que la recherche exhaustive contre 8 tours d'AES-256 (cf table 2).

attaques	**taille de clé**	**tours**	**clairs**	**complexité**	**mémoire**
square [74]	128	5	2^{11}	2^{40}	2^{11}
boomerang [39]	128	5	2^{39}	2^{39}	2^{33}
impossible [31]	128	5	$2^{29.5}$	2^{31} hors précalcul	2^{40}
square [74]	128	6	2^{32}	2^{72}	2^{32}
partial sum [95]	128	6	2^{35}	2^{44}	2^{32}
boomerang [39]	128	6	2^{71}	2^{71}	2^{33}
partial sum [95]	192-256	7	2^{35}	2^{172}	2^{32}
bottleneck [109]	128	7	2^{32}	$2^{128}(1-\varepsilon)$	2^{80}
bottleneck [109]	192-256	7	2^{32}	2^{144}	2^{80}
partial sum [95]	192	8	$2^{128}(1-\varepsilon)$	2^{188}	2^{128}
partial sum [95]	256	8	$2^{128}(1-\varepsilon)$	2^{204}	2^{128}
MitM [126]	256	8	2^{40}	2^{208} hors précalcul	2^{216}
MitM-TM [126]	256	8	2^{72}	2^{210+n} hors précalcul	2^{216-n}

Tab. 2 – Attaques contre des versions réduites de l'AES. Aucune attaque à clés reliées n'est représentée sur cette table.

5 Conception d'algorithmes et preuves de sécurité

Bien que j'aie constamment recherché, dans mes travaux sur les algorithmes par blocs, un équilibre entre les activités complémentaires que constituent à mes yeux la cryptanalyse et la conception d'algorithmes, mes publications dans ce domaine ne

reflètent pas cet équilibre. Elles portent exclusivement sur des cryptanalyses ou sur des preuves partielles de sécurité de constructions préexistantes.

5.1 Conception d'algorithmes par blocs

Cette apparente contradiction tient à ce que mes activités de concepteur d'algorithmes se sont exercées essentiellement de deux manières :

1°) A travers la spécification d'algorithmes privés, dont les plus largement déployés sont d'une part des algorithmes d'authentification et d'établissement de clé implantés dans les puces SIM ou USIM d'un grand nombre d'opérateurs du groupe Orange, et d'autre part des codes d'authentification de messages à très bas coût appelés fonctions anticlones[27], conçus en coopération avec mes collègues D. Arditti, M. Girault et J.C. Paillès ;

2°) A travers ma participation depuis sa création, il y a une quinzaine d'années, au groupe d'experts ETSI/SAGE.[28] Ce groupe a conçu une dizaine d'algorithmes symétriques par blocs ou à flot, dont seulement une partie a été publiée. J'ai contribué selon le cas à la conception ou à l'analyse mathématique de ces algorithmes.

Le fait d'être le concepteur ou l'un des concepteurs d'algorithmes non publiés ne m'interdisait pas de publier par ailleurs des propositions de nouveaux algorithmes par blocs. Peut-être la conviction, tirée de mon travail de cryptanalyste, qu'il est très difficile de proposer un algorithme qui soit d'emblée indemne de toute imperfection au regard des critères académiques m'a-t-elle incité à une prudence excessive, et je n'exclus pas de le faire un jour.

La conception d'algorithmes par blocs est demeurée un domaine très empirique, et il existe une étonnante analogie de structure entre les principaux algorithmes de chiffrement par blocs actuels, tels que ceux énumérés dans la table 1 ci-dessus. Cette consanguinité s'explique en grande partie par l'influence très forte qu'a eue le DES sur tous les algorithmes ultérieurs.

Il m'est évidemment impossible de décrire ici en détail les algorithmes non publiés que j'ai conçus ou à la conception desquels j'ai participé. Ils ne diffèrent pas radicalement des algorithmes précédemment cités, mais certains d'entre eux s'en démarquent quelque peu par l'emploi d'autres ingrédients élémentaires de confusion et de diffusion. J'ai ainsi par exemple conçu, en coopération avec mon collègue G. Macario-Rat, des algorithmes dont le bloc courant est un registre $(x_{t-m+1}, \ldots, x_t)$ constitué de m mots de w bits ou plus généralement de m mots d'un alphabet fini Σ, dont le principal ingrédient de confusion est un carré latin sur l'alphabet Σ, et dans lesquels la diffusion est assurée en itérant une relation de récurrence linéaire ou non linéaire très simple, de la forme

[27] Les fonctions anticlones ont été implantées à ce jour dans plus d'un milliard de puces à logique cablée.

[28] European Telecommunications Standards Institute, Security Algorithms Group of Experts, chargé de la conception d'algorithmes de sécurité pour le secteur des télécommunications en Europe.

$x_{t+1} = f(x_{t-m+1}, \ldots, x_t)$ ou de la forme $x_{t+1} = f_{k_t}(x_{t-m+1}, \ldots, x_t)$, où k_t représente un mot de clé étendue. De telles récurrences, bien qu'elles puissent être vues comme un cas particulier de schéma de Feistel généralisé, sont actuellement beaucoup plus usitées dans le domaine des algorithmes à flot que celui des algorithmes par blocs. Elles présentent des avantages, en matière de performances et de coût d'implémentation, sur les constructions de matrices linéaires utilisées comme ingrédient de diffusion dans de nombreux algorithmes par blocs.

Lorsque l'on cherche à concevoir un algorithme réalisant un compromis optimal entre efficacité et solidité conjecturée, l'on est très naturellement amené à définir des **critères** de conception des composants élémentaires de l'algorithme et à rechercher des compromis optimaux entre ces critères. Je n'en considère pas moins cet aspect du travail de concepteur avec une certaine circonspection. La dialectique qui s'est instaurée entre la découverte de nouvelles classes d'attaques et la mise au point de critères liés à la résistance à ces attaques a sans doute contribué à améliorer la solidité et l'élégance des algorithmes par blocs récemment proposés. Mais elle ne me semble pas de nature à faire à elle seule sortir les algorithmes par blocs de l'empirisme qui préside à leur conception. En effet, les critères que l'on est amené à introduire ne me semblent pas donner une assurance globale de sécurité très forte. Dans le meilleur des cas, l'on parvient à définir des critères directement liés à la résistance des algorithmes à des classes spécifiques d'attaques : tel est par exemple le cas des critères de résistance à la cryptanalyse linéaire et différentielle. Cependant, d'autres critères, comme ceux fondés sur des notions comme l'immunité algébrique des boîtes S, n'ont à l'heure actuelle (dans le cas des algorithmes par blocs) qu'un lien assez indirect avec la résistance à une classe d'attaques. Tout en reconnaissant que les deux points de vue ne sont pas antinomiques, j'accorde de ce fait plus d'importance à l'analyse structurelle des algorithmes par blocs qu'à celle des propriétés locales de leurs composants élémentaires, et la voie de recherche consistant à essayer d'étendre aux algorithmes par blocs l'approche réductionniste de la sécurité prouvée presque systématiquement adoptée en cryptographie asymétrique me semble à terme plus prometteuse.

5.2 Preuves de sécurité dans le modèle de Luby et Rackoff

Comme cela a été rappelé ci-dessus, un algorithme par blocs est un générateur de permutations qui associe à une clé $K \in \{0,1\}^k$ une permutation E_K de l'ensemble P_n des permutation de l'ensemble $\{0,1\}^n$. Il est considéré comme sûr si (E_K) est indistinguable d'une permutation aléatoire parfaite tirée au sort selon la loi uniforme P_n au moyen d'un algorithme probabiliste de test de complexité inférieure à un seuil t (par exemple 2^{80} ou 2^{128} opérations) et réalisant au plus q appels à entrée choisie (par exemple 2^{80} appels) à E_K et la permutation inverse D_K.

Les conditions de secret parfait énoncées par Shannon dans son article de 1949 [189] interdisent que, dans le cas toujours rencontré en pratique où $qn > k$, un algorithme par blocs offre une **sécurité inconditionnelle**, i.e. reste sûr face à un adversaire disposant de capacités de calcul illimitées. Par contre, si l'on se satisfait d'une **sécurité algorithmique**, i.e. si l'on ne considère que les tests de complexité inférieure ou égale à

un seuil t fini, on peut raisonnablement conjecturer que des algorithmes par blocs sûrs au sens précédent existent. Mais la construction d'un algorithme par blocs pour lequel on dispose d'une preuve absolue de sécurité dans le second modèle (pour des valeurs de q supérieures à celles assurant une sécurité inconditionnelle) est hors de portée des connaissances actuelles.

Le modèle de sécurité proposé par M. Luby et C. Rackoff dans l'article fondateur [154] contourne ces limitations fondamentales en cherchant à valider non pas la sécurité d'algorithmes par blocs considérés dans leur totalité, mais seulement la partie externe Φ de leur construction, qui à de « petites » fonctions ou permutations $f_1, \ldots, f_r$ dépendant de la clé, associe la « grande » permutation $E_K = \Phi(f_1, \ldots, f_r)$ que constitue la fonction de chiffrement.[29] Dans ce modèle, on cherche à démontrer que si dans la génération de la permutation E_K à partir de la clé les fonctions ou permutations réelles $f_1, \ldots, f_r$ utilisées dans la construction sont remplacées par des fonctions ou permutations parfaitement aléatoires $f_1^*, \ldots, f_r^*$, indépendantes et tirées au sort selon la loi uniforme dans l'ensemble des fonctions ou permutation de mêmes tailles d'entrée et de sortie que les f_i, alors la fonction de chiffrement $F = \Phi(f_1^*, f_2^*, \ldots, f_r^*)$ qui en résulte ne peut être distinguée avec un avantage non négligeable d'une permutation parfaitement aléatoire à l'aide d'un algorithme d'attaque utilisant un nombre q d'entrées ou de sorties choisies de façon adaptative, et ce quelle que soit la complexité des calculs effectués par l'attaquant. En d'autres termes, on cherche à démontrer que F est **super-pseudoaléatoire** : $\mathbf{Adv}^{SPRP}_{\Phi(f_1^*,\ldots,f_r^*)}(\infty, q) < \varepsilon$, où ε représente une probabilité négligeable. L'on se retreint parfois à des algorithmes d'attaque utilisant q entrées choisies de façon adaptative au lieu de q entrées ou sorties choisies de façon adaptative : l'on cherche alors seulement à démontrer que le générateur de permutations F est **pseudoaléatoire** : $\mathbf{Adv}^{PRP}_{\Phi(f_1^*,\ldots,f_r^*)}(\infty, q) < \varepsilon$. Le principal résultat de M. Luby et C. Rackoff [154] relatif au générateur de permutations $\Psi(f_1^*, f_2^*, f_3^*)$ de P_{2n} associé au schéma de Feistel à trois étages peut être résumé par l'inégalité $\mathbf{Adv}^{PRP}_{\Psi(f_1^*,f_2^*,f_3^*)}(\infty, q) < \frac{q^2}{2^n}$.

Les preuves de sécurité dans le modèle de Luby et Rackoff se sont révélées un domaine de recherche fécond où de nouvelles avancées continuent à être faites. Pour ne citer que quelques uns des travaux fondateurs de cette ligne de recherche, on doit à J. Patarin et à U. Maurer d'avoir les premiers perçu que ces preuves relèvent entièrement de la sécurité inconditionnelle et non de la sécurité algorithmique. Elles peuvent être

[29] Un algorithme de chiffrement par blocs est généralement défini à l'aide d'une construction présentant plusieurs niveaux d'imbrication. On rencontre en particulier très fréquemment la situation suivante : à la clé K sont associées des fonctions ou des permutations f_1 à f_r d'un ensemble de $\{0,1\}^m$ où $m \leq n$ et où r représente par exemple le nombre de tours de l'algorithme, et la fonction de chiffrement E_K est à son tour déduite des fonctions f_1 à f_r à l'aide d'une transformation Φ. Le dernier niveau de la construction est ainsi représenté par la relation $E_K = \Phi(f_1, f_2, \ldots, f_r)$. Par exemple, dans le cas du DES, si l'on omet la permutation initiale et la permutation finale qui ne contribuent pas à la sécurité de l'algorithme, les sous-clés des $r = 16$ tours permettent d'associer à la clé K de 56 bits des fonctions f_i, $i = 1$ à r de $\{0,1\}^{32}$ dans lui-même et la fonction de chiffrement E_K se déduit des f_i à l'aide de la transformation Feistel : si l'on désigne par Ψ la transformation de Feistel, on a donc $E_K = \Psi(f_1, f_2, \ldots, f_r)$.

facilement étendues à des preuves relevant seulement de la sécurité algorithmique en montrant que si au lieu d'être parfaitement aléatoires, les fonctions ou permutations f_1 à f_r utilisées en entrée de Φ sont informatiquement indistinguables de fonctions pseudoaléatoires f_1^* à f_r^* indépendantes, alors $F = \Phi(f_1, \ldots, f_r)$ est également informatiquement indistinguable d'une permutation parfaitement aléatoire en un temps de calcul et avec un nombre de questions et une borne de probabilité dépendant simplement de ceux des f_i. J. Patarin a établi un lien très simple entre l'avantage du meilleur distingueur d'une attaque contre un générateur de fonctions ou de permutations F et les probabilités de transition entre q-uplets d'entrées et de sorties de F qui permet de simplifier considérablement les preuves dans ce modèle en les réduisant à des considérations purement combinatoires.[30] Il a intitulé cette approche fondée sur l'évaluation de probabilités de transition entre q-uplets **méthode des coefficients H**, et l'a appliquée au calcul de bornes d'indinstinguabilité très précises des schémas de Feistel à r tours [175, 177]. S. Vaudenay a introduit sous le nom de **théorie de la décorrélation** une extension de la méthode des coefficients H : l'ensemble des matrices de probabilités de transition entre q-uplets d'entrée et de sortie des fonctions aléatoires de n bits vers m bits est muni de normes, appelées normes de décorrélation ; ces dernières permettent d'exprimer simplement l'avantage du meilleur distingueur pour différents types d'adversaires, et d'automatiser le calcul de bornes sur l'avantage d'un adversaire dans les cas de fonctions aléatoires composées représentant par exemple les tours d'un algorithme par blocs. M. Bellare, P. Rogaway et leurs coauteurs ont démontré, dans le modèle de sécurité de Luby et Rackoff et des modèles apparentés, la sécurité de nombreuses autres constructions que le schéma de Feistel, comme le mode CBC-MAC, qui permet de convertir un algorithme par blocs en code d'authentification de messages [15] et divers modes chiffrants d'un algorithme par blocs [14].

Mes contributions à la validation de constructions cryptographiques dans le modèle de sécurité de Luby et Rackoff sont au nombre de deux. Ces contributions sont liées à ma participation aux travaux de spécification d'algorithmes de sécurité conçus pour le système UMTS : leur principal objectif était de développer des arguments de sécurité pour ces algorithmes ou des modes opératoires de ces algorithmes. La méthodologie générale de preuve employée est la méthode des coefficients H due à J. Patarin. Bien que les preuves de ce type reposent sur l'enchaînement d'arguments de dénombrement en eux-mêmes très simples, la recherche d'une partition des transitions $\mathbf{x} \stackrel{f}{\longmapsto} \mathbf{y}$ entre q-uplets d'entrée et de sortie induites par le générateur de fonctions étudié qui soit exploitable pour aboutir aux bornes souhaitées est parfois délicate.

La première de ces contributions, qui résulte d'une étude réalisée en coopération

[30] Dans le cas où on souhaite prouver l'indistinguabilité d'un générateur de fonctions F de n bits vers m bits d'une fonction parfaitement aléatoire F^* de même taille d'entrée et de sortie, on peut s'appuyer sur le théorème suivant, dû à J. Patarin [175] : *si pour tous les q-uplets $\boldsymbol{x} = (x_1, \ldots, x_q)$ d'entrées distinctes et pour une fraction des q-uplets de sortie $\boldsymbol{y} = (y_1, \ldots, y_q)$ supérieure ou égale à $1 - \varepsilon_1$, la probabilité de transition $Pr(\boldsymbol{x} \stackrel{f}{\longmapsto} \boldsymbol{y})$ associée à une instance aléatoire de F est supérieure ou égale à $(1 - \varepsilon_2)p^*$, où p^* désigne la probabilité de transition $Pr(\boldsymbol{x} \stackrel{f^*}{\longmapsto} \boldsymbol{y}) = \frac{1}{2^{mq}}$ à laquelle conduirait une fonction parfaitement aléatoire f^*, alors l'avantage pour distinguer F d'une fonction parfaitement aléatoire à l'aide de q questions est majoré par $\varepsilon_1 + \varepsilon_2$:* $\mathbf{Adv}_F^{PRF}(\infty, q) \leq \varepsilon_1 + \varepsilon_2$.

avec M. Minier, a été publiée en 2001 [110]. Elle porte sur une alternative au schéma de Feistel appelée schéma de MISTY, du nom de l'algorithme par blocs proposé par M. Matsui où elle apparaît pour la première fois. Cette construction est utilisée dans l'algorithme par blocs KASUMI[31], sur lequel reposent le chiffrement et l'authentification de messages du système UMTS. Le schéma de MISTY à r tours permet de construire, à partir de r permutations aléatoires de $\{0,1\}^n$, notées c_1 à c_r, une permutation aléatoire $F = \Psi(c_1, \ldots, c_r)$ de $\{0,1\}^{2n}$. Il avait été établi par K. Sakurai et Y. Zheng [185] qu'en appliquant le schéma de MISTY à quatre étages à quatre permutations aléatoires parfaites indépendantes, on obtient un générateur de permutations $F = \Psi(c_1^*, \ldots, c_4^*)$ qui est pseudoaléatoire, mais n'est pas super-pseudoaléatoire. Notre principal résultat est une preuve que si l'on porte le nombre d'étages et le nombre de permutations aléatoires parfaites indépendantes à cinq, on obtient un générateur de permutations $F = \Psi(c_1^*, \ldots, c_5^*)$ qui est super-pseudoaléatoire. Plus précisément, nous établissons que l'avantage pour distinguer F d'une permutation parfaitement aléatoire à l'aide d'un algorithme utilisant q chiffrements ou déchiffrements est borné par l'inégalité :

$$\mathbf{Adv}^{SPRP}_{\Psi(c_1^*,\ldots,c_5^*)}(\infty, q) \leq \frac{9}{2}\frac{q^2}{2^n}.$$

Un résultat similaire a été établi indépendamment par T. Iwata *et al.* [128].

La seconde de ces contributions [103], publiée en 2003, porte sur les modes opératoires d'un algorithme par blocs permettant de construire une fonction expansive (i.e. une fonction dont la taille de sortie est strictement supérieure à la taille d'entrée) de un bloc vers $t > 1$ blocs. De tels modes opératoires peuvent être considérés comme une sorte de dual de modes opératoires contractants, de t blocs vers un bloc, tels que le mode opératoire CBC-MAC. Deux des algorithmes de sécurité de l'UMTS utilisent des fonctions expansives déduites d'un algorithme par blocs : d'une part l'exemple d'algorithme d'authentification et d'établissement de clé MILENAGE [87], fondé sur un enchaînement de six appels à l'AES, et d'autre part l'algorithme de chiffrement UEA1 de l'UMTS, fondé sur l'utilisation de l'algorithme par blocs KASUMI selon un mode opératoire intitulé $f8$ [86]. Les constructions étudiées ont en commun que l'algorithme par blocs sous-jacent est appliqué une première fois au bloc d'entrée pour obtenir une valeur intermédiaire. J'ai démontré dans [103] que si l'algorithme par blocs sous-jacent est une PRP (ou une PRF), alors le générateur de fonctions F résultant est une PRF.

[31] de l'adjectif japonais « kasumi », équivalent approximatif de « misty » en anglais, et qui n'a pas semble-t-il la signification quelque peu péjorative de « brumeux » en français, mais plutôt celle plus poétique de « nimbé de brume ».

6 Nouveaux paradigmes

Les algorithmes par blocs, même employés à travers des modes opératoires non chiffrants, ne suffisent pas à répondre à des besoins comme la protection des puces électroniques contre la recopie, la protection des contenus numériques, ou la protection logicielle. Les nouveaux besoins qui sont apparus dans ces domaines au cours des dernières années ont suscité l'émeergence de nouvelles primitives plus paradoxales et souvent plus malaisées à construire que les algorithmes par blocs, mais qui conservent une certaine parenté avec ces derniers. Le présent chapitre présente succinctement mes contributions à la conception ou à l'analyse de telles primitives.

6.1 Cryptographie en boîte blanche

La notion de cryptographie en boîte blanche a été proposée par S. Chow, P.A. Eisen, H. Johnson et P.C. van Oorschot dans deux articles fondateurs parus en 2002 [59, 60]. Il s'agit d'une technique visant à protéger les implantations logicielles d'algorithmes cryptographiques contre l'extraction de leur clé secrète par un programme contrôlé par un adversaire et exécuté sur le même processeur. Le constat sur lequel se fondent les auteurs de [59, 60] est double : (1) les algorithmes cryptographiques actuels ne sont conçus que pour être sûrs dans un environnement d'exécution **en boîte noire**, i.e. face à un adversaire n'ayant accès qu'aux entrées non secrètes et aux sorties de ces algorithmes, mais ni aux clés secrètes dont ils dépendent, ni aux données intermédiaires générées au cours de leur exécution ; (2) cependant, ces algorithmes sont de plus en plus rarement exécutés dans un environnement dans lequel le modèle d'exécution en boîte noire est pertinent. Ils sont le plus souvent implantés dans un environnement logiciel dans lequel d'autres processus que le calcul cryptographique proprement dit et potentiellement hostiles s'exécutent. Ces autres processus ont éventuellement accès au code de l'algorithme et aux données statiques, ainsi qu'aux données générées dynamiquement au cours des calculs cryptographiques. Ils sont donc susceptibles de lire ou reconstituer les clés secrètes mises en œuvre et de les redistribuer à l'extérieur. Rien ou presque ne subsistant dans un tel environnement de l'opacité supposée dans le modèle d'exécution en boîte noire, S. Chow *et al.* proposent pour le qualifier la notion de modèle d'exécution **en boîte blanche**.

Les techniques de protection d'algorithmes cryptographiques implantés en boîte blanche proposées par S. Chow *et al.* consistent à décrire une instance d'un algorithme cryptographique tel qu'AES ou DES dépendant d'une clé K fixée au moyen d'une succession d'appels à des tables dans lesquelles la clé se trouve dissimulée. Bien qu'une telle description soit intermédiaire entre la spécification de l'algorithme et le code d'une implantation de l'enchaînement des appels à ces tables, elle est appelée improprement **implantation en boîte blanche**. Il doit être impossible à un adversaire possédant une telle description (ou disposant, ce qui revient au même, d'un accès en boîte blanche à une implantation logicielle déduite d'une telle description) de reconstituer la clé secrète de l'algorithme ou plus généralement une description courte de l'instance de l'algorithme. La principale technique mise en œuvre pour dissimuler la clé consiste à décomposer les

tours de l'algorithme en fonctions élémentaires, à composer à droite et à gauche ces fonctions élémentaires avec des transformations parasites appelées **codages externes** et **codages internes**, et à représenter les fonctions composées sous la forme de tables. Les codages externes s'appliquent avant le premier tour et après le dernier tour. Les codages internes s'appliquent en entrée et en sortie de chaque fonction élémentaire, et sont choisis de telle sorte que les codages internes appliqués en sortie d'une fonction élémentaire et en entrée de la fonction élémentaire qui la suit se compensent. De telles techniques ne préviennent pas la recopie du code d'une implantation en boîte blanche : la protection procurée par la cryptographie en boîte blanche est donc intrinsèquement limitée. Mais comme le code d'une implantation en boîte blanche d'un algorithme cryptographique est généralement volumineux, par exemple quelques mégaoctets, sa redistribution à des fins de piratage peut être nettement plus malcommode que celle d'une clé courte. De plus, des techniques complémentaires peuvent éventuellement empêcher l'extraction de la portion de code correspondant à l'algorithme cryptographique, ou permettre le traçage des traîtres.[32]

O. Billet, C. Ech Chaatbi et moi-même avons découvert une cryptanalyse efficace de l'implantation en boîte blanche de l'AES proposée dans [60], que nous avons publiée en 2004 [37]. L'idée de départ de notre attaque est la suivante : au lieu d'attaquer isolément les tables dont sont constituées de telles implantations, nous analysons des composées bien choisies des fonctions représentées par ces tables. Ces composées fournissent des fonctions F de 4 octets vers 4 octets représentant, à des permutations parasites inconnues près sur les octets d'entrée et de sortie, l'une des quatre bijections de 32 bits vers 32 bits dont est formé un tour d'AES. Une première étape de la reconstitution des transformations parasites consiste à retrouver les permutations parasites inconnues à une permutation linéaire près, en construisant une représentation de groupes de bijections d'un octet vers un octet induits par les fonctions F. Dans une seconde et une troisième étape, les permutations GF(2)-linéaires résiduelles sont reconstituées à une permutation GF(256)-linéaire près, puis entièrement. La complexité de l'attaque est inférieure à 2^{30}. Notre attaque exploite certaines spécificité de l'AES, en particulier les propriétés de la transformation MixColumns. Cependant, nous ne connaissons pas d'algorithme par blocs dont une implantation en boîte blanche fondée sur les mêmes techniques soit sûre.

L'implantation en boîte blanche de DES proposée par S. Chow *et al.* dans [59] a été cryptanalysée en deux temps. Une cryptanalyse d'une version affaiblie dans laquelle aucun codage externe n'est mis en œuvre (et dont les tours externes sont de ce fait particulièrement vulnérables) a été publiée par M. Jacob, D. Boneh et E. Felten en 2002 [160]. Deux cryptanalyses d'une implantation de DES comportant des codages externes ont ensuite été découvertes indépendamment, en 2007, par B. Wyseur, W. Michiels, P. Gorissen et B. Preneel [207], et L. Goubin, J-M. Masereel et M. Quisquater [116].

Plus aucune implantation en boîte blanche d'AES ou DES ne subsiste donc actuel-

[32]On pourrait non sans quelques raisons objecter qu'invoquer la possiblité de recourir à ces autres techniques ne fait que transporter les problèmes non résolus par la cryptographie en boîte blanche à un autre niveau, où rien ne garantit qu'ils soient plus faciles à résoudre.

lement. L'investigation de nouvelles techniques de cryptographie en boîte blanche se poursuit selon plusieurs directions : la recherche de techniques applicables aux algorithmes par blocs existants ; la recherche de nouveaux algorithmes par blocs possédant une clé courte, mais se prêtant mieux que les algorithmes par blocs actuels à une implantation en boîte blanche ; enfin, la recherche d'algorithmes par blocs ne possédant pas de clé courte. Le dernier problème peut sembler trivial parce que peu contraint, mais il se transforme en un problème intéressant si l'on exige en outre d'un tel algorithme que sa solidité soit fondée sur des arguments de sécurité plus forts que ceux dont on dispose pour les algorithmes par blocs proposés à ce jour.

6.2 Algorithmes par blocs traçables

La problématique du traçage des traîtres, notion proposée en 1994 par B. Chor, A. Fiat et M. Naor dans l'article fondateur [57], est celle de la protection contre le piratage de systèmes de diffusion de contenus numériques chiffrés également appelés systèmes à accès conditionnel. Les exemples de tels systèmes sont nombreux : télévision à péage, services payants de distribution en ligne de contenus multimédia, de livres ou de logiciels fondés sur un système de gestion de droits numériques ou DRM (digital rights management), distribution de contenus audio-visuels sur des supports optiques comme les DVD et des vidéodisques haute densité. Dans de tels systèmes, la fonction du chiffrement est de limiter l'accès au contenu diffusé à des utilisateurs ou des équipements légitimes, c'est-à-dire autorisés par le système. Le piratage est une utilisation détournée aboutissant à ce que des utilisateurs illégitimes, appelés **pirates**, accèdent au contenu en clair. Son origine est le plus souvent la redistribution à des fins de fraude des informations privilégiées contenues dans le décodeur[33] d'un utilisateur légitime ou d'une coalition d'utilisateurs légitimes, comportement qui vaut à ce(s) dernier(s) l'appellation de **traître**(s). Plusieurs formes de piratage peuvent être distinguées selon la nature des informations privilégiées redistribuées par les traîtres : (1) redistribution de clés de déchiffrement ; (2) redistribution d'une implantation de l'algorithme de déchiffrement extraite du décodeur d'un traître ou plus généralement déduite des implantations contenues dans les décodeurs de plusieurs traîtres ; (3) redistribution du contenu en clair.

Les techniques de traçage des traîtres visent à prévenir les risques (1) et (2) précédents[34] par un mécanisme permettant de remonter au traître, ou à l'un des membres de la coalition de traîtres qui est à l'origine du piratage. Un tel mécanisme est fondé sur l'analyse de la procédure de déchiffrement des contenus en clair implantée dans un décodeur pirate.[35] Le traçage des traîtres est une fonctionnalité paradoxale, puisqu'elle

[33]Dans un système à accès conditionnel, on désigne par décodeur l'équipement où s'effectue le déchiffrement d'un contenu diffusé.

[34]Le traçage des traîtres ne prévient pas le risque (3) de redistribution des contenus en clair. Dans certaines applications comme la télévision à péage, on peut considérer qu'une telle redistribution est malcommode et ne constitue pas une menace sérieuse, contrairement à la redistribution d'une information aussi compacte qu'une clé de déchiffrement. Dans d'autres applications, la mise en œuvre de techniques marquage de contenus peuvent contribuer à prévenir ou limiter le risque (3).

[35]Dans le cas particulier où cette analyse peut s'effectuer en observant seulement le comportement

vise à doter de caractéristiques singulières le déchiffrement des contenus diffusés mis en œuvre dans le décodeur de chaque utilisateur légitime en y inscrivant de manière indélébile l'identité de ce dernier, et ce bien que le déchiffrement effectué dans chaque décodeur produise exactement les mêmes clairs à partir des mêmes données diffusées.

Un schéma de traçage des traîtres consiste en cinq procédures : (1) la génération d'une métaclé $MK \in \mathcal{MK}$ détenue par une autorité ; (2) la génération, par cette autorité, d'au plus N clés personnelles K^i distinctes, à partir de la métaclé MK et de l'identité i de leur détenteur ; (3) le chiffrement de contenus numériques à l'aide de la méta-clé MK ; (4) le déchiffrement de contenus numériques dans le décodeur i à l'aide de la clé personnelle K^i ; (5) le traçage de traîtres, c'est-à-dire l'obtention, à partir de la description de la fonction de déchiffrement contenue dans un décodeur pirate produit par une coalition d'au plus t traîtres à partir des informations contenues dans leurs décodeurs, de l'identité de l'un au moins des traîtres.

Une partie des schémas de traçage des traîtres proposés à ce jour sont de nature combinatoire. Chaque clé personnelle de déchiffrement est constituée d'un sous ensemble K^i propre à l'usager i d'un grand ensemble de clés de base, éventuellement liées entre elles de manière hiérarchique. Un schéma de traçage des traîtres de nature non combinatoire associé à un algorithme de chiffrement asymétrique fut proposé par D. Boneh et M. Franklin en 1999 [46]. Le schéma de Boneh et Franklin possède des avantages sur les schémas combinatoires. En particulier, les clés individuelles de déchiffrement sont courtes. Cependant, les performances de ce schéma sont très sensibles à la taille maximale t des coalitions de traîtres contre lesquelles on souhaite se prémunir puisque le facteur d'expansion des clairs est proportionnel à t.

O. Billet et moi-même avons proposé en 2003 un nouveau type de schéma de traçage de traîtres non combinatoire, fondé sur la notion d'**algorithme de chiffrement par blocs traçable**, et essayé d'instancier ce concept à d'aide d'un algorithme relevant de la cryptographie multivariée. La dénomination d'**algorithme de chiffrement symétrique traçable** aurait peut-être été plus appropriée que celle d'algorithme de chiffrement par blocs traçable, puisque le chiffrement symétrique proposé ne reposait pas nécessairement sur une famille de bijections de n bits vers n bits, mais plus généralement sur une famille de fonctions de n bits vers s bits et qu'aucune propriété d'inversibilité de ces fonctions n'est requise. Un algorithme de chiffrement par blocs traçable de paramètres n et s, t (la taille maximale des coalitions de traîtres) et N (le nombre maximal d'utilisateurs) peut être défini informellement comme une famille $(\mathcal{F}_K)$ de fonctions de $\{0,1\}^n$ vers $\{0,1\}^s$ paramétrée par une clé $K \in \mathcal{K}$ et satisfaisant les conditions suivantes : (a) (F_K) est une famille pseudoaléatoire de fonctions[36] ; (b) il existe un algorithme de génération de clés personnelles permettant de déduire en un petit nombre d'opérations une clé personnelle $K^i \in \mathcal{K}$ d'une métaclé $MK \in \mathcal{MK}$ et d'un identifiant i compris entre 1 et N. Les fonctions F_{K^i} associées aux différentes va-

externe d'un décodeur pirate, c'est-à-dire la manière dont il déchiffre des chiffrés judicieusement choisis, on parle de traçage des traîtres en boîte noire (black block traitor tracing).

[36]Cette première condition est moins forte que celle habituellement imposée à un algorithme par blocs (à savoir être une famille super-pseudoaléatoire de permutations) du fait que, comme on le verra dans la suite, le déchiffrement ne nécessite pas d'inverser la fonction F^{MK}.

leurs de i doivent être égales, bien que les clés personnelles K^i soient distinctes : les clés personnelles K^i déterminent donc des représentations distinctes d'une même fonction dépendant de la métaclé MK, que l'on notera dans la suite F^{MK} ; (c) il existe un algorithme de traçage de traîtres qui étant donné une métaclé MK, et une description de F^{MK} déduite de la connaissance de $k \leq t$ clés individuelles K^{i_1} à K^{i_k} dérivées de MK et d'identités i_1 à i_k comprises entre 1 et N, permet de retrouver efficacement l'une au moins de ces identités avec une probabilité de succès proche de 1 et une probabilité de réponse erronée négligeable.

Si l'on dispose d'un algorithme par blocs traçable et d'autre part d'un algorithme de chiffrement par blocs $(E_{KS})_{KS\in\{0,1\}^s}$ paramétré par une clé KS de longueur s bits et de longueur de blocs m bits, l'algorithme par blocs traçable (F_K) peut être utilisé de la manière suivante pour chiffrer de manière probabiliste un bloc de clair M de longueur m bits : un en-tête aléatoire IV de n bits est généré, et sert à calculer successivement un mot de contrôle $KS = F^{MK}(IV)$, de longueur s bits, puis un cryptogramme $C = E_{KS}(M)$, de longueur m bits. Le chiffré de M est constitué du couple (IV, C), et a pour longueur $n+m$ bits. La connaissance d'une des clés personnelles K^i déduites de la clé MK permet de déchiffrer un chiffré (IV, C) en reconstituant le mot de contrôle $KS = F_{K^i}(IV)$, puis en appliquant l'algorithme de déchiffrement E_{KS}^{-1} au bloc C. La procédure de chiffrement précédente peut être étendue de différentes manières à un chiffrement probabiliste de messages de longueur quelconque.

Les conditions (b) et (c) précédentes sont en apparence antinomiques puisqu'elles imposent qu'il existe un grand nombre de descriptions équivalentes F_{K^i} de F^{MK}, mais interdisent que la connaissance d'une clé K^i permette de reconstituer la métaclé MK ou une clé équivalente. La métaclé MK représente donc une sorte de trappe et un algorithme par blocs traçable doit posséder des caractéristiques hybrides, le rapprochant à la fois d'un algorithme par blocs et d'un algorithme asymétrique. Le fait que les schémas issus de la cryptographie mutivariable présentent précisément de telles caractéristiques nous suggéra de nous tourner vers cette branche de la cryptographie pour essayer de construire un algorithme par blocs traçable.

Dans l'article [35] nous avons proposé un algorithme par blocs conjecturé traçable fondé sur la cryptographie multivariable. Notre schéma reposait sur la difficulté supposée de la restriction du problème IP2S d'isomorphisme de polynômes à deux secrets à des instances associées au schéma C^* de Matsumoto et Imai [159] ou plus exactement à des instances associées à une généralisation de ce schéma. Dans le schéma C^* de Matsumoto et Imai, les équations publiques, qui relient des n-uplets d'entrée et de sortie d'éléments d'un corps fini $\mathrm{GF}(q)$, sont isomorphes à une fonction monomiale $g : x \longmapsto x^{1+q^\theta}$ d'un corps fini $\mathrm{GF}(q^n)$ et sont donc de degré deux. Nous considérions des versions généralisées de C^* dont les équations publiques, de degré $d > 2$, sont isomorphes à une fonction puissance $g : x \longmapsto x^{1+q_1^\theta+\ldots+q_{d-1}^\theta}$ du corps fini $\mathrm{GF}(q)^n$. L'idée conductrice de la construction était d'exploiter le fait que les fonctions monomiales considérées commutent pour produire un grand nombre de descriptions équivalentes, sous la forme de composées de r schémas C^* généralisés, d'une bijection de $\mathrm{GF}(q)^n$ de la forme $G = T \circ g_r \circ \ldots \circ g_1 \circ S$, où S et T sont des bijection $\mathrm{GF}(q)$-linéaires et où g_1 à g_r sont des fonctions monomiales de corps $\mathrm{GF}(q^n)$, identifié à l'espace vectoriel $\mathrm{GF}(q)^n$

par le choix d'une représentation de ce corps.

Les progrès récents de la résolution de la restriction du problème d'isomorphisme de polynômes à deux secrets à des systèmes construits autour d'une fonction monomiale cachée de faible degré ont eu pour conséquence la découverte d'une cryptanalyse partielle, puis d'une cryptanalyse complète de notre schéma. Dans un premier temps, dans un travail présenté à Eurocrypt 2006 [93], J.-C. Faugère et L. Perret ont constaté expérimentalement que pour le premier jeu de paramètres suggéré dans [35] (dans lequel le nombre n de variables est seulement égal à 5), le problème IP à deux secrets pouvait être résolu efficacement à l'aide de l'algorithme de calcul de bases de Gröbner F5.[37] Cette attaque semble inapplicable au second jeu de paramètres proposé dans [35]. Dans un deuxième temps, à la suite de la publication en 2007, par V. Dubois, P.A. Fouque, J. Stern et A. Shamir [82], d'une attaque contre l'algorithme de signature SFLASH, P.A Fouque, G. Macario-Rat et J. Stern ont découvert une méthode efficace de résolution de la restriction du problème d'isomorphisme de polynômes aux équations publiques d'une instance de C^* ou d'instances généralisées de C^* telles que celles que nous avions utilisées [98]. Cette méthode permet d'attaquer efficacement notre schéma.

Il est difficile de se prononcer avec certitude sur l'intérêt du paradigme du chiffrement symétrique traçable compte tenu de la faiblesse du schéma que nous avions proposé. L'existence d'algorithmes de chiffrement symétriques traçables est une question ouverte, qui me semble mériter un supplément d'investigation.

6.3 Authentification symétrique à très bas coût

L'une des principales applications de la cryptographie symétrique est l'authentification d'entité, qui sera appelée authentification dans la suite lorsque qu'il n'existe pas de risque de confusion avec l'authentification de messages. Ce mécanisme a pour but de prévenir l'**usurpation d'identité**, qui représente dans de très nombreuses situations une menace plus forte que l'écoute des communications. L'authentification est l'un des mécanismes cryptographiques essentiels des cartes à puces ou plus généralement des puces électroniques sécurisées. Conjuguée avec la protection physique et logique contre l'extraction de leur clé secrète d'authentification qu'elles contiennent, elle permet de prévenir la fabrication de puces contrefaites reproduisant leur comportement.

Un **schéma d'authentification** symétrique fait intervenir un prouveur et un vérifieur détenant un secret partagé K, et permet au vérifieur de s'assurer de la possession de ce secret par le prouveur. La probabilité du prouveur de passer avec succès l'authentification doit être proche de 1. Celle d'un imposteur ne détenant pas le secret doit être négligeable, et ce y compris dans le cas où celui-ci a observé un grand nombre d'échanges

[37] De premiers résultats d'analyse de notre schéma avaient été précédemment obtenus par S. Alt et R. Lercier, qui proposaient dans [3] une stratégie d'implantation visant à noyer la fonction G dans une fonction opérant sur des blocs de taille beaucoup plus grande afin de déjouer la procédure de traçage. A la suite de [3] et de [93], J. Bringer, H. Chabanne et E. Dottax ont proposé [49] une variante de notre schéma consistant à introduire des perturbations dans la description de la fonction G dans le but de dissimuler les problèmes IP sous-jacents.

d'authentification antérieurs ou même joué un rôle actif dans de tels échanges.[38] Un imposteur ne doit ni pouvoir réutiliser les données ainsi observées pour se faire passer pour le prouveur légitime (**non rejeu**), ni plus généralement être capable de réaliser des calculs lui permettant d'augmenter significativement ses chances de passer avec succès une nouvelle authentification (**non forgeabilité**).

Le non rejeu est généralement assuré en faisant intervenir dans le protocole une donnée dont le vérifieur a l'assurance qu'elle n'a pas été utilisée antérieurement, par exemple un nombre aléatoire généré par le vérifieur ou la valeur d'un compteur incrémenté à chaque authentification et dont le vérifieur s'assure qu'elle est strictement supérieure à celle de la dernière authentification réussie du même prouveur. La non forgeabilité découle souvent (mais pas nécessairement, comme on le verra dans la suite) de l'utilisation par le prouveur et le vérifieur d'une fonction pseudoaléatoire paramétrée par le secret K et appliquée à une donnée non rejouable. La plupart des schémas d'authentification comportent seulement deux passes et sont de type question-réponse, mais il existe également des schémas d'authentification à une seule passe et à trois passes. Un exemple élémentaire de schéma d'authentification à deux passes reposant sur l'utilisation d'un algorithme par blocs $(E_K)_{k\in\mathcal{K}}$, ou plus généralement d'une PRF, est le suivant : le vérifieur tire au sort un bloc R de n bits, appelé question ou défi, et l'envoie au prouveur ; le prouveur calcule à l'aide du secret K qu'il détient la valeur $S = E_K(R)$ et la renvoie au vérifieur, qui utilise sa connaissance de K pour effectuer le même calcul et s'assurer que le résultat auquel il aboutit est bien égal à S.

L'authentification de puces électroniques sophistiquées dotées d'un micro-processeur peut donc être résolue à l'aide d'un algorithme par blocs courant tel que l'AES et le schéma d'authentification par défi-réponse décrit précédemment. Mais cette méthode n'est pas compatible avec les contraintes des puces électroniques dotées de ressources de calcul rudimentaires sous la forme de logique câblée telles que les étiquettes radiofréquences RFID et les puces des réseaux de capteurs, ou du moins ne satisait pas pleinement ces contraintes. La production de tels composants, qui se chiffre actuellement en milliards d'unités par an, est en augmentation très rapide. Il est généralement admis que la complexité des algorithmes ou schémas d'authentification implantés dans les versions les moins onéreuses de ces composants doit être inférieure à l'équivalent de 1000 portes logiques.[39] L'ordre de grandeur de la complexité des algorithmes par blocs adaptés à une implantation rapide en logiciel est le plus souvent de 10000 à 20000 portes logiques, à l'exception notable de DES et AES-128, dont des implantations à faible débit requérant seulement 2300 (resp. 3400) portes logiques ont été proposées [94, 152]. Le problème de la conception de schémas d'authentification sûrs satisfaisant des contraintes aussi sévères fait depuis quelques années l'objet de recherches actives.

[38] Un imposteur peut par exemple simuler le comportement d'un vérifieur auprès d'un prouveur légitime, ou encore modifier les données échangées par un prouveur et un vérifieur légitimes et observer les conséquences de cette perturbation sur le déroulement du protocole et le succès ou l'échec de l'authentification.

[39] Les autres contraintes auxquels sont soumis les schémas d'authentification implantés dans de telles puces concernent notamment la consommation électrique, la complexité de communication, i.e. le nombre de bits échangés lors de chaque authentification, et le temps de calcul.

Son intérêt est non seulement pratique compte tenu des enjeux industriels, mais aussi théorique puisqu'il est directement lié à la question de l'estimation des ressources de calcul minimales nécessaires pour construire une PRF. Plusieurs algorithmes par blocs spécifiquement conçus pour minimiser la complexité d'une implantation matérielle ont été récemment proposés. Il permettent de se rapprocher du seuil de 1000 portes logiques, voire même de l'atteindre pour un implantation à faible débit, dite « série », de l'algorithme par blocs PRESENT-80 [45, 183]. Une autre direction de recherche prometteuse, illustrée par exemple par des tentatives comme le schéma HB$^+$ [135] ou le code d'authentification de messages SQUASH [187], est l'investigation de schémas d'authentification symétrique non fondés sur l'emploi d'un algorithme par blocs. Mes principales contributions à l'authentification symétrique à très bas coût se rattachent à cette ligne de recherche.

1°) En coopération avec mes collègues D. Arditti, M. Girault et J.-C. Paillès, j'ai conçu des fonctions d'authentification d'entité et d'authentification de messages destinées à des puces à très bas coût appelées **fonctions anti-clones** (FAC). L'ordre de grandeur de la complexité d'une implantation matérielle de ces fonctions est l'équivalent de seulement 500 portes logiques. Des représentants de cette famille d'algorithmes ont été massivement implantés dans des cartes téléphoniques (télécartes dites de deuxième génération utilisées dans les publiphones de France Télécom, cartes téléphoniques d'autres opérateurs exportées notamment au Mexique et en Chine), et commencent à l'être dans des titres électroniques de transport et des puces RFID. La spécification de ces fonctions n'a pas été publiée. Les deux caractéristiques suivantes distinguent les algorithmes de cette famille des algorithmes par blocs usuels : (1) les valeurs de sortie qu'ils produisent sont très courtes : si l'on note K la clé secrète d'une fonction anticlone F, R la donnée d'entrée non rejouable, D la donnée à authentifier et $S = F(K, R, D)$ la sortie correspondante, la longueur de S est en pratique de 4 ou 16 bits. Si l'on souhaite que le taux de faux positifs (i.e. la probabilité qu'un imposteur soit authentifié avec succès) soit strictement inférieur à $2^{-|S|}$, il est nécessaire d'invoquer la fonction F à plusieurs reprises ; (2) les valeurs de sortie sont calculées au moyen d'un automate dont l'état courant évolue sous l'action d'une séquence déduite des entrées K, R et D et de longueur très largement supérieure à la somme des tailles de ces entrées.

2°) J'ai proposé dans l'article [102] une alternative aux fonctions anticlones à très bas coût offrant une sécurité inconditionnelle en contrepartie d'un allongement de la longueur de clé à quelques centaines ou milliers de bits et d'une limitation du nombre d'authentifications à quelques dizaines.[40] Ces fonctions inconditionnellement sûres sont, de même que les fonctions anticlones, des fonctions d'authentification d'entité optionnellement utilisables à des fins d'authentification de messages. Mais à la différence de ces dernières, elles sont linéaires en la clé et la donnée d'entrée. Les authentifications

[40] Cette limitation doit être maintenue sous le contrôle de la puce elle-même, qui doit incrémenter ou décrémenter un compteur et se bloquer lorsqu'un seuil est atteint, ou passer dans un mode de fonctionnement où seul un rechargement d'une nouvelle clé est possible.

successives fournissent des équations linéaires en la clé et provoquent donc son usure graduelle, mais cette usure est contrôlée : la limitation du nombre d'authentifications assure que la sortie correspondant à de nouvelles entrées est imprédictible sauf pour un ensemble négligeable d'entre elles. Notre schéma possède une certaine parenté avec la construction de codes d'authentification de messages fondés sur l'utilisation de fonctions de hachage dites universelles proposée M. Wegman et L. Carter [203] et avec son instanciation à l'aide d'un registre à décalage à rétroaction linéaire proposée par K. Krawczyk [147, 148]. Il ne relève cependant pas directement de ce paradigme.

3°) En coopération avec mes collègues M. Robshaw et H. Sibert, j'ai mis en évidence [124] l'existence d'une attaque très efficace et très simple contre le schéma d'authentification HB^+. Proposé en 2005 par A. Juels et S. Weis [135], HB^+ est fondé sur la difficulté du problème NP-complet LPN « Learning from Parity with Noise » de résolution approchée d'un système linéaire bruité. Le grand intérêt suscité par HB^+ tient à deux raisons principales : l'existence d'une preuve réductionniste reliant sa résistance à des attaques actives à la difficulté conjecturée du problème LPN[41] et le fait, que contrairement aux schémas d'authentification usuels, il se fonde exclusivement sur des opérations linéaires extrêmement simples.

La contradiction entre l'existence d'une attaque efficace contre HB^+ et celle de preuves de sécurité contre les attaques actives n'est qu'apparente. Le modèle de sécurité dans lequel se placent les preuves de sécurité de [135, 137] est très restrictif, puisqu'il ne recouvre que des attaques dans lesquelles l'adversaire interagit avec une puce légitime, puis essaie de se faire passer pour elle au cours d'une unique interaction avec un système de vérification légitime. Notre attaque [124] est une attaque par le milieu, dans laquelle un adversaire perturbe des échanges d'authentification entre une puce et un système de vérification légitimes, accède à l'information de succès ou échec de ces différentes authentifications, puis essaie de se faire passer pour la puce légitime auprès du système de vérification : bien qu'un tel scénario d'attaque type paraisse réaliste, il n'est pas couvert par le modèle de sécurité considéré dans la preuve de HB^+.

4°) A la suite de la publication de [124], plusieurs auteurs ont essayé de réparer HB^+ en spécifiant une variante qui soit résistante à des attaques par le milieu du même type. M. Robshaw, Y. Seurin et moi-même avons montré dans [111] qu'aucune des propositions dont nous avions connaissance, à savoir les schémas HB-MP [167], HB^* [83] et HB^{++} [136], n'atteignait cet objectif : le premier est vulnérable à une attaque passive ; le second à une attaque similaire à celle de [124] et de complexité également linéaire ; le troisième à une attaque de même type que celle de [124], mais plus complexe du fait que dans ce schéma la clé d'authentification est renouvelée à chaque échange : notre attaque nécessite d'observer environ 2^{35} échanges d'authentification.[42] Nous avons d'autre part

[41] J. Katz et J.S. Shin ont publié dans [137] une extension des preuves contenues dans l'article [135]. Ces preuves sont applicable à la version originale de HB^+ et à sa variante dite parallèle, qui comporte trois passes en tout au lieu de quelques dizaines d'itérations d'un protocole élémentaire à trois passes.

[42] J. Bringer et H. Chabanne ont publié très récemment une nouvelle proposition de réparation de HB^+ intitulée Trusted-HB [48].

publié en 2008 dans [112] une nouvelle proposition de modification de HB$^+$ appelée HB$^\sharp$, qui est beaucoup plus proche du schéma HB$^+$ que les variantes précédemment citées. Nous démontrons dans [112] que la version la plus générale non optimisée de notre schéma, appelée RANDOM-HB$^\sharp$, est sûre sous des conditions faciles à satisfaire sur la valeur de ses paramètres si le problème LPN est difficile.[43], dans un modèle de sécurité plus large que celui de HB$^+$ et englobant toutes les attaques telles que [124].[44] Nous proposons d'utiliser en pratique une version optimisée du schéma précédent fondée sur l'emploi de matrices de Toeplitz, pour laquelle une partie des arguments de sécurité de la version générale peuvent être conservés. Le schéma HB$^\sharp$ corrige deux défauts pratiques rédhibitoires de HB$^+$: un taux de faux négatifs inacceptablement élevé, qui se mesure en dixièmes pour les paramètres envisagés par les auteurs de ce schéma, et une très forte complexité de communication, qui se mesure en dizaines de milliers de bits. Ces deux défauts résultent d'un déséquilibre inhérent à HB$^+$ entre la taille des questions et des réponses : au moins 512 bits de question sont requis pour l'obtention d'un unique bit de réponse. Le remplacement dans HB$^\sharp$ des produits scalaires par des produits matrice-vecteur permet d'opérer un rééquilibrage entre ces paramètres.

[43]RANDOM-HB$^\sharp$ est fondé sur une transposition du problème LPN, qui fait intervenir un vecteur secret, à un problème faisant intervenir une matrice secrète que nous proposons d'appeler « puzzle MHB », dont nous prouvons qu'il est difficile si le problème LPN est difficile.

[44]Nous envisageons dans ce modèle toutes les attaques dans lesquelles un adversaire modifie les messages envoyés par un lecteur légitime à un tag légitime et a accès à l'information de succès/échec de l'authentification perturbée. Bien que nous considérions comme souhaitable qu'un schéma d'authentification soit également résistant à des attaques dans lesquelles un adversaire peut en outre modifier les messages envoyés par le tag, nous ne disposons pas de preuves de résistance à cette classe des attaques par le milieu les plus générales. K. Ouafi, R. Overbeck et S. Vaudenay ont montré lors de la « rump session » d'Eurocrypt 2008 que pour les valeurs de paramètres suggérées dans [124], RANDOM-HB$^\sharp$ et HB$^\sharp$ sont vulnérables à une attaque de ce type, dans laquelle l'adversaire modifie les messages dans les deux sens de transmission. La complexité de l'attaque est polynomiale à l'intérieur d'une classe de paramètres. Il existe cependant une autre classe de paramètre à l'intérieur de laquelle la complexité de l'attaque devient exponentielle, et il est possible nous semble-t-il de sélectionner les paramètres de RANDOM-HB$^\sharp$ et HB$^\sharp$ dans cette classe sans augmenter significativement leur taille et la complexité de ces schémas. De nouvelles propositions de paramètres sont donc en cours d'étude. L'extension des preuves de sécurité de [124] à des preuves de sécurité contre les attaques par le milieu les plus générales est une question ouverte.

Deuxième partie

Algorithmes à flot

1 Introduction

Au début des années 90, la recherche académique s'est fortement investie dans l'étude des algorithmes par blocs et pendant plus d'une décennie, le niveau de l'effort de recherche jusqu'alors à peu près équivalent consacré aux algorithmes de chiffrement à flot [184] a été nettement moindre. Pourtant, les deux algorithmes de chiffrement les plus massivement utilisés dans le monde étaient et restent des algorithmes de chiffrement à flot.[45] Ce décalage entre recherche et utilisation industrielle s'est comblé au cours des dernières années, et dans les principales conférences de cryptographie, le nombre des publications consacrées aux algorithmes à flot a même dépassé depuis peu celui des publications consacrées aux algorithmes par blocs. Une légère différence de maturité n'en demeure pas moins perceptible entre les deux domaines : aucun algorithme à flot aussi reconnu que des algorithmes par blocs comme DES ou AES ne s'est imposé à ce jour. Ce constat et l'observation que les algorithmes à flot représentent une primitive moins universelle que les algorithmes par blocs ont conduit A. Shamir à diagnostiquer en 2004 que les algorithmes à flot étaient une espèce en voie de disparition. Il s'agissait là cependant d'une prédiction auto-destructrice plutôt qu'auto-réalisatrice, et tel était d'ailleurs probablement le souhait de son auteur. L'initiative eSTREAM [1], coordonnée par le réseau européen d'excellence en cryptographie ECRYPT, témoigne du regain de vitalité de cette famille d'algorithmes. Elle a abouti en avril 2008 à la sélection, parmi les seize des trente-quatre algorithmes candidats retenus comme finalistes [181], d'un portefeuille restreint de huit algorithmes à flot adaptés à une implantation matérielle ou logicielle.

On peut définir informellement un algorithme de chiffrement à flot comme un **générateur de nombres** qui à une clé secrète associe non pas une fonction comme dans le cas d'un algorithme par blocs, mais une suite de symboles d'un alphabet fini appelée **suite chiffrante**. Pour chiffrer un clair représenté par une suite de symboles du même alphabet, on combine chacun des symboles du clair avec le symbole correspondant de la suite chiffrante à l'aide d'une opération de groupe sur l'alphabet choisi. La suite de symboles ainsi obtenue est appelée suite chiffrée. Le déchiffrement s'effectue en appliquant à la suite chiffrée la suite des inverses des symboles de la suite chiffrante.[46]

La définition précédente est quelque peu restrictive, puisqu'elle ne recouvre que les algorithmes à flot dits **synchrones**, qui produisent une suite chiffrante indépendante du clair. Elle ne recouvre pas les classes suivantes d'algorithmes à flot, qui produisent une suite chiffrante dépendant du clair :
- les algorithmes à flot **autosychronisants**, dont la caractéristique essentielle est que les chiffrés précédents rétroagissent sur le calcul de la suite chiffrante. La conception

[45] Il s'agit de l'algorithme A5/1, qui sert au chiffrement de la voie radio dans le système de communication avec les mobiles GSM et de l'algorithme RC4, employé dans le protocole SSL pour le chiffrement des transactions sécurisées de l'Internet. Ces deux algorithmes ont été conçus vers la fin des années 80. Leur spécification est restée secrète pendant plusieurs années.

[46] L'alphabet utilisé est le plus souvent l'alphabet binaire $\{0, 1\}$, et l'opération de groupe employée est le « ou exclusif », i.e. l'addition modulo 2 : les procédures de chiffrement et de déchiffrement sont donc identiques.

d'algorithmes autosynchronisants efficaces et solides est une gageure ;
- les algorithmes de **chiffrement authentifié**, dans lesquels le clair intervient dans le calcul de la suite chiffrante et qui produisent non seulement une suite chiffrante, mais également un suffixe dépendant du clair et de la clé possédant les propriétés d'un code d'authentification de messages.

Seuls des algorithmes à flot synchrones seront considérés dans la suite. Un des principaux avantages du chiffrement à flot synchrone est la non propagation des erreurs : une erreur de transmission d'un symbole de chiffré n'affecte après déchiffrement qu'un unique symbole de clair. Cette propriété est indispensable dans de nombreux systèmes de communication radio.

Tous les algorithmes à flot proposés depuis une dizaine d'années possèdent, en plus de la clé, un second paramètre d'entrée, non secret, appelé **valeur d'initialisation**, ou **vecteur d'initialisation**, ou **IV**. La valeur d'initialisation servant au chiffrement d'un message doit être une donnée non rejouable accessible aux processus de chiffrement et de déchiffrement, par exemple un nombre aléatoire généré par l'entité émettrice et transmis en clair à l'entité réceptrice. L'introduction de valeurs d'initialisation dans les algorithmes à flot présente des avantages pratiques incontestables : elle permet d'associer à une même clé secrète plusieurs suites chiffrantes indépendantes, et donc de chiffrer plusieurs messages avec la même clé (et des IV distincts). Mais elle a radicalement transformé la nature des algorithmes à flot et les conditions pour que ces derniers soient sûrs.

Une caractérisation très simple des propriétés de sécurité attendues d'un algorithme à flot ne dépendant que d'une clé a été dégagée dès le début des années 80. Un tel algorithme est considéré comme sûr s'il s'agit d'un **générateur pseudoaléatoire de nombres** ou **PRNG**, i.e. s'il est informatiquement impossible à un adversaire capable d'accéder à une suite chiffrante associée à une clé aléatoire inconnue K de distinguer cette suite chiffrante d'une suite parfaitement aléatoire de même longueur au moyen d'un test informatiquement réalisable de probabilité de succès non négligeable. Cette condition assure notamment qu'il est impossible de retrouver la clé, ou une clé équivalente, ou de déduire de la connaissance d'une partie de la suite chiffrante des informations sur les autres symboles de cette suite.

Dans le cas d'un algorithme à flot dépendant d'un IV, il n'existe pas de définition aussi unanimement adoptée. La différence avec la définition d'un algorithme par blocs sûr esquissée dans la partie précédente s'estompe quelque peu puisqu'il devient nécessaire de considérer des attaques dans lesquelles l'adversaire connaît et/ou contrôle une partie des données d'entrée de l'algorithme, à savoir les attaques à IV connu ou choisi également appelées attaques par resynchronisation. L'on peut caractériser très simplement un algorithme à flot sûr dépendant d'un IV à l'aide de la notion de PRF ; cette question sera examinée plus en détail dans la section 4.2.

La plupart des algorithmes à flot existants ont une structure d'**automate à états finis**. L'état initial de l'automate dépend (généralement de manière non linéaire) de la clé et de la valeur d'initialisation. Le calcul de l'état initial est parfois appelé **établissement de clé et d'IV** (key and IV setup). A chaque étape du processus de génération de la suite chiffrante qui suit l'établissement de clé et d'IV, l'état courant de l'automate est

mis à jour à l'aide d'une **fonction de transition**, et un ou plusieurs nouveaux symboles de la suite chiffrante sont déduits de l'état courant à l'aide d'une **fonction de sortie**. Le plus souvent, la fonction de transition et la fonction de sortie sont indépendantes de la clé secrète et de la valeur d'initialisation.

Il serait malaisé et fastidieux d'établir une typologie systématique des algorithmes à flot. L'on se restreindra ici à un rappel succinct des principales familles élémentaires d'automates à états finis que l'on rencontre dans les algorithmes à flot actuels.[47]

1°) **Registres à décalage à rétroaction linéaire ou LFSR**, de l'anglais « Linear Feedback Shift Register ». Un LFSR est caractérisé par son état initial et son polynôme de rétroaction ; il produit une suite récurrente linéaire à valeurs dans un corps fini, le plus souvent $\mathrm{GF}(2)$ ou $\mathrm{GF}(2^w)$. Les LFSR, dont on pourra trouver une théorie complète dans [186], constituent l'un des objets mathématiques les plus simples et les plus étudiés rencontrés en cryptographie.[48] L'automate d'un algorithme à flot peut mettre en œuvre :

(a) Des LFSR à avancées régulières, i.e. dont le registre subit un nombre constant d'avancées et produit à chaque transition de l'automate un nombre constant de bits chiffrants ; ainsi, le LFSR de l'algorithme SNOW 2.0 [84], algorithme conçu par P. Ekdahl et T. Johansson, est à valeurs dans $\mathrm{GF}(2^{32})$. L'automate de SNOW 2.0, constitué d'un LFSR et de deux mots de mémoire évoluant selon une logique non linéaire, subit exactement une avancée et produit exactement 32 bits chiffrants à chaque transition ;

(b) Des LFSR à avancées irrégulières, i.e. dont le registre subit un nombre variable d'avancées ou produit un nombre variable de bits chiffrants à chaque transition de l'automate. On peut citer comme exemples d'algorithmes à flot mettant en œuvre des LFSR à avancées irrégulières le **Shrinking Generator** [64], conçu par D. Coppersmith, P. Rogaway et Y. Mansour, le **Self-Shrinking Generator** [163], conçu par W. Meier et O. Staffelbach, ou encore l'algorithme **A5/1** du système GSM.

2°) **Registres à décalage à rétroaction non linéaire ou NFSR**, de l'anglais « Non-linear Feedback Shift Register », produisant des suites récurrentes non linéaires. GRAIN [122], conçu par M. Hell, T. Johansson et W. Meier, et TRIVIUM [52], conçu par C. De Cannière et B. Preneel, sont deux exemples caractéristiques d'algorithmes à flot reposant sur l'emploi de NFSR.

3°) **Automates dont l'état courant est constitué de tables évoluant dynamiquement**. Deux exemples d'algorithmes reposant sur un automate de ce type sont RC4 [179], conçu par R. Rivest, et HC, conçu par H. Wu [204, 205, 206].

[47] De nombreux algorithmes à flot reposent sur un automate hybride, contenant simultanément des composantes relevant de plusieurs de ces familles élémentaires d'automates.

[48] Leur utilisation dans un grand nombre d'algorithmes à flot est motivée d'une part par leur coût d'implantation très faible, et d'autre part par le fait qu'elle permet d'assurer des propriétés relatives à la période et à l'équilibre de la suite récurrente qu'ils produisent.

4°) **Automates dont la fonction de transition utilise des transformations comparables aux tours d'un algorithme par blocs**, et notamment des transformations fondées sur des boîtes S. Les algorithmes SCREAM [117], conçu par S. Halevi, D. Coppersmith et C. Jutla et MUGI [202], conçu par D. Watanabe, S. Furuya, K. Takaragi et B. Preneel, sont des exemples caractéristiques de cette famille d'automates.

D'autres types d'automates comme les **automates cellulaires** ou les FCSR (Feedback with Carry Shift Registers), sont utilisés dans un nombre beaucoup plus restreint d'algorithmes.

Les attaques contre les algorithmes à flot peuvent être caractérisées selon trois critères.

1°) **Le but de l'adversaire** peut être l'un des suivants : retrouver la clé secrète ou une clé équivalente ; prédire une partie de la suite chiffrante, par exemple en reconstituant l'état initial ou l'un des états ultérieurs de l'algorithme ; distinguer au moyen d'un test les suites produites par l'algorithme de suites parfaitement aléatoires. L'attaque est alors désignée sous le nom d'**attaque par distingueur**.

2°) **Les moyens de l'adversaire** : dans le cas d'un algorithme à flot dépendant d'un IV, on peut distinguer les attaques à IV connus et les attaques à IV choisis (éventuellement de manière adaptative). Dans les deux cas, l'adversaire est supposé disposer de suites chiffrantes correspondant à une clé inconnue, ou ce qui revient au même de chiffrés et des clairs correspondants.

3°) **La technique d'attaque** : la plupart des attaques connues relèvent de l'une des quatre techniques cryptanalytiques suivantes qui peuvent être le cas échéant combinées entre elles : **cryptanalyse statistique**, dont une sous-famille importante est celle des attaques par corrélation ; attaques par **test d'hypothèses partielles**, traduction approximative de l'anglais « guess and determine » ; **attaques algébriques** ; **compromis temps/données/mémoire**.

J'ai porté depuis le début de mes recherches en cryptographie un intérêt à peu près égal aux algorithmes par blocs et aux algorithmes à flot. Cependant, mes premiers travaux sur les algorithmes à flot étaient liés à mes activités de conception et d'analyse d'algorithmes pour les systèmes mobiles qui, au début des années 90, se prêtaient mal à publication. C'est en grande partie pour cette raison que mes publications dans le second domaine sont beaucoup plus récentes que dans le premier.

2 Contributions à la cryptanalyse des algorithmes à flot

Ce chapitre présente succinctement mes principales contributions à la cryptanalyse des algorithmes à flot, qui reflètent la grande diversité des angles potentiels d'attaque qui se présentent à un cryptanalyste. Bien que certaines d'entre elles étendent quelque peu la palette des méthodes d'attaque connues, chacune d'entre elles se rattache cependant clairement à l'une des quatre familles d'attaques mentionnées ci-dessus.

2.1 Cryptanalyse statistique de SEAL

Ma première publication dans le domaine des algorithmes de chiffrement à flot date de 1997. Il s'agit d'une cryptanalyse statistique de l'algorithme SEAL, mise au point en coopération avec H. Handshuh [118]. SEAL est un algorithme à flot très rapide proposé par P. Rogaway et D. Coppersmith d'IBM [182]. Notre attaque fait appel à des techniques également applicables aux algorithmes par blocs, et peut être qualifiée de cryptanalyse χ^2, méthode fondée sur le test de même nom introduite par S. Vaudenay pour désigner une variante de la cryptanalyse linéaire du DES [197]. Une des principales caractéristiques de SEAL est qu'il fait intervenir des tables dépendant de la clé. La partie la plus critique de SEAL du point de vue des performances et de la sécurité a une structure comparable à celle d'un algorithme par blocs très simple, opérant par chaînage : si l'on désigne le bloc courant constitué de $m = 4$ mots de $w = 32$ bits par $(x_i)_{i=0\ldots m-1}$, nous désignons ici par chaînage une succession de transformations de la forme $x_{i+1} = x_{i+1} * f(x_i)$, où $*$ et f représentent respectivement une opération inversible à droite et à gauche et une fonction, et où les indices sont considérés modulo m. Notre attaque exploite le fait qu'alors que ce type de chaînage induit une diffusion rapide de l'influence des différents mots du bloc d'entrée vers les mots du bloc de sortie, la diffusion se fait beaucoup plus lentement dans la direction inverse.[49] Cette remarque permet de prédire que la distribution de probabilité d'un mot de 4 bits s'exprimant comme une fonction linéaire de trois mots de 32 bits de la suite chiffrante est déséquilibrée. Un test du χ^2 appliqué à ce mot permet de confirmer ce biais. Ce distingueur nécessite environ 2^{34} octets de suite chiffrante. Au prix d'une très légère augmentation de la complexité de l'attaque il est possible de reconstituer une partie des informations de l'une des tables secrètes. A la suite de notre attaque, les auteurs de SEAL proposèrent une nouvelle version, SEAL 3.0, pour laquelle un nouveau distingueur nécessitant environ 2^{43} octets de chiffré fut découvert par S. Fluhrer en 2001 [97]. Un algorithme de la même famille très légèrement moins rapide, SCREAM, fut ensuite proposé par IBM en remplacement de SEAL [117].

[49] Une propriété analogue a été exploitée récemment dans une cryptanalyse découverte par S. Babbage *et al.* [9] de l'un des candidats de la compétition eSTREAM, l'algorithme Hermes8.

2.2 Attaques par corrélation et application à la cryptanalyse de Grain

Les **attaques par corrélation** s'appliquent principalement, mais pas exclusivement, à des algorithmes reposant sur l'utilisation d'un ou plusieurs LFSR. Elles exploitent l'existence d'une corrélation statistique[50] entre la suite chiffrante produite par l'algorithme réel G et la suite ou plus généralement les k suites produite(s) par un générateur de nombres plus simple g induit par G. L'observation d'une suite chiffrante suffisamment longue produite par G permet de déterminer les paramètres de g et fournit un distingueur entre la distribution de sortie de G et la distribution uniforme. Elle représente de plus, le plus souvent, une première étape vers la détermination des paramètres de G tout entier, par exemple l'état initial ou la clé de G.

La notion d'attaque par corrélation a été introduite par T. Siegenthaler [191, 192], qui l'a initialement appliquée à des **générateurs à combinaison de registres** et à des **registres à décalages à rétroaction linéaire filtrés non linéairement**, deux familles d'algorithmes à flot reposant sur l'emploi de LFSR et de fonctions booléennes assurant le filtrage non linéaire de l'état des LFSR. Bien que ces deux familles de générateurs soient assez peu utilisées en pratique, elles sont devenues les exemples d'algorithmes à flot dont la cryptanalyse a été le plus étudiée. Elles ont servi de référence d'une part pour définir des critères sur les fonction booléennes pour rendre inopérantes les attaques par corrélation, et d'autre part pour définir et comparer des techniques plus avancées d'attaque par corrélation.

Je n'aborderai pas ici la question de l'étude des fonctions booléennes utilisées dans les algorithmes à flot, de la définition de critères liés à leur résistance à la cryptanalyse, et des méthodes de génération de fonctions satisfaisant un ou plusieurs ces critères, sujets dont je ne suis pas un spécialiste, et dont on pourra trouver une synthèse dans des références comme [55, 53]. Pour ce qui concerne le second aspect, qui est plus lié à mes travaux, il me semble utile de mentionner, sans prétendre à l'exhaustivité, quatre des principales avancées dans le domaine des attaques par corrélation depuis la publication de l'attaque de Siegenthaler.

(1) L'attaque par corrélation rapide de Meier et Staffelbach [162] s'applique, comme l'attaque de Siegenthaler, à des situations où il existe une corrélation, mesurée par un coefficient de corrélation moyen χ, entre la suite chiffrante et une suite récurrente linéaire binaire engendrée par un LFSR S. La longueur de S peut éventuellement être très grande, mais l'attaque n'est réalisable que si le cryptanalyste est capable de trouver au moins un multiple $q(x)$ du polynôme de rétroaction $p(x)$ du LFSR dont le poids de Hamming soit faible, i.e. ne possède qu'un petit nombre w de coefficients non nuls. La reconstitution de l'état initial de S à partir de N bits consécutifs de la suite chiffrante repose sur le décodage d'un code linéaire possédant des équations de parité de poids w induites par $q(x)$ et par les puissances de $q(x)$ de la forme $q(x)^{2^i}$. Deux algorithmes de

[50] Deux suites binaires $(a_i)_{i\in[1,\ldots,n]}$ et $(b_i)_{i\in[1,\ldots,n]}$ fixes ou dépendant d'un paramètre aléatoire $\omega \in \Omega$ sont dites corrélées si leur coefficient de corrélation $\chi = \frac{1}{n}\sum_{i=1}^{n}(-1)^{a_i \oplus b_i}$, resp. $\chi = \frac{1}{n}\sum_{i=1}^{n} \Pr_\omega(a_i = b_i) - \Pr_\omega(a_i \neq b_i)$, est non nul. Le coefficient de corrélation χ est compris entre -1 et $+1$, et égal au double du biais moyen de la suite $(a_i \oplus b_i)$.

complexité très inférieure à celle de l'attaque de Siegenthaler lorsque w est suffisamment petit et N et $|\chi|$ suffisamment grands sont proposés dans [162]. Diverses extensions de cette attaque fondatrice ont été proposées par la suite [54].

(2) Dans une série d'articles parus à la fin des années 90, T. Johansson et F. Jönsson ainsi que V. Chepyzhov, T. Johansson et B. Smeet ont proposé de nouvelles attaques par corrélation rapides applicables, contrairement à l'attaque initiale de Meier et Staffelbach, au cas où la suite chiffrante est corrélée à celle produite par un LFSR de polynôme de rétroaction $p(x)$ quelconque [133, 132, 134, 56]. L'une de ces attaques [56], qui a apporté des gains de performances spectaculaires, exploite l'idée très simple suivante, reprise dans la plupart des attaques par corrélation ultérieures : au lieu de rechercher comme dans l'attaque de Meier et Staffelbach des équations de parité de poids faible satisfaites par les bits de la suite récurrente, l'on peut rechercher des équations de parité dont le nombre de coefficients non nuls soit faible hors d'une « fenêtre » de $m < n$ bits consécutifs de la suite récurrente, pour lesquels on autorise des coefficients quelconques. Cette fenêtre peut par exemple être constituée des m premiers bits de la suite récurrente. L'attaque de [56] comporte deux phases, que l'on retrouve dans la plupart des attaques par corrélation récentes : le précalcul d'un grand nombre d'équations de parité et la recherche du m-uplet de bits maximisant un indicateur tel que le nombre d'équations de parité satisfaites par la suite chiffrante.

(3) Parallèlement au développement des attaques par corrélation rapides et indépendamment de ces dernières, une autre idée d'amélioration de l'attaque de Siegenthaler a été proposée au début des années 90 : l'utilisation de la transformée de Fourier, plus précisément de sa variante discrète appelée transformée de Walsh-Hadamard ou transformée de Walsh.[51] Le recours à l'algorithme de transformée de Walsh rapide, analogue discret de la transformée de Fourier rapide, permet en effet d'accélérer les calculs de corrélation. L'idée d'utiliser la transformée de Walsh rapide dans le contexte des attaques par corrélation apparaît à ma connaissance pour la première fois dans un article

[51]La transformée de Walsh peut être définie de la manière suivante. On se place dans l'espace euclidien $\mathcal{F}_n$ des fonctions réelles d'une variable discrète $x \in \{0,1\}^n$, muni du produit scalaire défini par $(f|g) = \sum_{x\in\{0,1\}^n} f(x)g(x)$. Une base orthogonale de $\mathcal{F}_n$ est constituée des 2^n fonctions de la forme $(-1)^{(\omega|\cdot)}$, où ω désigne un vecteur de $\{0,1\}^n$ et $(\omega|\cdot)$ la forme linéaire de $\{0,1\}^n$ associée. La transformée de Walsh d'une fonction f de $\mathcal{F}_n$ est la fonction $T(f)$ de $\mathcal{F}_n$ définie par $T(f) : \omega \in \{0,1\}^n \mapsto T(f)(\omega) = (f|(-1)^{(\omega|\cdot)})$. Il existe un lien étroit entre transformée de Walsh et corrélation. La corrélation $c(f,g)$ (resp. la distance $d(f,g)$) entre deux fonctions booléennes dépendant de n variables binaires est la différence entre (1) le nombre de valeurs $x \in \{0,1\}$ pour lesquelles f et g sont égales et (2) le nombre de valeurs pour lesquelles elles sont distinctes (resp. le nombre de valeurs où elles sont distinctes). On a donc $c(f,g) = ((-1)^f|(-1)^g) = 2d(f,g) - 2^n$. La définition de la transformée de Walsh montre que si f est une fonction booléenne dépendant de n variables et ω est un vecteur de $\{0,1\}^n$, la valeur en ω de $T((-1)^f)$ peut être interprétée comme la corrélation entre f et la forme linéaire $(\omega|\cdot)$. Il en résulte que pour trouver la ou les meilleures approximations linéaires ou affines de f, il suffit de calculer la transformée de Walsh de $(-1)^f$ et de retenir la ou les valeurs de ω pour lesquelles la corrélation obtenue (resp. la valeur absolue de la corrélation obtenue) est maximale. Le coût d'une telle recherche est de $n2^n$ au lieu de 2^{2n} pour une méthode naïve.

de T. Beth, D. Gollman et S. Mund [166]. On la retrouve formulée d'une manière moins liée au détail de l'attaque de Siegenthaler dans le mémoire de thèse de M. Dodd [81] dont j'ai eu l'occasion de consulter une version préliminaire dès 1999. Cette technique permet à un attaquant de retrouver l'état initial d'un LFSR de longueur l bits générant la suite récurrente à laquelle la suite chiffrante est corrélée en environ $l \cdot 2^l$ opérations au lieu de $N \cdot 2^l$ opérations, où N désigne la longueur de la suite chiffrante. La réduction du temps de calcul résultant de cette technique peut être considérable et représenter en pratique un facteur de plusieurs milliards.

(4) Une extension naturelle des attaques par corrélation consiste à rechercher des corrélations impliquant non pas les bits de la suite chiffrante, mais les bits d'une suite déduite de la suite chiffrante en lui appliquant un filtrage linéaire adéquat. Les corrélations recherchées doivent lier la suite filtrée à une suite produite par un générateur g' plus simple que G. La recherche du masque linéaire caractérisant un tel filtrage repose sur l'analyse des composantes non linéaires et linéaires de l'algorithme à flot étudié. Cette méthodologie d'attaque a été intitulée **cryptanalyse par masquage linéaire** par D. Coppersmith, S. Halevi et C. Jutla [63], et appliquée par ces derniers aux algorithmes SNOW 1.0 et SCREAM-0. Le cas le plus souvent rencontré en pratique est celui où la suite filtrée présente un biais : elle est alors corrélée à la suite nulle, et g' se réduit au générateur produisant la suite nulle. S'il existe un biais suffisamment fort, l'algorithme est vulnérable à une attaque par distingueur.

Une étude réalisée en 2000 en cooopération avec P. Audoux nous permit d'établir que les techniques (2) et (3) ci-dessus peuvent être combinées, et que les performances d'attaques par corrélation rapides telles que celle de [56] peuvent être améliorées de plusieurs ordres de grandeur par l'emploi de la transformée de Walsh rapide. Les attaques par corrélation rapides considérées se décomposent en deux phases. La première phase consiste à rechercher un grand nombre R d'équations de parité de poids faible hors d'une fenêtre de m bits, et la seconde phase à exploiter ces équations pour rechercher la valeur de ces m bits. La transformée de Walsh rapide permet de ramener la complexité de la seconde phase de $R \cdot 2^m$ pour un calcul de corrélation naïf à seulement $m \cdot 2^m$; la quantité de mémoire nécessaire est de $\min(R, 2^m)$ environ. Lorsque $R < 2^m$, une méthode hybride de calcul de corrélation combinant recherche exhaustive de $m_1 < m$ des m bits inconnus et transformée de Walsh rapide sur les $m_2 = m - m_1$ variables rédiduelles permet en effet de réduire la quantité de mémoire nécessaire aux calculs à environ R sans augmentation notable de la complexité en temps. Cette diminution de la complexité de la deuxième phase de l'attaque rend possible le traitement au cours de cette phase de valeurs de m nettement supérieures. Elle permet d'alléger les précalculs de la première phase en augmentant la valeur de m conduisant à un compromis optimal entre les deux phases, et donc de diminuer notablement la complexité globale de l'attaque. Un article décrivant cette amélioration des attaques par corrélation rapides fut soumis en 2000, mais refusé [104]. L'idée d'utiliser la transformée de Walsh rapide dans le contexte de telles attaques fut découverte indépendamment de notre travail par P. Chose, A. Joux et M. Mitton, et publiée en 2002 [58]. Une amélioration supplémentaire, relative à la

phase de précalcul, était proposée dans l'article [58].

Ce n'est qu'en 2006, dans un article résultant d'une coopération avec C. Berbain et A. Maximov consacré à la cryptanalyse de l'algorithme à flot GRAIN [19], que je trouvai l'occasion de publier une attaque illustrant pleinement le potentiel de l'utilisation combinée des techniques (2) et (3) dans les attaques par corrélation.

Cryptanalyse de Grain

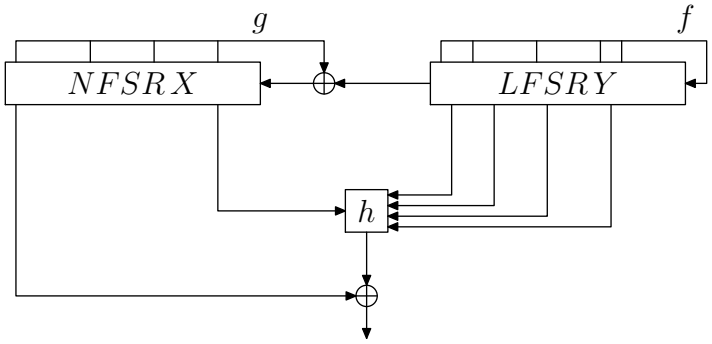


Fig. 1 – GRAIN : génération de la suite chiffrante

GRAIN est un algorithme à flot remarquablement adapté à une implantation matérielle proposé par M. Hell, T. Johansson et W. Meier en réponse à l'appel à propositions eSTREAM [122]. Les longueurs de clé et d'IV sont égales à 80 bits. L'automate produisant la suite chiffrante est illustré à la figure 1. Il est constitué d'un NFSR X de longueur $n = 80$ bits perturbé par les bits de sortie d'un LFSR Y de même longueur. Les bits chiffrants sont déduits de l'état courant des registres X et Y par un « ou exclusif » entre un bit de masquage issu de X et l'image par une fonction booléenne h de quatre des bits de Y et l'un des bits de X. L'état initial de l'automate résulte d'une procédure d'établissement de clé et d'IV très simple.

Notre cryptanalyse de GRAIN exploite l'existence d'une corrélation entre les bits de la suite obtenue en appliquant à la suite chiffrante un filtrage linéaire bien choisi et certaines fonctions affines connues des bits d'initialisation du LFSR. Bien que conscients de l'existence d'approximations affines de la fonction h ne faisant intervenir que des bits du LFSR, les concepteurs de GRAIN avaient semble-t-il considéré que le masquage de la sortie de h à l'aide du bit x_t produit par le NFSR interdisait de les exploiter. Or les bits de masquage ne sont pas indépendants. La fonction de rétroaction du NFSR induit un fort déséquilibre dans la somme $x_t \oplus x_{t+80} \oplus y_t$ du bit de masquage x_t et du bit de rétroaction $g(X_t) = x_{t+80} \oplus y_t$ produits à l'instant t. Par conséquent, la somme σ_t des bits chiffrants produits aux instants t et $t+80$ est corrélée à la somme des sorties de h à ces deux instants et de y_t. L'existence d'approximations affines de h ne dépendant que des bits d'entrée issus du LFSR induit seize approximations affines de σ_t dépendant des 80 bits de l'état initial du LFSR et de biais η égal à environ 2^{-9}.

Comme le biais η est fort, il est avantageux, au lieu d'exploiter directement les N approximations affines obtenues, de considérer les sommes deux à deux de telles

approximations ne faisant intervenir qu'un sous ensemble fixé de $m < n$ inconnues. En d'autres termes, l'on recherche les couples d'approximations pour lesquelles une collision entre les coefficients des $n - m$ autres inconnues se produit. Cette méthode fondée sur le paradoxe des anniversaires relève de la famille (2) d'améliorations des attaques par corrélation précédemment décrite. Elle fournit $R \simeq N^2 \cdot 2^{m-n}$ relations de biais $\epsilon = 2\eta^2$. Les valeurs de m considérées sont en pratique voisines de $\frac{n}{2} = 40$. Des variantes plus élaborées fondées sur l'itération de cette méthode ou sur l'utilisation de sommes de quatre équations de biais η au lieu de deux [58] ne semblent pas conduire dans le cas de GRAIN à une attaque plus performante.

Le cryptanalyste est donc ramené au problème d'estimer, à partir d'un grand nombre R d'équations affines approchées en m bits inconnus satisfaites avec une probabilité $\frac{1}{2}+\epsilon$, la valeur de ces m bits. Il s'agit plus précisément de déterminer pour quelle(s) valeur(s) de ces m bits le plus grand nombre d'équations approchées est satisfait ou, ce qui revient au même, la différence entre le nombre d'équations satisfaites et non satisfaites est la plus grande. L'on est ramené à un calcul de 2^m corrélations entre deux vecteurs de R bits puisque pour chaque hypothèse sur les m bits inconnus, il est nécessaire de calculer la corrélation entre le R-uplet de sommes de la forme $\sigma_t \oplus \sigma_{t'}$ observées et le R-uplet des valeurs prédites par les R approximations affines correspondantes. La technique fondée sur l'utilisation de la transformée de Walsh que nous avions mise au point en 2000 [104] permet d'accélérer tout calcul de ce type. Il est en effet facile de montrer que les 2^m corrélations recherchées sont données par la transformée de Walsh de la fonction qui à un m-uplet $\mathbf{a}$ de coefficients binaires associe la différence entre les nombres (éventuellement nuls) d'approximations affines de la forme $\mathbf{a} \cdot \mathbf{x} = 1$ et d'approximations affines de la forme $\mathbf{a} \cdot \mathbf{x} = 0$ satisfaites. La méthode hybride combinant recherche exhaustive et transformée de Walsh mentionnée précédemment peut en outre être appliquée pour diminuer la quantité de mémoire requise.

On peut comparer le problème ainsi résolu de l'estimation de n bits inconnus de l'état initial de GRAIN au problème LPN (Learning from Parity with Noise), issu du codage et de l'apprentissage automatique et récemment porté à l'attention des chercheurs en cryptographie par la proposition de schéma d'authentification symétrique HB$^+$ [135]. La méthode de reconstitution des n bits esquissée ci-dessus est très proche de la méthode de résolution du problème LPN récemment proposée par E. Levieil et P.-A. Fouque [153], qui est une amélioration de la méthode de résolution BKW proposée par A. Blum*et al.* [42]. Contrairement à BKW, la méthode de [153] fait appel à la transformée de Walsh rapide.

Une fois les m bits inconnus déterminés, des étapes de moindre complexité permettent de reconstituer la totalité de l'état initial du LFSR, puis l'état initial du NFSR, et finalement la clé. Notre attaque de [122] requiert environ 2^{38} bits chiffrants résultant d'une même clé inconnue et d'une même valeur d'initialisation et une mémoire d'environ 2^{42} octets. Sa complexité est d'environ 2^{43} opérations élémentaires. Une attaque par distingueur nécessitant environ 2^{51} bits chiffrants découverte par A. Maximov indépendamment de l'attaque précédente est également mentionnée dans notre article commun, et décrite plus en détail dans [161]. Une autre attaque par distingueur nécessitant environ 2^{61} bits chiffrants a été proposée par S. Khazaei *et al.* dans [140].

A la suite de ces attaques, les auteurs de GRAIN ont proposé une nouvelle version, désignée sous le nom de GRAIN v1, dans laquelle plusieurs modifications ont été introduites, notamment un nombre substantiel de bits de masquage supplémentaires [123, 1]. Aucune attaque contre GRAIN v1 n'a été découverte à ce jour excepté une « sliding attack » permettant de réduire d'un facteur de deux la complexité d'une recherche exhaustive de la clé [51]. GRAIN v1 figure dans le portefeuille des huit algorithmes à flot sélectionnés à l'issue de la compétition eSTREAM, dont il est l'un des plus prometteurs.[52]

2.3 Attaque par resynchronisation de Pomaranch

C. Cid, T. Johansson et moi-même avons publié en 2006 [61] une attaque contre un autre candidat de la compétition eSTREAM, l'algorithme à flot Pomaranch. Pomaranch [131], conçu par C. J. Jansen, T. Helleseth et A. Kholosha, a une longueur de clé et d'IV de 128 bits. Il est fondé sur une classe particulière de LFSR à avancées irrégulières commandés par un bit de contrôle appelés « jumping registers ». Selon la valeur du bit de contrôle, un jumping register est susceptible d'effectuer à chaque coup d'horloge (1) une unique avancée, sous l'action de la matrice de transition A du LFSR, ou (2) l'équivalent d'un grand nombre d'avancées de matrice A, sous l'action de la matrice de transition $I + A$. L'automate de Pomaranch est constitué de neuf jumping registers en cascade. A chaque transition de l'automate, un bit chiffrant égal à une combinaison linéaire de bits extraits de ces registres est produit. L'attaque que nous avons proposée est une attaque par resynchronisation. Elle nécessite de disposer des premiers bits des suites chiffrantes associées à un très petit nombre d'IV choisis. Elle exploite un défaut de conception de la procédure d'établissement de clé et d'IV dont il résulte que certains des bits de l'IV ne se diffusent que vers un nombre restreint de registres. Elle tire également parti d'une limitation inhérente aux jumping registers : comme les matrice A et $I + A$ commutent, la matrice de transition représentant n étapes ne peut prendre que l'une des $n + 1$ valeurs $A^p(I + A)^{n-p}$, $0 \leq p \leq n$, entre lesquelles elle se répartit selon une distribution binomiale. La complexité de l'attaque est d'environ 2^{65} opérations élémentaires, et peut être ramenée à 2^{52} si l'on exploite un défaut supplémentaire de la procédure d'initialisation de l'algorithme. Une autre attaque contre Pomaranch exploitant l'existence de biais forts sur certaines combinaisons linéaires de bits produits par un jumping register fut découverte par S. Khazaei [139]. A la suite de ces attaques, une deuxième version de l'algorithme fut proposée puis cryptanalysée par M. Hell et T. Johansson [121]. Elle fut bientôt suivie d'une troisième version qui figurait parmi les seize finalistes de la compétition eSTREAM [181]. Cette dernière n'est cependant pas non plus exempte de tout défaut [85].

[52] Outre GRAIN v1, une variante supplémentaire de longueur de clé 128 bits, GRAIN-128 [155], a été proposée par M. Hell, T. Johansson et W. Meier.

3 Cryptanalyse algébrique

La cryptanalyse algébrique des algorithmes à flot, entendue au sens large, désigne l'ensemble des méthodes d'attaque dans lesquelles on cherche à résoudre un système d'équations algébriques en la clé ou certains bits d'état de l'algorithme résultant de la connaissance de bits chiffrants.

Une première famille d'attaques élémentaires de ce type, appelées **attaques par linéarisation**, est connue de très longue date. Un algorithme à flot est vulnérable à une attaque par linéarisation lorsque les bits chiffrants s'expriment comme des polynômes en les n bits de la clé ou de l'état initial de degré total d suffisamment faible. Les équations fournies par chaque bit chiffrant peuvent être alors être considérées comme des équations affines en les $\sum_{i=1}^{d} \binom{n}{i}$ monômes de $\mathrm{GF}(2)[x_1, \ldots, x_n]$ de degré total au plus d. La connaissance de $\sum_{i=1}^{d} \binom{n}{i}$ bits chiffrants fournissant des équations indépendantes permet de déterminer ces monômes, et donc le secret recherché, en résolvant un système linéaire. Cette situation se rencontre par exemple dans le cas d'un générateur à combinaison de registres dont le degré de la fonction de sortie est faible.

Dans le contexte des algorithmes à flot, la dénomination de cryptanalyse algébrique est cependant le plus souvent employée selon une acception beaucoup plus restreinte que celle mentionnée plus haut. Elle désigne alors une amélioration de la méthode d'attaque par linéarisation qui en étend considérablement la portée, découverte indépendamment par N. Courtois et W. Meier [69] et J.-C. Faugère et G. Ars [91]. Cette méthode s'applique notamment aux générateurs à combinaison de registres lorsque le nombre n de variables d'entrée de la fonction booléenne $g(x_1, \ldots, x_n)$ de degré d utilisée comme fonction de sortie est suffisamment petit. Elle est fondée sur la recherche, dans les idéaux I_0 et I_1 de $\mathrm{GF}(2)[x_1, \ldots, x_n]$ engendrés par les polynômes $g(x_1, \ldots, x_n)$ et $g(x_1, \ldots, x_n)+1$, de polynômes de degré aussi faible que possible. L'idéal I_0 est constitué des polynômes annulateurs de $g+1$, i.e. des polynômes $h_0(x_1, \ldots, x_n)$ satisfaisant $h_0 \cdot (g+1) = 0$ ou ce qui revient au même dont la valeur est nulle lorsque $g = 0$; l'idéal I_1 est constitué des polynômes annulateurs de g, i.e. des polynômes $h_1(x_1, \ldots, x_n)$ satisfaisant $h_1 \cdot g = 0$. Il est facile de montrer qu'il existe dans les idéaux I_0 et I_1 au moins un polynôme h_0 (resp. h_1) de degré inférieur ou égal à $\frac{n}{2}$. On définit l'**immunité algébrique** de g comme le plus petit degré d'un annulateur. Une attaque algébrique plus efficace qu'une attaque par linéarisation peut être menée lorsqu'il existe un polynôme annulateur h_0 ou h_1 de degré d' strictement inférieur au degré d de g, ce qui d'après ce qui précède se produit nécessairement lorsque $d > \frac{n}{2}$. L'existence d'un tel polynôme annulateur fournit, pour tout bit chiffrant égal à 0 (resp. à 1) une équation de degré d'. Le système d'équations de degré d' ainsi obtenu peut être résolu par linéarisation ou par une méthode de résolution de systèmes algébriques plus élaborée. On peut par exemple recourir à l'algorithme XL proposé par N. Courtois, A. Klimov, J. Patarin et A. Shamir [68, 70] ou à l'un des deux algorithmes de calcul de bases de Gröbner les plus performants actuellement connus, F4 et F5, dus à J.-C. Faugère [89, 90].

Diverses généralisations ou améliorations de l'attaque précédente ont été proposées au cours des dernières années : extension au cas de générateurs comportant des bits de mémoire proposée par F. Armknecht et M. Krause [8] ; attaques algébriques dites rapides

contre les générateurs à combinaison de registres [66, 6, 7].[53] Les attaques algébriques rapides s'inscrivent dans une ligne de recherche plus générale de la cryptanalyse des algorithmes à flot, à savoir l'investigation d'attaques fondées sur l'étude de la **fonction augmentée** qui à n bits de l'état courant du générateur associe les m premiers bits chiffrants qu'il engendre.

Mes contributions à la cryptanalyse algébrique des algorithmes à flot sont au nombre de deux : l'analyse, en coopération avec O. Billet, d'une vulnérabilité potentielle de l'algorithme SNOW 2.0 à une attaque algébrique [36], et une généralisation des attaques algébriques à certains générateurs comportant un ou plusieurs registres à rétroaction non linéaire (NFSR) étudiée en coopération avec C. Berbain et non publiée à ce jour. Je ne décrirai ici que la première contribution. On trouvera une description du second résultat dans la thèse de C. Berbain [16].

3.1 Analyse d'une vulnérabilité potentielle de SNOW 2.0 et contribution à la spécification de SNOW 3G

SNOW 2.0, proposé en 2002 par P. Ekdahl et T. Johansson [84], est actuellement considéré comme l'un des algorithmes de chiffrement à flot adaptés à une implantation logicielle et matérielle les plus rapides et les plus sûrs. L'automate de génération de la suite chiffrante comprend un registre à décalage à rétroaction linéaire (LFSR) dont l'état courant est constitué de seize éléments du corps fini $\mathrm{GF}(2^{32})$, et une machine à états finis (FSM). L'état courant de la FSM est composé de deux mots d'état supplémentaires de 32 bits, les registres R_1 et R_2. La fonction de transition de la FSM et la fonction de sortie du générateur font intervenir deux opérations non linéaires modulo 2 : une bijection P de 32 bits vers 32 bits déduite de la boîte S et de la transformation MixColumns de l'AES, et l'addition modulo 2^{32}. L'algorithme produit 32 bits chiffrants à chaque itération.

Nous montrons dans [36] que l'emploi dans SNOW 2.0 de l'unique ingrédient non linéaire de l'AES, à savoir sa boîte S fondée sur l'inversion dans le corps $\mathrm{GF}(256)$, ne constituerait pas à lui seul une source de non linéarité suffisante pour protéger l'algorithme contre les attaques algébriques s'il n'était complété par l'emploi de l'addition modulo 2^{32}. La variante de SNOW 2.0 obtenue en remplaçant toutes les additions modulo 2^{32} par des « ou exclusif » entre mots de 32 bits est en effet vulnérable à une attaque par linéarisation permettant de reconstituer l'état initial du générateur, puis la clé. Pour cette variante affaiblie de SNOW 2.0, la connaissance de la suite chiffrante permet en efffet d'exprimer à chaque instant chaque bit de l'état des registres R_1 et R_2 comme une combinaison linéaire modulo 2 connue de l'état initial (512 bits inconnus) et des bits de la valeur d'initialisation de R_2 (32 bits inconnus supplémentaires). Les 39 équations de degré 2 liant les 8 bits d'entrée et les 8 bits sortie d'une boîte S de l'AES induisent pour chaque itération du générateur $39 \times 4 = 156$ équations quadratiques reliant les

[53] Dans les attaques algébriques rapides, l'on précalcule des équations polynomiales de faible degré total d faisant intervenir les bits de l'état initial et un petit nombre m de bits consécutifs de la suite chiffrante, puis l'on additionne t répliques judicieusement espacées de telles relations afin d'éliminer les monômes de degré d en les bits de l'état initial, et donc d'obtenir une équation de degré $e < d$ en les bits de l'état initial pouvant servir de point de départ à une résolution de moindre complexité.

544 bits inconnus. Lorsque l'attaquant dispose d'environ 1000 mots consécutifs de la suite chiffrante chiffrants (32000 bits chiffrants), le système quadratique précédent peut être résolu par linéarisation. La complexité de l'attaque est d'environ 2^{50}. Si l'on revient à l'algorithme SNOW 2.0 non modifié, l'attaque précédente n'est plus applicable. L'introduction dans les équations d'inconnues supplémentaires représentant les bits de retenue des additions modulo 2^{32} permet de représenter le problème de la cryptanalyse de SNOW 2.0 sous forme d'un système relativement creux et surdéterminé comprenant quelques milliers d'équations quadratiques. Une étude des équations quadratiques entre entrées et sorties de l'addition modulo 2^n a conduit récemment N. Courtois et B. Debraize à représenter SNOW 2.0 sous forme d'un système plus creux et surdéterminé que celui que nous décrivions dans [36]. Les techniques connues ne permettent pas de résoudre efficacement de tels systèmes.

Lorsque le groupe européen d'experts en cryptologie ETSI/SAGE où je représente France Télécom s'est vu confier la charge de spécifier un second algorithme de chiffrement pour le système UMTS, il fut décidé, avec l'accord de P. Ekdahl et T. Johansson, de prendre SNOW 2.0 comme base de la conception d'un nouvel algorithme offrant dans la mesure du possible une sécurité accrue et en particulier de meilleures garanties de résistance à la cryptanalyse algébrique. Notre analyse de [36] et les résultats de recherches qui venaient d'être menées sur la construction de fonctions satisfaisant des critères d'immunité algébrique servirent de point de départ pour concevoir un petit nombre de variations simples de SNOW 2.0 qui ont abouti à l'algorithme SNOW 3G. Les deux principales variations finalement retenues furent l'adjonction d'un troisième registre de 32 bits et d'une second permutation non linéaire sur 32 bits. Cette dernière est déduite d'une permutation polynomiale de GF(256) fondée sur un polynôme de Dickson, choix suggéré par E. Pasalic. Une attention particulière fut également apportée à la résistance de SNOW 3G contre les distingueurs par masquage linéaire : une amélioration du meilleur distingueur linéaire de SNOW 2.0 alors connu, dû à Watanabe *et al.* [201], fut découverte à cette occasion et publiée en 2006 par K. Nyberg et J. Wallen [172]. On trouvera dans [88] la spécification de SNOW 3G ainsi qu'un rapport relatif à sa conception et son évaluation.

4 Algorithmes à flot et preuves de sécurité

Il existe actuellement dans le domaine des algorithmes à flot un contraste frappant entre théorie et pratique.

D'un point de vue théorique, les notions d'algorithme à flot ne dépendant que d'une clé et de générateur pseudoaléatoire de nombres (PRNG) sont confondues. La théorie des PRNG, qui s'est constituée dans les années 80 à travers les travaux fondateurs d'A. Shamir [188], A. Yao [209], M. Blum et S. Micali [44], O. Goldreich et L. Levin [115], représente l'un des développements les plus spectaculaires et les plus féconds de l'approche réductionniste de la sécurité des objets cryptographiques qu'il est convenu de désigner d'une manière quelque peu abusive sous le nom de sécurité prouvée. La théorie des PRNG a apporté deux sortes de résultats : (1) des construc-

tions génériques, permettant de relier l'existence de PRNG à des conjectures relevant de la théorie de la complexité. Une construction due à J. Håstad, R. Impagliazzo, L. Levin et M. Luby [120] permet notamment d'établir que s'il existe des fonctions à sens unique, alors il existe des PRNG ; (2) des constructions concrètes de PRNG dont la sécurité repose de manière prouvée sur un problème conjecturé difficile issu des mathématiques discrètes [2, 43, 127, 96, 100, 47, 194], et dans le cas de l'algorithme BMGL sur l'hypotèse que la dépendance en la clé des calculs AES est à sens unique [193]. Cependant, aucune des constructions concrètes de PRNG proposées à ce jour n'est pleinement satisfaisante en termes de performances.

A l'inverse, les algorithmes à flot utilisés en pratique possèdent d'excellentes performances matérielles et logicielles, mais leur sécurité repose sur des bases fragiles. Le processus de génération de la suite chiffrante fait généralement appel à un automate dont la fonction de sortie et la fonction de transition ne sont pas à sens unique. La fonction de transition doit rendre inopérant le cumul des informations partielles sur l'état courant que la fonction de sortie laisse filtrer. C'est donc en quelque sorte par un mécanisme de « fuite en avant » que les algorithmes à flot actuels préviennent les attaques. En outre, la dépendance des algorithmes à flot en un vecteur d'initialisation modifie en profondeur les propriétés de sécurité requises : plusieurs algorithmes à flot prometteurs se sont révélés vulnérables à des attaques par resynchronisation exploitant des insuffisances du processus de chargement de clé et d'IV.

L'algorithme à flot QUAD, que j'ai mis au point en coopération avec J. Patarin et C. Berbain [20], constitue une tentative pour rapprocher le plus possible les constructions issues de la théorie des PRNG des algorithmes à flot utilisés en pratique. Notre objectif était de concevoir un algorithme à flot dont la sécurité repose de manière prouvée sur un problème conjecturé difficile sans que les performances logicielles et matérielles n'en soient pénalisées d'une manière rédhibitoire. L'idée séminale de l'algorithme, à savoir itérer la fonction multivariée associée à un système d'équations quadratiques, revient à J. Patarin. Mais la spécification complète de QUAD et l'obtention des preuves de réduction à la difficulté du problème MQ qui l'accompagnent sont le résultat d'un travail auquel nous avons tous les trois contribué.

4.1 L'algorithme à flot QUAD

QUAD repose sur la difficulé du problème MQ de résolution d'un système quadratique multivarié. Ce problème consiste, étant donné un système de m équations quadratiques en n variables à coefficients dans un corps fini $\mathrm{GF}(q)$, à trouver une solution $(x_1, \ldots, x_n)$ dans $\mathrm{GF}(q^n)$ s'il en existe. Le problème MQ est NP-dur sur $\mathrm{GF}(2)$ [99] et plus généralement quel que soit le corps fini considéré [178]. Il existe des algorithmes de résolution efficaces dans le cas d'un système fortement sous-déterminé ($m < n$) [67], ainsi que dans le cas d'un système fortement surdéterminé lorsque le nombre m d'équations indépendantes est suffisamment proche du seuil $\binom{n}{2}$ à partir duquel le système est entièrement linéarisable. Mais le problème MQ est conjecturé difficile à résoudre pour la plupart des instances lorsque n et m sont suffisamment grands et suffisamment proches l'un de l'autre. Pour une valeur fixée $t \geq 1$ du rapport

$\frac{m}{n}$, la complexité moyenne des meilleurs algorithmes de résolution connus, fondés sur les algorithmes de recherche d'une base de Gröbner F4 et F5 de J.-C. Faugère, est exponentielle [10].

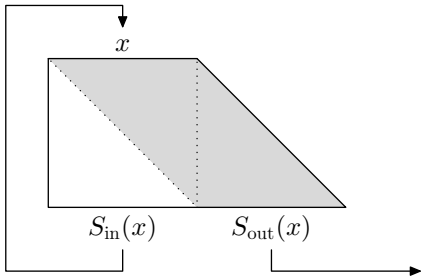


FIG. 2 – QUAD : génération de la suite chiffrante

QUAD, dont on trouvera une spécification détaillée dans [20], possède une structure extrêmement simple. Il peut être décrit comme un automate dont l'état courant est constitué d'un n-uplet $x = (x_1, \ldots, x_n)$ d'éléments d'un corps fini $\mathrm{GF}(q)$. Cet automate évolue selon deux modes de fonctionnement successifs, le chargement de clé et d'IV et la génération de la suite chiffrante. La clé K est constituée d'un n-uplet d'éléments de $\mathrm{GF}(q)$ et le vecteur d'initialisation IV est un mot de k bits. La génération de la suite chiffrante est illustrée à la figure 2 : à chaque étape, un m-uplet $S = (Q_1, \ldots, Q_m)$ fixe et public de m fonctions quadratiques en n variables à coefficients dans $\mathrm{GF}(q)$ est appliqué à l'état courant. Le n-uplet $S_{in}(x) = (Q_1(x), \ldots, Q_n(x))$ des n premières composantes de S est substitué à l'état courant, et une portion de suite chiffrante constituée du $(m-n)$-uplet $S_{out}(x) = (Q_{n+1}(x), \ldots, Q_m(x))$ est produite. La phase de chargement de clé et d'IV repose sur l'utilisation de deux n-uplets S_0 et S_1 fixes et publics de fonctions quadratiques en n variables. On peut sans inconvénient employer les n-uplets S_0 et S_1 constitués des n premières fonctions et des n fonctions suivantes de S. La clé K constitue la valeur initiale de l'état $x = (x_1, \ldots, x_n)$ de l'automate. Pour chacun des bits IV_i, $i = 1$ à k de l'IV, la transformation S_0 ou S_1 est ensuite appliquée à l'état courant, selon que $IV_i = 0$ ou $IV_i = 1$. Au lieu de commencer immédiatement la génération de la suite chiffrante à l'aide de la fonction S, on peut optionnellement la faire précéder de n tours à vide au cours desquels l'état courant est mis à jour à l'aide de S sans qu'une portion de suite chiffrante soit produite.

Nous établissons dans [20] que dans le cas binaire ($q = 2$), la sécurité de la génération pseudoaléatoire de QUAD, qui à un état initial associe la suite chiffrante qui en découle, est réductible à la difficulté du problème MQ.[54] Pour ce faire, nous nous plaçons dans le modèle de sécurité concret et non asymptotique, et relions la difficulté de distinguer la suite chiffrante produite par une instance aléatoire de QUAD à partir d'un état initial aléatoire inconnu à celle de la résolution d'une instance aléatoire du problème MQ. Plus précisément, nous prouvons dans [20] que si l'on dispose d'un algorithme permettant de

[54]Dans une publication ultérieure [18], nous avons étendu ce résultat à une réduction de la sécurité de la totalité de l'algorithme, y compris le chargement de clé et d'IV, à la difficulté du problème MQ. Ce second résultat sera présenté dans la section suivante.

distinguer la suite chiffrante associée à λ itérations d'une fonction quadratique aléatoire connue S à n variables et m équations et un état initial x en temps T avec un avantage ϵ, alors on peut en déduire un algorithme permettant de résoudre une instance aléatoire du problème MQ à m équations et n variables en un temps T' majoré par une expression polynomiale en n, λ et $\frac{1}{\epsilon}$ et linéaire en T avec une probablité $\epsilon' = \frac{\epsilon}{4\lambda^2}$. Le terme principal de l'expression de T' est $\frac{2^7 n^2 \lambda^2 (T+(2+\lambda))T_S}{\epsilon^2}$, où T_S désigne un majorant du temps de calcul d'une fonction quadratique S. On trouvera l'expression exacte de T' dans le théorème 3 de [20].

Cette réduction permet par exemple d'établir que s'il existait une attaque de complexité inférieure à 2^{80} permettant de distinguer une suite chiffrante de $L = 2^{40}$ bits produite par une instance aléatoire de QUAD de paramètres $n = 350$ et $m = 700$ avec un avantage $\epsilon = 1\%$, alors il existerait un algorithme permettant de résoudre une fraction non négligeable des instances de MQ et de complexité largement inférieure à celle du meilleur algorithme de résolution connu du problème MQ. Pour les valeurs $n = 160$ et $m = 320$ de la version de référence de QUAD proposée dans [20], nous n'obtenons pas de contradiction de ce type. Le lien très direct entre la sécurité de QUAD et la difficulté de la résolution d'un système quadratique de m équations à n inconnues corroboré par la réduction qui précède nous semble cependant justifier heuristiquement, sinon rigoureusement, ce choix de paramètres. L'existence pour la version de référence de QUAD d'attaques de complexité strictement inférieure au niveau de sécurité visé, équivalent à 2^{80} itérations de QUAD, est une question ouverte. Un article proposant des estimations de complexité de la recherche de l'état interne de QUAD par une méthode de résolution de systèmes algébriques apparentée à XL et F4 a été publié par B. Yang, O. Chen, D. Bernstein et J. Chen en 2008 [208]. Il est montré que des choix de paramètres tels que $q = 256$, $n = 20$, et $m = 40$ ne sont pas sûrs. Pour la version de référence de QUAD (de paramètres $q = 2$, $n = 160$, $m = 320$), les estimations de complexité trouvées sont nettement supérieures à 2^{80}.

La preuve de la réduction précédente se fait en trois étapes, décrites ci-dessous de manière succincte et informelle. La structure générale de la preuve présente certaines analogies avec celle des preuves d'autres générateurs pseudoaléatoires itératifs produisant plusieurs bits chiffrants à chaque itération, notamment les PRNG proposés par R. Impagliazzo et M. Naor [127] et J.B. Fischer et J. Stern [96]. Seule la seconde étape utilise pleinement les spécificités du problème MQ. Lorsque dans la description qui suit l'on parle de distingueur ou de prédicteur, l'on sous-entend qu'il s'agit d'un distingueur ou d'un prédicteur d'avantage non négligeable.

Première étape : l'on établit que s'il existait un distingueur (au sens d'un générateur de nombres) pour une instance de QUAD associée à une fonction quadratique aléatoire S, alors il existerait un distingueur entre la distribution de sortie de S et la distribution uniforme.[55] La preuve est fondée sur une technique fréquemment utilisée dans le domaine de la sécurité prouvée : elle fait intervenir des distributions de probabilité hybrides, intermédiaires entre la distribution de sortie d'un générateur fondé sur l'itération de la

[55]Ce résultat vaut également si l'on replace dans l'énoncé précédent « aléatoire » par « fixée ».

fonction S et la distribution uniforme.

Deuxième étape : l'on démontre que s'il existait un distingueur de la sortie d'une fonction quadratique aléatoire S supposée connue, alors pour toute forme linéaire L de n bits vers 1 bit il existerait un prédicteur de $L(x)$ à partir de $S(x)$, où x désigne un valeur aléatoire inconnue de n bits.

Troisième étape : il s'agit de la preuve que si pour une fonction quadratique aléatoire connue S il existait pour toute forme linéaire L un prédicteur de $L(x)$ à partir de $S(x)$, alors il existerait un algorithme de résolution du problème MQ. La preuve est fondée sur une adaptation du théorème de Goldreich et Levin [115], et s'inspire des versions simplifiées de la preuve initiale proposées par O. Goldreich et C. Rackoff [113] et M. Bellare [13] et d'une technique d'amélioration de la réduction reposant sur l'utilisation de la transformation de Walsh rapide initialement proposée par J. Håstad et M. Näslund [193].

Si l'on se restreint à la combinaison de la deuxième et de la troisième étape de la réduction précédente, on obtient un résultat général, relatif à la difficulté du problème MQ, et non lié au contexte particulier de l'analyse de la sécurité de l'algorithme QUAD. Ce résultat, énoncé ici informellement, est le suivant : si la résolution d'une instance aléatoire du problème MQ est difficile, alors la distribution de sortie d'une fonction quadratique aléatoire est indistinguable de la distribution uniforme.[56]

Des estimations de complexité d'implantations logicielles et matérielles[57] de QUAD dues pour l'essentiel à mes collègues D. Arditti, C. Berbain et O. Billet peuvent être trouvées dans les articles [17, 5]. Elles montrent que pour les paramètres de sa version de référence, QUAD est adapté aux contraintes des types d'environnement suivants : (1) implantation logicielle sur un PC ou un serveur, pour des applications pour lesquelles des débits de chiffrement de quelques Mbit/s suffisent ; (2) implantation matérielle dans des terminaux embarqués disposant d'une capacité de chiffrement hardware, comme les terminaux mobiles ; (3) implantation matérielle sur des puces électroniques à très bas coût telles que celles qui équipent les étiquettes RFID. Pour des débits de chiffrement de quelques Kbit/s adaptés à l'environnement (3), l'ordre de grandeur d'une implantation matérielle de QUAD est de seulement 3000 portes logiques.

[56] Ce résultat est utilisé dans la récente proposition de fonction de hachage MQ-HASH, fondée de même que QUAD sur la difficulté supposée de la résolution de systèmes quadratiques aléatoires [38].

[57] Dans le cas d'une exécution matérielle, les coefficients du système quadratique S doivent être régénérés à chaque exécution à l'aide d'un générateur de nombres non linéaire, et ne peuvent donc pas être tirés au sort. Il n'est pas exigé de ce générateur de nombres qu'il représente lui-même un générateur pseudoaléatoire solide. Ce choix diminue évidemment la portée des arguments de sécurité qui précèdent. Nous n'avons cependant connaissance d'aucune faiblesse particulière résultant en pratique des générateurs de nombres considérés dans [5].

4.2 Sécurité des algorithmes à flot dépendant d'un IV

Dans un article intitulé « On the Security of IV Dependent Stream Ciphers » et publié en 2007 [18], Côme Berbain et moi-même avons analysé de quelle manière la sécurité d'un algorithme à flot dépendant d'un IV peut être formalisée, et dégagé des conditions suffisantes pour qu'un algorithme à flot dépendant d'un IV soit sûr. L'application à QUAD de ces résultats généraux permet d'étendre la preuve partielle de sécurité de QUAD rappelée précédemment, dans laquelle seule la phase de génération de la suite chiffrante de l'algorithme était considérée, en lui incorporant le chargement de clé et d'IV. Nous obtenons une preuve sécurité réductionniste qui s'applique à l'ensemble de l'algorithme. Les principaux résultats de notre travail sont au nombre de trois.

1°) Nous montrons que si l'on cherche à étendre d'une manière simple et naturelle la notion de PRNG à un algorithme dépendant d'un vecteur d'initialisation[58], on aboutit à la conclusion qu'une condition nécessaire et suffisante pour qu'un algorithme à flot dépendant d'un IV soit sûr est que la famille de fonctions paramétrées par la clé qui à l'IV associent la suite chiffrante soit une famille pseudoaléatoire de fonctions (PRF). Cette caractérisation avait déjà été proposée par d'autres auteurs [182, 117].

2°) Nous prouvons que dans le cas le plus souvent rencontré en pratique où l'algorithme à flot est la composition (1) d'une fonction d'établissement de clé et d'IV f_K paramétrée par une clé $K \in \mathcal{K}$ qui à l'IV associe l'état initial et (2) d'une fonction fixe g de génération de la suite chiffrante, non paramétrée par la clé et qui à un état initial associe une suite chiffrante, une condition suffisante[59] pour que l'algorithme soit sûr au sens de la caractérisation précédente est que f_K soit une PRF et que g soit un PRNG. Ce résultat découle d'un théorème de composition très simple : la composée $(g_K) = (g \circ f_K)$ d'une PRF et d'un PRNG dont les tailles de sortie et d'entrée coïncident est une PRF, sous certaines conditions sur les bornes d'indistinguabilité de la PRF et du PRNG que l'on compose entre eux.[60]

3°) Nous montrons enfin que si l'on dispose d'une fonction fixe g de n bits vers $2n$ bits

[58] Pour ce faire, nous proposons d'assimiler, pour une clé fixée, la famille des suites chiffrantes associées aux différentes valeurs de l'IV à une suite unique, de longueur exponentielle, dans laquelle un adversaire peut accéder directement à la portion de la suite chiffrante correspondant à un IV de son choix.

[59] sous certaines conditions sur les bornes d'indistinguabilité de (f_K) et g

[60] Bien que les conditions précédentes soient suffisantes et non nécessaires et que l'on ne puisse donc pas affirmer que la famille de fonctions d'établissement de clé et d'IV (f_K) doit être une PRF, l'argument suivant montre qu'en pratique, pour un algorithme à flot rapide, la complexité de la fonction f_K ne peut être nettement inférieure à celle que nécessite une PRF de mêmes tailles d'entrée et de sortie que f_K : pour que l'algorithme soit sûr, la fonction paramétrée par K qui à l'IV associe les n premiers bits chiffrants doit être une PRF. Or elle est la composée de f_K et du calcul par le générateur des n premiers bits chiffrants à partir de l'état initial. Pour les algorithmes à flot rapides rencontrés en pratique, la complexité de ce second calcul est nettement inférieure à la complexité minimale des fonctions actuellement conjecturées être des PRF et par conséquent, la complexité de la fonction f_K ne peut pas être nettement inférieure à cette complexité minimale.

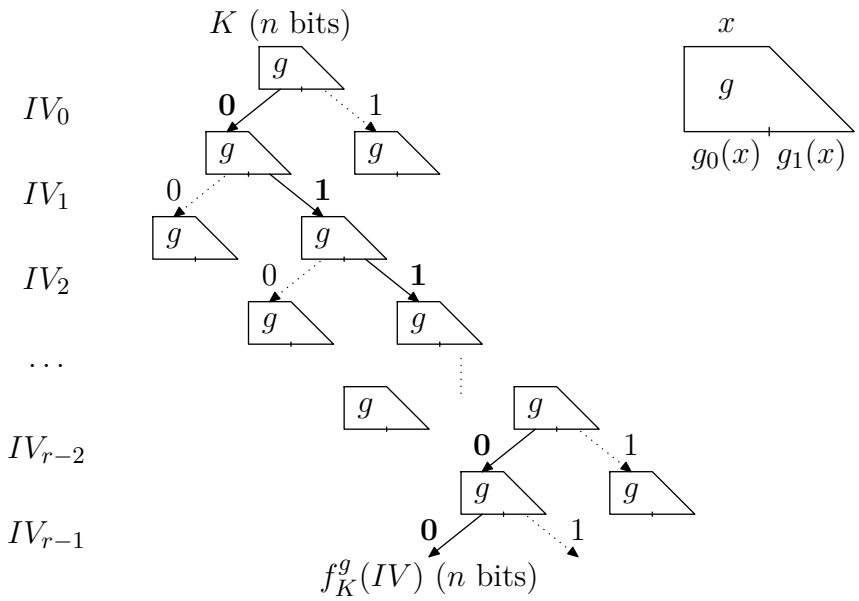


FIG. 3 – Construction arborescente

qui soit un PRNG et donc[61] d'un PRNG de n bits vers $L > n$ obtenu par itération de g, alors l'on peut[62] construire à partir de g un algorithme à flot dépendant d'un IV qui soit sûr. Le chargement de clé et d'IV de cet algorithme est obtenu en appliquant au PRNG g une construction arborescente (tree-based construction) proposée en 1986 par O. Goldreich, S. Goldwasser et S. Micali [114] afin de démontrer que s'il existe un générateur de nombres sûr (PRNG), alors il existe un générateur de fonctions sûr (PRF). Cette construction est illustrée à la figure 3. Elle permet de déduire de la fonction de n bits vers $2n$ bits $g = (g_0, g_1)$, où g_0 et g_1 sont deux fonctions de n bits vers n bits, une famille $F = (f_K^g(.))$ de fonctions de r bits vers n bits indexée par un paramètre $K \in \{0,1\}^n$. Nous instancions cette construction de la manière suivante : les r bits successifs de l'IV déterminent un parcours à travers un arbre binaire reliant des mots de n bits et dont la racine est constituée de la valeur de la clé. A chaque étape, le mot courant de n bits x est remplacé par les n bits de gauche ou de droite de $g(x)$ selon que la valeur du bit courant de l'IV est égal à zéro ou un. Le mot de n bits obtenu à l'issue de ce traitement constitue l'état initial du générateur. En transposant la preuve de O. Goldreich *et al.* [114] qui est donnée dans le modèle de sécurité asymptotique, au modèle de la sécurité concrète, nous établissons que le chargement de clé et d'IV obtenu est une PRF. L'application du théorème de composition précédent à la PRF déduite de g ainsi construite et au PRNG associé à l'itération de g permet ensuite de montrer que l'algorithme à flot dépendant d'un IV résultant de leur composition est encore une PRF, et donc qu'il est sûr.

Le résultat précédent est applicable à l'algorithme à flot QUAD, dont le chargement

[61] sous certaines conditions sur les bornes d'instinguabilité de g

[62] ici encore, sous certaines conditions sur les bornes d'instinguabilité de g

de clé et d'IV repose sur la construction arborescente de la figure 3. Il permet d'étendre l'argument de sécurité relatif à la génération de la suite chiffrante au chargement de clé et d'IV, et donc d'établir que pour des valeurs suffisantes des paramètres de l'algorithme, QUAD considéré dans sa globalité est sûr s'il n'existe pas d'algorithme de résolution du problème MQ plus performant que les méthodes les plus efficaces actuellement connues. QUAD est à notre connaissance la première proposition d'algorithme à flot dépendant d'un IV dont la sécurité soit réductible à un problème mathématique connu et conjecturé difficile.

Bibliographie

[1] eSTREAM: ECRYPT Stream Cipher Project, IST-2002-507932. Available from http://www.ecrypt.eu.org/stream/, 2005.

[2] Werner Alexi, Benny Chor, Oded Goldreich, and Claus-Peter Schnorr. RSA/Rabin Bits are 1/2 + 1/poly(log N) Secure. In *Foundations of Cryptography FOCS 1984*, pages 449–457, 1984.

[3] Stéphanie Alt and Reynald Lercier. How to Make a Traceable Blockcipher Stealthy. Private communication, 2004.

[4] Kazumaro Aoki and Kazuo Ohta. Strict Evaluation of the Maximum Average of Differential Probability and the Maximum Average of Linear Probability (Special Section on Cryptography and Information Security). *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 80(1):2–8, 1997.

[5] David Arditti, Côme Berbain, Olivier Billet, and Henri Gilbert. Compact FPGA implementations of QUAD. In Feng Bao and Steven Miller, editors, *Proceedings of the 2007 ACM Symposium on Information, Computer and Communications Security - ASIACCS 2007*, pages 347–349. ACM, 2007.

[6] Frederik Armknecht. Improving Fast Algebraic Attacks. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption - FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 65–82. Springer, 2004.

[7] Frederik Armknecht and Gwénolé Ars. Introducing a New Variant of Fast Algebraic Attacks and Minimizing Their Successive Data Complexity. In Ed Dawson and Serge Vaudenay, editors, *Progress in Cryptology - Mycrypt 2005*, volume 3715 of *Lecture Notes in Computer Science*, pages 16–32. Springer, 2005.

[8] Frederik Armknecht and Matthias Krause. Algebraic Attacks on Combiners with Memory. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 162–175. Springer, 2003.

[9] Steve Babbage, Carlos Cid, Norbert Pramstaller, and Håvard Raddum. An Analysis of the Hermes8 Stream Ciphers. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *Information Security and Privacy - ACISP 2007*, volume 4586 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2007.

[10] Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie.* PhD thesis, Université Paris VI, 2004.

[11] Thierry Baritaud, Henri Gilbert, and Marc Girault. FFT Hashing is not Collision-free. In *EUROCRYPT*, pages 35–44, 1992.

[12] Olivier Baudron, Henri Gilbert, Louis Granboulan, Helena Handschuh, Robert Harly, Phong Nguyen Antoine Joux, Fabrice Noilhan, David Pointcheval, Thomas Pornin, Guillaume Poupard, Jacques Stern, and Serge Vaudenay. Decorrelated Fast Cipher: an AES candidate. Proceedings of the Second Advanced Encryption Standard Candidate Conference, National Institute of Standards and Technology, August 1998.

[13] Mihir Bellare. The Goldreich-Levin Theorem. Available from: `http://www-cse.ucsd.edu/users/mihir/courses.html`, 1999.

[14] Mihir Bellare, Anand Desai, Eron Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *FOCS*, pages 394–403, 1997.

[15] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of Cipher Block Chaining. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 341–358. Springer, 1994.

[16] Côme Berbain. Analyse et conception d'algorithmes de chiffrement à flot. Thèse de doctorat, Université Paris 7, octobre 2007.

[17] Côme Berbain, Olivier Billet, and Henri Gilbert. Efficient Implementations of Multivariate Quadratic Systems. In Eli Biham and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2006*, volume 4356 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 2007.

[18] Côme Berbain and Henri Gilbert. On the Security of IV Dependent Stream Ciphers. In Alex Biryukov, editor, *Fast Software Encryption - FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 254–273. Springer, 2007.

[19] Côme Berbain, Henri Gilbert, and Alexander Maximov. Cryptanalysis of Grain. In Matthew J. B. Robshaw, editor, *Fast Software Encryption - FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 2006.

[20] Côme Berbain, Henri Gilbert, and Jacques Patarin. QUAD: A Practical Stream Cipher with Provable Security. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 109–128. Springer, 2006.

[21] Eli Biham. New types of Cryptoanalytic Attacks Using Related Keys (Extended Abstract). In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 398–409. Springer, 1994.

[22] Eli Biham, Ross J. Anderson, and Lars R. Knudsen. Serpent: A New Block Cipher Proposal. In Serge Vaudenay, editor, *Fast Software Encryption - FSE '98*, volume 1372 of *Lecture Notes in Computer Science*, pages 222–238. Springer, 1998. Serpent-1, the variant that became one of the five AES finalists, is available on http://www.cs.technion.ac.il/ biham/Reports/Serpent/.

[23] Eli Biham and Alex Biryukov. An Improvement of Davies' Attack on DES. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 461–467. Springer, 1995.

[24] Eli Biham and Alex Biryukov. An improved Davies' attack on DES. *Journal of Cryptology*, 10(3):195–205, 1997.

[25] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 12–23. Springer, 1999.

[26] Eli Biham, Alex Biryukov, and Adi Shamir. Miss in the Middle Attacks on IDEA and Khufu. In Lars R. Knudsen, editor, *Fast Software Encryption - FSE '99*, volume 1636 of *Lecture Notes in Computer Science*, pages 124–138. Springer, 1999.

[27] Eli Biham, Orr Dunkelman, and Nathan Keller. The Rectangle Attack - Rectangling the Serpent. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 340–357. Springer, 2001.

[28] Eli Biham, Orr Dunkelman, and Nathan Keller. New Results on Boomerang and Rectangle Attacks. In *FSE*, pages 1–16, 2002.

[29] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-Key Boomerang and Rectangle Attacks. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 507–525. Springer, 2005.

[30] Eli Biham, Orr Dunkelman, and Nathan Keller. Related-Key Impossible Differential Attacks on 8-Round AES-192. In David Pointcheval, editor, *The Cryptographers' Track at the RSA Conference - CT-RSA 2006*, volume 3860 of *Lecture Notes in Computer Science*, pages 21–33. Springer, 2006.

[31] Eli Biham and Nathan Keller. Cryptanalysis of Reduced Variants of Rijndael. In *AES Candidate Conference*, 2000.

[32] Eli Biham and Adi Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[33] Eli Biham and Adi Shamir. Differential Cryptanalysis of Feal and N-Hash. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 1991.

[34] Eli Biham and Adi Shamir. *Differential Cryptanalysis of the Data Encryption Standard*. 1993.

[35] Olivier Billet and Henri Gilbert. A Traceable Block Cipher. In Chi-Sung Laih, editor, *Advances in Cryptology - ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 331–346. Springer, 2003.

[36] Olivier Billet and Henri Gilbert. Resistance of SNOW 2.0 Against Algebraic Attacks. In Alfred Menezes, editor, *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA*, volume 3376 of *Lecture Notes in Computer Science*, pages 19–28. Springer, 2005.

[37] Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi. Cryptanalysis of a White Box AES Implementation. In Helena Handschuh and M. Anwar Hasan, editors, *Selected Areas in Cryptography - SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, pages 227–240. Springer, 2004.

[38] Olivier Billet, Matthew J. B. Robshaw, and Thomas Peyrin. On Building Hash Functions from Multivariate Quadratic Equations. In Josef Pieprzyk, Hossein Ghodosi, and Ed Dawson, editors, *Australasian Conference on Information Security and Privacy - ACISP 2007*, volume 4586 of *Lecture Notes in Computer Science*, pages 82–95. Springer, 2007.

[39] Alex Biryukov. The Boomerang Attack on 5 and 6-Round Reduced AES. In Hans Dobbertin, Vincent Rijmen, and Aleksandra Sowa, editors, *4th AES Conference - AES 2004*, volume 3373 of *Lecture Notes in Computer Science*, pages 11–15. Springer, 2005.

[40] Alex Biryukov and Christophe De Cannière. Block Ciphers and Systems of Quadratic Equations. In Thomas Johansson, editor, *Fast Software Encryption - FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 274–289. Springer, 2003.

[41] Alex Biryukov and Adi Shamir. Structural Cryptanalysis of SASAS. In Birgit Pfitzmann, editor, *Advances in Cryptology - EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 394–405. Springer, 2001.

[42] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant Learning, the Parity Problem, and the Statistical Query Model. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 435–440. ACM, 2000.

[43] Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, May 1986.

[44] Manuel Blum and Silvio Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. Comput.*, 13(4):850–864, 1984.

[45] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[46] Dan Boneh and Matthew K. Franklin. An Efficient Public Key Traitor Tracing Scheme. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 338–353. Springer, 1999.

[47] Dan Boneh, Shai Halevi, and Nick Howgrave-Graham. The Modular Inversion Hidden Number Problem. In Colin Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 36–51. Springer, 2001.

[48] Julien Bringer and Hervé Chabanne. Trusted-HB: a low-cost version of HB+ secure against Man-in-The-Middle attacks. *CoRR*, abs/0802.0603, 2008.

[49] Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. Perturbing and Protecting a Traceable Block Cipher. In Herbert Leitold and Evangelos P. Markatos, editors, *Communications and Multimedia Security - CMS 2006*, volume 4237 of *Lecture Notes in Computer Science*, pages 109–119. Springer, 2006.

[50] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[51] Christophe De Cannière, Özgül Küçük, and Bart Preneel. Analysis of grain's initialization algorithm. In Serge Vaudenay, editor, *Progress in Cryptology - AFRICACRYPT 2008*, volume 5023 of *Lecture Notes in Computer Science*, pages 276–289. Springer, 2008.

[52] Christophe De Cannière and Bart Preneel. Trivium: Specifications. eSTREAM, ECRYPT Stream Cipher Project, 2005.

[53] Anne Canteaut. Analyse et conception de chiffrements à clef secrète. Mémoire d'habilitation à diriger des recherches, Université Paris 6, septembre 2006.

[54] Anne Canteaut and Michaël Trabbia. Improved Fast Correlation Attacks Using Parity-Check Equations of Weight 4 and 5. In Bart Preneel, editor, *Advances*

in Cryptology - EUROCRYPT 2000, volume 1807 of *Lecture Notes in Computer Science*, pages 573–588. Springer, 2000.

[55] Claude Carlet. Boolean Functions for Cryptography and Error Correcting Codes, 2006.

[56] Vladimir V. Chepyzhov, Thomas Johansson, and Ben Smeets. A Simple Algorithm for Fast Correlation Attacks on Stream Ciphers. In Bruce Schneier, editor, *Fast Software Encryption - FSE 2000*, number 1978, pages 181–195, 2000.

[57] Benny Chor, Amos Fiat, and Moni Naor. Tracing Traitors. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270. Springer, 1994.

[58] Philippe Chose, Antoine Joux, and Michel Mitton. Fast Correlation Attacks: an Algorithmic Point of View. In Lars R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 209–221. Springer, 2002.

[59] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. A White-Box DES Implementation for DRM Applications. In Joan Feigenbaum, editor, *Security and Privacy in Digital Rights Management, ACM CCS Workshop, DRM 2002*, volume 2696 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2003.

[60] Stanley Chow, Philip A. Eisen, Harold Johnson, and Paul C. van Oorschot. White-Box Cryptography and an AES Implementation. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography - SAC 2002*, volume 2595 of *Lecture Notes in Computer Science*, pages 250–270. Springer, 2003.

[61] Carlos Cid, Henri Gilbert, and Thomas Johansson. Cryptanalysis of Pomaranch. In *IEE Proceedings Information Security*, volume 153, pages 51–53, 2006.

[62] Carlos Cid and Gaëtan Leurent. An Analysis of the XSL Algorithm. In Bimal K. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 333–352. Springer, 2005.

[63] Don Coppersmith, Shai Halevi, and Charanjit S. Jutla. Cryptanalysis of Stream Ciphers with Linear Masking. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 515–532. Springer, 2002.

[64] Don Coppersmith, Hugo Krawczyk, and Yishay Mansour. The Shrinking Generator. In Douglas Robert Stinson, editor, *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 22–39. Springer, 1993.

[65] Nicolas Courtois. The Security of Hidden Field Equations (HFE). In David Naccache, editor, *Topics in Cryptology - CT-RSA 2001*, volume 2020 of *Lecture Notes in Computer Science*, pages 266–281. Springer, 2001.

[66] Nicolas Courtois. Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. Springer, 2003.

[67] Nicolas Courtois, Louis Goubin, Willi Meier, and Jean-Daniel Tacier. Solving Underdefined Systems of Multivariate Quadratic Equations. In David Naccache and Pascal Paillier, editors, *Public Key Cryptography - PKC 2002*, volume 2274 of *Lecture Notes in Computer Science*, pages 211–227. Springer, 2002.

[68] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In Bart Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer, 2000.

[69] Nicolas Courtois and Willi Meier. Algebraic Attacks on Stream Ciphers with Linear Feedback. In Eli Biham, editor, *Advances in Cryptology - EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer, 2003.

[70] Nicolas Courtois and Jacques Patarin. About the XL Algorithm over $GF(2)$. In Marc Joye, editor, *Topics in Cryptology - CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 141–157. Springer, 2003.

[71] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. Cryptology ePrint Archive, Report 2002/044, 2002. http://eprint.iacr.org/.

[72] Nicolas Courtois and Josef Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Yuliang Zheng, editor, *Advances in Cryptology - ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer, 2002.

[73] Joan Daemen, Antoon Bosselaers, René Govaerts, and Joos Vandewalle. Collisions for Schnorr's Hash Function FFT-Hash Presented at Crypto '91. In Hideki Imai, Ronald L. Rivest, and Tsutomu Matsumoto, editors, *Advances in Cryptology - ASIACRYPT '91*, volume 739 of *Lecture Notes in Computer Science*, pages 477–480. Springer, 1993.

[74] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The Block Cipher Square. In Eli Biham, editor, *Fast Software Encryption - FSE '97*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer, 1997.

[75] Joan Daemen and Vincent Rijmen. AES Proposal: Rijndael. The First Advanced Encryption Standard Candidate Conference, N.I.S.T., 1998.

[76] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[77] Daniel Augot, Alex Biryukov, Anne Canteaut, Nicolas Courtois, Christophe De Cannière, Cédric Lauradoux, Matthew Parker, Bart Preneel, Matt Robshaw, and Yannick Seurin.

[78] Donald W. Davies. Investigation of a Potential Weakness in the DES Algorithm. Private communication, 1987.

[79] Donald W. Davies and Sean Murphy. Pairs and triplets of DES S-Boxes. *Journal of Cryptology*, 8:1–25, 1995.

[80] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22:644–654, 1976.

[81] Matthew W. Dodd. Applications of the Discrete Fourier Transform in Information Theory and Cryptology. PhD thesis, University of London, 2003.

[82] Vivien Dubois, Pierre-Alain Fouque, Adi Shamir, and Jacques Stern. Practical Cryptanalysis of SFLASH. In Alfred Menezes, editor, *Advances in Cryptology - CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*. Springer, 2007.

[83] D.N. Duc and K. Kim. Securing HB Against GRS Man-in-the-Middle Attack. In *Institute of Electronics, Information and Communication Engineers, Symposium on Cryptography and Information Security*, Jan. 23–26, 2007.

[84] Patrik Ekdahl and Thomas Johansson. A New Version of the Stream Cipher SNOW. In Kaisa Nyberg and Howard M. Heys, editors, *Proceedings of Selected Areas in Cryptography – SAC'02*, number 2595 in Lecture Notes in Computer Science, 2002.

[85] Håkan Englund, Martin Hell, and Thomas Johansson. Two General Attacks on Pomaranch-Like Keystream Generators. In Alex Biryukov, editor, *Fast Software Encryption- FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 274–289. Springer, 2007.

[86] ETSI/SAGE. 3GPP TS 35.201 and 35.202: 3G Security; Specification of the 3GPP confidentiality and integrity algorithms; Document 1: f8 and f9 specification; Document 2: Kasumi specification. Available from http://www.3gpp.org/TB/Other/algorithms.htm.

[87] ETSI/SAGE. 3GPP TS 35.205 and 35.206: 3G Security; Specification of the MILENAGE algorithm set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5. Available from http://www.3gpp.org/ftp/Specs/html-info/35-series.htm.

[88] ETSI/SAGE. Universal Mobile Telecommunications System (UMTS); Specification of the 3GPP Confidentiality and Security Algorithms UEA2 and UIA2; Document 1: UEA2 and UIA2 specification; Document 2: SNOW 3G specification; Document 5: Document 5: design and evaluation report.

[89] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases (F_4). *Journal of Pure and Applied Algebra*, 139, 1-3:61–88, 1999.

[90] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In Teo Mora, editor, *Proceedings of International Symposium on Symbolic and Algebraic Computation – ISSAC'02*, pages 75–83. ACM Press, 2002. Also available at http://www-calfor.lip6.fr/ jcf/Papers/@papers/f5/pdf.

[91] Jean-Charles Faugère and Gwénolé Ars. An Algebraic Cryptanalysis of Nonlinear Filter Generators using Gröbner bases. Research Report, INRIA, RR-4739, February 2003.

[92] Jean-Charles Faugère and Antoine Joux. Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2003.

[93] Jean-Charles Faugère and Ludovic Perret. Polynomial Equivalence Problems: Algorithmic and Theoretical Aspects. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 30–47. Springer, 2006.

[94] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer. Strong Authentication for RFID Systems Using the AES Algorithm. In Marc Joye and Jean-Jacques Quisquater, editors, *Cryptographic Hardware and Embedded Systems - CHES 2004*, volume 3156 of *Lecture Notes in Computer Science*, pages 357–370. Springer, 2004.

[95] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. Improved Cryptanalysis of Rijndael. In Bruce Schneier, editor, *Fast Software Encryption- FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 213–230. Springer, 2001.

[96] Jean-Bernard Fischer and Jacques Stern. An Efficient Pseudo-Random Generator Provably as Secure as Syndrome Decoding. In Ueli M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 245–255. Springer, 1996.

[97] Scott R. Fluhrer. Cryptanalysis of the SEAL 3.0 Pseudorandom Function Family. In Mitsuru Matsui, editor, *Fast Software Encryption - FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 135–143. Springer, 2002.

[98] Pierre-Alain Fouque, Gilles Macario-Rat, and Jacques Stern. Key Recovery on Hidden Monomial Multivariate Schemes. In Nigel Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, Lecture Notes in Computer Science. Springer. To be published in 2008.

[99] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, chapter 7.2 Algebraic Equations over GF(2). W H Freeman & Co, 1979.

[100] Rosario Gennaro. An Improved Pseudo-random Generator Based on Discrete Log. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 469–481. Springer, 2000.

[101] Henri Gilbert. Cryptanalyse statistique des algorithmes de chiffrement. Thèse de doctorat, Université Paris-Sud, 1997.

[102] Henri Gilbert. Techniques for Low Cost Authentication and Message Authentication. In Jean-Jacques Quisquater and Bruce Schneier, editors, *Smart Card Research and Applications - CARDIS '98*, volume 1820 of *Lecture Notes in Computer Science*, pages 183–192. Springer, 2000.

[103] Henri Gilbert. The Security of "One-Block-to-Many" Modes of Operation. In Thomas Johansson, editor, *Fast Software Encryption - FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 376–395. Springer, 2003.

[104] Henri Gilbert and Pascal Audoux. Improved Fast Correlation Attacks on Stream Ciphers using FFT Techniques. Personal communication, 2000.

[105] Henri Gilbert and Guy Chassé. A Statistical Attack of the FEAL-8 Cryptosystem. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 22–33. Springer, 1991.

[106] Henri Gilbert and Pascal Chauvaud. A Chosen Plaintext Attack of the 16-round Khufu Cryptosystem. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 359–368. Springer, 1994.

[107] Henri Gilbert, Marc Girault, Philippe Hoogvorst, Fabrice Noilhan, Thomas Pornin, Guillaume Poupard, Jacques Stern, and Serge Vaudenay. Decorrelated Fast Cipher: an AES candidate. Proceedings of the First Advanced Encryption Standard Candidate Conference, National Institute of Standards and Technology, August 1998.

[108] Henri Gilbert, Helena Handschuh, Antoine Joux, and Serge Vaudenay. A Statistical Attack on RC6. In Bruce Schneier, editor, *Fast Software Encryption - FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 64–74. Springer, 2001.

[109] Henri Gilbert and Marine Minier. A Collision Attack on 7 Rounds of Rijndael. In *AES Candidate Conference*, pages 230–241, 2000.

[110] Henri Gilbert and Marine Minier. New Results on the Pseudorandomness of Some Blockcipher Constructions. In Mitsuru Matsui, editor, *Fast Software Encryption - FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 248–266. Springer, 2002.

[111] Henri Gilbert, Matthew J.B. Robshaw, and Yannick Seurin. Good Variants of HB^+ are Hard to Find. In Gene Tsudik, editor, *Financial Cryptography - FC 2008*, Lecture Notes in Computer Science. Springer. To be published in 2008.

[112] Henri Gilbert, Matthew J.B. Robshaw, and Yannick Seurin. $HB^\sharp$: Increasing the Security and Efficiency of HB^+. In Nigel Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, Lecture Notes in Computer Science. Springer. To be published in 2008.

[113] Oded Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.

[114] Oded Goldreich, Shafi Goldwasser, and Silvio. How to construct random functions. *Journal of the ACM*, 33(4):792–807, 1986.

[115] Oded Goldreich and Leonid A. Levin. A hard core predicate for any one way function. In *Proceedings of Symposium on Theory of Computing – STOC'89*, pages 25–32. ACM Press, 1989.

[116] Louis Goubin, Jean-Michel Masereel, and Michaël Quisquater. Cryptanalysis of White Box DES Implementations. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography - SAC 2007*, volume 4876 of *Lecture Notes in Computer Science*, pages 278–295. Springer, 2007.

[117] Shai Halevi, Don Coppersmith, and Charanjit S. Jutla. Scream: A Software-Efficient Stream Cipher. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encrytion - FSE 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 195–209. Springer, 2002.

[118] Helena Handschuh and Henri Gilbert. χ^2 Cryptanalysis of the SEAL Encryption Algorithm. In Eli Biham, editor, *Fast Software Encryption - FSE '97*, volume 1267 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 1997.

[119] C. Harpes and J.L. Massey. Partitioning cryptanalysis. In Eli Biham, editor, *Fast Software Encryption – FSE'97*, number 1267, pages 13–27, 1997.

[120] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[121] Martin Hell and Thomas Johansson. On the Problem of Finding Linear Approximations and Cryptanalysis of Pomaranch Version 2. In Eli Biham and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2006*, volume 4356 of *Lecture Notes in Computer Science*, pages 220–233. Springer, 2007.

[122] Martin Hell, Thomas Johansson, and Willi Meier. Grain - A Stream Cipher for Constrained Environments. eSTREAM, ECRYPT Stream Cipher Project, 2005.

[123] Martin Hell, Thomas Johansson, and Willi Meier. Grain: a stream cipher for constrained environments. *IJWMC*, 2(1):86–93, 2007.

[124] Henri Gilbert, Matthew J.B. Robshaw, and Hervé Sibert. An Active Attack Against HB^+, A Provably Secure Lightweight Authentication Protocol. IEE Electronics Letters, volume 41, number 21, pp. 1169–1170, 2005.

[125] Seokhie Hong, Jongsung Kim, Sangjin Lee, and Bart Preneel. Related-Key Rectangle Attacks on Reduced Versions of SHACAL-1 and AES-192. In Henri Gilbert and Helena Handschuh, editors, *Fast Software Encryption - FSE 2005*, volume 3557 of *Lecture Notes in Computer Science*, pages 368–383. Springer, 2005.

[126] Huseyin Demirci and Ali Aydin Selçuk. A Meet-in-the-Middle Attack on 8-Round AES. In *Fast Software Encryption - FSE 2008*, Lecture Notes in Computer Science. Springer, 2008. To appear.

[127] Russell Impagliazzo and Moni Naor. Efficient Cryptographic Schemes Provably as Secure as Subset Sum. *Journal of Cryptology*, 9(4):199–216, 1996.

[128] Tetsu Iwata, Tomonobu Yoshino, Tomohiro Yuasa, and Kaoru Kurosawa. Round Security and Super-Pseudorandomness of MISTY Type Structure. In Mitsuru Matsui, editor, *Fast Software Encryption - FSE 2001*, volume 2355 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2002.

[129] Goce Jakimoski and Yvo Desmedt. Related-Key Differential Cryptanalysis of 192-bit Key AES Variants. In Mitsuru Matsui and Robert J. Zuccherato, editors, *Selected Areas in Cryptography - SAC 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 208–221. Springer, 2004.

[130] Thomas Jakobsen and Lars R. Knudsen. The Interpolation Attack on Block Ciphers. In Eli Biham, editor, *Fast Software Encryption - FSE'97*, number 1267, pages 28–40, 1997.

[131] Cees J.A. Jansen, Tor Helleseth, and Alexander Kholosha. Cascade Jump Controlled Sequence Generator (CJCSG). In SKEW, Workshop Record, ECRYPT Network of Excellence in Cryptology, 2005.

[132] Thomas Johansson and Fredrik Jönsson. Fast Correlation Attacks Based on Turbo Code Techniques. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, number 1666 in Lecture Notes in Computer Science, pages 181–197. Springer, 1999.

[133] Thomas Johansson and Fredrik Jönsson. Improved Fast Correlation Attacks on Stream Ciphers via Convolution Codes. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT'99*, number 1592 in Lectures Notes in Computer Science, pages 347–362. Springer, 1999.

[134] Thomas Johansson and Fredrik Jönsson. Fast Correlation Attacks through Reconstruction of Linear Polynomials. In Mihir Bellare, editor, *Advances in Cryptology - CRYPTO 2000*, number 1880 in Lecture Notes in Computer Science, pages 300–315. Springer, 2000.

[135] Ari Juels and Stephen A. Weis. Authenticating Pervasive Devices with Human Protocols. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 293–308. Springer, 2005.

[136] Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. HB^{++}: A Lightweight Authentication Protocol Secure Against Some Attacks. In P. Georgiadis, J. Lopez, S. Gritzalis, and G. Marias, editors, *Proceedings of SecPerU 2006*, pp. 28-33, IEEE Computer Society Press, 2006.

[137] Jonathan Katz and Ji Sun Shin. Parallel and Concurrent Security of the HB and HB^{+} Protocols. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2006.

[138] John Kelsey, Tadayoshi Kohno, and Bruce Schneier. Amplified Boomerang Attacks Against Reduced-Round MARS and Serpent. In Bruce Schneier, editor, *Fast Software Encryption - FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 75–93. Springer, 2001.

[139] Shahram Khazaei. Cryptanalysis of Pomaranch. Posted on eSTREAM webpage www.ecrypt.eu.org/stream/, 2005.

[140] Shahram Khazaei, Mehdi Hassanzadeh, and Mohammad Kiaei. Distinguishing Attack on Grain. posted on eSTREAM webpage www.ecrypt.eu.org/stream/, 2005.

[141] Jongsung Kim, Seokhie Hong, and Bart Preneel. Related-Key Rectangle Attacks on Reduced AES-192 and AES-256. In Alex Biryukov, editor, *Fast Software Encryption - FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 225–241. Springer, 2007.

[142] Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 19–30. Springer, 1999.

[143] Lars R. Knudsen. Cryptanalysis of LOKI91. In Jennifer Seberry and Yuliang Zheng, editors, *Advances in Cryptology - ASIACRYPT '92*, volume 718 of *Lecture Notes in Computer Science*, pages 196–208. Springer, 1993.

[144] Lars R. Knudsen and Vincent Rijmen. Known-Key Distinguishers for Some Block Ciphers. In Kaoru Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 315–324. Springer, 2007.

[145] Lars Ramkilde Knudsen and Willi Meier. Correlations in RC6 with a Reduced Number of Rounds. In Bruce Schneier, editor, *Fast Software Encryption - FSE 2000*, number 1978, pages 94–108, 2000.

[146] Lars Ramkilde Knudsen and David Wagner. Integral Cryptanalysis (Extended Abstract). In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption - FSE 2002*, number 2365, pages 112–127, 2002.

[147] Hugo Krawczyk. LFSR-based Hashing and Authentication. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 129–139. Springer, 1994.

[148] Hugo Krawczyk. New Hash Functions For Message Authentication. In Louis C. Guillou and Jean-Jacques Quisquater, editors, *Advances in Cryptology - EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 301–310. Springer, 1995.

[149] Xuejia Lai. Higher Order Derivatives and Differential Cryptanalysis. In *Proceedings of "Symposium on Communication, Coding and Cryptography", in honor of James L. Massey on the occasion of his 60th birthday, Feb.10-13, 1994, Monte-Verita, Ascona, Switzerland, 1994.*

[150] Xuejia Lai and James L. Massey. Markov Ciphers and Differentail Cryptanalysis. In Donald W. Davies, editor, *Advances in Cryptology - EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer, 1991.

[151] Susan K. Langford and Martin E. Hellman. Differential-Linear Cryptanalysis. In Yvo Desmedt, editor, *Proceedings of CRYPTO '94*, number 839 in Lecture Notes in Computer Science, pages 17–25. Springer, 1994.

[152] Gregor Leander, Christof Paar, Axel Poschmann, and Kai Schramm. New Lightweight DES Variants. In Alex Biryukov, editor, *Fast Software Encryption - FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 196–210. Springer, 2007.

[153] Éric Levieil and Pierre-Alain Fouque. An Improved LPN Algorithm. In Roberto De Prisco and Moti Yung, editors, *Security and Cryptography for Networks - SCN 2006*, volume 4116 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2006.

[154] Michael Luby and Charles Rackoff. How to Construct Pseudorandom Permutations from Pseudorandom Functions. *SIAM Journal on Computing*, 17(2):373–386, April 1988.

[155] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. A Stream Cipher Proposal: Grain-128. Information Theory, 2006 IEEE International Symposium, 2006.

[156] Mitsuru Matsui. Linear Cryptanalysis Method for DES Cipher. In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer Verlag, 1994.

[157] Mitsuru Matsui. The First Experimental Cryptanalysis of the Data Encryption Standard. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 1994.

[158] Mitsuru Matsui. Block Encryption Algorithm MISTY. In Eli Biham, editor, *Fast Software Encryption - FSE'97*, number 1267 in Lecture Notes in Computer Science, pages 64–74. Springer, 1997.

[159] Tsutomu Matsumoto and Hideki Imai. Public Quadratic Polynominal-Tuples for Efficient Signature-Verification and Message-Encryption. In Christoph G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 419–453. Springer, 1988.

[160] Dan Boneh Matthias Jacob and Edward Felten. Attacking an Obfuscated Cipher by Injecting Faults. In Joan Feigenbam, editor, *Digital Rights Management - DRM 2002*. Springer.

[161] Alexander Maximov. Cryptanalysis of the "Grain" Family of Stream Ciphers. In Ferng-Ching Lin, Der-Tsai Lee, Bao-Shuh Lin, Shiuhpyng Shieh, and Sushil Jajodia, editors, *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security - ASIACCS 2006*, pages 283–288. ACM, 2006.

[162] Willi Meier and Othmar Staffelbach. Fast Correlation Attacks on Stream Ciphers. In C. G. Günther, editor, *Advances in Cryptology - EUROCRYPT '88*, volume 330 of *Lecture Notes in Computer Science*, pages 301–314. Springer, 1988.

[163] Willi Meier and Othmar Staffelbach. The Self-Shrinking Generator. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 205–214. Springer, 1994.

[164] Marine Minier and Henri Gilbert. Stochastic Cryptanalysis of Crypton. In Bruce Schneier, editor, *Fast Software Encryption - FSE 2000*, volume 1978 of *Lecture Notes in Computer Science*, pages 121–133. Springer, 2001.

[165] Shoji Miyaguchi. News on Feal Cipher, talk at the Rump session. In *Advances in Cryptology - CRYPT0 '90*.

[166] Sibylle Mund, Dieter Gollmann, and Thomas Beth. Some Remarks on the Cross Correlation Analysis of Pseudo Random Generators. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology - EUROCRYPT '87*, volume 304 of *Lecture Notes in Computer Science*, pages 25–35. Springer, 1988.

[167] J. Munilla and A. Peinado. HB-MP: A Further Step in the HB-Family of Lightweight Authentication protocols. *Computer Networks*, 51(9):2262–2267, 2007.

[168] Sean Murphy, Fred Piper, Michael Walker, and Peter Wild. Maximum Likelihood Estimation for Block Cipher Keys, RHUL research report, 1992.

[169] Sean Murphy and Matthew Robshaw. Essential Algebraic Structure Within the AES. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2002.

[170] Kaisa Nyberg. Linear Approximation of Block Ciphers. In Alfredo De Santis, editor, *Advances in Cryptology - EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 439–444. Springer, 1995.

[171] Kaisa Nyberg and Lars R. Knudsen. Provable Security Against a Differential Attack. *Journal of Cryptology*, 8(1):27–38, 1995.

[172] Kaisa Nyberg and Johan Wallén. Improved Linear Distinguishers for SNOW 2.0. In Matthew J. B. Robshaw, editor, *Fast Software Encryption - FSE 2006*, volume 4047 of *Lecture Notes in Computer Science*, pages 144–162. Springer, 2006.

[173] National Institute of Standards and Technology. FIPS-197: Advanced Encryption Standard, November 2001. Available at http://csrc.nist.gov/publications/fips/.

[174] National Bureau of Standards (NBS). Data Encryption Standard (DES). Federal Information Processing Standards Publication 46, 1977.

[175] Jacques Patarin. Etude des générateurs de permutations pseudo-aléatoires basés sur le schéma du D.E.S. PhD Thesis, University Paris VI, 1991.

[176] Jacques Patarin. Cryptoanalysis of the Matsumoto and Imai Public Key Scheme of Eurocrypt'88. In *Advances in Cryptology - CRYPTO '95*, volume 963 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 1995.

[177] Jacques Patarin. Security of Random Feistel Schemes with 5 or More Rounds. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 106–122. Springer, 2004.

[178] Jacques Patarin and Louis Goubin. Trapdoor one-way permutations and multivariate polynominals. In Yongfei Han, Tatsuaki Okamoto, and Sihan Qing, editors, *Information and Communication Security - ICICS'97*, volume 1334 of *Lecture Notes in Computer Science*, pages 356–368. Springer, 1997.

[179] Ronald L. Rivest. The RC4 Encryption Algorithm. RSA Data Security, Inc, March 12, 1992.

[180] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[181] M. Robshaw and O. Billet (Eds.). *New Stream Cipher Designs: The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*. Springer, 2008.

[182] Phillip Rogaway and Don Coppersmith. A Software-Oriented Encryption Algorithm. In Bart Preneel, editor, *Fast Software Encryption - FSE'94*, number 1008, pages 56–63, 1995.

[183] Carsten Rolfes, Axel Poschmann, and Christof Paar. Security for 1000 Gate Equivalents. In *Proceedings of ECRYPT workshop Secure Component and System Identification*, 2008.

[184] Rainer A. Rueppel. *Analysis and Design of stream ciphers*. Springer, 1986.

[185] Kouichi Sakurai and Yuliang Zheng. On Non-Pseudorandomness from Block Ciphers with Provable Immunity Against Linear Cryptanalysis. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, 1997.

[186] Ernst S. Selmer. Linear Recurrence Relations over Finite Fields. Department of Mathematics, University of Bergen, Norway, 1966.

[187] Adi Shamir. SQUASH - a New MAC With Provable Security Properties for Highly Constrained Devices Such As RFID Tags . In Kaisa Nyberg, editor, *Fast Software Encryption - FSE 2008*, Lecture Notes in Computer Science. Springer.

[188] Adi Shamir. The Generation of Cryptographically Strong Pseudo-Random Sequences. In Allen Gersho, editor, *Advances in Cryptology - CRYPTO '81*, page 1, 1981.

[189] Claude Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28:656–715, 1949.

[190] Akihiro Shimizu and Shoji Miyaguchi. Fast Data Encipherment Algorithm FEAL. In David Chaum and Wyn L. Price, editors, *Advances in Cryptology - EUROCRYPT '87*, volume 304 of *Lecture Notes in Computer Science*, pages 267–278. Springer, 1988.

[191] Thomas Siegenthaler. Correlation-immunity of Nonlinear Combining Functions for Cryptographic Applications. *IEEE Trans. Inform. Theory*, IT-30(5):776–780, 1984.

[192] Thomas Siegenthaler. Decrypting a Class of Stream Ciphers using Ciphertext only. *IEEE Trans. Computers*, C-34(1):81–84, 1985.

[193] Johan Håstad and Mats Näslund. BMGL: Synchronous Key-stream Generator with Provable Security. Submitted to the Nessie Project, 2000.

[194] Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. On the Provable Security of an Efficient RSA-Based Pseudorandom Generator. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology - ASIACRYPT 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 194–209. Springer, 2006.

[195] Anne Tardy-Corfdir and Henri Gilbert. A Known Plaintext Attack of FEAL-4 and FEAL-6. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 172–181. Springer, 1992.

[196] Serge Vaudenay. FFT-Hash-II is not yet Collision-free. In Ernest F. Brickell, editor, *Advances in Cryptology - CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 587–593. Springer, 1993.

[197] Serge Vaudenay. An Experiment on DES Statistical Cryptanalysis. In *ACM Conference on Computer and Communications Security*, pages 139–147, 1996.

[198] Serge Vaudenay. Provable Security for Block Ciphers by Decorrelation. In Michel Morvan, Christoph Meinel, and Daniel Krob, editors, *STACS 98, 15th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1373 of *Lecture Notes in Computer Science*, pages 249–275. Springer, 1998.

[199] Serge Vaudenay. Resistance Against General Iterated Attacks. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 255–271. Springer, 1999.

[200] David Wagner. The Boomerang Attack. In Lars Ramkilde Knudsen, editor, *Fast Software Encryption – FSE'99*, number 1636, pages 156–170, 1999.

[201] Dai Watanabe, Alex Biryukov, and Christophe De Cannière. A Distinguishing Attack of SNOW 2.0 with Linear Masking Method. In Mitsuru Matsui and Robert J. Zuccherato, editors, *Selected Areas in Cryptography - SAC 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 222–233. Springer, 2004.

[202] Dai Watanabe, Soichi Furuya, Hirotaka Yoshida, Kazuo Takaragi, and Bart Preneel. A New Keystream Generator MUGI. *IEICE Transactions*, 87-A(1):37–45, 2004.

[203] Mark N. Wegman and Larry Carter. New Classes and Applications of Hash Functions. In *Foundations of Cryptography FOCS 1979*, pages 175–182, 1979.

[204] Hongjun Wu. A New Stream Cipher HC-256. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption - FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 226–244. Springer, 2004.

[205] Hongjun Wu. Stream Cipher HC-128. eSTREAM, ECRYPT Stream Cipher Project, 2005.

[206] Hongjun Wu. Stream Cipher HC-256. eSTREAM, ECRYPT Stream Cipher Project, 2005.

[207] Brecht Wyseur, Wil Michiels, Paul Gorissen, and Bart Preneel. Cryptanalysis of White-Box DES Implementations with Arbitrary External Encodings. In Carlisle M. Adams, Ali Miri, and Michael J. Wiener, editors, *Selected Areas in Cryptography - SAC 2007*, volume 4876 of *Lecture Notes in Computer Science*, pages 264–277. Springer, 2007.

[208] Bo-Yin Yang, Owen Chia-Hsin Chen, Daniel J. Bernstein, and Jiun-Ming Chen. Analysis of QUAD. In Alex Biryukov, editor, *Fast Software Encryption - FSE 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 290–308. Springer, 2007.

[209] Andrew Yao. Theory and Applications of Trapdoor Function. In *Foundations of Cryptography FOCS 1982*, 1982.

Annexes

A Statistical Attack on RC6

Henri Gilbert[1], Helena Handschuh[2], Antoine Joux[3], and Serge Vaudenay[4]

[1] France Telecom
[2] Gemplus
[3] SCSSI
[4] Ecole Normale Supérieure – CNRS
Contact Helena.Handschuh@gemplus.com

Abstract. This paper details the attack on RC6 which was announced in a report published in the proceedings of the second AES candidate conference (March 1999). Based on an observation on the RC6 statistics, we show how to distinguish RC6 from a random permutation and to recover the secret extended key for a fair number of rounds.

1 Introduction

RC6 is one of the 15 candidate algorithms that were presented at the first Advanced Encryption Standard candidate conference in August 1998. It was submitted by RSA laboratories [9] and has been selected as one of the five finalists for the second round of the AES contest organized by NIST [1].

In this paper, we first show the existence of a statistical weakness in RC6 which allows to mount a distinguisher attack on a reduced number of rounds. This means that given a certain number of plaintext-ciphertext pairs, an attacker is able to distinguish RC6 from a random permutation. A distinguisher for the r-round version of a cipher may often be converted into a key-recovery attack on $r + 1$ or even more rounds. Matsui's linear cryptanalysis of DES provides a typical example of such a situation [7]. This also holds for RC6 : we show that we can gain one round as compared with our distinguisher to recover the extended keys of RC6 reduced to 14 rounds (or equivalently 15 RC6 inner rounds).

The paper is organised as follows : in the next Section we give the outlines of RC6 and in Section 3 we present the probabilistic event which leaks information. In Section 4 we explicitely construct the distinguisher and in Section 5 we adapt the latter to recover the extended secret key. Finally, we shortly discuss the case of RC5 and conclude.

2 RC6 Outlines

RC6 is characterized by three parameters (w, r, b). It is dedicated to w-bit microprocessors and encrypts $4w$-bit blocks by using four registers. (We assume that w

is an integral power of 2.) It has r rounds and uses a b-byte secret key. The nominal parameters for AES are $(32, 20, 16)$, $(32, 20, 24)$ and $(32, 20, 32)$, respectively for a 128, 196 and 256-bit user key. There is a key scheduling algorithm which extends the original b-byte key into an $2r + 4$-word array $S = (S_0, \ldots, S_{2r+3})$. In this paper, we will only use the w and r parameters, so we consider that the encryption is performed by using an arbitrary $2r + 4$-word array S which plays the role of the secret key.

The encryption is performed by using four registers A, B, C, D. The algorithm is described by the following pseudo-code.

Input: (A, B, C, D)
1. $B \leftarrow B + S_0$, $D \leftarrow D + S_1$
2. for $i = 1$ to r do
   $A \leftarrow ((A \oplus f(B)) \lll f(D)) + S_{2i}$
   $C \leftarrow ((C \oplus f(D)) \lll f(B)) + S_{2i+1}$
   $(A, B, C, D) \leftarrow (B, C, D, A)$
3. $A \leftarrow A + S_{2r+2}$, $C \leftarrow C + S_{2r+3}$

Output: (A, B, C, D)

Here the f function plays the role of a pseudo-random generator defined by

$$f(x) = g(x) \bmod 2^w \lll log_2 w = x(2x + 1) \bmod 2^w \lll log_2 w \ .$$

A picture of the RC6 encryption algorithm is given hereafter.

RC6 is very similar to RC5 in that it uses only simple operations such as binary addition, exclusive or and circular rotations. In addition, RC6 performs a simple modular multiplication.

Our results show that a reduced number of rounds of RC6 may be distinguished from a random permutation, which in turn enables an attacker to recover the secret keys of RC6 with one more round. This analysis also partly transposes to RC5. We would like to mention that an outline of our attack was introduced for the first time at the second AES conference in Rome in March 1999 [2], and that another paper dealing with the same kind of RC6 statistics [6] appears in these proceedings. However, the work reported in [6] and the work reported here are quite independent, both approaches for handling the RC6 statistics differ to some extent, and we feel it is important to present the attack announced in [2] in details here. Interestingly, both papers show that we can distinguish RC6 from a random permutation in polynomial time for a fair number of rounds, although it has been made clear [4,8] that the RC6 frame provides a pseudorandom permutation after five rounds once the data-dependent rotations are removed.

3 A Probabilistic Event on RC6 Encryption

For $1 \leq i \leq r$ and $0 \leq j < 4$, we let $R_{i,j}(S, a, b, c, d)$ denote the value of the register with index j (considering that index 0 is for A, index 1 is for B, ...) after

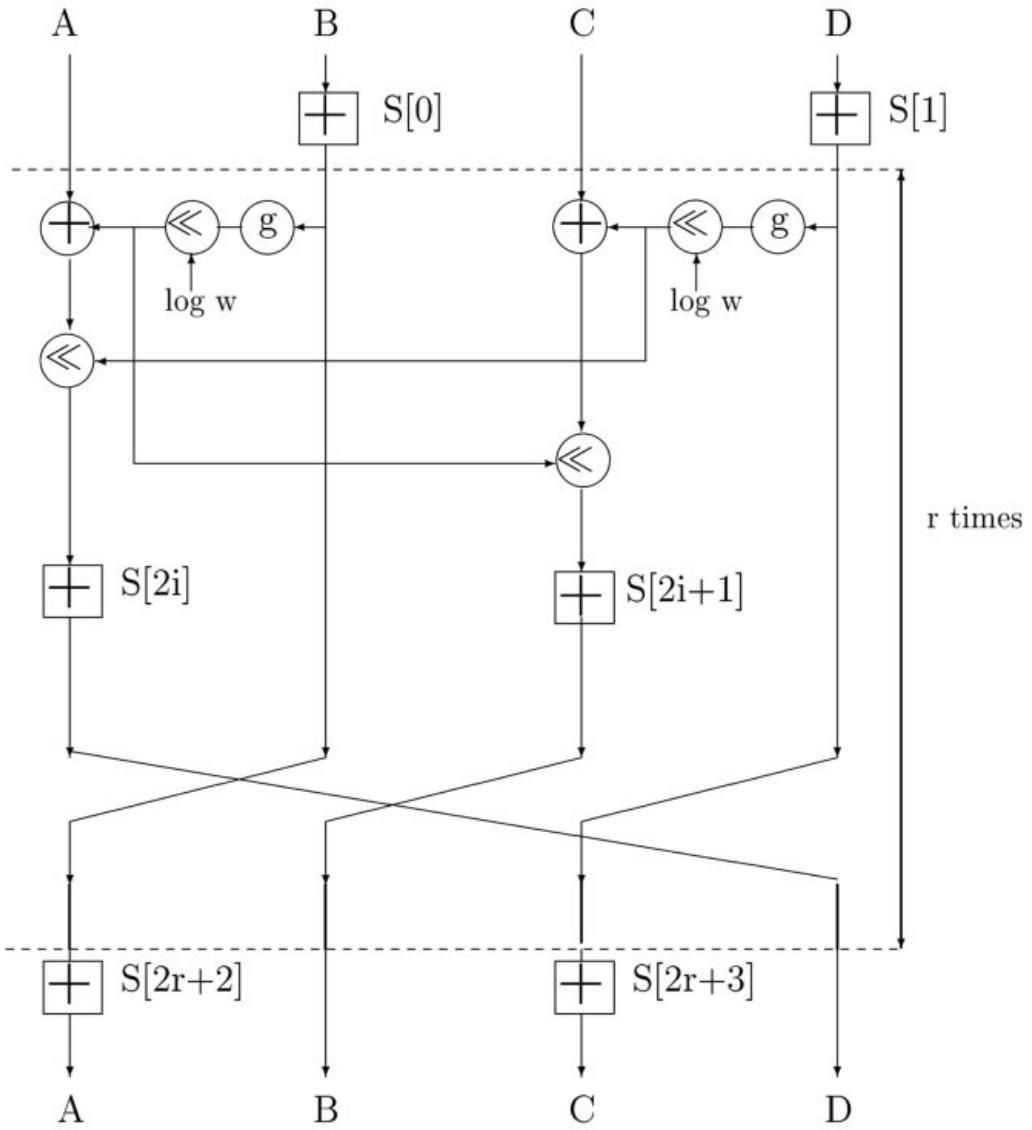


Fig. 1. Encryption with RC6-$w/r/b$ where $g(x) = x \times (2x + 1)$.

the ith round when the input of the encryption is (a, b, c, d) and the key is S. We can extend the above notation to $i = 0$ letting $R_{0,j}(S, a, b, c, d)$ denote the input words to the first round. We will omit (S, a, b, c, d) in most cases. In the sequel we implicitely assume that the j index is taken modulo 4. We start with the following simple fact.

Lemma 1. *For any i and any (S, a, b, c, d), we have*

$$\left.\begin{array}{l} f(R_{i-1,1}) \equiv 0 \pmod{w} \\ f(R_{i-1,3}) \equiv 0 \pmod{w} \end{array}\right\} \Longrightarrow \left\{\begin{array}{l} R_{i,3} - R_{i-1,0} \equiv S_{2i} \pmod{w} \\ R_{i,1} - R_{i-1,2} \equiv S_{2i+1} \pmod{w} \end{array}\right.$$

and in addition, $R_{i,0} = R_{i-1,1}$ and $R_{i,2} = R_{i-1,3}$.

This comes from the fact that if the mod w part of $f(B)$ and $f(D)$ are both zero in the ith round, then nothing is XORed onto the mod w part of A and C, and none are rotated.

This fact extends into the following

Lemma 2. *If we have* $f(R_{i-1,1}) \equiv f(R_{i-1,3}) \equiv 0 \pmod{w}$ *for* $i = k, k+2, \ldots, k+2\ell$, *then* $R_{k+2\ell,-2\ell-1} - R_{k-1,0} \bmod w$ *and* $R_{k+2\ell,1-2\ell} - R_{k-1,2} \bmod w$ *are constants which only depend on* S.

Assuming that the outputs of $f \bmod w$ behave like random numbers, this event holds with probability $w^{-2\ell}$. We thus have the following heuristic result which has been confirmed by statistical experiments for $w = 32$ and small values of r.

Theorem 1. *Under heuristic assumptions, there exists some functions* $c_1(S)$ *and* $c_2(S)$ *such that for random* $(R_{0,0}, \ldots, R_{0,3})$ *and a random* S *we have*

$$\Pr\begin{bmatrix} R_{r,1-r}(S) - R_{0,1}(S) \bmod w = c_1(S) \\ R_{r,3-r}(S) - R_{0,3}(S) \bmod w = c_2(S) \end{bmatrix} \approx w^{-2}\left(1 + w^{-2\lfloor \frac{r}{2} \rfloor}\right).$$

4 On Distinguishing RC6 from a Random Permutation

We can construct a distinguisher between RC6 and a random permutation by using the above theorem through a known plaintext attack.

1. The distinguisher first gets n random samples $(x_i, \mathrm{Enc}(x_i))$ where

   $$x_i = (x_{i,0}, x_{i,1}, x_{i,2}, x_{i,3})$$

   and

   $$\mathrm{Enc}_i = (y_{i,0}, y_{i,1}, y_{i,2}, y_{i,3}).$$

2. Then it hashes the samples onto

   $$h_i = (y_{i,1-r} - x_{i,1} \bmod w, y_{i,3-r} - x_{i,3} \bmod w).$$

3. It then creates w^2 counters which correspond to possible h_i values and counts the number $n_{(u,v)}$ of i indices such that $h_i = (u, v)$.
4. If the maximum of all $n_{(u,v)}$ is greater than a given threshold t, output 1, otherwise, output 0.

We let $\epsilon \approx w^{-2\lfloor \frac{r}{2} \rfloor}$ denote the probability that the event of Theorem 1 occurs for RC6. We need to compute the advantage in terms of n, t, ϵ of this attack for distinguishing RC6 from a random permutation.

Let us choose $t = n.w^{-2} + \delta$. (nw^{-2} is the expected value of one counter for random hashes so δ measures the deviation from the ideal expected case.)

The probability p that the distinguisher outputs 1 for RC6 is greater than the probability that the counter which corresponds to the constant values in

Theorem 1 is greater than t. When n is large, this counter tends towards a normal law with expected value $n(w^{-2}(1-\epsilon)+\epsilon)$ (which we approximate by $nw^{-2}+n\epsilon$) and variance approximately nw^{-2}. We let

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt.$$

We have

$$p \approx \varphi\left(-\frac{t-nw^{-2}-n\epsilon}{\sqrt{n}w^{-1}}\right) + \left(1-\varphi\left(\frac{t-nw^{-2}}{\sqrt{n}w^{-1}}\right)^{w^2-1}\right)$$

which is

$$p \geq \varphi\left(-\frac{t-nw^{-2}-n\epsilon}{\sqrt{n}w^{-1}}\right).$$

This means

$$p \geq \varphi\left(-\delta w n^{-\frac{1}{2}} + w\epsilon\sqrt{n}\right). \tag{1}$$

Now the probability p^* that the distinguisher outputs 1 for a random permutation is less that w^2 times the probability that one given counter is greater then t. This counter tends to behave like a normal law with expected value nw^{-2} and variance nw^{-2}. We thus have

$$p^* \leq w^2\varphi\left(-\frac{t-nw^{-2}}{\sqrt{n}w^{-1}}\right)$$

which means

$$p^* \leq w^2\varphi\left(-\delta w n^{-\frac{1}{2}}\right). \tag{2}$$

Therefore the advantage for distinguishing RC6 from a random permutation is

$$\text{Adv} \geq \varphi\left(-\delta w n^{-\frac{1}{2}} + w\epsilon\sqrt{n}\right) - w^2\varphi\left(-\delta w n^{-\frac{1}{2}}\right).$$

If we derive this function with respect to δ, we obtain the maximum when the derivative is equal to zero. The choice of δ which maximizes this right hand term is

$$\delta = \frac{2\log w}{\epsilon w^2} + \frac{\epsilon n}{2}$$

for which

$$\text{Adv} \geq \varphi\left(-\frac{2\log w}{\epsilon w\sqrt{n}} + \frac{\epsilon w\sqrt{n}}{2}\right) - w^2\varphi\left(-\frac{2\log w}{\epsilon w\sqrt{n}} - \frac{\epsilon w\sqrt{n}}{2}\right).$$

This analysis leads to the following result.

Theorem 2. *Let* $\alpha = \frac{\epsilon w \sqrt{n}}{2\sqrt{\log w}}$. *Under heuristic assumptions, the above distinguisher, when used with*

$$t = \frac{n}{w^2} + \frac{2\log w}{\epsilon w^2} + \frac{\epsilon n}{2} \quad \text{and} \quad n \geq 4\alpha^2 w^{4\left\lfloor \frac{r}{2} \right\rfloor - 2} \log w$$

has an advantage greater than

$$\text{Adv} \geq \varphi\left(\sqrt{\log w}(-\alpha^{-1} + \alpha)\right) - w^2 \varphi\left(\sqrt{\log w}(-\alpha^{-1} - \alpha)\right).$$

Considering $\alpha = 5$ we have

$$\text{Adv} \geq \varphi\left(\frac{24}{5}\sqrt{\log w}\right) - w^2 \varphi\left(-\frac{26}{5}\sqrt{\log w}\right)$$

with a complexity of

$$n \geq 100 w^{4\left\lfloor \frac{r}{2} \right\rfloor - 2} \log w.$$

We have to be concerned that the total number of samples cannot be greater than 2^{4w}, which is the total number of possible plaintexts. Hence the above attack is significant for

$$r \leq 2\left\lfloor \frac{4w - 7 - \log_2 \log w}{4 \log_2 w} + \frac{1}{2} \right\rfloor + 1 \ .$$

As an application, with the nominal choice $w = 32$ we obtain an advantage greater than $1 - 2^{-60}$ with a complexity of $n \approx 2^{20\left\lfloor \frac{r}{2} \right\rfloor - 2}$. Thus we can break up to $r = 13$ rounds (with $n = 2^{118}$).

5 On Recovering the Secret Key

5.1 A Simplified Approach for Recovering S_0 and S_1

Let us focus on nominal RC6 reduced to $r = 14$ rounds for a moment. Then a way of adapting the distinguisher to recover the whole secret key for RC6 reduced to 14 rounds is by a known plaintext attack which proceeds in the following way.

Suppose we black box encrypt a multiple m of the n plaintexts required by the previously described distinguisher on 13 rounds, thus obtaining m (x_i, y_i) plaintext-ciphertext pairs for the 14-round RC6. Let $\Delta A = A_{\text{out}} - A_{\text{in}} \pmod w$ where $A_{\text{out}} = y_{i,-14}$ and $A_{\text{in}} = x_{i,0}$ denote the input-output difference of the $\log_2 w$ least significant bits of the input word A and similarly let $\Delta C = C_{\text{out}} - C_{\text{in}} \pmod w$ where $C_{\text{out}} = y_{i,2-14}$ and $C_{\text{in}} = x_{i,2}$ denote the difference modulo w on input word C. For those (x_i, y_i) pairs such that the A and

C input words are not rotated at the first round, ΔA and ΔC are equal, up to the unknown constants $S_2 \pmod w$ and $S_3 \pmod w$, to the h_i differences considered in the 13-rounds distinguisher of Section 4.

The exhaustive trial of all the S_0 and S_1 keys, i.e. the computation for each (S_0, S_1) key assumption, of the $(\Delta A, \Delta C)$ frequencies distribution on the subset of plaintext-ciphertext pairs such that no A and C rotations occur at the first round, followed by the 13-round distinguisher test of Section 4, can be performed in an efficient way which avoids processing each plaintext individually for each key assumption.

• First we generate a table of $2^{2w+2\log_2 w}$ (e.g. 2^{74} for nominal RC6) elements, where each entry is the frequency observed for the plaintext-ciphertext pairs according to the value of the B and D input words as well as the ΔA and ΔC input-output differences modulo w (i.e. a potential value of the constant differences if all the rotations were zero as in our model).

• Now for approximately $2^{w-\log_2 w}$ (e.g. 2^{27}) "good" B values, we obtain that $f(B+S_0) \equiv 0 \pmod w$ in the first round. Therefore for each possible choice of the first subkey S_0, we may add together the frequencies of the $2^{w-\log_2 w}$ corresponding good B values. This requires a work load of about $2^{w-\log_2 w+w+2\log_2 w} = 2^{2w+\log_2 w}$ (e.g. 2^{69}) operations per S_0 guess. We are left with a table of $2^{w+2\log_2 w}$ - e.g. 2^{42} - $(D, \Delta A, \Delta C)$ frequencies.

• Next, for every possible S_1 value, we can do the same. For a given guess, we select the $2^{w-\log_2 w}$ possible values for D which achieve $f(D+S_1) \equiv 0 \pmod w$ in the first round, and add their frequencies together. We are left with a table of w^2 $(\Delta A, \Delta C)$ frequencies, the maximum of which corresponds to the sum of some key bits when the two subkeys are correctly guessed. This step requires an effort of $2^{w+\log_2 w}$ operations for all (S_0, S_1) subkey guesses.

• Once such a table of frequencies of the $(\Delta A, \Delta C)$ values has been obtained, the distinguisher of Section 4 may be applied quite naturally to it. If (S_0, S_1) is the correct subkey guess, one of the frequencies is expected to pass the test, whereas the test is expected to fail when wrong values have been picked. Thus this procedure allows us to recover the first two subkeys using a memory of less than $2 \cdot 2^{2w+2\log_2 w}$ words (e.g. 2^{75}) and a workload

$$C = 2^w \left(2 \cdot 2^{2w+\log_2 w}\right) = 2^{3w+\log_2 w+1} \ ,$$

(e.g. 2^{102}), which is far less than the number of encryptions needed for the distinguisher anyway. However, using this technique, we filter out about w^2

plaintext-ciphertext pairs, therefore we have to start off with far more pairs at the beginning of the attack in order to make sure the distinguisher gets enough information after the filtering phase. This leads to a required number of known plaintexts :

$$m \geq 100 w^{4\left\lfloor \frac{r-1}{2} \right\rfloor} \log w.$$

Note that for $w = 32$ and $r = 14$, m is about equal to the 2^{128} limit.

5.2 Improved Approach Without Filtering

As we saw in the last section the fact that we filter pairs for which the first two rotations are zero "costs" a factor w^2 in the number of plaintext-ciphertext pairs. We want to avoid this and use only the n pairs required by the distinguisher. We actually guess the first two rotations at the cost of some more memory.

• Let $\beta = f(B) \pmod w$ and $\delta = f(D) \pmod w$ be the two rotations of the first round. For each of the w^2 potential values of (β, δ), we generate a hash table for the frequencies of the tuples

$$\big(B, D, (A_{\text{in}} \lll \delta) \bmod w, A_{\text{out}} \bmod w, (C_{\text{in}} \lll \beta) \bmod w, C_{\text{out}} \bmod w\big)$$

Thus we have w^2 tables of size $2^{2w+4\log_2 w}$ (e.g. 2^{84}) each, giving all the frequencies for the various potential (β, δ) couples of rotations. Note that the generation of such tables may be optimized (avoiding an extra work factor of w^2 for each plaintext-ciphertext pair) in a way which will be discussed below.

• Now for every guess of S_0, for each of the w possible β values, we may select the $2^{w-\log_2 w}$ (e.g. 2^{27}) B values such that $f(B+S_0) = \beta \bmod w$ and, for each of the w potential δ values, add together, in the (β, δ) table, the frequencies of those t-uples for which the values of D, $\Delta A = (A_{\text{out}} - ((A_{\text{in}} \oplus f(B+S_0)) \lll \delta)) \bmod w$, $C_{\text{in}} \lll \beta \bmod w$ and $C_{\text{out}} \bmod w$ are the same. We thus obtain w^2 tables providing $(D, \Delta A, C_{\text{in}} \lll \beta \bmod w, C_{\text{out}} \bmod w)$ frequencies, at the expense of a $2^{2w+5\log_2 w}$ (e.g. 2^{89}) work load per S_0 assumption.

• Next, for each guess of S_1, for each of the w possible δ values, we may select the $2^{w-\log_2 w}$ (e.g. 2^{27}) D values such that $f(D+S_1) = \delta \bmod w$ and, for each of the w potential β values, add together, in the (β, δ) table derived at the former step, the frequencies of those $(D, \Delta A, C_{\text{in}} \lll \beta \bmod w, C_{\text{out}} \bmod w)$ tuples for which the values of ΔA and $\Delta C = (C_{\text{out}} - ((C_{\text{in}} \oplus f(D+S_1)) \lll \beta)) \bmod w$ are the same. By adding up all the $(\Delta A, \Delta C)$ frequencies obtained for all the (β, δ) pairs, we are left with a table of w^2 $(\Delta A, \Delta C)$ frequencies which can be used as an input to the distinguisher of Section 4. The distinguisher is expected

to succeed only when the two subkeys S_0 and S_1 are correctly guessed. Thus the above procedure provides the first two subkeys. The work load for this step is about $2^{w+4\log_2 w}$ for each (S_0,S_1) assumption.

Discussion. In order to optimize the generation of the w^2 (β, δ) tables of the (B, D, $A_{\text{in}} \ll \delta \bmod w$, $A_{\text{out}} \bmod w$, $C_{\text{in}} \ll \beta \bmod w, C_{\text{out}} \bmod w$) frequencies, we suggest the following technique. We denote by A_L and C_L (resp A_H and C_H) the $w/2+\lfloor\frac{\log_2 w}{2}\rfloor$ (e.g. 18) lowest (resp highest) weight bits of A_{in} and C_{in}. From the n plaintext-ciphertext pairs used in the attack, we first derive the four tables containing the (B, D, A_L, $A_{\text{out}} \bmod w$, C_L, $C_{\text{out}} \bmod w$), (B, D, A_L, $A_{\text{out}} \bmod w$, C_H, $C_{\text{out}} \bmod w$), (B, D, A_H, $A_{\text{out}} \bmod w$, C_L, $C_{\text{out}} \bmod w$), and (B, D, A_H, $A_{\text{out}} \bmod w$, C_H, $C_{\text{out}} \bmod w$) frequencies. Each of the w^2 (β, δ) tables of (B, D, $A_{\text{in}} \ll \delta \bmod w$, $A_{\text{out}} \bmod w$, $C_{\text{in}} \ll \beta \bmod w$, $C_{\text{out}} \bmod w$) frequencies can then be deduced from one of the four above tables. This way, we process the n samples only once, and the additional complexity factor of w^2 corresponding to all possible choices for (β, δ) will apply essentially to the number of entries in each table, which is about $2^{3w+2\lfloor\frac{\log_2 w}{2}\rfloor+2\log_2 w}$ For example, for $w = 32$ and $n = 2^{118}$, this complexity is about $2^{10} \cdot 2^{110}$ instead of $2^{10} \cdot 2^{118}$.

The complexity of the entire procedure for recovering the first two subkeys S_0 and S_1 is less than $2 \cdot 2^{3w+5\log_2 w}$ (e.g. 2^{122}).

Once the first two subkeys are found, we can decrypt one round using the data in the previously described tables and may apply the same technique on the next two subkeys. As we go on recovering the extended key piece by piece, the required number of plaintext-ciphertext pairs to make the distinguisher work decreases very fast. Thus the overall complexity of this attack stays well below the effort of an exhaustive search for the key.

6 On the Existence of Similar RC5 Statistics

RC6 is an enhancement of the RC5 encryption algorithm. RC5 is characterized by three parameters w (word size ; note that the RC5 block size is $2w$), r (number of rounds ; unlike an RC6 round, an RC5 round consists of two half rounds) and b (number of key bytes).

The following statistical property of RC5 is closely related to the RC6 properties summarised in Section 3 above : if, in ρ consecutive RC5 half rounds, the rotation amounts applied at each second half round are all equal to zero, then after ρ half rounds the $\log_2 w$ lowest weight bits of one of the two plaintext halfes A and

B has been simply added (modulo w) with a constant value derived from the key.

The analysis of Section 4 is easy to transpose, to show that ρ half rounds of RC5 can be distinguished from a random permutation using a number n of known plaintexts which stays within a small factor of $w^{2\left\lfloor \frac{\rho}{2} \right\rfloor - 1}$. For sufficiently low r values, this distinguisher can be used to guess the RC5-$w/r/b$ expanded key, using a number n of known plaintexts which stays within a small factor of $w^{2(r-1)-1}$. However, for usual RC5 parameter choices such as $r = 12$ and a 64-bit block size, the number of available plaintexts is far too low to mount such an attack.

There are some connections beween the above outlined RC5 attack and the RC5 linear attacks mentioned in [5], which require about $4w^{2(r-1)}$ known plaintexts. Both approaches are based related RC5 properties, and the main difference consists in handling $log_2 w$-bit statistics versus binary statistics. We conjecture - but are not fully sure, since we did not check the RC5 key derivation details - that the treatment of $log_2 w$-bit statistics might provide a slight performance improvement over the linear cryptanalysis approach.

7 Conclusion

Extending the work presented at the second AES conference, we have shown the existence of a special statistical phenomenon on RC6 which enables to mount a distinguisher attack on up to 13 rounds. As usual, this kind of attack is shown to be convertible into a known plaintext attack which can break up to 14 rounds of RC6 (or equivalently 15 inner rounds with or without post-whitening), requiring about 2^{118} known plaintexts, 2^{112} memory and a work load of 2^{122} operations. Of course this attack is not anywhere near practical, but still leads us to the conclusion that due to the existence of a slight but iterative statistical weakness in its round function, RC6 does not have a very conservative number of rounds.

References

1. http://www.nist.gov/aes
2. O. Baudron, H. Gilbert, L. Granboulan, H. Handschuh, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, S. Vaudenay, "Report on the AES Candidates," *The Second Advanced Encryption Standard Candidate Conference,* N.I.S.T., 1999, pp. 53–67.
3. FIPS 46, *Data Encryption Standard,* US Department of Commerce, National Bureau of Standards, 1977 (revised as FIPS 46–1:1988; FIPS 46–2:1993).
4. T. Iwata, K. Kurosawa, "On the Pseudorandomness of AES Finalists – RC6 and Serpent", These proceedings.

5. B. S. Kaliski Jr., Y. L. Yin, “On the Security of the RC5 Encryption Algorithm”, RSA Laboratories Technical Report TR-602, Version 1.0 - September 1998.
6. L. Knudsen, W. Meier, “Correlations in RC6 with a reduced number of rounds ”, These proceedings.
7. M. Matsui, “The first experimental cryptanalysis of the Data Encryption Standard”. In *Advances in Cryptology - Crypto'94*, pp 1-11, Springer Verlag, New York, 1994.
8. S. Moriai, S. Vaudenay, “Comparison of randomness provided by several schemes for block ciphers”, Preprint, 1999.
9. R.L. Rivest, M.J.B. Robshaw, R. Sidney and Y.L. Yin,“The RC6 Block Cipher”, v1.1, August 20, 1998.
10. S. Vaudenay, “An experiment on DES - Statistical Cryptanalysis”. In *3rd ACM Conference on Computer Security*, New Dehli, India, pp139-147, ACM Press, 1996.

Stochastic Cryptanalysis of Crypton

Marine Minier and Henri Gilbert

France Télécom R&D
38-40, rue du Général Leclerc
92794 Issy les Moulineaux Cedex 9 - France
Tel: +33 1 45 29 44 44

Abstract. Crypton is a 12-round blockcipher proposed as an AES candidate by C.H. Lim in 1998. In this paper, we show how to exploit some statistical deficiencies of the Crypton round function to mount stochastic attacks on round-reduced versions of Crypton. Though more efficient than the best differential and linear attacks, our attacks do not endanger the practical security offered by Crypton.

1 Introduction

Crypton [Li98] is a 12-round blockcipher which was submitted by C.H. Lim as one of the 15 candidates at the first Advanced Encryption Standard conference in August 1998. Crypton offers several interesting features. The encryption and decryption processes are strictly identical up to the key schedule (a quite remarkable property given the substitution/permutation structure of the cipher). Crypton is highly parallelizable and flexible, and thus well suited for efficient implementation on nearly any hardware or software platform. Moreover, Crypton provides some provable resistance against linear and differential cryptanalysis.

The main cryptanalytic results obtained on Crypton so far are the analysis of the best differential and linear attacks by the algorithm designer [Li98], a transposition of the square attack to the 6-round Crypton by C. D'Halluin et al. [Hal99], the discovery of some weak keys by Vaudenay [Ba99], and statistical observations contained in an annex of [Ba99].

C.H. Lim introduced in 1999 [Li99] a modified Crypton (denoted by Crypton v1.0) with a new keyschedule and new S-boxes designed as to lower the number of high probability differential and linear characteristics. Though most of this paper is dedicated to the analysis of the initial version of Crypton, the impact of the Crypton v1.0 S-box modifications is also discussed in the last Section.

We present here two attacks of round reduced versions of Crypton (up to 8 rounds) which are based on iterative statistical properties of the round function. A short outline of preliminary (unquantified) versions of these attacks has been already published in a paper presented at the second AES conference [Ba99]. Based on extra analysis and computer experiments, we provide here more precise assessments of the statistical biases and the performance of these attacks.

Both attacks can be broadly described as stochastic attacks. The block values (or a subset of the difference values if the attack uses differential statistics) are

partitioned into a small number of classes, and any k-round partial encryption is modeled as a stochastic process (i.e. a sequence of random variables providing the class values at the boundaries of the various rounds). Each round is characterized by one matrix of key dependent or key-independent transition probabilities between input and output classes. Under the heuristic assumption that the above process is nearly markovian, the behaviour of such a k-rounds partial encryption is well approximated by the product of the k one-round transition probabilities matrices [1].

We do not claim that the idea of that kind of generalization of more traditional "characteristics-based" attacks is new : Murphy et al.' likehood estimation [Mu], Vaudenay's χ^2 cryptanalysis[Va95], Lai and Massey's modeling of markovian ciphers [LM91], Harpes and Massey's partition cryptanalysis [HM97]provide frameworks which describe similar generalizations. However, there are not yet numerous examples of ciphers where such approaches bring some real added value. We believe Crypton is a good example of an algorithm for which this is the case.

A stochastic attack is feasible if there exists a partition of the blocks (or of the considered subset of difference values) such that for each key value, the transition probabilities among classes differ substantially from the transition probabilities a random permutation of the block or difference values would provide. The key cryptanalytic issue consists in finding such a suitable partition. In the case of Crypton, the partitions of the block values and of difference values we are using are based upon some invariance properties of the linear part of the round function which involve only four "active" bytes of the inputs or outputs to the non linear part.

The rest of this paper is organized as follows : Section 2 briefly summarizes the Crypton cipher. Section 3 presents the iterative statistical properties which form the starting point for our attacks. Section 4 presents a stochastic attack based upon a partition of difference values into 257 classes. Section 5 presents a stochastic attack based upon a partition of blocks into 16 classes. Section

[1] Stochastic attacks represent a generalization of "Characteristics-based attacks", such as linear attacks, differential attacks, or truncated differential attacks. As a matter of fact, characteristics based attacks are based upon a partition of block or difference values at each round into only two classes. For instance, in linear attacks, blocks are partitioned according to the binary value of a fixed linear combination of the key bits. In differential attacks one considers the unbalanced partition of the differences between one single difference value on one hand and the complementary set of all other difference values on the other hand. In truncated differential attacks, one considers a partition of difference values according to the characteristic function of a set of difference values satisfying certain constraints, etc. In characteristics based attacks of blockciphers, one single transition probability, namely the probability of the considered characteristic, entirely determines the (2x2) transition probabilities matrix associated with a partial encryption. The stochastic attacks considered in this paper are not "characteristics-based" because they involve partitions in strictly more than two classes.

6 investigates the (quite positive) impact of the modifications introduced in Crypton v1.0 and Section 7 concludes the paper.

2 An Outline of Crypton

Crypton encrypts 128-bit blocks under the control a key of length up to 256 bits. The nominal value of the r number of rounds is 12. The algorithm consists of the encryption function itself and a keyschedule that derives $(r+1)$ 128-bit subkeys from the key and an encryption function. Since the attacks presented here do not rely at all upon the properties of the key schedule, we do not describe it here.

The encryption function consists of r rounds surrounded by an input transformation (defined by an XOR between the plaintext block and the first subkey) and an output transformation (defined as a fixed modulo 2 linear mapping).

Let us represent a 128-bit block A by :

$$A = \begin{pmatrix} a_{0,3}\ a_{0,2}\ a_{0,1}\ a_{0,0} \\ a_{1,3}\ a_{1,2}\ a_{1,1}\ a_{1,0} \\ a_{2,3}\ a_{2,2}\ a_{2,1}\ a_{2,0} \\ a_{3,3}\ a_{3,2}\ a_{3,1}\ a_{3,0} \end{pmatrix} \begin{matrix} A[0] \\ A[1] \\ A[2] \\ A[3] \end{matrix}$$

where each $a_{i,j}$ is a byte.
One round consists of a byte substitution γ, followed by a bit permutation π, a bytes transposition τ and a subkey addition σ. γ (γ_o for odd round, γ_e for even round) uses two S-boxes S_0 and S_1. We only describe of γ_o ; to obtain γ_e one just needs to exchange S_0 and S_1.

$$\begin{pmatrix} b_{0,3}\ b_{0,2}\ b_{0,1}\ b_{0,0} \\ b_{1,3}\ b_{1,2}\ b_{1,1}\ b_{1,0} \\ b_{2,3}\ b_{2,2}\ b_{2,1}\ b_{2,0} \\ b_{3,3}\ b_{3,2}\ b_{3,1}\ b_{3,0} \end{pmatrix} \xleftarrow{\gamma_o} \begin{pmatrix} S_1(a_{0,3})\ S_0(a_{0,2})\ S_1(a_{0,1})\ S_0(a_{0,0}) \\ S_0(a_{1,3})\ S_1(a_{1,2})\ S_0(a_{1,1})\ S_1(a_{1,0}) \\ S_1(a_{2,3})\ S_0(a_{2,2})\ S_1(a_{2,1})\ S_0(a_{2,0}) \\ S_0(a_{3,3})\ S_1(a_{3,2})\ S_0(a_{3,1})\ S_1(a_{3,0}) \end{pmatrix}$$

π (π_o for odd rounds, π_e for even rounds) is a bit permutation. We only describe effects of π_o for odd rounds, effects of π_e are similar. π_o is given by

$$\begin{aligned} T &= A[0] \oplus A[1] \oplus A[2] \oplus A[3], \\ B[0] &\leftarrow (A[0] \wedge MI_0) \oplus (A[1] \wedge MI_1) \oplus (A[2] \wedge MI_2) \oplus (A[3] \wedge MI_3) \oplus T, \\ B[1] &\leftarrow (A[0] \wedge MI_1) \oplus (A[1] \wedge MI_2) \oplus (A[2] \wedge MI_3) \oplus (A[3] \wedge MI_0) \oplus T, \\ B[2] &\leftarrow (A[0] \wedge MI_2) \oplus (A[1] \wedge MI_3) \oplus (A[2] \wedge MI_0) \oplus (A[3] \wedge MI_1) \oplus T, \\ B[3] &\leftarrow (A[0] \wedge MI_3) \oplus (A[1] \wedge MI_0) \oplus (A[2] \wedge MI_1) \oplus (A[3] \wedge MI_2) \oplus T, \end{aligned}$$

where $MI_0 = c0300c03, MI_1 = 03c0300c, MI_2 = 0c03c030, MI_3 = 300c03c0$ (in hexadecimal). We can notice that if we omit the addition with T, π results in permutations of 4 2-bit words in each of the 16 2-bit words columns of the A matrix.

The bytes transposition τ just consists in exchanging all $a_{i,j}$ and $a_{j,i}$ pairs of bytes in the A matrix, and the key addition σ_K just consists of an exclusive-or between A and a 128-bit subkey K.

3 Statistical Properties of the Round Function

In this Section we introduce two iterative properties of the $\tau \circ \pi$ linear part of the Crypton round function which involve only four bytes of the inputs or outputs to the non linear part - and thus represent limitations in the diffusion achieved by $\tau \circ \pi$. These two properties, which are outlined in [Ba99] are to some extent dual of each other, and can be summarized as follows : (1) some $\tau \circ \pi$ input values equal to zero except on at most four bytes are transformed into output values equal to zero except for at most four other bytes ; (2) for certain sets of 4 of the 16 $\tau \circ \pi$ input bytes and certain associated sets of 4 of the 16 $\tau \circ \pi$ output bytes, there exist a (linear) four bytes to 4 bits function Φ such that the images by Φ of the four input bytes and the four output bytes are equal. Property (1), as applied to difference values in the encryption of pairs of plaintexts, can be seen as an iterative truncated differential whereas property (2) can be to a certain extent compared to a set of iterative linear characteristics.

We introduce some additional notation to split each $a_{i,j}$ byte of an A block into four 2-bit words : $a_{i,j} = (a_{3,i,j}, a_{2,i,j}, a_{1,i,j}, a_{0,i,j})$ where i is the line index and j the column index. $i \in [0,3]$ and $j \in [0,3]$. $a_{k,i,j}$ will be the 2-bits word of line i, column j and position k. We define the "square" associated with the (k,i,j) triplet as the $((a_{k,i,j}, a_{k,i+2,j}, a_{k,i,j+2}, a_{k,i+2,j+2}))$ quartet of 2-bit words. Note that indexes are implicitly taken modulo 4. Under some conditions, squares are preserved (up to a modification of the associated (k,i,j) indexes) by the γ_o, γ_e, π_o, π_e and τ functions.

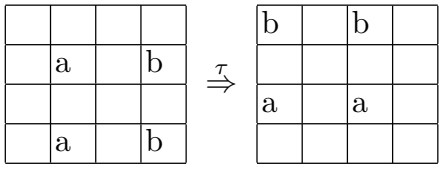
3.1 Property (1)

One can see that if we omit T, π_o and π_e just permute the $a_{k,i,j}$ 2-bit words (more precisely, they only permute the j indexes), and thus they transform any square associated to a (k,i,j) triplet into the same square associated with another (k,i',j') triplet. This stays valid for the real π_o and π_e functions under the two additional conditions (i) $a_{k,i,j} = a_{k,i+2,j}$ and (ii) $a_{k,i,j+2} = a_{k,i+2,j+2}$. Squares are also preserved by the τ function (up to a modification of the i and j indexes). Moreover, under conditions (i) and (ii), "twin squares" associated with two (k,i,j) and $(k+2,i,j)$ triplets of indexes are transformed into two other "twin squares" associated with two (k,i',j') and $(k+2,i',j')$ triplets of indexes.

In order to cryptanalytically exploit property (1), we can consider special difference values equal to zero except on two "twin squares" (k,i,j) and $(k+2,i,j)$ and such that the (i) and (ii) conditions are satisfied for both squares. For instance, squares of difference values of the following form stay entirely invariant under the π_o mapping :

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0\ 0 0 x_1 x_2 0 0 y_1 y_2 & 0\ 0 0 x'_1 x'_2 0 0 y'_1 y'_2 \\ 0 & 0 & 0 & 0 \\ 0\ 0 0 x_1 x_2 0 0 y_1 y_2 & 0\ 0 0 x'_1 x'_2 0 0 y'_1 y'_2 \end{pmatrix} = \begin{pmatrix} 0\ 0\ 0\ 0 \\ 0\ a\ 0\ b \\ 0\ 0\ 0\ 0 \\ 0\ a\ 0\ b \end{pmatrix}$$

Moreover, for the above example, τ just modifies location of squares :



In the sequel we will keep using a representation of twin squares at the byte level (as in the above representation of τ) to more compactly depict the effects of π_o, π_e and τ.

3.2 Property(2)

Let us consider any two twin squares with a (k, i, j) triplet and a $(k + 2, i, j)$ triplet of indexes. As seen before, if there was no T term in the π definition, these two twin squares would be just displaced by the $\tau \circ \pi$ linear function (without any change in the quartet values) to two twin squares associated with (k, i', j') and $(k + 2, i', j')$ triplets of indexes. Now if we take the T term into account, we can see that the $\phi_{k,i',j'}$ 2-bit XOR of the four 2-bit words of the (k, i', j') output square is still equal to the $\phi_{k,i,j}$ XOR of the four 2-bit words of the (k, i, j) input square (just because the additional terms introduced by T twowise compensate). The same property obviously holds for the squares associated with the $(k+2, i, j)$ input triplet and the $(k + 2, i', j')$ output triplet : $\phi_{k+2,i,j} = \phi_{k+2,i',j'}$. Thus, if we denote by $\Phi_{k,i,j}$ the 4-bit word $\phi_{k,i,j} \oplus 4.\phi_{k+2,i,j}$, we can summarize the obtained invariance property by the equality $\Phi_{k,i,j} = \Phi_{k,i',j'}$. In other words, the 4-bit linear combination $\Phi_{k,i,j}$ of the four bytes involved in two twin squares is kept invariant by the linear part of Crypton (up to a change of considered four bytes positions).

In Section 5, we will mount an attack based upon partitioning block values in 16 classes according to the value of such a 4-bit Φ values.

In summary, we have identified in properties (1) and (2) some correlations between the input and the output of the $\tau \circ \pi$ part of the round function which involve only 4 "active" input and output bytes. It remains to study how much correlation is left on entire rounds if one also takes the non linear part of the Crypton round function into account.

4 Stochastic Attack Using Differential Properties

4.1 Computing Transition Probabilities

The cryptanalysis is based upon property (1) and uses differences of the form described in part 3.1. That's why, in order to describe elements which stay invariant by π and τ, we introduce two sets of possible difference values at the byte level :

$$D_1 = \{0,1,2,3,16,17,18,19,32,33,34,35,48,49,50,51\}$$
$$D_2 = \{0,4,8,12,64,68,72,76,128,132,136,140,192,196,200,204\}$$

We include zero in both sets although in practice this difference value can't appear in an attack. D_1 corresponds to all possible choices of the 00xx00xx byte and D_2 to all possible choices of the xx00xx00 byte. Those sets permit us to create "twin square" invariant under the linear part. We denote by (δ_1, δ_2) differences of the form

$$\Delta_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \delta_1 & 0 & \delta_2 \\ 0 & 0 & 0 & 0 \\ 0 & \delta_1 & 0 & \delta_2 \end{pmatrix}$$

where Δ_1 is represented by a 4×4 matrix of bytes with $\delta_1 \in D_1$ and $\delta_2 \in D_1$ or of the form

$$\Delta_2 = \begin{pmatrix} 0 & \delta_1 & 0 & \delta_2 \\ 0 & 0 & 0 & 0 \\ 0 & \delta_1 & 0 & \delta_2 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

with $\delta_1 \in D_2$ and $\delta_2 \in D_2$. The above Δ_1 and Δ_2 difference matrices are just examples of all possible difference values that can be taken. As a matter of fact, (δ_1, δ_2)can be any "twin square" difference satisfying conditions (i) and (ii) of Section 2.
We can say by the invariance properties seen above that there exist δ'_1 and δ'_2 in D_1 such that

$$\tau(\pi_i(\Delta_1)) = \begin{pmatrix} \delta'_2 & 0 & \delta'_2 & 0 \\ 0 & 0 & 0 & 0 \\ \delta'_1 & 0 & \delta'_1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

So we can deduce that with a certain probability p, the "twin squares" represented by the (δ'_1, δ'_2) pair could result, after being passed through S-boxes, into a (δ_3, δ_4) pair such that

$$\gamma_i(\tau(\pi_i(\Delta_1))) = \begin{pmatrix} \delta_3 & 0 & \delta_4 & 0 \\ 0 & 0 & 0 & 0 \\ \delta_3 & 0 & \delta_4 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

An S-box output with this form permits us to use another time the invariance property of $\gamma_i(\tau(\pi_i(\Delta_1)))$ into π_i and τ on an other "twin square". We obtain, with a certain probability p', the invariance of Δ_1 on one entire round. With the same construction, we can establish the invariance of Δ_2 on one round with a certain probability p".
For all (δ_1, δ_2) pairs of D_1 and all (δ_3, δ_4) pairs of D_1, we can compute the probability of getting a (δ_3, δ_4) output difference from a (δ_1, δ_2) input difference after

one round. All probabilities are key independent and only depend on differential properties of S-boxes.

$$\begin{aligned} Pr[\Delta' = (\delta_3, \delta_4) | \Delta = (\delta_1, \delta_2)] &= Pr[\delta_1 \rightarrow \delta_3].Pr[\delta_2 \rightarrow \delta_3] \\ &\quad .Pr[\delta_1 \rightarrow \delta_4].Pr[\delta_2 \rightarrow \delta_4] \\ &= \frac{d_{\delta_1\delta_3}}{256}.\frac{d_{\delta_2\delta_3}}{256}.\frac{d_{\delta_1\delta_4}}{256}.\frac{d_{\delta_2\delta_4}}{256} \end{aligned}$$

where $d_{\delta_i\delta_j}$ is the number of bytes such that : $\delta_j = S(x) \oplus S(x \oplus \delta_i)$, S represents the appropriate S-box of Crypton. We obtain a 256×256 matrix M of transition probabilities over one round by computing p for 256 couples of possible input values (δ_1, δ_2) and for 256 couples of possible output values (δ_3, δ_4) with $\delta_1, \delta_2, \delta_3, \delta_4 \in D_1$. Columns of M represent all probabilities of transition between input (δ_1, δ_2) and output (δ_3, δ_4).We use the same method to compute probabilities of transition associated with D_2.

Now let us consider differentials over two rounds, i.e. transition between an input value (δ_1, δ_2) and an output value (δ_3, δ_4). A lower bound on the probabilities of such differentials is provided by summing up the probabilities of all intermediate values which belong to D_1 :

$$p_{i,j} = \sum_{i=0}^{255} \sum_{j=0}^{255} p_{i,k} p_{k,j}$$

where $p_{i,j}$ represent coefficients of M. With this relation, we consider all the possible intermediate values. There exists a stochastic dependence between the difference values at the various rounds. We make the heuristic assumption that the sequence of difference values over several rounds satisfies the "Markov property", i.e. the distribution of probabilities of the differences at the output of any round only depend on the distribution of probabilities of the differences at the input of the same round. So, we can compute M^n, which represents key-independent transition probabilities between input and output differences on an n-rounds scheme. For example, to compute transition probabilities for 6 rounds, we compute M^6. One cryptanalytically meaningful measure of the unbalanceness of the obtain matrix M consists of computing the sums of the probabilities in each column. The performance of the attack hereafter depends upon the value obtained for the "best column", i.e. the best pair of input difference values (δ_1, δ_2).

4.2 Attack Procedure

We present here an attack on a complete eight-round Crypton (i.e. taking into account the first addition of subkey K_0 and the final transformation). Under the heuristic assumptions summarized above, we can compute probabilities of best couples of difference on several rounds. In particular we obtain the following figures for 6 inner rounds of Crypton : if the input difference is the (18,

18) pair of Δ_1 values, the probability that the output be a (δ_1, δ_2) pair with (δ_1,δ_2)$\in D_1 \times D_1$ is $2^{-120.72}$. In the same way, if input is the (128, 128) pair of D_2 values, probability that the output be a (δ_1, δ_2) pair of D_2 values is $2^{-112.62}$.

This result on six inner rounds can be used to attack an eight inner rounds attack. Due to the late occurrence of the σ key addition in the first inner round, we can control differences at the output of the first occurrence of γ, and thus the first round can be treated just as a keyless "initial permutation". A 1R-attack permits to also gain the eighth round. So, we can exploit the a priori known 6-rounds properties in a chosen plaintext attack to obtain some bits of information on the last round key K_8.

In order to efficiently generate pairs of chosen plaintexts which difference is equal to the $\Delta_2 = (128, 128)$ value at the output of the first occurrence of γ, we group plaintexts in structures. Two 128-bits elements belong to the same structure if and only their images by γ are equal except for some of the 16 bits of Δ_2. Each structure has 2^{16} elements. We know that, if two X and X' plaintexts belong to the same structure, then with probability $2^{-112.62}$, the correponding inputs to the the eighth round are of the form Y and $Y \oplus \Delta_2$ (where Δ_2 is of D_2 values). We only consider those (X, X') pairs such that the C$\oplus$C' ciphertext difference is null everywhere except at most on the four non zero bytes of the Δ_2 Y differences. For those pairs which pass that filtering condition, we go up the last round starting from C and C' and checking whether the resulting $Y \oplus Y'$ has the right form. Some couples are "false" alarms, i.e. pass the filtering condition but don't belong to our set (this happens with a probability equal to 2^{-96}). Once selected "good" couples, we test the possible values of four bytes of key in going up at the end of eighth round. The candidate value which appears most often is the right one. We obtain four bytes of information on K_8 [2].

We can go up through the final transformation Φ_e because it does not change output values of eighth round. Taking into account the first key-addition σ_0 just increases the number of "false" alarms. The number of plaintexts to cipher is $N = 2^{112.62}$. But we must take a security for "false" alarms. We claim that taking $N = 2^{114.62}$ is enough. So we can obtain 32 bits of information of K_8 by ciphering $2^{114.62}$ couples X and X$\oplus\Delta$ in a complete eight-rounds version of Crypton. Complexity of this attack is $2^{114.62}$ encryptions and 2^{96} additional computations.

This attack is faster than an exhaustive search and than all differential attacks. As a matter of fact, it can be shown that the probability of the best characteristic for an eigth-round attack is 2^{-120}.

[2] The same procedure can be repeated with other square locations (at the expanse of a slight increase of the N number of chosen plantexts) to entirely derive K_8. Once the last subkey has been entirely derived, the same procedure can be repeated (with the same plaintexts) to derive the entire expanded key.

5 Stochastic Cryptanalysis of Crypton Using a Partition of Blocks in 16 Classes

5.1 Computation of Transition Probabilities

The cryptanalysis presented in this Section is based on Property (2) of Section 2. Let us consider inside an intermediate block

$$A = \begin{pmatrix} * & * & * & * \\ * & X & * & Y \\ * & * & * & * \\ * & Z & * & T \end{pmatrix}$$

encountered in the Crypton encryption process, a (X, Y, Z, W) quartet of bytes associated with a given (i, j) pair of indices. We can partition the blocks space into 16 classes according to the 4-bit value $\Phi_0[X, Y, Z, W] = \Phi_{0,i,j}$ (or alternatively into 16 other classes according to the 4-bit value $\Phi_1[X, Y, Z, W] = \Phi_{1,i,j}$).

Property (2) states that the linear part of Crypton leaves Φ_0 and Φ_1 values unchanged (provided that the final Φ_0 or Φ_1 value is computed from four appropriately selected bytes associated with a (i', j') pair of indices deduced from (i, j)).

It is easy to see that the σ key addition transformation just results in XORing the $\Phi_0[X, Y, Z, W]$ (resp $\Phi_1[X, Y, Z, W]$) value associated with a (X, Y, Z, W) quartet of bytes with a $\Phi_0[K_X, K_Y, K_Z, K_W]$ (resp $\Phi_1[K_X, K_Y, K_Z, K_W]$) 4-bit constant which depends on four subkey bytes.

We now investigate the effect of the S_0 and S_1 S-boxes of the γ non linear part of the Crypton round function on the $\Phi_0[X, Y, Z, W]$ (resp $\Phi_1[X, Y, Z, W]$) class values. For that purpose, for each of the S_0 and S_1 S-boxes, we compute the values

$$\begin{gathered} \#\{\mathrm{X,Y,Z,W}\in[0,255]^4 \ /\Phi_0 \ [\mathrm{X,Y,Z,T}]=\mathrm{a} \\ \text{and } \Phi_0 \ [S_\epsilon(\mathrm{X}),S_\epsilon(\mathrm{Y}),S_\epsilon(\mathrm{Z}),S_\epsilon(\mathrm{T})]=\mathrm{b}\}\text{-}(256)^3 \\ \text{and} \\ \#\{\mathrm{X,Y,Z,W}\in[0,255]^4 \ /\Phi_1 \ [\mathrm{X,Y,Z,T}]=\mathrm{a} \\ \text{and } \Phi_1 \ [S_\epsilon(\mathrm{X}),S_\epsilon(\mathrm{Y}),S_\epsilon(\mathrm{Z}),S_\epsilon(\mathrm{T})]=\mathrm{b}\}\text{-}(256)^3 \end{gathered}$$

where ϵ takes values 0 or 1, and a and b are in [0,15].These values represent biases with respect to the average value $(256)^3$. We obtain four 16×16 matrices (one for Φ_0 and S_1 (see appendix A), one for Φ_0 and S_0, one for Φ_1 and S_1, one for Φ_1 and S_0). The columns of such matrices are indexed by a and their lines by b, and the value associated with column a and line b represents (up to a multiplicative factor of $(256)^4$) the bias of the transition probability from the class associated with the Φ input value a to the class associated with the Φ output value b.

Now it clearly results from the above properties of γ, $\tau \circ \pi$, and σ that the way one entire round of Crypton affects the Φ_0 or Φ_1 values can be represented by a 16×16 matrix of transition probabilities (or equivalently of biases) obtained by multiplying one of the four above matrices by the 16×16 key-dependent

permutation matrix which entry associated with the a column and the b line is equal to 1 if $b = a \oplus \Phi_0[K_X, K_Y, K_Z, K_W]$.

Under the heuristic assumption that the behaviour of the cipher is nearly markovian, we can multiply those matrices to represent transitions of Crypton on n rounds. Of course the obtained matrix is key dependent (it depends upon 4 linear combinations of the key bits per round). However the orders of magnitude of the biases encountered in the product matrix are the same for most values of the keybits (they are larger than average for a few special values, e.g. when all 4-bit key words are equal to zero).

5.2 Attack Procedure

We present here a chosen plaintext of 8 inner rounds of Crypton (i.e. without the first key-addition and without the final transformation) which can be extended to an attack of the full 8-rounds version of Crypton (including the initial and the final transformation), with very similar performance. This basic attack is a 1-R attack based upon the above described computations of 6-rounds transition matrices (given by the product of 6 one-round transition matrices).

We select a (i, j) pair of first-round indexes and bit ϵ (0 or 1) and encrypt N chosen plaintext blocks such at the input to the first occurrence of the σ transformation (at the end of the first round), the $\Phi = \Phi(b, i, j)$ is equal to a constant 4-bit value α. We can expect the resulting Φ value associated with the input to the last round to be distributed according to unbalanced probabilities (or equivalently biases) given by the α column of the 6-rounds transition matrix.

We are using the χ^2 test in the same way as described in [HG97] to test four key bytes derived from the last round subkeys (i.e. those linear combinations of the last round subkey bits enabling to recover the four bytes involved in the Φ value of the input to the last round). For each of the 2^{32} key assumptions, we can partially decrypt the last round and compute (in at most 2^{32} operations, provided that ciphertext blocks have been first partitioned according to the value of a suitable 4-bytes word) the distribution of the Φ values at the input to the last round and the χ^2 indicator associated with the obtained distribution of 16 empiric frequencies. The obtained indicator is expected to be substantially higher for the right assumption on the four key bytes.

The N number of plaintexts required by the attack is inversely proportional to the sum of the squares of the a priori expected biases. Thus the required number of plaintexts can be deduced from the biases in the above introduced 6-round matrices. The best result are obtained with Φ_1 computations (instead of Φ_2 computations. For average values on the 6-uples of 4-bit key words, the obtained N value is close to 2^{112}. For the best 6-uple values (e.g. the null 6-uple), about 2^{104} suffice to recover the 32 last round key bits.

So in summary we obtain 32 bits of information about the last round keybit using 2^{112} chosen plaintexts. Therefore we can recover the entire last round subkey (and then, once having decrypted the last round, derive the other subkeys, using the same method) with say 2^{116} chosen plaintexts.

6 Results Concerning Crypton v1.0

Crypton v1.0 was introduced by Chae Hoon Lim in [Li99] to modify the initial key schedule and improve the S-boxes.

Instead of using two S-boxes like the initial version, Crypton v1.0 uses four S-boxes S_0, S_1, S_2, S_3 which verify $S_0^{-1} = S_2$ and $S_1^{-1} = S_3$. The γ_o and γ_e transformations are redefined as follows :

$$B = \gamma_o(A) \Leftrightarrow b_{i,j} = S_{i+j \bmod 4}(a_{i,j})$$
$$B = \gamma_e(A) \Leftrightarrow b_{i,j} = S_{i+j+2 \bmod 4}(a_{i,j})$$

We still have : $\gamma_o^{-1} = \gamma_e$ and $\gamma_e^{-1} = \gamma_o$ and thus the encryption and decryption processes are still identical.

The new S-boxes S_0, S_1, S_2, S_3 are all designed from an 8x8 involutive S-box S, chosen for its good diffusion properties. The purpose of the replacement of the Crypton S-boxes was to lower the number of the most probable low weight differential and linear characteristics, and thus to speed up the diffusion achieved by Crypton.

According to our computer experiments, these S-box modifications very significantly improve the resistance of Crypton against the stochastic attacks presented in Sections 4 and 5.

Let us first consider stochastic cryptanalysis using differential properties on 6 rounds of Crypton v1.0. We found that if input difference is the (49, 49) pair of Δ_1 values, then the probability that the output be a (δ_1, δ_2) pair with $(\delta_1, \delta_2) \in D_1 \times D_1$ is $2^{-151.23}$, taking into account all intermediate values with an alternation of elements of Δ_1 and Δ_2. The $2^{-151,23}$ value represents the best probability associated with 6 rounds of Crypton v1.0. It is significantly lower than the $2^{-112.62}$ best probability associated with 6 rounds of Crypton.

For stochastic cryptanalysis using a partition of blocks in 16 classes, the number of plaintexts required for a 1-R attack is at least 2^{135}. This figure was derived from Φ_1, and corresponds to the most favorable values of the 6-uple of 4-bit key words involved in the computations. It is significantly higher than the $N = 2^{104}$ minimal number of required plaintexts obtained for the initial Crypton, and higher than the number of distinct plaintexts (2^{128}).

Thus in summary Crypton v1.0 resists the stochastic attacks of Sections 4 and 5 much better than the initial version of Crypton. This seems to be a direct consequence of the design criteria of the new Crypton S-boxes. Because of the decrease of the number of low weight highest probability differential and linear characteristics at each round, the number of "high probability paths" that together form the transition probabilities considered in our stochastic attacks is decreased and the performance of the attacks is significantly affected. Changes in S-boxes proposed in Crypton v1.0 were very discerning.

7 Conclusion

We have described two stochastic attacks on the eight-round version of the Crypton block cipher which are faster than an exhaustive search and more efficient than the best attacks discovered so far.

The first of these two attacks is close to a truncated differential attack, but we believe that probability matrices computations are more precise than truncated differentials probabilities computations would be. It seems to us that an analysis based on truncated-differential probabilities would have led to an overestimate of the performance of the attack.

Our attacks do not threaten the security of Crypton in a full version, but nevertheless put the highlight on a (slight) diffusion weakness in the linear part of the Crypton round function. Even if finding the most relevant cryptographic criteria on the linear part of substitution-permutation blockciphers is still to a large extent an open issue, diffusion criteria related to the number of "active" S-boxes (e.g. MDS properties) offer some clues. Our attacks exploit the existence of low weigh (and iterative) relations between the input and the output of the linear part of Crypton, which lead to statistics involving only 4 active S-boxes per round.

Finally, the comparison between the results obtained on the initial Crypton and Crypton v1.0 confirms that the S-box modifications introduced in Crypton v1.0 significantly improve its resistance against some classes of attacks.

References

Ba99. O. Baudron, H. Gilbert, L. Granboulan, H. Handschuh, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, S. Vaudenay, "Report on the AES Candidates", *The Second Advanced Encryption Standard Candidate Conference,* N.I.S.T., 1999.

Hal99. C. D'Halluin, G. Bijnens, V. Rijmen, B. Preneel, "Attack on Six Rounds of Crypton". In *Fast Software Encryption - FSE'99* , p. 46, Springer Verlag, Rome, Italy, March 1999.

HG97. H. Handschuh, H. Gilbert, "χ^2 Cryptanalysis of SEAL Encryption Algorithm". In *Fast Software Encryption - FSE'97*, pp. 1-12, Springer Verlag, Haifa, Israel, 1997.

Li98. C.H. Lim, "Crypton : A New 128-bit Block Cipher", *The First Advanced Encryption Standard Candidate Conference,* N.I.S.T., 1998.

Li99. C.H. Lim, "A Revisited Version of Crypton : Crypton V1.0". In *Fast Software Encryption - FSE'99* , p. 31, Springer Verlag, Rome, Italy, March 1999.

Ma93. M. Matsui, "Linear Cryptanalysis Method for DES Cipher". In *Advances in Cryptology - Eurocrypt'93*, pp. 386-396, Springer Verlag, Lofthus, Norway, 1993.

LM91. X. Lai, J.L.Massey and S. Murphy, "Markov Ciphers and Differential Cryptanalysis". In *Advances in Cryptology - Eurocrypt'91*, p. 17, Springer Verlag, Brighton, UK, 1991.

HM97. C. Harpes and J.L.Massey, "Partitioning Cryptanalysis". In *Fast Software Encryption - FSE'97* , p. 13, Springer Verlag, Haifa, Israel, January 1997.

Mu. S. Murphy, F. Piper, M. Walker, P. Wild, "Likehood Estimation for Block Cipher Keys". Unpublished.

Va95. S. Vaudenay, "La sécurité des Primitives Cryptographiques". Doctoral Dissertation, 1995.

A Matrix of Transition Biases for the Φ Value

	0	1	2	3	4	5	6	7
0	204800	-36864	124928	-28672	-53248	77824	-18432	69632
1	-28672	-83968	-4096	-94208	-94208	63488	-118784	36864
2	-55296	53248	-28672	45056	112640	-94208	53248	-86016
3	-69632	102400	-94208	75776	28672	-45056	53248	-51200
4	-114688	53248	-112640	45056	-36864	-94208	6144	-86016
5	-20480	67584	-12288	110592	143360	-47104	135168	-53248
6	34816	-36864	77824	-28672	-92160	77824	-102400	69632
7	86016	-86016	77824	-92160	-45056	28672	-36864	67584
8	28672	-135168	14336	-126976	-57344	12288	-71680	4096
9	102400	-2048	94208	-12288	-61440	145408	-53248	118784
10	-55296	36864	-61440	28672	112640	-77824	86016	-69632
11	-86016	102400	-77824	75776	45056	-45056	36864	-51200
12	-77824	118784	-59392	110592	106496	4096	116736	12288
13	-53248	-14336	-77824	28672	12288	-129024	36864	-135168
14	34816	-53248	45056	-45056	-92160	94208	-69632	86016
15	69632	-86016	94208	-92160	-28672	28672	-53248	67584

	8	9	10	11	12	13	14	15
0	4096	-77824	-30720	-86016	-114688	36864	-116736	45056
1	143360	-47104	151552	-36864	-20480	100352	-28672	61440
2	-75776	94208	-69632	102400	51200	-53248	12288	-61440
3	-45056	61440	-53248	22528	86016	-86016	94208	-79872
4	4096	94208	-14336	102400	106496	-53248	161792	-61440
5	-126976	30720	-102400	53248	4096	-83968	-20480	-77824
6	63488	-77824	118784	-86016	-38912	36864	-61440	45056
7	61440	-45056	36864	-38912	-102400	69632	-77824	96256
8	90112	-12288	88064	-20480	-36864	135168	-55296	143360
9	45056	-129024	20480	-118784	-86016	18432	-61440	-20480
10	-75776	77824	-102400	86016	51200	-36864	45056	-45056
11	-61440	61440	-36864	22528	102400	-86016	77824	-79872
12	-73728	-4096	-75776	4096	20480	-118784	43008	-126976
13	-61440	112640	-69632	135168	102400	-2048	110592	4096
14	63488	-94208	86016	-102400	-38912	53248	-28672	61440
15	45056	-45056	53248	-38912	-86016	69632	-94208	96256

Table 1. Matrix of distribution for S_1 and Φ_0

A collision attack on 7 rounds of Rijndael

Henri Gilbert and Marine Minier

France Télécom R & D
38-40, rue du Général Leclerc
92794 Issy les Moulineaux Cedex 9 - France
email : henri.gilbert@cnet.francetelecom.fr

Abstract

Rijndael is one of the five candidate blockciphers selected by NIST for the final phase of the AES selection process. The best attack of Rijndael so far is due to the algorithm designers ; this attack is based upon the existence of an efficient distinguisher between 3 Rijndael inner rounds and a random permutation, and it is limited to 6 rounds for each of the three possible values of the keysize parameter (128 bits, 196 bits and 256 bits). In this paper, we construct an efficient distinguisher between 4 inner rounds of Rijndael and a random permutation of the blocks space, by exploiting the existence of collisions between some partial functions induced by the cipher. We present an attack based upon this 4-rounds distinguisher that requires 2^{32} chosen plaintexts and is applicable to up to 7-rounds for the 196 keybits and 256 keybits version of Rijndael.
Since the minimal number of rounds in the Rijndael parameter settings proposed for AES is 10, our attack does not endanger the security of the cipher, indicate any flaw in the design or prove any inadequacy in selection of number of rounds. The only claim we make is that our results represent improvements of the previously known cryptanalytic results on Rijndael.

1 Introduction

Rijndael [DaRi98], a blockcipher designed by Vincent Rijmen and Joan Daemen, is one of the 5 finalists selected by NIST in the Advanced Encryption Standard competition [AES99]. It is a variant of the Square blockcipher, due to the same authors [DaKnRi97]. It has a variable block length b and a variable key length k, which can be set to 128, 192 or 256 bits. The recommended nr number of rounds is determined by b and k, and varies between 10 and 14. In the sequel we will sometimes use the notation Rijndael/b/k/nr to refer to the Rijndael variant determined by a particular choice of the b, k and nr parameters.

The best Rijndael attack published so far is due to the algorithm designers [DaRi98]. It is a variant of a the "Square" attack, and exploits the byte-oriented structure of Rijndael [DaKnRi97]. This attack is based upon an efficient distinguisher between 3 Rijndael inner rounds and a random permutation. It is stated in [DaRi98] that "for the different block lengths of Rijndael no extensions to 7 rounds faster than exhaustive search have been found".
In this paper we describe an efficient distinguisher between 4 Rijndael inner rounds and a random permutation, and we present resulting 7-rounds attacks of Rijndael/b=128 which are substantially faster than an exhaustive key search for the $k = 196$ bits and $k = 256$ bits versions and marginally faster than an exhaustive key search for the $k = 128$ bits version.

This paper is organised as follows. Section 2 provides an outline of the cipher. Section 3 investigates partial functions induced by the cipher and the existence of collisions between such partial functions, and describes a resulting distinguisher for 4 inner rounds. Section 4 presents 7-rounds attacks based on the 4-rounds distinguisher of Section 3. Section 5 concludes the paper.

2 An outline of Rijndael/$b = 128$

In this Section we briefly described the Rijndael algorithm. We restrict our description to the b=128 bits blocksize and will consider no other blocksize in the rest of this paper.
Rijndael/$b/k/nr$ consists of a key schedule and an iterated encryption function with nr rounds. The key schedule derives $nr + 1$ 128-bit round keys K_0 to K_{nr} from the $k = 128, 196$ or 256 bits long Rijndael key K. Since attacks presented in the sequel do not use the details of the dependence between round keys, we do not provide a description of the key schedule.
The Rijndael encryption function is the composition of nr block transformations. The current 128-bit block value B is represented by a 4×4 matrix :

$$B = \begin{array}{|c|c|c|c|} \hline b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ \hline b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ \hline b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ \hline b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \\ \hline \end{array}$$

The definition of the round functions involves four elementary mappings :

- the σ=ByteSub byte substitution transforms each of the 16 input bytes under a fixed byte permutation P (the Rijndael S-box).
- the ρ=ShiftRow rows shift circularly shifts row i ($i = 0$ to 3) in the B matrix by i bytes to the right.
- the μ=MixColumn is a matrix multiplication by a fixed 4×4 matrix of non-zero GF(2^8) elements.
- the κ_r=KeyAddition is a bitwise addition with a 128-bit round key K_r.

The Rijndael cipher is composed by an initial round key addition κ_0, $nr-1$ inner rounds and a final transformation. The rth inner round ($1 \leq r \leq nr-1$) is defined as the $\kappa_r \circ \mu \circ \rho \circ \sigma$ function. The final transformation at the round nr is an inner round without MixColumn mapping : FinalRound = $\kappa_{nr} \circ \rho \circ \sigma$. We can thus summarise the cipher as follows:

B:=$\kappa_0(B)$;
For $r = 1$ to $nr-1$
  B:=InnerRound(B);
FinalRound(B);

Remarks :

- σ is the single non $GF(8)$-linear function of the whole cipher.
- The Rijndael S-box P is the composition of the multiplicative inverse function in GF(8) (NB : '00' is mapped into itself) and a fixed GF(2)-affine byte transformation. If the affine part of P was omitted, algebraic methods (e.g. interpolation attacks) could probably be considered for the cryptanalysis of Rijndael.
- The $\mu \circ \rho$ linear part of Rijndael appears to have been carefully designed. It achieves a full diffusion after 2 rounds, and the Maximum Distance Separability (MDS) property of μ prevents good differential or linear "characteristics" since it ensures that two consecutive rounds involve many active S-boxes.

3 Distinguishing 4 inner rounds of Rijndael/b=12 from a random permutation

3.1 Notation

Figure 1 represents 4 consecutive inner round functions of Rijndael associated with any 4 fixed unknown 128-round keys. Y, Z, R, S represent the input blocks of the 4 rounds and T represents the output of the 4th round. We introduce short notations for some particular bytes of Y, Z, R, S, T, which play a particular role in the sequel : $y = Y_{0,0}$, $z_0 = Z_{0,0}$, $z_1 = Z_{1,0}$, $z_2 = Z_{2,0}$, $z_3 = Z_{3,0}$, and so on. Finally we denote by c the ($c_0 = Y_{1,0}$, $c_1 = Y_{2,0}$, $c_2 = Y_{3,0}$) triplet of Y bytes.

Let us fix all the Y bytes but y to any 11-uple of constant values. So the c triplet is assumed to be equal to a constant $c = (c_0, c_1, c_2)$ triplet, and the 12 $Y_{i,j}$, i=1 to 3, j=0 to 3 are also assumed to be constant. The Z, R, S, T bytes z_0 to z_3, r_0 to r_3, s, and t_0 to t_3 introduced in Figure 1 can be seen as c-dependent functions of the y input byte. In the sequel we sometimes denote by $z_0^c[y]$ to $z_3^c[y]$, $r_0^c[y]$ to $r_3^c[y]$, $s^c[y]$, $t_0^c[y]$ to $t_3^c[y]$ the z_i, r_i, s, t_i byte value associated with a c constant and one $y \in 0..255$ value.

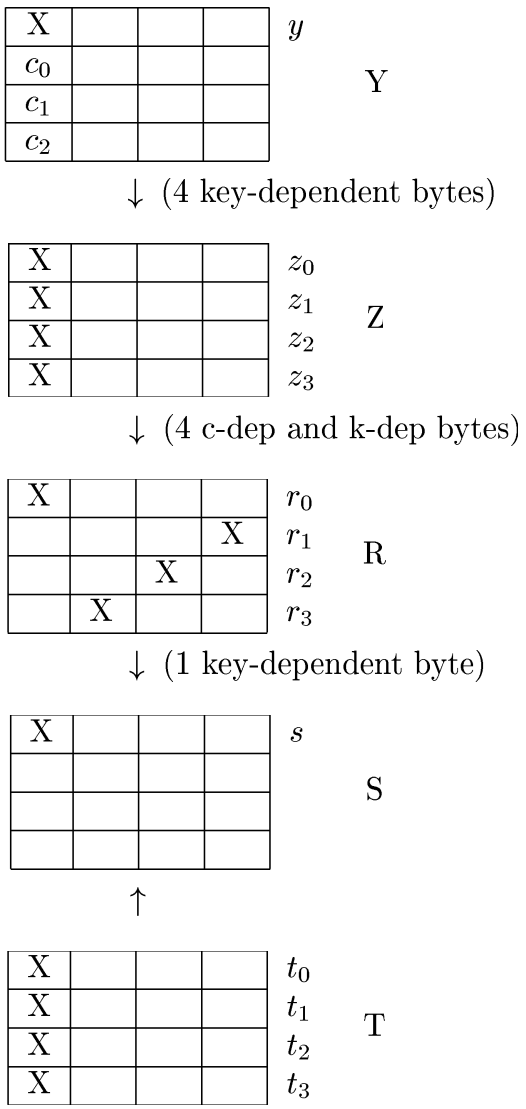


Figure 1: 4 inner rounds of Rijndael

3.2 The 3-rounds distinguisher used in the Rijndael/b=128 designers' attack

The Rijndael designers' attack is based upon the observations that :
- bytes z_0 to z_3 are one to one functions of y and the other Z bytes are constant.
- bytes r_0 to r_3 are one to one functions of y (as well as the 12 other R bytes).
- s is the XOR of four one to one functions of y and thus $\sum_{y=0}^{255} s[y] = 0$.

Thus 3 consecutive inner rounds of Rijndael have the distinguishing property that if all Y bytes but y are fixed and y is taken equal to each of the 256 possible values, then the sum of the 256 resulting s values is equal to zero.

This leads to a 6-rounds attack (initial key addition followed by 5 inner rounds followed by final round). As a matter of fact an initial round (i.e. an initial key addition followed by 1 inner round) can be added on top, at the expense of testing assumptions on 4 key bytes of the initial key addition. Moreover, two additional rounds can be added at the end (namely one additional inner round followed by one final round), at the expense of testing assumptions on 4 final round key bytes. Combining both extensions provides an attack which requires 2^{32} plaintexts and has a complexity of 2^{72} encryptions.

3.3 A 4-rounds distinguisher for Rijndael/$b = 128$

We now analyse in detail the dependency of the byte oriented functions introduced in Section 3.1 in the c constant and the expanded key. We show that the $s^c[y]$ function is entirely determined by a surprisingly small number of unknown bytes, which either only depend upon the key or depend upon both the key and the c value, and that as a consequence there exist (c', c'') pairs of distinct c values such that the $s^{c'}[\cdot]$ and $s^{c''}[\cdot]$ partial functions collide, i.e. $s^{c'}[y] = s^{c''}[y]$ for $y = 0, 1, \cdots, 255$. This provides an efficient test for distinguishing 4 inner rounds of Rijndael from a random permutation.

The construction of the proposed distinguisher is based upon the following observations, which are illustrated in Figure 1.

Property 1 : At round 1, the $y \rightarrow z_0^c[y]$ one to one function is independent of the value of the c triplet and is entirely determined by one key byte. The same property holds for z_1, z_2, z_3. This is because at the output of the first round ShiftRow the c_0 to c_2 constants only affect columns 1 to 3 of the current block value, whereas the z_0 to z_3 bytes entirely depend upon column 0. For similar reasons, the other bytes of Z are independent of y : each of the bytes of column 1 (resp 2, resp 3) of Z is entirely determined by the c_0 (resp c_1, resp c_2) byte and one key-dependent byte.
More formally, there exist 16 MixColumn matrix coefficients $a_{i,j}$,i=0..3, j=0..3 and 16 key-dependent constants $b_{i,j}$, i=0..3, j=0..3 such that $z_i = a_{i,0}P(y) + b_{i,0}$, i=0..3 and $z_{i,j} = a_{i,0}P(c_{j-1}) + b_{i,j}$, i=1..3, j=0..3.

Property 2 : At round 2, each of the four bytes $r_i[y]$, $i = 0..3$ is a one to one function of $z_i[y]$, and the $r_i[y] \rightarrow z_i[y]$ is entirely determined by one single unknown constant byte that is entirely determined by c and the key.

More formally, there exist 16 MixColumn coefficients α_i, $i = 0..3$, β_i, $i = 0..3$ γ_i, $i = 0..3$ and δ_i, $i = 0..3$ and 4 key-dependent constants ϵ_i, $i = 0..3$ such that $r_i = \alpha_i \cdot P(z_{i,0}) + \beta_i \cdot P(z_{i,1}) + \gamma_i \cdot P(z_{i,2}) + \delta_i \cdot P(z_{i,3}) + \epsilon_i$, $i = 0..3$. The r_i bytes are thus related to c and y by the relations : $r_i = \alpha_i \cdot P(a_{i,0}P(y) + b_{i,0}) + \beta_i \cdot P(a_{i,1}P(c_0) + b_{i,1}) + \gamma_i \cdot P(a_{i,2}P(c_1) + b_{i,2}) + \delta_i \cdot P(a_{i,3}P(c_2) + b_{i,3}) + \epsilon_i$, $i = 0..3$.
Consequently, the $r_0[y]$ to $r_3[y]$ one to one functions of y are entirely determined by the 4 key-dependent constant unknown bytes $b_{i,0}$ introduced in property (1) and the 4 c- and k-dependent bytes $b_i = \beta_i \cdot P(a_{i,1}P(c_0) + b_{i,1}) + \gamma_i \cdot P(a_{i,2}P(c_1) + b_{i,2}) + \delta_i \cdot P(a_{i,3}P(c_2) + b_{i,3}) + \epsilon_i$, i$= 0 \cdot \cdot 3$.

Property 3 : At round 3, the s byte can be expressed as a function of the r_0 to r_3 bytes and one c-independent and key-dependent unknown constant. Consequently, the $s^c[y]$ function is entirely determined by 4 key-dependent and c-dependent constants and 5 c-independent and key-dependent constants.

Property 4 : Let us consider the decryption of the fourth inner round : s can be expressed as $s = p^{-1}[(0E.t_0 + 0B.t_1 + 0D.t_2 + 09.t_3) + k_5]$ where p represents the single S-box. In other words $0E.t_0 + 0B.t_1 + 0D.t_2 + 09.t_3$ is a one to one function of s, and that function is entirely determined by one single key byte k_5. Thus $0E.t_0 + 0B.t_1 + 0D.t_2 + 09.t_3$ is a function of y that is entirely determined by 6 unknown bytes which only depend upon the key and by 4 additional unknown bytes which depend both upon c and the key.

The above properties provide an efficient 4-rounds distinguisher. We can restate property (3) in saying that the $s^c[y]$ function is entirely determined (in a key-dependent manner) by the 4 c-dependent bytes b_0 to b_3. Let us make the heuristic assumption that these 4 unknown c-dependent bytes behave as a random function of the c triplet of bytes. By the birthday paradox, given a C set of about 2^{16} c triplet values, there exist with a non negligible probability two distinct c' and c'' in C such that the $s^{c'}[y]$ and $s^{c''}[y]$ functions induced by c' and c'' are equal (i.e. in other words such that the $(s^{c'}[y])_{y=0..255}$ and $(s^{c''}[y])_{y=0..255}$ lists of 256 bytes are equal). Property (4) provides a method to test such a "collision", using the t_0 to t_3 output bytes of 4 inner rounds : c' and c'' collide if and only if $\forall y \in [0, ..., 255]$, $0E.t_0^{c'} + 0B.t_1^{c'} + 0D.t_2^{c'} + 09.t_3^{c'} = 0E.t_0^{c''} + 0B.t_1^{c''} + 0D.t_2^{c''} + 09.t_3^{c''}$. Note that it is sufficient to test the above equality on a limited number of y values (say 16 for instance) to know with a quite negligible "false alarms" probability whether the $s^{c'}[y]$ and $s^{c''}[y]$ functions collide.

We performed some computer experiments which confirmed the existence, for arbitrarily chosen key values, of (c', c'') pairs of c value such that the $s^{c'}[y]$ and $s^{c''}[y]$ functions collide. For some key values, we could even find four byte values c'_1, c'_2, c''_1 and c''_2 such that for each of the 256 possible values of the

c_0 byte, the $s^{c'}[y]$ and $s^{c''}[y]$ functions associated with the $c' = (c_0, c'_1, c'_2)$ and $c'' = (c_0, c''_1, c''_2)$ triplets of bytes collide. This stronger property, which is rather easy to explain using the expression of the b_i constants introduced in Property (2), is not used in the sequel.

The proposed 4 rounds distinguisher uses the collision test derived from property (4) is the following manner :

- select a C set of about 2^{16} c triplet values and a subset of $\{0..255\}$, say for instance a Λ subset of 16 y values.
- for each c triplet value, compute the $L_c = (0E.t_0^c + 0B.t_1^c + 0D.t_2^c + 09.t_3^c)_{y\in\Lambda}$. We claim that such a computation of 16 linear combinations of the outputs represents substantially less than one single Rijndael operation.
- check whether two of the above lists, $L_{c'}$ and $L_{c''}$ are equal. The 4 round distinguisher requires about 2^{20} chosen inputs Y, and since the collision detection computations (based on the analysis of the corresponding T values) require less operations than the 2^{20} 4-inner rounds computations, the complexity of the distinguisher is less than 2^{20} Rijndael encryptions.

Note that property (4) also provides another method to distinguish 4 inner round from a random permutation, using $N \leq 256$ plaintexts and 2^{80} N operations, namely performing an exhaustive search of the 10 unknown constants considered in property (4). Note that a value such as $N = 16$ is far sufficient in practice. However, we only consider in the sequel the above described birthday test, which provides a more efficient distinguisher.

4 An attack of the 7-rounds Rijndael/b=128 cipher with 2^{32} chosen plaintexts

In this Section we show that the 4 inner rounds distinguisher of Section 3 provides attacks of the 7-rounds Rijndael for the b=128 blocksize and the various keysizes. We present two slightly different attacks. The first one (cf Section 4.2 hereafter) is substantially faster than an exhaustive search for the k=196 and k=256 keysizes, but slower than exhaustive search for the k=128 bits keysize. The second attack (cf Section 4.2) is dedicated to the k=128 keysize, and is marginally faster than an exhaustive search for that keysize.

The 7-rounds Rijndael is depicted at Figure 2. X represents a plaintext block, and V represents a ciphertext block. In Figure 2 the 4 inner rounds of Figure 1 are surrounded by one initial $X \rightarrow Y$ round (which consists of an initial key addition followed by one round), and two final rounds (which consist of one $T \rightarrow U$ inner round followed by an $U \rightarrow V$ final round).

Our attack method is basically a combination of the 4-round distinguisher presented in Section 3 and an exhaustive search of some keybytes (or combinations of keybytes) of the initial and the two final rounds. In the attack of Section

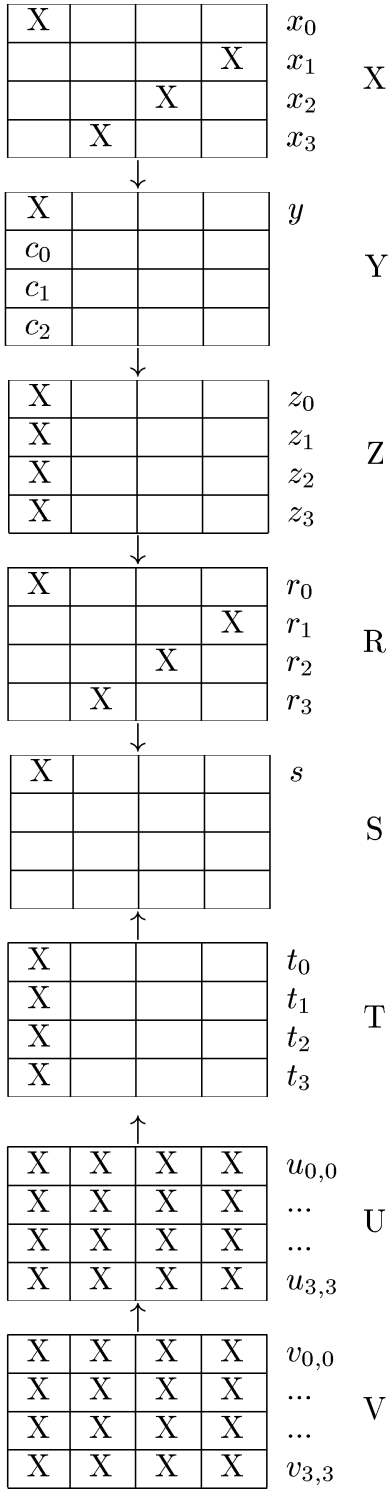


Figure 2: 7-rounds Rijndael

4.1 we are using the property that in the equations provided by the 4-rounds distinguisher there is a variables separations in terms which involve one half of the 2 last rounds key bytes and terms which involve a second half of the 2 last round key bytes in order to save a 2^{80} factor in the exhaustive search complexity. In the attack of Section 4.2, we are using precomputations on colliding pairs of c values to test each 128-bits key assumption with less operations than one single Rijndael encryption.

4.1 An attack of the 7-rounds Rijndael/b=128/k=196 or 256 with 2^{32} chosen plaintexts and a complexity of about 2^{140}

We now explain the attack procedure in some details, using the notation introduced in Figure 2. We fix all X bytes except the four bytes x_0 to x_3 equal to 12 arbitrary constant values. We encrypt the 2^{32} plaintexts obtained by taking all possible values for the x_0 to x_3 bytes, thus obtaining 2^{32} V ciphertext blocks. We are using the two following observations :

Property 5 : If the 4 key bytes added with the x_0 to x_3 bytes in the initial key addition are known (let us denote them by $k_{ini} = (k_0, k_1, k_2, k_3)$, then it is possible to partition the 2^{32} plaintexts in 2^{24} subsets of 256 plaintext values satisfying the conditions of Section 3, i.e. such that the corresponding 256 Y values satisfy the following conditions :
- the y byte takes 256 distinct values (which are known up to an unknown constant first round key byte which is not required for the attack).
- the $c = (c_0, c_1, c_2)$ triplet of bytes is constant ; moreover, each of the 2^{24} subsets corresponds to a distinct c value (the c value corresponding to each subset is known up to three constant first round keybytes which are not required for the attack).
- the 12 other Y bytes are constant and their constant values $Y_{i,j}$ for i=1..3 and j=0..3 is the same for all subsets.
Note that the same property is used in the Rijndael designers' attack.

Property 6 : Each of the t_0, t_1, t_2, t_3 bytes can be expressed as a function of four bytes of the V ciphertext and five unknown key bytes (i.e. 4 of the final round key bytes and one linear combination of the penultimate round key bytes). Therefore, we can "split" the $t^c[y] =' 0E'.t_0^c[y] +' 0B'.t_1^c[y] +' 0D'.t_2^c[y] +' 09'.t_3^c[y]$ combination of the four $t_i^c[y]$ bytes considered in the 4-rounds distinguisher as the XOR of two terms $\tau_1^c[y]$ and $\tau_2^c[y]$ which can both be expressed as a function of 8 ciphertext bytes and 10 unknown key bytes, namely $\tau_1 =' 0E'.t_0^c[y] +' 0B'.t_1^c[y]$ and $\tau_2 =' 0D'.t_2^c[y] +' 09'.t_3^c[y]$. We denote in the sequel by k_{τ_1} those 10 unknown keybytes which allow to derive τ_1 from 8 bytes of the V ciphertext, and by k_{τ_2} those 10 keybytes which allow to derive τ_1 from 8 bytes of the V ciphertext.

We perform an efficient exhaustive search of the k_{ini}, k_{τ_1} and k_{τ_2} keys in the following way :

- For each of the 2^{32} possible k_{ini} assumptions, we can partition the set of the 256^4 possible X values in 256^3 subsets of 256 X values each, according to the value of the c constant, and select say 256^2 of these 256^3 subsets. Thus each of the 256^2 selected subsets is associated with a distinct value of the c constant. Note that the c value associated with a subset and the y values associated with each of the X plaintexts of a subset are only known up to unknown keybits, but this does not matter for our attack. We can denote by $c*$ and $y*$ the known values which only differ from the actual values by fixed unknown key bits.

- Now for each subset associated with a $c*$ constant triplet, based on the say 16 ciphertexts associated with the $y* = 0$ to $y* = 15$ values, we can precompute the $(\tau_1^c(y))_{y*=0..15}$ 16-tuple of bytes for each of the 2^{80} possible k_{τ_1} keys. We can also precompute the $(\tau_2^c(y))_{y*=0..15}$ 16-tuple for each of the 2^{80} possible k_{τ_2} keys.

Based on this precomputation, for each $(c'*, c''*)$ pair of distinct c* values :

- We precompute a (sorted) table the $(\tau_1^{c'}(y) \oplus \tau_1^{c''}(y))_{y*=0..15}$ 16-tuple of bytes for each of the 2^{80} possible k_{τ_1} keys (the computation of each 16-tuple just consists in xoring two precomputed values)

- For each of the 2^{80} possible values of the k_{τ_2} key, we compute the $(\tau_2^{c'}(y) \oplus \tau_2^{c''}(y))_{y*=0..15}$ 16-tuple of bytes associated with the observed ciphertext, and check whether this t-uple belongs to the precomputed table of 16-tuple $(\tau_1^{c'}(y) \oplus \tau_1^{c''}(y))_{y*=0..15}$. If for a given k_{τ_1} value there exists a k_{τ_2} value such that $(\tau_1^{c'}(y) \oplus \tau_1^{c''}(y))_{y*=0..15} = (\tau_2^{c'}(y) \oplus \tau_2^{c''}(y))_{y*=0..15}$, (i.e. $t^{c'}[y] = t^{c''}[y]$ for each of the y values associated with $y* = 0$ to 16, this represents an alarm). The equality between the t bytes associated with c' and c'' is checked for the other $y*$ values. If the equality is confirmed, this means that a collision between the $s^c[y]$ functions associated with c' and c''. This provides 20 bytes of information on the last and penultimate key values, since with overwhelming probability, the right values of k_{ini}, k_{τ_1} and k_{τ_2} have then been found.

Since the above procedure tests whether the exist collisions inside a random set of 256^2 of the 256^4 possible $s^c[y]$ functions, the probability of the procedure to result in a collision, and thus to provide k_{ini}, k_{τ_1} and k_{τ_2} is high (say about 1/2). In other words, the success probability of the attack is about 1/2.

Once k_{ini}, k_{τ_1} and k_{τ_2} have been found, the 16-bytes final round key is entirely determined and the final round can be decrypted, so one is left with the problem of cryptanalysing the 6-round version of Rijndael. One might object that the last round of the left 6-round cipher is not a final round, but an inner round. However, it is easy to see that by applying a linear one to one change of variable to U and the 6th round key (i.e. replacing U by a U' linear function of U and K_6 by a linear function K'_6 of K_6), the last round can be represented as a final round (i.e. U' is the image of T by the final round associated with

K'_6). Thus we are in fact left with the cryptanalysis of the 6-round Rijndael, and the last round subkey is easy to derive. The process of deriving the subkeys of the various rounds can then be continued (using a subset of the 2^{32} chosen plaintexts used for the derivation of k_{ini}, k_{τ_1} and k_{τ_2}), with negligible additional complexity, until the entire key has eventually been recovered.

4.2 An attack of the 7-rounds Rijndael/b=128/k=128 2^{32} chosen plaintext

We now outline a variant of the former attack that is dedicated to the k=128 bits version of Rijndael and is marginally faster than an exhaustive search. This attack requires a large amount of precomputations.

As a matter of fact, it can be shown that the 4 c-dependent bytes that determine the mapping between four $z_i^c[y]$ bytes and the four $r_i^c[y]$ are entirely determined by 12 key-dependent (and c-independent) bytes. For each of the 256^{12} possible values of this $\phi(K)$ 12-tuple of bytes, we can compute colliding c' and c'' triplets of bytes (this can be done performing about 256^2 partial Rijndael computations corresponding to say 256^2 distinct c values and looking for a collision. One can accept that no collision be found for some $\phi(K)$ values : this just means that the attack will fail for a certain fraction (say 1/2) of the key values. We store c' and c'' (if any) in a table of 256^{12} $\phi(K)$ entries.

Now we perform an exhaustive search of the K key. To test a key assumption, we compute the k_{ini}, k_{τ_1}, k_{τ_2} and $\phi(K)$ values. Then we find the (c', c'') pair of colliding c values in the precomputed table, compute the two associated $c'*$ and c'' values, determine which two precomputed lists $(V^{c'}[y])_{y*=0..15}$ $(V^{c''}[y])_{y*=0..15}$of 16 ciphertext values each are associated with $c'*$ and $c''*$, and finally compute the associated $(t^{c'}[y])_{y*=0..15}$ and $(t^{c''}[y])_{y*=0..15}$ bytes by means of partial Rijndael decryption. The two values associated with $y* = 0$ are first computed and compared. The two values associated with $y* = 1$ are only computed and compared if they are equal, etc, thus in average only two partial decryption are performed. It the two lists of 16 t bytes are equal, then there is an alarm, and the K is further tested with a few plaintexts and ciphertexts.

We claim than the complexity of the operations performed to test one K key is marginally less than one Rijndael encryption.

5 Conclusion

We have shown that the existence of collisions between some partial byte oriented functions induced by the Rijndael cipher provides a distinguisher between 4 inner rounds of Rijndael and a random permutation, which in turn enables to mount attacks on a 7-rounds version of Rijndael for any key-length.
Unlike many recent attacks on block ciphers, our attacks are not statistical in

nature. They exploit (using the birthday paradox) a new kind of cryptanalytic bottleneck, namely the fact that a partial function induced by the cipher (the $s^c[y]$ function) is entirely determined by a remarkably small number of unknown constants. Therefore, unlike most statistical attacks, they require a rather limited number of plaintexts (about 2^{32}). However, they are not practical because of their high computational complexity, and do not endanger the full version of Rijndael. Thus we do not consider they represent arguments against Rijndael in the AES competition.

References

[AES99] http://www.nist.gov/aes

[DaKnRi97] J. Daemen, L.R. Knudsen and V. Rijmen, "The Block Cipher Square". In *Fast Software Encryption - FSE'97* , pp. 149-165, Springer Verlag, Haifa, Israel, January 1997.

[DaRi98] J. Daemen, V. Rijmen, "AES Proposal : Rijndael", *The First Advanced Encryption Standard Candidate Conference,* N.I.S.T., 1998.

New Results on the Pseudorandomness of Some Blockcipher Constructions

Henri Gilbert and Marine Minier

France Télécom R&D
38-40, rue du Général Leclerc
92794 Issy les Moulineaux Cedex 9
France

Abstract. In this paper, we describe new results on the security, in the Luby-Rackoff paradigm, of two modified Feistel constructions, namely the L-scheme, a construction used at various levels of the MISTY blockcipher which allows to derive a $2n$-bit permutation from several n-bit permutations, and a slightly different construction named the R-scheme. We obtain pseudorandomness and super-pseudorandomness proofs for L-schemes and R-schemes with a sufficient number of rounds, which extend the pseudorandomness and non superpseudorandomness results on the 4-round L-scheme previously established by Sugita [Su96] and Sakurai et al. [Sa97]. In particular, we show that unlike the 3-round L-scheme, the 3-round R-scheme is pseudorandom, and that both the 5-round L scheme and the 5-round R scheme are super pseudorandom (whereas the 4 round versions of both schemes are not super pseudorandom). The security bounds obtained here are close to those established by Luby and Rackoff for the three round version of the original Feistel scheme.

1 Introduction

A key dependent cryptographic function such as a blockcipher can be viewed as a random function associated with a randomly selected key value. It is generally defined using a recursive construction process. Each step of the recursion consists of deriving a random function (or permutation) f from r previously defined random functions (or permutations) $f_1, .., f_r$, and can be represented by a relation of the form $f = \Phi(f_1, .., f_r)$. The most studied example so far is the $f = \Psi(f_1, .., f_r)$ r-round Feistel construction, which allows to derive a $2n$-bit to $2n$-bit random permutation from r n-bit to n-bit functions. But there exist other well known constructions such as for instance Massey and Lai's alternative to the Feistel scheme used in IDEA [La90] and the constructions allowing to deduce a $2n$-bit permutation from several n-bit permutations used in Matsui's MISTY blockcipher [Ma93].

The strongest security requirement one can put on a f random function or permutation representing a key dependent cryptographic function is (informally speaking) that f be undistinguishable with a non negligible success probability from a perfect random function f^* or permutation c^*, even if a probabilistic

testing algorithm A of unlimited power is used for that purpose and if the q number of adaptively chosen queries of A to the random instance of f or f^* to be tested is large.

It is generally not possible to prove undistiguishability properties for "real life" cryptologic random function f and large q numbers of queries, because this would require a far too long key length. However, it is often possible to prove or disprove that if a random function f encountered at a given level of a cryptologic function construction is related to random functions encountered at the lower recursion level by a relation of the form $f = \Phi(f_1, .., f_r)$, then if we replace the actual f_1 to f_r random functions of the cipher by independent perfect random functions or permutations f_1^* to f_r^* (or, in a more sophisticated version of the same approach, by f_1' to f_r' functions which are sufficiently undistinguishable from f_1^* to f_r^*), the resulting modified f random function is undistinguishable from a random function (or permutation). This provides a useful method for assessing the soundness of blockcipher constructions. For instance, in the case of a three-round Feistel construction, a well known theorem first proved by Luby and Rackoff [Lu88] provides upper bounds on the $|p - p^*|$ advantage of any A testing algorithm in distinguishing the $f = \Psi(f_1^*, f_2^*, f_3^*)$ $2n$-bit random permutation deduced from three independent ideal random functions f_1^*, f_2^* and f_3^* from a $2n$-bit perfect random permutation c^* with q adaptively chosen queries to the tested instance of f or f^*. This advantage is bounded over by $\frac{q^2}{2^n}$.

The research on pseudorandomness properties of cryptographic constructions initiated Luby and Rackoff's seminal paper [Lu88] has represented a very active research study for the last decade. Just to mention a few examples, Zheng, Matsumoto and Imai and later on Sugita and Sakurai et al. investigated generalised Feistel constructions [Zh89],[Su96],[Su97], Patarin explicited the link between the best advantage of a q-queries distinguisher and the q-ary transition probabilities associated with f and proved undistinguihability bounds for numerous r-round Feistel constructions [Pa91], Maurer showed how to generalise undistinguishability results related to perfect random functions to undistinguishability results related to nearly perfect random functions (e.g. locally random functions)[Ma92], Bellare, Kilian, Rogaway et al. [Be94] investigated the application of similar techniques to modes of operation such as CBC MACs, Aiello and al. proved undistiguihability results on some parallelizable alternatives to the Feistel construction [Ai96] , Vaudenay embedded techniques for deriving undistinguishability bounds into a broader framework he named the decorrelation theory, and applied bounds provided by decorrelation techniques to proving the resistance of actual ciphers, e.g. DFC, against differential and linear cryptanalysis.

In this paper, we describe new results on the security of some blockcipher constructions in the above described paradigm, i.e. we investigate some $f = \Phi(f_1, .., f_k)$ constructions and upper bound the probability of distinguishing f from a perfect random function when Φ is applied to perfect random functions f_i^* or to perfect random permutations c_i^*. We consider alternatives to the Feistel construction allowing to derive a $2n$-bit permutation from several n-bit permutations, namely the so-called L-scheme and R-scheme constructions. The L-type

construction is used for instance at various levels of the construction of Matsui and al. Misty blockcipher [Ma93], as well as in the Kasumi variant of Misty recently adopted as the standard blockcipher for encryption and integrity protection in third generation mobile systems [Ka]. We obtain pseudorandomness and superpseudorandomness proofs for L-scheme and R-scheme constructions with a sufficient number of rounds, which extend the results on the pseudo randomness of the 4-round L-scheme previously established by Sugita [Su96] and Sakurai et al. [Sa97]. In particular, we show that unlike the 3-round L scheme, the 3-round R scheme is pseudorandom, and that both the 5-round L scheme and the 5-round R scheme are super pseudorandom (whereas the 4 round versions of both schemes are not super pseudorandom).

This paper organised as follows: Section 2 introduces basic definitions and useful general results on random functions and techniques for proving that two random functions are undistiguishable. Section 3 describes the R and L schemes. Sections 4 and 5 present our results on the pseudo-randomness and the superpseudorandomness of the L-scheme and the R-scheme respectively, for various numbers of rounds, and Section 6 concludes the paper.

2 Preliminaries

2.1 Notation

Through this paper we are using the following notation: I_n denotes the $\{0,1\}^n$ set. $F_{n,m}$ denotes the ${I_n}^{I_m}$ set of functions from I_n into I_m: thus $|F_{n,m}| = 2^{m.2^n}$. F_n denotes the $F_{n,n}$ set: thus $|F_n| = 2^{n.2^n}$. P_n denotes the set of permutations on I_n: thus $|P_n| = 2^n!$.

2.2 Random Functions

A random function of $F_{n,m}$ is defined as a random variable f of $F_{n,m}$, and can be viewed as a probability distribution $(Pr[f = \varphi])_{\varphi \in F_{n,m}}$ over $F_{n,m}$, or equivalently as a $(f_\omega)_{\omega \in \Omega}$ family of $F_{n,m}$ elements. In particular:

- A n-bit to m-bit key dependent cryptographic function is determined by a randomly selected key value $K \in \mathcal{K}$, and can thus be represented by the random function $f = (f_K)_{K \in \mathcal{K}}$ of $F_{n,m}$.
- A cryptographic construction of the form $f = \Phi(f_1, f_2, .., f_r)$ can be viewed as a random function of $F_{n,m}$ determined by r random functions $f_i \in F_{n_i,m_i}, i = 1..r$

Definition 1. *We define a perfect random function f^* of $F_{n,m}$ as a uniformly drawn element of $F_{n,m}$. In other words, f^* is associated with the uniform probability distribution over $F_{n,m}$. We define a c^* perfect random permutation on I_n as a uniformly drawn element of P_n. In other words, c^* is associated with the uniform probability distribution over P_n.*

Definition 2. *(t-ary transition probabilities associated with f). Given a random function f of $F_{n,m}$, we define the $Pr[x \stackrel{f}{\mapsto} y]$ transition probability associated with a x t-uple of I_n inputs and a y t-uple of I_m outputs as*

$Pr[x \stackrel{f}{\mapsto} y] = Pr[f(x_1) = y_1 \wedge f(x_2) = y_2 \wedge ... \wedge f(x_t) = y_t]$
$= Pr_{\omega \in \Omega}[f_\omega(x_1) = y_1 \wedge f_\omega(x_2) = y_2 \wedge ... \wedge f_\omega(x_t) = y_t]$

In the sequel we will use the following simple properties:

Property 1 *If f^* is a perfect random function $F_{n,m}$ and if $x = (x_1, ..., x_t)$ is a t-uple of pairwise distinct I_n values and if y is any t-uple of I_m values, then $Pr[x \stackrel{f^*}{\mapsto} y] = \frac{1}{|I_m|^t} = 2^{-m.t}$*

Property 2 *Let c^* be a perfect random permutation on I_n. If $x = (x_1, ..., x_t)$ is a t-uple of pairwise distinct I_n values $y = (y_1, ..., y_t)$ is a t-uple of pairwise distinct I_n values then $Pr[x \stackrel{c^*}{\mapsto} y] = (I_n - t)!/|In|! = \frac{(2^n - t)!}{(2^n)!}$*

Property 3 *Let c^* be a perfect random permutation on I_n. If x and x' are two distinct elements of I_n and δ is a given value of I_n then $Pr[c^*(x) \oplus c^*(x') = \delta] \leq \frac{2}{2^n}$.*

Proof: $\Pr[c^*(x) \oplus c^*(x') = 0] = 0$ since $x \neq x'$. If $\delta \neq 0$, $\Pr[c^*(x) \oplus c^*(x') = \delta] = \frac{2^n \cdot 2^{n-2} ... 1}{2^n!} = \frac{1}{2^n - 1} \leq \frac{2}{2^n}$. So, $\Pr[c^*(x) \oplus c^*(x') = \delta] \leq \frac{2}{2^n}$.

2.3 Distinguishing Two Random Functions

In proofs of security such as the one presented here, we want to upper bound the probability of any algorithm to distinguish whether a given fixed function φ is an instance of a random function $f = \Phi(f_1^*, f_2^*, .., f_r^*)$ of $F_{n,m}$ or an instance of the perfect random function f^*, using less than q queries to φ.

Let A be any distinguishing algorithm of unlimited power that, when input with a function φ of $F_{n,m}$ (which can be modeled as an " oracle tape " in the probabilistic Turing Machine associated with A) selects a fixed number q of distinct chosen or adaptively chosen input values X_i (the queries), obtains the q corresponding output values $Y_i = f(X_i)$, and based on these results outputs 0 or 1. Denote by p (resp by p^*) the probability for A to answer 1 when fed with a random instance of f (resp of f^*). We want to find upper bounds on the $Adv_A(f, f^*) = |p - p^*|$ advantage of A in distinguishing f from f^* in q queries.

As first noticed by Patarin [Pa91], the best $Adv_A(f, f^*)$ advantage of any A distinguishing algorithm for distinguishing f from f^* is entirely determined by the $Pr[x \stackrel{f}{\mapsto} y]$ q-ary transition probabilities associated with each $X = (X_1, \cdots, X_q)$ q-uple of pairwise distinct I_n values and each $Y = (Y_1, \cdots, Y_q)$ q-uple of I_m values. The following Theorem, which was first proved in [Pa91], and equivalent versions of which can be found in [Va99], is a very useful tool for deriving establishing upper bounds on the $Adv_A(f, f^*)$ based on properties of the $Pr[x \stackrel{f}{\mapsto} y]$ q-ary transition probabilities.

Theorem 1 *Let f be a random function of $F_{n,m}$ and f^* a perfect random function representing a uniformly drawn random element of $F_{n,m}$. Let q be an integer. Denote by $\mathcal{X}$ the ${I_n}^q$ set of all $X = (X_1, \cdots, X_q)$ q-tuples of pairwise distinct elements. If there exists a $\mathcal{Y}$ subset of ${I_m}^q$ and two positive real numbers ϵ_1 and ϵ_2 such that*

1) $|\mathcal{Y}| > (1 - \epsilon_1) \cdot |I_m|^q$ *(i)*

2) $\forall X \in \mathcal{X} \ \forall Y \in \mathcal{Y} Pr[X \stackrel{f}{\mapsto} Y] \geq (1 - \epsilon_2) \cdot \frac{1}{|I_m|^q}$ *(ii)*

then for any A distinguishing using q queries

$$Adv_A(f, f^*) \leq \epsilon_1 + \epsilon_2$$

In order to improve the selfreadability of this paper, a short proof of Theorem 1 is provided in appendix at the end of this paper.

3 Description of the L- and R-Schemes

We now describe two simple variants of the Feistel scheme, that we propose to name L-scheme and R-scheme, following the terminology proposed by Kaneko and al. in their paper on the provable security against differential and linear cryptanalysis of generalised Feistel ciphers [Ka97].

The L-scheme and R-scheme both allow to derive a 2n-bit to 2n-bit permutation from several n-bit to n bit permutations (not only n-bit to n-bit functions as in the Feistel scheme), using only one n-bit to n bit permutation per round.

The 1-round L-scheme is depicted in Figure 1. It transforms a c_1 permutation of I_n into the $\psi_L(c_1)$ permutation of I_{2n} defined by

$$\psi_L(c_1)(x^1, x^0) = (x^0, c_1(x^1) \oplus x^0)$$

The extension to r rounds is straightforward: the r-round L-scheme transforms r I_n permutations c_1 to c_r into the I_{2n} permutation defined by

$$\psi_L(c_1, c_2, ..., c_r) = \psi_L(c_r) \circ \cdots \circ \psi_L(c_1)$$

The L-scheme is used at several levels of the construction of the MISTY and KASUMI ciphers, namely the derivation of the so-called FI and FO functions, and also the upper level of the construction in the case of MISTY2. One remarkable feature of the r-round L-scheme is that two c_i permutations can be processed in paralell.

The 1-round R-scheme is depicted in Figure 1 too. It transforms a $c1$ permutation of I_n into the $\psi_R(c_1)$ permutation of I_{2n} defined by

$$\psi_R(c_1)(x^1, x^0) = (c_1(x^1) \oplus x^0, c_1(x^1))$$

The r-round R-scheme transforms r I_n permutations c_1 to c_r into the I_{2n} permutation defined by $\psi_R(c_1, c_2, ..., c_r) = \psi_R(c_r) \circ \cdots \circ \psi_R(c_1)$.

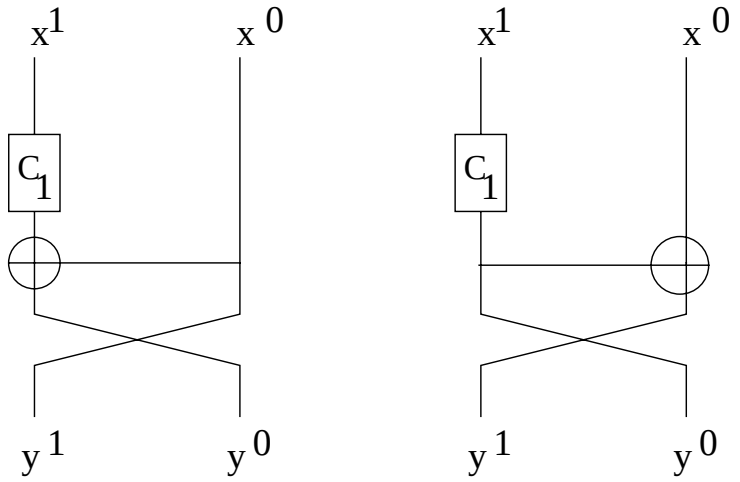


Fig. 1. L-scheme one round at left and R-scheme one round at right

In the sequel, we will several times consider the slightly simplified versions $\psi'_L(c_1, c_2, ..., c_r)$ and $\psi'_R(c_1, c_2, ..., c_r)$ of $\psi_L(c_1, c_2, ..., c_r)$ and $\psi_R(c_1, c_2, ..., c_r)$ obtained by omitting the XOR operation and the exchange of the left and right halves in the final round. We will sometimes analyse such simplified variants, whose pseudorandomness properties are obviously the same as those of the full r-round L or R scheme from which they are derived, instead of the full r-round L or R scheme, in order to simplify some discussions. We can notice that the $\psi'_R(c_1, c_2, ..., c_r)$ and $\psi'_L(c_r^{-1}, c_{r-1}^{-1}, ..., c_1^{-1})$ permutations are inverse of each other. This remark will be useful when it comes to analysing the super pseudorandomness properties of the L and R schemes.

Through the two next Sections, especially in proofs, we are using the following additional notation:

- I is an abreviation for the $(I_n)^q$ set.
- $I^{\neq} = ((I_n)^q))^{\neq}$ denotes the subset of $(I_n)^q$ consisting of all the q-tuples of pairwise distinct I_n values and $I^{=} = (I_n)^q \backslash I^{\neq}$.
- $\mathcal{X}$ denotes the subset of $(I_{2n})^q$ consisting of all $(x_1, \cdots, x_q)$ q-tuples of pairwise distinct I_{2n} values.
- $\mathcal{Y}$ will denote a subset of $(I_{2n})^q$ consisting of $(y_1, \cdots, y_q)$ q-tuples of I_{2n} values. The exact definition of $\mathcal{Y}$ will vary. This $\mathcal{Y}$ will be redefined in each Section where this notation is needed.

4 Analysis of the L-Scheme

In this Section, we compare, for various values of the r number of rounds of an L-scheme, the $f = \psi_L(c_1^*, c_2^*, ..., c_r^*)$ $2n$-bit random permutation deduced from r independent perfect random n-bit permutations $c_1^*, c_2^*, ..., c_r^*$ with a perfect $2n$-bit function f^* or a $2n$-bit perfect permutation c^*.

4.1 Three-Round L-Scheme: $\psi_L(c_1^*, c_2^*, c_3^*)$ Is Not a Pseudo-Random Function

As already noticed by several authors [Zh89], the function $f = \psi_L(c_1^*, c_2^*, c_3^*)$ associated with the three-round L-scheme is not pseudo-random.

Since the omission of the final XOR and the final exchange of the left and right output halves does not affect the pseudorandomness properties of f, we can consider the function $f = \psi'_L(c_1^*, c_2^*, c_3^*)$, instead of $\psi_L(c_1^*, c_2^*, c_3^*)$.

Let us show that 4 chosen input queries suffice to distinguish f from a the perfect random function f^* with a very large probability. Let us consider the encryption, under f, of two distinct $2n$-bit (x^1, x^0) plaintext blocks (a, b) and (a', b) which right halves are equal, and denote by (c, d) and (c', d') the two corresponding (y^1, y^0) ciphertext blocks. We can notice that $d \oplus d'$ is equal to $c_1^*(a) \oplus c_1^*(a')$, and thus independent of b. Therefore if we replace b by any other value b' and do the same computation as before, the new obtained value of $d \oplus d'$ will be left unchanged. This property allows to distinguish f from a perfect random function of I_{2n} with an advantage close to 1.

4.2 Four-Round L-Scheme: $\psi_L(c_1^*, c_2^*, c_3^*, c_4^*)$ Is a Pseudo-Random Function

As already established by Sakurai et al. [Sa97], the four-round version of the L-scheme is indistinguishable from a perfect pseudo-random function. In order for this paper to provide a self contained summary of the properties of the L and R schemes inside the security framework introduced in Section 2, we restate this result as follows.

Theorem 2 *Let n be an integer, c_1^*, c_2^*, c_3^*, c_4^* be four independent random function from I_n to I_n and f^* be the perfect random function on the I_{2n} set. Let $f = \psi_L(f_1^*, f_2^*, f_3^*, f_4^*)$ denote the random permutation associated with the four rounds of L-scheme. For any adaptative distinguisher $\mathcal{A}$ with q queries we have:*

$$Adv_{\mathcal{A}}^{q}(f, f^*) \leq \frac{7}{2} q^2 2^{-n}$$

A short proof for Theorem 2 is provided in appendix at the end of this paper. Since the proof technique is rather similar to the one used in the more detailed proof of Theorem 5 on the pseudorandomness of the 3-round R-scheme, we omitted some details in the proof of Theorem 2.

4.3 Four-Round L-Scheme: $\psi_L(c_1^*, c_2^*, c_3^*, c_4^*)$ Is Not a Super Pseudo-Random Permutation

As already established by Sakurai and al. [Sa97], the 4-round L-scheme does not provide a super pseudo random permutation, i.e. it is possible with a small number of encryption and decryption queries to distinguish $\psi_L(c_1^*, c_2^*, c_3^*, c_4^*)$ from a perfect random permutation.

Instead of providing here a direct proof of this property, let us show that this is a straightforward consequence of the fact (established in the next Section) that the 4-round R-scheme does not provide a super pseudo random function. As a matter of fact, $\psi'_R(c_1, c_2, c_3, c_4)$ and $\psi'_L(c_4^{-1}, c_3^{-1}, c_2^{-1}, c_1^{-1})$ are inverse of each other, as stated in Section 2 and therefore, the distinguisher for $\psi_R(c_1^*, c_2^*, c_3^*, c_4^*)$ can be converted in a distinguisher for $\psi_L(c_1^*, c_2^*, c_3^*, c_4^*)$ with the same number of queries and the same advantage.

4.4 Five-Round L-Scheme: $\psi_L(c_1^*, c_2^*, c_3^*, c_4^*, c_5^*)$ Is a Super Pseudo-Random Permutation

Recall that a super pseudo random distinguisher is an adaptative distinguisher which can call at one and the same time the cipher c and the cipher c^{-1}. The following Theorem shows that the five-round version of the L-scheme provides a super pseudorandom permutation.

Theorem 3 *Let n be an integer, c_1^*, c_2^*, c_3^*, c_4^*, c_5^* be five independent random functions from I_n to I_n and c^* be the perfect random permutation on the I_{2n} set. Let $c = \psi_L(c_1^*, c_2^*, c_3^*, c_4^*, c_5^*)$ denote the random permutation associated with the five round L scheme. For any adaptative super pseudorandom permutation distinguisher $\mathcal{A}$ with q queries, we have:*

$$Adv_{\mathcal{A}}^{q}(c, c^*) \leq \frac{9}{2} \cdot \frac{q^2}{2^n}$$

To prove this theorem, we need to use a variant of Theorem 1 due to Patarin in [Pa91] concerning permutations (that we provide here without proof):

Theorem 4 *Let m be an integer, ϵ be a positive real number, c be a random permutation on the $\{0,1\}^m$ set, c^* be the perfect random permutation on the same set. We denote by $\mathcal{X}$ the subset of $(X_1, \cdots, X_q)$ q-tuples that are pairwise distinct. Let $\mathcal{A}$ be any super pseudo random distinguisher with q queries. If $Pr[X \stackrel{c}{\mapsto} Y] \geq (1-\epsilon) \cdot \frac{1}{|I_m|^q}$ for all X and Y q-tuples in $\mathcal{X}$ then $Adv_{\mathcal{A}}^{q}(c, c^*) \leq \epsilon + \frac{q(q-1)}{2 \cdot 2^m}$*

Proof of Theorem 3: We will compare the $c = \psi'_L(c_1^*, c_2^*, c_3^*, c_4^*, c_5^*)$ permutation generator of Figure 2 (wich superpseudorandomness properties are exactly the same as for $\psi_L(c_1^*, c_2^*, c_3^*, c_4^*, c_5^*)$) with the perfect random permutation c^* of I_{2n}. For that purpose, let us consider any $X = (x_i^1, x_i^0) \in \mathcal{X}$ q-tuple of pairwise distinct values of I_{2n} and any $Y = (y_i^1, y_i^0)$ q-tuple of pairwise distinct values of I_{2n}. We want to establish lower bound on $\Pr[X \stackrel{c}{\mapsto} Y]$ and then apply Theorem 4 above. We are using the notation $x^2 = (x_i^2)_{i=1..q}$, $x^3 = (x_i^3)_{i=1..q}$, $x^4 = (x_i^4)_{i=1..q}$ to refer to the q-tuples of I_n intermediate words induced by the q considered f computations, at the locations marked in Figure 2.

$$\begin{aligned}\Pr[X \stackrel{c}{\mapsto} Y] = \sum_{x^2,x^3,x^4} \Pr[&(c_1^*(x^1) \oplus x^0 = x^2) \wedge (c_2^*(x^0) \oplus x^2 = x^3) \\ &\wedge (c_3^*(x^2) \oplus x^3 = x^4)\end{aligned}$$

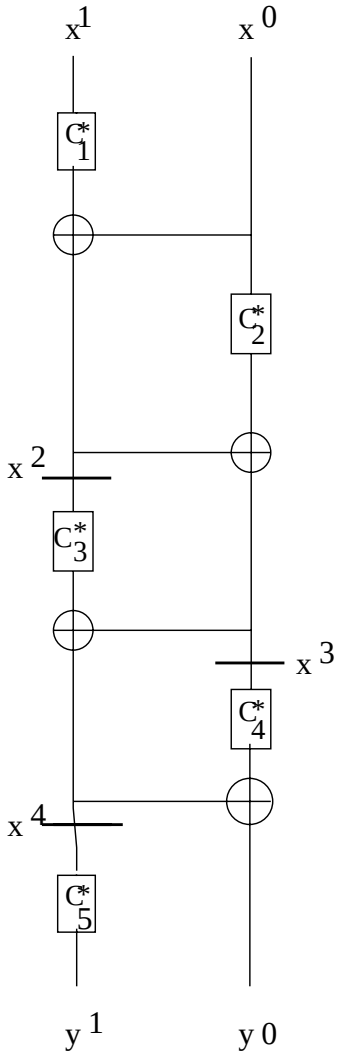


Fig. 2. L-scheme five rounds

$$\begin{aligned} & \wedge(c_4^*(x^3) \oplus x^4 = y^0) \wedge (c_5^*(x^4) = y^1)] \\ \geq & \sum_{x^2,x^3 \in I^{\neq}} \Pr[(c_1^*(x^1) \oplus x^0 = x^2) \wedge (c_2^*(x^0) \oplus x^2 = x^3)] \\ \cdot & \sum_{x^4} \quad \Pr\left[\begin{array}{c}(c_3^*(x^2) \oplus x^3 = x^4) \wedge \\ (c_4^*(x^3) \oplus x^4 = y^0) \wedge (c_5^*(x^4) = y^1)\end{array}\right] \end{aligned} \tag{1}$$

Let us consider any fixed x^2, x^3 q-tuples of $I^{\neq}$. In order to establish a lower bound on the $\sum_{x^4} \Pr[(c_3^*(x^2) \oplus x^3 = x^4) \wedge (c_4^*(x^3) \oplus x^4 = y^0) \wedge (c_5^*(x^4) = y^1)]$ factor in (2), we define the following set Z of x^4 q-tuples:

$$Z = \{x^4 | x^4 \sim y^1 \wedge x^4 \oplus y^0 \in I^{\neq} \wedge x^4 \oplus x^3 \in I^{\neq}\}$$

where $x^4 \sim y^1$ means that $\forall i, j \; x_i^4 = x_j^4$ if and only if $y_i^1 = y_j^1$. Let us denote by $q_1 \leq q$, the number of distinct y_i^1 values. there exist $i_1, \cdots, i_{q_1}$ indexes such that $y_{i_1}^1, \cdots, y_{i_{q_1}}^1$ are pairwise distinct. Each $i_k \in \{i_1, \cdots, i_{q_1}\}$ index determine a class such that for all elements i of this class, $y_i^1 = y_{i_k}^1$. So, $\forall i \in [1, .., q], \exists! i_k \in \{i_1, \cdots, i_{q_1}\} / y_i^1 = y_{i_k}^1, Cl(i) =_{def} i_k$.

There exist $\alpha = \frac{2^n!}{(2^n-q_1)!}$ x^4 values such that $x^4 \sim y^1$ (as a matter of fact such an x^4 is entirely determined by q_1 distinct values). Now:

$$\begin{aligned}|Z| &\geq |\{x^4 | x^4 \sim y^1\}| - |\{x^4 | x^4 \sim y^1 \wedge x^4 \oplus y^0 \notin I^{\neq}\}| \\ &\quad -|\{x^4 | x^4 \sim y^1 \wedge x^4 \oplus x^3 \notin I^{\neq}\}| \\ &\geq |\{x^4 | x^4 \sim y^1\}| - \sum_{i \neq j} |\{x^4 | x^4 \sim y^1 \wedge x_i^4 \oplus x_j^4 = y_i^0 \oplus y_j^0\}| \\ &\quad - \sum_{i \neq j} |\{x^4 | x^4 \sim y^1 \wedge x_i^4 \oplus x_j^4 = x_i^3 \oplus x_j^3\}|\end{aligned}$$

Given $i \neq j$, we can upper bound the size of $S_{ij} = \{x^4 | x^4 \sim y^1 \wedge x_i^4 \oplus x_j^4 = y_i^0 \oplus y_j^0\}$ by $\frac{2\alpha}{2^n}$.
As a matter of fact:

- if $y_i^1 = y_j^1$, then $y_i^0 \oplus y_j^0 \neq 0$ (because otherwise the (y_i^1, y_i^0) word would be equal to (y_j^1, y_j^0)), but $x^4 \sim y^1$ implies $x_i^4 = x_j^4$ and thus $x_i^4 \oplus x_j^4$ cannot be equal to $y_i^0 \oplus y_j^0$. So, $|S_{ij}| = 0$
- $y_i^1 \neq y_j^1$, then if $x^4 \in S_{ij}$, $x_{Cl(j)}^4$ is entirely determined by $x_{Cl(i)}^4$ since $x_{Cl(j)}^4 = x_{Cl(i)}^4 \oplus y_i^0 \oplus y_j^0$. Thus $|S_{ij}|$ contains at most $2^n(2^n-2)\cdots(2^n-q_1) = \frac{\alpha}{2^n-1} \leq \frac{2\alpha}{2^n}$ elements.

Similary, using the fact that $x_i^3 \neq x_j^3$, we can upper bound the size of $\{x^4 | x^4 \sim y^1 \wedge x_i^4 \oplus x_j^4 = x_i^3 \oplus x_j^3\}|$ by $\frac{2\alpha}{2^n}$. So, we have:

$$|Z| \geq \alpha \left[1 - \frac{q(q-1)}{2}\frac{2}{2^n} - \frac{q(q-1)}{2}\frac{2}{2^n}\right] \text{ i.e. } |Z| \geq \frac{2^n!}{(2^n-q_1)!}[1 - \frac{2q^2}{2^n}]$$

Now, $\sum_{x^4} \Pr[(c_3^*(x^2) \oplus x^3 = x^4) \wedge (c_4^*(x^3) \oplus x^4 = y^0) \wedge (c_5^*(x^4) = y^1)] \geq \sum_{x^4 \in Z} \Pr[(c_3^*(x^2) \oplus x^3 = x^4) \wedge (c_4^*(x^3) \oplus x^4 = y^0) \wedge (c_5^*(x^4) = y^1)]$. But, for any $x^4 \in Z$, we have $\Pr[c_5^*(x^4) = y^1] = \frac{(2^n-q_1)!}{2^n!} = \frac{1}{\alpha}$ and $\Pr[(c_3^*(x^2) = x^4 \oplus x^3)] = \frac{(2^n-q)!}{2^n!}$ due to Property 2 and the fact that the x_i^2 and the $x_i^4 \oplus x_i^3$ are pairwise distinct. We also have $\Pr[(c_3^*(x^2) = x^4 \oplus x^3)] = \frac{(2^n-q)!}{2^n!}$ for the same reasons, so that:
$\sum_{x^4} \Pr[(c_3^*(x^2) \oplus x^3 = x^4) \wedge (c_4^*(x^3) \oplus x^4 = y^0) \wedge (c_5^*(x^4) = y^1)]$

$$\begin{aligned}&\geq |Z| \cdot \left(\frac{(2^n-q)!}{2^n!}\right)^2 \cdot \frac{1}{\alpha} \\ &\geq \alpha \cdot \left(1 - \frac{2q^2}{2^n}\right)\left(\frac{(2^n-q)!}{2^n!}\right)^2 \cdot \frac{1}{\alpha} \\ &\geq \left(1 - \frac{2q^2}{2^n}\right)\left(\frac{(2^n-q)!}{2^n!}\right)^2\end{aligned}$$

If we now come back to inequality (2), we thus have:

$$\Pr[X \overset{c}{\mapsto} Y] \geq \sum_{x^2,x^3 \in I^{\neq}} \Pr\begin{bmatrix}(c_1^*(x^1) \oplus x^0 = x^2) \\ \wedge \\ (c_2^*(x^0) \oplus x^2 = x^3)\end{bmatrix} \cdot \left(1 - \frac{2q^2}{2^n}\right)\left(\frac{(2^n-q)!}{2^n!}\right)^2 \quad \text{(i)}$$

Let us now establish a lower bound on

$$\begin{aligned} B &= \sum_{x^2,x^3 \in I^{\neq}} \Pr[(c_1^*(x^1) \oplus x^0 = x^2) \wedge (c_2^*(x^0) \oplus x^2 = x^3)] \\ &= \Pr[x^2 \in I^{\neq} \wedge x^3 \in I^{\neq}] \\ &\quad \cdot \Pr[(c_1^*(x^1) \oplus x^0) \in I^{\neq} \wedge (c_2^*(x^0) \oplus x^2) \in I^{\neq} | x^2 \in I^{\neq}] \end{aligned}$$

But $\Pr[(c_1^*(x^1) \oplus x^0) \in I^{\neq}] \geq 1 - \sum_{i \neq j} \Pr[c_1^*(x_i^1) \oplus c_1^*(x_j^1) = x_i^0 \oplus x_j^0]$ and it is easy to establish (using the fact that $(x_i^1, x_i^0) \neq (x_j^1, x_j^0)$ and Property 3), that for any two distinct indexes i and j, $\Pr[c_1^*(x_i^1) \oplus c_1^*(x_j^1) = x_i^0 \oplus x_j^0] \leq \frac{1}{2^n - 1} \leq \frac{2}{2^n}$ Thus $\Pr[(c_1^*(x^1) \oplus x^0) \in I^{\neq}] \geq 1 - \frac{q(q-1)}{2} \cdot \frac{2}{2^n} \geq 1 - \frac{q^2}{2^n}$. for similar reasons, $\Pr[(c_2^*(x^0) \oplus x^2) \in I^{\neq} | x^2 \in I^{\neq}] \geq 1 - \frac{q^2}{2^n}$. Thus $B \geq \left(1 - \frac{q^2}{2^n}\right)^2 \geq 1 - \frac{2q^2}{2^n}$ (ii) . Now, by combinig (i) and (ii), we obtain:

$$\Pr[X \overset{c}{\mapsto} Y] \geq \left(1 - \frac{2q^2}{2^n}\right)^2 \left(\frac{(2^n-q)!}{2^n!}\right)^2$$

Now, $\left(\frac{(2^n-q)!}{2^n!}\right)^2 \geq \frac{1}{2^{2nq}}$ and $\left(1 - \frac{2q^2}{2^n}\right)^2 \geq 1 - \frac{4q^2}{2^n}$, so that

$$\Pr[X \overset{c}{\mapsto} Y] \geq \left(1 - \frac{4q^2}{2^n}\right) \cdot \frac{1}{2^{2nq}}$$

We can now apply Theorem 4 with $\epsilon = \frac{4q^2}{2^n}$ and we obtain:

$$\begin{gathered} \mathrm{Adv}_{\mathcal{A}}^q(c, c^*) \leq \frac{4q^2}{2^n} + \frac{q(q-1)}{2 \cdot 2^{2n}} \\ \mathrm{Adv}_{\mathcal{A}}^q(c, c^*) \leq \frac{9}{2} \cdot \frac{q^2}{2^n} \end{gathered}$$

□

5 Analysis of the R-Scheme

In this Section, we compare, for various values of the r number of rounds of an R-scheme, the $f = \psi_R(c_1^*, c_2^*, ..., c_r^*)$ $2n$-bit random permutation deduced from r independent perfect random n-bit permutations $c_1^*, c_2^*, ..., c_r^*$ with a perfect $2n$-bit function f^*.

5.1 Three-Round R-Scheme

We first establish the following theorem for a 3-round version of the R-scheme.

Theorem 5 *Let n be an integer, c_1^*, c_2^*, c_3^* be three independent perfect random permutation from I_n to I_n and f^* be the perfect random function on the I_{2n} set. Let $f = \psi_R(c_1^*, c_2^*, c_3^*)$ denote the random permutation associated with the 3-rounds R-scheme. For any adaptive distinguisher $\mathcal{A}$ with q queries, we have:*

$$Adv_{\mathcal{A}}^{q}(f, f^*) \leq 3q^2 2^{-n}$$

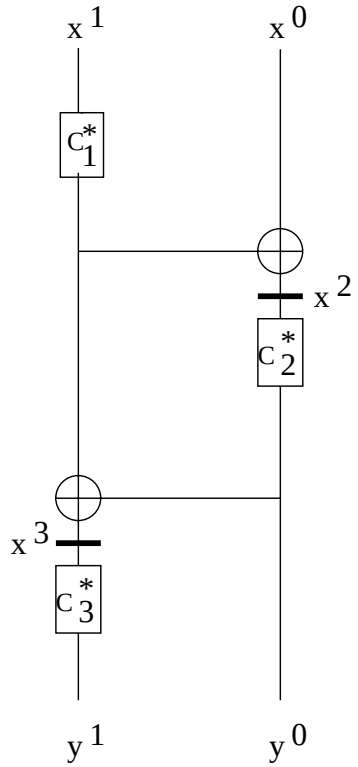


Fig. 3. R-scheme three rounds

Proof: We will compare the $f = \psi'_R(c_1^*, c_2^*, c_3^*)$ permutation generator of Figure 3 (which pseudorandomness properties are exactly the same as for $\psi_R(c_1^*, c_2^*, c_3^*)$) with the perfect random function f^*. Let us first introduce some notation. We consider a $X = (X_i)_{i\in[1..q]} = (x_i^1, x_i^0)$ q-tuple of $2n$-bit f input words. We denote the corresponding q-tuple of f output words by $Y = (Y_i)_{i\in[1..q]} = (y_i^1, y_i^0)_{i\in[1..q]}$. For each $(y_i^1, y_i^0) = \psi'_R(c_1^*, c_2^*, c_3^*)(x_i^1, x_i^0)$ computation, we denote by x_i^2 and x_i^3 the intermediate values which locations are marked in Figure 3. More explicitly:

$$x_i^2 = c_1^*(x_i^1) \oplus x_i^0$$
$$x_i^3 = c_1^*(x_i^1) \oplus c_2^*(x_i^2) = y_i^0 \oplus c_1^*(x_i^1)$$

Finally, we denote the $(x_i^0)_{i\in[1..q]}$, $(x_i^1)_{i\in[1..q]}$, $(x_i^2)_{i\in[1..q]}$, $(x_i^3)_{i\in[1..q]}$, $(y_i^0)_{i\in[1..q]}$ and $(y_i^1)_{i\in[1..q]}$ q-tuples of n-bit words by x^0, x^1, x^2, x^3, y^0, y^1 respectively.

We now define $\mathcal{X}$ as the set of X q-tuples of pairwise distinct I_{2n} words (i.e. such that for any distinct i,j numbers in $[1..q]$, $x_i^1 \neq x_j^1$ or $x_i^0 \neq x_j^0$), and define $\mathcal{Y}$ as the set of those Y q-tuples of I_{2n} words such that the corresponding y^1 and y^0 q-tuples both consist of pairwise distinct I_n words: $\mathcal{Y} = \{(Y_1, \cdots, Y_q) \in (I_{2n})^q \ / \ y^1 \in I^{\neq},\ y^0 \in I^{\neq}\}$.

We want to establish a lower bound on the size of $\mathcal{Y}$ and the $\Pr[X \overset{f}{\mapsto} Y]$ transition probability associated with any X q-tuple in $\mathcal{X}$ and any Y q-tuple in $\mathcal{Y}$ and show that there exists ϵ_1 and ϵ_2 real numbers satisfying conditions of Theorem 1.

Let us first establish a lower bound on $|\mathcal{Y}|$. We have:

$$\begin{aligned}
|\mathcal{Y}| &= |I^{2n}|^q \cdot (1 - \Pr[y^1 \notin I^{\neq} \vee y^0 \notin I^{\neq}]) \\
&\geq |I^{2n}|^q (1 - \sum_{i,j\in[1..q], i\neq j} \Pr[y_i^1 = y_j^1] - \sum_{i,j\in[1..q], i\neq j} \Pr[y_i^0 = y_j^0]) \\
&\geq |I^{2n}|^q (1 - \frac{q(q-1)}{2} \cdot 2^{-n} - \frac{q(q-1)}{2} \cdot 2^{-n}) \\
&\geq |I^{2n}|^q (1 - \frac{q(q-1)}{2^n})
\end{aligned}$$

So, we can take $\epsilon_1 = \frac{q(q-1)}{2^n}$.

Now, given any X q-tuple of $\mathcal{X}$ and any Y q-tuple of $\mathcal{Y}$ let us establish a lower bound on $\Pr[X \overset{f}{\mapsto} Y]$.

$$\begin{aligned}
\Pr[X \overset{f}{\mapsto} Y] &= \textstyle\sum_{x^2,x^3\in I} \Pr[(c_1^*(x^1) \oplus x^0 = x^2) \wedge (c_1^*(x^1) \oplus y^0 = x^3) \\
&\qquad \wedge (c_2^*(x^2) = y^0) \wedge (c_3^*(x^3) = y^1)] \\
&\geq \textstyle\sum_{x^2,x^3\in I^{\neq}} \Pr[(c_1^*(x^1) \oplus x^0 = x^2) \wedge (c_1^*(x^1) \oplus y^0 = x^3)] \\
&\qquad \cdot \Pr[(c_2^*(x^2) = y^0) \wedge (c_3^*(x^3) = y^1)] \qquad (2)
\end{aligned}$$

First, for any x^2 q-tuple of $I^{\neq}$ and any x^3 q-tuple of $I^{\neq}$, let us compute $\Pr[(c_2^*(x^2) = y^0) \wedge (c_3^*(x^3) = y^1)] = \Pr[(c_2^*(x^2) = y^0)] \cdot \Pr[(c_3^*(x^3) = y^1)]$. Since x^2 and y^0 both belong to $I^{\neq}$, we can can apply Property 2 of Section 2 concerning random permutations, so that $\Pr[(c_2^*(x^2) = y^0)] = \frac{(2^n-q)!}{2^n!}$. For the same reason, we also have $\Pr[(c_3^*(x^3) = y^1)] = \frac{(2^n-q)!}{2^n!}$.

So, we have $\Pr[(c_2^*(x^2) = y^0) \wedge (c_3^*(x^3) = y^1)] = \left(\frac{(2^n-q)!}{2^n!}\right)^2$.

Now, $\left(\frac{(2^n-q)!}{2^n!}\right)^2 \geq \frac{1}{2^{2nq}}$.

Therefore, inequality (2) implies:

$$\Pr[X \overset{f}{\mapsto} Y] \geq \sum_{x^2,x^3\in I^{\neq}} \frac{1}{2^{2nq}} \cdot \Pr[(c_1^*(x^1) \oplus x^0 = x^2) \wedge (c_1^*(x^1) \oplus y^0 = x^3)] \quad (i)$$

Let us now estimate $B = \sum_{x^2,x^3 \in I^{\neq}} \Pr[(c_1^*(x^1) \oplus x^0 = x^2) \wedge (c_1^*(x^1) \oplus y^0 = x^3)]$

We have:

$$\begin{aligned} B &= \Pr[(c_1^*(x^1) \oplus x^0 \in I^{\neq} \wedge (c_1^*(x^1) \oplus y^0 \in I^{\neq}] \\ &= 1 - \Pr[(c_1^*(x^1) \oplus x^0 \notin I^{\neq} \vee (c_1^*(x^1) \oplus y^0 \notin I^{\neq}] \\ &\geq 1 - \sum_{i,j,i\neq j} Pr[c_1^*(x_i^1) \oplus x_i^0 = c_1^*(x_j^1) \oplus x_j^0] \\ &\quad - \sum_{i,j,i\neq j} Pr[c_1^*(x_i^1) \oplus y_i^0 = c_1^*(x_j^1) \oplus y_j^0] \end{aligned}$$

Let us evaluate $\Pr[c_1^*(x_i^1) \oplus x_i^0 = c_1^*(x_j^1) \oplus x_j^0]$ and $\Pr[c_1^*(x_i^1) \oplus c_1^*(x_j^1) = x_i^0 \oplus x_j^0]$. Due to Property 3 of Section 2, if $x_i^1 \neq x_j^1$ given any fixed difference δ (here equal to $x_i^0 \oplus x_j^0$), $\Pr[c_1^*(x_i^1) \oplus c_1^*(x_j^1) = \delta] \leq \frac{2}{2^n}$. On the other hand, if $x_i^1 = x_j^1$, then $x_i^0 \neq x_j^0$, so that $\Pr[c_1^*(x_i^1) \oplus x_i^0 = c_1^*(x_j^1) \oplus x_j^0]$. In all cases, $\Pr[c_1^*(x_i^1) \oplus x_i^0 = c_1^*(x_j^1) \oplus x_j^0] \leq \frac{2}{2^n}$. Applying this property to the $\frac{q(q-1)}{2}$ $(i,j), i \neq j$ pairs of $[1..q]$ indexes, we obtain $\sum_{i,j} Pr[c_1^*(x_i^1) \oplus x_i^0 = c_1^*(x_j^1) \oplus x_j^0] \leq \frac{q(q-1)}{2^n}$. For the same reasons, $\sum_{i,j,i\neq j} Pr[c_1^*(x_i^1) \oplus y_i^0 = c_1^*(x_j^1) \oplus y_j^0] \leq \frac{q(q-1)}{2^n}$. Thus:

$$B \geq 1 - \frac{2q(q-1)}{2^n} \quad (ii)$$

By using inequalities (i) and (ii), we obtain:

$$\Pr[X \overset{f}{\mapsto} Y] \geq \left(1 - \frac{2q(q-1)}{2^n}\right) \cdot \frac{1}{2^{2nq}}$$

We can notice that $Pr[X \overset{f^*}{\mapsto} Y] = \frac{1}{2^{2nq}}$. So we can apply Theorem 1 with $\epsilon_1 = \frac{q(q-1)}{|I^n|}$ and $\epsilon_2 = \frac{2q(q-1)}{|I^n|}$. We obtain:

$$\begin{gathered} \mathrm{Adv}_{\mathcal{A}}^q(f, f^*) \leq \frac{3q(q-1)}{2^n} \\ \mathrm{Adv}_{\mathcal{A}}^q(f, f^*) \leq \frac{3q^2}{2^n} \end{gathered}$$

□

5.2 Four-Round R-Scheme: $\psi_R(c_1^*, c_2^*, c_3^*, c_4^*)$ Is Not a Super Pseudo-Random Permutation

We consider the four-round permutation generator f deduced from $\psi_R(c_1^*, c_2^*, c_3^*, c_4^*)$ by omitting the final XOR (this does obviously not matter for the super pseudo randomness issue considered here). The random function f can be represented by extending the 3-round function of Figure 3 above by one round.

Let us show that 2 chosen encryption and two chosen decryption queries suffice to distinguish f from a the perfect random permutation $c*$ with a very large probability. Let us consider the encryption, under the function f, of two

distinct $2n$-bit (x^1, x^0) plaintext blocks (a, b) and (a, b') which left halves are equal, and denote by (c, d) and (c', d') the two obtained (y^1, y^0) ciphertext blocks. It is easy to check that if we swap the left halves of the two obtained ciphertexts, thus obtaining two modified ciphertexts (c', d) and (c, d') and if we decrypt (c', d) and (c, d') under f^{-1}, we obtain two plantext values α, β and α', β' which left halves are equal: $\alpha = \alpha'$. This would be extremely unlikely to happen if f was replaced by a perfect random permutation $c*$.

The above test allows to distinguish f from a perfect random permutation of I_{2n} with a probability close to 1.

5.3 Five-Round R-Scheme: $\psi_R(c_1^*, c_2^*, c_3^*, c_4^*, c_5^*)$ Is a Super Pseudo-Random Permutation

The following theorem is a direct consequence of Theorem 3 due to the fact that $\psi'_R(c_1, c_2, c_3, c_4, c_5)$ and $\psi'_L(c_1^{-1}, c_2^{-1}, c_3^{-1}, c_4^{-1}, c_5^{-1})$ are inverse of each other, every distinguisher for $\psi_R(c_1^*, c_2^*, c_3^*, c_4^*, c_5^*)$ can be converted into a distinguisher for $\psi_L(c_1^*, c_2^*, c_3^*, c_4^*, c_5^*)$ with the same number of encryption and decryption queries. Therefore, Theorem 3 results in the following analogue theorem for the 5-round R-scheme.

Theorem 6 *Let n be an integer, c_1^*, c_2^*, c_3^*, c_4^*, c_5^* be five independent random functions from I_n to I_n and c^* be the perfect random permutation on the I_{2n} set. Let $c = \psi_R(c_1^*, c_2^*, c_3^*, c_4^*, c_5^*)$ denote the random permutation associated with the five round R-scheme. For any adaptive super pseudorandom permutation distinguisher $\mathcal{A}$ with q queries, we have:*

$$Adv_{\mathcal{A}}^{q}(c, c^*) \leq \frac{9}{2} \cdot \frac{q^2}{2^n}$$

6 Conclusion

As a consequence of previous results, the security properties of the L-scheme and the R-scheme are distinct when it comes to chosen plaintext attacks, but equivalent when it comes to chosen plaintext or ciphertext attacks. As a matter of fact, the minimal number of rounds required in order of the R-scheme to be undistinguishable from a pseudorandom function with adaptively chosen encryption queries is less than for the L-scheme (3 rounds instead of 4), whereas the minimal numbers of rounds required by the R-scheme and the L-scheme in order to be undistinguishable from a pseudorandom permutation with adaptively chosen encryption or decryption queries are equal (5 rounds for both schemes).

A Appendix

A.1 A Short Proof of Theorem 1

Let us restrict ourselves to the case of any fixed deterministic algorithm A which uses q adaptively chosen queries (the generalisation to the case of a probabilistic algorithm is easy).

A has the property that if the q-uple of outputs encountered during an A computation is $Y = (Y_1, \cdots, Y_q)$, the value of the $X = (X_1, \cdots, X_q)$ q-uple n is of query inputs encountered during this computation is entirely determined (this is easy to prove by induction: the initial query input X_1 is fixed ; if for a given A computation the first query output is Y_1, then X_2 is determined, etc.). We denote by $X(Y)$ the single q-uple of query inputs corresponding to any possible Y q-uple of query outputs, and we denote by S_A the subset of those $Y \in {I_m}^q$ values such that if the $X(Y)$ and Y q-uples query inputs and outputs are encountered in a A computation, then A outputs a 1 answer.

The p and p^* probabilities can be expressed using S_A as

$$p = \sum_{Y \in S_A} Pr[X(Y) \stackrel{f}{\mapsto} Y] \text{ and } p^* = \sum_{Y \in S_A} Pr[X(Y) \stackrel{f^*}{\mapsto} Y]$$

We can now lower bound p using the following inequalities:

$p \geq \sum_{Y \in S_A \cap \mathcal{Y}} (1 - \epsilon_2) \cdot Pr[X(Y) \stackrel{f^*}{\mapsto} Y]$ (due to inequality (ii))

$\geq \sum_{Y \in S_A} (1 - \epsilon_2) Pr[X(Y) \stackrel{f^*}{\mapsto} Y] - \sum_{Y \in {I_m}^q - m athcal Y} (1 - \epsilon_2) \cdot Pr[X(Y) \stackrel{f^*}{\mapsto} Y]$

But

$$\sum_{Y \in S_A} (1 - \epsilon_2) \cdot Pr[X(Y) \stackrel{f^*}{\mapsto} Y] = (1 - \epsilon_2) p^*$$

and

$\sum_{Y \in {I_m}^q - \mathcal{Y}} (1 - \epsilon_2) \cdot Pr[X(Y) \stackrel{f^*}{\mapsto} Y] = (1 - \epsilon_2) \cdot \frac{|I_m|^q - |\mathcal{Y}|}{|I_m|} \leq (1 - \epsilon_2) \cdot \epsilon_1$ (due to inequality (i)).

Therefore $p \geq (1 - \epsilon_2)(p^* - \epsilon_1) = p^* - \epsilon_1 - \epsilon_2 \cdot p^* + \epsilon_1 \cdot \epsilon_2$

thus finally (using $p^* \leq 1$ and $\epsilon_1 \cdot \epsilon_2 \geq 0$)

$p \geq p* - \epsilon_1 - \epsilon_2$ (a)

If we now consider the A' distinguisher which outputs are the inverse of those of A (i.e. A' answers 0 iff A answers 1), we obtain an inequality involving this time $1 - p$ and $1 - p*$:

$(1 - p) \geq (1 - p*) - \epsilon_1 - \epsilon_2$ (b)

Combining inequalities (a) and (b), we obtain $|p - p*| \leq \epsilon_1 + \epsilon_2$ QED.

A.2 A Proof Sketch for Theorem 2

We will compare the $f = \psi'_L(c_1^*, c_2^*, c_3^*, c_4^*)$ permutation generator of Figure 4 (which pseudorandomness properties are exactly the same as for $\psi_L(c_1^*, c_2^*, c_3^*, c_4^*)$) with the perfect random function f^*. This proof is near to the proof of Section 5.1. That's why we do not detail some computations that are the same that in Section 5.1.

Let us first introduce some notation. We consider a $X = (X_i)_{i \in [1..q]} = (x_i^1, x_i^0)$ q-tuple of 2n-bit f input words. We denote the corresponding q-tuple of f output words by $Y = (Y_i)_{i \in [1..q]} = (y_i^1, y_i^0)_{i \in [1..q]}$. For each $(y_i^1, y_i^0) = \psi'_L(c_1^*, c_2^*, c_3^*, c_4^*)(x_i^1, x_i^0)$ computation, we denote by x_i^2 and x_i^3 the intermediate values which locations are marked in Figure 4. More explicitly:

$$x_i^2 = c_1^*(x_i^1) \oplus x_i^0$$
$$x_i^3 = c_2^*(x_i^0) \oplus x_i^2$$

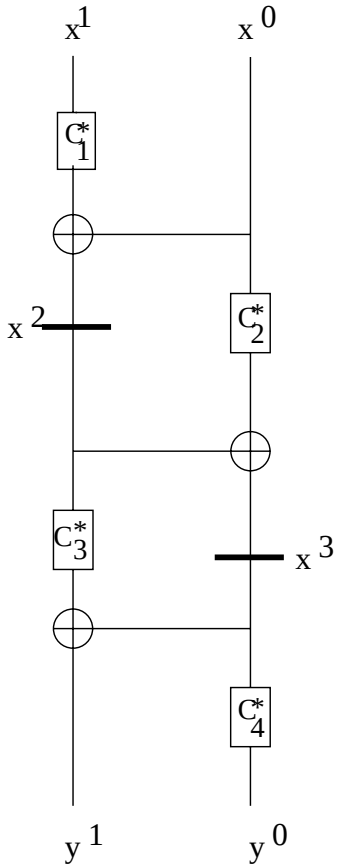


Fig. 4. L-scheme four rounds

Finally, we denote the $(x_i^0)_{i\in[1..q]}$, $(x_i^1)_{i\in[1..q]}$, $(x_i^2)_{i\in[1..q]}$, $(x_i^3)_{i\in[1..q]}$, $(y_i^0)_{i\in[1..q]}$ and $(y_i^1)_{i\in[1..q]}$ q-tuples of n-bit words by x^0, x^1, x^2, x^3, y^0, y^1 respectively.

We now define $\mathcal{X}$ as the set of X q-tuples of pairwise distinct I_{2n} words (i.e. such that for any distinct i,j numbers in $[1..q]$, $x_i^1 \neq x_j^1$ or $x_i^0 \neq x_j^0$), and define $\mathcal{Y}$ as the set of those Y q-tuples of I_{2n} words such that the corresponding y^1 q-tuples consists of pairwise distinct I_n words: $\mathcal{Y} = \{(Y_1, \cdots, Y_q) \in (I^{2n})^q \ / \ y^1 \in I^{\neq},\ y^0 \in I^{\neq}\}$.

We want to lower bound the size of $\mathcal{Y}$ and the $\Pr[X \overset{f}{\mapsto} Y]$ transition probability associated with any X q-tuple in $\mathcal{X}$ and any Y q-tuple in $\mathcal{Y}$ and show that there exists ϵ_1 and ϵ_2 real numbers satisfying conditions of Theorem 1.

We have (for more details, see section 5.1):

$$\begin{aligned} |\mathcal{Y}| &= |I^{2n}|^q \cdot (1 - \Pr[y^1 \notin I^{\neq}]) \\ &\geq |I^{2n}|^q (1 - \frac{q(q-1)}{2.2^n}) \end{aligned}$$

So, we can take $\epsilon_1 = \frac{q(q-1)}{2.2^n}$.

Now, given any X q-tuple of $\mathcal{X}$ and any Y q-tuple of $\mathcal{Y}$ let us establish a lower bound on $\Pr[X \overset{f}{\mapsto} Y]$ (for more details, see section 5.1).

$$\Pr[X \overset{f}{\mapsto} Y] \geq \sum_{x^2,x^3,x^3\oplus y^1 \in I^{\neq}} \Pr[(c_1^*(x^1) \oplus x^0 = x^2) \wedge (x^3 = c_2^*(x^0))] \oplus x^2) \cdot \Pr[(c_3^*(x^2) \oplus x^3 = y^0) \wedge (c_4^*(x^3) = y^1)] \quad (3)$$

First, for any x^2, x^3, $x^3 \oplus y^1$ q-tuple of $I^{\neq}$, we have $\Pr[(c_3^*(x^2) \oplus x^3 = y^0) \wedge (c_3^*(x^3) = y^1)] = \left(\frac{(2^n-q)!}{2^n!}\right)^2 \geq \frac{1}{2^{2nq}}$ (for more details, see section 5.1). Therefore, inequality (1) implies:

$$\Pr[X \overset{f}{\mapsto} Y] \geq \sum_{x^2,x^3,x^3\oplus y^1 \in I^{\neq}} \frac{1}{2^{2nq}} \cdot \Pr[(c_1^*(x^1) \oplus x^0 = x^2) \wedge (c_1^*(x^1) \oplus y^0 = x^3)] \quad (i)$$

Let us now estimate:

$$B = \sum_{x^2,x^3,x^3\oplus y^1 \in I^{\neq}} \Pr[(c_1^*(x^1) \oplus x^0 = x^2) \wedge (c_2^*(x^0) \oplus x^2 = x^3)]$$

We have:

$$\begin{aligned} B &= \Pr[(c_1^*(x^1) \oplus x^0) \in I^{\neq} \wedge (c_2^*(x^0) \oplus c_1^*(x^1) \oplus x^0) \in I^{\neq} \wedge (x^3 \oplus y^1) \in I^{\neq}] \\ &= 1 - \Pr[(c_1^*(x^1) \oplus x^0 \in I^{=}) \vee (c_2^*(x^0) \oplus c_1^*(x^1) \oplus x^0) \in I^{=} \vee (x^3 \oplus y^1) \in I^{=}] \\ &\geq 1 - 3.\frac{q(q-1)}{2}.\frac{2}{2^n} \end{aligned}$$

By using inequalities (i) and (ii), we obtain:

$$\Pr[X \overset{f}{\mapsto} Y] \geq \left(1 - \frac{3q(q-1)}{2^n}\right) \cdot \frac{1}{2^{2nq}}$$

We can notice that $Pr[X \overset{f^*}{\mapsto} Y] = \frac{1}{2^{2nq}}$. So we can apply Theorem 1 with $\epsilon_1 = \frac{q(q-1)}{2|I^n|}$ and $\epsilon_2 = \frac{3q(q-1)}{|I^n|}$. We obtain:

$$\begin{aligned} \mathrm{Adv}_{\mathcal{A}}^q(f, f^*) &\leq \tfrac{7q(q-1)}{2.2^n} \\ \mathrm{Adv}_{\mathcal{A}}^q(f, f^*) &\leq \tfrac{7q^2}{2.2^n} \end{aligned}$$

□

References

[Ai96] W. Aiello, R. Venkatesan, "Foiling Birthday Attacks in Length-Doubling Transformations". In *Advances in Cryptology - Eurocrypt'96*, LNCS 1070, p. 307, Springer Verlag, Saragossa, Spain, May 1996.

[Be94] M. Bellare, J. Kilian, P. Rogaway, "The Security of Cipher Block Chaining". In *Advances in Cryptology - CRYPTO'94*,LNCS 839, p. 341, Springer-Verlag, Santa Barbara, U.S.A., 1994.

[Ka] Specification of the 3GPP confidentiality and Integrity algorithm KASUMI. Documentation available on http://www.etsi.org/

[Ka97] Y. Kaneko, F. Sano, K. Sakurai, "On Provable Security against Differential and Linear Cryptanalysis in Generalized Feistel Ciphers with Multiple Random Functions". In *Selected Areas in Cryptography - SAC'97*, Ottawa, Canada, August 1997.

[La90] X. Lai, J.L. Massey, "A Proposal for a New Block Encryption Standard". In *Advances in Cryptology - Eurocrypt'90*, LNCS 473 , p. 389, Springer Verlag, Aarhus, Denemark, 1991.

[Lu88] M. Luby, C. Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Function". In *Siam Journal on Computing* , vol. 17, p. 373, 1988.

[Ma92] U. Maurer, "A Simplified and generalised treatment of Luby-Rackoff Pseudorandom Permutation Generators", In *Advances in Cryptology - Eurocrypt'92*, LNCS 658 , p. 239, Springer Verlag, New York, USA, 1992.

[Ma93] M. Matsui, "New Block Encryption Algorithm MISTY", In *Fast Software Encryption - FSE'97*, LNCS 1267, p. 54, Springer Verlag, Haifa, Israel, 1997.

[Pa91] J. Patarin, "Etude de Générateurs de Permutation Basés sur le Schéma du D.E.S. ", Phd. Thesis, University of Paris VI, 1991.

[Sa97] K. Sakurai, Y. Zheng, "On Non-Pseudorandomness from Block Ciphers with Provable Immunity Against Linear Cryptanalysis, In *IEICE Trans. Fundamentals*, vol. E80-A, n. 1, January 1997.

[Su96] M. Sugita, "Pseudorandomness of a Block Cipher MISTY", Technical Reporrt of IEICE, ISEC96-9.

[Su97] M. Sugita, "Pseudorandomness of a Block Cipher with Recursive Structures", Technical Report of IEICE, ISEC97-9.

[Va99] S. Vaudenay, "On Provable Security for Conventional Cryptography", In *ICISC'99*, invited lecture.

[Zh89] Y. Zheng, T. Matsumoto, H. Imai, "On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses". In *Advances in Cryptology - CRYPTO'89*, LNCS 435, p. 461, Springer-Verlag, Santa Barbara, U.S.A., 1990.

The Security of "One-Block-to-Many" Modes of Operation

Henri Gilbert

France Télécom R&D
henri.gilbert@francetelecom.com

Abstract. In this paper, we investigate the security, in the Luby-Rackoff security paradigm, of blockcipher modes of operation allowing to expand a one-block input into a longer t-block output under the control of a secret key K. Such "one-block-to-many" modes of operation are of frequent use in cryptology. They can be used for stream cipher encryption purposes, and for authentication and key distribution purposes in contexts such as mobile communications. We show that although the expansion functions resulting from modes of operation of blockciphers such as the counter mode or the output feedback mode are not pseudorandom, slight modifications of these two modes provide pseudorandom expansion functions. The main result of this paper is a detailed proof, in the Luby-Rackoff security model, that the expansion function used in the construction of the third generation mobile (UMTS) example authentication and key agreement algorithm MILENAGE is pseudorandom.

1 Introduction

In this paper, we investigate the security of modes of operation of blockciphers allowing to construct a length increasing function, i.e. to expand a 1-block input value x into a longer t-block output $(z_1, z_2, \ldots, z_t)$ (where $t \geq 2$), under the control of a secret key K.

Such length increasing modes of operation of blockciphers associated with a one block to t blocks expansion function are of extremely frequent use in cryptology, mainly for pseudo-random generation purposes. They can be considered as a kind of dual of length decreasing modes of operation associated with a t blocks to one block compression function used for message authentication purpose (e.g. CBC MAC). In both cases, the essential security requirement is that the resulting one block to t blocks (respectively t blocks to one block) function be pseudorandom, i.e. (informally speaking) indistiguishable, by any reasonable adversary, from a perfect random function with the same input and output sizes. Thus the Luby and Rackoff security paradigm [LR88], which allows to relate the pseudo-randomness of a function resulting from a cryptographic construction to the pseudorandomness of the elementary function(s) encountered at the lower level of the same construction, represents a suitable tool for analysing the security of both kinds of modes of operation. However, the security and the efficiency of length increasing modes of operation have been much less investigated so far than the one of length decreasing modes of operation such as CBC MAC

[BKR94,PR00], R-MAC [JJV02], etc., or than constructions of length-preserving functions or permutations such as the Feistel scheme [LR88,Pa91].

The practical significance of length increasing modes of operation of blockciphers comes from the fact that they provide the two following kinds of pseudorandom generation functions, which both represent essential ingredients for applications such as mobile communications security.

Example 1: Stream cipher modes of operation of blockciphers. It has become usual for stream ciphers (whether they are derived or not from a mode of operation of a blockcipher) to require that the generated pseudo-random sequences used to encrypt data be not only dependent upon a secret key, but also upon an additional (non secret) input value x, sometimes referred to as an initialization vector or as an initial value (IV). This holds for most recently proposed stream ciphers, e.g. SEAL [RC98], SCREAM [HCCJ02], SNOW [EJ02], BGML [HN00], and for the stream cipher mode of operation of the KASUMI blockcipher used in the third generation mobile system UMTS [Ka00]. As a consequence, stream ciphers are more conveniently modelled as a length increasing pseudo-random function $F_K : \{0,1\}^n \rightarrow \{0,1\}^{nt}$; $x \mapsto F_K(x) = (z_1, z_2, \cdots, z_t)$ than as a mere pseudo-random numbers generator allowing to derive a pseudo-random sequence $(z_1, z_2, \cdots, z_t)$ of nt bits from a secret seed K. The advantage of modelling a stream cipher as a length increasing function generator rather than as a numbers generator is that it allows to reflect the security conditions on the dependance of the pseudo-random sequence in the input value, by requiring that F_K be a pseudo-random function, indistinguishable from a perfect random function with the same input and output sizes by any reasonable adversary.

Example 2: Combined authentication and key distribution. In mobile communication systems (GSM, UMTS, etc.) and more generally in most secret key security architectures where authentication and encryption are provided, protected communications are initiated with a kind of "handshake" where authentication or mutual authentication between the user's device and the network and session key(s) distribution are performed. Such an initial handshake is followed by a protected communication, where the session key(s) resulting from the handshake phase are used to encrypt and/or to authenticate the data exchanges. In order for the handshake protocol not to delay the actual protected communication phase, it is essential to restrict it to two passes and to minimize the amount of data exchanged. For that purpose one of the parties (typically the network in the case of mobile communications) sends a random challenge (accompanied by additional data such as a message authenticated counter value if mutual authentication is needed), and this random challenge serves as an input to a secret key function allowing to derive an authentication response and one or several session key(s). In recent mobile communication systems such as UMTS, the length of the outputs to be produced (measured in 128-bit blocks) far exceeds the 1-block length of the random challenge. Thus, one single operation of a blockcipher does not suffice to produce the various outputs needed. In order to base the security of the cryptologic computations performed during the handshake upon the security

of a trusted blockcipher, a suitable one-block-to-many mode of operation of the underlying blockcipher has to be defined. The security requirements are not only that each of the output blocks be unpredictable by an adversary. In addition, the information on one subset of the outputs (say for instance an authentication response) should not help an adversary to derive any information about the rest of the outputs (say for instance the session key used to encrypt the subsequent exchanges). These various security requirements can be again reflected, as in the example of stream cipher modes of operation, in saying that the one to t blocks function $F_K : \{0,1\}^n \rightarrow \{0,1\}^{n.t}$; $x \mapsto F_K(x) = (z_1, z_2, \cdots, z_t)$ used to derive the various output values must be indistiguishable from a perfect random function with the same input and output sizes.

In this paper, we show that although the one block to t blocks functions associated with well known modes of operation of blockciphers such as the Output Feedback mode (OFB) or the so-called Counter mode are not pseudorandom, slightly modified modes of operation in which the one-block input is first "prewhitened" before being subject to an expansion process are pseudorandom in a formally provable manner. The main result of this paper is a detailed pseudorandomness proof, in the Luby and Rackoff security model, for the one to t blocks mode of operation of a blockcipher used in the UMTS example authentication and key distribution algorithm MILENAGE [Mi00], which can be considered as a modified counter mode. We also provide pseudorandomness proofs for a modified version of the OFB mode.

Related work. The study of pseudorandomness properties of cryptographic constructions initiated Luby and Rackoff's seminal paper [LR88] has represented a very active research area for the last decade. In particular, Patarin clarified the link between the best advantage of a q-queries distinguisher and the q-ary transition probabilities associated with f and proved indistinguihability bounds for numerous r-round Feistel constructions [Pa91], Maurer showed how to generalise indistinguishability results related to perfect random functions to indistinguishability results related to nearly perfect random functions [Ma92], Bellare, Kilian, Rogaway [BKR94], and later on several other authors [PR00,JJV02,BR00] investigated the application of similar techniques to various message authentication modes of operation, Vaudenay embedded techniques for deriving indistinguishability bounds into a broader framework named the decorrelation theory [Va98,Va99]. In this paper, we apply general indistinguishability proof techniques due to Patarin [Pa91] in an essential manner.

Our approach to expansion functions constructions based on blockcipher modes of operation has some connections, but also significant differences, with the following recently proposed blockcipher based expansion function constructions:

– in [DHY02], Desai, Hevia and Yin provide security proofs, in the Luby-Rackoff paradigm, for the ANSI X9.17 pseudo random sequences generation mode of operation of a blockcipher, and for an improved version of this mode which is essentially the same as the modified OFB mode considered in this paper. However, the security model considered in [DHY02] is quite distinct (and somewhat

complementary): we consider the pseudorandomness properties of the one to t blocks expansion function resulting from the considered mode of operation, whereas [DHY02] models a PRG mode of operation as the iteration a "smaller" keyed state transition and keystream output function, and consider the pseudorandomness properties of such state transition functions.

– in [HN00], Hastad and Näslund propose a pseudorandom numbers generator named BMGL. BGML is based on a "key feedback" mode of operation of a blockcipher. The security paradigm underlying BMGL (namely the indistinguishability of pseudorandom numbers sequences from truly random sequences, based upon a combination of the Blum-Micali PRG construction [BM84] and a variant of the Goldreich Levin hard core bits construction [GL89], in which the conjectured onewayness of the key dependance of the blockcipher is used to construct PR sequences of numbers) is quite different from the one considered here (namely the indistinguishability of the constructed expansion function from a perfect random function, assuming that the underlying blockcipher is indistinguishable from a perfect random one-block permutation). The advantage of the BGML approach it that it relies upon less demanding security assumptions for the underlying blockcipher than in our approach, but the disadvantage is that it leads to less efficient constructions in terms of the number of blockcipher invocations per output block.

– in [BDJR97], Bellare, Desai, Jokipii and Rogaway provide security proofs for stream cipher modes of operation, namely the XOR scheme and a stateful variant named CTR schemes. These two modes have some connections with the insecure one block to t blocks mode of operation referred to as the counter mode in this paper. However, a major difference between these modes is that in the XOR and CTR schemes, and adversary has no control at all of the inputs to the underlying blockcipher f (she can only control the plaintext), whereas in all the one to many blocks modes we consider in this paper, an adversary can control the one-block input value. Thus, there is no contradiction between the facts that the XOR and CTR encryption schemes are shown to be secure in [BDJR97] and that the counter mode of operation can easily be shown to be totally insecure.

This paper is organized as follows: Section 2 introduces basic definitions and results on random functions and security proof techniques in the Luby-Rackoff security model. Section 3 describes various "one-block-to-many" modes of operation of blockciphers, and introduces a modified variant of the counter mode used in MILENAGE and an improved variant of the OFB mode. Sections 4 and 5 present pseudorandomness proofs for the two latter modes.

2 Security Framework

2.1 The Luby-Rackoff Security Paradigm

A key dependent cryptographic function such as a blockcipher or a mode of operation of a blockcipher can be viewed as a random function associated with a randomly selected key value. It is generally defined using a recursive construction

process. Each step of the recursion consists of deriving a random function (or permutation) F from r previously defined random functions (or permutations) $f_1, \cdots, f_r$, and can be represented by a relation of the form $F = \Phi(f_1, \cdots, f_r)$.

One of the strongest security requirement one can put on such a random function or permutation F is that F be impossible to distinguish with a non negligible success probability from a perfect random function or permutation F^* uniformly drawn from the set of all functions (or permutations) with the same input and output sizes, even if a probabilistic testing algorithm A of unlimited power is used for that purpose and if the number q of adaptively chosen queries of A to the random instance of F or F^* to be tested is large.

It is generally not possible to prove indistiguishability properties for "real life" cryptologic random functions and large numbers of queries, because this would require a far too long key length. However, it is often possible to prove or disprove that if a random function F encountered at a given level of a cryptologic function construction is related to random functions encountered at the lower recursion level by a relation of the form $f = \Phi(f_1, \cdots, f_r)$, then if we replace the actual f_1 to f_r random functions of the cipher by independent perfect random functions or permutations f_1^* to f_r^* (or, in a more sophisticated version of the same approach, by f_1' to f_r' functions which are sufficiently indistinguishable from f_1^* to f_r^*), then the resulting modified random function F is indistinguishable from a random function (or permutation). This provides a useful method for assessing the soundness of blockcipher constructions.

For instance, in the case of a three-round Feistel construction, a well known theorem first proved by Luby and Rackoff [LR88] provides upper bounds on the $|p - p^*|$ advantage of any testing algorithm A in distinguishing the $2n$-bit random permutation $F = \Psi(f_1^*, f_2^*, f_3^*)$ deduced from three independent perfect random functions f_1^*, f_2^* and f_3^* from a perfect random $2n$-bit permutation F^* with q adaptively chosen queries to the tested instance of F or F^*. This advantage is less than $\frac{q^2}{2^n}$. Another example is for the $F = \Phi_{CBCMAC}(f)$ CBC-MAC construction allowing to derive a tn-bit to n-bit message authentication function from chained invocations of a an n-bit to n-bit function f. It was shown by Bellare, Kilian and Rogaway in [BKR94] that if $q^2t^2 \leq 2^{n+1}$, then the advantage of any testing algorithm A in distinguishing the random function $F = \Phi_{CBCMAC}(f^*)$ derived from a perfect nt -bit to n-bit random function using q adaptively chosen queries is less than $3\frac{q^2t^2}{2^{n+1}}$.

In this paper, we will consider constructions of the form $F = \Phi(f)$, allowing to derive a n-bit to nt-bit function from several invocations of the same instance of an n-bit permutation f, representing a blockcipher of blocksize n. We will show that for suitable modes of operation Φ, the random function $F = \Phi(f^*)$ derived from a perfect n-bit random permutation is indistinguishable from a perfect n-bit to nt-bit random function F^*.

2.2 Random Functions

Through the rest of this paper we are using the following notation:

- I_n denotes the set $\{0,1\}^n$
- $F_{n,m}$ denotes the set ${I_n}^{I_m}$ of functions from I_n into I_m. Thus $|F_{n,m}| = 2^{m.2^n}$
- P_n denotes the set of permutations on I_n. Thus $|P_n| = 2^n!$.

A random function of $F_{n,m}$ is defined as a random variable F of $F_{n,m}$, and can be viewed as a probability distribution $(Pr[F = \varphi])_{\varphi \in F_{n,m}}$ over $F_{n,m}$, or equivalently as a family $(F_\omega)_{\omega \in \Omega}$ of $F_{n,m}$ elements. In particular:

- A n-bit to m-bit key dependent cryptographic function is determined by a randomly selected key value $K \in \mathcal{K}$, and can thus be represented by the random function $F = (f_K)_{K \in \mathcal{K}}$ of $F_{n,m}$.
- A cryptographic construction of the form $F = \Phi(f_1, f_2, \cdots, f_r)$ can be viewed as a random function of $F_{n,m}$ determined by r random functions $f_i \in F_{n_i,m_i}$, $i = 1 \cdots r$.

Definition 1. *We define a perfect random function F^* of $F_{n,m}$ as a uniformly drawn element of $F_{n,m}$. In other words, F^* is associated with the uniform probability distribution over $F_{n,m}$. We define a perfect random permutation f^* on I_n as a uniformly drawn element of P_n. In other words, f^* is associated with the uniform probability distribution over P_n.*

Definition 2. *(q-ary transition probabilities associated to F). Given a random function F of $F_{n,m}$, we define the transition probability $Pr[\mathbf{x} \overset{F}{\mapsto} \mathbf{y}]$ associated with a q-tuple $\mathbf{x}$ of $I_n$inputs and a q-tuple $\mathbf{y}$ of I_m outputs as*

$Pr[\mathbf{x} \overset{F}{\mapsto} \mathbf{y}] = Pr[F(x^1) = y^1 \wedge F(x^2) = y^2 \wedge ... \wedge F(x^q) = y^q]$
$= Pr_{\omega \in \Omega}[F_\omega(x^1) = y^1 \wedge F_\omega(x^2) = y^2 \wedge ... \wedge F_\omega(x^q) = y^q]$

In the sequel we will use the following simple properties:

Property 1. Let f^* be a perfect random permutation on I_n. If $\mathbf{x} = (x^1, ..., x^q)$ is a q-tuple of pairwise distinct I_n values and $\mathbf{y} = (y^1, ..., y^q)$ is a q-tuple of pairwise distinct $I_n$values then $Pr[\mathbf{x} \overset{f^*}{\mapsto} \mathbf{y}] = (|I_n| - q)!/|I_n|! = \frac{(2^n-q)!}{(2^n)!}$

Property 2. Let f^* be a perfect random permutation on I_n. If x and x' are two distinct elements of I_n and δ is any fixed value of I_n, then $\Pr[f^*(x) \oplus f^*(x') = \delta] \leq \frac{2}{2^n}$.

Proof: $\Pr[f^*(x) \oplus f^*(x') = 0] = 0$ since $x \neq x'$. If $\delta \neq 0$, $\Pr[f^*(x) \oplus f^*(x') = \delta] = \frac{2^n \cdot 2^{n-2} \cdots 1}{2^n!} = \frac{1}{2^n - 1} \leq \frac{2}{2^n}$. So, $\Pr[f^*(x) \oplus f^*(x') = \delta] \leq \frac{2}{2^n}$.

2.3 Distinguishing Two Random Functions

In proofs of security such as the one presented in this paper, we want to upper bound the probability of any algorithm to distinguish whether a given fixed φ

function is an instance of a $F = \Phi(f_1^*, f_2^*, .., f_r^*)$ random function of $F_{n,m}$ or an instance of the perfect random function F^*, using less than q queries to φ.

Let A be any distinguishing algorithm of unlimited power that, when input with a φ function of $F_{n,m}$ (which can be modelled as an "oracle tape" in the probabilistic Turing Machine associated with A) selects a fixed number q of distinct chosen or adaptively chosen input values x^i (the queries), obtains the q corresponding output values $y^i = F(x^i)$, and based on these results outputs 0 or 1. Denote by p (resp by p^*) the probability for A to answer 1 when applied to a random instance of F (resp of F^*). We want to find upper bounds on the advantage $Adv_A(F, F^*) = |p - p^*|$ of A in distinguishing F from F^* with q queries.

As first noticed by Patarin [Pa91], the best advantage $Adv_A(F, F^*)$ of any distinguishing algorithm A in distinguishing F from F^* is entirely determined by the q-ary transition probabilities $Pr[\mathbf{x} \stackrel{F}{\mapsto} \mathbf{y}]$ associated with each $\mathbf{x} = (x^1, \cdots, x^q)$ q-tuple of pairwise distinct I_n values and each $\mathbf{y} = (y^1, \cdots, y^q)$ q-tuple of I_m values. The following Theorem, which was first proved in [Pa91] and an equivalent version of which is stated in [Va99], is a very useful tool for deriving upper bounds on $Adv_A(F, F^*)$ based on properties of the $Pr[\mathbf{x} \stackrel{F}{\mapsto} \mathbf{y}]$ q-ary transition probabilities.

Theorem 1. *Let F be a random function of $F_{n,m}$ and F^* be a perfect random function representing a uniformly drawn random element of $F_{n,m}$. Let q be an integer. Denote by X the subset of ${I_n}^q$ containing all the q-tuples $\mathbf{x} = (x^1, \cdots, x^q)$ of pairwise distinct elements. If there exists a subset Y of ${I_m}^q$ and two positive real numbers ϵ_1 and ϵ_2 such that*

1) $|Y| \geq (1 - \epsilon_1) \cdot |I_m|^q$ *(i)*

2) $\forall \mathbf{x} \in X \forall \mathbf{y} \in Y Pr[\mathbf{x} \stackrel{F}{\mapsto} \mathbf{y}] \geq (1 - \epsilon_2) \cdot \frac{1}{|I_m|^q}$ *(ii)*

then for any A distinguishing algorithm using q queries

$$Adv_A(F, F^*) \leq \epsilon_1 + \epsilon_2.$$

In order to improve the selfreadability of this paper, a short proof of Theorem 1, which structure is close to the one of the proof given in [Pa91], is provided in appendix at the end of this paper.

3 Description of Length Increasing Modes of Operation of Blockciphers

We now describe a few natural length increasing modes of operation of a blockcipher. Let us denote the blocksize (in bits) by n, and let us denote by t a fixed integer such that $t \geq 2$. The purpose of one to t blocks modes of operation is to derive an n-bit to tn-bit random function F from an n-bit to tn-bit random function f (representing a blockcipher associated with a random key value K) in such a way that F be indistinguishable from a perfect n-bit to tn bit random function if f is indistinguishable from a perfect random permutation f^*. We

show that the functions associated with the well known OFB mode and with the so-called counter mode of operation are not pseudorandom and introduce enhanced modes of operation, in particular the variant of the counter mode encountered in the UMTS example authentication and key distribution algorithm MILENAGE.

3.1 The Expansion Functions Associated with the Counter and OFB Modes of Operation Are Not Pseudorandom

Definition 3. *Given any t fixed distinct one-block values $c_1, \cdots, c_t \in \{0,1\}^n$ and any random permutation f over $\{0,1\}^n$, the one block to t blocks function F_{CNT} associated with the Counter mode of operation of f is defined as follows:*

$$F_{CNT}(f) : \{0,1\}^n \to \{0,1\}^{nt} \qquad x \mapsto (z_1, \cdots, z_t) = (f(x \oplus c_1), \cdots, f(x \oplus c_t))$$

Given any random permutation f over $\{0,1\}^n$, the 1 block to t blocks function F_{OFB} associated with the output feedback mode of operation of f is defined as follows:

$$F_{OFB}(f) : \{0,1\}^n \to \{0,1\}^{nt} \qquad x \mapsto (z_1, \cdots, z_t)$$

where the z_i are recursively given by $z_1 = f(x); z_2 = f(z_1); \cdots; z_t = f(z_{t-1})$

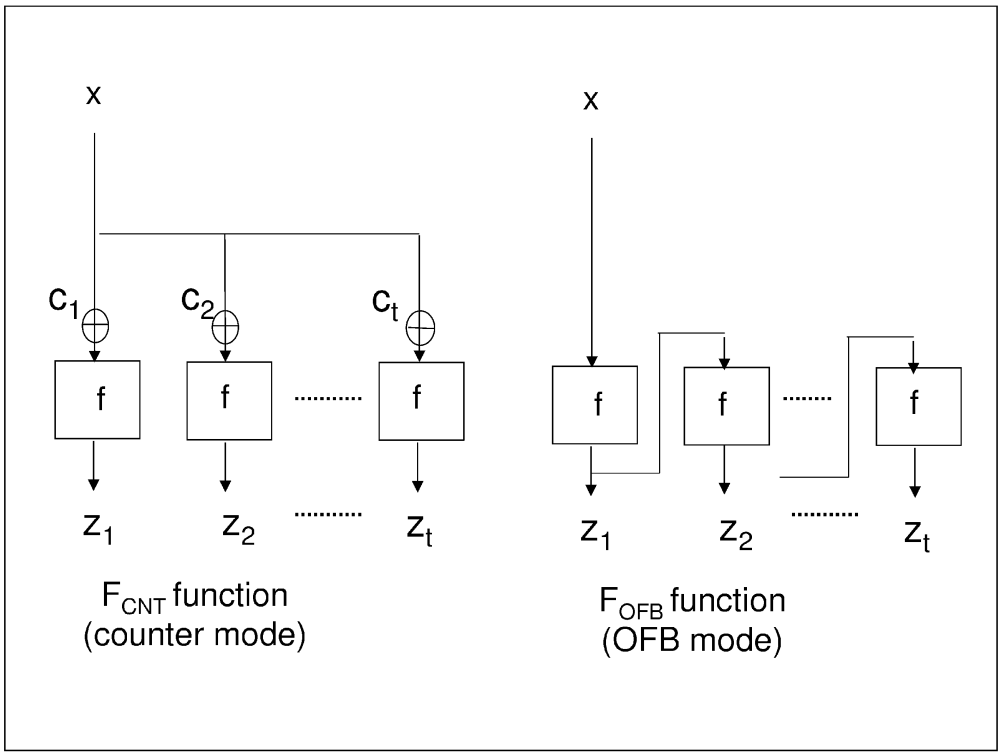


Fig. 1. The counter and OFB modes of operation.

It is straightforward that F_{CNT} and F_{OFB} are not a pseudorandom. As a matter of fact, let us consider the case where F_{CNT} and F_{OFB} are derived

from a perfect random permutation f^*. Let x denote any arbitrary value of $\{0,1\}^n$, and $(z_1, \cdots, z_t)$ denote the $F_{CNT}(x)$ value. For any fixed pair (i,j) of distinct elements of $\{1,2,..,t\}$, let us denote by $(z'_1, \cdots, z'_t)$ the F_{CNT} output value corresponding to the modified input value $x' = x \oplus c_i \oplus c_j$. The obvious property that $z'_i = z_j$ and $z'_j = z_i$ provides a distinguisher of F_{CNT} from a perfect one block to t-blocks random function F^* which requires only two oracle queries. Similarly, to proof that F_{OFB} is not pseudorandom, let us denote by x and $(z_1, \cdots, z_t)$ any arbitrary value of $\{0,1\}^n$ and the $F_{CNT}(x)$ value. With an overwhelming probability, $f^*(x) \neq x$, so that $z_1 \neq x$. Let us denote by x' the modified input value given by $x' = z_1$, and by $(z'_1, \cdots, z'_t)$ the corresponding F_{OFB} output value. It directly follows from the definition of F_{OFB} that for $i = 1, \cdots, t-1$, $z'_i = z_{i+1}$. This provides a distinguisher of F_{OFB} from a perfect one block to t-blocks random function F^* which requires only two oracle queries.

The above distinguishers indeed represent serious weaknesses in operational contexts where the input value of F_{CNT} or F_{OFB} can be controlled by an adversary. For instance if F_{CNT} or F_{OFB} is used for authentication and key distribution purposes, these distinguishers result in a lack of cryptographic separation between the output values z_i. For certain pairs (i,j) of distinct $\{1, \cdots, t\}$ values, an adversary knows how to modify the input x to the data expansion function in order for the i-th output corresponding to the modified input value x', which may for instance represent a publicly available authentication response), to provide her with the j-th output corresponding to the input value x, which may for instance represent an encryption key.

3.2 Modified Counter Mode: The MILENAGE Construction

Figure 2 represents the example UMTS authentication and key distribution algorithm MILENAGE [Mi00]. Its overall structure consists of 6 invocations of a 128-bit blockcipher E_K, e.g. AES associated with a 128-bit subscriber key K. In Figure 2, c_0 to c_4 represent constant 128-bit values, and r_0 to r_5 represent rotation amounts (comprised between 0 and 127) of left circular shifts applied to intermediate 128-bit words. OP_C represents a 128-bit auxiliary (operator customisation) key.

MILENAGE allows to derive four output blocks z_1 to z_4 (which respectively provide an authentication response, an encryption key, a message authentication key, and a one-time key used for masking plaintext data contained in the authentication exchange) from an input block x representing a random authentication challenge. It also allows to derive a message authentication tag z_0 from the x challenge and a 64-bit input word y (which contains an authentication sequence number and some additional authentication management data) using a close variant of the CBC MAC mode of E_K. The security of the MAC function providing z_0, the independence between z_0 and the other output values are outside of the scope of this paper. Some analysis of these features can be found in the MILENAGE design and evaluation report [Mi00]. Let us also ignore the involvement of the OPc constant, and let us focus on the structure of the one block to t block construction allowing to derive the output blocks z_1 to z_4 from

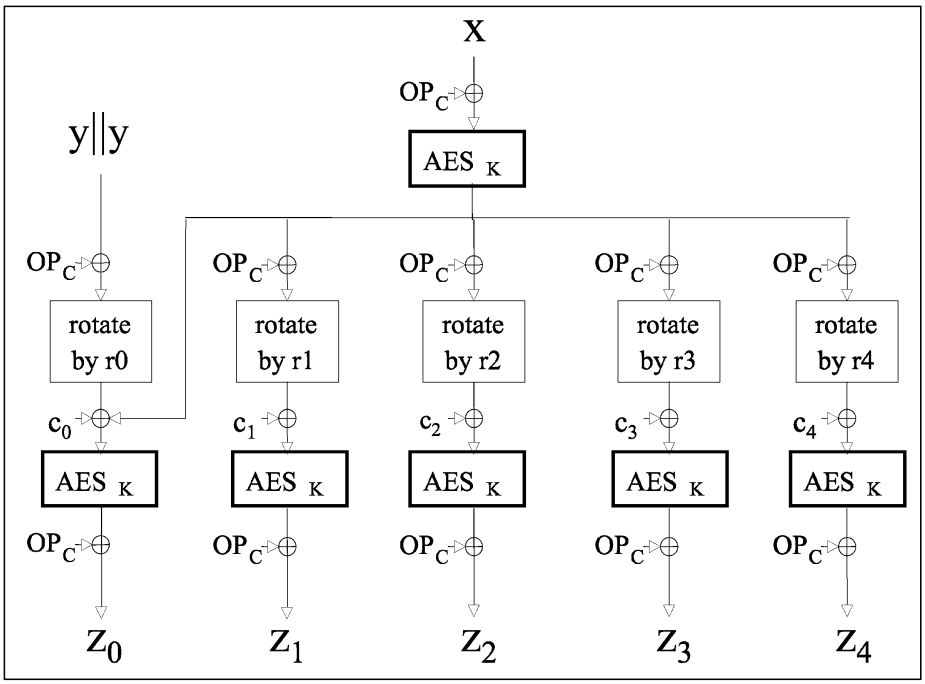


Fig. 2. Milenage.

the input block x . This construction consists of a prewhitening computation, using E_K, of an intermediate block y, followed by applying to y a slight variant (involving some circular rotations) of the counter mode construction.

More formally, given any random permutation f over $\{0,1\}^n$, the 1 block to t blocks function $F_{MIL}(f)$ associated with the MILENAGE construction is defined as follows (cf Figure 3):

$$F_{MIL}(f) : \{0,1\}^n \to \{0,1\}^{nt} \qquad x \mapsto (z_1, \cdots, z_t)$$

$$\texttt{where } z_k = f(rot(f(x), r_k) \oplus c_k) \texttt{ for } k = 1 \texttt{ to } t$$

A detailed statement and proof of the pseudorandomness of the MILENAGE construction are given in Theorem 2 in the next Section. Theorem 2 confirms, with slightly tighter indistinguishability bounds, the claim concerning the pseudorandomness of this construction stated (without the underlying proof) in the MILENAGE design and evaluation report [Mi00].

3.3 Modified OFB Construction

Figure 4 represents a one block to t blocks mode of operation of an n-bit permutation f which structure consists of a prewhitening computation of f providing an intermediate value y, followed by an OFB expansion of y.

More formally, the $F_{MOFB}(f)$ expansion function associated with the modified OFB construction of Figure 4 is defined as follows:

$$F_{MOFB}(f) : \{0,1\}^n \to \{0,1\}^{nt} \qquad x \mapsto (z_1, \cdots, z_t)$$

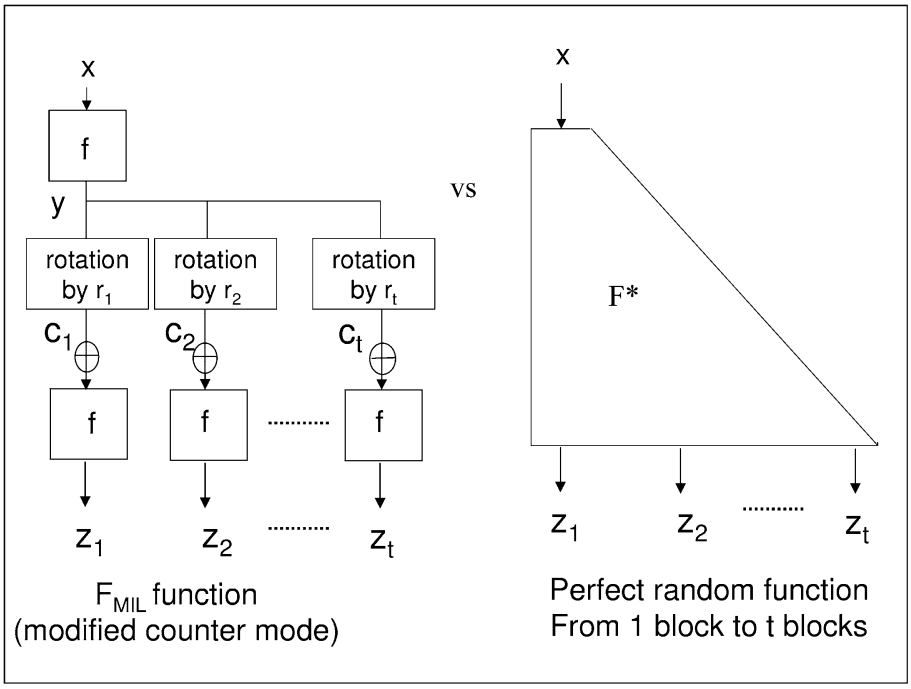


Fig. 3. The MILENAGE modified counter mode construction.

$$\texttt{where } z_1 = f((f(x)) \texttt{ and } z_k = f(f(x) \oplus z_{k-1}) \texttt{ for } k = 2 \texttt{ to } t$$

A short proof of the pseudorandomness of this modified OFB construction is given in Section 5 hereafter.

It is worth noticing that the construction of the above modified OFB mode operation is identical to the one of the ANSI X9.17 PRG mode of operation introduced by Desai et al in [DHY02], so that the pseudorandomness proof (related the associated expansion function) provided in Section 5 is to some extent complementary to the pseudorandomness proof (related to the the associated state transition function) established in [DHY02]. The modified OFB mode of operation is also similar to the keystream generation mode of operation of the KASUMI blockcipher used in the UMTS encryption function f8 [Ka00], up to the fact that in the f8 mode, two additional precautions are taken: the key used in the prewhitening computation differs from the one in the rest of the computations, and in order to prevent collisions between two output blocks from resulting in short cycles in the produced keystream sequence, a mixture of the OFB and counter techniques is applied.

4 Analysis of the Modified Counter Mode Used in MILENAGE

In this Section we proof that if some conditions on the constants $c_k, k \in \{1 \cdots t\}$ and $r_k, k \in \{1 \cdots t\}$ encountered in the MILENAGE construction of Section 3 are satisfied, then the one block to t blocks expansion function $F_{MIL}(f*)$ resulting from applying this construction to the perfect random one-block permutation f^*

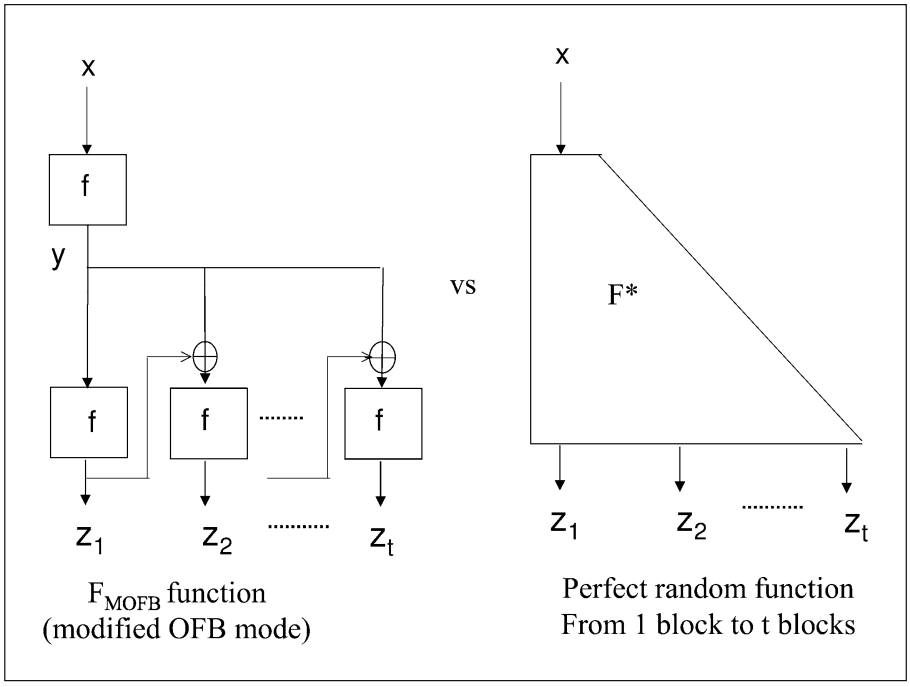


Fig. 4. The modified OFB mode of operation.

is indistinguishable from a perfect random function of $F_{n,tn}$, even if the product of t and the number of queries q is large.

In order to formulate conditions on the constants c_k and r_k, we need to introduce some notation:

- the left circular rotations of a n-bit word w by r bits is denoted by $rot(w, r)$. Rotation amounts (parameter r) are implicitly taken modulo n.
- for any $GF(2)$-linear function $L : \{0,1\}^n \mapsto \{0,1\}^n$, $Ker(L)$ and $Im(L)$ respectively denote the kernel and image vector spaces of L.

With the above notation, these conditions can be expressed as follows:

$$\forall k, l \in \{1 \cdots t\} k \neq l \Rightarrow (c_k \oplus c_l) \notin Im(L) \qquad (C)$$
$$\texttt{where } L = rot(., r_k) \oplus rot(., r_l)$$

The purpose of the above condition is to ensure that for any $y \in \{0,1\}^n$ and any two distinct integers k and $l \in \{1 \cdots t\}$, the values $rot(y, r_k) \oplus c_k$ and $rot(y, r_l) \oplus c_l$ be distinct. If t is less than 2^n, it is easy to find constants c_k and r_k satisfying condition (C) above. In particular, if one takes all r_k equal to zero, condition (C) boils down to requiring that the c_i constants be pairwise distinct.

Theorem 2. *Let n be a fixed integer. Denote by f^* a perfect random permutation of I_n. Let $F = F_{MIL}(f^*)$ denote the random function of $F_{n,tn}$ obtained by applying the MILENAGE construction of Figure 3 to f^*, and F^* denote a perfect random function of $F_{n,t\cdot n}$. If the constants c_k and r_k $(k = 1 \cdots t)$ of the construction satisfy condition (C) above, then for any distinguishing algorithm*

A using any fixed number q of queries such that $\frac{t^2q^2}{2^n} \leq \frac{1}{6}$ *we have*

$$Adv_A(F, F^*) \leq \frac{t^2q^2}{2^{n+1}}$$

Proof. Let us X denote the set of q-tuples $\mathbf{x} = (x^1, \cdots, x^q)$ of pairwise distinct I_n values and Z denote the set of q-tuples $\mathbf{z} = (z^1 = (z_1^1, \cdots, z_t^1), z^2 = (z_1^2, \cdots, z_t^2), \cdots, z^q = (z_1^q, \cdots, z_t^q))$ of pairwise distinct I_{nt} values, such that the tq values $z_1^1, \cdots, z_t^1, \cdots, z_1^q, \cdots, z_t^q$ be pairwise distinct. We want to show that there exist positive real numbers ϵ_1 and ϵ_2 such that:

$$|Z| > (1 - \epsilon_1)|I_{nt}|^q \qquad (i)$$

and

$$\forall \mathbf{x} \in X \forall \mathbf{z} \in Z Pr[x \overset{F}{\mapsto} z] \geq (1 - \epsilon_2) \cdot \frac{1}{|I_{nt}|^q} \qquad (ii)$$

so that that Theorem 1 can be applied.

We have

$$\begin{aligned} \frac{|Z|}{|I_{nt}|^q} &= \frac{2^n \cdot (2^n - 1) \cdots (2^n - tq + 1)}{2^{nqt}} \\ &= 1 \cdot (1 - \frac{1}{2^n}) \cdots (1 - \frac{qt - 1}{2^n}) \\ &\geq 1 - \frac{1}{2^n} \cdot (1 + 2 + \cdots + (qt - 1)) \end{aligned}$$

Since $\frac{1}{2^n} \cdot (1+2+\cdots+(qt-1) = \frac{(qt-1)qt}{2^{n+1}} \leq \frac{q^2t^2}{2^{n+1}}$, we have $|Z| > (1-\epsilon_1)|I_{nt}|^q$, with $\epsilon_1 = \frac{q^2t^2}{2^{n+1}}$.

Let us now show that for any fixed q-tuple of I_n values $\mathbf{x} \in X$ and any q-tuple of I_{nt} values $\mathbf{z} \in Z$, we have $Pr[\mathbf{x} \overset{F}{\mapsto} \mathbf{z}] \geq \frac{1}{2^{ntq}}$.

For that purpose, let us consider from now on any two fixed q-tuples $\mathbf{x} \in X$ and $\mathbf{z} \in Z$. Let us denote by Y the set of q-tuples of pairwise distinct I_n values $\mathbf{y} = (y^1, .., y^q)$. We can partition all the possible computations $\mathbf{x} \overset{F}{\mapsto} \mathbf{z}$ according to the intermediate value $\mathbf{y} = (f^*(x^1), \cdots, f^*(x^q))$ in the F computation.

$$Pr[\mathbf{x} \overset{F}{\mapsto} \mathbf{z}] = \sum_{\mathbf{y} \in Y} Pr[\mathbf{x} \overset{f^*}{\mapsto} \mathbf{y} \wedge \forall i \in \{1..q\} \forall k \in \{1..t\} (rot(y^i, r_k) \oplus c_k) \overset{f^*}{\mapsto} z_k^i]$$

Let us denote by Y' the Y subset of those values $\mathbf{y}$ satisfying the three following additional conditions, which respectively express the requirement that all the f^* input values encountered in the q F computations be pairwise distinct (first and second condition), and that all the f^* outputs encountered in the same computations be also pairwise distinct (third condition).

(I) $\forall i \in \{1..q\} \forall j \in \{1..q\} \forall k \in \{1..t\} x^i \neq rot(y^j, r_k) \oplus c_k$

(II) $\forall i \in \{1..q\} \forall j \in \{1..q\} \forall k \in \{1..t\} \forall l \in \{1..t\}$
$(i, k) \neq (j, l) \Rightarrow rot(y^i, r_k) \oplus c_k \neq rot(y^j, r_l) \oplus c_l$

$(III)\ \forall i \in \{1..q\} \forall j \in \{1..q\} \forall k \in \{1..t\} y^i \neq z^j_k$

We have

$$Pr[\mathbf{x} \overset{F}{\mapsto} \mathbf{z} \geq \sum_{\mathbf{y} \in Y'} Pr[\mathbf{x} \overset{f^*}{\mapsto} \mathbf{y} \wedge \forall i \in \{1..q\} \forall k \in \{1..t\} (rot(y^i, r_k) \oplus c_k) \overset{f^*}{\mapsto} z^i_k]$$

However, if $\mathbf{y} \in Y'$, Property 1 of Section 2 can be applied to the $(t+1)q$ pairwise distinct f^* input values $x^i, i \in \{1..q\}$ and $rot(y^i, r_k) \oplus c_k, i \in \{1..q\}, k \in \{1..t\}$ and to the $(t+1)q$ distinct output values $x^i, i \in \{1..q\}$ and $z^i_k, i \in \{1..q\}, k \in \{1..t\}$, so that

$$Pr[\mathbf{x} \overset{f^*}{\mapsto} \mathbf{y} \wedge \forall i \in \{1..q\} \forall k \in \{1..t\} (rot(y^i, r_k) \oplus c_k) \overset{f^*}{\mapsto} z^i_k] = \frac{(|I_n| - (t+1)q)!}{I_n!} = \frac{(2^n - (t+1)q)!}{2^n!}$$

Therefore, $Pr[\mathbf{x} \overset{F}{\mapsto} \mathbf{z}] \geq |Y'| \frac{(2^n - (t+1)q)!}{2^n!}$ (1)

A lower bound on $|Y'|$ can be established, based on the fact that

$$|Y| = \frac{2^n!}{(2^n - q)!} \quad (2)$$

and on the following properties:

- The fraction of $\mathbf{y}$ vectors of Y such that condition (I) is not satisfied is less than $\frac{q^2 t}{2^n}$ since for any fixed $i \in \{1..q\}, j \in \{1..q\}$ and $k \in \{1..t\}$ the number of $\mathbf{y} \in Y$ q-tuples such that $x^i = rot(y^j, r_k) \oplus c_k$ is $(2^n - 1) \cdots (2^n - q + 1) = \frac{|Y|}{2^n}$ and the set of the $\mathbf{y}$ vectors of Y such that condition (I) is not satisfied is the union set of these $q^2 t$ sets.
- The fraction of $\mathbf{y}$ vectors of Y such that condition (III) is not satisfied is less than $\frac{q^2 t}{2^n}$, by a similar argument.
- The fraction of $\mathbf{y}$ vectors of Y such such that condition (II) is not satisfied is upper bounded by $\frac{q(q-1)}{2} \cdot \frac{t(t-1)}{2} \cdot \frac{1}{2^n - 1}$. As a matter of fact, given any two distinct pairs $(i,k) \neq (j,l)$ of $\{1 \cdots q\} \times \{1 \cdots t\}$, we can upper bound the number of $\mathbf{y}$ vectors of Y such that $rot(y^i, r_k) \oplus c_k = rot(y^j, r_l) \oplus c_l$ by distinguishing the three following cases:

  case 1: $i = j$ and $k \neq l$. Since condition (C) on the constants involved in F is satisfied, there exists no $\mathbf{y}$ vector of Y such that $rot(y^i, r_k) \oplus c_k = rot(y^i, r_l) \oplus c_l$. So case 1 does never occur.

  case 2: $i \neq j$ and $k = l$. For any $\mathbf{y}$ vector of Y, $y^i \neq y^j$. But the $rot(\cdot, r_k) \oplus c_k$ GF(2)-affine mapping of I_n is one to one. Thus, $rot(y^i, r_k) \oplus c_k \neq rot(y^j, r_k) \oplus c_k$. In other words, case 2 does never occur.

  case 3: $i \neq j$ and $k \neq l$ The number of Y q-tuples such that $rot(y^i, r_k) \oplus c_k = rot(y^j, r_l) \oplus c_l$ is $2^n \cdot (2^n - 2) \cdot (2^n - 2) \cdot (2^n - 3) \cdots (2^n - q + 1) = \frac{|Y|}{2^n - 1}$.

  Consequently, the set of $\mathbf{y}$ vectors of Y such such that condition (II) is not satisfied is the union set of the $\frac{q(q-1)}{2} \cdot \frac{t(t-1)}{2}$ sets of cardinal $\frac{|Y|}{2^n - 1}$ considered in case 3, so that the fraction of $\mathbf{y}$ vectors of Y such such that condition (II) is not satisfied is upper bounded by $\frac{q(q-1)}{2} \cdot \frac{t(t-1)}{2} \cdot \frac{1}{2^n - 1}$, as claimed before.

As a consequence of the above properties, the overall fraction of the Y vectors which do not belong to Y' is less than $\frac{2q^2t}{2^n} + \frac{q(q-1)}{2} \cdot \frac{t(t-1)}{2} \cdot \frac{1}{2^n-1}$, i.e.

$$|Y'| \geq (1 - (\frac{2q^2t}{2^n} + \frac{q(q-1)}{2}\frac{t(t-1)}{2}\frac{1}{2^n-1}))|Y| \qquad (3)$$

Now (1) (2) and (3) result in the following inequality:

$$Pr[x \overset{F}{\mapsto} z] \geq (1 - (\frac{2q^2t}{2^n} + \frac{q(q-1)}{2} \cdot \frac{t(t-1)}{2} \cdot \frac{1}{2^n-1})) \cdot \frac{(2^n-(t+1)q)!}{(2^n-q)!}$$

The $\frac{(2^n-(t+1)q)!}{(2^n-q)!}$ term of the above expression can be lower bounded as follows

$$\begin{aligned}
\frac{(2^n-(t+1)q)!}{(2^n-q)!} &= \frac{1}{(2^n-q)(2^n-q-1)\cdots(2^n-((t+1)q-1))} \\
&= \frac{1}{2^{ntq}} \cdot \frac{1}{(1-\frac{q}{2^n}) \cdot (1-\frac{q+1}{2^n}) \cdots (1-\frac{(t+1)q-1}{2^n})} \\
&\geq \frac{1}{2^{ntq}} \cdot (1+\frac{q}{2^n}) \cdot (1+\frac{q+1}{2^n}) \cdots (1+\frac{(t+1)q-1}{2^n}) \\
&\quad (\texttt{due to the fact that if } u<1, \frac{1}{1-u} \geq 1+u) \\
&\geq \frac{1}{2^{ntq}} \cdot (1+\frac{q}{2^n}+\frac{q+1}{2^n}+\cdots+\frac{(t+1)q-1}{2^n}) \\
&= \frac{1}{2^{ntq}}(1+tq\frac{(t+2)q-1}{2^n})
\end{aligned}$$

Thus we have

$$\begin{aligned}
Pr[\mathbf{x} \overset{F}{\mapsto} \mathbf{z}] &\geq \frac{1}{2^{ntq}}(1-(\frac{2q^2t}{2^n}+\frac{q(q-1)}{2} \cdot \frac{t(t-1)}{2} \cdot \frac{1}{2^n-1})) \cdot (1+tq\frac{(t+2)q-1}{2^n}) \\
&= \frac{1}{2^{ntq}}(1+\varepsilon)(1-\varepsilon') \\
\texttt{where } \varepsilon &\triangleq tq\frac{(t+2)q-1}{2^n} \\
\texttt{and } \varepsilon' &\triangleq \frac{2q^2t}{2^n} + \frac{q(q-1)}{2} \cdot \frac{t(t-1)}{2} \cdot \frac{1}{2^n-1}
\end{aligned}$$

Let us show that $\varepsilon > \frac{4}{3}\varepsilon'$. Due to the inequality $\frac{1}{2^n-1} \leq \frac{2}{2^n}$, we have

$$\varepsilon' \leq \frac{qt}{2^{n+1}}(qt+3q-t+1)$$

On the other hand, ε can be rewritten

$$\varepsilon = \frac{qt}{2^{n+1}}(2qt+4q-2)$$

Therefore

$$\varepsilon - \frac{4}{3}\varepsilon' \geq \frac{qt}{2^{n+1}}(\frac{2}{3}qt + \frac{4}{3}t - \frac{10}{3})$$
$$\geq 0 \texttt{ since } t \geq 2 \texttt{ and } q \geq \texttt{ imply } (\frac{2}{3}qt + \frac{4}{3}t - \frac{10}{3}) \geq 0$$

Moreover, it is easy to see (by going back to the definition of ε and using the fact that $t \geq 2$) that $\varepsilon \leq \frac{2t^2q^2}{2^n}$, so that the condition $\frac{t^2q^2}{2^n} \leq \frac{1}{6}$ implies $\varepsilon \leq \frac{1}{3}$.

The relations $\varepsilon \geq \frac{4}{3}\varepsilon'$ and $\varepsilon \leq \frac{1}{3}$ imply $(1+\varepsilon)(1-\varepsilon') \geq 1$ As a matter of fact

$$\begin{aligned}(1+\varepsilon)(1-\varepsilon') &= 1 + \varepsilon - \varepsilon' - \varepsilon\varepsilon' \\ &\geq 1 + \varepsilon - \varepsilon' - \frac{\varepsilon'}{3} \\ &= 1 + \varepsilon - \frac{4}{3}\varepsilon' \\ &\geq 1\end{aligned}$$

Thus we have shown that $Pr[\mathbf{x} \overset{F}{\mapsto} \mathbf{z}] \geq \frac{1}{2^{ntq}}$.

We can now apply Theorem 1 with $\epsilon_1 = \frac{q^2t^2}{2^{2^{n+1}}}$ and $\epsilon_2 = 0$, so that we obtain the upper bound

$$Adv_A(F, F^*) \leq \frac{q^2t^2}{2^{n+1}} \qquad \texttt{QED}$$

The unconditional security result of Theorem 2 is easy to convert (using a standard argument) to a computational security analogue.

Theorem 3. *Let f denote any random permutation of I_n. Let $F = F_{MIL}(f)$ denote the random function of $F_{n,tn}$ obtained by applying to f the MILENAGE construction of Figure 3 (where the constants c_k and r_k $(k = 1 \cdots t)$ are assumed to satisfy condition (C)). Let F^* denote a perfect random function of $F_{n,t\cdot n}$. For any q number of queries such that $\frac{t^2q^2}{2^n} \leq \frac{1}{6}$, if there exists $\varepsilon > 0$ such that for any testing algorithm T with $q(t+1)$ queries and less computational resources (e.g. time, memory, etc.) than any fixed finite or infinite bound R the advantage $Adv_T(f, f^*)$ of T in distinguishing f from a perfect n-bit random permutation $f*$ be such that $Adv_T(f, f^*) < \varepsilon$, then for any distinguishing algorithm A using q queries and less computational resources than R,*

$$Adv_A(F, F^*) < \varepsilon + \frac{t^2q^2}{2^{n+1}}$$

Proof. Let us show that if there existed a testing algorithm A capable to distinguish $F_{MIL}(f)$ from a perfect random function F^* of $F_{n,nt}$ with an advantage $|p - p^*|$ better than $\varepsilon + \frac{q^2t^2}{2^{n+1}}$ using less computational resources than R, then there would exist a testing algorithm T allowing to distinguish f from a perfect random permutation with $q(t+1)$ queries and less computational resources than R with a distinguishing advantage better that ϵ. The test T of a permutation φ would just

consist in performing the test A on $F_{MIL}(\varphi)$. The success probability p' of the algorithm A applied to $F(f^*)$ would be such that $|p' - p^*| \leq \frac{q^2t^2}{2^{n+1}}$ (due to Theorem 2), and therefore, due to the triangular inequality $|p-p'|+|p'-p^*| \geq |p-p^*|$, one would have $|p-p'| \geq \varepsilon$, so that the advantage of T in distinguishing f from f^* would be at least ε QED.

The following heuristic estimate of the success probability of some simple distinguishing attacks against the MILENAGE mode of operation indicates that the $\frac{q^2t^2}{2^{n+1}}$ bound obtained in Theorem 2 is very tight, at least in the case where the r_i rotation amounts are equal to zero. Let us restrict ourselves to this case. Let us consider a $z = (z^1, \cdots, z^q)$ q-tuple of F_{MIL} output value, where each z^i represents a t-tuple of distinct I_n values $z^i_1, \cdots, z^i_t$ Given any two distinct indexes i and j, the occurrence probability of a collision of the form $z^i_k = z^j_l$ can be approximated (under heuristic assumptions) by $\frac{t^2}{2^n}$, so that the overall collision probability among the qt output blocks of F_{MIL} is about $\frac{q(q-1)}{2}\frac{t^2}{2^n}$. Moreover, each collision represents a distinguishing event with an overwhelming probability, due to the fact that $z^i_k = z^j_l$ implies $z^j_k = z^i_l$. Thus the distinguishing probability given by this "attack" is less than (but close to) $\frac{q^2t^2}{2^{n+1}}$. This does not hold in the particular case where $q = 1$, but in this case then another statistical bias, namely the fact that no collisions never occur among the t output blocks, provides a distinguishing property of probability about $\frac{t(t-1)}{2^{n+1}}$, which is again close to $\frac{q^2t^2}{2^{n+1}}$.

5 Analysis of the Modified OFB Mode of Operation

The following analogue of Theorem 2 above can be established for the modified OFB mode of operation (cf Figure 4) introduced in Section 3 .

Theorem 4. *Let n be a fixed integer. Denote by f^* a perfect random permutation of I_n. Let $F = F_{MOFB}(f^*)$ denote the random function of $F_{n,tn}$ obtained by applying the modified construction of Figure 4 to f^*, and F^* denote a perfect random function of $F_{n,t\cdot n}$. For any distinguishing algorithm A using any fixed number of queries q such that $\frac{t^2q^2}{2^n} \leq 1$ we have*

$$Adv_A(F, F*) \leq \frac{7t^2q^2}{2^{n+1}}$$

Proof sketch: the structure of the proof is the same as for the MILENAGE construction. We consider the same X and Z sets of q-tuples as in Section 4. As established in Section 4, $|Z| \geq (1 - \epsilon_1)$, where $\epsilon_1 = \frac{q^2t^2}{2^{n+1}}$. For any fixed $\mathbf{x} \in X$ and $\mathbf{z} \in Z$ q-tuples of input and output values, it can be shown that $Pr[\mathbf{x} \overset{F_{MOFB}(f*)}{\mapsto} \mathbf{z}] \geq \frac{1}{2^{ntq}}(1 - \epsilon_2)$, with $\epsilon_2 = \frac{3q^2t^2}{2^n}$. We can now apply Theorem 1 with $\epsilon_1 = \frac{q^2t^2}{2^{n+1}}$ and $\epsilon_2 = \frac{3q^2t^2}{2^n}$, so that we obtain the upper bound

$$Adv_A(F, F*) \leq \frac{7q^2t^2}{2^{n+1}} \qquad \texttt{QED}$$

6 Conclusion

We have given some evidence that although "one-block-to-many" modes of operation of blockciphers are not as well known and systematically studied so far as "many-blocks-to-one" MAC modes, both kinds of modes are of equal significance for applications such as mobile communications security. We have given security proofs, in the Luby-Rackoff security paradigm, of two simple one to many blocks modes, in which all invocations of the underlying blockciphers involve the same key. We believe that the following topics would deserve some further research:

- systematic investigation of alternative one to many blocks modes, e.g. modes involving more than one key, or modes providing security "beyond the birthday paradox";
- formal proofs of security for hybrid modes of operation including an expansion function, for instance for the combination of the expansion function $x \mapsto (z_1, z_2, z_3, z_4)$ and the message authentication function $(x, y) \mapsto z_0$ provided by the complete MILENAGE construction.

Acknowledgements

I would like to thank Steve Babbage, Diane Godsave and Kaisa Nyberg for helpful comments on a preliminary version of the proof of Theorem 2. I would also like to thank Marine Minier for useful discussions at the beginning of this work.

References

[BDJR97] M. Bellare, A. Desai, E. Jokipii, P. Rogaway, " A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation", Proceedings of 38th Annual Symposium on Foundations of Computer Science, IEEE, 1997.

[BKR94] M. Bellare, J. Kilian, P. Rogaway, "The Security of Cipher Block Chaining". , Advances in Cryptology - CRYPTO'94, LNCS 839, p. 341, Springer-Verlag, Santa Barbara, U.S.A., 1994.

[BM84] M. Blum, S. Micali, "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits" SIAM J. Comput. 13(4), p. 850-864, 1984

[BR00] J. Black, P. Rogaway, "A Block-Cipher Mode of Operation for Parallelizable Message Authentication", Advances in Cryptology – Eurocrypt 2002, Lecture Notes in Computer Science, Vol. 2332, Springer-Verlag, pp. 384–397, 2002.

[DHY02] A. Desai, A. Hevia, Y. Yin, "A Practice-Oriented Treatment of Pseudorandom Number Generators", Eurocrypt 2002, Lecture Notes in Computer Science, Vol. 2332, Springer-Verlag, 2002.

[EJ02] P. Ekdahl, T. Johansson, "A new version of the stream cipher SNOW", proceedings of SAC'02.

[GL89] O.Goldreich, L.Levin, "A hard-core predicate for all one-way functions", Proc. ACM Symp. on Theory of Computing, pp. 25-32, 1989

[HCCJ02] S. Halevi, D. Coppersmith, C.S. Jutla,"Scream: A Software-Efficient Stream Cipher", Advances in Cryptology - FSE 2002, p. 195-209, Springer Verlag, 2002.

[HN00] J. Hastad and M. Näslund, "BMGL: Synchronous Key-stream Generator with Provable security", Revision 1, March 6, 2001) and "A Generalized Interface for the NESSIE Submission BGML", March 15, 2002, available at http://www.cosic.esat.kuleuven.ac.be/nessie/

[JJV02] E. Jaulmes, A. Joux, F. Valette, " On the Security of Randomized CBC-MAC Beyond the Birthday Paradox Limit: A New Construction.", Advances in Cryptology - FSE 2002, p. 237-251, Springer Verlag, 2002, and iacr eprint archive 2001/074

[Ka00] 3rd Generation Partnership Project - Specification of the 3GPP confidentiality and integrity algorithms ; Document 2 (TS 35.202): KASUMI algorithm specification ; Document 1:TS 35.201 f8 and f9 specifications ; Docment TR 33.904: Report on the Evaluation of 3GPP Standard Confidentiality and Integrity Algorithms, available at http://www.3gpp.org

[LR88] M. Luby, C. Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Function", Siam Journal on Computing , vol. 17, p. 373, 1988.

[Ma92] U. Maurer, "A Simplified and generalised treatment of Luby-Rackoff Pseudo-random Permutation Generators", Advances in Cryptology - Eurocrypt'92, LNCS 658 , p. 239, Springer Verlag, 1992.

[Mi00] 3rd Generation Partnership Project - Specification of the MILENAGE algorithm set: An example algorithm Set for the 3GPP Authentication and Key Generation functions f1, f1*, f2, f3, f4, f5 and f5* - Document 2 (TS 35.206): Algorithm specification ; Document 5 (TR 35.909): Summary and results of design and evaluation, available at http://www.3gpp.org

[Pa91] J. Patarin, "Etude de Générateurs de Permutation Basés sur le Schéma du D.E.S.", Phd. Thesis, University of Paris VI, 1991.

[Pa92] J. Patarin, "How to Construct Pseudorandom and Super Pseudorandom Permutations from One Single Pseudorandom Function", Advances in Cryptology - Eurocrypt'92, LNCS 658 , p. 256, Springer Verlag, 1992.

[PR00] E. Petrank, C. Rackoff,"CBC MAC for Real-Time Data Sources", Journal of Cryptology 13(3), p. 315–338, 2000

[RC98] P. Rogaway, D. Coppersmith, "A Software-Optimized Encryption Algorithm", Journal of Cryptology 11(4), p. 273-287, 1998

[Va98] S. Vaudenay, "Provable Security for Block Ciphers by Decorrelation", STACS'98, Paris, France,Lecture Notes in Computer Science No. 1373, p. 249-275, Springer-Verlag, 1998.

[Va99] S. Vaudenay, "On Provable Security for Conventional Cryptography", Proc. ICISC'99, invited lecture.

Appendix: A Short Proof of Theorem 1

Let us restrict ourselves to the case of any fixed deterministic algorithm A which uses q adaptively chosen queries (the generalization to the case of a probabilistic algorithm is easy).

A has the property that if the q-tuple of outputs encountered during an A computation is $\mathbf{y} = (y^1, \cdots, y^q)$, the value of the q-tuple $\mathbf{x} = (x^1, \cdots, x^q)$ of

query inputs encountered during this computation is entirely determined. This is easy to prove by induction: the initial query input x^1 is fixed ; if for a given A computation the first query output is y^1, then x^2 is determined, etc.. We denote by $\mathbf{x}(\mathbf{y})$ the single q-tuple of query inputs corresponding to any possible $\mathbf{y}$ q-tuple of query outputs, and we denote by S_A the subset of those $\mathbf{y} \in {I_m}^q$ values such that if the q-tuples $\mathbf{x}(\mathbf{y})$ and $\mathbf{y}$ of query inputs and outputs are encountered in a A computation, then A outputs the answer 1.

The probabilities p and p^* can be expressed using S_A as

$p = \sum_{\mathbf{y} \in S_A} Pr[\mathbf{x}(\mathbf{y}) \stackrel{F}{\mapsto} \mathbf{y}]$ and

$p^* = \sum_{\mathbf{y} \in S_A} Pr[\mathbf{x}(\mathbf{y}) \stackrel{F^*}{\mapsto} \mathbf{y}]$

We can now lower bound p using the following inequalities:

$p \geq \sum_{\mathbf{y} \in S_A \cap Y} (1 - \epsilon_2) \cdot Pr[\mathbf{x}(\mathbf{y}) \stackrel{F^*}{\mapsto} \mathbf{y}]$ due to inequality (ii)

$\geq \sum_{\mathbf{y} \in S_A} (1 - \epsilon_2) \cdot Pr[\mathbf{x}(\mathbf{y}) \stackrel{F^*}{\mapsto} \mathbf{y}] - \sum_{\mathbf{y} \in {I_m}^q - Y} (1 - \epsilon_2) \cdot Pr[\mathbf{x}(\mathbf{y}) \stackrel{F^*}{\mapsto} \mathbf{y}]$

But $\sum_{\mathbf{y} \in S_A} (1 - \epsilon_2) \cdot Pr[\mathbf{x}(\mathbf{y}) \stackrel{F^*}{\mapsto} \mathbf{y}] = (1 - \epsilon_2) \cdot p^*$

and

$\sum_{\mathbf{y} \in {I_m}^q - Y} (1 - \epsilon_2) \cdot Pr[\mathbf{x}(\mathbf{y}) \stackrel{F^*}{\mapsto} \mathbf{y}] = (1 - \epsilon_2) \cdot \frac{|I_m|^q - |Y|}{|I_m|^q} \leq (1 - \epsilon_2) \cdot \epsilon_1$ due to inequality (i).

Therefore, $p \geq (1 - \epsilon_2)(p^* - \epsilon_1) = p^* - \epsilon_1 - \epsilon_2 \cdot p^* + \epsilon_1 \cdot \epsilon_2$

thus finally (using $p^* \leq 1$ and $\epsilon_1 \cdot \epsilon_2 \geq 0$)

$p \geq p * -\epsilon_1 - \epsilon_2$ (a)

If we now consider the distinguisher A' which outputs are the inverse of those of A (i.e. A' answers 0 iff A answers 1), we obtain an inequality involving this time $1 - p$ and $1 - p^*$:

$(1 - p) \geq (1 - p^*) - \epsilon_1 - \epsilon_2$ (b)

Combining inequalities (a) and (b), we obtain $|p - p^*| \leq \epsilon_1 + \epsilon_2$ QED.

Cryptanalysis of a White Box AES Implementation

Olivier Billet, Henri Gilbert, and Charaf Ech-Chatbi*

France Télécom R&D
38–40, rue du Général Leclerc
92794 Issy les Moulineaux Cedex 9 — France
{olivier.billet,henri.gilbert}@francetelecom.com
charaf_echchatbi@yahoo.fr

Abstract. The white box attack context as described in [1, 2] is the common setting where cryptographic software is executed in an untrusted environment—i.e. an attacker has gained access to the implementation of cryptographic algorithms, and can observe or manipulate the dynamic execution of whole or part of the algorithms. In this paper, we present an efficient practical attack against the obfuscated AES implementation [1] proposed at SAC 2002 as a means to protect AES software operated in the white box context against key exposure. We explain in details how to extract the whole AES secret key embedded in such a white box AES implementation, with negligible memory and worst time complexity 2^{30}.

Keywords: white box, AES, block ciphers, tamper resistance, software piracy, implementation.

1 Introduction

One of the consequences of the ever spreading use of cryptology within mass applications—e.g. email, web servers access, digital content distribution, and so on—implemented in software on standard terminals, like PCs, PDAs, or mobile phones, is that cryptologic algorithms are quite often executed in an untrusted environment. The usual "black box" model, where keys and cryptographic algorithms are confined and executed in a logically protected and tamper resistant cryptographic module, like a smart card, is no longer applicable. This situation motivated the introduction of a new setting, coined "white box" context of execution: the software representing cryptographic algorithms, cryptographic keys when separate from the cryptographic software, and dynamic data produced during the execution of all or part of the cryptographic algorithms, are exposed to being accessed or even manipulated by malicious processes hosted by the same machine—which may be controlled either by an outsider or by the legitimate user of the host terminal. Cryptographic applications running in the white box context of execution are highly vulnerable to the most severe form

* work performed at France Télécom R&D

of attack, namely the leakage of the cryptographic keys. Thus, the protection cryptographic algorithms would offer in the black box model of execution vanish.

This security issue is at the origin of the introduction, in a pair of seminal articles [2, 1] S. Chow, P. Eisen, H. Johnson, and P.C. van Oorschot, of a new protection technique preventing from key leakage for cryptographic software run in the white box context. It consists in implementing key-instantiated versions of an algorithm, as the composition of a series of lookup tables, each look-up table concealing some components of the algorithm. Implementations of an algorithm resulting from this protection technique are named white-box implementations. White box implementations of the DES and AES blockciphers were respectively described in [2] and [1]. Short after the publication of [2], it was shown by M. Jacob, D. Boneh and D. Felten in [3], that the obfuscation technique applied in [2] was insecure, i.e. that a low complexity attack requiring few accesses (with partly chosen input values) to lookup tables representing external DES rounds, allowed to extract the key from a white box DES implementation. However, the attack technique of [3] is not applicable to the white box implementation of AES described in [1] due to the additional protection provided by some extra features introduced by [1]. More precisely, a fundamental difference between both implementations results from the application, in the case of AES, of so-called external encodings. One of the main security consequences of this extra feature—which description is provided in Sec. 2—is that in the case of AES, and unlike DES, the protection of external rounds is not weaker than the protection of internal rounds. Since the attack strategy of [3] is essentially based upon the extra weakness of external rounds, it is not applicable to the AES implementation described in [1]. To the best of our knowledge, no realistic attack against the white box implementation of [1] has been proposed so far.

In this paper, we present a practical low complexity attack—i.e. with negligible memory, and work factor $3 \cdot 2^{28} < 2^{30}$—of the AES white box implementation proposed in [1]. The conducting idea of the attack is that though none of the lookup tables, when considered individually, leaks sensitive information related to the AES key in an obvious way, the analysis (based on the observation of related input/output values) of lookup tables composition, reveals information on the encodings embedded in those lookup tables. We show that the information provided by the analysis of such tables during three consecutive encoded rounds, allows an attacker to entirely recover the AES 128-bit secret key of an obfuscated AES implementation. The key steps of the proposed attack were successfully implemented in `C++`, and confirmed by computer experiments.

This paper is organized as follows. In Section 2, we describe the white box AES implementation as proposed by [1]. In Section 3 we show how to extract the secret key. The last section concludes the paper.

2 Description of the White Box AES Implementation

We now describe the implementation proposed in [1]. The general strategy is to merge several steps of the AES round function into table lookups, blended by input/output encodings, and mixing bijections.

Internal encodings (resp. mixing bijections) are non-linear (resp. GF(2)-linear) and introduce confusion (resp. diffusion) in the representation of the intermediate blocks of the computation. Their inclusion in the implementation must respect the fact that two consecutive tables in the data flow have matching output and input encodings, as well as matching mixing bijections, at their boundary.

Apart from the above pairwise canceling internal transformations, another obfuscation technique called external encoding is used. It consists in feeding the obfuscated implementation with AES inputs in an encoded form. At the same time, the implementation also outputs the AES encrypted values in an encoded form. Thus, the implementation does not exactly achieve an AES computation $Y = E_K(X)$, but a modified computation $Y = E'_K(X) = G \circ E_K \circ F^{-1}(X)$. The external input/output encodings G and F^{-1} have to be annihilated on the peer site—e.g. a server when the AES obfuscated implementation is embedded in a software player—in order to compute ${E'}_K^{-1}$. Though the encodings G and F^{-1} suggested in [1] are hereafter taken into account, our attack is not highly dependent upon their exact specification. One of the main consequences of using external encodings is that internal input/output encodings can be used to blend the first and last round, in addition to inner rounds' blending. This prevents attackers from exploiting specific weaknesses one would otherwise encounter against external rounds of obfuscated implementations [3].

Let us hereafter denote AES-128 the AES version operating on 128 bits blocks. Recall [4, 5] that the AES-128 round function is made of the four steps described in Fig. 1 operating on the 16 bytes of a 4×4 state array. The AES-128

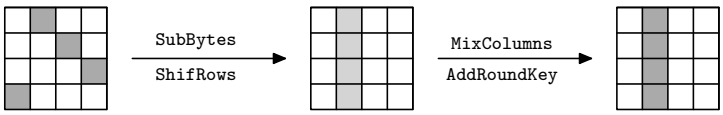


Fig. 1. tracking four bytes during an AES round

considered in [1] consists of 10 such rounds; a preliminary `AddRoundKey` step is performed before the first round, and `MixColumns` is omitted in the final round. Let us index the state bytes by their row and column numbers (i, j) in the state array. If the S-box function operating on bytes during the `SubBytes` step is denoted by S, define for any round r and any byte (i, j) with indexes taken modulo 4:

$$1 \leq r \leq 9 \qquad T^r_{i,j}(x) := S\left(x \oplus k^r_{i,j}\right) ,$$
$$T^{10}_{i,j}(x) := S\left(x \oplus k^{10}_{i,j}\right) \oplus k^{11}_{i,j-i} .$$

(Note that we shifted the round index of the original AES-128 by 1, and that the post-whitening key $k_{i,j}^{11}$ occuring in the last round is absorbed by the definition of the last function $T_{i,j}^{10}$.) Now each 4-byte column of the output of the `SubByte` plus `ShiftRows` steps will contribute to the 4-byte column of the state array after `MixColumns`, and those four bytes are related to the former by a 32×8 submatrix MC_i of the 32×32 matrix MC representing `MixColumns`. Now the entire function can be described by a lookup table. However, it is necessary to obfuscate this table, which leads to encode its 4-bit input and output nibbles—using concatenated non-linear permutations in and out respectively.

To add to the diffusion, 8×8 affine "mixing" bijection is inserted before $T_{i,j}^r$ and a 32×32 affine bijection MB is inserted after the `MixColumn` part. The resulting lookup table is depicted in Fig. 2 as the `sub` table. The 32×8 linear mapping of Fig. 2 is associated with $\mathrm{MB} \times \mathrm{MC}_i$.

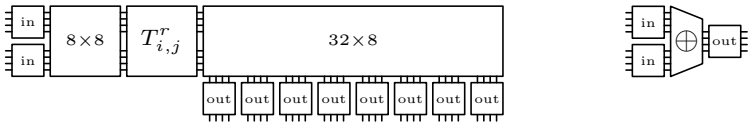


Fig. 2. `sub` table (type II) and `xor` table (type IV)

To cancel the effect of MB, a lookup table takes care of the inversion. However, instead of constructing a huge table for the entire 32×32 matrix, the mapping MB^{-1} is split into four submatrices $\left(\mathrm{MB}^{-1}\right)_i$, just like with the `MixColumns` matrix MC. This results in the lookup table depicted in Fig. 3.

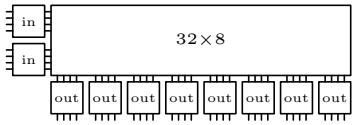


Fig. 3. `untwist` table (type III)

Finally, external input and output encodings are implemented, using two sets of sixteen 8-bit to 128-bit lookup tables depicted in Fig. 4. Each external input encoding table represents the linear mapping associated with one 128×8 vertical stripe of a 128×128 matrix—the composition of M_F and the concatenation of the input mixing bijections for $T_{i,j}^1$'s inverses—surrounded by 4-bit to 4-bit non-linear encodings. Each external output encoding table represents one 128×8 vertical stripe of a 128×128 parasitic matrix—the composition of one round 10's output mixing bijection's inverse, one of the mappings $T_{i,j}^{10}$, and 128×8 vertical stripe of a 128×128 parasitic matrix M_G—surrounded by 4-bit to 4-bit non linear encodings. The outputs of the 16 `external input encoding` tables have to be decoded, xored together and reencoded to complete the implementation.

This is done by using 15×32 additional `xor` tables per 128-block. The same number of `xor` tables is needed to support the 16 `extern_encode` tables.

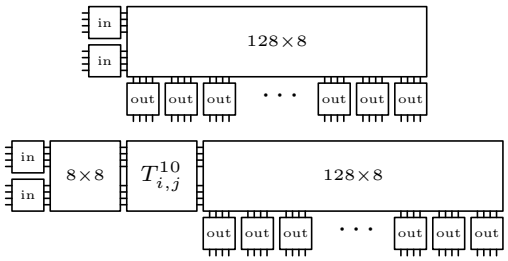


Fig. 4. `extern_encode` tables for input and output respectively (type I)

Thus, in order to implement a white box instance of AES-128 associated with a key K, $9 \cdot 4 \cdot 4$ `sub` tables, $9 \cdot 4 \cdot 4$ `untwist` tables, $9 \cdot 4 \cdot 3 \cdot 8$ `xor` tables supporting `sub` tables, $9 \cdot 4 \cdot 3 \cdot 8$ `xor` tables supporting `untwist` tables, $2 \cdot 16$ `extern_encode` tables, and $2 \cdot 15 \cdot 32$ `xor` tables supporting `extern_encode` tables are needed. Therefore, the total size of lookup tables in an AES-128 white box implementation is 770 048 bytes.

3 Cryptanalysis of the White Box AES Implementation

We now describe a very efficient attack against the white box AES implementation of [1]. The leading idea is that, though recovering information about the key by a local inspection of the lookup tables seems difficult—lookup tables were designed to satisfy so-called diversity and ambiguity criteria—recovering information by analyzing compositions of lookup tables corresponding to one encoded AES round is easier. More precisely, it is convenient to analyze each of the four mappings between four bytes of the input state array, and the four corresponding bytes of the output state array, which together form an encoded AES round. Each such mapping can be conceptualized by the box in Fig. 5, where we can choose inputs and observe outputs, whereas intermediate values remain concealed. Let us denote this box by R^r_j. Each R^r_j box is made of four 8-bit to 8-bit parasitic input permutations $P^r_{i,j}$ (resp. output permutations $Q^r_{i,j}$) constructed as the composition of two concatenated 4-bit to 4-bit input (resp. output) encodings, and one 8-bit to 8-bit linear mixing bijection. Due to the fact that internal input encodings plus linear mixing bijections and linear mixing bijections plus output encodings mutually cancel out at the boundary between two rounds r and $r+1$, each $Q^r_{i,j}$ is the inverse of $P^{r+1}_{i,j}$.

The attack proceeds in three steps. First of all, we recover the non-affine part of the parasites Q^r_i in round $r = 1, \ldots, 9$, i.e. we determine Q^r_i up to unknown affine bijections, and thus get at the same time the non-affine part of the inverse P^{r+1}_i of round $r+1$, $r = 1, \ldots, 9$. At this stage we are in the setting depicted in Fig. 5, but this time the permutations P_i and Q_i are now

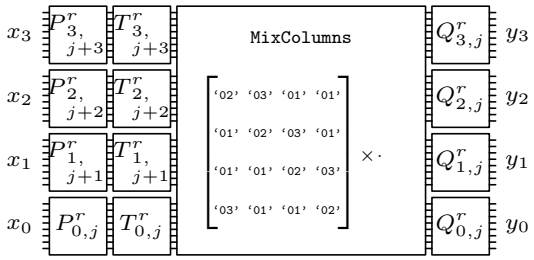


Fig. 5. One of the four R_j^r mappings, $j = 0, \ldots, 3$

GF(2)-affine, except for the permutation $P_{i,j}^1$ whose non-affine part has not been determined. In a second step, we recover those GF(2)-affine mappings (but $P_{i,j}^1$ and $Q_{i,j}^1$), first up to an unknown GF(2^8)-affine bijection, and then entirely. Eventually combining all this information in a third step, we extract the AES-128 key.

3.1 Recovering Non-Linear Parts

Consider the mapping R_j^r. We are trying to remove the non-linearity in the parasites $(Q_i^r)_{i=0,\ldots,3}$. To this end consider y_0 as a function of (x_0, x_1, x_2, x_3), and fix the values of x_1, x_2, and x_3 to some constants, say c_1, c_2, and c_3. One easily checks that there exists two constants in GF(2^8), namely α independent of c_1, c_2, c_3, and β_{c_1,c_2,c_3}, such that

$$y_0(x, c_1, c_2, c_3) = Q_{0,j}^r \left(\alpha T_{0,j}^r \left(P_{0,j}^r(x) \right) \oplus \beta_{c_1,c_2,c_3} \right) \ .$$

Since x only takes 256 values, those mappings are known by input/output, as well as their inverses. Also, varying one constant (say c_3) into the whole GF(2^8), and keeping the other one fixed, has the effect that $\beta_{c_1,c_2,c_3'}$ takes all the values in GF(2^8). We are thus able to produce—as lookup tables, of course—all the functions

$$y_0(x, c_1, c_2, c_3) \circ y_0(x, c_1, c_2, c_3')^{-1} = Q_0 \left(Q_0^{-1}(x) \oplus \beta \right) \ , \tag{1}$$

where $\beta = \beta_{c_1,c_2,c_3'} \oplus \beta_{c_1,c_2,c_3}$ takes all the values in GF(2^8). This leads to the problem of recovering Q_0, or at least its non-linear part from the set of all those lookup tables. Note that since functions are given as lookup tables, we are not provided with the underlying translations: we only know the unordered set of functions corresponding to the 256 translations. As this problem is of independent interest, we state it, along with a solution, in a standalone context.

Theorem 1. *Given a set of functions* $\mathcal{S} = \{Q \circ \oplus_\beta \circ Q^{-1}\}_{\beta \in \mathrm{GF}(2^8)}$ *given by values, where* Q *is a permutation of* GF(2^8) *and* $\oplus_\beta$ *is the translation by* β *in* GF(2^8)*, one can construct a particular solution* $\widetilde{Q}$ *such that there exists an affine mapping* A *so that* $\widetilde{Q} = Q \circ A$.

Proof. There is an isomorphism between the commutative groups $\big(\mathrm{GF}(2)^8, \oplus\big)$ and $(\mathcal{S}, \circ)$, given by

$$\begin{array}{rccc} \varphi : & \mathcal{S} & \longrightarrow & \mathrm{GF}(2)^8 \\ & Q \circ \oplus_\beta \circ Q^{-1} & \longmapsto & [\beta] \,, \end{array}$$

where $[\beta]$ denotes the embedding of the element β into the vector space $\mathrm{GF}(2)^8$ with canonical base $([e_i])_{i=1,\ldots,8}$. The issue is we do not know this isomorphism. The general idea of the proof is to recover this isomorphism up to an unknown linear bijection, i.e. to recover a known isomorphism ψ equal to φ up to an unknown linear bijection. To this end, first select from $\mathcal{S}$ a tuple $(f_1, \ldots, f_8)$ of 8 functions such that their images through φ constitute a base of $\mathrm{GF}(2)^8$. Although we do not know φ—and thus the underlying translations $[\beta_i] = \varphi(f_i)$ for each $f_i = Q \circ \oplus_{\beta_i} \circ Q^{-1}$—this can easily be done by gradually selecting f_1 to f_8 so that they span the whole set $\mathcal{S}$ through composition, that is

$$\forall f \in \mathcal{S}, \quad \exists!(\varepsilon_1, \ldots, \varepsilon_8) \in \{0,1\}^8, \qquad f = f_8^{\varepsilon_8} \circ f_7^{\varepsilon_7} \circ \cdots \circ f_1^{\varepsilon_1} \,, \tag{2}$$

where $f_i^1 = f_i$ and f_i^0 denotes the identity function. An efficient algorithm to compute such a tuple of functions $(f_1, \ldots, f_8)$ is described at the end of this paragraph.

Now since $([\beta_i])_{i=1\ldots8}$ is a base of $\mathrm{GF}(2)^8$, there exists a unique one-to-one linear change of base L mapping $[e_i]$ onto $[\beta_i]$ for all $i = 1, \ldots, 8$. Also define the isomorphism $\psi \stackrel{\text{def}}{=} L^{-1} \circ \varphi$ between $(\mathcal{S}, \circ)$ and $\big(\mathrm{GF}(2)^8, \oplus\big)$. One checks that ψ can be efficiently recovered, by using the unique decomposition given by Eq. 2. Indeed, for any $f \in \mathcal{S}$ the unique tuple of binary values $(\varepsilon_1, \ldots, \varepsilon_8)$ verifying Eq. 2 is easily computed—an exhaustive search would be quick enough, but we give a better algorithm at the end of this paragraph. By successively applying φ

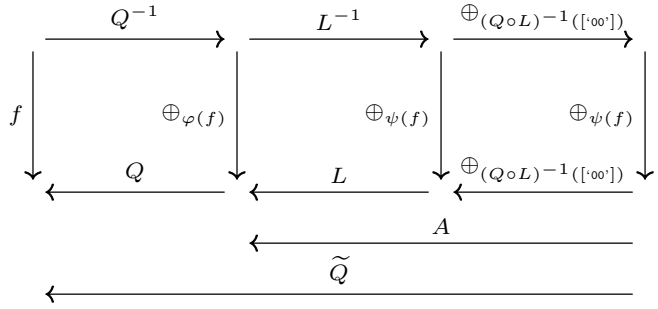


Fig. 6. relating f, $\psi(f)$, and $\tilde{Q}$

and L^{-1} to f, one obtains

$$\psi(f) = L^{-1}\big(\varphi(f)\big) = L^{-1}\left(\bigoplus_{i=1\ldots8} \varepsilon_i[\beta_i]\right) = \bigoplus_{i=1\ldots8} \varepsilon_i[e_i] \,.$$

Thus the isomorphism ψ is entirely determined.

Let us explain how to recover Q from the knowledge of ψ, up to an unknown affine transformation A. For that purpose, consider the commutative diagram of Fig. 6, and define the GF(2)-affine one-to-one mapping A by

$$A(x) \stackrel{\text{def}}{=} L\big(x \oplus (Q \circ L)^{-1}([\text{`00'}])\big) = L(x) \oplus Q^{-1}([\text{`00'}]) \ ,$$

and let us set

$$\widetilde{Q} \stackrel{\text{def}}{=} Q \circ A \ .$$

One verifies that $\widetilde{Q}^{-1}(\text{`00'}) = [\text{`00'}]$. By applying the above definition of $\widetilde{Q}$, or equivalently by inspecting the commutative diagram of Fig.6, one checks that $f = \widetilde{Q} \circ \oplus_{\psi(f)} \circ \widetilde{Q}^{-1}$. Hence,

$$f(\text{`00'}) = \widetilde{Q}(\psi(f)) \ .$$

From our knowledge of ψ and f, we can therefore compute $\widetilde{Q} = Q \circ A$. □

Now, we propose an efficient algorithm—time complexity is at most 2^{24}—that chooses a tuple $(f_1, \ldots, f_8)$ on the fly, and computes the corresponding mapping ψ. It was successfully implemented in C++.

INPUT : $\mathcal{S}$
OUTPUT : $\mathcal{R} \subset \mathcal{S} \times \mathrm{GF}(2^8)$ such that $\forall (f,\beta) \in \mathcal{R},\ \psi(f) = [\beta]$
ALGORITHM :
$\mathcal{R} \leftarrow \{(id, \text{`00'})\}$
$\psi(id) = [\text{`00'}]$
$e \leftarrow$ `01'
while $\#\mathcal{R} < 2^8$ **do**
  $\mathcal{S} \leftarrow \mathcal{S} \setminus \{f\}$
  if $(f, \cdot) \notin \mathcal{R}$ **then**
    $e \leftarrow$ `02' $\times\, e$
    $\psi(f) = [e]$
    foreach $(g, \eta) \in \mathcal{R}$ **do**
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{(f \circ g, [e] \oplus [\eta])\}$
      $\psi(f \circ g) = [e] \oplus [\eta]$
    enddo
  endif
endwhile

Going back to our initial motivation, Theorem 1 enables us to recover for any round $r = 1, \ldots, 9$, the non-linear part $\widetilde{Q}^r_{i,j}$ of $Q^r_{i,j}$, i.e. such that ${\widetilde{Q}^r_{i,j}}^{-1} \circ Q^r_{i,j}$ is an affine mapping ${A^r_{i,j}}^{-1}$. Given the fact that for the next round, the input encoding $P^{r+1}_{i,j}$ must match the output encoding $Q^r_{i,j}$ of the previous round—that is $P^{r+1}_{i,j} \circ Q^r_{i,j}$ must be the identity—we have that $P^{r+1}_{i,j} \circ \widetilde{Q}^r_{i,j}$ is exactly the mapping $A^r_{i,j}$. Thus, we have reduced the original problem depicted in Fig. 5 where all P and Q are non-linear and matching, to one where they are affine and still matching. The next step is to recover those affine mappings, which is the subject of next sections.

3.2 Relations Between Affine Parasites

So let us start again with the setting depicted in Fig.5, except for the fact that all parasitic mappings $P^r_{i,j}$ and $Q^r_{i,j}$ are now affine. Since the problem is identical for each round, we drop the subscripts r and j without loss of generality. We have access to the following functions as lookup tables

$$\begin{cases} y_0(x_0,x_1,x_2,x_3) = Q_0\left(\text{`02'}\cdot T'_0(x_0) \oplus \text{`03'}\cdot T'_1(x_1) \oplus \text{`01'}\cdot T'_2(x_2) \oplus \text{`01'}\cdot T'_3(x_3)\right) \\ y_1(x_0,x_1,x_2,x_3) = Q_1\left(\text{`01'}\cdot T'_0(x_0) \oplus \text{`02'}\cdot T'_1(x_1) \oplus \text{`03'}\cdot T'_2(x_2) \oplus \text{`01'}\cdot T'_3(x_3)\right) \\ y_2(x_0,x_1,x_2,x_3) = Q_2\left(\text{`01'}\cdot T'_0(x_0) \oplus \text{`01'}\cdot T'_1(x_1) \oplus \text{`02'}\cdot T'_2(x_2) \oplus \text{`03'}\cdot T'_3(x_3)\right) \\ y_3(x_0,x_1,x_2,x_3) = Q_3\left(\text{`03'}\cdot T'_0(x_0) \oplus \text{`01'}\cdot T'_1(x_1) \oplus \text{`01'}\cdot T'_2(x_2) \oplus \text{`02'}\cdot T'_3(x_3)\right) \end{cases}$$

with the shortcut $T'_i = T_i \circ P_i$. Actually there is one more issue, which is that we do know the set $\{y_i\}_{i=0,\ldots,3}$ but we do not know the labels to put on each function. Put in another way, we know those functions have the general form

$$y_i(x_0,x_1,x_2,x_3) = Q\left(\alpha_{i,0}\cdot T'_0(x_0) \oplus \alpha_{i,1}\cdot T'_1(x_1) \oplus \alpha_{i,2}\cdot T'_2(x_2) \oplus \alpha_{i,3}\cdot T'_3(x_3)\right)$$

but we do not know what the underlying coefficients $\alpha_{i,j}$ occurring from the `MixColumn` step are. Let us hereafter denote by Λ_α the matrix over $\mathrm{GF}(2)^8$ of the multiplication by α.

Before going any further, let us state a very useful property. Though simple, it is a corner stone in the strategy we designed for the affine parasites' recovery, as well as in resolving the above mentioned renaming issue.

Proposition 1. *For any pair (y_i, y_j) as introduced above, there exists a unique linear mapping L and a unique constant c such that,*

$$\forall x_0 \in \mathrm{GF}(2^8), \quad y_i(x_0, \text{`00'}, \text{`00'}, \text{`00'}) = L\left(y_j(x_0, \text{`00'}, \text{`00'}, \text{`00'})\right) \oplus c\,. \tag{3}$$

Proof. Decompose the affine maps $Q_i(x) = A_i(x) \oplus q_i$ and $Q_j(x) = A_j(x) \oplus q_j$, where A_i and A_j are linear, q_i and q_j constants. Hence,

$$\begin{aligned} y_i(x, \text{`00'}, \text{`00'}, \text{`00'}) &= A_i(\alpha_{i,0}\cdot T'_0(x) \oplus c_i) \oplus q_i\ , \\ y_j(x, \text{`00'}, \text{`00'}, \text{`00'}) &= A_j(\alpha_{j,0}\cdot T'_0(x) \oplus c_j) \oplus q_j\ . \end{aligned}$$

Thus, by taking $L = A_i \circ \Lambda_{\alpha_{i,0}/\alpha_{j,0}} \circ A_j^{-1}$ and $c = q_i \oplus A_i(c_i) \oplus L\left[q_j \oplus A_j(c_j)\right]$, Eq. 3 holds, which shows the existence of a solution.

The other way round, assuming there is a linear mapping L and a constant c such that Eq. 3 holds, amounts to saying that $(A_i \circ \Lambda_{\alpha_{i,0}} \oplus L \circ A_j \circ \Lambda_{\alpha_{j,0}}) \circ T'_0$ is a constant mapping. Since $T'_0 = T_0 \circ P_0$ is one-to-one, and $(A_i \circ \Lambda_{\alpha_{i,0}} \oplus L \circ A_j \circ \Lambda_{\alpha_{j,0}})$ is a linear mapping, this constant must be `00`. Thus $L = A_i \circ \Lambda_{\alpha_{i,0}/\alpha_{j,0}} \circ A_j^{-1}$, which uniquely defines L. Then $\alpha_{i,0}\cdot y_i \oplus L \circ \alpha_{j,0}\cdot y_j$ is constant, and this constant uniquely defines c. □

Obviously, there are analogous statements where one varies the second, third, or fourth variable and keep the other one constant. Also note that given two functions y_i and y_j, there is a straightforward practical algorithm to get the

corresponding affine mapping (L, c) connecting their affine parts together. Indeed, considering the 64 entries of the matrix L as well as the 8 entries of the constant vector of c as unknowns over GF(2), and using our knowledge of the functions y_i and y_j by values, one can form a highly overdefined linear system of $2^8 \times 8$ equations involving the 72 unknowns and solve it with time complexity much lower than 2^{16}.

3.3 Recovering the Affine Parasites

We note that Prop. 1 of the previous section enables us to directly compute the linear parts of Q_1, Q_2, and Q_3 from the knowledge of Q_0's linear part. We will therefore focus on Q_0's determination. This section is organized in two steps. First, we show how to recover the linear part of Q_0 up to Λ_γ, for some non-zero γ in GF(2^8). Then we show how this information can be used to recover both γ and the constant part q_0 of Q_0.

About Q_0's Linear Part Let us recall that we decompose each affine transformation Q_i into its linear and constant parts: $Q_i(x) = A_i(x) + q_i$. Applying Prop. 1 with $i = 0$ and $j = 1$, we get $L_0 = A_0 \circ \Lambda_{\alpha_{0,0}/\alpha_{1,0}} \circ A_1^{-1}$. Then, using the variant of Prop. 1 with $i = 0$ and $j = 1$, but where one varies x_1 instead of x_0, we obtain $L_1 = A_0 \circ \Lambda_{\alpha_{0,1}/\alpha_{1,1}} \circ A_1^{-1}$. We are thus able to compute $L = L_0 \circ L_1^{-1}$, that is $L = A_0 \circ \Lambda_\beta \circ A_0^{-1}$ where $\beta = \alpha_{0,0}\alpha_{1,1}/\alpha_{0,1}\alpha_{1,0}$. Remembering that values α are standing for the `MixColumn` coefficients—i.e., taking their values in the set {'01', '02', '03'}—only 16 values for β remain possible, which are collected in the following set

$$B = \{\text{'02'}, \text{'d8'}, \text{'03'}, \text{'6f'}, \text{'04'}, \text{'bc'}, \text{'06'}, \text{'b7'}, \text{'05'}, \text{'25'}, \text{'4a'}, \text{'f8'}, \text{'7f'}, \text{'c8'}, \text{'64'}, \text{'5f'}\}\,.$$

(One checks that no element of B is contained in any subfield of GF(2^8).)

Thus, the new starting point is a matrix L, with the form $A_0 \circ \Lambda_\beta \circ A_0^{-1}$, and we want to retrieve both β and A_0. Given that β is chosen from B, computing the characteristic polynomial of L reduces the number of possibilities for β to at most 2; actually, either β is already determined, or $\beta \in \{b, b^2\} \subset B$. To ease the exposition, we assume that β is known, for instance by testing the two possibilities, and using Prop. 3 of the next section to determine the correct one.

Proposition 2. *Given an element β of* GF(2^8) *not in any subfields of* GF(2^8) *and its corresponding matrix $L = A_0 \circ \Lambda_\beta \circ A_0^{-1}$, we can compute with time complexity lower than 2^{16}, a matrix $\widetilde{A}_0$ such that there exists a unique non-zero constant γ in* GF(2^8)*, so that $\widetilde{A}_0 = A_0 \circ \Lambda_\gamma$.*

Proof. We seek for $\widetilde{A}_0$ such that $L \circ \widetilde{A}_0 = \widetilde{A}_0 \circ \Lambda_\beta$. Considering $\widetilde{A}_0$'s entries as unknowns, this equation gives 64 equations in the 64 unknowns. Some non-trivial solution can be computed in time complexity $64^\omega < 2^{16}$, which we hereafter denote by $\widetilde{A}_0$. Then, define $A = A_0^{-1} \circ \widetilde{A}_0$. The equation $L \circ \widetilde{A}_0 = \widetilde{A}_0 \circ \Lambda_\beta$ also reads $\Lambda_\beta \circ A = A \circ \Lambda_\beta$. The only GF(2)-affine mappings that commutes with the

multiplication by β, are the multiplications by a $\mathrm{GF}(2^8)$ element. (To see this, write $A(x) = \sum_{i=0}^{7} \gamma_i \cdot x^{2^i}$. The commutativity constraint is then expressed by $\sum_{i=0}^{7} \gamma_i \beta^{2^i} \cdot x^{2^i} = \sum_{i=0}^{7} \beta\gamma_i \cdot x^{2^i}$ for all $x \in \mathrm{GF}(2^8)$. Since β is not contained in any subfield of $\mathrm{GF}(2^8)$, this in turn implies $\gamma_i = \text{`00'}$ for all i but $i = 0$. Therefore, as announced, $A(x) = \gamma_0\, x$.) Thus, there exists a unique $\gamma \in \mathrm{GF}(2^8)$ such that $A = \Lambda_\gamma$, and remembering that $A = A_0^{-1} \circ \widetilde{A}_0$, we have computed $\widetilde{A}_0 = A_0 \circ \Lambda_\gamma$. □

Now we only have to recover γ of Prop. 2 in order to fully determine A_0, the linear part of Q_0. In the following paragraph we explain how to compute it, as well as the constant part q_0 of Q_0, that is to recover Q_0 entirely.

Recovering P_i up to the Key, and Q_0 Let us return to the function we originally studied, namely

$$y_0(x_0, x_1, x_2, x_3) = Q_0 \left(\bigoplus_{i=0}^{3} \alpha_{0,i} \cdot T_i \circ P_i(x_i) \right) . \tag{4}$$

Remember that T_i stands for the key addition, followed by the AES-128's S-box application, that is $T_i(z) = S(z \oplus k_i)$. Hence, the mapping $x \mapsto S^{-1} \circ T_i \circ P_i(x) = P_i(x) \oplus k_i$ is affine. Now, from Prop. 2 we get some matrix $\widetilde{A}_0 = A_0 \circ \Lambda_{1/\gamma}$. We have the following:

Proposition 3. *There exists unique pairs $(\delta_i, c_i)_{i=0,\ldots,3}$ of elements in $\mathrm{GF}(2^8)$, δ_i being non-zero, such that*

$$\begin{aligned}
\widetilde{P}_0 &: x \longmapsto (S^{-1} \circ \Lambda_{\delta_0} \circ \widetilde{A}_0^{-1}) \left(y_0(x, \text{`00'}, \text{`00'}, \text{`00'}) \oplus c_0 \right) , \\
\widetilde{P}_1 &: x \longmapsto (S^{-1} \circ \Lambda_{\delta_1} \circ \widetilde{A}_0^{-1}) \left(y_0(\text{`00'}, x, \text{`00'}, \text{`00'}) \oplus c_1 \right) , \\
\widetilde{P}_2 &: x \longmapsto (S^{-1} \circ \Lambda_{\delta_2} \circ \widetilde{A}_0^{-1}) \left(y_0(\text{`00'}, \text{`00'}, x, \text{`00'}) \oplus c_2 \right) , \\
\widetilde{P}_3 &: x \longmapsto (S^{-1} \circ \Lambda_{\delta_3} \circ \widetilde{A}_0^{-1}) \left(y_0(\text{`00'}, \text{`00'}, \text{`00'}, x) \oplus c_3 \right) ,
\end{aligned}$$

are affine mappings. Any pair (δ_i, c_i) can be computed with time complexity 2^{24}. Moreover, those mappings are exactly $\widetilde{P}_i = P_i(x) \oplus k_i$.

Proof. The proposition amounts to saying that $x \to S^{-1}(\delta \cdot S(x) \oplus c)$ is affine and non-constant. Since S represent the AES-128 S-box, and δ in non-zero, this is only possible if $(\delta, c) = (\text{`01'}, \text{`00'})$, hence the existence and uniqueness of (δ_i, c_i). (This is also very easy to verify by an exhaustive search, which we have done.)

Since c is '00', we have $c_0 = y_0(x, \text{`00'}, \text{`00'}, \text{`00'}) \oplus \alpha_{0,0} \cdot T_0(P_0(x))$, and since $\widetilde{A}_0 = A_0 \circ \Lambda_{1/\gamma}$, we get $\widetilde{P}_0(x) = S^{-1} \circ \Lambda_{\delta_0 \cdot \gamma \cdot \alpha_{0,0}} \circ S(P_0(x) \oplus k_0)$, where k_0 is a byte of the corresponding round key. As shown above, $\delta_0 \cdot \gamma \cdot \alpha_{0,0}$ must be '01', hence $\widetilde{P}_0(x) = P_0(x) \oplus k_0$. The proof goes the same for $\widetilde{P}_1$, $\widetilde{P}_2$, and $\widetilde{P}_3$.

For every possible values for the pairs (δ_i, c_i)—there are 2^{16} possible pairs—we test if the corresponding mapping is affine. The lookup table has to be evaluated 2^8 times, and then 8 systems of 9 unknowns over $\mathrm{GF}(2)$, or equivalently

one system of 72 unknowns which can be precomputed, has to be solved. Since the mapping evaluation through the lookup table dominates, the total time complexity is bounded by 2^{24}. □

Since $\delta_i^{-1} = \gamma \cdot \alpha_{0,i}$, and given the fact that two of those $\alpha_{0,i}$ are '01', another is '02' and the last one is '03', exactly two of the δ_i^{-1} are equal and share the common value γ. Therefore we know Λ_γ, and thus the matrix $A_0 = \widetilde{A}_0 \circ \Lambda_\gamma$, as well as the underlying MixColumn coefficients $\alpha_{0,i}$.

Also note that we recover at the same time the constant q_0 of the affine mapping Q_0. Indeed, let us define $c_4 = y_0(\text{'00'}, \text{'00'}, \text{'00'}, \text{'00'})$. Considering Eq. 4, it can also be written as

$$c_4 = \left(\bigoplus_{i=0}^{3} \alpha_{0,i} \cdot T_i \circ P_i(\text{'00'}) \right) \oplus q_0 \ .$$

Then, remembering that

$$\begin{aligned}
c_0 &= y_0(x, \text{'00'}, \text{'00'}, \text{'00'}) \oplus \alpha_{0,0} \cdot T_0(P_0(x)) \ , \\
c_1 &= y_0(\text{'00'}, x, \text{'00'}, \text{'00'}) \oplus \alpha_{0,1} \cdot T_1(P_1(x)) \ , \\
c_2 &= y_0(\text{'00'}, \text{'00'}, x, \text{'00'}) \oplus \alpha_{0,2} \cdot T_2(P_2(x)) \ , \\
c_3 &= y_0(\text{'00'}, \text{'00'}, \text{'00'}, x) \oplus \alpha_{0,3} \cdot T_3(P_3(x)) \ ,
\end{aligned}$$

which holds for every x and thus in particular for '00', we easily check that the constant part of Q_0 is given by $q_0 = c_0 \oplus c_1 \oplus c_2 \oplus c_3 \oplus c_4$, which achieves to fully recover the mapping Q_0.

3.4 Putting Everything Together

Let us now summarize the whole process of recovering the white box AES-128 implementation's original parasites. In Sec. 3.1 we have shown how to compute, for any round $r = 1, \ldots, 9$ and any index $j = 0, \ldots, 3$, with time complexity 2^{24}, the non-linear part of any parasitic mapping $Q_{i,j}^r$, $i = 0, \ldots, 3$—and thus at the same time, the non-linear part of its inverse parasitic mapping $P_{i,j}^{r+1}$—up to some affine application $x \mapsto A_i^r(x) \oplus q_i^r$. Section 3.2 showed how to recover A_1, A_2, and A_3 from the knowledge of A_0, with time complexity lower than $3 \cdot 2^{16}$. Finally, Sec. 3.3 explained how to recover the affine mapping $x \mapsto A_0^r(x) \oplus q_0^r$, for $r = 2, \ldots, 9$, with time complexity lower than 2^{16}. At the same time, Sec. 3.3 also retrieved the missing affine part of $P_{i,j}^r$ up to the key addition, which will allow us, as explained in the next section, to extract the key embedded in the AES-128 white box implementation.

Hence the time complexity to compute the parasites for a complete obfuscated AES-128 round, is bounded by $4 \cdot 4 \cdot 2^{24} = 2^{28}$.

3.5 Key Extraction

We now give the procedure for the key extraction. The white box implementation of AES-128 key embeds round keys produced by the AES-128 key derivation

algorithm. Thus the keys for two different rounds are related to each other. Using this property, one can obviously ease the recovery of the keys.

In a first step, we determine $Q^r_{i,j}$'s non-linear part for some round plus the entire parasites of two consecutive AES-128 obfuscated rounds. For instance, recover the parasitic mappings $Q^2_{i,j}$, as well as $\widetilde{P}^3_{i,j}$, $Q^3_{i,j}$, and $\widetilde{P}^4_{i,j}$, for $i = 0, \ldots, 3$ and $j = 0, \ldots, 3$ as described in Sec. 3.4. Then, since $P^{r+1}_{i,j} \circ Q^r_{i,j}$ must be the identity, we get the round key bytes as the composition of the affine mappings $\widetilde{P}$ and the affine part of Q which is denoted here by $\bar{Q}$, that is $k^3_{i,j} = \widetilde{P}^3_{i,j} \circ \bar{Q}^2_{i,j}$, and $k^4_{i,j} = \widetilde{P}^4_{i,j} \circ \bar{Q}^3_{i,j}$.

We now have the key bytes $k^3_{i,j}$ and $k^4_{i,j}$, however they are not necessarily in the right order. Still, the data flow exposed by the implementation, rules the way each round r key bytes relates to the next round $r+1$ key bytes. If we assume—according to Sec. 3.1 of [1]—that the round keys were generated using the key derivation algorithm of AES-128, the added constraint between the 16 bytes $k^3_{i,j}$ and the 16 bytes $k^4_{i,j}$ allows us to rearrange them the right way. Thus, having correctly recovered an AES-128 round key, we are able to derive the whole set of round keys.

Acknowledgements

The authors thank the anonymous referees for their valuable comments.

4 Conclusion

This paper explained how to extract, in a very efficient way, the whole secret key of a white box AES-128 implementation suggested in [1]. Some of our attack methods, for instance the technique of Sec. 3.1 used to recover the non linear parts of the encodings, are potentially applicable to other iterated blockciphers white box implementations using similar encoding and linear mixing techniques. However, parts of our attack take advantage from AES specificities. Therefore, no general conclusion can be drawn about the possibility to construct a strong white box AES implementation, or a strong white box implementations of other iterated blockciphers. Despite the general impossibility results concerning obfuscation [6], there is no evidence so far that strong white box implementation of blockciphers is unachievable; there is only some practical evidence that this is not an easy task. An interesting avenue for further research on obfuscation techniques might consist in developing a dedicated blockcipher, designed bottom-up with white box implementation in mind.

References

1. Chow, S., Eisen, P.A., Johnson, H., van Oorschot, P.C.: White-Box Cryptography and an AES Implementation. In Nyberg, K., Heys, H.M., eds.: Selected Areas in Cryptography – SAC 2002. Volume 2595 of Lecture Notes in Computer Science., Springer Verlag (2003) 250–270

2. Chow, S., Eisen, P.A., Johnson, H., van Oorschot, P.C.: A White-Box DES Implementation for DRM Applications. In Feigenbaum, J., ed.: Digital Rights Management Workshop – DRM 2002. Volume 2696 of Lecture Notes in Computer Science., Springer Verlag (2003) 1–15
3. Jacob, M., Boneh, D., Felten, E.W.: Attacking an Obfuscated Cipher by Injecting Faults. In Feigenbaum, J., ed.: Digital Rights Management – DRM 2002. Volume 2696 of Lecture Notes in Computer Science., Springer Verlag (2003) 16–31
4. Daemen, J., Rijmen, V.: The Design of Rijndael. Springer Verlag (2002)
5. National Institute of Standards and Technology: Advanced encryption standard. FIPS publication 197 (2001) `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.
6. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (Im)possibility of Obfuscating Programs. In Kilian, J., ed.: Advances in Cryptology – CRYPTO 2001. Volume 2139 of Lecture Notes in Computer Science., Springer Verlag (2001) 1–18
7. Biryukov, A., Preneel, B., Braeken, A., de Cannire, C.: A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms. In Biham, E., ed.: Advances in Cryptology – EUROCRYPT 2003. Volume 1267 of Lecture Notes in Computer Science., Springer Verlag (2003) 33–50

A Traceable Block Cipher

Olivier Billet and Henri Gilbert

France Télécom R&D
38-40, rue du Général Leclerc
92794 Issy les Moulineaux Cedex 9 - France
{olivier.billet,henri.gilbert}@francetelecom.com

Abstract. In this paper we propose a new symmetric block cipher with the following paradoxical traceability properties: it is computationally easy to derive many equivalent secret keys providing distinct descriptions of the same instance of the block cipher. But it is computationally difficult, given one or even up to k equivalent keys, to recover the so called meta-key from which they were derived, or to find any additional equivalent key, or more generally to forge any new untraceable description of the same instance of the block cipher. Therefore, if each legitimate user of a digital content distribution system based on encrypted information broadcast (e.g. scrambled pay TV, distribution over the Internet of multimedia content, etc.) is provided with one of the equivalent keys, he can use this personal key to decrypt the content. But it is conjectured infeasible for coalitions of up to k traitors to mix their legitimate personal keys into untraceable keys they might redistribute anonymously to pirate decoders. Thus, the proposed block cipher inherently provides an efficient traitor tracing scheme [4]. The new algorithm can be described as an iterative block cipher belonging to the class of multivariate schemes. It has advantages in terms of performance over existing traitor tracing schemes and furthermore, it allows to restrict overheads to one single block (*i.e.* typically 80 to 160 bits) per encrypted content payload. Its strength relies upon the difficulty of the "Isomorphism of Polynomials" problem [17], which has been extensively investigated over the past years. An initial security analysis is supplied.

Keywords: traitor tracing, block ciphers, Matsumoto-Imai, multivariate cryptology, symmetric cryptology, collusion resistance.

1 Introduction

One of the most employed digital content distribution methods consists in broadcasting encrypted information. Applications include pay TV systems, server-based services for the distribution of pre-encrypted music, videos, documents or programs over the Internet, distribution of digital media such as CDs or DVDs, and more generally, conditional access systems. In content distribution systems broadcasting encrypted information, each user is equipped with a "decryption box" which may be a smart card combined with an unscrambling device as in several existing pay TV systems, or even of software on a personal computer.

The decryption box of each legitimate user is provided with a decryption key, allowing him to recover the plaintext content from the broadcast information during some validity period or for a given subset of the content. The delivery and update of decryption keys may be performed using various key distribution methods and is generally subject to the payment of subscriptions, digital right management licenses, etc.

The following security problem arises in this setting: if any legitimate user manages to recover the decryption key contained in his decryption box or to duplicate the keyed decryption software, then he can redistribute it to illegitimate users, allowing them to get the plain content as the legitimate users, without having to pay any subscription, digital right management license, etc. This quite often represents a much more serious threat than the redistribution of the plaintext content, which is so far not considered very practical in contexts like pay-TV. The use of tamper resistant devices (e.g. smart cards) to store decryption keys and associated algorithm(s) obviously helps protecting these systems, but can hardly be considered a sufficient countermeasure to entirely prevent this kind of attacks. Over the past years, more and more sophisticated attacks against tamper resistant devices have emerged—e.g. side-channel attacks, see for instance [12]. Because attacking a single decryption box may lead to massive fraud, attackers can afford using sophisticated and expensive attacks, so that countermeasures proposed in other contexts will often be ineffective for encrypted content broadcast systems.

Traitor tracing provides a natural countermeasure to prevent the decryption key redistribution threat described above. The concept of traitor tracing scheme was first introduced by B. Chor, A. Fiat and M. Naor in the seminal paper [4] and we use as far as possible the same terminology to describe the proposed scheme. In traitor tracing schemes, each legitimate user is provided with a unique personal decryption key which unambiguously identifies him, while enabling him to decrypt the broadcast information. The system must accommodate a large number N of users and it must be infeasible for any coalition of up to k legitimate users to mix their personal keys into a new untraceable description of the decryption key. Most of the traitor tracing schemes proposed so far, e.g. those described in [4], [14] and [18] are combinatorial in nature. Each legitimate user is provided with several base keys, which together form his personal key and the broadcast information contains large overheads of encrypted values under some of the base keys, allowing legitimate users to recover a content decryption key. A non-combinatorial alternative, namely a public key encryption scheme in which there is one public encryption key but many private decryption keys, was proposed by D. Boneh and M. Franklin in [3]. It has the advantage to avoid large overheads and to have very small decryption keys. However, the performance of this scheme is extremely sensitive to the maximum number k of tolerated colluding traitors, since the data expansion factor of the public key encryption is proportional to k.

The approach developed in this paper is non combinatorial in nature and has stronger connection with the one developed in [3] than with combinatorial

schemes, up to the essential difference that we construct an untraceable symmetric cipher rather than an untraceable asymmetric cipher. The proposed cipher has the paradoxical property that many equivalent secret keys (used for decryption purposes) can be generated, while it is conjectured to be computationally impossible, given at most k equivalent secret keys, either to forge another untraceable equivalent secret key or to reconstruct the "meta key" from which the original equivalent secret keys were derived. More precisely, the knowledge of the meta key allows to efficiently determine at least one of the equivalent secret keys used to forge the new description.

The proposed construction can be described as an iterative block cipher. Its strength relies upon the intractability of the "Isomorphism of Polynomials," a problem which has been extensively investigated over the past years [2, 11, 17] and which conjectured intractability has not been directly affected by recent advances in the cryptanalysis of multivariate schemes like HFE [9, 10]. One of the advantages of the proposed scheme is to avoid generated overhead compared to the combinatorial approach taken in [3] where the data expansion is proportional to k. Another advantage is the intrinsic structure which is rather close to the one of usual block ciphers, so that the performance of the cipher in encryption/decryption modes is better than for existing traitors tracing schemes. Also the proposed scheme is much less sensitive to the maximum number k of traitors tolerated in a coalition, or to the maximum number of users N in the system. On the negative side, one should mention that the tracing procedures described in this paper require the knowledge of the description of the decryption function owned by a pirate. Thus no "black box" tracing procedure limiting interaction with the pirate decoder to "oracle queries" is provided. Another limitation of the proposed algorithm is that as usual in symmetric cryptography, no provable reduction to the difficulty of a well studied mathematical problem (e.g. the isomorphism of polynomial problem) could be found. Thus, the security analysis we supply can only achieve the next desirable goal, *i.e.* investigate various attack strategies and make sure that identified attacks are thwarted. Because of the higher requirements on a traceable cipher, risks are obviously much higher than for usual symmetric ciphers.

This paper is organized as follows. In Section 2, we describe the requirements on a symmetric cipher with an associated non-combinatorial traitor tracing scheme. In Section 3, we describe the proposed iterative block cipher construction and the associated traitor tracing scheme. Section 4 provides an initial security analysis. Section 5 addresses performance issues and provides an example instance of the proposed algorithm with explicit practical parameter values, in order to stimulate improved cryptanalysis. Section 6 concludes the paper.

2 Traceable Block Ciphers: Requirements and Operation

Let us denote by $F_{\mathcal{K}}$, $\mathcal{K} \in \mathbf{K}$ a symmetric block cipher of block size l, *i.e.* a key-dependent function from the set $\{0,1\}^l$ of l-bit input values to itself. As will be seen in the sequel, it is not required that $F_{\mathcal{K}}$ be easy to invert. It is not even

an absolute requirement that the function $F_{\mathcal{K}}$ be one to one, although the block ciphers proposed in this paper are actually one to one and can be inverted: in practice they are operated in the forward direction alone, except in some traitor tracing procedures.

A traitor tracing scheme for N users associated with a traceable symmetric block cipher $F_{\mathcal{K}}$ consists of the following components:

- **A user initialization scheme** deriving users' secret keys $(\mathcal{K}_j)_{j=1,\ldots,N}$ from a meta key $\mathcal{K} \in \mathbf{K}$. All user secret keys $\mathcal{K}_j$ must be distinct (though equivalent) descriptions $F_{\mathcal{K}_j}$ of the meta function $F_{\mathcal{K}}$. Each description $F_{\mathcal{K}_j}$ must allow to efficiently compute $F_{\mathcal{K}}$ in the forward direction.
- **Encryption and decryption processes**, respectively used by the operator of the broadcast distribution system to encrypt some digital content using $F_{\mathcal{K}}$, and by the legitimate user j to decrypt this content using his recovery key $\mathcal{K}_j$ through the associated description $F_{\mathcal{K}_j}$ of $F_{\mathcal{K}}$. As explained in [4], the structure of the broadcast information typically consists of pairs $(\mathrm{EB}_i, \mathrm{CB}_i)$ of an overhead information named "enabling block" and an encrypted content block named "cipher block." The enabling block is used to generate a symmetric key, hereafter called "control word," to decrypt the cipher block via an additional symmetric scheme S, like for instance AES or one-time pad. As said before, $F_{\mathcal{K}}$ needs not to be invertible: it is used in the forward direction in both the encryption and decryption processes.
- **A tracing procedure** allowing the owner of the meta key, when provided with any pirate description of the decryption function forged by any coalition of up to k traitors, to trace at least one traitor of the coalition.

In this setting, the meta key's holder creates cipher blocks CB_i from blocks of plain text content B_i using an additional symmetric scheme S and enabling blocks EB_i (produced for instance by a pseudo-random generator) via the formula $\mathrm{CB}_i := S_{\mathrm{CW}_i}(\mathrm{B}_i)$, where the control words CW_i are derived from the enabling blocks using the traceable block cipher $\mathrm{CW}_i := F_{\mathcal{K}}(\mathrm{EB}_i)$.

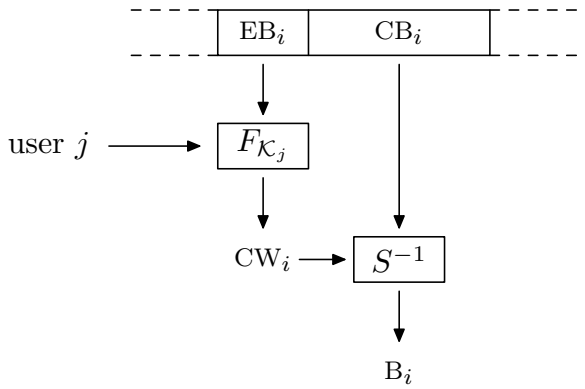


Fig. 1. Scheme's Architecture

The operations performed by legitimate users to decrypt these content blocks are summarized in Fig. 1. User j first derives the control word CW_i from the enabling block EB_i via his description $F_{\mathcal{K}_j}$ of the meta function: $\mathrm{CW}_i = F_{\mathcal{K}_j}(\mathrm{EB}_i)$. Then he uses the control word CW_i to decrypt the cipher block CB_i via the additional symmetric scheme S and recovers associated block(s) of plain content $\mathrm{B}_i = S^{-1}_{\mathrm{CW}_i}(\mathrm{CB}_i)$. For instance, $\mathrm{B}_i = \mathrm{CB}_i \oplus \mathrm{CW}_i$ when S is the one time pad algorithm, and $\mathrm{B}_i = \mathrm{AES}^{-1}_{\mathrm{CW}_i}(\mathrm{CB}_i)$ when S is the AES. In this context, control words must be frequently generated to prevent attacks by redistribution of these control words to pirate decryption boxes from being much easier than the redistribution of plaintext. This is difficult to achieve with existing combinatorial traitor tracing schemes due to the large data expansion incurred by such schemes. Another consequence is that the throughput (bit/s) of the $F_{\mathcal{K}}$ block cipher must be as close as possible to the throughput of classical block ciphers such as AES and much larger than the one of asymmetric ciphers such as RSA. An additional requirement for systems where $\mathcal{K}$ needs to be updated frequently, e.g. to manage dynamic modifications of lists of subscribers, is that each description $F_{\mathcal{K}_j}$ be reasonably short for the distribution via any symmetric encryption or key distribution algorithm to be practical.

In order for the content distribution system to resist attacks against the decryption scheme, the descriptions $F_{\mathcal{K}_j}$ must satisfy the usual security requirements of a block cipher. This implies that given any set of $F_{\mathcal{K}_j}$ input/output pairs with known, chosen or even adaptively chosen input values an adversary could obtain, it must be computationally infeasible for this adversary to predict any additional $F_{\mathcal{K}_j}$ input/output pair with a non negligible success probability. In particular input/outputs pairs must not reveal $\mathcal{K}_j$ or any other equivalent description of $F_{\mathcal{K}}$.

The last and most demanding requirement is the existence of an efficient traitor tracing procedure for the owner of the meta key $\mathcal{K}$. Our definition of a traitor tracing scheme follows the one proposed in the seminal paper [4]. We do not require the traitor tracing scheme to be black box (*i.e.* to be operable using say only inputs EB_i and outputs CW_i of the key distribution function). We restrict ourselves to traitor tracing scenarios where an authority is able to access the description of the description of $F_{\mathcal{K}}$ contained in the pirate decryption box. Note that it does not seem unrealistic to assume that decryption boxes of pirate users can be tampered by an authority, taking into account the fact that traitor tracing is only needed if the decryption boxes of legitimate users can be tampered. Traitor tracing requirements can be informally stated as follows. Attacks by any coalition of up to k traitors should be traceable, that is k traitors able to access their individual descriptions $F_{\mathcal{K}_j}$ should not be computationally able to forge any additional description F' from their k equivalent descriptions $F_{\mathcal{K}_j}$ without revealing at least one of their $\mathcal{K}_j$—and thus the identity j of one of the traitors. We further require that the probability for the tracing procedure applied to any k-traitors coalition to either output no suspected traitor (non detection) or to output the identity j of an innocent user (false alarms) be negligible.

3 Description of the Traceable Scheme

Among the requirements identified in the former Section, the most demanding one is not the existence of many equivalent descriptions of the symmetric function $F_{\mathcal{K}}$—this is frequent in symmetric cryptography, see for instance [1]—but the property that the provision to a user of one of these numerous representations $F_{\mathcal{K}_j}$ should not disclose information allowing him to construct any other representation of $F_{\mathcal{K}}$ unrelated to $\mathcal{K}_j$. In other words, the meta key $\mathcal{K}$ must act as a kind of trapdoor allowing to perform other operations than those allowed by the descriptions $F_{\mathcal{K}_j}$ of $F_{\mathcal{K}}$. Thus, even in the symmetric setting considered in this paper, public key cryptography properties are required and generic block ciphers will not be usable like in the case of combinatorial traitor tracing schemes. However we would like to keep performance advantages of symmetric cryptography since generation of control words at high rate is necessary for the security of the system.

Multivariate cryptography appears to be a natural candidate to meet these requirements. As a matter of fact, features of this recently developed family of algorithms are to many extents intermediate between those of public key algorithms (e.g. trapdoors) and those of secret key algorithms. Many of them can be described as iterative ciphers resulting of the composition of several rounds, and their complexity is substantially lower than the one of usual public key ciphers and not much higher than the one of usual block ciphers. Typical examples of multivariate algorithms are C^* proposed by T. Matsumoto and H. Imai in [13], SFLASHv2 (one of the Nessie finalists [19]), and HFE [16]. All the schemes mentioned above rely on the intractability of the so-called "Isomorphism of Polynomials" problem for the *secret key recovery*. See [7] for more information about known attacks against this problem. The C^* scheme was attacked by Patarin in [15] and Dobbertin independently, but these attacks do not allow to recover the secret key and thus to break the underlying IP problem. An attack allowing to solve the IP problem underlying some instances of HFE, using so-called relinearization techniques was published by Kipnis and Shamir in 1999 [11], and appears to be also applicable to the IP problem underlying some instances of the basic (quadratic) version of C^*. More recently, enhanced decryption or signature forgery attacks against HFE and more generally various multivariate cryptosystems have been proposed [8, 6, 9, 10]. But none of these recent attacks allows to recover the secret key and to break the underlying IP problem. Thus in summary, as far as we know, the best known attacks against the IP problem underlying multivariate schemes are those described in [7, 11].

3.1 Building Blocks

Let us briefly recall the basic quadratic C^* from which the building block of our scheme is directly derived by generalizing it to monomials of higher degree. It involves the following elements:

– A finite field $\mathbb{K} = \mathbb{F}_q$ of size q.

– An extension $\mathbb{L}$ over $\mathbb{K}$ of degree n, with a defining primitive polynomial $P(X)$ of degree n such that $\mathbb{L} = \mathbb{K}[X]/(P(X))$. We will represent elements of $\mathbb{L}$ as n-tuples $(a_0, \ldots, a_n)$ of $\mathbb{K}$ through the usual identification function $\varphi : (a_0, \ldots, a_n) \mapsto \sum_{i=0}^{n} a_i X^i \pmod{P(X)}$.
– A private key made of two linear one to one mappings s and t from $\mathbb{K}^n$ to itself and an integer θ such that $q^\theta + 1$ be prime to $q^n - 1$.
– A public key $G = t \circ \varphi^{-1} \circ E_\theta \circ \varphi \circ s$, published as a system of n multivariate polynomials in n variables, where E_θ is a monomial function defined to be $\mathbb{L} \to \mathbb{L}$, $a \mapsto a^{1+q^\theta}$. Assuming the trapdoor (s, t) unknown, function G was believed to be one-way, but J. Patarin showed in [15] that it can be computationally inverted. However, one-wayness is not needed in our scheme.

The actual building blocks of our construction are higher degree variants of C^* obtained by considering a more generic—but still monomial—function E, namely $E_\Theta : \mathbb{L} \to \mathbb{L}$, $a \mapsto a^{1+q^{\theta_1}+\ldots+q^{\theta_{d-1}}}$ where d is a fixed integer and Θ is a $(d-1)$-tuple $(\theta_1, \ldots, \theta_{d-1})$ such that $q^n - 1$ be prime to $1 + q^{\theta_1} + \ldots + q^{\theta_{d-1}}$, hereafter called the degree of the building block G. Indeed, G can be described as a system of n multivariate polynomial equations as suggested in Fig. 2, and the polynomials P_i involved have total degree d. For instance, in the special case where $d = 3$, G can be described as $(i = 1, \ldots, n)$:

$$y_i = \sum_{0 \le j,k,l \le n-1} \alpha_{i,j,k,l} x_j x_k x_l + \sum_{0 \le j,k \le n-1} \beta_{i,j,k} x_j x_k + \sum_{0 \le j \le n-1} \gamma_{i,j} x_j \ . \qquad (1)$$

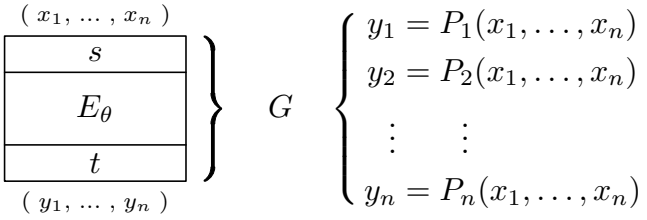


Fig. 2. An extended C^* building block.

The basic idea underlying the proposed traitor tracing scheme is to use several of those extended C^* instances as building blocks for our construction and to take opportunity of the commutativity of the various monomial functions E_θ involved—that is $E_{\theta_1} \circ E_{\theta_2} = E_{\theta_2} \circ E_{\theta_1}$ for all θ_1, θ_2.

3.2 Meta Key, Users' Keys

Let us keep the notation of the previous Section. Moreover, let r be the number of building blocks. The meta secret key $\mathcal{K}$ is defined as the set of two one to one linear mappings s and t from $\mathbb{K}^n$ to itself, and a collection of r $(d-1)$-tuples Θ_i such that all the values $1 + q^{\theta_{1,i}} + \ldots + q^{\theta_{d-1,i}}$ for $i = 1, \ldots, r$ be distinct. Then the function $F_\mathcal{K}$ is defined as $F_\mathcal{K} = s \circ E_{\Theta_r} \circ \cdots \circ E_{\Theta_2} \circ E_{\Theta_1} \circ t$.

Now assign to each user j a private key $\mathcal{K}_j$ generated after the meta key $\mathcal{K}$ using a set of $r-1$ linear one to one mappings $L_{1,j}, \ldots, L_{r-1,j}$ from $\mathbb{K}^n$ to itself, and a permutation σ_j of the set $\{1, \ldots, n\}$. The user gets his key $\mathcal{K}_j$ as a list of functions $G_{1,j}, \ldots, G_{r,j}$, which are provided as systems of n multivariate equations of homogeneous degree as described in Figs. 2 and 3.

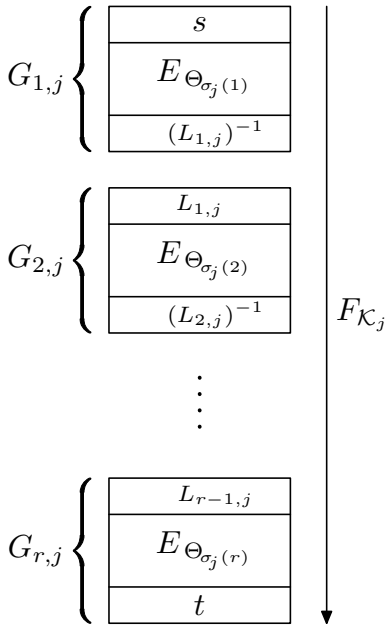


Fig. 3. Description $F_{\mathcal{K}_j} = G_{r,j} \circ \cdots \circ G_{2,j} \circ G_{1,j}$

A user initialization scheme needed to derive a user's key from the meta key $\mathcal{K}$ follows. From any input j one creates the permutation σ_j and the $r-1$ one to one mappings $L_{i,j}$ by any pseudo- random generation mechanism or by any diversification algorithm.

We can now check that the users' functions $F_{\mathcal{K}_j}$ are distinct but equivalent descriptions of the meta function $F_{\mathcal{K}}$. Indeed, for each user j, the one to one mappings at the end of $G_{k,j}$ and at the beginning of $G_{k+1,j}$ cancel out, and since the functions E_Θ are commuting, the effect of the permutation is annihilated.

3.3 Encryption and Decryption

In order to encrypt a digital content, the station may broadcast enabling block and cipher block pairs $(\mathrm{EB}_i, \mathrm{CB}_i)$ produced with the help of any additional symmetric algorithm S_{CW} where the symmetric key is the control word generated as $\mathrm{CW} := F_{\mathcal{K}}(\mathrm{EB}_i)$. Thus, the construction is given by $\mathrm{CB}_i := S_{F_{\mathcal{K}}(\mathrm{EB}_i)}(\mathrm{B}_i)$, where B_i denotes the content block. Now any user j can recover the content block by following a similar procedure, that is by computing $\mathrm{B}_i := S^{-1}_{F_{\mathcal{K}_j}(\mathrm{EB}_i)}(\mathrm{CB}_i)$.

3.4 Traitor Tracing Procedure

The procedure to identify traitors relies upon the two following claims which are substantiated in the security analysis given in the next section.

Claim 1. When the leakage originates from a single traitor l, the analysis of the description F' constructed by the traitor based on his description $F_{\mathcal{K}_l}$ allows the authority to decompose F' in r components G'_1 to G'_r such that $F' = G'_r \circ \cdots \circ G'_1$. Moreover, each G'_i can be split as the composition of the functions G_i of the traitor and other "parasitic" functions which may differ from the identity function. Thus, the analysis reveals the order of composition of the functions G_i which in turn reveals the identity of the traitor through the knowledge of σ_l.

This first claim allows an authority provided with the meta key $\mathcal{K}$ to efficiently derive the permutation σ_l associated to the description $F_{\mathcal{K}_l}$ of the traitor from the leaked function F', and thus to recover the identity l of the traitor.

Claim 2. When the leakage originates from a coalition of at most k traitors, the analysis of the description F' constructed by the k colluding traitors allows to decompose F' in r components G'_1 to G'_r such that the middle $r - 2\rho$ values come from "parasitized" functions G_i of a single traitor, for a well chosen ρ.

This second claim allows, by properly choosing the parameters of the system, specially ρ which exact definition is to be given in the next Section, to recover the identity of one of the traitors—say j—by deriving the values of the permutation σ_j on the set of integers $[\rho, r-\rho]$ from the values of the functions $G_{\rho,j}$ to $G_{r-\rho,j}$ alone. To achieve this goal, we must ensure that the middle part of the pirate description F' originates from the middle parts ρ to $r - \rho$ of one single traitor, while mixing traitors' descriptions in the ranges $[1, \rho]$ and $[r - \rho, r]$ can still be tolerated.

4 Security Discussion

4.1 The IP Problem

The security of the proposed traceable iterated symmetric cipher relies to a large extent upon the security of special instances of the "Isomorphism of Polynomials" problem—hereafter called IP—namely the problem of *finding the hidden monomial* of the extended Matsumoto-Imai C^* scheme described in Section 3.1.

The IP problem with two secrets—see also [7, 17]—consists in finding a pair (s, t) of one to one linear mappings between two sets A and B of multivariate polynomial equations of total degree d over a finite field $\mathbb{K}$. Denoting by $x = (x_1, \ldots, x_n)$ an element of $\mathbb{K}^n$, we can write $y = A(x)$ as a system of polynomial equations:

$$\begin{cases} y_1 = P_1(x_1, \ldots, x_n) \\ y_2 = P_2(x_1, \ldots, x_n) \\ \vdots \quad \vdots \\ y_n = P_n(x_1, \ldots, x_n) \end{cases},$$

and similarly for B. In this setting the IP problem consists in finding a pair of one to one linear mappings s and t such that:

$$B\big(s(x)\big) = t\big(A(x)\big). \tag{2}$$

This problem is assumed to be difficult and it has been shown to be at least as hard as the "Graph Isomorphism" problem. Even for very special instances complexity remains high [2, 6]. Note also that an efficient solution to the IP problem would lead to an efficient attack on SFLASHv2 [19] that has been selected by the European Nessie project.

4.2 Resisting Attacks Against the Decryption Scheme

As explained in Section 2, the descriptions $F_{\mathcal{K}_j}$ of any user j must satisfy the usual security requirements of block ciphers. In particular, given any realistic number of input/output pairs of $F_{\mathcal{K}_j}$ corresponding to chosen or adaptively chosen input values, it must be computationally infeasible to infer any additional output value. Based on an investigation of the most natural attack strategies, we conjecture that this property is satisfied provided that:

1. Parameters q and n be chosen so that even if the monomial functions E_{Θ_1}, E_{Θ_2}, ..., E_{Θ_n} can be guessed, solving the IP problem which consists of guessing s and t given a sufficient large number of input/output pairs of $F_{\mathcal{K}_j}$ be intractable. Based on the results in [7, 2] we expect this condition to be satisfied provided that the complexity q^n of the best know attack be large enough, say at least 2^{80}. Since an enhanced attack of complexity $q^{n/2}$ is reported in the quadratic case in [7], an even more conservative choice would be to consider $q^n > 2^{160}$ in order to prevent a generalization of this attack to other instances of IP.
2. The value q^D, where D is the degree of the system of polynomial equations in n variables representing any $F_{\mathcal{K}_j}$ be large enough, say at least 2^{80}, to prevent attacks based on higher order derivation. Indeed, this would allow an attacker to predict one more output given an affine set of q^{D+1} input values and and all but one of their corresponding outputs. D is about nq when r is large enough and q is the size of the finite field $\mathbb{K}$;
3. The number of monomials of the system of n polynomial equations in n variables representing any $F_{\mathcal{K}_j}$, which is usually close to $n\binom{n+D-1}{D}$, be large enough to prevent an attacker from recovering the coefficients of this system using linear algebra and a sufficient number of input/output pairs of $F_{\mathcal{K}_j}$.

4.3 Tracing Single Traitor's Pirate Description

We anticipate that in trying to produce an untraceable version of his description $F_{\mathcal{K}_j}$, a traitor j would adopt one of the following strategies:

1. Try to find one of the $r+1$ one to one linear mappings s, $L_{1,j}$, $L_{2,j}$, ..., $L_{r-1,j}$ and t, hidden to the attacker j. If an attacker j could recover one

of these $r+1$ linear mappings, say $L_{l,j}$, this would obviously allow him to incrementally recover all the $L_{i,j}$ for $i < l$, and all the $L_{i,j}$ for $i > l$, using the information provided by the mappings $G_{1,j}$ to $G_{r,j}$, and thus to recover the value of $\mathcal{K}_j$ and to easily produce variants of his description $F_{\mathcal{K}_j}$ in an untraceable manner. Conversely, we conjecture this to be as hard as solving the IP problem of at least one of the $G_{i,j}$. The complexity of the best attacks reported in [7] are $O(q^n)$ in case $d > 2$ and $O(q^{n/2})$ in case $d = 2$.

2. Try to directly use the functions $G_{\cdot,j}$ without analyzing them, by modifying them so as to produce a concealed variant of the original description by composing the basic blocks $G_{\cdot,j}$ in the same order, but with "parasitic" functions whose effects eventually cancel out. That is the traitor tries to produce a sequence $(G'_{i,j})_{i\in[1,w]}$ with *two types* of blocks G': those which can be written as $\varphi_i \circ G_{i,j} \circ \psi_{i+1}$ and those that do not rely on the available $G_{i,j}$ blocks and are denoted by Π_i. These data must be such that the effects of adding/composing the φ, Π and ψ mappings to the original blocks $G_{i,j}$ eventually cancel out, that is so that $F_{\mathcal{K}_j} = G'_{w,j} \circ \cdots \circ G'_{1,j}$. (Please note that w can be greater than r because of the *second type* of blocks.) Also note that φ_i, ψ_i and Π_i have to be simple enough—for instance a reasonable number of monomials and a limited total degree—so that they could be easily constructed and efficiently computed.
3. Try to compose several blocks $G_{i,j}$ of his description. This attack is impossible as soon as the number of monomial in such composition is impractical. Since composition must be formally computed, $\binom{n+d-1}{d}$ terms must be formally put to the power of d which is quickly intractable. As will be seen in the sequel, composition of a small number of blocks $G_{i,j}$, say 2 of them, do not substantially complexify the tracing procedure. Therefore, only the composition of more than 3 blocks must be prevented.
4. Use a combination of any of the above strategies.

To trace traitor j from a pirate description $G'_1, \ldots, G'_w$, the authority proceeds as follows. First, note that G'_1 is necessarily of the form $\psi_1 \circ L_{1,j} \circ E_{\Theta_{\sigma_j(1)}} \circ s$, that is of the *first type*. The authority thus searches for $\sigma_j(1)$ by using its knowledge of s^{-1}, and all the $E^{-1}_{\Theta_i}$: it computes $G'_1 \circ s^{-1} \circ E^{-1}_{\Theta_i}$ for each i, and guesses the right value i by testing the "simplicity" of the resulting function by means of chosen input/output pairs. The simplicity is evaluated by estimating the degree and the number of monomials. In case of a correct guess, the function has a low degree and a predetermined number of monomials whereas in case of a bad guess the function has terms of high degree. Having guessed the value $\sigma_j(1)$, we denote it by $\alpha(1)$.

The authority then has to get rid of terms of *second type* Π_i, until another term of *first type* is found. This is done again by evaluating the simplicity of the successive compositions:

$$\begin{aligned} G'_2 \circ G'_1 \circ s^{-1} \circ E^{-1}_{\Theta_{\alpha(1)}} \circ E^{-1}_{\Theta_i} \ , \\ G'_3 \circ G'_2 \circ G'_1 \circ s^{-1} \circ E^{-1}_{\Theta_{\alpha(1)}} \circ E^{-1}_{\Theta_i} \ , \\ \vdots \end{aligned}$$

each time for all i until a simple composed function is found. The authority then finds the value $\sigma_j(2)$ and denotes it by $\alpha(2)$. The process goes on iteratively and eventually gives the permutation σ_j allowing the authority to trace traitor j.

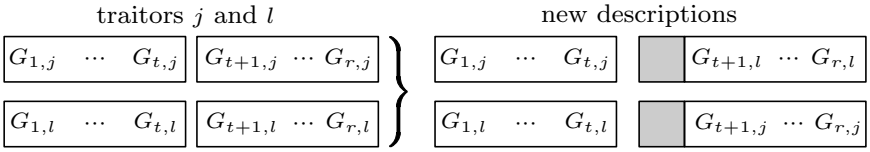
While choosing the parameters of the system, we will make it hard for an attacker to formally compose two extended C^* blocks and totally intractable to compose three of them. The composition of two consecutive blocks can be easily thwarted since the above guessing procedure remains valid when replacing $E_{\Theta_i}^{-1}$ by $E_{\Theta_i}^{-1} \circ E_{\Theta_j}^{-1}$ varying both i and j at the same time, thus allowing to trace such compositions of two blocks as well.

4.4 Tracing k Traitors' Pirate Descriptions

The best collusion strategy we identified for a coalition of at most k traitors provided with distinct descriptions $F_{\mathcal{K}_j} = G_{r,j} \circ \cdots \circ G_{1,j}$ associated with the same meta description $F_{\mathcal{K}}$ is the following one.

The basic idea is that the traitors may take advantage of the fact that the initial mapping s and the final mapping t are identical for every user. This could allow them to detect a partial collision between their respective hidden permutation σ. Let us take the example of two traitors j and l searching for such a collision. They know their first blocks begin with the same mapping s, and if their first functions $E_{\sigma_j(1)}$ and $E_{\sigma_l(1)}$ were equal, then blocks $G_{1,j}$ and $G_{1,l}$ would be equal up to a one to one linear mapping, namely $L_{1,j}^{-1} \circ L_{1,l}$. Otherwise it would not be a one to one linear mapping. This is easy to test and provides a way for a pair of traitors to guess if their permutations take the same values on 1, *i.e.* if $\sigma_j(1) = \sigma_l(1)$.

Now, whether they succeed or not in the last step, the pair of traitors go further in the process by checking whether $G_{2,j} \circ G_{1,j}$ and $G_{2,l} \circ G_{1,l}$ are equal up to another hidden one to one linear mapping. (Remember that the commutativity of the functions E_{Θ_i} makes this possible.) In case of success, this would allow them to deduce that the images of the unordered set $\{1, 2\}$ under both permutations are equal: $\sigma_j\big(\{1,2\}\big) = \sigma_l\big(\{1,2\}\big)$, and provide them with the value of $L_{2,j}^{-1} \circ L_{2,l}$. By iterating the process, the pair of traitors may identify any collision of their respective permutations on the set of integers $[1, t]$ for any $t \in [1, r]$, hereafter called a t-collision. Moreover, any detected t-collision provides a way to forge two new pirate descriptions by exchanging the first t components of their respective descriptions of the meta function $F_{\mathcal{K}}$, as shown in Fig. 4.



$$\Big(G_{1,j}, G_{2,j}, \ldots, (L_{t,j}^{-1} \circ L_{t,l})^{-1} \circ G_{t,j}, G_{t+1,l}, \ldots, G_{r,l}\Big) \ ,$$
$$\Big(G_{1,l}, G_{2,l}, \ldots, (L_{t,l}^{-1} \circ L_{t,j})^{-1} \circ G_{t,l}, G_{t+1,j}, \ldots, G_{r,j}\Big) \ .$$

Fig. 4.

Note that the traitors can search for all collisions. That is, when no u-collision was found for $u < t$, it still remains possible for them to find a t-collision, *when such a collision exists*. Of course, this scenario can be replayed with other traitor pairs, or even with the newly forged descriptions, leading to a possible great amount of untraceable pirate keys.

To avoid this situation, one encodes the identity of any user i in the values taken by the permutation σ_i on the middle interval $[\rho, r-\rho]$ of the original one $[1, r]$, for some well chosen $\rho < r/2$ so that the probability of *any* t-collision for $t \in [\rho, r-\rho]$ is arbitrarily small.

Obviously, attacks involving a single traitor can also be used by coalition of traitors in addition to the specific techniques discussed in this Section, but those can be handled the same way.

4.5 Non-Detection and False Alarms.

Let us derive the requirements the attack scenario of the previous Section puts on parameter ρ. First, for any traitors' pair, the probability that a t-collision holds is $1/\binom{r}{t}$. Thus the probability that a t-collision for a coalition of up to k traitors occurs for $t \in [\rho, r-\rho]$ is at most

$$P_k = \frac{k(k-1)}{2} \sum_{t=\rho}^{r-\rho} \frac{1}{\binom{r}{t}} \ .$$

At the same time, permutations of users must be distinguishable from their values in the interval $[\rho, r-\rho]$. This implies that the number of distinct identities available for the system will be at most $M = r!/(2\rho)!$.

Now if the scheme needs to handle at most N users where $N < M$, and assuming a coalition of up to k traitors, the probability of non-detection (the authority detects a collusion, but no matching identity is found) is given by $P_{k,\mathrm{ND}} = (1 - N/M)\, P_k$ while the probability of false alarm (a wrong identity is pointed out) is given by $P_{k,\mathrm{FA}} = N/M\, P_k$. This comes from the fact that there are $(M - N)$ permutations that do not correspond to any valid identity.

5 Practical Example

We provide realistic example parameters such that the scheme accommodates $N = 10^6$ users. The field of operation $\mathbb{K}$ is taken to be $\mathrm{GF}(2^{16})$ so that $m = 16$ and $q = 2^{16}$. Moreover, we chose $n = 5$ and the degree of the monomials in an extended C^* block to be $d = 4$. There is a total of 32 distinct $(d-1)$-tuples Θ such that $1 + q^{\theta_1} + \ldots + q^{\theta_{d-1}}$ is prime to $q^n - 1$.

Letting $r = 32$ and $\rho = 13$ makes the probability of false alarms smaller than $2\,10^{-10}$ for any coalition of up to $k = 10$ traitors, smaller than $2.2\,10^{-8}$ for $k = 100$ traitors and smaller than $2.3\,10^{-6}$ for $k = 1000$. Probability of non-detection is smaller than $1.2\,10^{-7}$ when $k = 10$, smaller than $1.5\,10^{-3}$ when $k = 1000$. Other security requirements are met since $q^n = 2^{80}$ and furthermore

the number of monomials in a building block of $F_{\mathcal{K}}$ is 350, so that in any formal composition of three of them the number of monomials is already more than $4\,10^6$, and in any formal composition of four blocks it is about 10^9.

With this choice of parameters, the total size of any description equivalent to $F_{\mathcal{K}}$ is 21,8 KB. Speed of encryption is essentially determined by the number of multiplications in $F_{\mathcal{K}_j}$ to be performed and can roughly be estimated as follows: the 70 terms $x_1^{\nu_1} \cdots x_5^{\nu_5}$ of total degree four can be computed once for each block and then multiplied by the appropriate leading coefficients of the polynomials describing each output variable of a block. So one can compute the 70 homogeneous terms of degree 4 in 85 multiplications in $\mathbb{K}$ and eventually compute $y_1, \ldots, y_5$ in at most $5 \cdot 70$ multiplications in $\mathbb{K}$. Since there are 32 blocks, that makes a total of about 15000 multiplications to process any $F_{\mathcal{K}_j}$ on the 80 bit input. Additionally, the size of the overhead in this example is obviously 80 bits.

We propose another realistic set of parameters, hopefully more conservative, for applications where storage and speed of encryption are less critical concerns. The scheme handles up to $N = 10^6$ users. The field of operation is taken to be $\mathrm{GF}(2^9)$, while the number of variables is set to $n = 19$ and the degree of the monomials is set to $d = 3$. There is a total of 190 distinct $(d-1)$-tuples Θ such that $1 + q^{\theta_1} + \ldots + q^{\theta_{d-1}}$ is prime to $q^n - 1$. Choosing $r = 33$ and $\rho = 10$ makes the probability of false alarms smaller than $1.4\,10^{-19}$ for any coalition of up to $k = 10$ traitors, smaller than $1.52\,10^{-15}$ for any coalition of up to $k = 1000$ traitors, and the probability of non-detection smaller than $5\,10^{-7}$ for any coalition of up to $k = 10$ traitors, smaller than $5.4\,10^{-3}$ for any coalition of up to $k = 100$ traitors. Security requirements are met since $q^n = 2^{171}$ and the number of monomials in a building block of $F_{\mathcal{K}}$ is 25270, so that in any formal composition of three of them the number of monomials is already more than $90\,10^6$ and in any formal composition of three blocks it is already more than $3\,10^{13}$. In that case, the size of any equivalent decryption key is 916 KB. The 1330 monomials can be computed in 1520 multiplications in $\mathbb{K}$ so that a building block requires 26790 multiplications and it takes about 900000 multiplications to evaluate any description $F_{\mathcal{K}}$ on the 171 bits of the input. The overhead is obviously of 171 bits.

k	10	100	1000
$P_{k,\mathrm{FA}}$	$< 2\,10^{-10}$	$< 2.2\,10^{-8}$	$< 2.3\,10^{-6}$
$P_{k,\mathrm{ND}}$	$< 1.2\,10^{-7}$	$< 1.5\,10^{-5}$	$< 1.5\,10^{-3}$

$N = 10^6$, $r = 32$, $\rho = 13$, $n = 5$, $\mathbb{F} = \mathrm{GF}(2^8)$.

k	10	100	1000
$P_{k,\mathrm{FA}}$	$< 1.4\,10^{-19}$	$< 1.5\,10^{-17}$	$< 1.6\,10^{-15}$
$P_{k,\mathrm{ND}}$	$< 5\,10^{-7}$	$< 5.4\,10^{-5}$	$< 5.4\,10^{-3}$

$N = 10^6$, $r = 33$, $\rho = 10$, $n = 19$, $\mathbb{F} = \mathrm{GF}(2^9)$.

Fig. 5. Summary of parameters and corresponding probabilities

6 Conclusion

A novel iterative block cipher which can be operated in a traceable manner has been introduced. The attacks investigated in our initial security analysis are easy to prevent by properly selecting system parameters. Improvements in these attacks are of course not precluded, since no reduction proof of the security to a well identified mathematical problem was found apart from obvious connection to the "Isomorphism of Polynomial" problem. Risks are obviously higher than for usual symmetric ciphers. Natural questions also arise: What security does the "Isomorphism of Polynomials" problem provide for small values of the number n of variables like those suggested in Section 5? Also, other building blocks could be considered, e.g. variants with two or more branches in each extended C^* block. Studying the effects of releasing the constraint that the monomial functions be distinct may lead to some performance improvements. We also note that since each user possesses an equivalent description, he is able to broadcast data to every other user. Besides traitor tracing, another interesting application of the proposed construction is whitebox cryptography [5]. Indeed, advantage can be taken from the fact that one can easily construct a huge number of equivalent descriptions, while those descriptions can be made arbitrarily large.

In its current shape, the proposed traceable block cipher has the advantage of being very insensitive to the maximum number of traitors tolerated while accommodating a large number of users. Due to its intrinsic block cipher structure and due to the fact that it does not generate any data expansion overhead, its implementation can be made very efficient.

References

1. Elad Barkan and Eli Biham, *In how many ways can you write Rijndael?*, available from the e-print at `http://eprint.iacr.org/2002/157/`.
2. Alex Biryukov, Christophe De Canniere, An Braeken, and Bart Preneel, *A Toolbox for Cryptanalysis: Linear and Affine Equivalence Algorithms*, Advances in Cryptology – EUROCRYPT 2003 (Eli Biham, ed.), Lecture Notes in Computer Science, vol. 2656, Springer-Verlag, 2003, pp. 33–50.
3. Dan Boneh and Matthew Franklin, *An Efficient Public Key Traitor Tracing Scheme*, Advances in Cryptology – CRYPTO '99 (Michael Wiener, ed.), Lecture Notes in Computer Science, vol. 1666, Springer-Verlag, 1994, pp. 338–353.
4. Benny Chor, Amos Fiat, and Moni Naor, *Tracing Traitors*, Advances in Cryptology – CRYPTO '94 (Yvo G. Desmedt, ed.), Lecture Notes in Computer Science, vol. 839, Springer-Verlag, 1994, pp. 257–270.
5. Stanley Chow, Philip Eisen, Harold Johnson, and Paul C. Van Oorschot, *White-Box Cryptography and an AES Implementation*, Selected Areas in Cryptography – SAC 2002 (K. Nyberg and H. Heys, eds.), Lecture Notes in Computer Science, vol. 2595, Springer-Verlag, 2002, pp. 250–270.
6. Nicolas Courtois, Louis Goubin, Willi Meier, and Jean-Daniel Tacier, *Solving Underdefined Systems of Multivariate Quadratic Equations*, Public Key Cryptography – PKC 2002 (David Naccache and Pascal Paillier, eds.), Lecture Notes in Computer Science, vol. 2274, Springer-Verlag, 2002, pp. 211–227.

7. Nicolas Courtois, Louis Goubin, and Jacques Patarin, *C^{*-+} and HM: Variations around two schemes of T. Matsumoto and H. Imai*, Advances in Cryptology – ASIACRYPT '98 (Kazuo Ohta and Dingyi Pei, eds.), Lecture Notes in Computer Science, vol. 1514, Springer-Verlag, 1998, pp. 35–49.
8. Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir, *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*, Advances in Cryptology – EUROCRYPT 2000 (Bart Preneel, ed.), Lecture Notes in Computer Science, vol. 1807, Springer-Verlag, 2000, pp. 392–407.
9. Nicolas T. Courtois, Magnus Daum, and Patrick Felke, *On the Security of HFE, HFEv- and Quartz*, Public Key Cryptography – PKC 2003 (Yvo G. Desmedt, ed.), Lecture Notes in Computer Science, vol. 2567, Springer-Verlag, 2003, pp. 337–350.
10. Jean-Charles Faugère and Antoine Joux, *Algebraic Cryptanalysis of Hidden Field Equation (HFE) Cryptosystems Using Gröbner Bases*, Advances in Cryptology – CRYPTO 2003 (Dan Boneh, ed.), Lecture Notes in Computer Science, vol. 2729, Springer-Verlag, 2003, pp. 44–60.
11. Aviad Kipnis and Adi Shamir, *Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization*, Advances in Cryptology – CRYPTO '99 (Michael Wiener, ed.), Lecture Notes in Computer Science, vol. 1666, Springer-Verlag, 1999, pp. 19–30.
12. Paul Kocher, Joshua Jaffe, and Benjamin Jun, *Differential Power Analysis*, Advances in Cryptology – CRYPTO '99 (Michael Wiener, ed.), Lecture Notes in Computer Science, vol. 1666, Springer-Verlag, 1999, pp. 388–397.
13. Tsutomu Matsumoto and Hideki Imai, *Public Quadratic Polynomial-tuples for Efficient Signature Verification and Message Encryption*, Advances in Cryptology – EUROCRYPT '88 (Cristoph G. Günther, ed.), Lecture Notes in Computer Science, vol. 330, Springer-Verlag, 1988, pp. 419–453.
14. Moni Naor and Benny Pinkas, *Threshold Traitor Tracing*, Advances in Cryptology – CRYPTO '98 (Hugo Krawczyk, ed.), Lecture Notes in Computer Science, vol. 1462, Springer-Verlag, 1998, pp. 502–517.
15. Jacques Patarin, *Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt '88*, Advances in Cryptology – CRYPTO '95 (Vangalur S. Alagar and Maurice Nivat, eds.), Lecture Notes in Computer Science, vol. 963, Springer-Verlag, 1995, pp. 248–261.
16. Jacques Patarin, *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*, Advances in Cryptology – EUROCRYPT 1996 (U. Maurer, ed.), Lecture Notes in Computer Science, vol. 1070, Springer-Verlag, 1996, pp. 33–48.
17. Jacques Patarin, Louis Goubin, and Nicolas Courtois, *Improved Algorithms for Isomorphisms of Polynomials*, Advances in Cryptology – EUROCRYPT '98 (Kaisa Nyberg, ed.), vol. 1403, 1998, pp. 184–200.
18. Douglas R. Stinson and Ruizhong Wei, *Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes*, SIAM Journal on Discrete Mathematics **11** (1998), no. 1, 41–53.
19. *Specifications of SFLASH*, available from the site of the NESSIE workshop at `https://www.cosic.esat.kuleuven.ac.be/nessie/workshop/`.

Active attack against HB$^+$: a provably secure lightweight authentication protocol

H. Gilbert, M. Robshaw and H. Sibert

Much research has focused on providing RFID tags with lightweight cryptographic functionality. The HB$^+$ authentication protocol was recently proposed and claimed to be secure against both passive and active attacks. A linear-time active attack against HB$^+$ is proposed.

Introduction: Much research has focused on providing RFID tags with lightweight cryptographic functionality. Particular interest has been paid to the issue of authentication, in order to both prevent counterfeiting and enhance privacy. In this Letter, we focus on an authentication protocol by Juels and Weis [1], which is to be presented at Crypto'05. This protocol, called HB$^+$, provides a symmetric authentication scheme that is claimed to be well-suited to low-cost devices such as RFID tags. In [1], HB$^+$ is presented as an enhanced variant of a protocol due to Hopper and Blum [2] (and known as the HB protocol). While HB was proven secure against passive attacks under the 'learning parity with noise' (LPN) hardness assumption, HB$^+$ is claimed to be secure against both passive and active attacks and a security proof is provided [1]. In this Letter, we show that HB$^+$ is vulnerable to an efficient active attack with linear computational and communication complexity. In this Letter, we first provide an outline of the LPN problem and the HB and HB$^+$ protocols; we then describe the attack and assess its cost; finally, we consider the implications of our observations.

LPN problem and HB and HB$^+$ protocols: In this Section we briefly review the HB and HB$^+$ protocols. It is interesting to note that they have much in common with a scheme first presented in [3]. Roughly speaking, the LPN problem requires an adversary to recover a k-bit secret x after being given several equations of the form $b_i = a_i \cdot x \oplus v_i$, with unknowns x and the v_is. Here v_i is a (noise) bit equal to 1 with a probability $\eta \in [0, 1/2]$. Throughout we denote the Hamming weight of a vector x by $|x|$.

Definition 1: The LPN problem with security parameters q, k, η, with $\eta \in [0, 1/2]$ is defined as follows: given a random $q \times k$ binary matrix A, a random k-bit vector x, a vector v such that $|v| \leq \eta q$, and the product $z = A \cdot x \oplus v$, find a k-bit vector x' such that $|A \cdot x' \oplus z| \leq \eta q$.

The HB scheme is a symmetric-key authentication protocol that is directly related to the LPN problem. The round described in Fig. 1 is repeated r times. The tag is authenticated if the checking procedure fails at most ηr times.

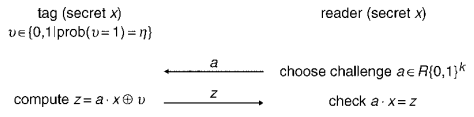


Fig. 1 *One round of HB protocol*

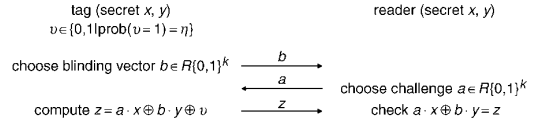


Fig. 2 *One round of HB$^+$ protocol*

Note that the HB scheme is not secure against active attacks. Since v is strictly less than $1/2$, by challenging the tag with some chosen a several times the value $a \cdot x$ will be revealed. Gaussian elimination will therefore give x once k equations with linearly independent as have been retrieved. The HB$^+$ protocol is an augmented version of the basic HB scheme. The aim of the HB$^+$ protocol [1] is to prevent the extraction of tag secrets by corrupt readers using such chosen challenges. In HB$^+$, the basic HB protocol in Fig. 2 is repeated r times and the tag successfully authenticated if the check fails at most ηr times (note 1).

Active attack against HB$^+$: Here we show a simple active attack against the HB$^+$ protocol. The attack requires that the adversary is capable of manipulating challenges sent by a legitimate reader to a legitimate tag during the authentication exchanges, and to check whether this manipulation results (or not) in an authentication failure. In detail, the attack consists of choosing a constant k-bit vector δ and using it to perturb the challenges sent by a legitimate reader to the tag: δ is XOR'ed to each authentication challenge for each of the r rounds of authentication. If the authentication process is successful, then we must have that $\delta \cdot x = 0$ with overwhelming probability. If authentication does not succeed then $\delta \cdot x = 1$ with overwhelming probability.

The attack is illustrated in Fig. 3 for one round of the HB$^+$ protocol. We use the same δ in all r rounds of the protocol. Acceptance or rejection by the reader would thereby reveal one bit of secret information. To retrieve the k-bit secret x, it is sufficient to repeat the full protocol k times for linearly independent δs, and to solve the resulting system. Conveniently, one can choose δs with a single nonzero bit. Once x has been derived, the attacker can either immediately impersonate the tag using commitment values $b = 0$, or the attacker can then derive (see note 2) the k-bit secret y using linearly independent linear combinations $b \cdot y$. Another side-effect of the disclosure of x is that the privacy of the tag's identity is also compromised.

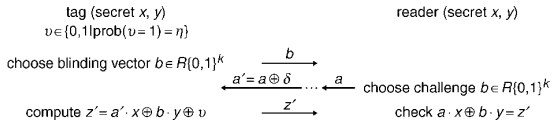


Fig. 3 *Attack on one round of HB$^+$ protocol*

Discussion: We have described an active attack against the HB$^+$ protocol [1] that has a complexity linear in the length of the keys and number of rounds. It is interesting to consider how such an attack evades the proof of security that accompanies the HB$^+$ protocol [1]. The main problem is that the security model in [1] does not take account of the potential leak of information by a legitimate verifier as well as a legitimate prover. In the attack, each 'accept' or 'reject' outcome from a legitimate verifier provides one bit of information about the shared secret key x. Moreover, an attacker is not restricted to attacking the tag only, and then the reader only, as the proof of security demands. Instead the adversary interacts with both at the same time to gain an advantage.

From a practical point of view, the most obvious way to mount the attack is to use a false reader to communicate with the legitimate tag and a false tag to communicate with the legitimate reader. Note that the false reader and tag need not be in the same physical place, they need only communicate with each other. However, such a man-in-the-middle configuration is not really required. Instead an adversary need only cause controlled perturbations to the challenges sent from the reader to the tag.

It is worth noting that, while the attacker interacts with both the tag and the reader, this is done in an unintrusive manner. From the point of view of the reader, either authentication with a legitimate tag has been successful or it has been unsuccessful (due, for instance, to a noisy transmission). In both cases the attacker gains information and the reader is unlikely to be aware that an attack has taken place.

Conclusion: While protocols with a proof of security are to be welcomed, caution demands that the security model be sufficiently robust. Given the practical nature of the attack outlined here, it is fair to conclude that the security model considered in [1] is too restrictive and that the HB$^+$ protocol is vulnerable to a realistic active attack.

Notes: 1. A straightforward generalisation of HB$^+$ consists in replacing the authentication acceptance threshold ηr by $\eta' r$, where η' is a constant which may differ from η. It is easy to see that the attack described in this Letter is also applicable to this slight variant of HB$^+$.

2. These can be obtained by using, for instance, a false tag that sends

challenge a. If the authentication is successful then $b \cdot y = 0$ with overwhelming probability. If authentication does not succeed then $b \cdot y = 1$ with overwhelming probability.

19 July 2005
Electronics Letters online no: 20052622
doi: 10.1049/el:20052622

H. Gilbert and M. Robshaw (*France Télécom, R&D Division, 38–40, rue du Général Leclerc, 92794 Issy les Moulineaux, Cedex 9, France*)

E-mail: matt.robshaw@francetelecom.com

H. Sibert (*France Télécom, R&D Division, 42, rue des Coutures, BP 6243, 14066 Caen, Cedex 4, France*)

References

1 Juels, A., and Weis, S.A.: 'Authenticating pervasive devices with human protocols' *in* Shoup, V. (Ed.): Advances in Cryptology – Crypto 05, *Lect. Notes Comput. Sci.* (Springer-Verlag) (to appear 2005). Also available via http://www.rsasecurity.com/rsalabs/

2 Hopper, N.J., and Blum, M.: 'Secure human identification protocols' *in* Boyd, C. (Ed.): 'Advances in cryptology – Asiacrypt'01, *Lect. Notes Comput. Sci.*, 2001, **2248**, pp. 52–66

3 Gilbert, H.: 'Techniques for low cost authentication and message authentication' *in* Quisquater, J.J. (Ed.): 'Smart card research applications', Proc. CARDIS'98, Louvain-la-Neuve, Belgium, September 1998, *Lect. Notes Comput. Sci.*, 2000, **1820**, pp. 183–192

Good Variants of HB$^+$ are Hard to Find

H. Gilbert, M.J.B. Robshaw, and Y. Seurin

Orange Labs,
38–40 rue General Leclerc,
Issy les Moulineaux, France
{forename.name}@orange-ftgroup.com

Keywords: HB$^+$, RFID tags, authentication, LPN

Abstract. The strikingly simple HB$^+$ protocol of Juels and Weis [11] has been proposed for the authentication of low-cost RFID tags. As well as being computationally efficient, the protocol is accompanied by an elegant proof of security. After its publication, Gilbert *et al.* [8] demonstrated a simple man-in-the-middle attack that allowed an attacker to recover the secret authentication keys. (The attack does not contradict the proof of security since the attacker lies outside the adversarial model.) Since then a range of schemes closely related to HB$^+$ have been proposed and these are intended to build on the security of HB$^+$ while offering resistance to the attack of [8]. In this paper we show that many of these variants can still be attacked using the techniques of [8] and the original HB$^+$ protocol remains the most attractive member of the HB$^+$ family.

1 Introduction

The extension of cryptographic functions to low-cost RFID tags is an active area of research. The combination of novel security requirements and demanding physical environments provides a major incentive to the development of new designs and techniques.

Juels and Weis introduced HB$^+$ at Crypto 2005 [11]. The protocol is a multi-round symmetric key authentication protocol where each round consists of three communications between the reader and the tag. On the tag, HB$^+$ is computationally lightweight since it requires only simple bit-wise operations. Furthermore, the protocol is supported by a proof of security against an active attacker in what the HB$^+$ designers call the *detection-based* model. In this model adversaries can interrogate a tag in any way they wish, and then they must try and pass themselves off as an authentic tag to a legitimate reader. In loose terms, Juels and Weis show that for such an attack to succeed the attacker would be able to break an instance of the *Learning Parity with Noise (LPN)* problem which is believed to be hard.

However, if we allow the attacker to do a little more—*i.e.* if we leave the detection-based model—then HB$^+$ becomes susceptible to a simple attack. In particular, if an attacker can slightly modify messages from the reader and observe whether the legitimate reader still accepts the legitimate tag, then the

attacker can recover secret key information. This is, in essence, the attack of Gilbert *et al.* [8] which we will refer to as the GRS attack in what follows. Some commentators suggest that interfering with the tag-reader communication would be technically difficult. Others claim that forbidding such manipulation during analysis ignores the full characteristics of a potential attack and makes potentially dangerous assumptions on the limitations of an attacker. However this is not the concern of this paper. Instead we will focus on the body of research that has evolved from both HB^+ and the GRS attack.

In his paper introducing the block cipher RC5, Rivest states that " ... a simpler structure is perhaps more interesting to analyze and evaluate ... " [19]. This is now a well-established principle in cryptographic design and the simplicity of both the original HB^+ proposal and the GRS attack have given rise to a number of HB-related protocols in the literature. The goal of these protocols is that they retain some of the successful properties of HB^+ while also resisting the GRS attack. In this paper we will take a critical look at such variants. We can show that despite claims to the contrary, the GRS attack can often be applied or extended to these new variants. Thus the tolerance of the new schemes to the GRS attack is often equivalent to that of HB^+ and yet, at the same time, they suffer from additional complexity and/or reduced practicality. In short, we show that HB^+ variants that resist the GRS attack are not that easy to come by.

Our paper is organised as followed. After introducing the HB^+ protocol we turn our attention to the variants HB^{++}, HB^*, HB-MP′, and HB-MP. These are treated in the order they appear in the literature and in Sections 3, 4 and 5 we provide a description and security analysis of each. We then discuss the implications of our work in Section 6 and draw our conclusions. It should be noted that our work is not concerned with the proofs of security for HB^+ or its variants. Instead our focus is on applications of the GRS attack.

Throughout we aim to use established notation. There will be some interplay between vectors $\boldsymbol{x} \in \{0,1\}^k$ and scalars in $\mathbb{F}_2$ and we use bold type $\boldsymbol{x}$ to indicate a vector while scalars x are written in normal text. The *scalar product* of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ will be written as $\boldsymbol{x} \cdot \boldsymbol{y}$ while their bitwise addition will be denoted using $\oplus$ just as for single bits. We denote the *Hamming weight* of $\boldsymbol{x}$ by $\text{Hwt}(\boldsymbol{x})$. Several protocols require a rotation of $\boldsymbol{x}$ by i bit positions to the left; we denote this operation by $\text{ROT}_i(\boldsymbol{x})$.

2 The HB^+ Protocol and the GRS Attack

There are now several protocols based on HB^+ and these offer a variable level of security and practicality. We start by reviewing the original protocol, though all depend for their security on the conjectured hardness of the *Learning Parity with Noise* (LPN) problem [11].

> **LPN Problem**. Let A be a random $(q \times k)$-binary matrix, let $\boldsymbol{x}$ be a random k-bit vector, let $\eta \in]0, \frac{1}{2}[$ be a noise parameter, and let $\boldsymbol{\nu}$ be a random q-bit vector such that $\text{Hwt}(\boldsymbol{\nu}) \leq \eta q$. Given A, η, and $\boldsymbol{z} = A \cdot \boldsymbol{x}^t \oplus \boldsymbol{\nu}^t$, find a k-bit vector $\boldsymbol{y}^t$ such that $\text{Hwt}(A \cdot \boldsymbol{y}^t \oplus \boldsymbol{\nu}^t) \leq \eta q$.

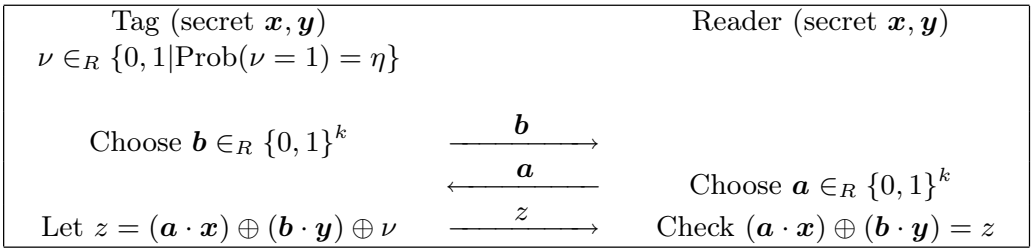


Fig. 1. One single round of HB$^+$ [11]. The entire authentication process requires r rounds and, in this basic form, each round consists of the three passes shown. Provided the tag fails less than some threshold t number of rounds, the tag is authenticated.

We will not consider the intractability of the LPN problem directly in this paper, though we observe that the problem is not as difficult as was originally thought [7,15]. This means that the parameters for HB$^+$ and its variants often need to be increased.

2.1 The HB$^+$ protocol

The HB$^+$ protocol is outlined in Figure 1. The tag and the reader share two k-bit secrets $\boldsymbol{x}$ and $\boldsymbol{y}$. One round of HB$^+$ is as follows: the tag selects a random k-bit blinding vector $\boldsymbol{b}$ and sends it to the reader. The reader challenges the tag with a random k-bit vector $\boldsymbol{a}$. The tag computes the response $z = (\boldsymbol{a} \cdot \boldsymbol{x}) \oplus (\boldsymbol{b} \cdot \boldsymbol{y}) \oplus \nu$, where ν is a random noise bit taking the value 1 with probability $\eta \in]0, \frac{1}{2}[$. This is repeated for r rounds, and the tag is authenticated if the number of errors (*i.e.* z distinct from $(\boldsymbol{a} \cdot \boldsymbol{x}) \oplus (\boldsymbol{b} \cdot \boldsymbol{y})$) is less than a threshold $t = ur$ where $u \in]\eta, \frac{1}{2}[$. The difficulty of the LPN problem [7,11,13,15] is related to both k and the parameter η which governs how much noise is added to the correct computations by a valid tag. In its original state HB$^+$ consists of multiple rounds each of three passes. The parallel version of HB$^+$—for which a proof of security also exists [13,14]—compresses the multiple rounds into one single three-pass round.

Immediately one can see that HB$^+$ requires very modest on-tag computation. Leaving aside generating $\boldsymbol{b}$ and the bit ν, computation on the tag is reduced to a dot-product, which can be computed bit-wise, and a single bit exclusive-or. The novelty and simplicity of HB$^+$ immediately generated considerable interest. Katz and Shin [13] closed gaps and extended the original proof of security while follow-on work by Katz and Smith [14] considered different noise levels.

2.2 An active attack on HB$^+$

A simple active attack on HB$^+$ is provided in [8]. The attack applies equally to the serial and the parallel versions of HB$^+$. For this attack it is assumed that an adversary can manipulate challenges sent by a legitimate reader to a legitimate tag during authentication. Further, we assume that the adversary learns whether such manipulation leads to an authentication failure or not.

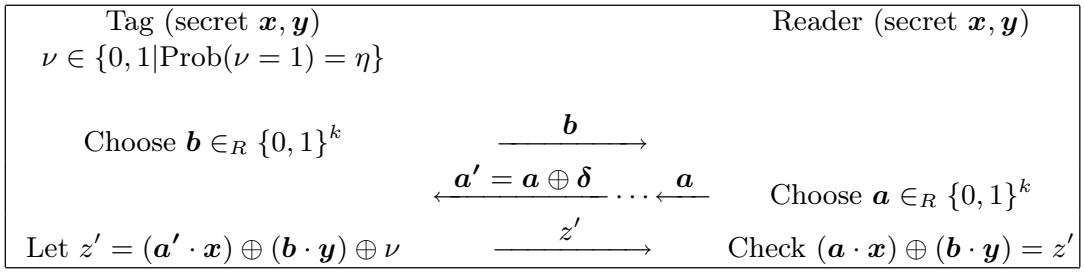


Fig. 2. The attack of [8] on HB^+. The adversary modifies the communications between reader and tag (by adding some perturbation $\boldsymbol{\delta}$ and notes whether authentication is still successful. This reveals one bit of secret information.

The attack consists of choosing a constant k-bit vector $\boldsymbol{\delta}$ and using it to perturb the challenges sent by a legitimate reader to the tag; $\boldsymbol{\delta}$ is exclusive-or'ed to each authentication challenge for each of the r rounds of authentication. If the authentication process is successful then we must have that $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ with overwhelming probability. Otherwise $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$ with overwhelming probability. Thus we gain one bit of secret information. The attack is illustrated in Figure 2 for one round of the HB^+ protocol.

To retrieve the k-bit secret $\boldsymbol{x}$ one can repeat the attack k times for linearly independent $\boldsymbol{\delta}$'s and solve the resulting system. Conveniently, an adversary can choose $\boldsymbol{\delta}$'s with a single non-zero bit. With $\boldsymbol{x}$ an attacker can impersonate the tag by setting $\boldsymbol{b} = \boldsymbol{0}$. Alternatively, an attacker can emulate a false tag using $\boldsymbol{x}$, send a chosen blinding factor $\boldsymbol{b}$ to a legitimate reader, and return $\boldsymbol{a} \cdot \boldsymbol{x}$ to the challenge $\boldsymbol{a}$. If successful $\boldsymbol{b} \cdot \boldsymbol{y} = 0$, otherwise $\boldsymbol{b} \cdot \boldsymbol{y} = 1$, with overwhelming probability. Thus $\boldsymbol{y}$ can be recovered with k linearly independent $\boldsymbol{b}$.

The attack is mathematically simple though it is not covered by the existing proof of security since the attacker needs to manipulate challenges and know whether authentication is successful [11]. Yet, despite the technical difficulties of interfering in a tag-reader exchange, the attack should be viewed as *certificational*. Certainly a variant of HB^+ that is both computationally simple and resistant to the GRS attack would be of some considerable interest.

All the variants to HB^+we will consider in the following sections share some properties with HB^+. In particular, they all consist of the repetition of r basic rounds. An honest tag interacting with an honest reader may be rejected with a probability we denote P_{FR} (false rejection probability). An adversary answering randomly at each round will be authenticated with a probability we denote P_{FA} (false acceptance probability). For HB^+ these are given by $P_{\text{FR}} = \sum_{i=t+1}^{r} \binom{r}{i} \eta^i (1-\eta)^{r-i}$ and $P_{\text{FA}} = \frac{1}{2^r} \sum_{i=0}^{t} \binom{r}{i}$.

3 The Variant HB^{++}

Description of HB^{++}. The protocol HB^{++} is proposed by Bringer *et al.* [3]. The complete proposal consists of two stages. In the first, illustrated in Figure 3,

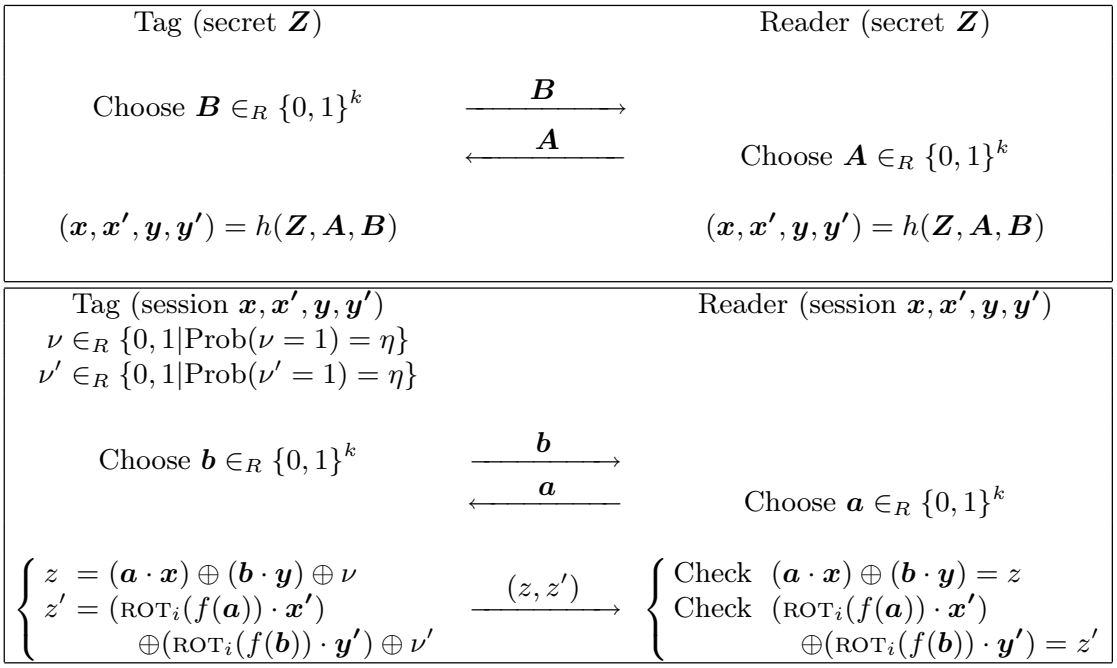


Fig. 3. The HB^{++} protocol. Above: At the start of each authentication, a preliminary exchange of $2k$ bits and the use of a universal hash function h are required to derive the session secrets $\boldsymbol{x}, \boldsymbol{x'}, \boldsymbol{y}, \boldsymbol{y'}$. Below: One single round i of HB^{++}. The entire authentication process requires r rounds and, in this basic form, each round consists of the three passes shown. Provided the tag fails both tests less than some threshold t number of rounds, the tag is authenticated.

four k-bit secrets $\boldsymbol{x}, \boldsymbol{x'}, \boldsymbol{y}, \boldsymbol{y'}$ are derived by the tag and the reader from a shared secret $\boldsymbol{Z}$. These derived secrets might be viewed as session keys. Then HB^{++} consists of r rounds where each round consists of three passes, just as in HB^+.

A single round of HB^{++} is illustrated in Figure 3. We can see that things are slightly more complicated than in HB^+. In particular, once the blinding vector $\boldsymbol{b}$ and the challenge $\boldsymbol{a}$ have been sent, there are two on-tag computations.

The first looks like the HB^+ on-tag computation and simply consists in computing $z = (\boldsymbol{a} \cdot \boldsymbol{x}) \oplus (\boldsymbol{b} \cdot \boldsymbol{y}) \oplus \nu$. The second involves a permutation f (which is in fact a layer of five-bit S-boxes) and also requires that k-bit quantities be rotated by i bit positions where i denotes the round (rounds are numbered from 0 to $r-1$). The second response bit is given by $z' = (\text{ROT}_i(f(\boldsymbol{a})) \cdot \boldsymbol{x'}) \oplus (\text{ROT}_i(f(\boldsymbol{b})) \cdot \boldsymbol{y'}) \oplus \nu'$. Both noise bits ν and ν' are randomly chosen according to the noise parameter η. For the tag to be authenticated, the number of erroneous z answers *and* the number of erroneous z' answers must be less than some threshold $t = ur$, where $u \in]\eta, \frac{1}{2}[$. Consequently the false rejection and false acceptance probabilities are:

$$P_{\text{FR}} = 1 - \left(\sum_{i=0}^{t} \binom{r}{i} \eta^i (1-\eta)^{r-i} \right)^2 \quad \text{and} \quad P_{\text{FA}} = \left(\frac{1}{2^r} \sum_{i=0}^{t} \binom{r}{i} \right)^2 .$$

The proposed number of rounds is not given, but the parameters in [3], in particular $k = 80$, give a much-reduced level of security when compared to HB^+.

Variant of Piramuthu. Piramuthu [18] proposes a modification to HB^{++} but the details are unclear. The main difference with HB^{++} appears to be the removal of the first on-tag computation. However this means that what remains is equivalent to HB^+ itself. Thus it will have all the characteristics of HB^+ while at the same time possessing a heavier on-tag computation. We do not consider this variant further.

Attacking HB^{++} without the renewed secrets. We first show how to attack HB^{++} when the preliminary phase to renew the secrets $(\boldsymbol{x}, \boldsymbol{x'}, \boldsymbol{y}, \boldsymbol{y'})$ is omitted. We note that Wagner described an attack on a preliminary version of HB^{++} where the rotations are omitted, which was described in the original paper [3]. In this attack, the attacker guesses a short portion of the secrets $\boldsymbol{x}$ and $\boldsymbol{x'}$ and then modifies the challenges sent by the reader but also the answer returned by the tag accordingly to his guess. If the tag is authenticated, the attacker knows that with high probability his guess was right. Bringer *et al.* introduced the rotations to counter this attack. The rationale is that this way, even if the perturbation of $\boldsymbol{a}$ is localized, the perturbation of $f(\boldsymbol{a})$ will affect all bits of the secret $\boldsymbol{x'}$. It seems however that the following fact was overlooked: it is not necessary for the attacker to perturb *all* the rounds of the protocol but only a fixed fraction to be able to gain information through the decision of the reader. As we will show now, this leads to an efficient variant of the GRS attack.

Unlike the attack of Wagner, the attack we describe doesn't require that we modify the answers of the tag. As in the GRS attack, the attacker adds a fixed vector $\boldsymbol{\delta}$ to the challenges $\boldsymbol{a_i}$ sent by the reader, *but only for a fixed number of rounds* $s < r$ (say the first s rounds). Let σ_i and σ'_i denote the total error vectors on the answers z_i and z'_i of the tag at round i. For rounds $i = 0$ to $s - 1$, one has $\sigma_i = \nu_i \oplus \boldsymbol{\delta} \cdot \boldsymbol{x}$ and $\sigma'_i = \nu'_i \oplus \boldsymbol{\delta'_i} \cdot \boldsymbol{x'}$ where $\boldsymbol{\delta'_i} = \mathrm{ROT}_i(f(\boldsymbol{a_i} \oplus \boldsymbol{\delta}) \oplus f(\boldsymbol{a_i}))$, whereas for rounds $i = s$ to $r - 1$, one simply has $\sigma_i = \nu_i$ and $\sigma'_i = \nu'_i$. Let N (resp. N') denote the number of answers z_i (resp. z'_i) in error. The function f was chosen to satisfy good differential properties, meaning that for a fixed $\boldsymbol{\delta}$ and a fixed $\boldsymbol{c}$, $\Pr_{\boldsymbol{a}}[f(\boldsymbol{a} \oplus \boldsymbol{\delta}) \oplus f(\boldsymbol{a}) = \boldsymbol{c}]$ is very small for most values of $\boldsymbol{\delta}$. Hence the noise bits σ'_i for rounds 0 to $s - 1$ are close to uniformly distributed and we may assume[1] that, whatever $\boldsymbol{\delta}$, N' is distributed as the sum of s Poisson trials taking the value 0 or 1 with probability $\frac{1}{2}$ and $r - s$ Poisson trials taking the value 0 with probability $1 - \eta$ and 1 with probability η. The expected value of N' is $\mu' = \frac{s}{2} + \eta(r - s) = \frac{1}{2}(1 - 2\eta)s + \eta r$. Unlike N', the distribution of N depends on the value of $\boldsymbol{\delta} \cdot \boldsymbol{x}$. When $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$, the answers z_i are undisturbed and N is distributed as the sum of r Poisson trials taking the value 0 with probability $1-\eta$

[1] Note that this is strictly speaking an approximation and that in fact the distribution of $(\sigma'_0, \ldots, \sigma'_{s-1})$ will be nearly uniform for an overwhelming fraction of $\boldsymbol{x'}$ and $\boldsymbol{\delta}$. Concrete values will depend on the parameter Δ_f defined in [3].

and 1 with probability η. The expected value of N in this case is $\mu_0 = \eta r < t$. When $\boldsymbol{\delta}\cdot\boldsymbol{x} = 1$, the s first answers z_i are correct with probability η and incorrect with probability $1-\eta$, while the $r-s$ remaining rounds are undisturbed. In that case, N' is distributed as the sum of s Poisson trials taking the value 0 with probability η and 1 with probability $1-\eta$ and $r-s$ Poisson trials taking the value 0 with probability $1-\eta$ and 1 with probability η. The expected value of N is $\mu_1 = (1-\eta)s + \eta(r-s) = (1-2\eta)s + \eta r$. Consequently, if we choose s such that $\mu' < t$, and $\mu_1 > t$, the number of errors on z' will be less than t with high probability, and the reader's decision will indicate whether the number of errors on z was less or more than t, which in turn will indicate whether $\boldsymbol{\delta}\cdot\boldsymbol{x} = 0$ or 1.

Going into details, we will compute the advantage of the attacker guessing $\boldsymbol{\delta}\cdot\boldsymbol{x} = 0$ when the reader accepts and $\boldsymbol{\delta}\cdot\boldsymbol{x} = 1$ when the reader rejects. Denoting WG the event that the guess is wrong, we will upper bound the probability of WG as follows:

$$\begin{aligned}
\Pr[\mathrm{WG}] &= \frac{1}{2}\left(\Pr[\mathrm{WG} \mid \boldsymbol{\delta}\cdot\boldsymbol{x} = 0] + \Pr[\mathrm{WG} \mid \boldsymbol{\delta}\cdot\boldsymbol{x} = 1]\right) \\
&= \frac{1}{2}\left(\Pr[\mathcal{R} \text{ rejects} \mid \boldsymbol{\delta}\cdot\boldsymbol{x} = 0] + \Pr[\mathcal{R} \text{ accepts} \mid \boldsymbol{\delta}\cdot\boldsymbol{x} = 1]\right) \\
&= \frac{1}{2}\Big(\Pr[(N > t) \vee (N' > t) \mid \boldsymbol{\delta}\cdot\boldsymbol{x} = 0] \\
&\qquad\qquad + \Pr[(N \le t) \wedge (N' \le t) \mid \boldsymbol{\delta}\cdot\boldsymbol{x} = 1]\Big) \\
&\le \frac{1}{2}\left(\Pr[N' > t] + \Pr[N > t \mid \boldsymbol{\delta}\cdot\boldsymbol{x} = 0] + \Pr[(N \le t) \mid \boldsymbol{\delta}\cdot\boldsymbol{x} = 1]\right) \\
&\le \frac{1}{2}\left(e^{-\frac{(t-\mu')^2}{3\mu'}} + e^{-\frac{(t-\mu_0)^2}{3\mu_0}} + e^{-\frac{(\mu_1-t)^2}{2\mu_1}}\right)
\end{aligned}$$

where the last inequalities come from the Chernoff bounds (see Appendix). According to the expressions of μ' and μ_1, the condition on s to have $\mu' < t$ and $\mu_1 > t$ is

$$\frac{t-\eta r}{1-2\eta} < s < 2\frac{t-\eta r}{1-2\eta}.$$

Whether such s exist will depend on the parameters of the scheme, however we note that in order for the protocol to have a low false rejection probability, t has to be sufficiently distinct from ηr. In particular, taking $t = \lceil \eta r \rceil$ yields $P_{\mathrm{FR}} \simeq 0.4$ (see Section 6), which is unacceptable. Hence, it is arguable that such s will exist. However, concrete values in the formulae show that it is uncertain for the attacker to make a guess when the reader rejects, as the probability for this to happen when $\boldsymbol{\delta}\cdot\boldsymbol{x} = 0$ (due to $N' > t$) may be quite high when μ' is close to t. A much better strategy is to make a guess only when the reader accepts, guessing that $\boldsymbol{\delta}\cdot\boldsymbol{x} = 0$. In this case, the probability of a wrong guess is given by $\Pr[\mathrm{WG_a}] = \Pr[\boldsymbol{\delta}\cdot\boldsymbol{x} = 1 \mid \mathcal{R} \text{ accepts}] = \frac{1}{2}P_{\mathrm{a}}^{-1}\Pr[\mathcal{R} \text{ accepts} \mid \boldsymbol{\delta}\cdot\boldsymbol{x} = 1]$, where P_{a} is the probability that the reader accepts for a random $\boldsymbol{\delta}$. $\Pr[\mathrm{WG_a}]$ decreases with s as the gap between t and μ_1 increases. The cost is that a higher number of attempts will be required to retrieve $\boldsymbol{x}$, namely $O(k \cdot P_{\mathrm{a}}^{-1})$, which may become impractical as s tends to r since P_{a} becomes negligible. However,

for $s = \left\lfloor 2\frac{t-\eta r}{1-2\eta} \right\rfloor$, $\mu' \simeq t$ so that N' is more or less than t with probability roughly $1/2$, and hence the reader accepts with probability roughly $1/4$. We computed concrete values for different set of parameters. For example, when $(r, t, \eta) = (80, 30, 0.25)$ we obtain, with $s = 40$, $\Pr[\mathrm{WG_a}] \simeq 0.007$ and $P_{\mathrm{a}}^{-1} \simeq 3.62$, whereas for $(r, t, \eta) = (160, 60, 0.25)$, we obtain, with $s = 80$, $\Pr[\mathrm{WG_a}] \simeq 0.0002$ and $P_{\mathrm{a}}^{-1} \simeq 3.73$.

Once $\boldsymbol{x}$ has been retrieved with high confidence, $\boldsymbol{x'}$ can be obtained by adding to the i-th challenge a vector $\boldsymbol{\delta_i}$ such that $\boldsymbol{\delta_i} \cdot \boldsymbol{x} = 0$ and $\textsc{rot}_i((f(\boldsymbol{a_i} \oplus \boldsymbol{\delta_i}) \oplus f(\boldsymbol{a_i}))$ is constant, which will give linear equations on $\boldsymbol{x'}$.

Attacking HB^{++} with renewed secrets. Let us now consider the situation where HB^{++} is operated with renewed secrets at each authentication, as recommended by the authors of [3]. We show that while secret renewal apparently protects HB^{++} against a simple application of the GRS attack, a slightly more complex attack remains.

To explain this attack, we need to introduce the function h that is used to derive the 320-bit temporary authentication key $(\boldsymbol{x}, \boldsymbol{x'}, \boldsymbol{y}, \boldsymbol{y'})$ from a permanent 768-bit secret $\boldsymbol{Z}$. This function is derived from the hash functions family WH, a variant of the hash functions family NH on which the UMAC message authentication code is based [2] and which was proposed by Kaps *et al.* in [12].

The instance of WH used to construct h is defined as follows: given two 160-bit words $\boldsymbol{K} = (K_1, \ldots, K_n) \in (\mathbb{F}_{2^{16}})^n$, and $\boldsymbol{M} = (M_1, \ldots, M_n) \in (\mathbb{F}_{2^{16}})^n$, where $n = 10$, the 16-bit word $\mathrm{WH}_{\boldsymbol{K}}(\boldsymbol{M})$ is defined as

$$\mathrm{WH}_{\boldsymbol{K}}(\boldsymbol{M}) = \sum_{i=1}^{n/2} (M_{2i-1} + K_{2i-1}) \cdot (M_{2i} + K_{2i}) \cdot c_i,$$

where the c_i are $\mathbb{F}_{2^{16}}$ constants defined in [12]. The function h results from $t = 20$ invocations of this instance of WH, according to the construction of a hash function family with a larger key and output size named WH^{T} proposed in [12]. Given $\boldsymbol{Z} = (Z_1, \ldots, Z_{n+2(t-1)}) \in (\mathbb{F}_{2^{16}})^{n+2(t-1)} = (\mathbb{F}_{2^{16}})^{48}$, and $\boldsymbol{M} = (M_1, \ldots, M_n) \in \mathbb{F}_{2^{16}}^n = (\mathbb{F}_{2^{16}})^{10}$, the t-uple $\mathrm{WH}^{\mathrm{T}}_{\boldsymbol{Z}}(\boldsymbol{M})$ of $\mathbb{F}_{2^{16}}$ words is defined as $\mathrm{WH}^{\mathrm{T}}_{\boldsymbol{Z}}(\boldsymbol{M}) = (\mathrm{WH}_{Z_1 \ldots Z_n}(\boldsymbol{M}), \mathrm{WH}_{Z_3 \cdots Z_{n+2}}(\boldsymbol{M}), \cdots, \mathrm{WH}_{Z_{2t-1} \cdots Z_{n+2t-2}}(\boldsymbol{M}))$. With the previous notation h is defined as $h(\boldsymbol{Z}, \boldsymbol{A}, \boldsymbol{B}) = \mathrm{WH}^{\mathrm{T}}_{\boldsymbol{Z}}(\boldsymbol{M})$ where $\boldsymbol{M} = (\boldsymbol{A} || \boldsymbol{B})$.

One can see from the former equations that given any fixed pair $(\boldsymbol{A}, \boldsymbol{B})$, $h(\boldsymbol{Z}, \boldsymbol{A}, \boldsymbol{B})$ is a known quadratic function of $(Z_1, \ldots, Z_{48})$. However, the security advantage that results from having a quadratic expression rather than a linear one is quite marginal for this particular function. This is due to the following property that immediately results from the definition of WH: for all $(\boldsymbol{A}, \boldsymbol{B})$ pairs, each of the t 16-bit words of $h(\boldsymbol{Z}, \boldsymbol{A}, \boldsymbol{B})$ can be expressed as a known affine function with $\mathbb{F}_{2^{16}}$ coefficients of only 15 unknown words, namely 10 consecutive values of the sequence $(Z_1, \cdots Z_{48})$ and 5 of the 24 products $Z_1 \cdot Z_2$, $Z_3 \cdot Z_4, \ldots, Z_{47} \cdot Z_{48}$. Equivalently, if we consider equations over $\mathbb{F}_2$ instead of $\mathbb{F}_{2^{16}}$, each bit of $h(\boldsymbol{Z}, \boldsymbol{A}, \boldsymbol{B})$ can be expressed for all $(\boldsymbol{A}, \boldsymbol{B})$ pairs as a known

affine function of only 240 unknown bits, namely 160 $\boldsymbol{Z}$ bits and 80 quadratic functions of $\boldsymbol{Z}$ bits. We call hereafter such unknown bits *expanded key bits*.[2]

We now present the cryptanalysis of HB^{++}. We have shown in the former section that, by disturbing a subset of s rounds of an authentication and exploiting the authentication success or failure information for the disturbed protocol, an adversary is capable of getting approximate linear equations involving a subset of the bits of $\boldsymbol{x}$, say the 16 first bits of $\boldsymbol{x}$ (which all linearly depends on the same 240 expanded key bits). If we collect a sufficient number m of such equations, relating to m temporary values $\boldsymbol{x}$, we get an LPN problem in 240 expanded key bits. According to the previous analysis, the error parameter for this LPN problem will typically not be more than 0.01. Levieil and Fouque [15] estimate that such instances of the LPN problem can be solved with about 2^{30} noisy samples and 2^{41} steps of computation and bytes of memory. Thus 240 bits of the expanded key can be recovered by solving an LPN problem of medium complexity. The same method can be applied to recover 240-bit portions of the expanded key allowing the attacker to predict the other 16-bit words of $\boldsymbol{x}$. Once this is done, $\boldsymbol{x}$ can be predicted by the adversary for each authentication. This renders the derivation of m approximate linear equations on $\boldsymbol{x'}$ bits even easier than the initial derivation of approximate equations on $\boldsymbol{x}$ bits and therefore the parts of the expanded key that allow the attacker to compute the value of $\boldsymbol{x'}$ at each authentication can now be derived.

At this stage, the adversary has enough information to impersonate the tag without having to derive the rest of the expanded key and derive $\boldsymbol{y}$ and $\boldsymbol{y'}$. The adversary can re-use the masking vectors $\boldsymbol{b}$ used by the tag in a successful authentication along with its knowledge of $\boldsymbol{x}$ and $\boldsymbol{x'}$ to correct the z and z' values in an appropriate way. All in all, HB^{++} can be cryptanalyzed by solving 10 LPN problems of size 240 bits with small noise parameters. The total number of authentications needed is multiplied by $P_{\mathrm{a}}^{-1} \simeq 4$ as only authentications where the reader accepts are used. For example, for $(r, t, \eta) = (80, 30, 0.25)$, the noise parameter of the LPN problem is roughly 0.01 so that the total number of authentications needed is $4 \times 10 \times 2^{30} \simeq 2^{35}$ and the total complexity is about 2^{44}. Moreover, it is possible to reduce the number of authentications needed at the expense of an increased complexity. Hence HB^{++} offers a much reduced level of security considering the complexity of the operations it requires.

4 The Variant HB*

Description of HB*. The variant HB* is proposed by Duc and Kim [5]. Again it consists of r rounds where each round consists of three passes. This is illustrated in Figure 4. There is an additional secret $\boldsymbol{s}$ which is used to secretly transmit from the tag to the reader a random bit γ, which is 1 with probability $\eta' \in]0, \frac{1}{2}]$, and which determines whether the right answer is computed as $(\boldsymbol{a} \cdot \boldsymbol{x}) \oplus (\boldsymbol{b} \cdot \boldsymbol{y})$ or $(\boldsymbol{a} \cdot \boldsymbol{y}) \oplus (\boldsymbol{b} \cdot \boldsymbol{x})$. As in HB$^+$, the tag is authenticated if the

[2] Thus the function h involves 1152 expanded key bits in overall, namely 768 $\boldsymbol{Z}$ bits and 384 quadratic functions of the $\boldsymbol{Z}$ bits.

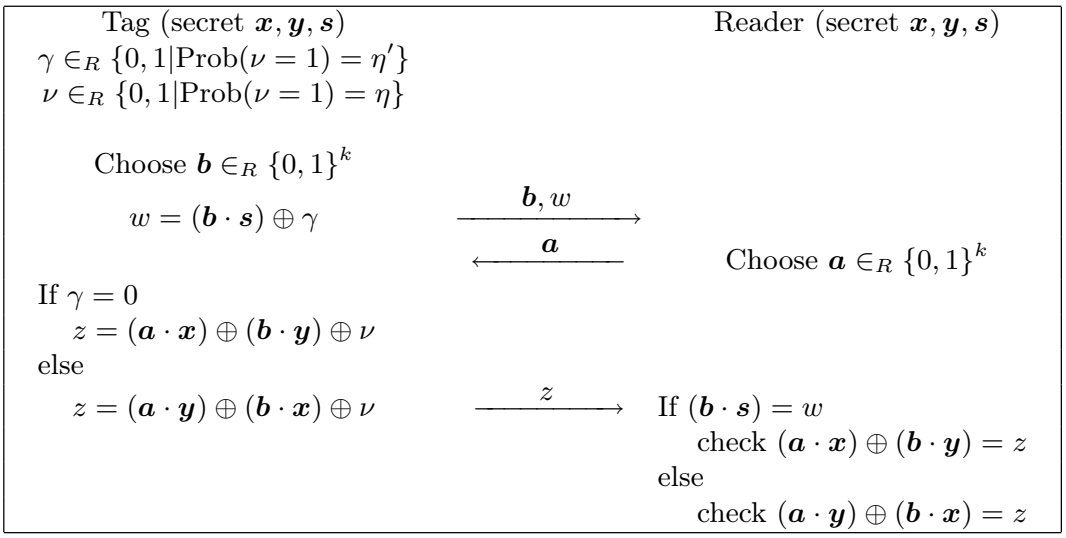


Fig. 4. One single round of HB*. The entire authentication process requires r rounds and, in this basic form, each round consists of the three passes shown. Provided the tag fails less than some threshold t number of rounds, the tag is authenticated.

number of errors is less than some threshold $t = ur$, where $u \in]\eta, \frac{1}{2}[$. Note that the false rejection and false acceptance probabilities P_{FR} and P_{FA} are given by the same formulas as in the case of HB^+. In particular these probabilities are independent of η'. The on-tag computation is roughly twice that of HB^+ (but less than that required in HB^{++}) while resistance to the GRS attack is claimed. In the next section we apply the GRS attack to HB* and show that HB* offers no advantage over HB^+.

Attacking HB*. We show that HB* remains vulnerable to an extremely close variant of the GRS attack. The first phase of the attack aims to gather information on $\boldsymbol{\delta} \cdot \boldsymbol{x}$ and $\boldsymbol{\delta} \cdot \boldsymbol{y}$ for independent vectors $\boldsymbol{\delta}$. For this, the adversary proceeds exactly as in the GRS attack and modifies the challenges sent by the reader by adding a vector $\boldsymbol{\delta}$ to $\boldsymbol{a}$. When $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ and $\boldsymbol{\delta} \cdot \boldsymbol{y} = 0$, the protocol is undisturbed and the tag will be authenticated with high probability. In all other cases, the authentication will be less likely to succeed, so that the output of the reader gives information about $\boldsymbol{x}$ and $\boldsymbol{y}$. More precisely, depending on the values of $\boldsymbol{\delta} \cdot \boldsymbol{x}$ and $\boldsymbol{\delta} \cdot \boldsymbol{y}$, each round of the protocol will be successful or not with the following probabilities:

1. *if* $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ *and* $\boldsymbol{\delta} \cdot \boldsymbol{y} = 0$, then none of the r rounds of the protocol are disturbed. The response of the tag is incorrect each time $\nu = 1$, hence with probability $\tau_1 = \eta$ and the reader accepts with probability $1 - P_{\text{FR}}$ and rejects with probability P_{FR}.
2. *if* $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ *and* $\boldsymbol{\delta} \cdot \boldsymbol{y} = 1$, the response of the tag is incorrect each time $(\gamma = 0, \nu = 1)$ or $(\gamma = 1, \nu = 0)$, hence with probability

$$\tau_2 = (1 - \eta)\eta' + (1 - \eta')\eta = \eta + (1 - 2\eta)\eta' > \eta.$$

3. *if* $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$ *and* $\boldsymbol{\delta} \cdot \boldsymbol{y} = 0$, the response of the tag is incorrect each time $(\gamma = 0, \nu = 0)$ or $(\gamma = 1, \nu = 1)$, hence with probability

$$\tau_3 = (1-\eta)(1-\eta') + \eta\eta' = \eta + (1-2\eta)(1-\eta') > \eta.$$

4. *if* $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$ *and* $\boldsymbol{\delta} \cdot \boldsymbol{y} = 1$, the response of the tag is incorrect each time $\nu = 0$, whatever γ, hence with probability $\tau_4 = 1 - \eta = \eta + (1-2\eta) > \eta$.

Note that $\tau_1 < \tau_2 \leq \frac{1}{2} \leq \tau_3 < \tau_4$. Note also that when $\eta' \to 0$ ($\eta' = 0$ corresponds to the classical HB^+ protocol), $\tau_2 \to \tau_1$ and $\tau_3 \to \tau_4$, whereas when $\eta' \to \frac{1}{2}$, $\tau_2 \to \frac{1}{2}$ and $\tau_3 \to \frac{1}{2}$. In each of the cases 2, 3 and 4, the reader will reject with probability greater than P_{FR}, namely $P_i^{\mathrm{rej}} = \Pr[\mathcal{R} \text{ rejects} \mid \text{case } i] = \sum_{j=t+1}^{r} \binom{r}{j} \tau_i^j (1-\tau_i^{r-j})$.

According to the Chernoff bound (see Appendix), the adversary will be able to discriminate between case i and j as soon as $|P_i^{\mathrm{rej}} - P_j^{\mathrm{rej}}|$ is non-negligible. We have to distinguish two cases: either $\tau_2 \leq u$, or $\tau_2 > u$.

When $\tau_2 \leq u$, *i.e.* $\eta' \leq \frac{u-\eta}{1-2\eta}$, we are "almost" in the HB^+ case: the reader will accept with overwhelming probability when $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ and reject with overwhelming probability when $\boldsymbol{\delta} \cdot \boldsymbol{x} = 1$, independently of $\boldsymbol{\delta} \cdot \boldsymbol{y}$. The GRS attack applies as it is, meaning that the adversary can retrieve $\boldsymbol{x}$ with high probability in linear time. Once this is done, it can impersonate a tag by sending $(\boldsymbol{b}, \omega) = (\boldsymbol{0}, 0)$ as first message.

When $\tau_2 > u$, *i.e.* $\eta' > \frac{u-\eta}{1-2\eta}$, the attacker can only discriminate case 1 from cases 2, 3, and 4. Indeed the reader will accept with overwhelming probability when $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ and $\boldsymbol{\delta} \cdot \boldsymbol{y} = 0$, and reject with overwhelming probability in the three other cases. However this does not prevent a slight variant of the GRS attack as follows.

We assume that $\boldsymbol{x}$ and $\boldsymbol{y}$ are linearly independent. For a random $\boldsymbol{\delta}$, case 1 happens with probability $\frac{1}{4}$, so that the adversary will be able to find with $\Theta(4k)$ attempts $k-2$ independent vectors $\boldsymbol{\delta}$ such that $\boldsymbol{\delta} \cdot \boldsymbol{x} = 0$ and $\boldsymbol{\delta} \cdot \boldsymbol{y} = 0$. Put a different way, he is able to learn the two-dimensional vectorial space $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$. Let $\boldsymbol{c_1}$, $\boldsymbol{c_2}$ and $\boldsymbol{c_3}$ denote the three non-null vectors in this vectorial space. Once they are found, the adversary can directly impersonate a valid tag with probability roughly $\frac{1}{8}$ by choosing at random two vectors among $(\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3})$ (say $\boldsymbol{c_1}$ and $\boldsymbol{c_2}$), fixing two arbitrary values for $(\boldsymbol{b}, \omega)$ that he will send at each round, and then answering $(\boldsymbol{c_1} \cdot \boldsymbol{a}) \oplus (\boldsymbol{c_2} \cdot \boldsymbol{b})$ at each round. The adversary will be successfully authenticated when $(\boldsymbol{b} \cdot \boldsymbol{s} = \omega, \boldsymbol{c_1} = \boldsymbol{x}, \boldsymbol{c_2} = \boldsymbol{y})$ or $(\boldsymbol{b} \cdot \boldsymbol{s} \neq \omega, \boldsymbol{c_1} = \boldsymbol{y}, \boldsymbol{c_2} = \boldsymbol{x})$, which happens with probability $\frac{1}{8}$.

Alternatively, the adversary can do a little more work and identify from the three values $(\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3})$ the one which is equal to $\boldsymbol{x} \oplus \boldsymbol{y}$. For this, the attacker queries the honest tag with challenges $\boldsymbol{a}$ systematically equal to the blinding vector $\boldsymbol{b}$ sent by the tag. That way, the answer of the tag is always equal to $\boldsymbol{b} \cdot (\boldsymbol{x} \oplus \boldsymbol{y}) \oplus \nu$ and the attacker deduces that $\boldsymbol{x} \oplus \boldsymbol{y}$ is the value $\boldsymbol{c_i}$ such that the number of $\boldsymbol{b}$'s such that $\boldsymbol{b} \cdot \boldsymbol{c_i}$ is equal to the answer of the tag is maximal. Once this is done, the adversary knows the unordered set $\{\boldsymbol{x}, \boldsymbol{y}\}$. This is enough to impersonate the tag with probability $\frac{1}{2}$. Assume that the vector $\boldsymbol{c_3}$ has been

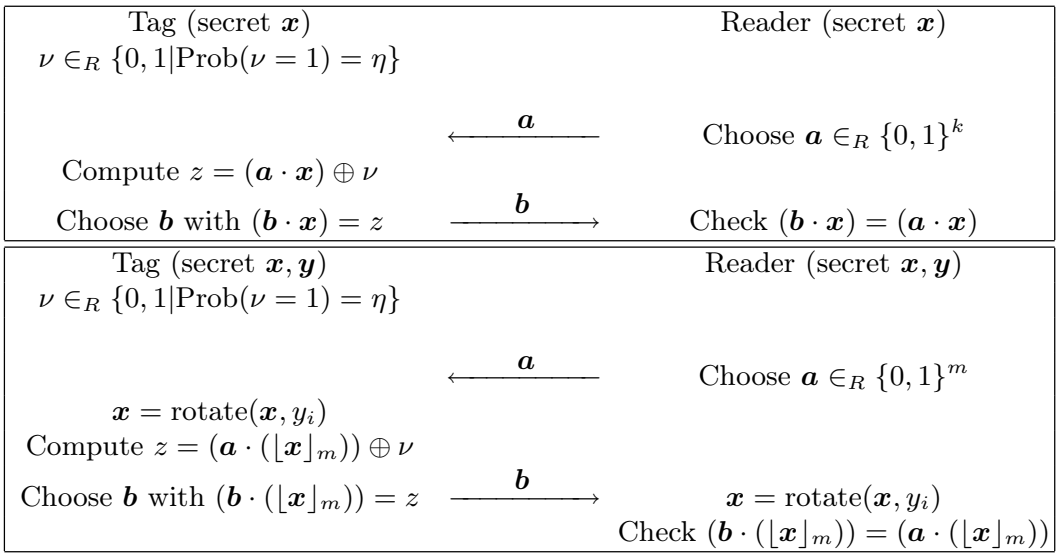


Fig. 5. Round i of HB-MP′ (above) and HB-MP (below). The entire authentication process requires r rounds and, in this basic form, each round consists of the two passes shown. Provided the tag fails less than some threshold t number of rounds, the tag is authenticated. For HB-MP $\lfloor\boldsymbol{x}\rfloor_m$ denotes the m least significant bits of $\boldsymbol{x}$ and y_i is the i^{th} bit of $\boldsymbol{y}$ which is used as the argument to a bitwise rotation.

ruled out as being $\boldsymbol{x} \oplus \boldsymbol{y}$. The adversary randomly fixes values for $(\boldsymbol{b}, \omega)$ that he will send at each round, and then answers $(\boldsymbol{c_1}\cdot\boldsymbol{a}) \oplus (\boldsymbol{c_2}\cdot\boldsymbol{b})$ at each round. The adversary will be successfully authenticated when $(\boldsymbol{b}\cdot\boldsymbol{s} = \omega, \boldsymbol{c_1} = \boldsymbol{x}, \boldsymbol{c_2} = \boldsymbol{y})$ or $(\boldsymbol{b}\cdot\boldsymbol{s} \neq \omega, \boldsymbol{c_1} = \boldsymbol{y}, \boldsymbol{c_2} = \boldsymbol{x})$, which happens now with probability $\frac{1}{2}$. Note that whatever the outcome of this first attempt, the adversary will successfully pass the following attempt with probability 1. If the first attempt succeeded he can reuse the same $(\boldsymbol{b}, \omega)$ and answer $(\boldsymbol{c_1}\cdot\boldsymbol{a}) \oplus (\boldsymbol{c_2}\cdot\boldsymbol{b})$ at each round. If the first attempt failed, use the same $(\boldsymbol{b}, \omega)$ but answer $(\boldsymbol{c_2}\cdot\boldsymbol{a}) \oplus (\boldsymbol{c_1}\cdot\boldsymbol{b})$ at each round; the answer will always be correct and the tag will be successfully impersonated.

5 The Variants HB-MP′ and HB-MP

Description of HB-MP′ and HB-MP. Another prominent protocol due to Munilla and Peinado is HB-MP [17]. In a departure from the HB^+ approach, each of the r rounds consists of only a two-pass communication between the tag and the reader. This is illustrated in Figure 5 where two variants are depicted; the first variant HB-MP′ is claimed to be resistant to chosen challenges (presumably against the tag) while the second HB-MP is claimed to resist the GRS attack.

While HB-MP′ and HB-MP are reasonably lightweight, we show in the next section that both are less secure than HB^+ since they are vulnerable to a passive attack. These are the attacks that HB^+ provably resists and so HB-MP′ and HB-MP are not good alternatives.

Table 1. Error rates and transmission costs for HB^+ and different parameter choices.

r	η	k	False reject rate	False accept rate	Transmission cost (bits) $[k = 224]$	$[k = 512]$
100	0.25	224	0.45	3×10^{-7}	44,900	102,500
80	0.25	224	0.44	4×10^{-6}	35,920	82,000
60	0.25	224	0.43	6×10^{-5}	26,984	61,500
40	0.25	224	0.42	1×10^{-3}	17,960	41,000

Attacking HB-MP′ and HB-MP. In their paper, Munilla and Peinado claim that HB-MP is immune to passive attacks, but also active and man-in-the-middle attacks of the GRS type. However, there is a very simple passive attack which enables an adversary which simply eavesdrops the r rounds of one execution of the protocol to impersonate a valid tag with probability $1 - P_{\mathrm{FR}}$.

Note that the verification done by the reader consists in checking that $(\boldsymbol{a} \oplus \boldsymbol{b}) \cdot (\lfloor \boldsymbol{x} \rfloor_m) = 0$. This equation is always verified when $\boldsymbol{b} = \boldsymbol{a}$, so that Munilla and Peinado recommend that the reader rejects a tag as soon as it answers $\boldsymbol{a}$ in any round. However, for an adversary which has eavesdropped the r rounds of a previous execution of the protocol, it is easy to compute a vector $\boldsymbol{b}$ different from $\boldsymbol{a}$ and such that $(\boldsymbol{a} \oplus \boldsymbol{b}) \cdot (\lfloor \boldsymbol{x} \rfloor_m) = 0$ with high probability as follows.

The adversary simply records the r pairs $(\boldsymbol{a_i}, \boldsymbol{b_i})$ which are exchanged between the honest tag and the honest reader. Then we know that with probability $(1 - \eta)$, $(\boldsymbol{a_i} \oplus \boldsymbol{b_i}) \cdot (\lfloor \boldsymbol{x} \rfloor_m) = 0$. Hence, for any other challenge $\boldsymbol{a'_i}$, the answer $\boldsymbol{b'_i} = \boldsymbol{a'_i} \oplus \boldsymbol{a_i} \oplus \boldsymbol{b_i}$ is different from $\boldsymbol{a'_i}$ (because $\boldsymbol{b_i} \neq \boldsymbol{a_i}$) and $(\boldsymbol{a'_i} \oplus \boldsymbol{b'_i}) \cdot (\lfloor \boldsymbol{x} \rfloor_m) = (\boldsymbol{a_i} \oplus \boldsymbol{b_i}) \cdot (\lfloor \boldsymbol{x} \rfloor_m)$. Hence the adversary is authenticated as soon as the tag was authenticated in the eavesdropped execution of the protocol. The attack works exactly in the same way against HB-MP′.

6 Discussion and Implications

The computational challenges posed by low-cost RFID tags have generated many cryptographic proposals which rely exclusively on the simplest (typically bitwise) operations. While some might express the view that some security is better than no security, even claims for "some security" need to be verified. Weaknesses in some of the simpler RFID protocols has already been demonstrated before, *e.g.* [4], and will undoubtedly be demonstrated in the future.

Those working in the field of RFID security are correct when claiming that one doesn't necessarily need full security for a deployment. This is why a proposal like HB^+ is actually rather successful: it doesn't claim to protect against all adversaries, but for adversaries with a minimum technical capability it provides a reasonable level of security. HB^+ does as claims and no more. The variants described in this note have attempted to do more and have, arguably, delivered less. It is difficult to do a lot with such basic operations.

This is not to say, however, that HB^+ is currently ideal. While the on-tag computation is low, the GRS attack may be practically important to some (*i.e.* it might be more than certificational). Furthermore, the communication overheads for HB^+ are substantial while the false acceptance and false rejection rates are not suitable for deployment. These are shown in Table 1 for the parameter $k = 224$ and acceptance threshold $r\eta$ proposed in HB^+ [11]. Based on the work of [15] we also consider the data transmission costs when $k = 512$ which is a more appropriate value to use if we are seeking 80-bit security.

These are unfortunate barriers for any practical deployment of HB^+. Nevertheless, the computational complexity and simplicity of HB^+ are very attractive and it nicely complements other work that seeks to extend more conventional forms of cryptography [1,6,10,16]. It is therefore an interesting challenge to find the right variant of HB^+ that simultaneously improves both security and efficiency: one such proposal has been named $\text{HB}^\#$ by the authors [9].

7 Conclusions

In this paper we have considered variants to HB^+. While they were designed with the sole intention of resisting the GRS attack on HB^+, all of HB^{++}, HB^*, HB-MP′, and HB-MP are vulnerable to GRS-style attacks. In addition these variants sacrifice much of the simplicity and elegance of the original HB^+. Despite some questions on the practical implementation of HB^+ and the existence of the GRS attack, the computational efficiency and theoretical foundations of HB^+ are impressive. And while the work in this paper suggests that good variants to HB^+ are very hard to find, the right variant might offer a particularly interesting—and successful—solution to the problem of low-cost tag authentication.

Acknowledgements. We would like to thank Stanislaw Jarecki for his thoughtful feedback on a previous version of this paper.

References

1. A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In P. Paillier and I. Verbauwhede, editors, *Proceedings of CHES 2007*, Lecture Notes in Computer Science, vol. 4727, 450–466, Springer, 2007.
2. J. Black, S. Halevi, H. Krawczyk, T. Krovetz and P. Rogaway. UMAC: Fast and Secure Message Authentication. In M. J. Wiener, editor, *Proceedings of CRYPTO '99*, Lecture Notes in Computer Science, vol. 1666, 216–233, Springer, 1999.
3. J. Bringer, H. Chabanne, and E. Dottax. HB^{++}: A Lightweight Authentication Protocol Secure Against Some Attacks. In P. Georgiadis, J. Lopez, S. Gritzalis, and G. Marias, editors, *Proceedings of SecPerU 2006*, 28–33, IEEE Computer Society Press, 2006.
4. B. Defend, K. Fu, and A. Juels. Cryptanalysis of Two Lightweight RFID Authentication Schemes. In *International Workshop on Pervasive Computing and Communication Security, PerSec 2007*, 211–216, IEEE Computer Society Press, 2007.

5. D.N. Duc and K. Kim. Securing HB$^+$ Against GRS Man-in-the-Middle Attack. In *Institute of Electronics, Information and Communication Engineers, Symposium on Cryptography and Information Security*, Jan. 23–26, 2007.
6. M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong Authentication for RFID Systems Using the AES Algorithm. In M. Joye and J.-J. Quisquater, editors, *Proceedings of CHES 2004*, Lecture Notes in Computer Science, vol. 3156, 357–370, Springer, 2004.
7. M.P.C. Fossorier, M.J. Mihaljevic, H. Imai, Y. Cui, and K. Matsuura. A Novel Algorithm for Solving the LPN Problem and its Application to Security Evaluation of the HB Protocol for RFID Authentication. Available from `http://eprint.iacr.org/2006/197.pdf`.
8. H. Gilbert, M.J.B. Robshaw, and H. Sibert. An Active Attack Against HB$^+$: A Provably Secure Lightweight Authentication Protocol. *IEE Electronics Letters*, vol. 41, number 21, 1169–1170, 2005.
9. H. Gilbert, M.J.B. Robshaw, and Y. Seurin. HB$^\#$: Increasing the Security and Efficiency of HB$^+$. In submission.
10. M. Girault, G. Poupard and J. Stern. On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order. *Journal of Cryptology*, vol. 19, number 4, 463–488, 2006.
11. A. Juels and S.A. Weis. Authenticating Pervasive Devices With Human Protocols. In V. Shoup, editor, *Advances in Cryptology - Crypto 05*, Lecture Notes in Computer Science, vol. 3126, 293–198, Springer, 2005.
12. J.-P. Kaps, K. Yüksel and B. Sunar. Energy Scalable Universal Hashing. *IEEE Trans. on Computers*, vol. 54, number 12, 1484–1495, 2005.
13. J. Katz and J. Shin. Parallel and Concurrent Security of the HB and HB$^+$ Protocols. In S. Vaudenay, editor, *Advances in Cryptology - Eurocrypt 2006*, Lecture Notes in Computer Science, vol. 4004, 73–87, Springer, 2006.
14. J. Katz and A. Smith. Analysing the HB and HB$^+$ Protocols in the "Large Error" Case. Available from `http://eprint.iacr.org/2006/326.pdf`.
15. E. Levieil and P.-A. Fouque. An Improved LPN Algorithm. In R. De Prisco and M. Yung, editors, *Proceedings of SCN 2006*, Lecture Notes in Computer Science, vol. 4116, 348–359, Springer, 2006.
16. M. McLoone and M.J.B. Robshaw. Public Key Cryptography and RFID. In M. Abe, editor, *CT-RSA 2007*, Lecture Notes in Computer Science, vol. 4377, 372–384, Springer, 2007.
17. J. Munilla and A. Peinado. HB-MP: A Further Step in the HB-family of Lightweight Authentication Protocols. *Computer Networks*, vol. 51, 2262–2267, 2007.
18. S. Piramuthu. HB and Related Lightweight Authentication Protocols for Secure RFID Tag/Reader Authentication. *CollECTeR Europe Conference*, June 2006.
19. R.L. Rivest. The RC5 Encryption Algorithm. In B. Preneel, editor, *Proceedings of FSE 1994*, Lecture Notes in Computer Science, vol. 1008, 86–96, Springer, 1995.

A Chernoff Bounds

We recall here the classical Chernoff bounds. Let $X_1, \ldots, X_n$ be independent Poisson trials such that $\Pr[X_i = 1] = p_i$. Let $X = \sum_{i=1}^{n} X_i$ and μ be the expected value of X. Then for any $t < \mu$ and $t' > \mu$,

$$\Pr[X \leq t] \leq e^{-\frac{(\mu - t)^2}{2\mu}} \quad \text{and} \quad \Pr[X \geq t'] \leq e^{-\frac{(t' - \mu)^2}{3\mu}}.$$

HB$^{\#}$: Increasing the Security and Efficiency of HB^{+}

Henri Gilbert, Matthew J.B. Robshaw, and Yannick Seurin

Orange Labs, 38–40 rue General Leclerc, Issy les Moulineaux, France
{henri.gilbert,matt.robshaw,yannick.seurin}@orange-ftgroup.com

Abstract. The innovative HB^{+} protocol of Juels and Weis [10] extends device authentication to low-cost RFID tags. However, despite the very simple on-tag computation there remain some practical problems with HB^{+} and despite an elegant proof of security against some limited active attacks, there is a simple man-in-the-middle attack due to Gilbert *et al.* [8]. In this paper we consider improvements to HB^{+} in terms of both security and practicality. We introduce a new protocol that we denote RANDOM-HB$^{\#}$. This proposal avoids many practical drawbacks of HB^{+}, remains provably resistant to attacks in the model of Juels and Weis, and at the same time is provably resistant to a broader class of active attacks that includes the attack of [8]. We then describe an enhanced variant called HB$^{\#}$ which offers practical advantages over HB^{+}.

Key words: HB^{+}, RFID tags, authentication, LPN, Toeplitz matrix.

1 Introduction

The deployment of low-cost RFID tags is gathering pace. One familiar application is the inventory tracking of consumer items such as clothes, media products, and pharmaceuticals. However since blank tags can be programmed, there are opportunities for an attacker to clone an RFID tag and to introduce counterfeit goods into the supply chain. Thus, in this and other application areas there is much interest in deploying mechanisms for cryptographic tag authentication. However the physical demands for the deployment of cryptography on a cheap tag are substantial. Not only is space limited [10], but the peak and average power consumption often pose a demanding barrier for a tag that derives its power from a reader. Furthermore, since RFID tags pass fleetingly past a reader and are used in multi-tag and multi-reader environments, the communication is limited and its coordination complex.

Juels and Weis introduced HB^{+}, a three-pass symmetric key authentication protocol, at Crypto 2005 [10]. HB^{+} is computationally lightweight—requiring only simple bit-wise operations—and it is supported by a proof of security [10]. There are, however, some practical deficiencies in HB^{+} and the value of the proof of security has been somewhat limited by a simple active attack due to Gilbert *et al.* [8] which we will refer to as the GRS attack. Nevertheless, the simplicity

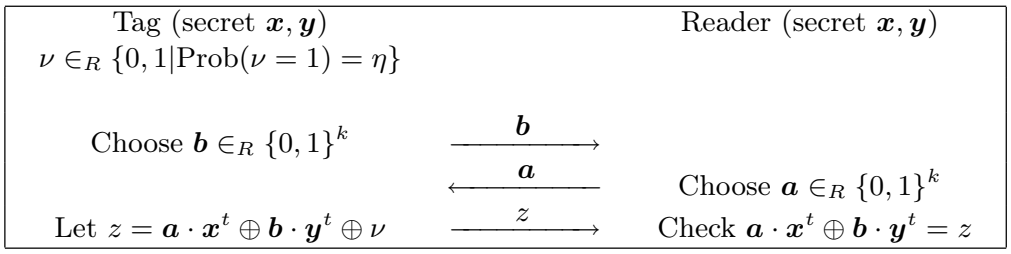


Fig. 1. One single round of HB$^+$ [10]. The entire authentication process requires r rounds and, in this basic form, each round consists of the three passes shown. Provided the tag fails less than some threshold t number of rounds, the tag is authenticated.

of both the original proposal and the active attack have led to a number of HB-related publications (see Section 2.2).

In this paper we propose solutions that improve on the practical problems of HB$^+$ while providing resistance to the GRS attack. The two simple proposals RANDOM-HB$^\#$ and HB$^\#$ provide more practical error rates than the original HB$^+$ and reduce the communication payload by a factor of around 20 (depending on the parameter sets). The protocol RANDOM-HB$^\#$ is provably secure in the *detection-based* model, the adversarial model used in *all* current proofs of security for HB$^+$ and its variants. But RANDOM-HB$^\#$ is also provably secure against the GRS attack and more generally in what we term the GRS-MIM model, an adversarial model that permits an active adversary to manipulate messages from the reader. The related protocol HB$^\#$ then gives a truly efficient scheme. While the same proofs do not immediately extend in their entirety to HB$^\#$, we can still say a surprising amount about the scheme in both theory and practice.

Our paper is organised as follows. First we describe HB$^+$ and some variants. Then, in Section 3, we introduce RANDOM-HB$^\#$ and provide full security proofs. In Section 4 we describe HB$^\#$ and its security and practical performance. We then highlight future work and draw our conclusions. Throughout we aim to use established notation. There will be some interplay between vectors $\boldsymbol{x} \in \{0,1\}^k$ (which we always consider to be row vectors) and scalars in GF(2). We use bold type $\boldsymbol{x}$ to indicate a row vector while scalars x are written in normal text. The bitwise addition of two vectors will be denoted $\oplus$ just as for scalars. We denote the *Hamming weight* of $\boldsymbol{x}$ by $\text{Hwt}(\boldsymbol{x})$.

2 HB$^+$ Variants and Tag Authentication

There are now several protocols based on HB$^+$ and these offer a variable level of security and practicality. We start by reviewing the original protocol. HB$^+$ is a three-pass authentication protocol built on the conjectured hardness of the *Learning from Parity with Noise* (LPN) problem [10].

LPN Problem. Let A be a random $(q \times k)$-binary matrix, let $\boldsymbol{x}$ be a random k-bit vector, let $\eta \in]0, \frac{1}{2}[$ be a noise parameter, and let $\boldsymbol{\nu}$ be

a random q-bit vector such that $\mathrm{Hwt}(\boldsymbol{\nu}) \leq \eta q$. Given A, η, and $\boldsymbol{z} = A \cdot \boldsymbol{x}^t \oplus \boldsymbol{\nu}^t$, find a k-bit vector $\boldsymbol{y}^t$ such that $\mathrm{Hwt}(A \cdot \boldsymbol{y}^t \oplus \boldsymbol{z}) \leq \eta q$.

The HB$^+$ protocol is outlined in Figure 1. One doesn't need to look long to see that the goal of low on-tag computation has been achieved. Leaving aside generating $\boldsymbol{b}$ and the bit ν, computation on the tag is reduced to a dot-product (which can be computed bit-wise) and a single bit exclusive-or. Also HB$^+$ is accompanied by a proof of security. The adversarial model for this proof is referred to as the *detection-based* model [10] and requires that the adversary queries a tag q times and then attempts to pass the HB$^+$ authentication process by interacting with the reader once. Some commentators are not convinced that this adversarial model is sufficiently strong and an active attack against HB$^+$ exists when the adversary can interact with both the tag and the reader before attempting to impersonate the tag [8]. That said, the proof of security still has considerable value. The original proof [10] was rather sophisticated and applied to an adversary attempting to fool the reader over a single round of HB$^+$. This was extended by Katz and Shin [12] who also considered the parallel version of HB$^+$ with communications batched into one round of a three-pass protocol.

2.1 Some problems with HB$^+$

While HB$^+$ is computationally lightweight it still has some practical defects. The possibility of a legitimate tag being rejected has been commented on [12], but other issues such as the complex and extensive tag-reader communication would make HB$^+$ difficult to use. First, however, we highlight the fact that methods to solve the LPN problem have improved since the original presentation of HB$^+$.

LPN security and parameter choices. When considering the security and implementation of HB$^+$ there are four parameters that we need to set:

k :	the length of the secrets,	η :	the noise level,
r :	the number of rounds,	t :	the threshold for tag acceptance.

The first two parameters, k and η, quantify the resistance of the underlying LPN problem to attack. In [11] it is suggested that the parameter sets $k = 224$ and $\eta = 0.25$ provide around 80-bit security. Katz and Shin [12] propose $k \approx 200$ with $\eta = 0.125$, but we note that the reduced level of noise means that the LPN problem instance becomes easier and would necessitate an increase[1] to k.

Since the publication of HB$^+$ the LPN problem has been studied in more detail and the BKW algorithm cited in [10,12] has been improved. Fossorier *et al.* [6] show that the parameter choices used by [10] offer a level of security no greater than 2^{61} operations rather than the 2^{80} claimed. However, this has been superseded by the work of Levieil and Fouque [16] which suggests that the real security level offered by the parameters in [10] is no more than 2^{52} operations.

[1] However [12] is concerned with security proofs and specific parameter choices are somewhat orthogonal to their work.

Table 1. Error rates and transmission costs for different parameter sets in HB$^+$. The threshold $t = r\eta$ is proposed in [10] so we use $\lceil r\eta \rceil$ in this table. For the other parameters, [10] suggest $k = 224$ and $\eta = 0.25$ (leaving r unspecified) while [12] suggests $k \approx 200$, $\eta = 0.125$, with $40 \leq r \leq 50$. Based on the work of [16], we also consider the data transmission costs when $k = 512$ in the last column.

r	η	k	False reject rate (P_{FR})	False accept rate (P_{FA})	Transmission cost (bits) [k as given]	Transmission cost (bits) [$k = 512$]
80	0.25	224	0.44	4×10^{-6}	35,920	82,000
60	0.25	224	0.43	6×10^{-5}	26,984	61,500
40	0.25	224	0.42	1×10^{-3}	17,960	41,000
50	0.125	200	0.44	2×10^{-8}	20,050	51,250
40	0.125	200	0.38	7×10^{-9}	16,040	41,000

Considering [16] we propose alternative parameter values in Section 4.2 that are more consistent with the intended security level. In particular we propose $k = 512$ and $\eta = 0.125$ or, more conservatively, $k = 512$ and $\eta = 0.25$.

Error rates. A false rejection, a legitimate tag being rejected by a legitimate reader, occurs when the number of incorrect authentications exceeds the threshold t. A false acceptance takes place when an illegitimate tag is accepted by a legitimate reader. This occurs when t or fewer verification errors take place and we assume the illegitimate tag is reduced to guessing the reply z at random. The probability of a false rejection, P_{FR}, and a false acceptance, P_{FA}, are given by

$$P_{\mathrm{FR}} = \sum_{i=t+1}^{r} \binom{r}{i} \eta^i (1-\eta)^{r-i} \text{ and } P_{\mathrm{FA}} = \sum_{i=0}^{t} \binom{r}{i} 2^{-r}.$$

Note that both the false rejection and acceptance rate are independent of k, the size of the secrets, while the false acceptance rate is also independent of the noise level η used in HB$^+$. In the original descriptions of HB$^+$ a threshold of $t = r\eta$ is suggested. However (see Table 1) such a choice gives an unacceptably high false rejection rate. It is hard to imagine any practical scenario where a probability higher than 1% of rejecting a legitimate tag could be tolerated.

Transmission costs. HB$^+$ is a three-pass protocol that runs over r rounds. This requires the exchange of $2k + 1$ bits per round and $2rk + r$ bits in total. In the parallel version of the protocol, the data transmission requirements are the same but the data is packed into three passes of rk, rk, and r bits respectively. A three-pass protocol is considerably more practical than a $3r$-pass protocol (this was also mentioned in [12] as a justification for parallel HB$^+$). However the total amount of data transferred in both cases remains unacceptably high. In Table 1 we provide some estimates for the transmission costs in using HB$^+$. In particular we use parameter values that cover those proposed in [10,12]. We

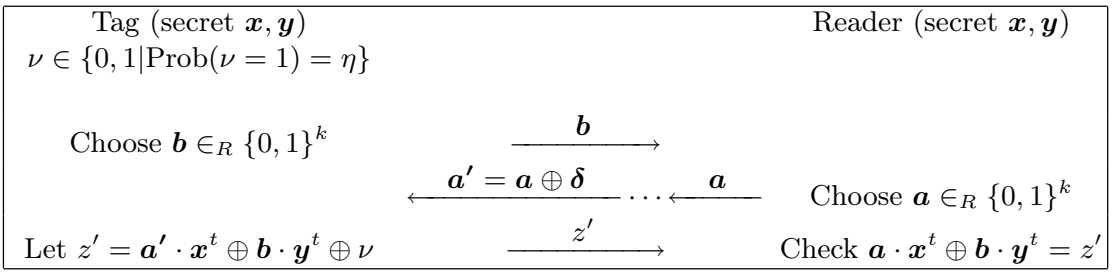


Fig. 2. The attack of Gilbert *et al.* [8] on HB$^+$. The adversary modifies the communications between reader and tag (by adding some perturbation $\boldsymbol{\delta}$) and notes whether authentication is still successful. This reveals one bit of secret information.

also include the transmission costs if we were to use parameter sizes that come closer to providing the intended 80-bit level of security.

An active attack. A simple active attack on HB$^+$ was provided in [8]. There it is assumed that an adversary can manipulate challenges sent by a legitimate reader to a legitimate tag during the authentication exchange, and can learn whether such manipulation gives an authentication failure. The attack consists of choosing a constant k-bit vector $\boldsymbol{\delta}$ and using it to perturb the challenges sent by a legitimate reader to the tag; $\boldsymbol{\delta}$ is exclusive-or'ed to each authentication challenge for each of the r rounds of authentication. If the authentication process is successful then we must have that $\boldsymbol{\delta} \cdot \boldsymbol{x}^t = 0$ with overwhelming probability. Otherwise $\boldsymbol{\delta} \cdot \boldsymbol{x}^t = 1$ with overwhelming probability and acceptance or rejection by the reader reveals one bit of secret information. The attack is illustrated in Figure 2 for one round of the HB$^+$ protocol. To retrieve the k-bit secret $\boldsymbol{x}$, one can repeat the attack k times for linearly independent $\boldsymbol{\delta}$'s and solve the resulting system. Conveniently, an adversary can choose $\boldsymbol{\delta}$'s with a single non-zero bit. With $\boldsymbol{x}$ an attacker can impersonate the tag by setting $\boldsymbol{b} = \boldsymbol{0}$. Alternatively, an attacker can emulate a false tag using $\boldsymbol{x}$, send a chosen blinding factor $\boldsymbol{b}$ to a legitimate reader, and return $\boldsymbol{a} \cdot \boldsymbol{x}^t$ to the challenge $\boldsymbol{a}$. If authentication is successful $\boldsymbol{b} \cdot \boldsymbol{y}^t = 0$, otherwise $\boldsymbol{b} \cdot \boldsymbol{y}^t = 1$, with overwhelming probability, and $\boldsymbol{y}$ can be recovered with k linearly independent $\boldsymbol{b}$.

Whether or not the attack is technically easy to mount it is *certificational*. The attack is mathematically simple and fully compromises HB$^+$. Protocols that resist this attack, while maintaining the computational simplicity of HB$^+$, would therefore be very attractive.

2.2 Other work on HB$^+$ and tag authentication

The novelty of the HB$^+$ protocol has generated considerable interest and much research. We have already mentioned the work of Katz and Shin [12] that closed gaps and extended the original proof of security. Follow-on work by Katz and Smith [13] has further extended these theoretical results to a larger range of noise levels $\frac{1}{4} \leq \eta < \frac{1}{2}$ whereas previous work [12] was only valid for $\eta < \frac{1}{4}$.

Other researchers have considered the active attack of Gilbert *et al.* [8]. Among them Bringer *et al.* [2] have outlined a protocol named HB^{++}. However the resulting protocol has some practical drawbacks. The data transmission costs of HB^{+} remain and the on-tag computation now includes bit-wise rotations and a small-block permutation f. Furthermore, an additional pre-protocol involving a universal hash function h is required to derive new tag/reader secrets at the start of each authentication. All this requires additional hardware and moves away from the essential simplicity of the HB^{+} protocol. Piramuthu [20] proposes a modification to HB^{++} in which the bit-wise rotations are varied for each round of the authentication and the message flow is simplified (saving one bit per authentication round). However the exact security claims are unclear. The variant HB* is proposed by Duc and Kim [4] while another prominent protocol is HB-MP [19]. While both claim to be resistant to the attacks of [8], linear time attacks by the authors [7] show that this is not the case.

Naturally, research into other mechanisms for unilateral and mutual authentication continue in parallel. Schemes based on symmetric cryptography might use a lightweight block cipher [1,21] in a challenge-response protocol while other schemes might use asymmetric techniques such as GPS [9,18]. Other proposals include SQUASH [22] which might be viewed as a dedicated MAC, though the security goals appear to be somewhat reduced when compared to HB^{+} and the proposals RANDOM-HB$^{\#}$ and HB$^{\#}$ in this paper.

But this parallel work only serves to emphasize the interest in tag authentication and the importance of understanding the limits of proposals like HB^{+}. Despite the mixed success of current proposals in the literature, HB^{+} still holds much promise. This is due to the exceptionally low on-tag computational requirements and the fact that a proof of security, even if the model is weaker than we might ideally like, is a positive attribute.

3 The Proposal RANDOM-HB$^{\#}$

We now introduce RANDOM-HB$^{\#}$ (RANDOM-*HB-sharp*). This goes a long way to fixing many of the practical problems of HB^{+}. Like many other HB^{+}-variants, we prove the security of RANDOM-HB$^{\#}$ in the *detection-based* model, referred to in what follows as the DET-model. But we go further and prove the security of RANDOM-HB$^{\#}$ against a class of attacks that includes the GRS attack in what we term the GRS-MIM-model. More details are given in Section 3.1, but this model allows an active attacker to change any message from the reader in any way that they wish and observe the decision of the reader of whether to accept or not.

In RANDOM-HB$^{\#}$ we generalise HB^{+} and change the form of the secrets $\boldsymbol{x}$ and $\boldsymbol{y}$ from k-bit vectors into $(k_X \times m)$- and $(k_Y \times m)$-binary matrices X and Y. We illustrate RANDOM-HB$^{\#}$ protocol in Figure 3. One way of looking at RANDOM-HB$^{\#}$ is to observe that it is equivalent to m iterations of HB^{+}, but each column of X and Y in RANDOM-HB$^{\#}$ effectively represents a different HB^{+} secret $\boldsymbol{x}$ and $\boldsymbol{y}$. However, while RANDOM-HB$^{\#}$ carries much of the appearance of the HB^{+} protocol, there are important differences. In particular, the final verification by

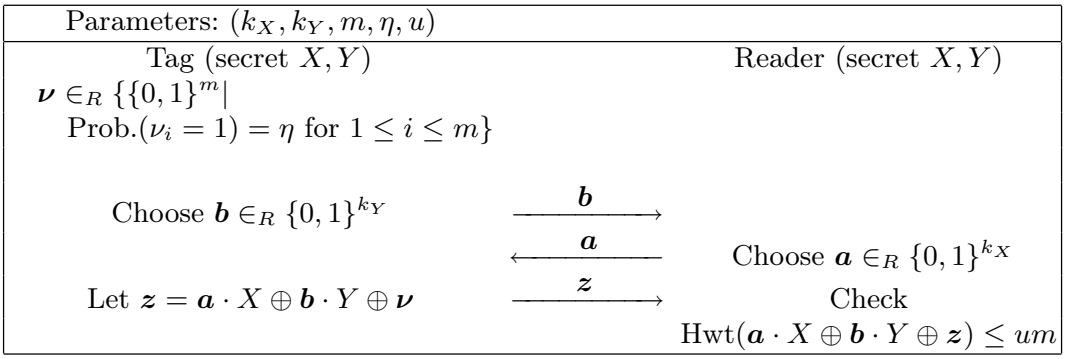


Fig. 3. The RANDOM-HB$^\#$ authentication protocol where the secrets X and Y are binary random matrices and the protocol has a single round. The verification step requires the comparison of two vectors and yields a PASS/FAIL verdict.

the reader consists of the comparison of two m-bit vectors $\boldsymbol{a} \cdot X \oplus \boldsymbol{b} \cdot Y$ and $\boldsymbol{z}$. For reader-verification we merely count the number of positions e that are in error and if $e \leq t$ for some threshold $t = um$, where $u \in]\eta, \frac{1}{2}[$, then we deduce that the tag is authentic. Thus RANDOM-HB$^\#$ and HB$^\#$ (see Section 4) consist of a single round.

3.1 Security results for RANDOM-HB$^\#$

We now provide security proofs for RANDOM-HB$^\#$ in two models. The first is the DET-model used in much of the founding work on HB$^+$ [10,12]. Here the adversary is only allowed to query an honest tag without access to the reader. The second permits an active attacker to manipulate messages sent by the reader and will be referred to as the GRS-MIM-model.

Security definitions. In the following, the security parameter will be k, to which the number of rows of the secret matrices X and Y are related by $k_X = \Theta(k)$ and $k_Y = \Theta(k)$. We will say that a function (from positive integers to positive real numbers) is *negligible* if it approaches zero faster than any inverse polynomial, and *noticeable* if it is larger than some inverse polynomial. An algorithm will be *efficient* if it is a *Probabilistic Polynomial-Time* Turing machine. By saying that LPN is a hard problem, we mean that any efficient adversary solves it with only negligible probability.

We will let $\mathcal{T}_{X,Y,\eta}$ denote the algorithm run by an honest tag in the RANDOM-HB$^\#$ protocol and $\mathcal{R}_{X,Y,u}$ the algorithm run by the tag reader. We will prove the security of RANDOM-HB$^\#$ in two models:

- The DET-model, defined in [10,12], where attacks are carried out in two phases: the adversary first interacts q times with the honest tag. Then the adversary interacts with the reader and tries to impersonate the valid tag.

– The GRS-MIM-model: in a first phase, the adversary can eavesdrop on all communications between an honest tag and an honest reader (including the reader-decision of whether to accept or not) and in addition the attacker can modify any message from the reader to the tag for q executions of the protocol. Then the adversary interacts only with the reader and tries to impersonate the valid tag.

Note that the DET-model is a restriction of the GRS-MIM-model as any attack in the DET-model can easily be converted into an attack in the GRS-MIM-model. By replying at random to a challenge, the probability an adversary impersonating a tag will succeed is the false acceptance rate $P_{\mathrm{FA}} = 2^{-m} \sum_{i=0}^{um} \binom{m}{i}$. This quantity is the best soundness we can achieve for RANDOM-HB$^\#$. Note that it is a function of m and u and not of the security parameter k, which will only set how close to P_{FA} the advantage of an adversary is bound to be. Note also that P_{FA} is negligible for any $u \in]\eta, \frac{1}{2}[$ and any $m = \Theta(k)$. We define the advantage of an adversary against the RANDOM-HB$^\#$ protocol in the DET and GRS-MIM models as its overhead success probability over P_{FA} in impersonating the tag:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{DET}}(k_X, k_Y, m, \eta, u, q) \stackrel{\mathrm{def}}{=} \Pr\left[X \stackrel{\$}{\leftarrow} \mathcal{M}_X, Y \stackrel{\$}{\leftarrow} \mathcal{M}_Y,\, \mathcal{A}^{\mathcal{T}_{X,Y,\eta}}(1^k) \,:\, \langle \mathcal{A}, \mathcal{R}_{X,Y,u} \rangle = \mathrm{ACC}\right] - P_{\mathrm{FA}};$$

$$\mathrm{Adv}_{\mathcal{A}}^{\text{GRS-MIM}}(k_X, k_Y, m, \eta, u, q) \stackrel{\mathrm{def}}{=} \Pr\left[X \stackrel{\$}{\leftarrow} \mathcal{M}_X, Y \stackrel{\$}{\leftarrow} \mathcal{M}_Y,\, \mathcal{A}^{\mathcal{T}_{X,Y,\eta}, \mathcal{R}_{X,Y,u}}(1^k) \,:\, \langle \mathcal{A}, \mathcal{R}_{X,Y,u} \rangle = \mathrm{ACC}\right] - P_{\mathrm{FA}}.$$

where $\mathcal{M}_X$ and $\mathcal{M}_Y$ denote resp. the sets of $(k_X \times m)$- and $(k_Y \times m)$-binary matrices and ACC denotes "accept".

Proof methods. We do not reduce the security of RANDOM-HB$^\#$ directly to the LPN problem. A preliminary step of our analysis is to define a natural matrix-based extension of the LPN problem and to prove its hardness. For this we appeal to the theory of "weakly verifiable puzzles". This is a notion introduced by Canetti, Halevi, and Steiner [3] and, informally, refers to a situation where only the entity that generates the puzzle holds secret information enabling the correctness of a candidate solution to be efficiently verified. As noticed by Katz and Shin [12], attacking the one-round HB protocol [10] in the passive model (that is, given q noisy samples $(\boldsymbol{a_i}, \boldsymbol{a_i} \cdot \boldsymbol{x}^t \oplus \nu_i)$, where $\boldsymbol{x}$ is a secret k-bit vector and the $\boldsymbol{a_i}$ are random k-bit vectors, and a random challenge $\boldsymbol{a}$, guess $\boldsymbol{a} \cdot \boldsymbol{x}^t$) may be viewed as a weakly verifiable puzzle. The result by Juels and Weis [10, Lemma 1] asserts, in essence, that this puzzle is $(1 - \frac{1}{2})$-hard if we assume the hardness of the LPN problem, which means that any efficient adversary trying to solve it has a success probability that is negligibly close (in k) to $\frac{1}{2}$. Canetti *et al.* [3] proved that if no efficient algorithm can solve a puzzle with probability more than ϵ, then no efficient algorithm can solve m independent puzzles simultaneously with probability more than ϵ^m. Thus, we define an extension of the HB puzzle that

we call the *MHB puzzle*: given q noisy samples $(\boldsymbol{a_i}, \boldsymbol{a_i} \cdot X \oplus \boldsymbol{\nu_i})$, where X is a secret $(k \times m)$-matrix and the $\boldsymbol{a_i}$ are random k-bit vectors, and a random challenge $\boldsymbol{a}$, guess $\boldsymbol{a} \cdot X$. Using Canetti *et al.*'s result, we prove that any efficient adversary trying to solve it has a success probability that is negligibly close (in k) to $\frac{1}{2^m}$. All the necessary definitions and results are given in the full version of this paper.[2]

The security analysis is carried out in two steps. First we reduce the security of RANDOM-HB$^\#$ in the DET-model to the MHB puzzle. Then we reduce the security in the GRS-MIM-model to the security in the DET-model.

Theorem 1 (Security of RANDOM-HB$^\#$ in the DET-model). *Let $\mathcal{A}$ be an adversary attacking the RANDOM-HB$^\#$ protocol with parameters (k_X, k_Y, m, η, u) in the DET-model, interacting with the tag in at most q executions of the RANDOM-HB$^\#$ protocol, running in time T, and achieving advantage greater than δ. Then there is an adversary $\mathcal{A}'$, running in time at most $2mLq(2 + \log_2 q)T$, solving the MHB puzzle with parameters (k_Y, m, η, q'), where $q' = mLq(2 + \log_2 q)$ and $L = \frac{512}{\delta^4(1-2u)^4}(\ln m - \ln\ln 2)$, with success probability $> \left(\frac{1}{2^m} + \frac{\delta}{4}\right)$. Hence, assuming the hardness of the LPN problem, the advantage of any efficient DET-adversary against the RANDOM-HB$^\#$ protocol is negligible in k. As a consequence, for parameters $m = \Theta(k)$, the probability of any efficient DET-adversary to impersonate a valid tag is negligible in k.*

Proof. We slightly adapt the proof of Juels and Weis [11, Appendix C]. We denote by $\{(\boldsymbol{b_i}, \boldsymbol{z_i})\}_{1 \le i \le q'}$ the set of samples obtained by $\mathcal{A}'$ from the MHB puzzle generator with secret matrix Y and $\boldsymbol{b}$ the challenge vector for which $\mathcal{A}'$ aims to output $\boldsymbol{z} = \boldsymbol{b} \cdot Y$. $\mathcal{A}'$ uses its samples to simulate a tag algorithm $\mathcal{T}_{X,Y,\eta}$ where X is random with one line equal to $\boldsymbol{z}$. $\mathcal{A}'$ proceeds as follows:

1. Choose a random j, $1 \le j \le k_X$, and construct the $k_X \times m$ matrix X' where all rows are random except the j-th one which is undefined (say, equal to zero). Let $\boldsymbol{x_l}$ denote the l-th row of X'.
2. Divide the $q' = mLq(1+r)$ samples $\{(\boldsymbol{b_i}, \boldsymbol{z_i})\}_{1 \le i \le q'}$ into mL sets of $q(1+r)$ samples. For each bit position $s = 1$ to m, repeat the following L times, considering a fresh set of $q(1+r)$ samples each time:
   (a) For $i = 1$ to q repeat the following: draw a random bit α_i (this is a guess at the j-th bit of the challenge $\boldsymbol{a_i^+}$ which will be sent by the adversary $\mathcal{A}$). If $\alpha_i = 0$, send to $\mathcal{A}$ the blinding vector $\boldsymbol{b_i^+} = \boldsymbol{b_i}$, if $\alpha_i = 1$, send to $\mathcal{A}$ the blinding vector $\boldsymbol{b_i^+} = \boldsymbol{b_i} \oplus \boldsymbol{b}$. $\mathcal{A}$ sends back the challenge $\boldsymbol{a_i^+}$. If the guess was right (i.e. $\alpha_i = \boldsymbol{a_i^+}[j]$), then answer with the vector

$$\boldsymbol{z_i^+} = \bigoplus_{l \ne j} \left(\boldsymbol{a_i^+}[l] \cdot \boldsymbol{x_l}\right) \oplus \boldsymbol{z_i}.$$

   Otherwise rewind adversary $\mathcal{A}$ to the beginning of its i-th query and try with a new $(\boldsymbol{b_{i'}}, \boldsymbol{z_{i'}})$ chosen among the rq supplementary samples.

[2] Available from http://eprint.iacr.org/2008/028

(b) If the rq samples are exhausted before the simulation of the query phase of $\mathcal{A}$ ends, randomly guess $\boldsymbol{z}[s]$.
(c) Otherwise, go to the cloning phase of $\mathcal{A}$: $\mathcal{A}$ sends a blinding vector $\hat{\boldsymbol{b}}$. Choose two random challenge vectors $\hat{\boldsymbol{a}}_{\boldsymbol{1}}$ and $\hat{\boldsymbol{a}}_{\boldsymbol{2}}$ such that they differ in their j-th bit. Transmit $\hat{\boldsymbol{a}}_{\boldsymbol{1}}$ to $\mathcal{A}$, record its response $\hat{\boldsymbol{z}}_{\boldsymbol{1}}$, rewind the adversary, transmit $\hat{\boldsymbol{a}}_{\boldsymbol{2}}$ to $\mathcal{A}$, and record its response $\hat{\boldsymbol{z}}_{\boldsymbol{2}}$ as well.
(d) Compute the guess for $\boldsymbol{z}[s]$ as

$$\hat{\boldsymbol{z}}_{\boldsymbol{1}}[s] \oplus \hat{\boldsymbol{z}}_{\boldsymbol{2}}[s] \oplus \left(\bigoplus_{l \neq j} (\hat{\boldsymbol{a}}_{\boldsymbol{1}}[l] \oplus \hat{\boldsymbol{a}}_{\boldsymbol{2}}[l]) \cdot \boldsymbol{x}_{\boldsymbol{l}}[s] \right).$$

3. Once L guesses have been made for each m bits of $\boldsymbol{z}$, take the majority outcome for each of them and output the answer accordingly.

Let us analyse what $\mathcal{A}'$ achieves. The repeated experiments on $\mathcal{A}$ share some common randomness ω (namely X and Y). Let us denote by ω' the randomness "renewed" at each experiment (that is the randomness used to simulate the tag, the random challenge $\hat{\boldsymbol{a}}$, and $\mathcal{A}$'s internal randomness). By a standard averaging argument, it holds that with probability greater than $P_{\mathrm{FA}} + \frac{\delta}{2}$ over ω, the answer returned by $\mathcal{A}$ is correct in at least $m - t$ positions with probability greater[3] than $\frac{\delta}{2}$ over ω'. Let us assume that this is the case and show that $\mathcal{A}'$ returns a correct answer $\boldsymbol{z}$ with probability greater than $\frac{1}{2}$. The theorem will follow since $P_{\mathrm{FA}} > \frac{2}{2^m}$ as soon as $t > 1$ and the overall probability of success for $\mathcal{A}'$ will be greater than $\frac{P_{\mathrm{FA}}}{2} + \frac{\delta}{4} > \frac{1}{2^m} + \frac{\delta}{4}$.

First we will show that, during phase 2(a), $\mathcal{A}'$ simulates a tag algorithm $\mathcal{T}_{X,Y,\eta}$, where X is the X' matrix with $\boldsymbol{z}$ as j-th row. To see this, observe that when $\alpha_i = \boldsymbol{a}_{\boldsymbol{i}}^{+}[j] = 0$, then

$$\boldsymbol{z}_{\boldsymbol{i}}^{+} = \boldsymbol{a}_{\boldsymbol{i}}^{+} \cdot X \oplus \boldsymbol{b}_{\boldsymbol{i}} \cdot Y \oplus \boldsymbol{\nu}_{\boldsymbol{i}} = \boldsymbol{a}_{\boldsymbol{i}}^{+} \cdot X \oplus \boldsymbol{b}_{\boldsymbol{i}}^{+} \cdot Y \oplus \boldsymbol{\nu}_{\boldsymbol{i}},$$

whereas when $\alpha_i = \boldsymbol{a}_{\boldsymbol{i}}^{+}[j] = 1$, then

$$\boldsymbol{z}_{\boldsymbol{i}}^{+} = \boldsymbol{a}_{\boldsymbol{i}}^{+} \cdot X \oplus \boldsymbol{z} \oplus \boldsymbol{b}_{\boldsymbol{i}} \cdot Y \oplus \boldsymbol{\nu}_{\boldsymbol{i}} = \boldsymbol{a}_{\boldsymbol{i}}^{+} \cdot X \oplus (\boldsymbol{b}_{\boldsymbol{i}} \oplus \boldsymbol{b}) \cdot Y \oplus \boldsymbol{\nu}_{\boldsymbol{i}} = \boldsymbol{a}_{\boldsymbol{i}}^{+} \cdot X \oplus \boldsymbol{b}_{\boldsymbol{i}}^{+} \cdot Y \oplus \boldsymbol{\nu}_{\boldsymbol{i}}.$$

Let us now analyse the advantage $\mathcal{A}'$ enjoys during a single guess for one bit of $\boldsymbol{z}$ during phase 2. First, one can upper bound the probability that $\mathcal{A}'$ enters phase 2(b) by the probability that any one of the q experiments results in the discarding of r pairs of the extra challenge-response pairs, which is $q2^{-r}$. Taking $r = \log_2 q + 1$ yields a probability not greater than $1/2$.

Consider phase 2(d) for a fixed bit position s. The guess of $\mathcal{A}'$ is right when both bits $\hat{\boldsymbol{z}}_{\boldsymbol{1}}[s]$ and $\hat{\boldsymbol{z}}_{\boldsymbol{2}}[s]$ are correct, or when they are both incorrect. Hence we are interested in lower bounding the probability p' of this event. First, we will lower bound the probability p over ω' that the s-th bit of the answer returned

[3] Otherwise the probability of success of the adversary would be upper bounded by $(1 - P_{\mathrm{FA}} - \frac{\delta}{2})\frac{\delta}{2} + P_{\mathrm{FA}} + \frac{\delta}{2} < \delta + P_{\mathrm{FA}}$, contradicting the hypothesis on $\mathcal{A}$.

by $\mathcal{A}$ is correct. We will assume *w.l.o.g.* that this probability is the same in all positions (otherwise one can "symmetrize" $\mathcal{A}$ by applying a random permutation of $\{1,\ldots,m\}$ to the problem). We can lower bound p as follows. Suppose we draw a *random* bit position s. Clearly, this bit is correct with probability p over the choice of s and ω'. At the same time, conditioned on the fact that more than $m-t$ bits are correct, the s-th bit of the answer is correct with probability greater than $1-u$. Consequently, the overall probability for the s-th bit to be correct is greater than $(1-u)\frac{\delta}{2}+\frac{1}{2}(1-\frac{\delta}{2})$, hence $p \geq \frac{1}{2}+\epsilon$ where $\epsilon = \frac{\delta}{2}(\frac{1}{2}-u)$. Juels and Weis proved [10, Lemma 2] that in this case, the probability, *conditioned on the fact that* $\hat{\boldsymbol{a}}_1$ *and* $\hat{\boldsymbol{a}}_2$ *differ in a single bit* j, that both bits $\hat{\boldsymbol{z}}_1[s]$ and $\hat{\boldsymbol{z}}_2[s]$ are correct or incorrect at the same time, is greater than $\frac{1}{2}+\epsilon^3/2-(\epsilon^3+1)/k_X$. However one can improve on their analysis by using Jensen's inequality[4]. Let γ denote the randomness except for $\hat{\boldsymbol{a}}$ in the experiment ω' we are considering. For a fixed γ, let p_γ denote the probability over $\hat{\boldsymbol{a}}$ that the s-th bit of the answer from $\mathcal{A}$ is correct. We've just proved that $\sum_\gamma p_\gamma \geq \frac{1}{2}+\epsilon$. Let p'_γ denote for a fixed γ, the probability, *conditioned on the fact that* $\hat{\boldsymbol{a}}_1$ *and* $\hat{\boldsymbol{a}}_2$ *differ in a single bit* j, that both bits $\hat{\boldsymbol{z}}_1[s]$ and $\hat{\boldsymbol{z}}_2[s]$ are correct or incorrect at the same time. Following the proof of [10, Lemma 2] we have $p'_\gamma \geq \phi(p_\gamma)$ where

$$\phi(x) = x^2\left(\frac{k_X+\log_2 x-1}{k_X}\right)+(1-x)^2\left(\frac{k_X+\log_2(1-x)-1}{k_X}\right).$$

As ϕ is convex, one has the following inequalities:

$$p' = \sum_\gamma p'_\gamma \geq \sum_\gamma \phi(p_\gamma) \geq \phi(\sum_\gamma p_\gamma) = \phi(p) \geq \phi(\frac{1}{2}+\epsilon) \geq \frac{1}{2}+2\epsilon^2-\frac{1}{k_X}.$$

As $\mathcal{A}'$ enters phase 2(b) with probability less than $1/2$, the probability that $\mathcal{A}'$ guesses bit $\boldsymbol{z}[s]$ correctly is lower-bounded by $\frac{1}{4}+\frac{p'}{2} \geq \frac{1}{2}+\epsilon'$, with $\epsilon' = \epsilon^2-\frac{1}{2k_X}$.

Using the Chernoff bound, taking the majority outcome of the L experiments allows $\mathcal{A}'$ to guess bit s with probability greater than

$$\pi = \left(1-e^{\frac{-L\epsilon'^2}{1+2\epsilon'}}\right) \geq \left(1-e^{\frac{-L\epsilon'^2}{2}}\right).$$

All m bits will be correct with probability greater than $\pi^m \geq \left(1-e^{\frac{-L\epsilon'^2}{2}}\right)^m$. A probability of success greater than $\frac{1}{2}$ can be attained by taking

$$L = \frac{2}{\epsilon'^2}\ln\left(\frac{1}{1-e^{-\frac{\ln 2}{m}}}\right) \sim \frac{512}{\delta^4(1-2u)^4}(\ln m - \ln\ln 2).$$

Hence, any efficient DET-adversary achieving a noticeable advantage against the RANDOM-HB$^\#$ protocol can be turned into an efficient solver of the MHB puzzle with a success probability greater than $\frac{1}{2^m}+\delta'$, where δ' is noticeable. This contradicts the assumption that LPN is hard. □

[4] Note that this will also improve the security reduction for HB$^+$.

Theorem 2 (Security of RANDOM-HB$^\#$ in the GRS-MIM-model). *Let $\mathcal{A}$ be an adversary attacking the RANDOM-HB$^\#$ protocol in the GRS-MIM-model, modifying at most q executions of the protocol between an honest tag and an honest reader, running in time T, and achieving advantage greater than δ. Then, under an easily met condition on the parameter set (see the proof and Section 4.2), there is an adversary $\mathcal{A}'$ attacking the RANDOM-HB$^\#$ protocol in the DET-model, interacting at most q times with an honest tag, running in time $O(T)$, and impersonating a valid tag with success probability greater than $(P_{FA}+\delta)(1-q\epsilon)$ for some negligible function ϵ. Hence, assuming the hardness of the LPN problem, the advantage of any efficient GRS-MIM-adversary against the RANDOM-HB$^\#$ protocol is negligible in k. As a consequence, for parameters $m=\Theta(k)$, the probability of any efficient GRS-MIM-adversary to impersonate a valid tag is negligible in k.*

Proof. As $\mathcal{A}'$ has access to an honest tag that it can query freely, there is no difficulty in simulating an honest tag to $\mathcal{A}$. The main challenge comes with the task of simulating the honest reader. Recall that in the GRS-MIM-model, the adversary is only allowed to modify the messages from the reader to the tag. $\mathcal{A}'$ launches the first phase of the adversary $\mathcal{A}$ and simulates the tag and the reader for q times as follows:

1. $\mathcal{A}'$ obtains from the real tag $\mathcal{T}_{X,Y,\eta}$ a blinding vector $\boldsymbol{b_i}$; $\mathcal{A}'$ sends $\boldsymbol{b_i}$ as the blinding vector of the simulated tag to the simulated reader.
2. $\mathcal{A}'$ sends a random vector $\boldsymbol{a_i}$ as the challenge of the simulated reader. $\mathcal{A}$ modifies it into $\boldsymbol{a'_i} = \boldsymbol{a_i} \oplus \boldsymbol{\alpha_i}$. $\mathcal{A}'$ forwards $\boldsymbol{a'_i}$ to the real tag.
3. The real tag returns an answer $\boldsymbol{z_i} = \boldsymbol{a'_i} \cdot X \oplus \boldsymbol{b_i} \cdot Y \oplus \boldsymbol{\nu_i}$ to $\mathcal{A}'$ which uses it as the answer of the simulated tag to the simulated reader.
4. If $\boldsymbol{\alpha_i}$ was the all zero vector, $\mathcal{A}'$ outputs "ACCEPT" as the answer of the simulated reader, otherwise it outputs "REJECT".

After this first phase, $\mathcal{A}'$ launches the cloning phase of $\mathcal{A}$ and replicates its behaviour with the real reader. From the point of view of $\mathcal{A}$, the tag $\mathcal{T}_{X,Y,\eta}$ is perfectly simulated by $\mathcal{A}'$. Let Sim_i denote the event that the reader $\mathcal{R}_{X,Y,u}$ is correctly simulated by $\mathcal{A}$ during the i-th execution of the protocol, and Sim be the event that the reader is correctly simulated for all the q executions of the protocol, $\mathrm{Sim} = \cap_{i=1}^{q} \mathrm{Sim}_i$. Conditioning on this event Sim, the success probability of $\mathcal{A}'$ is the same as the success probability of $\mathcal{A}$, *i.e.* $P_{\mathrm{FA}} + \delta$. Hence, we have to lower bound the probability of Sim.

Consider one execution of the disturbed protocol. When $\boldsymbol{\alpha_i} = \mathbf{0}$, $\mathcal{A}'$ clearly fails at simulating the reader with a probability equal to the probability of wrongly rejecting an honest tag, *i.e.* P_{FR}. For the case $\boldsymbol{\alpha_i} \neq \mathbf{0}$ we make the following reasoning. Assume that the error vector $\boldsymbol{\alpha_i} \cdot X$ added by $\mathcal{A}$ has a Hamming weight d. This vector is added *before* the Bernoullian noise added by the tag, so that $\boldsymbol{\nu_i}$ is independent of $\boldsymbol{\alpha_i} \cdot X$. Consequently, the resulting error vector $\boldsymbol{\nu_i} \oplus \boldsymbol{\alpha_i} \cdot X$ has a Hamming weight distributed as the sum of d Bernoulli variables taking the value 1 with probability $1-\eta$ and 0 with probability η, and $m-d$ Bernoulli variables taking the value 1 with probability η and 0 with probability $1-\eta$. Hence, the mean value of the Hamming weight of the error

vector is $\mu(d) = d(1-\eta) + (m-d)\eta$, and by the Chernoff bound, when $\mu(d) > t$, this weight is less than t with probability less than $e^{-\frac{(\mu-t)^2}{2\mu}}$, which remains true for any $d' \geq d$. Consequently, if the matrix X is such that for any $\boldsymbol{\alpha} \neq \mathbf{0}$, $\mathrm{Hwt}(\boldsymbol{\alpha} \cdot X)$ is high enough, outputting "REJECT" as soon as $\boldsymbol{\alpha_i} \neq \mathbf{0}$ will be a successful strategy. We formalize this as follows.

Let $d_{\min}(X) = \min_{\boldsymbol{\alpha} \neq \mathbf{0}} (\mathrm{Hwt}(\boldsymbol{\alpha} \cdot X))$ denote the minimal distance of the matrix X. We recall the following classical result of coding theory:

Lemma 1. *Let d be an integer in $[1.. \lfloor \frac{m}{2} \rfloor]$ and let H be the entropy function $H(x) = -x \log_2(x) - (1-x)\log_2(1-x)$. Then*

$$\Pr_X[d_{\min}(X) \leq d] \leq 2^{-\left(1 - \frac{k_X}{m} - H(\frac{d}{m})\right)m}.$$

This is a simple consequence of the following upper bound on the number of m-bit vectors of Hamming weight less than d: $\sum_{i=0}^{d} \binom{m}{i} \leq 2^{mH(\frac{d}{m})}$. For any non-zero vector $\boldsymbol{\alpha}$, $\boldsymbol{\alpha} \cdot X$ is uniformly distributed, and hence has Hamming weight less than d with probability less than $2^{m\left(H(\frac{d}{m})-1\right)}$. The lemma follows by a union bound.

Let $\tilde{d}$ be the least integer such that $\mu > t$, *i.e.* $\tilde{d} = 1 + \left\lfloor \frac{t - \eta m}{1 - 2\eta} \right\rfloor$. Then for any $d \geq \tilde{d}$ when $\boldsymbol{\alpha_i} \neq \mathbf{0}$, one can write

$$\begin{aligned}
\Pr_{X, \boldsymbol{\nu_i}}[\overline{\mathrm{Sim}_i}] &= \Pr_{\boldsymbol{\nu_i}}[\overline{\mathrm{Sim}_i} \,|\, d_{\min}(X) > d] \cdot \Pr_X[d_{\min}(X) > d] \\
&\qquad + \Pr_{\boldsymbol{\nu_i}}[\overline{\mathrm{Sim}_i} \,|\, d_{\min}(X) \leq d] \cdot \Pr_X[\min(X) \leq d] \\
&\leq \Pr_{\boldsymbol{\nu_i}}[\overline{\mathrm{Sim}_i} \,|\, d_{\min}(X) > d] + \Pr_X[d_{\min}(X) \leq d] \\
&\leq e^{-\frac{(\mu-t)^2}{2\mu}} + 2^{-\left(1 - \frac{k_X}{m} - H(\frac{d}{m})\right)m}.
\end{aligned}$$

For this upper bound to be useful, the coefficient $\left(1 - \frac{k_X}{m} - H(\frac{d}{m})\right)$ must be positive for some $d \geq \tilde{d}$, in particular for $\tilde{d}$ as it is a decreasing function of d. This is a condition which is easily met for typical values of the parameters (see Section 4.2). Note also that for the asymptotic reduction we have to define $\tilde{d}$ as the least integer such that $\mu(\tilde{d}) > (1+c)t$ for some $c > 0$ in order to ascertain that the first term in the upper bound will be negligible. This way one has, for all $d \geq \tilde{d}$, $e^{-\frac{(\mu-t)^2}{2\mu}} \leq e^{-\frac{uc^2}{2(1+c)}m}$.

Together we have $\Pr[\overline{\mathrm{Sim}_i}] \leq \epsilon$, where ϵ is a negligible function given by

$$\epsilon = \max\left\{P_{\mathrm{FR}}, \min_{d \geq \tilde{d}}\left(e^{-\frac{(\mu-t)^2}{2\mu}} + 2^{-\left(1 - \frac{k_X}{m} - H(\frac{d}{m})\right)m}\right)\right\}.$$

Consequently, $\Pr[\mathrm{Sim}] \geq (1 - q\epsilon)$ and $\mathcal{A}'$ has a success probability greater than $(P_{\mathrm{FA}} + \delta)(1 - q\epsilon)$.

If δ is noticeable then $q\epsilon(P_{\mathrm{FA}} + \delta) \leq \delta/2$ for k big enough, and the success probability of $\mathcal{A}'$ is greater than $P_{\mathrm{FA}} + \frac{\delta}{2}$. This contradicts Theorem 1. □

With RANDOM-HB$^{\#}$ we have a surprisingly successful proposal. It is as computationally efficient as HB^{+} since it consists of a series of bitwise dot-product computations. At the same time it is simpler in terms of communication since there is only a single round and the total amount of data transmitted is much less than for HB^{+}. It also possesses a proof of security in the detection-based model, exactly like HB^{+}, but also against man-in-the-middle adversaries of the type used in the GRS attack. However there remains one drawback: storage. We show how to remedy this situation in the next section.

4 The Proposal HB$^{\#}$

In RANDOM-HB$^{\#}$ the tag is required to store two random $(k_X \times m)$- and $(k_Y \times m)$-binary matrices X and Y where k_X, k_Y and m are three-digit figures. The storage costs on the tag would be insurmountable. With this in mind we propose the protocol HB$^{\#}$. This has very modest storage requirements while preserving the computational efficiency of HB^{+}. While there are some subtle technical issues that mean we cannot transfer all the provably security results from RANDOM-HB$^{\#}$ to HB$^{\#}$ we can transfer some. These, together with a plausible conjecture, allow us to claim that HB$^{\#}$ is secure in the GRS-MIM-model. HB$^{\#}$ depends on the notion of a *Toeplitz* matrix. These were used by Krawczyk in message authentication proposals where their good distribution properties and efficient implementation were noted [14,15].

A $(k \times m)$-binary *Toeplitz* matrix M is a matrix for which the entries on every upper-left to lower-right diagonal have the same value. Since the diagonal values of a Toeplitz matrix are fixed, the entire matrix is specified by the top row and the first column. Thus a Toeplitz matrix can be stored in $k + m - 1$ bits rather than the km bits required for a truly random matrix. For any $(k + m - 1)$-bit vector $\boldsymbol{s}$, we denote by $T_{\boldsymbol{s}}$ the Toeplitz matrix whose top row and first column are represented by $\boldsymbol{s}$. HB$^{\#}$ is defined exactly as RANDOM-HB$^{\#}$ except that X and Y are now two random $(k_X \times m)$ and $(k_Y \times m)$-binary Toeplitz matrices.

4.1 Security results for HB$^{\#}$

While there is every indication that HB$^{\#}$ is secure in the DET-model, this remains to be shown. A first obvious step in this direction would be to prove that the Toeplitz variant of the MHB puzzle remains hard. We state the following conjecture to stimulate further research:

Conjecture 1 (Hardness of the Toeplitz-MHB puzzle). Let k be a security parameter, $\eta \in]0, 1/2[$, and m and q be polynomials in k. Let X be a random secret $(k \times m)$-binary *Toeplitz* matrix, and $(\boldsymbol{a_1}, \ldots, \boldsymbol{a_q})$ be q random vectors of length k. Then any efficient algorithm, on input q noisy samples $(\boldsymbol{a_i}, \boldsymbol{a_i} \cdot X \oplus \boldsymbol{\nu_i})$, where each bit of $\boldsymbol{\nu_i}$ is 1 with probability η, and a random vector $\boldsymbol{a}$ of length k, outputs $\boldsymbol{z} = \boldsymbol{a} \cdot X$ with probability negligibly close to $\frac{1}{2^m}$.

Just as for RANDOM-HB$^{\#}$, we can relate the security of the HB$^{\#}$ protocol in the GRS-MIM-model to its security in the DET-model.

Table 2. Practical parameters for HB$^{\#}$.

HB$^{\#}$					False reject	False accept	Transmission	Storage
k_X	k_Y	m	η	t	rate (P_{FR})	rate (P_{FA})	(bits)	(bits)
80	512	1164	0.25	405	2^{-45}	2^{-83}	1,756	2,918
80	512	441	0.125	113	2^{-45}	2^{-83}	1,033	1,472

Theorem 3 (Security of HB$^{\#}$ in the GRS-MIM-model). *Let $\mathcal{A}$ be an adversary attacking the HB$^{\#}$ protocol in the* GRS-MIM*-model, modifying at most q executions of the protocol between an honest tag and an honest reader, running in time T, and achieving advantage greater than δ. Then, under an easily met condition on the parameter set (see proof of Theorem 2 and Section 4.2), there is an adversary $\mathcal{A}'$ attacking the HB$^{\#}$ protocol in the* DET*-model, interacting at most q times with an honest tag, running in time $O(T)$, and impersonating a valid tag with success probability greater than $(P_{FA}+\delta)(1-q\epsilon)$ for some negligible function ϵ.*

Proof. (Outline) The proof is analogous to that of Theorem 2 and omitted for reasons of space. It relies on the observation that Lemma 1 remains true when the probability is taken over the set of random $(k_X \times m)$-Toeplitz matrices. □

Hence, the security of HB$^{\#}$ in the DET-model (which we believe to be a likely conjecture) would directly transfer to the GRS-MIM-model.

4.2 Parameter values for HB$^{\#}$

When considering the error rates in HB$^{\#}$, we have considerable flexibility in how we set the acceptance threshold t. Recall that the false rejection rate depends on m, t, and η and the false acceptance rate depends on m and t only. The overall security of the scheme depends on k_X, k_Y and η. However, as already noted by Levieil and Fouque [16] for HB^{+}, and as is clear from the proof of Theorem 1, k_X and k_Y play two different roles: only k_Y is related to the difficulty of the LPN problem, while k_X need only be 80-bit long to achieve 80-bit security.

Some example parameters for different noise levels η are given by Levieil and Fouque [16]. These give very reasonable error rates of $P_{\mathrm{FR}} < 2^{-40}$ and $P_{\mathrm{FA}} < 2^{-80}$. When combined with the larger values of k_Y required for good security with the LPN problem, the HB$^{\#}$ protocol compares very favourably to HB^{+}. The practical characteristics are summarised in Table 2. The condition necessary for Theorems 2 and 3 to hold is verified for both sets of parameters: for the first one, $\tilde{d} = 229$ and $\left(1 - \frac{k_X}{m} - H(\frac{\tilde{d}}{m})\right) \simeq 0.216$, while for the second one $\tilde{d} = 78$ and $\left(1 - \frac{k_X}{m} - H(\frac{\tilde{d}}{m})\right) \simeq 0.145$. The storage cost of HB$^{\#}$ is $(k_X + k_Y + 2m - 2)$ bits which is larger than the $2k$ bits required for HB^{+}. However, depending on the choice of m this is not necessarily a substantial increase. The given parameter choices offer 80-bit security (using the latest results on the LPN problem), the

false acceptance and rejection rates are less than 2^{-80} and 2^{-40} respectively, and the total communication requirements are around 1,500 bits. This should be compared to error rates of 2^{-1} and 2^{-20} and transmission costs of up to 80,000 bits in the case of HB$^+$ (48,000 bits when $\boldsymbol{x}$ is only 80-bit long) for corresponding parameters. HB$^\#$ requires simple bit operations on-the-tag and thus remains computationally simple.

5 Further work and HB$^\#$ variants

General MIM adversaries. The result of Theorem 3 shows that an adversary successfully mounting an attack on HB$^\#$ must either (i) break HB$^\#$ in the DET-model (which we believe is highly improbable), or (ii) break the LPN problem, or (iii) use an undiscovered active attack involving more than manipulation of the messages from the reader. This raises the question of the security of HB$^\#$ against general man-in-the-middle adversaries allowed to perturb any message of the protocol. Though we do not have a formal proof of such a result, we can make the following heuristic analysis. To provide an appropriate context we might recall earlier work by Krawczyk [14,15]. Let us denote by $\mathcal{H}_{\mathcal{T}}$, where $\mathcal{T}$ stands for "random Toeplitz" matrix, the (k, m)-family of k-bit to m-bit linear functions $\boldsymbol{a} \mapsto \boldsymbol{a} \cdot T_{\boldsymbol{s}}$ associated with the set of $k \times m$ binary Toeplitz matrices $T_{\boldsymbol{s}}$, each associated with a $(k + m - 1)$-bit vector $\boldsymbol{s}$, and equipped with the uniform probability. The work of Krawczyk [15], which in turn references related work by Mansour *et al.* [17], in effect establishes that $\mathcal{H}_{\mathcal{T}}$ is $\frac{1}{2^m}$-balanced. In other words, for any non-zero vector $\boldsymbol{a}$, $\boldsymbol{a} \cdot T_{\boldsymbol{s}}$ is uniformly distributed over $\{0,1\}^m$. This results from the fact that if $\boldsymbol{a}$ is a non-zero vector then $\boldsymbol{a} \cdot T_{\boldsymbol{s}}$ can be re-written as the product of $\boldsymbol{s}$ with a $(k + m - 1) \times m$ matrix derived from $\boldsymbol{a}$ that has rank m.

We can use this property of Toeplitz matrices to argue in favour of the resistance of HB$^\#$ against arbitrary man-in-the-middle adversaries. Consider an attack where the adversary perturbs $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{z}$ by adding respectively three disturbance vectors $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, $\boldsymbol{\gamma}$. The modified error vector is then $\boldsymbol{\nu'} = \boldsymbol{\nu} \oplus \boldsymbol{\alpha} \cdot X \oplus \boldsymbol{\beta} \cdot Y \oplus \boldsymbol{\gamma}$. When $\boldsymbol{\alpha} \neq \boldsymbol{0}$ or $\boldsymbol{\beta} \neq \boldsymbol{0}$, then due to the $\frac{1}{2^m}$-balance of $\mathcal{H}_{\mathcal{T}}$, $\boldsymbol{\nu'}$ is uniformly distributed and the probability that modifications of the communication between tag and reader result in successful authentication is the false acceptance probability P_{FA}. The reader's decision has negligible entropy and hence yields no information on X or Y to the adversary. On the contrary, when $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = (\boldsymbol{0}, \boldsymbol{0})$, the answer $\boldsymbol{z}$ returned by the tag is uniformly random so that $\boldsymbol{\gamma}$ may be considered as independent of X and Y. The reader's decision depends only on $\boldsymbol{\nu} \oplus \boldsymbol{\gamma}$ and again yields no information on X or Y to the adversary. It is helpful to note the essential difference between a man-in-the-middle attack on HB$^\#$ and the same attack on HB$^+$. When attacking HB$^+$, *e.g.* as is done in the GRS attack, the adversary gains 1 bit of information on $\boldsymbol{x}$ at *every* tag and reader HB$^+$ authentication (independently of whether it is successful or not), leading to a linear-time attack. By contrast, in the case of HB$^\#$, whatever the strategy for choosing $(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma})$, the mutual information between the reader's decision and the

matrices X and Y is negligible and no efficient adversary can gather noticeable information on X or Y. Though we believe that these observations can be made rigorous, it remains an open problem to extend the technique used in proof of Theorems 2 and 3 to arbitrary man-in-the-middle attacks and to find the right way of simulating the reader when the adversary can also modify $\boldsymbol{b}$ and $\boldsymbol{z}$.

Variants and optimisations. Independently of this theoretical work, there are interesting variants to HB$^{\#}$ that might be of practical value. One interesting option, also mentioned in [12], is for the legitimate tag to test that the noise vector $\boldsymbol{\nu}$ contains no more than t ones before using it. This means the probability of a false rejection would fall to zero. The main advantage of this approach would be to allow the size of m to decrease while maintaining a reasonable false acceptance rate. For instance, with $m = 256$, $\eta = 0.125$, and $t = 48$ we would ordinarily have that $P_{\mathrm{FA}} \approx 2^{-81}$ while $P_{\mathrm{FR}} \approx 2^{-9}$. However, this relatively high false rejection rate can be eliminated by allowing the tag to check $\boldsymbol{\nu}$ before use.

Another possibility to decrease storage and communication costs is to reduce k_Y; for this, it might be interesting to consider the effect of using a larger noise level, *i.e.* to have $\eta > \frac{1}{4}$. In such circumstances k_Y could be reduced—while maintaining the same level of security—thereby leading to storage and communications savings. While it is not immediately clear that this would be a successful approach, when coupled with restrictions to the noise vector $\boldsymbol{\nu}$ this may be worth exploring. Another optimisation could be to use techniques inspired by Krawczyk [14,15] to efficiently re-generate the Toeplitz matrices (*e.g.* by using a *LFSR*). We leave such proposals as topics for future research.

6 Conclusions

In this paper we have presented two new lightweight authentication protocols. While close variants of HB^{+}, these new protocols offer considerable advantages over related work in the literature. RANDOM-HB$^{\#}$ is provably secure in the detection-based model, just like HB^{+}, but it is also provably resistant to a broader class of attacks that includes [8]. The protocol HB$^{\#}$ trades some of the theoretical underpinnings to RANDOM-HB$^{\#}$ and attains a truly practical performance profile. Both RANDOM-HB$^{\#}$ and HB$^{\#}$ offer practical improvements over HB^{+}, and this remains the case even when using the problem sizes required after recent progress on solving the underlying LPN problem.

References

1. A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *Proceedings of CHES 2007*, LNCS 4727, pp. 450–466, Springer, 2007.
2. J. Bringer, H. Chabanne, and E. Dottax. HB^{++}: A Lightweight Authentication Protocol Secure Against Some Attacks. In *Proceedings of SecPerU 2006*, pp. 28–33, IEEE Computer Society Press, 2006.

3. R. Canetti, S. Halevi and M. Steiner. Hardness Amplification of Weakly Verifiable Puzzles. In *Proceedings of TCC 2005*, LNCS 3378, pp. 17–33, Springer, 2005.
4. D.N. Duc and K. Kim. Securing HB$^+$ Against GRS Man-in-the-Middle Attack. In *Institute of Electronics, Information and Communication Engineers, Symposium on Cryptography and Information Security*, Jan. 23–26, 2007.
5. M. Feldhofer, S. Dominikus, and J. Wolkerstorfer. Strong Authentication for RFID Systems Using the AES Algorithm. In *Proceedings of CHES 2004*, LNCS 3156, pp. 357–370, Springer, 2004.
6. M.P.C. Fossorier, M.J. Mihaljevic, H. Imai, Y. Cui, and K. Matsuura. A Novel Algorithm for Solving the LPN Problem and its Application to Security Evaluation of the HB Protocol for RFID Authentication. Available from `http://eprint.iacr.org/2006/197.pdf`.
7. H. Gilbert, M.J.B. Robshaw, and Y. Seurin. Good Variants of HB$^+$ are Hard to Find. In *Proceedings of Financial Crypto 2008*, to appear.
8. H. Gilbert, M.J.B. Robshaw, and H. Sibert. An Active Attack Against HB$^+$: A Provably Secure Lightweight Authentication Protocol. *IEE Electronics Letters*, volume 41, number 21, pp. 1169–1170, 2005.
9. M. Girault, G. Poupard and J. Stern. On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order. *Journal of Cryptology*, volume 19, number 4, pp. 463–488, 2006.
10. A. Juels and S.A. Weis. Authenticating Pervasive Devices With Human Protocols. In *Proceedings of Crypto 2005*, LNCS 3126, pp. 293–198, Springer, 2005.
11. A. Juels and S.A. Weis. Authenticating Pervasive Devices With Human Protocols. Version of [10] with appendices. Available from `http://saweis.net/pdfs/lpn-paper.pdf`.
12. J. Katz and J. Shin. Parallel and Concurrent Security of the HB and HB$^+$ Protocols. In *Proceedings of Eurocrypt 2006*, LNCS 4004, pp. 73–87, Springer, 2006.
13. J. Katz and A. Smith. Analysing the HB and HB$^+$ Protocols in the "Large Error" Case. Available from `http://eprint.iacr.org/2006/326.pdf`.
14. H. Krawczyk. LFSR-based Hashing and Authentication. In *Proceedings of Crypto 1994*, LNCS 839, pp. 129–139, Springer, 1994.
15. H. Krawczyk. New Hash Functions for Message Authentication. In *Proceedings of Eurocrypt 1995*, LNCS 950, pp. 301–310, Springer, 1995.
16. E. Levieil and P.-A. Fouque. An Improved LPN Algorithm. In *Proceedings of SCN 2006*, LNCS 4116, pp. 348–359, Springer, 2006.
17. Y. Mansour, N. Nisan, and P. Tiwari. The Computational Complexity of Universal Hashing. In *Proceedings of STOC '90*, pp. 235–243, 1990.
18. M. McLoone and M.J.B. Robshaw. Public Key Cryptography and RFID. In *Proceedings of CT-RSA 2007*, LNCS 4377, pp. 372–384, Springer, 2007.
19. J. Munilla and A. Peinado. HB-MP: A Further Step in the HB-family of Lightweight Authentication Protocols. *Computer Networks*, volume 51, pp. 2262–2267, 2007.
20. S. Piramuthu. HB and Related Lightweight Authentication Protocols for Secure RFID Tag/Reader Authentication. *CollECTeR Europe Conference*, June 2006.
21. A. Poschmann, G. Leander, K. Schramm, and C. Paar. New Lightweight DES Variants Suited for RFID Applications. In *Proceedings of FSE 2007*, LNCS 4593, pp. 196–210, Springer, 2007.
22. A. Shamir. SQUASH - a New MAC With Provable Security Properties for Highly Constrained Devices Such as RFID Tags. In *Proceedings of FSE 2008*, to appear.

χ^2 Cryptanalysis of the SEAL Encryption Algorithm

Helena Handschuh *
Gemplus PSI
1, Place de la Méditerranée
95200 Sarcelles
France

Henri Gilbert
France Télécom
CNET PAA-TSA-SRC
38-40 Rue du Général Leclerc
92131 Issy-les-Moulineaux
France

Abstract. SEAL was first introduced in [1] by Rogaway and Coppersmith as a fast software- oriented encryption algorithm. It is a pseudorandom function which stretches a short index into a much longer pseudorandom string under control of a secret key pre-processed into internal tables. In this paper we first describe an attack of a simplified version of SEAL, which provides large parts of the secret tables from approximately 2^{24} algorithm computations. As far as the original algorithm is concerned, we construct a test capable of distinguishing SEAL from a random function using approximately 2^{30} computations. Moreover, we describe how to derive some bits of information about the secret tables. These results were confirmed by computer experiments.

1 Description of the SEAL Algorithm

SEAL is a length-increasing "pseudorandom" function which maps a 32-bit string n to an L-bit string SEAL(n) under a secret 160-bit key a. The output length L is meant to be variable, but is generally limited to 64 kbytes. In this paper, we assume it is worth exactly 64 kbytes (2^{14} 32-bit words), but all our results could be obtained with a smaller output length.

The key a is only used to define three secret tables R, S, and T. These tables respectively contain 256, 256 and 512 32-bit values which are derived from the Secure Hash Algorithm (SHA) [2] using a as the secret key and re-indexing the 160-bit output into 32-bit output words.

SEAL is the result of the two cascaded generators shown below.

* The study reported in this paper was performed while Helena Handschuh was working at France Télécom-CNET.

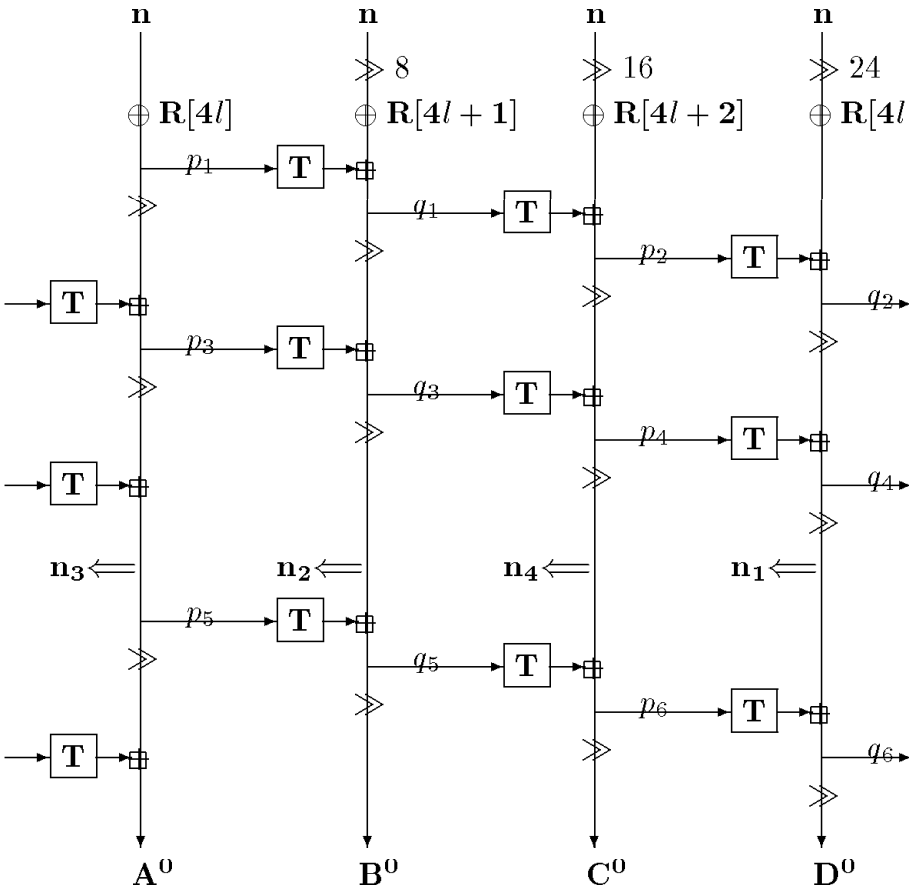


Fig. 1. The first generator of SEAL

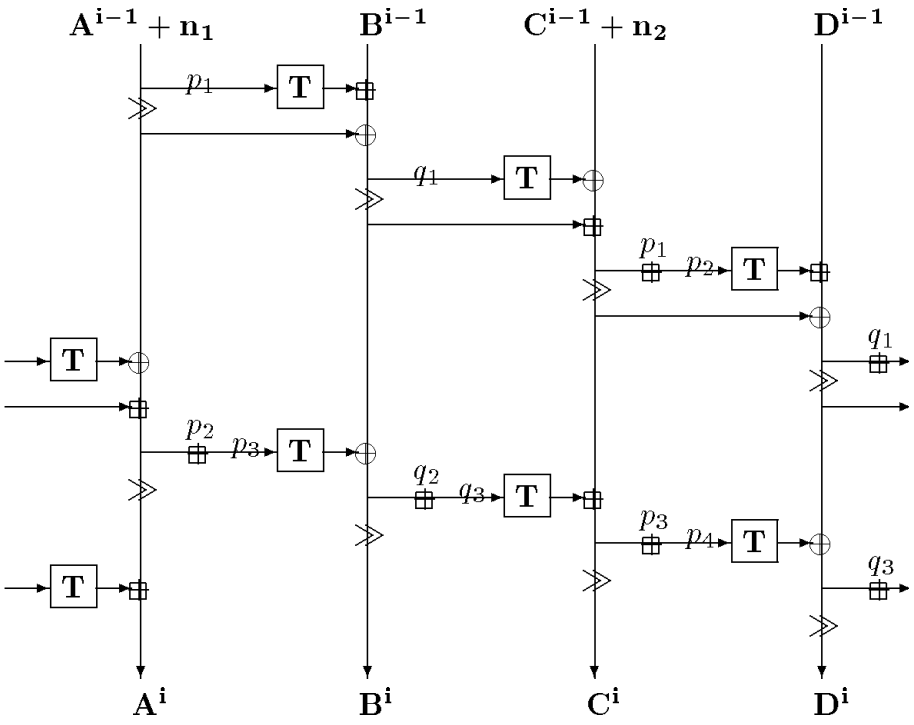


Fig. 2. The second generator of SEAL (i^{th} iteration)

The first generator uses a routine depending on the a-derived tables R and T depicted at figure 1. It maps the 32-bit string n and the 6-bit counter l to four 32-bit words A^0, B^0, C^0, D^0 and another four 32-bit words n_1, n_2, n_3, n_4. These eight words are to be used by the second generator.

The second generator uses a routine depending on the a-derived tables depicted at figure 2. There are 64 iterations of this routine, indexed by $i = 1$ to 64. $A^0B^0C^0D^0$ serves as an input to the first iteration, producing an $A^1B^1C^1D^1$ block. For the next iterations, the input block is alternately $(A^{i-1} + n_1, B^{i-1}, C^{i-1}+n_2, D^{i-1})$ for even i values and $(A^{i-1}+n_3, B^{i-1}, C^{i-1}+n_4, D^{i-1})$ for odd i values. At iteration i, the output block (Y_1^i, Y_2^i, Y_3^i, Y_4^i) is deduced from the intermediate block (A^i,B^i,C^i,D^i) using the a-derived table S as shown below in figure 3.

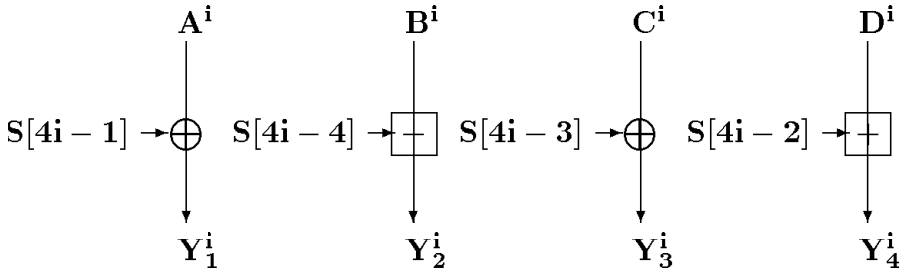


Fig. 3. Deriving the generator output

In the above figures :

- $\oplus$ stands for the XOR function;
- $\boxplus$ stands for the sum $\pmod{2^{32}}$;
- $\gg$ stands for a right rotation of 9 bits;
- $\gg N$ stands for a right rotation of N bits;
- p_1 through p_4 and q_1 through q_4 stand for the inputs of table T obtained from the 9 bits 2 to 11 of values A, B, C and D; for instance in figure 1, $p1 = A\&0x7fc$.

Concerning the definition of SEAL, more details can be found in [1] and in [2].

The algorithm is divided into three steps.

- First we compute the internal tables under the secret key a. The security of this step relies on SHA which is assumed to be highly secure. Therefore, R, S and T are pseudorandom tables.
- Second we compute A^0, B^0, C^0, D^0, n_1, n_2, n_3 and n_4 from n, l and table R. This is what we already called the first generator. Let us assume the output is pseudorandom as well.

– Finally, the second generator computes iteratively the $A^i B^i C^i D^i$ from which the Y_1^i, Y_2^i, Y_3^i, Y_4^i values are derived. We change the notations as follows :

  - $Y_1^i = A^i \oplus S_1^i$
  - $Y_2^i = B^i + S_2^i$
  - $Y_3^i = C^i \oplus S_3^i$
  - $Y_4^i = D^i + S_4^i$.

  In this part we found certain weaknesses which are investigated in S 3 and 4.

2 Preliminaries

2.1 Role of mod 2^{32} additions

Although the combined use of the + and $\oplus$ operations probably stre SEAL as compared with a situation where only one of these operation be used, we do not believe that this represents the main ingredient of t rity of SEAL, which is essentially a table- driven algorithm.

As a matter of fact, any $x + y$ sum can be written :

$$x + y = x \oplus y \oplus c(x, y)$$

where the carry word $c(x, y)$ is far from being uniformly distributed just introduces an additional, unbalanced term, as compared with $\oplus$.

This remark led us to assume that replacing in SEAL (more precisely second generator of figure 2) all + operators by xors would not fundar modify the nature of the algorithm, and that cryptanalytic results obtair such a simplified version could at least partially be transposed to the rea The results of our analysis of this simplified version of SEAL are summa Section 3 hereafter.

2.2 The three words D^{i-1}, C^i and D^i are correlated

Let us consider the function depicted at figure 2. Given a fixed valu iteration index i (say $i = 3$), the input and output to this function are from the generator outputs $(Y_1^{i-1}, Y_2^{i-1}, Y_3^{i-1}, Y_4^{i-1})$ and $(Y_1^i, Y_2^i, Y_3^i,$ to the following unknown words :

– the 8 words $(S_1^{i-1}, S_2^{i-1}, S_3^{i-1}, S_4^{i-1})$ and $(S_1^i, S_2^i, S_3^i, S_4^i)$, which va not depend upon the considered initial value (n, l).
– the 2 words n_1 and n_2, which value depends upon (n, l).

The involvement of the IV-dependent words n_1 and n_2 in the function considerably complicates the analysis of the i^{th} iteration because of the randomisation effect upon the input to output dependency.

In order to find statistics applicable to any IV value, we investigate how to "get rid" of any dependency in n_1 and n_2 in some relations induced by the equation of iteration i.

Let us consider the D^{i-1} input word and the C^i and D^i input words. Denote the output tables involved in the right part of figure 2 by : $T_1 = T[p_2]$, $T_2 = T[q_3]$ and $T_3 = T[p_4]$. It is easy to establish the relation :

(1)
$$(D^{i-1} + T_1) \oplus (C^i \ll 9 + T_2) = (D^i \ll 18) \oplus (T_3 \ll 9)$$

This relation does not involve n_1 and n_2. The T_1, T_2, T_3 terms in this relation can be seen as three random values selected from the T table. Since there are only 2^9 values in the T table, given any two words out of the (D^{i-1}, C^i, D^i) triplets, there are at most 2^{27} possible values for the third word of the triplet instead of 2^{32} if D^{i-1}, C^i and D^i were statistically independent. This gives some evidence that the D^{i-1} input and the C^i and D^i output are statistically correlated, in a way which does not depend upon n_1 and n_2. In other words, the SEAL generator derives from an (n, l) initial value three slightly correlated output words Y_4^{i-1}, Y_3^i and Y_4^i.

Relation (1) above represents the starting point for the various attacks reported in Sections 3 to 5 hereafter.

3 An Attack of a simplified version of SEAL

In this Section we present an attack of the simplified version of SEAL obtained by replacing in figure 2 all mod 2^{32} additions by xors. The attack is divided into four steps.

3.1 Step 1

We derive the unordered set of values of the T table, up to an unknown 32-bit constant D^i. Relation (1) above represents the starting point for this derivation. After replacing + by $\oplus$ in (1) and D^{i-1}, C^i, D^i by $X_4 = Y_4^i$, $Y_3 = Y_3^i$ and $Y_4 = Y_4^i$ respectively, we obtain the relation :

(2)
$$Y_4 \oplus Y_3 \gg 9 \oplus X_4 \gg 18 = T_3 \gg 9 \oplus (T_1 \oplus T_2) \gg 18 \oplus \Delta^i \gg 9$$

where the Δ^i constant depends upon the S table. T_1 and T_2 are 2 am values of table T. Statistically speaking, once in 2^9, $T_1 = T_2$, and $T_1 \oplus T$ we compute 2^{18} samples, each of the 512 values of table $T \oplus \Delta^i$ should once in average.

We collect the combination of the generator output words given by term of (2) for about 2^{21} (n, l) samples. Whenever one value appears m 4 times, we assume this is a value of table $T \oplus \Delta^i$. All the other valu a probability of about $\frac{2^{21}}{2^{32}}$ to appear. This way, we find about 490 ou values of table T up to one constant value.

3.2 Step 2

The purpose of the second step is to compute a constant α^i which is n the third step in order to find out statistics involving B^{i-1}, D^i, A^i and Fig. 2.). Consider the following equation (3) established in a similar wa from the relation between B^{i-1} and the output words :

(3) $Y_4 \gg 9 \oplus Y_2 \oplus Y_1 \gg 9 \oplus X_2 \gg 18 \oplus T_3 \gg 18 = (T'_1 \oplus T'_2) \gg 18 \oplus (T'_3 \oplus$ $9 \oplus (S_4^i \gg 9 \oplus S_2^{i-1} \gg 18 \oplus S_2^i \oplus S_1^i \gg 9)$

where

$$\alpha^i = S_4^i \gg 9 \oplus S_2^{i-1} \gg 18 \oplus S_2^i \oplus S_1^i \gg 9 \oplus \Delta^i \gg 18$$

For each sample, we can find out $T_3 \oplus \Delta^i$ by searching exhaustively t combination (T_1, T_2, T_3) in equation (2). In order to save time, we comp and for all a table with the 2^{18} values of $(T_1 \oplus T_2) \gg 18$ and search for t third value. We perform this search as well as the computation of the l of (3) for 2^{21} samples. Once in 2^{18}, $T'_1 = T'_2$ and $T'_3 = T'_4$. This way the value α^i we are looking for appears at least 4 times.

3.3 Step 3

The purpose of this step is to find out various values of n_1. Once we ha values, we can make out the relation between the in and outputs of tab to one constant value. Let us consider equation (4) established from the between A^{i-1} and the output words :

(4)

$X_1 \gg 18 \oplus Y_1 \oplus Y_4 \oplus T'_2 \gg 9 \oplus T'_4 \oplus T_3 \gg 9 = n_1 \gg 18 \oplus S_1^{i-1} \gg 18 \oplus$

We can find out $T'_2 \oplus \Delta^i$ and $T'_4 \oplus \Delta^i$ by searching the right combination of (T'_1, T'_2, T'_3, T'_4) in equation (2) using the value α^i we made out in step 2. For each sample we compute, we get about 16 possibilities, as (T'_1, T'_2, T'_3, T'_4) gives 2^{36} possible values for a 32- bit word.

In order to find the right combination, let us consider two distinct iteration indexes i and j : we know that for a given l value, if i is even (or odd), we always xor the same n_1 (or n_3) to the input A. Let us therefore take two rounds i and j that are both odd (or even). We need to know table $T \oplus \Delta^i$, table $T \oplus \Delta^j$, α^i and α^j.

We collect samples of :

- $n_1 \gg 18 \oplus \beta^i$
  where $\beta^i = S_1^{i-1} \gg 18 \oplus S_4^i \oplus S_1^i \oplus \Delta^i$;
- $n_1 \gg 18 \oplus \beta^j \oplus \Delta^j \gg 9 \oplus \Delta^i \gg 9$
  as value T_3 is found through table $T \oplus \Delta^i$ and values T'_2 and T'_4 through table $T \oplus \Delta^j$.

We xor all the samples of round i with all the samples of round j. One of these values is the right combination of $\beta^i \oplus \beta^j \oplus (\Delta^i \oplus \Delta^j) \gg 9$

Then we find all the samples for rounds i and j of another value n_1 (i.e. of round l). We compare these two sets of samples and make out the right value of $n_1 \gg 18 \oplus \beta^i$.

This step can be repeated various times to collect values of n_1 while computing only once the tables $T \oplus \Delta$ and the constants α.

3.4 Step 4

In this step we finally derive the in and outputs of table T from equation (5):

(5)
$$p_1 = ((X_1 \oplus n_1 \oplus S_1^{i-1}) \& 0x7fc)/4$$

In this equation p_1 is the input of table T. We have seen in the first three steps that we can derive the value of T_1 from in and output samples of SEAL.

So we finally derive several values of :

$T[p \oplus \delta^i] \oplus \Delta^i$

where $\delta^i = ((S_1^{i-1} \oplus \beta^i \ll 18) \& 0x7fc)/4$.

3.5 Summary

Summing up the four steps we have just described, we can break the T t
to one constant value using about 2×2^{21} samples of (n, l) for step 1,
samples of (n, l) for step 2 and about 2^9 values of (n, l) for steps 3 and
means, the T table can be broken using about 2^{24} samples of (n, l).

We could probably go on breaking the simplified version of SEAL by
out sets of values (n_1, n_2, n_3, n_4), then trying to break the first genera
find table R, but this is not our purpose here.

4 A Test of the real version of SEAL

In this Section we use some of the ideas of Vaudenay's Statistical Crypt
of Block Ciphers [3] to distinguish SEAL from a truely random functio

4.1 χ^2 Cryptanalysis

The purpose of Vaudenay's paper is to prove that statistical analysis on
such as DES may provide as efficient attacks as linear or differential cr
ysis. Statistical analysis enables to recover very low biases and a simple
can get very good results even without knowing exactly what happens in
inner loops of the algorithm or the S- boxes.

We intend to use this property to detect low biases of a certain com
of the output words of SEAL suggested by the analysis made in Sect
order to prove SEAL is far from being undistinguishable from a pseudo-
function. This provides a first test of the SEAL algorithm.

4.2 Number of samples needed for the χ^2 test to distinguis biased distribution from an unbiased one

We denote by N the number of samples computed. We assume the sam
drawn from a set of r values. We call n_i the number of occurances of t
the r values among the N samples and S_{χ^2} the associated indicator :

$$S_{\chi^2} = \frac{\sum_{i=1}^{r} (n_i - \frac{N}{r})^2}{\frac{N}{r}}$$

The χ^2 test compares the value of this indicator to the one an unbia
tribution would be likely to provide. If the n_i were drawn according to
ased multinomial distribution of parameters $(\frac{1}{r}, \frac{1}{r}, ..., \frac{1}{r})$, the expectation
standard deviation of the S_{χ^2} estimator would be given by :

- $E(S_{\chi^2}) \rightarrow \mu = r - 1$
- $\sigma(S_{\chi^2}) \rightarrow \sigma = \sqrt{2(r-1)}$

If the distribution of the n_i is still multinomial but biased, say with probabilities $p_1, ..., p_r$, then we can compute the new expected value μ' of the S_{χ^2} :

$$\mu' = E(\frac{\sum_{i=1}^{r}(n_i - \frac{N}{r})^2}{\frac{N}{r}}) = \frac{N}{r}\sum_{i=1}^{r} E((n_i - p_i N + p_i N - \frac{N}{r})^2)$$

It can be easily shown that :

$$\mu' \rightarrow \mu + r(N-1)\sum_{i=1}^{r}(p_i - \frac{1}{r})^2$$

An order of magnitude of the number N of samples needed by the χ^2 test to distinguish a biased distribution from an unbiased one with substantial probability is given by the condition :

$$\mu' - \mu \gg \sigma$$

which gives us the following order of magnitude for N :

$$N \gg \frac{\sqrt{2(r-1)}}{r\sum_{i=1}^{r}(p_i - \frac{1}{r})^2}$$

4.3 Model of the test

Let us consider equation (2) with the real scheme (including the sums). We can rewrite it :

$(6) Y_4 \oplus Y_3 \gg 9 \oplus X_4 \gg 18 = T_3 \gg 9 \oplus (T_1 \oplus T_2) \gg 18 \oplus (r_1 \oplus r_2 \oplus r_3) \gg 18 \oplus \Delta^i \gg 9 \oplus r_4$

where r_1 and r_2 are the carry bits created by the addition of T_1 and T_2, and r_3 and r_4 the ones of the addition of S_4^{i-1} and S_4^i.

We apply the χ^2 test to the four leftmost bits of $Y_4 \oplus Y_3 \gg 9 \oplus X_4 \gg 18$ suspecting a slight bias in this expression. Without having carefully analysed the exact distribution of the sum of the four carries, we intend to prove that its convolution with the biased distribution of $T_3 \gg 9 \oplus (T_1 \oplus T_2) \gg 18 \oplus \Delta^i \gg 9$ does result in a still slightly unbalanced distribution.

As we take 4-bit samples, we apply the χ^2 test with $r - 1 = 15$ degrees of freedom. Detailed information about this test can be found in [4].

4.4 Results

Whenever we analyse at least 2^{33} samples, the test proves with probabil to be wrong, that SEAL has a biased distribution. We have made sever and each time the value of the S_{χ^2} estimator we obtained for this order nitude of N was greater than 40 for the 4 least significant bits and grea 320 for the 8 least significant bits. In other words, the test proves that tribution is a biased one.

Figure 4 shows the value of the S_{χ^2} estimator for tests made with $a = 0x6$

S_{χ^2}	2^{23}	2^{24}	2^{25}	2^{26}	2^{27}	2^{28}	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}	2^{34}
4 bits	14.27	25	13.97	9.41	26.96	16.5	16.78	29.65	21.05	30.15	45.74	44.6
8 bits	261	293	274	238	229	227	246	225	278	313	331	378

Fig. 4. Results of the tests with up to 2^{35} samples of (n, l).

The former test test can be slightly improved as follows : let us denote least significant bits of S_4^i by s_4^i. For each of the 16 possible values of s the S_{χ^2} test to the 4 or the 8 least significant bits of $(Y_4 - s_4^i) \oplus Y_3 \gg 9 \oplus X$ The test with the correct s_4^i value detects a bias with 2^{30} (n, l) values o Figure 5). Note that a significant bias is also detected whenever the t significant bits of s_4^i are correct.

S_{χ^2}	2^{25}	2^{26}	2^{27}	2^{28}	2^{29}	2^{30}	2^{31}	2^{32}	2^{33}	2^{34}	2^{35}
4 bits	9.45	13.56	20.62	25.59	32.77	71.90	83.63	130	250	438	928
8 bits	253	271	292	321	276	357	379	438	569	838	1520

Fig. 5. Results of the test with the correct value of the four s_4^i bit

4.5 Deriving first information on SEAL

The interpretation of the above improved test is straightforward : when least significant bits of s_4^i are correct, r_4 is partly known and the χ^2 te much better results than with wrong bits of s_4^i. Therefore we can derive two bits of information on table S. If the test is applied to more than leftmost bits of the samples, more than 2 bits can be derived from secr T. Whenever these bits are right, the χ^2 rises much faster than for wrong

As the evolution of the χ^2 indicator is quite close to a straight line w divergence starts, the results can be checked applying the test to 2^{20} 2^{32} samples. Divergence becomes obvious when about 2^{30} samples ha computed.

5 Deriving information on the T table

In this Section we give some evidence that the initial step of the attack on the simplified version of SEAL introduced in Section 3 can be adapted to provide large parts of the T table for the real algorithm.

Let us consider relation (6).

As seen in Section 3.1, the distribution of the $T_3 \gg 9 \oplus (T_1 \oplus T_2) \gg 18 \oplus \Delta^i \gg 9$ value at the right of (6) is unbalanced. The most frequent values are provided by the 512 $T \oplus \Delta^i$ words.

On the other hand the distribution of the carry words $r_1 \oplus r_2 \oplus r_3$ and r_4 is also unbalanced. More precisely, due to the fact that in any carry word r each bit $r[j]$ has a $\frac{3}{4}$ probability of being equal to the next bit $r[j+1]$, the number of 'inversions', i.e. j values s.t. $r[j] \neq r[j+1]$ is likely to be small when r is a carry word or an exclusive or of carry words.

Thus we can expect the 512 $T \oplus \Delta^i$ values to give rise to 'spread' probability maxima in the distribution probability of the left term of (6).

Based on 2^{32} (n, l) values, we did the following experiment :

We analysed the probabililty distribution of the 23 lowest weight bits of the left combination of (6) in order to reduce the memory requirements. So in the rest of this Section, though we do not introduce any new notation, we implicitely refer to 23-bit words instead of 32-bit words.

For several $T \oplus \Delta^i$ values (about 25), we computed the sum of the probabilities of the neighbours of $T \oplus \Delta^i$, i.e. the values of the form $T \oplus \Delta^i + r$, where r is one of the approximately 2^{22} 23-bit words with at most 11 inversions.

We computed the same sum of approximately 2^{22} probabilities around arbitrarily chosen values other than the 512 $T \oplus \Delta^i$ values.

For more than half of the $T \oplus \Delta^i$ values, the obtained sum was larger than all the sums associated to the arbitrarily chosen values.

The complexity of the search of the $T \oplus \Delta^i$ values is quite high if an independent computation of a sum of 2^{22} probabilities is made for each of the 2^{23} candidate values. This approach leads to a 2^{45} complexity, far over the computing capabilities of the computer we used for the experiments ; however, substantial gains might be achieved by reusing appropriately selected partial sums of probabilities.

Thus in summary, we believe that with slightly more than 2^{32} (n, l) values, it should be possible to recover a substantial part of the information on the T

table, up to an unknown constant.

6 Conclusion

We have shown in some detail in Section 3 that the simpler scheme w instead of sums can be attacked with the generator output correspor about 2^{24} samples of (n, l), e.g. 2^{18} n values and 2^6 l values for each n

The test of Section 4 can be applied with about 2^{30} samples of (n, l), e. values and 2^6 l values for each n value. Moreover, information about tab be derived from this test with 2^{30} samples of (n, l) as well, and large am information contained in table T can be derived from approximately 2^{32} of (n, l) or slightly more.

Despite of their relatively low time and space complexity, which ena to perform the computer simulations mentioned in Sections 4 and 5, tacks reported in this paper do not seriously endanger the practical sec SEAL, because a too large amount of keystream samples (corresponding than 2^{30} (n, l) initialisation vectors) is required. These attacks suggest that simple modifications of some design features of SEAL, e.g. the the involvement of the IV-dependent values n_1 to n_4 in the second ge would probably strengthen the algorithm without significant impact performance.

7 Acknowledgements

We thank Thierry Baritaud and Pascal Chauvaud for the elaboratio LaTeX version of the figures and the paper. We would like to address thanks to François Allègre who gave us access to a quite powerful compu made the tests reasonably quick and to Alain Scheiwe who helped us w C source code of the tests.

References

1. P. Rogaway and D. Coppersmith, "A Software-Optimized Encryption Alg Proceedings of the 1993 Cambridge Security Workshop, Springer-Verlag,
2. B. Schneier, Applied Cryptography, Second Edition, John Wiley & Sons,
3. S. Vaudenay, "Statistical Cryptanalysis of Block Ciphers - χ^2 Cryptanalysi
4. J. Bass, Eléments de Calcul des Probabilités, 3^e édition, Masson Et Cie,

Cryptanalysis of Grain

Côme Berbain[1], Henri Gilbert[1], and Alexander Maximov[2]

[1] France Telecom Research and Development
38-40 rue du Général Leclerc, 92794 Issy-les-Moulineaux, France
[2] Dept. of Information Technology, Lund University, Sweden
P.O. Box 118, 221 00 Lund, Sweden
{come.berbain, henri.gilbert}@francetelecom.com
movax@it.lth.se

Abstract. Grain [11] is a lightweight stream cipher proposed by M. Hell, T. Johansson, and W. Meier to the eSTREAM call for stream cipher proposals of the European project ECRYPT [5]. Its 160-bit internal state is divided into a LFSR and an NFSR of length 80 bits each. A filtering boolean function is used to derive each keystream bit from the internal state. By combining linear approximations of the feedback function of the NFSR and of the filtering function, it is possible to derive linear approximation equations involving the keystream and the LFSR initial state. We present a key recovery attack against Grain which requires 2^{43} computations and 2^{38} keystream bits to determine the 80-bit key.

Keywords: Stream cipher, Correlation attack, Walsh transform

1 Introduction

Stream ciphers are symmetric encryption algorithms based on the concept of pseudorandom keystream generator. In the typical case of a binary additive stream cipher, the key and an additional parameter named initialization vector (IV) are used to generate a binary sequence called keystream which is bitwise combined with the plaintext to provide the ciphertext. Although it seems rather difficult to construct a very fast and secure stream cipher, some efforts to achieve this have recently been deployed. The NESSIE project [24] launched in 1999 by the European Union did not succeed in selecting a secure enough stream cipher. Recently, the European Network of Excellence in Cryptology ECRYPT launched a call for stream cipher proposals named eSTREAM [5]. The candidate stream ciphers were submitted in May 2005. Those candidates are divided into software oriented and hardware oriented ciphers.

[0] The work described in this paper has been supported in part by Grant VR 621-2001-2149, in part by the French Ministry of Research RNRT X-CRYPT project and in part by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

Hardware oriented stream ciphers are specially designed so that their implementation requires a very small number of gates. Such ciphers are useful in mobile systems, e.g. mobile phones or RFID, where minimizing the number of gates and power consumption is more important than very high speed.

One of the new hardware candidates submitted to eSTREAM is a stream cipher named Grain [11] which was developed by M. Hell, T. Johansson, and W. Meier[3] as an alternative to stream ciphers like GSM A5/1 or Bluetooth E_0. It uses a 80-bit key and a 64-bit initialization vector to fill in an internal state of size 160 bits divided into a *nonlinear feedback shift register* (NFSR) and a *linear feedback shift register* (LFSR) of length 80 bits each. At each clock pulse, one keystream bit is produced by selecting some bits of the LFSR and of the NFSR and applying a boolean function. It is well known that LFSR sequences satisfy several statistical properties one would expect from a random sequence, but do not offer any security. Their combination with NFSR sequences is expected to improve the security. However, NFSR based constructions have not yet been as well studied as LFSR based constructions. The claimed security level of Grain is 2^{80}, and it was conjectured by the authors of Grain that there exists no attack significantly faster than exhaustive search.

In this paper, we describe two key recovery attacks against Grain. The proposed attacks exploit linear approximations of the output function. The first one requires 2^{55} operations, 2^{49} bits of memory, and 2^{51} keystream bits, and the second one requires 2^{43} operations, 2^{42} bits of memory, and 2^{38} keystream bits.

This paper is organized as follows. We first describe the Grain stream cipher (Section 2) and we derive some linear approximations involving the LFSR and the keystream (Section 3). We then present two techniques for recovering the initial state of the LFSR (Section 4). Finally, we present a technique allowing to recover the initial state of the NFSR once we know the LFSR initial state (Section 5).

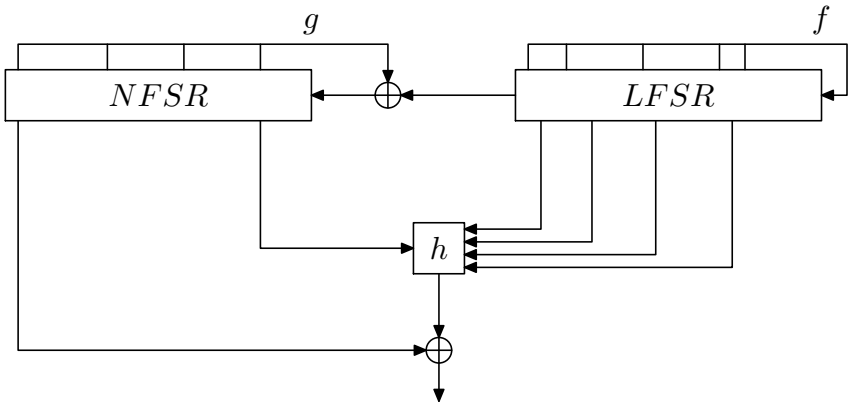
2 Description of Grain

Grain [11] is based upon three main building blocks: an 80-bit linear feedback shift register, an 80-bit nonlinear feedback shift register, and a nonlinear filtering function. Grain is initialized with the 80-bit key K and the 64-bit initialization value IV. The cipher output is an L-bit keystream sequence $(z_t)_{t=0,\ldots,L-1}$.

The current LFSR content is denoted by $Y^t = (y_t, y_{t+1}, \ldots, y_{t+79})$. The LFSR is governed by the linear recurrence:

$$y_{t+80} = y_{t+62} \oplus y_{t+51} \oplus y_{t+38} \oplus y_{t+23} \oplus y_{t+13} \oplus y_t.$$

[3] The design of Grain was also submitted and recently accepted for publication in *the International Journal of Wireless and Mobile Computing, Special Issue on Security of Computer Network and Mobile Systems.*



The current NFSR content is denoted by $X^t = (x_t, x_{t+1}, \ldots, x_{t+79})$. The NFSR feedback is disturbed by the output of the LFSR, so that the NFSR content is governed by the recurrence:

$$x_{t+80} = y_t \oplus g(x_t, x_{t+1}, \ldots, x_{t+79}),$$

where the expression of nonlinear feedback function g is given by

$$\begin{aligned} g(x_t, x_{t+1}, \ldots, x_{t+79}) = {} & x_{t+63} \oplus x_{t+60} \oplus x_{t+52} \oplus x_{t+45} \oplus x_{t+37} \oplus x_{t+33} \oplus x_{t+28} \\ & \oplus x_{t+21} \oplus x_{t+15} \oplus x_{t+9} \oplus x_t \oplus x_{t+63}x_{t+60} \oplus x_{t+37}x_{t+33} \\ & \oplus x_{t+15}x_{t+9} \oplus x_{t+60}x_{t+52}x_{t+45} \oplus x_{t+33}x_{t+28}x_{t+21} \\ & \oplus x_{t+63}x_{t+45}x_{t+28}x_{t+9} \oplus x_{t+60}x_{t+52}x_{t+37}x_{t+33} \\ & \oplus x_{t+63}x_{t+60}x_{t+21}x_{t+15} \oplus x_{t+63}x_{t+60}x_{t+52}x_{t+45}x_{t+37} \\ & \oplus x_{t+33}x_{t+28}x_{t+21}x_{t+15}x_{t+9} \\ & \oplus x_{t+52}x_{t+45}x_{t+37}x_{t+33}x_{t+28}x_{t+21}. \end{aligned}$$

The cipher output bit z_t is derived from the current LFSR and NFSR states as the exclusive or of the masking bit x_t and a nonlinear filtering function h as follows:

$$\begin{aligned} z_t &= x_t \oplus h(y_{t+3}, y_{t+25}, y_{t+46}, y_{t+64}, x_{t+63}) \\ &= h'(y_{t+3}, y_{t+25}, y_{t+46}, y_{t+64}, x_t, x_{t+63}) \\ &= x_t \oplus x_{t+63}p_t \oplus q_t, \end{aligned}$$

where p_t and q_t are the functions of $y_{t+3}, y_{t+25}, y_{t+46}, y_{t+64}$ given by:

$$\begin{aligned} p_t &= 1 \oplus y_{t+64} \oplus y_{t+46}(y_{t+3} \oplus y_{t+25} \oplus y_{t+64}), \\ q_t &= y_{t+25} \oplus y_{t+3}y_{t+46}(y_{t+25} \oplus y_{t+64}) \oplus y_{t+64}(y_{t+3} \oplus y_{t+46}). \end{aligned}$$

The boolean function h is correlation immune of the first order. As noticed in [11], "this does not preclude that there are correlations of the output of $h(x)$ to sums of inputs", but the designers of Grain appear to have expected the NFSR masking bit x_t to make it impractical to exploit such correlations.

The key and IV setup consists of loading the key bits in the NFSR, loading the 64-bit IV followed by 16 ones in the LFSR, and clocking the cipher 160 times in a special mode where the output bit is fed back into the LFSR and the NFSR. Once the key and IV have been loaded, the keystream generation mode described above is activated and the keystream sequence (z_t) is produced.

3 Deriving Linear Approximations of the LFSR Bits

3.1 Linear Approximations Used to Derive the LFSR Bits

The purpose of the attack is, based on a keystream sequence $(z_t)_{t=0\ldots L-1}$ corresponding to an unknown key K and a known IV value, to recover the key K. The initial step of the attack is to derive a sufficient number N of linear approximation equations involving the 80 bits of the initial LFSR state $Y^0 = (y_0, \ldots, y_{79})$ (or equivalently a sufficient number N of linear approximation equations involving bits of the sequence (y_t)) to recover the value of Y^0. Hereafter, as will be shown in Section 5, the initial NFSR state X^0 and the key K can then be easily recovered.

The starting point for the attack consists in noticing that though the NFSR feedback function g is balanced, the function g' given by $g'(X^t) = g(X^t) \oplus x_t$ is unbalanced. We have:

$$Pr\{g'(X^t) = 1\} = \frac{522}{1024} = \frac{1}{2} + \epsilon_{g'},$$

where $\epsilon_{g'} = \frac{5}{512}$. It is useful to notice that the restriction of g' to input values X^t such that $x_{t+63} = 0$ is totally balanced and that the imbalance of the function g' is exclusively due to the imbalance of the restriction of g' to input values X^t such that $x_{t+63} = 1$.

If one considers one single output bit z_t, the involvement of the masking bit x_t in the expression of z_t makes it impossible to write any useful approximate relation involving only the Y^t bits. But if one considers the sum $z_t \oplus z_{t+80}$ of two keystream bits output at a time interval equal to the NFSR length 80, the $x_t \oplus x_{t+80}$ contribution of the corresponding masking bits is equal to $g'(X^t) \oplus y_t$, and is therefore equal to y_t with probability $\frac{1}{2} + \epsilon_{g'}$. As for the other terms of $z_t \oplus z_{t+80}$, they can be approximated by linear functions of the bits of the sequence (y_t). In more details:

$$\begin{aligned} z_t \oplus z_{t+80} = g'(X^t) \oplus y_t \oplus h(y_{t+3}, y_{t+25}, y_{t+46}, y_{t+64}, x_{t+63}) \\ \oplus h(y_{t+83}, y_{t+105}, y_{t+126}, y_{t+144}, x_{t+143}). \end{aligned}$$

To find linear approximations of the term $h(y_{t+3}, y_{t+25}, y_{t+46}, y_{t+64}, x_{t+63})$, we can restrict our search, since the restriction of $g'(X^t)$ to input values such that $x_{t+63} = 0$ is balanced, to input values such that $x_{t+63} = 1$, which amounts to finding linear approximations of $p_t \oplus q_t$.

We found a set of two best linear approximations for this function, namely:

$$L_1 = \{y_{t+3} \oplus y_{t+25} \oplus y_{t+64} \oplus 1; y_{t+25} \oplus y_{t+46} \oplus y_{t+64} \oplus 1\}.$$

Each of the approximations of L_1 is valid with a probability $\frac{1}{2}+\epsilon_1$, where $\epsilon_1 = \frac{1}{4}$.

Now the term $h(y_{t+83}, y_{t+105}, y_{t+126}, y_{t+144}, x_{t+143})$ is equal to either q_{t+80} or $p_{t+80} \oplus q_{t+80}$, with a probability $\frac{1}{2}$ for both expressions. We found a set of 8 best simultaneous linear approximations for these two expressions, namely:

$$\begin{aligned} L_2 = \{ & y_{t+83} \oplus y_{t+144} \oplus 1; && y_{t+83} \oplus y_{t+105} \oplus y_{t+126} \oplus y_{t+144} \oplus 1; \\ & y_{t+83} \oplus y_{t+126} \oplus y_{t+144}; && y_{t+83} \oplus y_{t+105} \oplus y_{t+126}; \\ & y_{t+83} \oplus y_{t+105}; && y_{t+83} \oplus y_{t+105} \oplus y_{t+144} \oplus 1; \\ & y_{t+105} \oplus y_{t+144}; && y_{t+105} \oplus y_{t+126} \oplus y_{t+144} \oplus 1; \}. \end{aligned}$$

Each of the 8 approximations of L_2 has an average probability $\epsilon_2 = \frac{1}{8}$ of being valid.

Thus, we have found 16 linear approximations of $z_t \oplus z_{t+80}$, namely all the linear expressions of the form

$$y_t \oplus l_1(y_{t+3}, y_{t+25}, y_{t+46}, y_{t+64}) \oplus l_2(y_{t+83}, y_{t+105}, y_{t+126}, y_{t+144}),$$

where $l_1 \in L_1$ and $l_2 \in L_2$. Each of these approximations is valid with a probability $\frac{1}{2} + \epsilon$, where ϵ is derived from $\epsilon_{g'}$, ϵ_1, and ϵ_2 using the Piling-up Lemma:

$$\epsilon = \frac{1}{2} \cdot 2^2 \cdot \epsilon_{g'} \cdot \epsilon_1 \cdot \epsilon_2 = \frac{5}{4096} \simeq 2^{-9.67}.$$

The extra multiplicative factor of $\frac{1}{2}$ takes into account the fact that the considered approximations are only valid when $x_{t+63} = 1$. The LFSR derivation attacks of Section 4 exploit these 16 linear approximations.

3.2 Generalisation of the Attack Method

In this Section, we try to generalise the previous approximation method. The purpose is not to find better approximations than those identified in Section 3.1, but to derive some design criteria on the boolean functions g and h'. However in the previous approximation, we used the fact that the bias of g depends on the value of x_{t+63}, so that the approximations of g and h' are not correct independently. We do not take this phenomenon into account in this Section. Therefore, we only provide a simplified picture of potential generalised attacks.

The function $g(X^t, Y^t)$ operates on $w(g) = w_L(g) + w_N(g)$ variables taken from the LFSR and the NFSR, where $w_L(g)$ is the number of variables taken from the LFSR and $w_N(g)$ the number of variables taken from the NFSR. Let the function $A_g(X^t, Y^t)$ be a linear approximation of the function g, i.e.

$$A_g(X^t, Y^t) = \bigoplus_{i=0}^{w_N(g)-1} d_i x_{t+\phi_g(i)} \oplus \bigoplus_{j=0}^{w_L(g)-1} c_j y_{t+\psi_g(j)}, \quad c_j, d_i \in \mathbb{F}_2, \qquad (1)$$

such that the distance between $g(\cdot)$ and $A_g(\cdot)$ defined by:

$$d_g = \sharp\{x \in \mathbb{F}_2^{w(g)} : A_g(x) \neq g(x)\} > 0,$$

is strictly larger than zero. Then, we have

$$\Pr\{A_g(x) \neq g(x)\} = \frac{1}{2^{w(g)}} d_g,$$

i.e.

$$\Pr\{A_g(x) + g(x) = 0\} = 1/2 + \epsilon_g,$$

where the bias is:

$$\epsilon_g = 1/2 - 2^{-w(g)} d_g.$$

Similarly, the function $h'(X^t, Y^t)$ can also be approximated by some linear expressions of the form:

$$A_{h'}(X^t, Y^t) = \bigoplus_{i=0}^{w_N(h')-1} k_i x_{t+\phi_{h'}(i)} \oplus \bigoplus_{j=0}^{w_L(h')-1} l_j y_{t+\psi_{h'}(j)}, \quad k_j, l_i \in \mathbb{F}_2. \tag{2}$$

Recall, $z_t \stackrel{p}{=} A_{h'}(\cdot)_t$ with some probability p. Knowing the expressions (1) and (2), one can sum up together $w_N(A_g(\cdot))$ expressions of $A_{h'}(\cdot)$ at different times t, in such a way that all terms X^t will be eliminated (just because the terms X^t will be cancelled due to the parity check function $A_g(\cdot)$, leaving the terms Y^t and noise variables only). Note also that any linear combination of $A_{h'}(\cdot)$ is a linear combination of the keystream bits z_t.

The sum of $w_N(A_g(\cdot))$ approximations $A_{h'}(\cdot)$ will introduce $w_N(A_g(\cdot))$ independent noise variables due to the approximation at different time instances. Moreover, the cancellation of the terms X^t in the sum will be done by the parity check property of the approximation $A_g(\cdot)$. If the function $A_{h'}(\cdot)$ contains $w_N(A_{h'})$ terms from X^t, then the parity cancellation expression $A_g(\cdot)$ will be applied $w_N(A_{h'})$ times. Each application of the cancellation expression $A_g(\cdot)$ will introduce another noise variable due to the approximation $N_g : g(\cdot) \rightarrow A_g(\cdot)$. Therefore, the application of the expression $A_g(\cdot)$ $w_N(A_{h'})$ times will introduce $w_N(A_{h'})$ additional noise variables N_g. Accumulating all above and following the Piling-up Lemma, the final correlation of such a sum (of the linear expression on Y^t) is given by the following Theorem.

Theorem 1. *There always exists a linear relation in terms of bits from the state of the LFSR and the keystream, which have the bias:*

$$\epsilon = 2^{(w_N(A_{h'})+w_N(A_g)-1)} \cdot \epsilon_g^{w_N(A_{h'})} \cdot \epsilon_{h'}^{w_N(A_g)},$$

where $A_g(\cdot)$ and $A_{h'}(\cdot)$ are linear approximations of the functions $g(\cdot)$ and $h'(\cdot)$, respectively, and:

$$\Pr\{A_g(\cdot) = g(\cdot)\} = 1/2 + \epsilon_g, \qquad \Pr\{A_{h'}(\cdot) = h'(\cdot)\} = 1/2 + \epsilon_{h'}.$$

This theorem gives us a criteria for a proper choice of the functions $g(\cdot)$ and $h'(\cdot)$. The biases ϵ_g and $\epsilon_{h'}$ are related to the *nonlinearity* of these boolean functions, and the values $w_N(A_g)$ and $w_N(A_{h'})$ are related to the *correlation immunity* property; however, there is a well-known trade-off between these two properties [27]. Unfortunately, in the case of Grain the functions $g(\cdot)$ and $h'(\cdot)$ were improperly chosen.

4 Deriving the LFSR Initial State

In the former Section, we have shown how to derive an arbitrary number N of linear approximation equations in the $n = 80$ initial LFSR bits, of bias $\epsilon \simeq 2^{-9.67}$ each, from a sufficient number of keystream bits. Let us denote these equations by:

$$\bigoplus_{i=0}^{n-1} \alpha_i^j \cdot y_i = b^j, j = 1, \ldots, N.$$

In this Section we show how to use these relations to derive the initial LFSR state Y^0. This can be seen as a decoding problem, up to the fact that the code length is not fixed in advance and one has to find an optimal trade-off between the complexities of deriving a codeword (i.e. collecting an appropriate number of linear approximation equations) and decoding this codeword.

An estimate of the number N of linear approximation equations needed for the right value of the unknown to maximize the indicator

$$I = \sharp\left\{ j \in \{1, \ldots, N\} \;\middle|\; \bigoplus_{i=0}^{n-1} \alpha_i^j \cdot y_i = b^j \right\},$$

or at least to be very likely to provide say one of the two or three highest values of I, can be determined as follows.

Under the heuristic assumption that for the correct (respectively incorrect) value of Y^0, I is the sum of N independent binary variables x_i distributed according to the Bernoulli law of parameters $p = Pr\{x_i = 1\} = \frac{1}{2} - \epsilon$ and $q = Pr\{x_i = 0\} = \frac{1}{2} + \epsilon$ (resp. the Bernoulli law of parameters $Pr\{x_i = 1\} = \frac{1}{2}$ and $Pr\{x_i = 0\} = \frac{1}{2}$, mean value $\mu = \frac{1}{2}$, and standard deviation $\sigma = \frac{1}{2}$), N can be derived by introducing a threshold of say $T = N(\frac{1}{2} + \frac{3\epsilon}{4})$ for I and requiring: *(i)* that the probability that I is larger than T for an incorrect value of Y^0 is less than a suitably chosen false alarm probability p_{fa}; *(ii)* that the probability that I is lower than T for the correct value is less than a non detection probability p_{nd} of say 1%. For practical values of p_{fa}, the first condition is by far the most demanding. Setting the false alarm rate to $p_{fa} = 2^{-n}$ ensures that the number of false alarms is less than 1 in average.

Due to the Central Limit Theorem, $\frac{\sum x_i - N\mu}{\sqrt{N}\sigma}$ is distributed according to the normal law, so that:

$$Pr\left\{ \frac{1}{N} \sum x_i - \mu > \frac{3\epsilon}{4} \right\} = Pr\left\{ \frac{\sum x_i - N\mu}{\sqrt{N}\sigma} > \frac{3\sqrt{N}\epsilon}{4\sigma} \right\} \tag{3}$$

can be approximated by $\frac{1}{\sqrt{2\pi}} \int_{\lambda}^{+\infty} e^{-\frac{t^2}{2}} dt$, where $\lambda = \frac{3\sqrt{N}\epsilon}{2}$. Consequently, if N is selected in such a way that $\frac{3\sqrt{N}\epsilon}{2} = \lambda$, i.e.

$$N = \left(\frac{2\lambda}{3\epsilon} \right)^2,$$

where λ is given by:

$$\frac{1}{\sqrt{2\pi}} \int_{\lambda}^{+\infty} e^{-\frac{t^2}{2}} dt = p_{fa} = 2^{-n},$$

then inequality 3 is satisfied.

A naive LFSR derivation method would consist of collecting N approximate equations, computing the indicator I independently for each of the 2^n possible values of Y^0 and retaining those Y^0 candidates leading to a value of I larger than the $N(\frac{1}{2} + \frac{3\epsilon}{4})$ threshold. This method would require a low number of keystream bits (say $\frac{N+80}{16}$) but the resulting complexity $N \cdot 2^{80}$ would be larger than the one of exhaustive key search.

In the rest of this Section, we show that much lower complexities can be obtained by using the fast Walsh transform algorithm and a few extra filtering techniques in order to speed up computations of correlation indicators. Former examples of applications of similar Fast Fourier Transform techniques in order to significantly decrease the total complexity of correlation attacks can be found in [4] [9] [16].

4.1 Use of the Fast Walsh Transform to Speed up Correlation Computations

Basic Method. Let us consider the following problem. Given a sufficient number M of linear approximation equations of bias ϵ involving m binary variables y_0 to y_{m-1}, how to efficiently determine these m variables? Let us denote these M equations by $\sum_{i=0}^{m-1} \alpha_i^j \cdot y_j = b^j, j = 1, \ldots, M$. For a sufficiently large value of M, one can expect the right value of $(y_0, \ldots, y_{m-1})$ to be the one maximizing the indicator:

$$\begin{aligned} I(y_0, \ldots, y_{m-1}) &= \sharp\left\{ j \in \{1, \ldots, M\} \;\middle|\; \sum_{i=0}^{m-1} \alpha_i^j \cdot y_j = b^j \right\} \\ &= \frac{M}{2} + \frac{1}{2} \cdot S(y_0, \ldots, y_{m-1}), \end{aligned}$$

where:

$$\begin{aligned} S(y_0, \ldots, y_{m-1}) = \sharp\left\{ j \in \{1, \ldots, M\} \;\middle|\; \sum_{i=0}^{m-1} \alpha_i^j \cdot y_i = b^j \right\} \\ - \sharp\left\{ j \in \{1, \ldots, M\} \;\middle|\; \sum_{i=0}^{m-1} \alpha_i^j \cdot y_i \neq b^j \right\}. \end{aligned}$$

Equivalently one can expect $(y_0, \ldots, y_{m-1})$ to be the value which maximizes the indicator $S(y_0, \ldots, y_{m-1})$. Instead of computing all of 2^m values of $S(y_0, \ldots, y_{m-1})$ independently, one can derive these values in a combined way using fast Walsh transform computations in order to save time.

Let us recall the definition of the Walsh transform. Given a real function of m binary variables $f(x_1, \ldots, x_{m-1})$, the Walsh transform of f is the real function of m binary variables $F = W(f)$ defined by:

$$F(u_0, \ldots, u_{m-1}) = \sum_{x_0, \ldots, x_{m-1} \in \{0,1\}^m} f(x_0, \ldots, x_{m-1})(-1)^{u_0 x_0 + \ldots + u_{m-1} x_{m-1}}.$$

Let us define the function $s(\alpha_0, \ldots, \alpha_{m-1})$ by:

$$\sharp\big\{j \in \{1, \ldots, M\} \mid (\alpha_0^j, \ldots, \alpha_{m-1}^j) = (\alpha_0, \ldots, \alpha_{m-1}) \wedge b^j = 1\big\}$$
$$- \sharp\big\{j \in \{1, \ldots, M\} \mid (\alpha_0^j, \ldots, \alpha_{m-1}^j) = (\alpha_0, \ldots, \alpha_{m-1}) \wedge b^j = 0\big\}.$$

The function s can be computed in M steps. Moreover, it is easy to check that the Walsh transform of s is S, i.e.

$$\forall (y_0, \ldots, y_{m-1}) \in \{0,1\}^m, W(s)(y_0, \ldots, y_{m-1}) = S((y_0, \ldots, y_{m-1})).$$

Therefore, the computational cost of the estimation of all the 2^m values of S using fast Walsh transform computations is $M + m \cdot 2^m$; the required memory is 2^m.

Improved Hybrid Method. More generally, if $m_1 < m$, one can use the following hybrid method between exhaustive search and Walsh transform in order to save space.

For each of the 2^{m-m_1} values of $(y_{m_1}, \ldots, y_{m-1})$, define the associated restriction S' of S as the m_1 bit boolean function given by:

$$S'(y_0, \ldots, y_{m_1-1}) = \sharp\left\{j \in \{1, \ldots, M\} \;\middle|\; \sum_{i=0}^{m_1-1} \alpha_i^j \cdot y_i = \sum_{i=m_1}^{m} \alpha_i^j \cdot y_i \oplus b^j \right\}$$
$$- \sharp\left\{j \in \{1, \ldots, M\} \;\middle|\; \sum_{i=0}^{m_1-1} \alpha_i^j \cdot y_i \neq \sum_{i=m_1}^{m} \alpha_i^j \cdot y_i \oplus b^j \right\}.$$

It is easy to see that if we define $s'(\alpha_0, \ldots, \alpha_{m_1-1})$ as

$$\sharp\left\{j \in \{1, \ldots, M\} \;\middle|\; (\alpha_0^j, \ldots, \alpha_{m_1-1}^j) = (\alpha_0, \ldots, \alpha_{m_1-1}) \wedge \sum_{i=m_1}^{m} \alpha_i^j \cdot y_i \oplus b^j = 1 \right\}$$
$$-\sharp\left\{j \in \{1, \ldots, M\} \;\middle|\; (\alpha_0^j, \ldots, \alpha_{m_1-1}^j) = (\alpha_0, \ldots, \alpha_{m_1-1}) \wedge \sum_{i=m_1}^{m} \alpha_i^j \cdot y_i \oplus b^j = 0 \right\},$$

then S' is the Walsh transform of s'.

Therefore, the computational cost of the estimation of all the 2^m values of S using this method is $2^{m-m_1}(M + m_1 \cdot 2^{m_1})$. If we compare this with the former basic Walsh transform method, we see that the required memory decreases from 2^m to 2^{m_1}, whereas the time complexity increase remains negligible as long as $m_1 << log_2(M)$.

4.2 First LFSR Derivation Technique

In order to reduce the LFSR derivation complexity when compared with the naive method of complexity $N \cdot 2^n$, we can exploit more keystream to produce more linear approximation equations in the unknowns y_0 to y_{n-1}, and retain only those equations involving the $m < n$ variables y_0 to y_{m-1}, i.e. which coefficients in the $n - m$ variables y_m to y_{n-1} are equal to 0.

Thus a fraction of about 2^{m-n} of the relations are retained and we have to collect about $N2^{n-m}$ approximate relations to retain N relations. This requires a number of keystream bits of:

$$\frac{N2^{n-m} + 80}{16}.$$

As seen in the former Section, once the relations have been filtered, the computational cost of the derivation of the values of these m variables using fast Walsh transform computations is about $m2^m$ for the basic method, and more generally $2^{m-m_1}(N + m_1 2^{m_1})$ if fast Walsh transform computations are applied to a restricted set of $m_1 < m$ variables.

Thus, the overall time complexity of this method is:

$$N2^{n-m} + m2^m,$$

and more generally:

$$N2^{n-m} + 2^{m-m_1}(N + m_1 2^{m_1}).$$

Once the m variables y_0 to y_{m-1} have been recovered, one can either reiterate the same technique for other choices of the m unknown variables, which increases the complexity by a factor of less than 2 if $m \geq \frac{n}{2}$, or test each of the 2^{n-m} candidates in the next step of the attack (NFSR and key derivation).

An estimate of the number N of equations needed is given by

$$N = \left(\frac{2\lambda}{3\epsilon}\right)^2,$$

where λ is determined by the condition $\frac{1}{\sqrt{2\pi}} \int_\lambda^{+\infty} e^{-\frac{t^2}{2}} dt = 2^{-m}$. This condition ensures that the expected number of false alarm is less than 1.

The minimal complexity is obtained for $m = 49$. For this parameter value, we have $\lambda = 7.87$ and $N = 2^{24}$. The attack complexity is about 2^{55}, the number of keystream bits needed is around 2^{51}, and the memory needed is about 2^{49}.

4.3 Second LFSR Derivation Technique

An alternative method is to derive new linear approximation equations (of lower bias) involving $m < n$ unknown variables y_0 to y_{m-1} by combining the R available approximate equations of bias ϵ pairwise, and retaining only those pairs of

relations for which the $n-m$ last coefficients collide. One obtains in this way about $N' = R^2 \cdot 2^{m-n-1}$ new affine equations in y_0 to y_{m-1}, of bias $\epsilon' = 2\epsilon^2$. The allocation of the m variables maximizing the number of satisfied equations can be found by fast Walsh computations as explained in the former Section.

The number N' of relations needed is about $\left(\frac{2\lambda}{3\epsilon'}\right)^2$, where λ is determined by the condition $\frac{1}{\sqrt{2\pi}} \int_{\lambda}^{+\infty} e^{-\frac{t^2}{2}} dt = 2^{-m}$. The required number R of relations of bias ϵ is therefore $R = (N' 2^{n-m+1})^{\frac{1}{2}}$, and the number of keystream bits needed is about $\frac{R+80}{16}$. The complexity of the derivation of the N' relations is $\max(R, N') = \max((N' 2^{n-m+1})^{\frac{1}{2}}, N')$.

Once the N' relations have been derived, the computational cost of the derivation of the values of these m variables using fast Walsh transform computations is about $m \cdot 2^m$ for the basic method, and more generally if fast Walsh transform computations are applied to a restricted set of $m_1 < m$ variables it costs $2^{m-m_1}(N' + m_1 \cdot 2^{m_1})$.

Thus the total complexity of the derivation of the m LFSR bits is:

$$\max((N' 2^{n-m+1})^{\frac{1}{2}}, N') + m2^m,$$

and more generally:

$$\max((N' 2^{n-m+1})^{\frac{1}{2}}, N') + 2^{m-m_1}(N' + m_1 2^{m_1}).$$

The minimal complexity is obtained for $m = 36$. For this parameter value, we have $\lambda = 6.65$ and $N' = 2^{41}$. The attack complexity is about 2^{43}, the number of keystream bits needed is about 2^{38} and the memory required is about 2^{42}.

5 Recovering the NFSR Initial State and the Key

Once the initial state of the LFSR has been recovered, we want to recover the initial state $(x_0, \ldots, x_{79})$ of the NFSR. Fortunately, the knowledge of the LFSR removes the nonlinearity of the output function and we can express each keystream bit z_i by one of the following four equations depending on the initial state of the LFSR:

$$z_i = x_i, \qquad z_i = x_i \oplus 1,$$
$$z_i = x_i \oplus x_{63+i}, \; z_i = x_i \oplus x_{63+i} \oplus 1.$$

Since functions p and q underlying h are balanced, each equation has the same occurrence probability. We are going to use the non linearity of the output function to recover the initial state of the NFSR by writing the equations corresponding to the first keystream bits.

The 16 first equations are linear equations involving only bits of the initial state of the NFSR because $63 + i$ is lower than 80.

To recover all the bits of the initial state, we introduce a technique which consists of building chains of keystream bits. The equations for keystream bits z_{17} to z_{79} involve either one bit of the NFSR ($z_i = x_i$ or $z_i = x_i \oplus 1$) or two bits ($z_i = x_i \oplus x_{63+i}$ or $z_i = x_i \oplus x_{63+i} \oplus 1$). An equation involving only one

bit allows us to instantly recover the value of the corresponding bit of the initial state. This can be considered as a chain of length 0. On the other hand, an equation involving two bits does not allow this because we do not know the value of x_{63+i} (for $i > 16$).

However, by considering not only the equations for z_i but also all the equation for $z_{k\cdot 63+i}$ for $k \geq 1$, we can cancel the bits we do not know and retrieve the value of x_i. With probability $\frac{1}{2}$, the equation for z_{63+i} involves one single unknown bit. Then it provides the value of x_{63+i} and consequently the value of x_i. Here the chain is of length 1, since we have to consider one extra equation to retrieve x_i. The equation for z_{63+i} can also involve two bits with probability $\frac{1}{2}$. Then we have to consider the equation of $z_{2\cdot 63+i}$, which can also either involve only one bit (we have a chain of length 2) or two bits and we have to consider more equations to solve. Each equation has a probability $\frac{1}{2}$ to involve 1 or 2 bits. Consequently the probability that a chain is of length n is $\frac{1}{2^{n+1}}$ and the probability that a chain is of length strictly larger than n is $\frac{1}{2^{n+1}}$.

We want to recover the values of $x_{17}, \ldots, x_{79}$. We have to build 64 different chains. Let us consider $L = 63 \cdot n$ bits of keystream. The probability that one of the chains is of length larger than n is less than $= 64 \cdot 2^{-n-1}$ and therefore less than 2^{-n+5}. If we want this probability to be bounded by 2^{-10}, then $n > 15$ and $L > 945$ suffices. Consequently a few thousands of keystream bits are required to retrieve the initial state of the NFSR and the complexity of the operation is bounded by $64 \cdot n$.

Since the internal state transition function associated to the special key and IV setup mode is one to one, the key can be efficiently derived from the NFSR and LFSR states at the beginning of the keystream generation by running this function backward.

6 Simulations and Results

To confirm that our cryptanalysis is correct, we ran several experiments. First we checked the bias ϵ of Section 3.1 by running the cipher with a known initial state of both the LFSR and the NFSR, computing the linear approximations, and counting the number of fulfilled relations for a very large number of relations. For instance we found that one linear approximation is satisfied 19579367 times out of 39060639, which gives an experimental bias of $2^{-9.63}$, to be compared with the theoretical bias $\epsilon = 2^{-9.67}$.

To check the two proposed LFSR reconstruction methods of Section 4, we considered a reduced version of Grain in order to reduce the memory and time required by the attack on a single computer: we shortened the LFSR by a factor of 2. We used an LFSR of size 40 with a primitive feedback polynomial and we reduced by two the distances for the tap entries of function h: we selected taps number 3, 14, 24, and 33, instead of 3, 25, 46, and 64 for Grain.

The complexity of the first technique for the actual Grain is 2^{55} which is out of reach of a single PC. For our reduced version, the complexity given by the formula of Section 4.2 is only 2^{35}. We exploited the 16 linear approximations

to derive relations colliding on the first 11 bits. Consequently the table of the Walsh transform is only of size 2^{29}. We used $15612260 \simeq 2^{23}$ relations, which corresponds to a false alarm probability of 2^{-29}. Our implementation needed around one hour to recover the correct value of the LFSR internal state on a computer with a Intel Xeon processor running at 2.5 GHz with 3 GB of memory. The Walsh transform computation took only a few minutes.

For the actual Grain, the second technique requires only 2^{43} operations which is achievable by a single PC. However it also requires 2^{42} of memory which corresponds to 350 GB of memory. We do not have such an amount of memory but for the reduced version the required memory is only 2^{29}. Since the complexity given by the formula of Section 4.3 is dominated by the required number of relations to detect the bias, our simulation has a complexity close to 2^{43}. In practice, we obtained a result after 4 days of computation on the same computer as above and $2.5 \cdot 10^{12} \simeq 2^{41}$ relations where considered and allowed to recover the correct LFSR initial state.

Finally, we implemented the method of Section 5 to recover the NFSR. Given the correct initial state of the LFSR, and the first thousand keystream bits, our program recovers the initialization of the NFSR in a few seconds for a large number of different initializations of both the known LFSR and unknown NFSR. We also confirmed the failure probability assessed in Section 5 for this method (which corresponds to the occurrence probability of at least one chain of length larger than 15).

7 Conclusion

We have presented a key-recovery attack against Grain which requires 2^{43} computations, 2^{42} bits of memory, and 2^{38} keystream bits. This attack suggests that the following slight modifications of some of the Grain features might improve its strength:

- Introduce several additional masking variables from the NFSR in the keystream bit computation.
- Replace the nonlinear feedback function g in such a way that the associated function g' be balanced (e.g. replace g by a 2-resilient function). However this is not necessarily sufficient to thwart all similar attacks.
- Modify the filtering function h in order to make it more difficult to approximate.
- Modify the function g and h to increase the number of inputs.

Following recent cryptanalysis of Grain including the key recovery attack reported here and distinguishing attacks based on the same kind of linear approximations as those presented in Section 3 [19] [26], the authors of Grain proposed a tweaked version of their algorithm [12], where the functions g and h' have been modified. This novel version of Grain appears to be much stronger and is immune against the statistical attacks presented in this paper.

We would like to thank Matt Robshaw and Olivier Billet for helpful comments.

References

1. M. Briceno, I. Goldberg, and D. Wagner. A pedagogical implementation of A5/1. Available at `http://jya.com/a51-pi.htm`, Accessed August 18, 2003, 1999.
2. A. Canteaut and M. Trabbia. Improved fast correlation attacks using parity-check equations of weight 4 and 5. In B. Preneel, editor, *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 573–588. Springer-Verlag, 2000.
3. V. Chepyzhov and B. Smeets. On a fast correlation attack on certain stream ciphers. In D. W. Davies, editor, *Advances in Cryptology—EUROCRYPT'91*, volume 547 of *Lecture Notes in Computer Science*, pages 176–185. Springer-Verlag, 1991.
4. M. W. Dodd. *Applications of the Discrete Fourier Transform in Information Theory and Cryptology*. PhD thesis, University of London, 2003.
5. ECRYPT. eSTREAM: ECRYPT Stream Cipher Project, IST-2002-507932. Available at `http://www.ecrypt.eu.org/stream/`, Accessed September 29, 2005, 2005.
6. P. Ekdahl and T. Johansson. Another attack on A5/1. In *Proceedings of International Symposium on Information Theory*, page 160. IEEE, 2001.
7. P. Ekdahl and T. Johansson. Another attack on A5/1. *IEEE Transactions on Information Theory*, 49(1):284–289, January 2003.
8. H. Englund and T. Johansson. A new simple technique to attack filter generators and related ciphers. In *Selected Areas in Cryptography*, pages 39–53, 2004.
9. H. Gilbert and P. Audoux. Improved fast correlation attacks on stream ciphers using FFT techniques. personnal communication, 2000.
10. J.D. Golić. Cryptanalysis of alleged A5 stream cipher. In W. Fumy, editor, *Advances in Cryptology—EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 239–255. Springer-Verlag, 1997.
11. M. Hell, T. Johansson, and W. Meier. Grain - A Stream Cipher for Constrained Environments. ECRYPT Stream Cipher Project Report 2005/001, 2005. `http://www.ecrypt.eu.org/stream`.
12. M. Hell, T. Johansson, and W. Meier. Grain - A Stream Cipher for Constrained Environments, 2005. `http://www.it.lth.se/grain`.
13. T. Johansson and F. Jönsson. Fast correlation attacks based on turbo code techniques. In *Advances in Cryptology—CRYPTO'99*, volume 1666 of *Lecture Notes in Computer Science*, pages 181–197. Springer-Verlag, 1999.
14. T. Johansson and F. Jönsson. Improved fast correlation attacks on stream ciphers via convolutional codes. In *Advances in Cryptology—EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 347–362. Springer-Verlag, 1999.
15. F. Jönsson. *Some Results on Fast Correlation Attacks*. PhD thesis, Lund University, Department of Information Technology, P.O. Box 118, SE–221 00, Lund, Sweden, 2002.
16. A. Joux, P. Chose, and M. Mitton. Fast Correlation Attacks: An Algorithmic Point of View. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 209–221. Springer-Verlag, 2002.
17. B. S. Jr. Kaliski and M. J. B. Robshaw. Linear Cryptanalysis Using Multiple Approximations. In Yvo G. Desmedt, editor, *Advances in Cryptology – CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 26–39. Springer-Verlag, 1994.

18. M. Matsui. Linear cryptanalysis method for DES cipher. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1993.
19. A. Maximov. Cryptanalysis of the "Grain" family of stream ciphers. In *ACM Transactions on Information and System Security (TISSEC)*, 2006.
20. W. Meier and O. Staffelbach. Fast correlation attacks on stream ciphers. In C.G. Günter, editor, *Advances in Cryptology—EUROCRYPT'88*, volume 330 of *Lecture Notes in Computer Science*, pages 301–316. Springer-Verlag, 1988.
21. W. Meier and O. Staffelbach. Fast correlation attacks on certain stream ciphers. *Journal of Cryptology*, 1(3):159–176, 1989.
22. W. Meier and O. Staffelbach. The self-shrinking generator. In A. De Santis, editor, *Advances in Cryptology—EUROCRYPT'94*, volume 905 of *Lecture Notes in Computer Science*, pages 205–214. Springer-Verlag, 1994.
23. M. Mihaljevic and J.D. Golić. A fast iterative algorithm for a shift register initial state reconstruction given the noisy output sequence. In J. Seberry and J. Pieprzyk, editors, *Advances in Cryptology—AUSCRYPT'90*, volume 453 of *Lecture Notes in Computer Science*, pages 165–175. Springer-Verlag, 1990.
24. NESSIE. New European Schemes for Signatures, Integrity, and Encryption. Available at `http://www.cryptonessie.org`, Accessed August 18, 2003, 1999.
25. W.T. Penzhorn and G.J. Kühn. Computation of low-weight parity checks for correlation attacks on stream ciphers. In C. Boyd, editor, *Cryptography and Coding - 5th IMA Conference*, volume 1025 of *Lecture Notes in Computer Science*, pages 74–83. Springer-Verlag, 1995.
26. M. Hassanzadeh S. Khazaei and M. Kiaei. Distinguishing Attack on Grain. ECRYPT Stream Cipher Project Report 2005/001, 2005. `http://www.ecrypt.eu.org/stream`.
27. T. Siegenthaler. Correlation-immunity of non-linear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, 30:776–780, 1984.
28. T. Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, 34:81–85, 1985.

Cryptanalysis of Pomaranch

Carlos Cid[1], Henri Gilbert[2] and Thomas Johansson[3]

[1] Information Security Group,
Royal Holloway, University of London
Egham, Surrey TW20 0EX, United Kingdom
carlos.cid@rhul.ac.uk
[2] France Télécom, R&D Division
38–40, rue du Général Leclerc
92794 Issy les Moulineaux, Cedex 9, France
henri.gilbert@francetelecom.com
[3] Dept. of Information Technology, Lund University,
P.O. Box 118, 221 00 Lund, Sweden
thomas@it.lth.se

Abstract Pomaranch [3] is a synchronous stream cipher submitted to eSTREAM, the ECRYPT Stream Cipher Project. The cipher is constructed as a cascade clock control sequence generator, which is based on the notion of jump registers. In this paper we present an attack which exploits the cipher's initialization procedure to recover the 128-bit secret key. The attack requires around 2^{65} computations. An improved version of the attack is also presented, with complexity 2^{52}.

1 Introduction

Pomaranch[1] is one of the 34 stream ciphers submitted to eSTREAM, the ECRYPT Stream Cipher Project [1]. The cipher is implemented as a binary one clock pulse cascade clock control sequence generator, and uses 128-bit keys and IVs of length between 64 and 112 bits [3]. The construction is based on the notion of *jump registers*.

Jump controlled LFSRs were introduced in [2] as alternative to traditional clock-controlled registers. In jump controlled LFSRs, the registers are able to move to a state that is more than one step ahead without having to step through all the intermediate states (thus the name jump registers). The main motivation for the proposal of jump registers is to construct LFSR-based ciphers that can be efficiently protected against side-channel attacks while preserving the advantages of irregular clocking.

2 Outline of Pomaranch

Pomaranch is depicted in Figure 1, where only the key stream generation phase is represented (called Key Stream Generation Mode). The cipher consists of nine cascaded jump registers R_1 to R_9. The jump registers are implemented as autonomous Linear Finite State Machine (LFSM), built on 14 memory cells, which behave either

[1] The cipher is also referred in the specification document [3] as Cascade Jump Controlled Sequence Generator (CJCSG).

as simple delay shift cells or feedback cells, depending on the value of the so-called Jump Control (JC) signal. At any moment, half of the cells in the registers are shift cells, while the other half are feedback cells. The initial configuration of cells is determined by the LFSM transition matrix A, and is used if the JC value is zero. If JC is one, all cells are switched to the opposite mode. This is equivalent to switching the transition matrix to $(A + 1)$ [3].

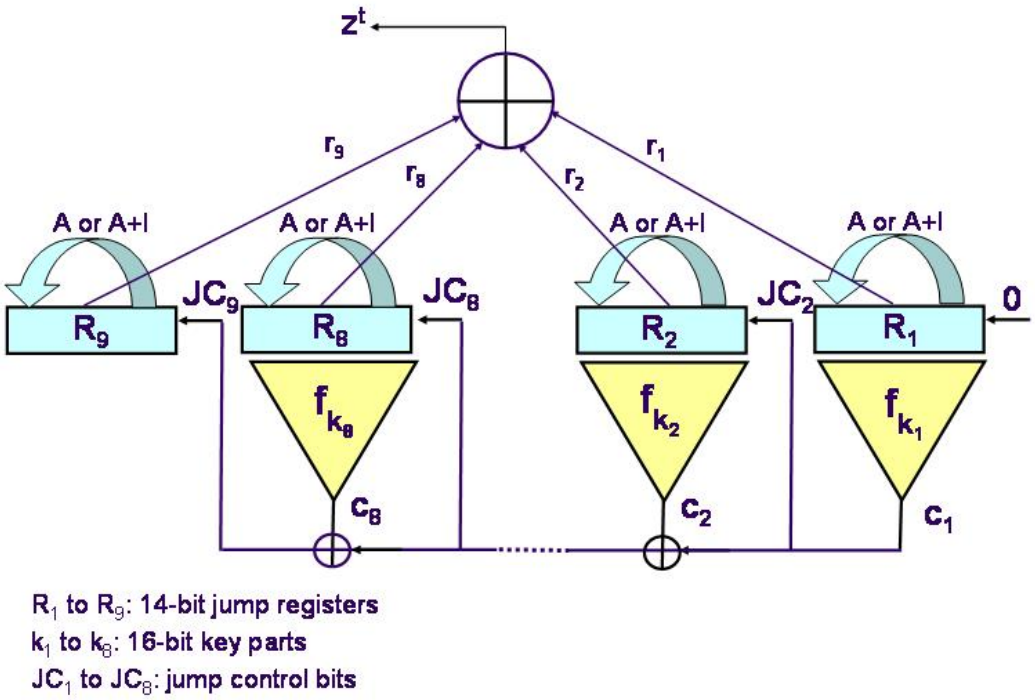


Figure1. The Pomaranch stream cipher

The 128-bit key K is divided into eight 16-bit subkeys k_1 to k_8. At time t, the current state of the registers R_1^t to R_8^t are non linearly filtered, using a function that involves the corresponding subkey k_i. These functions provide as output eight bits c_1^t to c_8^t, which are used to produce the jump control bits JC_2^t to JC_9^t controlling the registers R_2 to R_9 at time t, as following:

$$JC_i^t = c_1^t \oplus \ldots \oplus c_{i-1}^t \quad \text{for } i = 2, \ldots, 9.$$

The jump control bit JC_1 of register R_1 is permanently set to zero. The key stream bit z^t produced at time t is the XOR of nine bits r_1^t to r_9^t selected at fixed positions of the current register states R_1^t to R_9^t.

Key and IV Loading. During the cipher initialization, the content of registers R_1 to R_9 are first set to non-zero constant 14-bit values derived from π, then the

subkeys k_i are loaded and the registers are run for 128 steps in a special mode (called Shift Mode). The main difference between the Key Stream Generation Mode and the Shift Mode is that, in the latter the output of the filtering function of register R_i (denoted by c_i) is added to the feedback of register R_{i+1}, with the tap from cell 1 in the register R_9 being added to the register R_1, making then what can be seen as a "big loop". Note that the configuration of the jump registers do not change in this mode (they all operate as if $JC_i = 0$). This process ensures that the states of the registers R_1 and R_9 after this key loading phase depend upon the entire key K. We denote these states by R_1^K to R_9^K.

Next the IV is loaded into the registers. The IV can have any arbitrary length between 64 and 112 bits. If the IV length is shorter than 112 bits, it is expanded by cyclically repeating it until a length of exactly 112 bits is obtained. This new string is then loaded into the registers as described below. In the remaining of this paper, for the sake of simplicity, we assume that the IV length is exactly 112 bits.

The IV is loaded into the registers in the following manner: the 112-bit IV is split into eight 14-bit parts IV_1 to IV_8, which are XORed with the 14-bit states of registers R_1^K to R_8^K obtained at the end of the key loading. If any of the resulting states consists of 14 null bits, its lowest weight bit is set to one (this ensures that no state will be made up entirely of null bits[2]). The resulting register states R_1 to R_8 form together with R_9^K the nine initial states. We denote these resulting 14-bit state values by R_1^{-128} to R_9^{-128}. The key stream generation mode of Figure 1 is now activated, and the runup consists of 128 steps in which the produced key stream bits are discarded.

3 Description of the Attack

We have identified the following weakness in the Pomaranch IV initialization procedure: if for a given key K and IV value IV, we only modify the IV part IV_8 and keep the remaining parts IV_1 to IV_7 unchanged (thus obtaining a modified IV value IV'), on comparing the key stream generation under the key K with IV and IV', we have that for every $t \geq -128$

$$R_i^t(IV) = R_i^t(IV') \quad \text{for } i = 1, \ldots, 7 \ .$$

In other words, the Key and IV loading procedure does not diffuse all IV bits into the whole state of the generator. Consequently, if IV and IV' are chosen as above, the contributions from registers R_1 to R_7 cancel out on each key stream XOR $z^t(IV) \oplus z^t(IV')$, and we obtain the relation

$$z^t(IV) \oplus z^t(IV') = r_8^t(IV) \oplus r_8^t(IV') \oplus r_9^t(IV) \oplus r_9^t(IV').$$

[2] The Pomaranch specification does not mention this feature, which is described in the source code provided with the submission and has been confirmed by one of the designers [4]. We will show in the next section that, although the cipher can be attacked even if this feature is withdrawn, this represents an additional weakness that leads to improved attacks.

We now show how to exploit this weakness to recover the subkey k_8 of an unknown key K, in a chosen IV attack. Consider 3 distinct chosen IV values IV, IV' and IV'', which only differ by their part IV_8, IV'_8 and IV''_8. We can obtain the corresponding first m-bit key stream $z^t(IV)_{t=0\ \texttt{to}\ m-1}$, $z^t(IV')_{t=0\ \texttt{to}\ m-1}$, and $z^t(IV'')_{t=0\ \texttt{to}\ m-1}$, which in turn provide the pairwise XOR values

$$\begin{aligned}\delta^t &= z^t(IV) \oplus z^t(IV')_{t=0\ \texttt{to}\ m-1},\\ \delta'^t &= z^t(IV') \oplus z^t(IV'')_{t=0\ \texttt{to}\ m-1},\end{aligned}$$

In order to recover the value of k_8, we guess the following values:

- Subkey k_8:	16 bits;
- Registers R_8^K and R_9^K:	28 bits;
- $n_8 = \sharp\{t \in \{-128,..,-1\} \mid JC_8^t(IV) = 1\}$:	129 possible values;
- $n_9 = \sharp\{t \in \{-128,..,-1\} \mid JC_9^t(IV) = 1\}$:	129 possible values;
- $n'_9 = \sharp\{t \in \{-128,..,-1\} \mid JC_9^t(IV') = 1\}$:	129 possible values;
- $n''_9 = \sharp\{t \in \{-128,..,-1\} \mid JC_9^t(IV'') = 1\}$:	129 possible values.

The attack exploits the jump registers property that since the transition matrices A and $A+I$ commute, the transition matrix associated with a number s of steps can only take one of the at most $s+1$ values $A^p(A+I)^q$, with $p+q=s$. Due to this property, the knowledge of the values of (n_8, n_9, n'_9, n''_9) is sufficient to derive the R_8 and R_9 transition matrices of the form $A^{128-n}(A+I)^n$ associated with the 128-step runup for IV values IV, IV' and IV''. Note that although n_8, n_9, n'_9, n''_9 can take any of the 129 values in the $[0 \cdots 128]$ interval, their values are binomially distributed, so that in practice the 2^5-1 middle values in the interval $[49 \cdots 79]$ have an overwhelming occurrence probability.

Now since we have[3]

$$\begin{aligned}&R_8^{-128}(IV) = R_8^K \oplus IV,\\ &R_8^{-128}(IV') = R_8^K \oplus IV',\\ &R_8^{-128}(IV'') = R_8^K \oplus IV'',\\ &R_9^{-128}(IV) = R_9^{-128}(IV') = R_9^{-128}(IV'') = R_9^K,\end{aligned}$$

it follows that knowledge of R_8^K, R_9^K, n_8, n_9, n'_9 and n''_9 allows us to compute $R_8^0(IV)$, $R_8^0(IV')$, $R_8^0(IV'')$, $R_9^0(IV)$, $R_9^0(IV')$, and $R_9^0(IV'')$.

To test a $(k_8, R_8^K, R_9^K, n_8, n_9, n'_9, n''_9)$ assumption we need to compute the resulting values of $R_8^0(IV)$, $R_8^0(IV')$, $R_8^0(IV'')$, $R_9^0(IV)$, $R_9^0(IV')$, and $R_9^0(IV'')$ and iteratively try, for consecutive values of m, to guess the m-bit value $JC_8^t(IV)_{t=0\ \texttt{to}\ m-1}$ in order to derive the resulting values of $R_8^t(IV)$, $R_8^t(IV')$, $R_8^t(IV'')$, $R_9^t(IV)$, $R_9^t(IV')$, and $R_9^t(IV'')$. Following we verify whether the predicted values $(\delta^t, \delta'^t)_{t=0\ \texttt{to}\ m-1}$ are in agreement with the observed ones. The average number of m values to be tested until a wrong assumption is discarded (because no $JC_8^t(IV)_{t=0\ \texttt{to}\ m-1}$ m-tuple fits the observed values) is about 2.

[3] We are ignoring the cipher's non-zero state forcing feature at this stage.

Indeed, for a certain (k_8, R_8^K, R_9^K, n_8, n_9, n'_9, n''_9) assumption and a choice of $JC_8^t(IV)$, the pair (δ^t, δ'^t) can take one of four possible values. Assuming the values are randomly generated, there are three events to consider. First the case in which the pairs (δ^t, δ'^t) for both the choices of $JC_8^t(IV) = 0$ and $JC_8^t(IV) = 1$ are in agreement with the observed value. Its probability is 1/16, and it leaves us with two possible configurations that need to be further tested. The second event is when only one pair (δ^t, δ'^t) for either the choices of $JC_8^t(IV) = 0$ or $JC_8^t(IV) = 1$ is in agreement with the observed one. Its probability is 3/8, and it leaves us with one possible configuration that need to be further tested. The third event is when neither the pairs (δ^t, δ'^t) for the choices of $JC_8^t(IV) = 0$ and $JC_8^t(IV) = 1$ is in agreement with the observed one (i.e. the configuration is inconsistent). Its probability is 9/16, and no further tests using this configuration is necessary. Thus if X denotes the number of tests we need to perform, then

$$E(X) = 1 + \frac{1}{16} \cdot 2 \cdot E(X) + \frac{3}{8} \cdot 1 \cdot E(X) + \frac{9}{16} \cdot 0 \cdot E(X),$$

and $E(X) = 2$.

The attack described above allows us to recover the value of k_8. Its complexity is bounded over by $2^{16} \times 2^{28} \times (2^5)^4 \times 2 = 2^{65}$. Note that the attack also recovers the correct values for R_8^K and R_9^K. To recover the other key parts, we can proceed as following: repeat the same attack for another value of (IV, IV', IV''), call it $(\overline{IV}, \overline{IV'}, \overline{IV''})$, such that IV and $\overline{IV}$ only differ by their part IV_7 and $\overline{IV_7}$. Since we know already k_8, R_8^K and R_9^K, this second attack can be mounted much faster. Finally, we can guess the values of R_7^K and n_7 and check whether there exists a sequence $JC_7^t(IV)_{t=0 \texttt{ to } m-1}$ that is consistent with the already known sequences $JC_8^t(IV)_{t=0 \texttt{ to } m-1}$ and $JC_8^t(\overline{IV})_{t=0 \texttt{ to } m-1}$. This can be done for all the remaining key parts, until the entire key K has been recover. The complexity of the entire attack remains about 2^{65}.

Improved Attack. Note that so far we have not exploited the non-zero state forcing feature of Pomaranch, and the above attack works whether this feature is present or not. We now show that this feature results in a low complexity distinguisher, and also allows us to reduce the complexity of the key derivation procedure described above.

The distinguisher works as following: given an unknown key K, we can try the 2^{14} possible IV values obtained by keeping (say) IV_1 to IV_7 unchanged and taking all possible values for (say) IV_8. Now two of these 2^{14} IVs result in exactly the same states R_1^{-128} to R_9^{-128} after key and IV loading, namely the IV value resulting on a 14-bit R_8 state equal to zero (which will have one bit switched to 1 by the cipher non-zero state forcing procedure), and the IV value derived from the former one by swapping the same bit position. The key streams for these two IV values are exactly the same. If the key stream is sufficient long (e.g. more than 27 bits in order for collisions of a pair of IV values to be unlikely), this provides an efficient chosen IV

distinguisher of distinguishing probability close to 1, requiring generation of only 2^{14} key stream sequences of length (say) 64 bits each.

This distinguisher can be used to improve the key derivation attack described above. Indeed, the distinguisher allows us to recover the register value R_8^K up to one single bit, so that a factor of 2^{13} can be saved in the search of (k_8, R_8^K, R_9^K, n_8, n_9,n'_9, n''_9), and the attack complexity is reduced to 2^{52}.

4 Conclusion

We showed in this paper how to mount a chosen IV attack to recover the secret key of Pomaranch with complexity much lower than the one expected with 128-bit keys. The attack exploits a weakness in the cipher initialization procedure, namely the process does not diffuse all the IV bits into the whole state of the key stream generator. By exploiting another feature of the IV loading, we were able to substantially improve the attack.

References

1. eSTREAM, the ECRYPT Stream Cipher Project. `http://www.ecrypt.eu.org/stream/`.
2. C. J. Jansen. Streamcipher Design: Make your LFSRs jump! In *SASC, Workshop Record, ECRYPT Network of Excellence in Cryptology*, pages 94–108, 2004.
3. C.J. Jansen, T. Helleseth, and A. Kholosha. Cascade Jump Controlled Sequence Generator (CJCSG). In *SKEW, Workshop Record, ECRYPT Network of Excellence in Cryptology*, 2005.
4. A. Kholosha. Personal Communication.

Resistance of SNOW 2.0 against Algebraic Attacks

Olivier Billet and Henri Gilbert

France Télécom R&D
38–40, rue du Général Leclerc
92794 Issy les Moulineaux Cedex 9 — France
{olivier.billet,henri.gilbert}@francetelecom.com

Abstract. SNOW 2.0, a software oriented stream cipher proposed by T. Johansson and P. Ekdahl in 2002 as an enhanced version of the NESSIE finalist SNOW 1.0, is usually considered as one of the strongest stream ciphers designed so far. This paper investigates the resistance of SNOW 2.0 against algebraic attacks. This is motivated by the fact that the main source of non-linearity in SNOW 2.0 comes from a permutation build upon the AES S-box, which inputs and outputs are well known to be related by numerous quadratic equations. We show that a slightly modified version of SNOW 2.0 is susceptible to an algebraic attack with time complexity about 2^{50}, and which requires no more than 1000 words of output. We then explore various ways to extend this attack to the actual stream cipher.

Keywords: SNOW 2.0, stream ciphers, algebraic attacks.

1 Introduction

SNOW 2.0 [9] is a software oriented stream cipher proposed by T. Johansson and P. Ekdahl in 2002 as a replacement of an earlier version named SNOW 1.0 [8]. SNOW 2.0 is generally considered as one of the strongest stream cipher designs currently available, together with ciphers like the Shrinking Generator [3], SCREAM [12], and carefully initialized versions of RC4 [11]. SNOW 1.0 was one of the finalists of the European project NESSIE. One of the main reasons for the rejection of SNOW 1.0 from the NESSIE portfolio of recommended cryptographic primitives—which eventually lacked a stream cipher design—was the discovery of a statistical distinguisher with time complexity 2^{95} due to Coppersmith *et al.* [2]. A key recovery attack of expected complexity 2^{224} against SNOW 1.0 was also found H. Hawkes and G. Rose [13]. Both attacks require a known key stream length of 2^{95}. Those attacks motivated the introduction of a new version of SNOW, SNOW 2.0 [9], which eliminated at the same time some other minor flaws. The most characteristic features of SNOW 2.0 are

– an LFSR defined over a large field, GF(2^{32}) with a new feedback polynomial as to avoid the flaws detected in the previous design, SNOW 1.0;

– a finite state machine involving two non-linearly updated memory registers of size 32 bits. The non-linearity results from two modular additions, and a 32 bit to 32 bit function $\mathcal{S}$ based on the well-known and highly studied AES S-box [14].

The best attack against SNOW 2.0 so far is a distinguishing attack of complexity 2^{225} due to D. Watanabe, A. Biryukov, C. de Cannière [16], and requires 2^{225} key stream words. It consists in an enhanced variant of the linear masking method [2] which exploits the feedback polynomial of the LFSR over GF(2^{32}) instead of requiring low weight multiples with GF(2) coefficients, as in the original attack.

This paper investigate the resistance of SNOW 2.0 against algebraic attacks. Although the relevance of such attacks in the context of block ciphers—like AES, for instance—remains unclear, it has been proved to be of interest in the context of regularly clocked stream ciphers [5, 6, 1]. Considering that SNOW 2.0 is a regularly clocked stream cipher which non-linearity mainly rests on the AES S-box, it seems natural to probe its resistance against algebraic attacks.

We first establish that if the function $\mathcal{S}$ based on the AES S-box was the only source of non-linearity, SNOW 2.0 would be vulnerable to a very efficient algebraic attack. More precisely, we consider the close variant of SNOW 2.0 obtained by replacing the two modular additions by additions over GF(2^{32}), leaving the other parts (LFSR, $\mathcal{S}$ function based on AES S-box...) unchanged. We explain how to recover the initial state of the LFSR using a linearization attack of complexity about 2^{50}, requiring no more than 1000 clocks of key stream. We then examine the consequences of this result for the actual stream cipher, and show that the knowledge of a small key stream sequence (slightly more than 17 key stream outputs) allows the attacker to write a rather large—still, overdetermined and sparse—system of quadratic equations. Solving of such sparse quadratic systems and its complexity are not yet fully understood, but there is a growing research effort on the subject, due in large part to its potential application to the AES block cipher standard [15, 7].

The paper is organized as follows. Section 2 provides a brief description of the SNOW 2.0 stream cipher. Section 3 describe the algebraic attack against a slightly modified SNOW 2.0, while Sec. 4 analyzes different means to extend this attack to the actual stream cipher.

2 Description of SNOW

The stream cipher SNOW 2.0 is made of a linear feedback shift register (LFSR) with sixteen 32 bit words and a finite state machine (FSM) with two 32 bit memory registers. SNOW 2.0 mixes additions over GF(2^{32}) hereafter denoted by '$\oplus$', together with additions modulo 2^{32} denoted by '$\boxplus$'.

2.1 The Linear Feedback Shift Register

The linear feedback shift register (LFSR) is defined over GF(2^{32}), which allows good performance for software implementations. It is made of sixteen 32 bit

words, thus exhibiting 512 bit internal state size. The field of definition can be further described as $\mathrm{GF}(2^{32}) = \mathrm{GF}(2)(\alpha, \beta)$, where β is a root of the $\mathrm{GF}(2)[x]$ polynomial $x^8 + x^7 + x^5 + x^3 + 1$, and α is a root of the $\mathrm{GF}(2^8)[x]$ polynomial $x^4 + \beta^{23}x^3 + \beta^{245}x^2 + \beta^{48}x + \beta^{239}$. The feedback polynomial is then defined by

$$\alpha x^{16} + x^{14} + \alpha^{-1}x^5 + 1 \ .$$

This choice of a tower extension to describe $\mathrm{GF}(2^{32})$ is justified by the simple expression of the feedback polynomial in this context: it only consists of byte `shift`s/`xor`s, since each word can be expressed on the base $\{\alpha^3, \alpha^2, \alpha, 1\}$.

In the following, the word that the LFSR outputs at clock t is denoted by s^t.

2.2 Finite State Machine

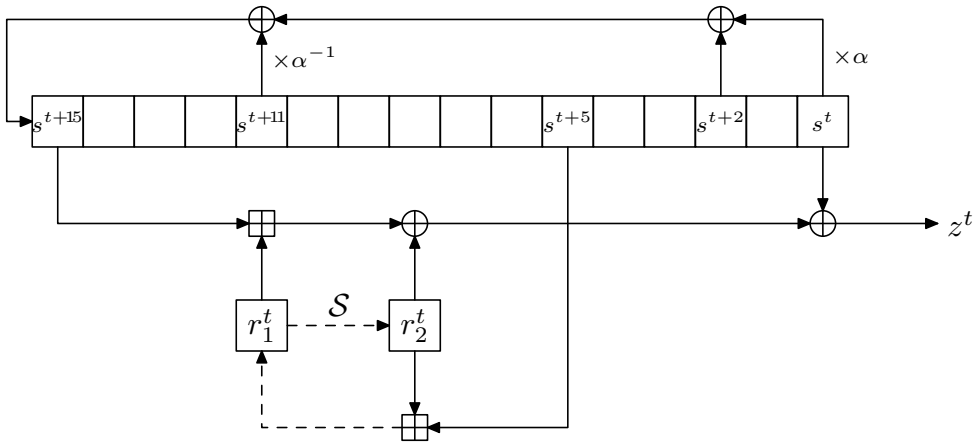


Fig. 1. SNOW 2.0

The other part of the stream cipher is a finite state machine (FSM), which contains two 32 bit memory registers r_1 and r_2. This FSM is intended to produce the non-linear part of the stream cipher. To this end, it contains a non-linear 32 bit to 32 bit non-linear bijection denoted by $\mathcal{S}$, based on the AES S-box [14], and defined as follows. If we decompose the register r_1 at clock t on the base $\{\alpha^3, \alpha^2, \alpha, 1\}$ as explained in the above section as $r_1^t = a_1^t\alpha^3 + b_1^t\alpha^2 + c_1^t\alpha + d_1^t$, and similarly the register r_2 at next clock as $r_2^{t+1} = a_2^{t+1}\alpha^3 + b_2^{t+1}\alpha^2 + c_2^{t+1}\alpha + d_2^{t+1}$, the rule $r_2^{t+1} = \mathcal{S}(r_1^t)$ to update r_2 from r_1 can be defined as

$$\begin{bmatrix} a_2^{t+1} \\ b_2^{t+1} \\ c_2^{t+1} \\ d_2^{t+1} \end{bmatrix} = \begin{bmatrix} X & X+1 & 1 & 1 \\ X+1 & 1 & 1 & X \\ 1 & 1 & X+1 & X \\ 1 & X+1 & X & 1 \end{bmatrix} \times \begin{bmatrix} S(a_1^t) \\ S(b_1^t) \\ S(c_1^t) \\ S(d_1^t) \end{bmatrix}$$

where S represents the AES S-box, the matrix is the one `MixColumn` step in AES when its four input bytes are considered as elements of the $\mathrm{GF}(2^8)$ definition of

the AES, i.e. $\mathrm{GF}(2)[X]/(X^8 + X^4 + X^3 + X + 1)$. This completes the definition of the non-linear function $\mathcal{S}$.

Now the rule to update the register r_1 from r_2 is given by $r_1^{t+1} = r_2^t \boxplus s^{t+5}$. The output of the FSM at clock t, which we denote by F^t, is finally defined by $F^t = (r_1^t \boxplus s^{t+15}) \oplus r_2^t$. Let us summarize the behavior of the FSM below

$$\begin{cases} r_2^{t+1} \stackrel{\text{def}}{=} \mathcal{S}\,(r_1^t)\ , \\ r_1^{t+1} \stackrel{\text{def}}{=} r_2^t \boxplus s^{t+5}\ , \\ \quad F^t \stackrel{\text{def}}{=} (r_1^t \boxplus s^{t+15}) \oplus r_2^t\ . \end{cases}$$

2.3 Output of the Stream Cipher

The output of SNOW 2.0 is a classical example of linear masking, that is the output of the LFSR is `xor`ed with the output of the (non-linear) FSM. Thus the key stream output at clock t, which we henceforth denote by z^t, is defined by $z^t = s^t \oplus F^t$, or equivalently by

$$z^t = (s^{t+15} \boxplus r_1^t) \oplus r_2^t \oplus s^t\ .$$

2.4 Key Initialization

The stream cipher SNOW 2.0 can be used with 128 bit or 256 bit keys. For the key initialization, the LFSR is loaded with the secret key K, a publicly known initialization vector IV, and the two memory registers are set to zero. The cipher is then clocked 32 times in a special mode where no key stream is produced, and the FSM output is injected in the feedback value

$$s^{t+16} \stackrel{\text{def}}{=} \alpha^{-1} s^{t+11} \oplus s^{t+2} \oplus \alpha s^t \oplus F^t\ .$$

The cipher is then switched into the normal mode described in 2.3, but the first output of the keystream is discarded.

3 Attack on a modified version of SNOW 2.0

We now describe the algebraic attack against the close variant of SNOW 2.0 where modular additions '$\boxplus$' are replaced with `xor`s '$\oplus$' in its description, while everything else remains identical.

3.1 Deriving the System

Let us construct a system of equation in the LFSR's initial state variables alone, and solve it. In order to do so, we need to eliminate the memory from the set of

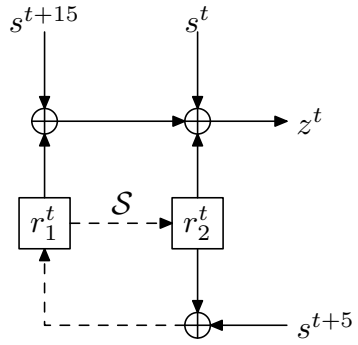


Fig. 2. a variant of SNOW 2.0

equations. This is done by looking at the key stream generation and the update rule for the register r_1. Indeed, combining those relations

$$\begin{cases} z^t = s^{t+15} \oplus r_1^t \oplus s^t \oplus r_2^t \ , \\ r_1^t = r_2^{t-1} \oplus s^{t+4} \ , \end{cases}$$

which can be further reduced into

$$r_2^t = r_2^{t-1} \oplus z^t \oplus s^{t+15} \oplus s^{t+4} \oplus s^t \ ,$$

we get an expression of the register r_2, for any clock t, which only involves the key stream, the LFSR initial state variables $s^0, \ldots, s^{15}$, and the initial state r_2^0 of the register r_2. Put it in equation, for each clock t, there are known binary coefficients ϵ_t^i such that

$$r_2^t = r_2^0 \bigoplus_{i=0}^{t} z^i \bigoplus_{j=0}^{15} \epsilon_t^j s^j \ .$$

Let us assign $t = 0$ to the clock of the first key stream output. One easily checks that the register r_1, updated against the rule $r_1^{t+1} = r_2^t \oplus s^{t+5}$, benefits from the same property. (Note that the initial state of the register r_1 can be derived from the knowledge of r_2^0 and the relation $r_1^0 = r_2^0 \oplus s^0 \oplus s^{15} \oplus z^0$.) In other words, we got rid of the memory, since for any clock $t > 0$, it can be expressed linearly in terms of the initial state variables and the initial memory value r_2^0.

The property that the knowledge of the key stream allows to track the linear functions of r_2^0, $s^0, \ldots, s^{15}$ contained in r_1 and r_2 may be visualized on Fig. 3. (Note that a similar property involving non-linear expressions, also holds for the actual stream cipher.) Now we need to derive some equations involving the initial LFSR state variable, and r_2^0. Those are obtained from the second update rule, namely $r_2^{t+1} = \mathcal{S}(r_1^t)$. Since the non-linear function $\mathcal{S}$ maps the four bytes of r_1^t to the four bytes of r_2^t via the AES S-box, and then mixes the resulting bytes linearly at the bit level, we are able to write down 156 linearly independent quadratic equations relating the bits of $r_1^t = r_2^t \oplus s^t \oplus s^{t+15} \oplus z^t$ and the bits of r_2^{t+1}.

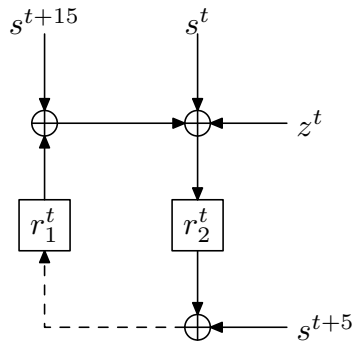


Fig. 3. tracking memory registers r_1 and r_2

To explain why, it is suffices to recall the well known property of the AES S-box: there are linearly independent quadratic equations involving the S-box input and output bits. To see why, just write $S = A \circ I$, where A denotes the GF(2)-affine mapping, and I maps zero to zero and equals the inversion over GF(2^8) everywhere else. Then if $w = S(u) = A \circ I(u)$ and $v = I(u)$, we get

$$uv = 1\,, \quad u^2v = u\,, \quad uv^2 = v\,, \quad uv^4 = v^3\,, \quad u^4v = u^3\,,$$

the first equation being true for all bits, except the least significant one, because $I(0) = 0$. And since $x \mapsto x^2$ is GF(2)-linear, we deduce that the bits of u and v are related by $5 \times 8 - 1 = 39$ quadratic equations. Now this property obviously remains true after the application of A.

Going back to $\mathcal{S}$, we are now able to write $4 \times 39 = 156$ quadratic equations relating r_1^t and r_2^{t+1}. Remember here that both registers are linear functions of the LFSR initial state variables $s^0, \ldots, s^{15}$ and r_2^0, for any $t > 0$.

3.2 Recovering the Initial State and the Key

The problem of recovering the initial state of the LFSR can be directly translated into that of solving the system of quadratic equations constructed in the previous section. However, two distinct strategies can be devised.

First one may wish to entirely linearize the system. The number of monomials involved are, in the worst case, all monomials of degree up to 2 involving a total of 512+32 variables over GF(2). There are $N = \sum_{k=0}^{2} \binom{544}{k}$ such variables, which is slightly more than 2^{17}. To be able to linearize the system, we thus need to get about $N/156 < 951$ key stream words, that is we need to get less than 1000 consecutive output words of the stream cipher—under the usual assumption that the small number of linear dependencies occurring before a full rank system is obtained do not much affect the required number of outputs. A very conservative estimation of the time complexity to solve the system is the cube of the number of variables, that is about 2^{51}.

One could also want to solve the system of quadratic equations as soon as it is overdefined and without requiring it to be linearized, since there exists algorithms especially designed for this task [7, 10]. We note that in such case,

only slightly more than 17 key stream output words are needed for the system to be overdefined. In this case however, the complexity to solve the system is not fully understood in the current state of the art, and is expected to be notably higher than for solving a linearized system.

Once the initial state $s^0, \ldots, s^{15}$, and r_2^0 have been recovered using the above linearization method, r_1^0 is given by the relation $r_1^0 = r_2^0 \oplus s^0 \oplus s^{15} \oplus z^0$, and so the entire state of the cipher at clock $t = 0$ is known. In order to derive the secret key K—and thus be able to predict the key stream sequence for other IVs—it suffices to run the cipher backward, one clock in the normal operation mode, then 32 clocks in the special feedback mode. It is easy to see that the state transitions of the SNOW 2.0 in both normal and special modes are invertible. Therefore, we are able to get the LFSR state at the initialization time, wich gives, from the knowledge of IV, the value of the secret key K.

4 Implications for SNOW 2.0

In this section, we seek for an extension of our attack described in Sec. 3 to the actual SNOW 2.0 stream cipher. We mainly identified two possible methods to take into account the extra source of non-linearity introduced by the modular additions of the FSM. The first one is to guess the carries' values for a small number of consecutive clocks. The other one consists in introducing new variables for the carries, and building a system of quadratic equations involving the LFSR initial state variables, the FSM initial memory variables and the extra carry variables. As will be shown in the sequel, the first method appears to require an impractical amount of guessing, while the second one seems more promising at first glance from a cryptanalytic point of view.

In the following, the carry corresponding to the addition $s^{t+15} \boxplus r_1^t$ of the FSM will be denoted by c_1^t, while the carry corresponding to the addition $s^{t+5} \boxplus r_2^t$ of the FSM will be denoted by c_2^t. Hence,

$$\begin{aligned} s^{t+15} \boxplus r_1^t &= s^{t+15} \oplus r_1^t \oplus c_1^t \ , \\ s^{t+5} \boxplus r_2^t &= s^{t+5} \oplus r_2^t \oplus c_2^t \ . \end{aligned} \tag{1}$$

As in previous section we denote by $t = 0$ the clock of the first observable output of SNOW 2.0 and call initial state the state of the LFSR at $t = 0$.

4.1 Guessing the Carries

This method strives to take benefit of the specificities of the carry bits' distribution occurring in modular additions. According to Eq. 1, we can track affine functions of r_2^0, s^0, $\ldots$, s^{15} contained in the memory registers r_1 and r_2 in the same way as done in Sec. 3—and then, apply the attack therein described—just by guessing the values of the carries c_1^t and c_2^t for about 16 consecutive clocks. The single difference with Sec. 3 is that the expressions of r_1^t and r_2^t now involve constant terms from the guessed carry values. However, due to the very particular distribution of carry bits, the cost of one guess is far less than 2^{32}.

Actually, it can be shown that the most probable carry—i.e. with no carry at all during the addition—has one chance over $\left(\frac{3}{4}\right)^{31}$ to occur. Indeed, this will happen when any two matching bits are not simultaneously 1, which represents three possibilities out of four. Thus a rough estimation for an upper bound on the probability to make a right guess for the carries c_1 and c_2 during 16 consecutive clocks is $\left(\frac{3}{4}\right)^{31\times 2\times 16}$. As it is much less than 2^{-256}, this approach seems impractical.

4.2 Quadratic System with Carry Variables

This second method consists in building a system of quadratic equations describing the actual SNOW 2.0 stream cipher. To this end, it suggests to introduce new GF(2) variables for the carry bits of the two modular additions '$\boxplus$' at each clock. This results in gathering quadratic equations during slightly more than 17 clocks—for the system to be overdetermined—and trying to solve the corresponding system.

Deriving the set of quadratic equations goes along the lines of the method exposed in Sec. 3. Indeed, just inserting the carries due to modular additions gives

$$\begin{cases} z^t = c_1^t \oplus s^{t+15} \oplus r_1^t \oplus s^t \oplus r_2^t \ , \\ r_1^t = c_2^{t-1} \oplus r_2^{t-1} \oplus s^{t+4} \ , \end{cases}$$

which is this time reduced into

$$r_2^t = r_2^{t-1} \oplus z^t \oplus s^{t+15} \oplus s^{t+4} \oplus s^t \oplus c_1^t \oplus c_2^{t-1} \ .$$

Eventually, we come to the fact that the memory registers can be expressed at any clock $t > 0$ as a linear combination of the initial LFSR state variables, the initial value r_2^0 of the register r_2, and all the carry bits occurring between clock 0 and clock t. The $(i+1)$th carry bit in the modular addition of two 32 bit words x and y can be defined as the majority of the ith bits of x, y, and ith carry bit. For each clock t, Eq. 1 thus implies

$$\begin{aligned} & c_{1,[0]}^t = 0 \\ 0 \le i < 32, \qquad & c_{1,[i+1]}^t = s_{[i]}^{t+15} r_{1,[i]}^t \oplus s_{[i]}^{t+15} c_{1,[i]}^t \oplus c_{1,[i]}^t r_{1,[i]}^t \ , \end{aligned}$$

as well as

$$\begin{aligned} & c_{2,[0]}^t = 0 \\ 0 \le i < 32, \qquad & c_{2,[i+1]}^t = s_{[i]}^{t+5} r_{2,[i]}^t \oplus s_{[i]}^{t+5} c_{2,[i]}^t \oplus c_{2,[i]}^t r_{2,[i]}^t \ , \end{aligned}$$

where $x_{[i]}$ denotes the ith bit of the 32 bit word x.

Of course, we have to add to these the quadratic equations holding between the registers r_1 and r_2. As stated above in Sec. 3, there are 156 such equations at each clock t, but this time involving the LFSR initial state variables, the variables for the bits of r_2^0, and all the carries' bits.

Let us now count the number of variables that appear in the system after n consecutive clocks. There are the 512 variables from the LFSR initial state, the 32 variables from r_2^0, plus for each clock, 62 carry bit variables. Hence, a total of $544 + 62n$ variables. On the other hand, there are $156n$ equations coming from the relation $\mathcal{S}$, and $62n$ equations from the definition of each carry bit, all at most quadratic, which amounts to a total of $218n$ equations. Hence the minimum value $n = 17$ for the system to be overdetermined, gives a total of 3706 equations with 1598 variables. For larger value of n, the system is more overdefined, but the equations to variables ratio is asymptotically bounded above by $\frac{7}{2}$.

The above system has been derived as to minimize the number of variables, not to maximize its sparsity. One can easily see that only the equations defining the carry bits are extremely sparse. Alternatively, one might write an equivalent, still overdefined and much more sparse system, by introducing the auxiliary variables r_1^t and r_2^t and their related linear equations, at the expense of increasing the number of variables. The system would have $544+126n$ variables, $282n$ quadratic or linear equations, and about the same sparsity as the equations on the AES block cipher. Its intractability remains, as in the case of the AES, an open question.

5 Conclusion

We exposed in this paper a very efficient attack against a close variant of the stream cipher SNOW 2.0. Various ways to extend this attack to the actual SNOW 2.0 design were also tried. The key search problem for the actual SNOW 2.0 was shown to be reducible to the solving of an overdetermined system of quadratic equations, the complexity of which remains unknown nowadays.

References

1. J. Y. Cho and J. Pieprzyk. Algebraic attacks on SOBER-t32 and SOBER-128. In W. Meier and B. K. Roy, editors, *Fast Software Encrytion – FSE 2004*, Lecture Notes in Computer Science. Springer-Verlag, 2004.
2. D. Coppersmith, S. Halevi, and C. S. Jutla. Cryptanalysis of stream ciphers with linear masking. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, Lecture Notes in Computer Science, pages 515–532. Springer-Verlag, 2002.
3. D. Coppersmith, H. Krawczyk, and Y. Mansour. The shrinking generator. In D. R. Stinson, editor, *Advances in Cryptology – CRYPTO '93*, Lecture Notes in Computer Science, pages 22–39. Springer-Verlag, 1993.
4. N. T. Courtois. Algebraic attacks on combiners with memory and several outputs. Cryptology ePrint Archive, Report 2003/125, 2003. http://eprint.iacr.org/.
5. N. T. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. Springer-Verlag, 2003.
6. N. T. Courtois and W. Meier. Algebraic attacks on stream ciphers with linear feedback. In E. Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer-Verlag, 2003.

7. N. T. Courtois and J. Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In Y. Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer-Verlag, 2002.
8. P. Ekdahl and T. Johansson. SNOW – a new stream cipher. Submission can be downloaded at `http://www.cryptonessie.org`, 2000.
9. P. Ekdahl and T. Johansson. A new version of the stream cipher SNOW. In K. Nyberg and H. M. Heys, editors, *Selected Areas in Cryptography – SAC 2002*, Lecture Notes in Computer Science, pages 47–61. Springer-Verlag, 2002.
10. J.-C. Faugère. A New Efficient Algorithm for Computing Groebner Bases without Reduction to Zero (F5). In *Proceedings of ISSAC*, pages 75–83. ACM Press, 2002.
11. S. R. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the key scheduling algorithm of rc4. In S. Vaudenay and A. M. Youssef, editors, *Selected Areas in Cryptography – SAC 2001*, Lecture Notes in Computer Science, pages 1–24. Springer-Verlag, 2001.
12. S. Halevi, D. Coppersmith, and C. S. Jutla. SCREAM: A software-efficient stream cipher. In J. Daemen and V. Rijmen, editors, *Fast Software Encrytion – FSE 2002*, Lecture Notes in Computer Science, pages 195–209. Springer-Verlag, 2002.
13. P. Hawkes and G. G. Rose. Guess-and-determine attacks on snow. In K. Nyberg and H. M. Heys, editors, *Selected Areas in Cryptography – SAC 2002*, Lecture Notes in Computer Science, pages 37–46. Springer-Verlag, 2002.
14. National Institute of Standards and Technology. Advanced encryption standard. FIPS publication 197, 2001.
`http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.
15. M. J. B. Robshaw and S. Murphy. Essential Algebraic Structure within the AES. In M. Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2002.
16. D. Watanabe, A. Biryukov, and C. de Cannière. A distinguishing attack on SNOW 2.0 with linear masking method. In M. Matsui and R. Zuccherato, editors, *Selected Areas in Cryptography – SAC 2003*, Lecture Notes in Computer Science, pages 222–233. Springer-Verlag, 2003.

QUAD: a Practical Stream Cipher with Provable Security

Côme Berbain[1], Henri Gilbert[1], Jacques Patarin[2]

[1] France Telecom Research and Development,
38-40 rue du Général Leclerc, F-92794 Issy-les-Moulineaux, France.
[2] Université de Versailles,
45 avenue des Etats-Unis, F-78035 Versailles cedex, France.

Abstract. We introduce a practical stream cipher with provable security named QUAD. The cipher relies on the iteration of a multivariate quadratic system of m equations in $n < m$ unknowns over a finite field. The security of the keystream generation of QUAD is provably reducible to the conjectured intractability of the MQ problem, namely solving a multivariate system of quadratic equations. Our recommended version of QUAD uses a 80-bit key, 80-bit IV and an internal state of $n = 160$ bits. It outputs 160 keystream bits ($m = 320$) at each iteration until 2^{40} bits of keystream have been produced.

1 Introduction

Stream ciphers represent, together with block ciphers, one of the two main classes of symmetric encryption algorithms. Generally speaking stream ciphers seem to allow faster encryption and to require lower computing resources than block ciphers, and the fastest known stream ciphers (e.g. SEAL, RC4, SNOW 2.0, the Shrinking Generator) are indeed significantly faster in software than an efficient block cipher such as AES [27]. However, the design of secure stream ciphers is not currently as well understood as the design of secure block ciphers. The state of the art of the cryptanalysis of stream ciphers, e.g. LFSR based stream ciphers, has evolved significantly over the last ten years and many recent proposals still suffer from security weaknesses. This is illustrated by the fact that none of the candidate stream ciphers submitted to the call for cryptographic primitives of the European project NESSIE were retained since attacks more efficient than exhaustive search were found for all candidates during the evaluation period. This is also illustrated by the ongoing eSTREAM [11] call for stream ciphers proposals of the European project ECRYPT. Stream ciphers complying with two main profiles have been called for, namely stream ciphers allowing much faster

[0] The work described in this paper has been supported by the French Ministry of Research RNRT X-CRYPT project and by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

software encryption than existing block ciphers (profile 1) and stream ciphers requiring much lower resources for hardware implementation than existing block ciphers (profile 2). However, more than one third of the 34 submitted stream ciphers, which cover these two profiles, have already been shown to be insecure.

Our aim is to propose a practical cipher with unusually strong security arguments. The novel stream cipher we propose was designed with another trade-off between security, speed and computing resources than reflected by the eSTREAM profiles 1 and 2. We slightly relax the requirements on speed and computing resources, i.e. we only require a stream cipher that is sufficiently fast for most practical purposes. But we introduce an unusually strong security requirement for symmetric cryptography (which is out of reach of the current state of the art for block ciphers), namely that the security of the cipher be provably reducible to the conjectured intractability of a well-known and studied mathematical problem. The security of the novel stream cipher is provably reducible to the intractability of the MQ problem [15], which consists of finding a solution (if any) to a multivariate quadratic system of m quadratic equations in n variables over a finite field $GF(q)$, typically $GF(2)$. The MQ problem is conjectured to be difficult for suitably chosen values of n and m. In general the associated decision problem is known to be NP-complete even in the case where the considered field is $GF(2)$, and moreover no efficient algorithm to solve MQ with a significant success probability is known to exist for sufficiently large values of n (say $n > 100$) when the quadratic equations are randomly chosen. The implementation complexity of our stream cipher is reasonable and the encryption speed (4.6 Mbit/s for a software implementation in C on a standard PC), though lower than AES, is more than sufficient for many practical purposes.

Constructing a provably secure stream cipher is not a novel topic. However, designing a practical provably secure stream cipher is an open problem. Following seminal work by Shamir, Blum and Micali [4], Yao [30], Levin and Goldreich [25] in the 80's, considerable research effort has been dedicated to the construction of provably secure pseudo-random number generators (PRNG) that expand a short seed (e.g. a key) into a larger bit string. This can be used as the keystream for encryption purposes. Available security results typically state that if the iterated function underlying the construction of a number generator satisfies suitable one-wayness properties, then the generator is a secure PRNG, i.e. its L-bit output is computationally indistinguishable from the uniform distribution over $\{0,1\}^L$. This research effort has led to remarkable generic results, e.g. the proof by Impagliazzo, Levin, Luby and Håstad [21] that a secure PRNG can be constructed based upon any one way function (OWF). It has also led to provably secure PRNG constructions based on the conjectured intractability of specific problems. The first provably secure PRNG was introduced by Blum and Micali [4] and relates the security of the PRNG to the one-wayness of exponentiation modulo a prime number. The provably secure PRNG proposed by L. Blum, M. Blum and M. Shub [3] exploits the conjectured intractability of quadratic residuosity modulo Blum integers. Alexi, Chor, Goldreich and Schnorr proposed a PRNG construction with security that relies upon the RSA assumption. Impagliazzo

and Naor [24] and Fisher and Stern [13] proposed PRNG constructions respectively relying on the difficulty of the subset sum problem and of the syndrome decoding problem. Even in the case of specific constructions, current provably secure PRNGs are too inefficient to provide a practical stream cipher. This is due to the fact that the function iterated by the PRNG is usually too computationally expensive, and that only a restricted number of bits can be produced at each iteration (this number is generally at most proportional to the logarithm of the input length n of the iterated function). However some efforts have been made to improve the constructions. A first idea is to extract more than $\log n$ bits at each round. Constructions based on the discrete logarithm problem makes it possible to extract $n - \log(n)$ bits at each iteration instead of $\log n$. Despite this fact, the fastest generator based on discrete logarithm proposed by Gennaro [16] is still impractical: it requires 350 multiplications of 3000-bit numbers to extract 2775 bits. Another problem for which it is possible to extract more than $\log n$ bits is the syndrome decoding problem. A PRNG has been proposed by Fisher and Stern in [13] but the number of extracted bits, although higher than $\log n$, is still small for practical values of n. Another recently proposed idea is to replace a slow iterated function by some primitive which is much faster to compute. Håstad and Näslund proposed BMGL [29], a stream cipher with security that relies on the difficulty of extracting the key from one plaintext ciphertext couple in AES. Their practical construction consists of iterating AES and extracting $\log n$ bits at each round. This cipher is fast, especially compared to other provably secure ciphers, but its security relies only on the security of the AES and not on a simple and well-studied mathematical problem.

On the contrary, MQ is a simple and well-studied mathematical problem and the values of n for which the problem is difficult are small (around 100 bits), particularly when compared to discrete logarithm or factorisation, where at least 1024 bits are required. Furthermore a large number of bits (e.g. $\frac{n}{2}$) bits or even more can be produced at each iteration.

This paper is organized as follows. We first give some preliminary background on the status of the MQ problem and basic security definitions in a concrete (non asymptotic) security model. Then we describe the new construction and give a formal proof of security for the associated keystream generator. Finally we give the encryption speed of software implementations of our stream cipher.

2 Preliminaries

2.1 Multivariate Quadratic Systems

We consider a finite field $GF(q)$. A multivariate quadratic equation (or equivalently a multivariate quadratic form) in n variables over $GF(q)$ is a polynomial of degree at most 2 in $GF(q)[x_1, \ldots, x_n]$ which can be written as

$$Q(x) = \sum_{1 \leq i \leq j \leq n} \alpha_{i,j} x_i x_j + \sum_{1 \leq i \leq n} \beta_i x_i + \gamma$$

with all the coefficients $\alpha_{i,j}$, β_i, and γ in $GF(q)$. In the particular case $q = 2$, which will be considered in the sequel, the monomial forms $x_i x_i$ and x_i are equal.

It is easy to see that the set $\mathcal{Q}$ of multivariate quadratic forms in n variables is an N-dimensional vector space over $GF(q)$, where $N = \frac{n(n+3)}{2} + 1$ if $q \neq 2$ and $N = \frac{n(n+1)}{2} + 1$ if $q = 2$. A basis of this vector space is given by the $N - 1$ distinct monomial functions of degree 1 or 2 and the constant form 1. Any element of $\mathcal{Q}$ can be represented by the N-tuple of its $GF(q)$ coefficients in this basis. Throughout the rest of this paper, we mean by a randomly chosen quadratic form in n unknowns the quadratic form represented in the above basis by a uniformly and independently drawn N-tuple of $GF(q)$ coefficients.

A multivariate quadratic system S of m quadratic equations in n variables over $GF(q)$ is a set $(Q_1, \ldots Q_m)$ of m quadratic equations in n variables over $GF(q)$. In the sequel, we mean by a randomly chosen system of m quadratic form in n unknowns, n independently and randomly chosen quadratic forms. Such a system is represented by mN uniformly and independently drawn $GF(q)$ coefficients.

A quadratic form Q over n unknowns over $GF(2)$ is called non degenerate iff Q is not equivalent to a quadratic form in strictly fewer than n linear combinations of the n input variables. There exists a polynomial time algorithm to check whether a given quadratic form is non degenerate and more generally to compute the so-called rank of a quadratic form [26]. The number of solutions of the quadratic equation $Q = 0$ associated with a non degenerate quadratic form Q over n unknowns is either 2^{n-1} or $2^{n-1} + 2^{\frac{n-2}{2}}$ or $2^{n-1} - 2^{\frac{n-2}{2}}$ depending on the parity of n and the value of γ. Thus for sufficient large values of n, say $n > 100$, non degenerate quadratic forms are either perfectly balanced (odd n values) or have an undetectable bias (even n values).

2.2 Status of the MQ problem

We define the problem of solving simultaneous **multivariate quadratic** equations (**MQ problem**) as follows: given a multivariate quadratic system of m quadratic equations over $GF(q)$ $S = (Q_1, \ldots, Q_m)$, find a value $x \in GF(q)^n$, if any, such that $Q_i(x) = 0$ for all $1 \leq i \leq m$.

Depending on the respective values of n and m, instances of MQ can be either easy or very difficult to solve. For $m = 1$ the number of solutions is known [26] and it is quite easy to find one solution. When m is significantly smaller than n, that is for an underdefined quadratic system, finding a solution is easy [6]. In the opposite situation of an overdefined system ($m > n$) providing $N = \frac{n(n+1)}{2} + 1$ ($q = 2$ case) or $\frac{n(n+3)}{2} + 1$ ($q \neq 2$ case) linearly independent quadratic equations, or more generally when nearly N linearly independent quadratic equations are available, solving an MQ problem is easy by linearization. The total complexity is then only $O(n^6)$. However for general values of m and n the MQ problem is known to be NP-hard, even when restricted to quadratic equations over $GF(2)$ [15] [14] or over any finite field [28].

Moreover, what seems to make the MQ problem particularly well suited to cryptographic applications is that it is conjectured to be very difficult not only asymptotically and in worst case, but already for small suitably selected values of m and n and in terms of the average complexity of solving a random instance. The problem seems to be most difficult when m is close to n. For $m = n$ and $q = 2$ the complexity of the best known solving algorithms is $2^{n-O(\sqrt{(}n))}$ and thus rather close to the 2^n complexity of exhaustive search, and totally out of reach of existing computers for a random instance and n values larger than 100. Even when $q = 2$, $m = kn$ and $k > 1$ is small enough compared with $\frac{n}{2}$, the best known computer algebra algorithms such as XL [10] and improved variants of Buchbergers's Groebner basis computation algorithm such as Faugère's F4 and F5 algorithms [12] are exponential in n for a randomly chosen quadratic system. Much research has been dedicated in the past years to the above problem [9], [7]. Magali Bardet's PHD thesis [1] provides an accurate analysis of the complexity of the most efficient known Groebner basis computation algorithm for solving a random system of $m = kn$ equations in n unknowns. We will use some complexity estimates of [1] when discussing practical recommendations of the parameter values of our cipher.

Though we expect degenerate instances of the systems used in our construction leading to a weak stream cipher to be extremely unlikely, we suggest the following extra precaution when drawing these systems at random to provide some extra guaranties that some of the weakest instances are avoided: check that each quadratic equation is non degenerate or at least has a high rank value close to the one of a non degenerate form, and discard any quadratic equation which would not satisfy this condition. In order to discard a slightly larger subset of weak instance, one can also check that low weight linear combinations of the selected quadratic equations satisfy the above rank conditions. Also check that the obtained quadratic equations are linearly independent in $\mathcal{Q}$.

2.3 Basic Security Notions

All the security definitions used throughout this paper relate to the concrete (non asymptotic) security model. We are using the following basic security notions that we state here informally. Two probability distributions D_1 and D_2 over a finite set Ω are said to be **computationally distinguishable** with computing resources R and advantage ϵ if there exits a probabilistic testing algorithm A which on any input value $x \in \Omega$ outputs a binary answer "1" (accept) or "0" (reject) using computing resources at most R and satisfies

$$|Pr_{x\in D_1}(A(x) = 1) - Pr_{x\in D_2}(A(x) = 1)| \geq \epsilon.$$

Though this is not explicitly reflected in our notation, the above probabilities are not only taken over x values distributed according to D_1 or D_2, but also over the random choices of algorithm A. Algorithm A is called a distinguisher with advantage ϵ. If no such algorithm exists, then we say that D_1 and D_2 are computationally indistinguishable with advantage better than ϵ. When the

computing resources R is not specified, we implicitly mean feasible computing resources (i.e. say less than 2^{80} simple operations).

Let n and L denote integers such that $L > n$. A n-bit to L-bit function G is said to be a **Pseudo Random Number Generator (PRNG)** if for a random n-bit input variable x selected according to the uniform law on $\{0,1\}^n$ the probability distribution of the random variable $G(x)$ is computationally indistinguishable from the uniform law over $\{0,1\}^L$.

3 QUAD: a New Stream Cipher

We now introduce the proposed stream cipher, named QUAD.

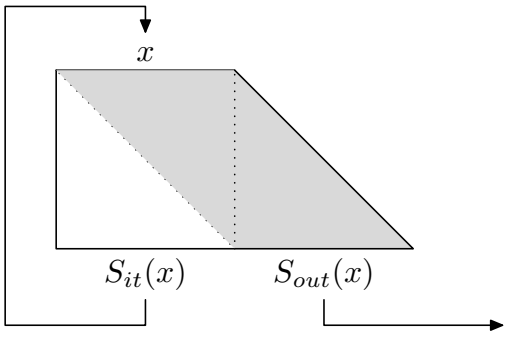
$S = (Q_1, \ldots, Q_{kn})$ denotes a multivariate quadratic system of kn randomly chosen equations in n variables over $GF(q)$, and S_0 and S_1 denote two (k times smaller) additional multivariate systems of n randomly chosen equations in n variables over $GF(q)$. S, S_0 and S_1 are fixed and publicly known. During the key and IV loading and the keystream generation, the internal register state is a $x = (x_1, \ldots, x_n)$ n-tuple of $GF(q)$ values.

3.1 Keystream Generation and Encryption

The keystream generation process simply consists in iterating the three following steps in order to produce $(k-1)n$ $GF(q)$ keystream values at each iteration.

- Compute the kn-tuple of $GF(q)$ values $S(x) = (Q_1(x), \ldots, Q_{kn}(x))$ where x is the current value of the internal state;
- Output the sequence $S_{out}(x) = (Q_{n+1}(x), \ldots, Q_{kn}(x))$ of $(k-1)n$ $GF(q)$ keystream values
- Update the internal state x with the sequence of n $GF(q)$ first generated values $S_{it}(x) = (Q_1(x), \ldots, Q_n(x))$

The maximal keystream sequence that may be generated with a single (key,iv) pair is L $GF(q)$ values. In order to encrypt a plaintext of length $l \leq L$ $GF(q)$ symbols, each of the first l $GF(q)$ values of the keystream sequence is added (using the $GF(q)$ addition) with the corresponding plaintext value.



3.2 Key and IV Setup

Before generating any keystream we need to initialize the internal state x, with the key K and the initialization vector IV, which are respectively represented by a sequence of $GF(q)$ elements of length $|K|$ and a binary sequence of $\{0,1\}$ values of length $|IV|$. We assume for the time being, for simplicity of the subsequent proofs [3] that $|K|$ is chosen exactly equal to n.

The initialization is done as follows : we use two carefully randomly chosen multivariate quadratic systems S_0 and S_1 of n equations over n unknowns. We initially set the internal state value x to the n bit value K. Then for each of the $|IV|$ bits IV_1 to $IV_{|IV|}$ of the IV value the internal state x is updated as follows: if $IV_i = 0$, x is replaced by the $GF(q)^n$ value $S_0(x)$; if $IV_i = 1$, x is replaced by the $GF(q)^n$ value $S_1(x)$. These $|IV|$ steps provide a key and IV dependent internal state value x. We then clock the cipher $|IV|$ additional times as described in section 3.1, but without outputting the keystream in order to further transform the internal state value x, and then enter the keystream generation mode to produce the keystream.

4 Security

We now give a proof that for a randomly chosen multivariate quadratic system our PRNG is secure. For simplicity of the proof we will work over $GF(2)$. The proof can be divided in three parts, which can be informally outlined as follows.

In the first part (Theorem 1), we prove that if the L-bit keystream sequence associated with a known fixed or randomly chosen system S of $m = kn$ quadratic equations and an unknown randomly chosen initial internal state $x \in \{0,1\}^n$ is distinguishable from the L-bit output of a perfectly uniform generator, then for a known random quadratic system S of $m = kn$ equations and an unknown randomly chosen input value $x \in \{0,1\}^n$, $S(x)$ is distinguishable from a random kn bit word.

In the second part (Theorem 2), we prove that if for a known randomly chosen quadratic system S and an unknown randomly chosen x, $S(x)$ is distinguishable from a random kn bit word then, for any n-bit to 1-bit quadratic form R (in particular any linear form R), one has the property that for a randomly chosen n bit value x, $R(x)$ can be predicted better than at random given $S(x)$.

In the third part (Theorem 3), we prove that, for a known fixed or randomly chosen S and a randomly chosen linear form R, $R(x)$ can be predicted better than at random given $S(x)$, then with non negligible probability a preimage of $S(x)$ can be efficiently computed given $S(x)$. Thus S is not strongly one way. This part is essentially a proof of Goldreich-Levin's theorem [25], in which a fast Walsh transform computation is used to get a tighter reduction.

[3] Note however that we will consider later on, in section 4.5, an extended key loading method allowing to set the key length to values strictly lower than n, for instance to $|K| = \frac{n}{2}$ if one wishes the key length to reflect the complexity of the best known attack.

4.1 Distinguishing the Keystream Allows to Distinguish the Output of a Random Quadratic System

Theorem 1 states that if one can distinguish the keystream of the generator based on the iteration of a quadratic system S from a random L-bit sequence, then one can distinguish the output of S from a random m-bit sequence. Though we consider a randomly chosen system S because we need distinguishing properties related to a random system for the sequel, the property we prove would also hold if we considered a fixed system S. Our proof is inspired by the proof given in [20] that a similar result holds for the generator based on iteration of any fixed n-bit to m-bit function, where $m > n$, but provides a tighter bound for the advantage.

Theorem 1. *Let $L = \lambda(k-1)n$ be the number of keystream bits produced in time λT_S using λ iterations of our construction. Suppose there is an algorithm A that distinguishes the L-bit keystream sequence associated with a known randomly chosen system S and an unknown randomly chosen initial internal state $x \in \{0,1\}^n$ from a random L-bit sequence in time T with advantage ϵ. Then there exists an algorithm B that for a randomly chosen S distinguishes $S(x)$ corresponding to an unknown random input x, from a random value of size kn in time $T' = T + \lambda T_S$ with advantage $\frac{\epsilon}{\lambda}$.*

Proof. We introduce the hybrid probability distributions $D^i(S)$ over $\{0,1\}^L$. For $0 \leq i \leq \lambda$ respectively associated with the random variables

$$t^i(S,x) = (r_1, r_2, \ldots, r_i, S_{out}(x), S_{out}(S_{it}(x)), \ldots, S_{out}(S_{it}^{\lambda-i-1}(x)))$$

where the r_j and x are random independent uniformly distributed values of $\{0,1\}^n$ and the notational conventions that $(r_1, r_2, \ldots, r_i)$ is the null string if $i = 0$ and that $(S_{out}(x), \ldots, S_{out}(S_{it}^{\lambda-i-1)}(x)))$ is the null string if $i = \lambda$. Consequently $D^0(S)$ is the distribution of the L-bit keystream and $D^\lambda(S)$ is the uniform distribution over $\{0,1\}^L$. We denote by $p^i(S)$ the probability that A accepts a random L-bit sequence distributed according to $D^i(S)$, and denote by p^i the average value of $p^i(S)$ over the $(k-1)n(n\frac{(n+1)}{2}+1)$-dimensional vector space of quadratic systems S. We have supposed that algorithm A distinguishes between $D^0(S)$ and $D^\lambda(S)$ with advantage ϵ, in other words that $|p^0 - p^\lambda| \geq \epsilon$. Algorithm B works as follows : on input $(x_1, x_2) \in \{0,1\}^{kn}$ with $x_1 \in \{0,1\}^n$ and $x_2 \in \{0,1\}^{(k-1)n}$, it selects randomly an i such that $0 \leq i \leq \lambda - 1$ and constructs the L-bit vector

$$t(S, x_1, x_2) = (r_1, r_2, \ldots, r_i, x_2, S_{out}(x_1), S_{out}(S_{it}(x_1)), \ldots, S_{out}(S_{it}^{\lambda-i-2}(x_1))).$$

If (x_1, x_2) is distributed accordingly to the output distribution of S, i.e. $(x_1, x_2) = S(x) = (S_{it}(x), S_{out}(x))$ for a uniformly distributed value of x, then

$$t(S, x_1, x_2) = (r_1, r_2, \ldots, r_i, S_{out}(x), S_{out}(S_{it}(x)), \ldots, S_{out}(S_{it}^{\lambda-i-1}(x)))$$

is distributed according to $D^i(S)$. Now if (x_1, x_2) is distributed according to the uniform distribution, then

$$t(S, x_1, x_2) = (r_1, r_2, \ldots, r_i, x_2, S_{out}(x_1), S_{out}(S_{it}(x_1)), \ldots, S_{out}(S_{it}^{\lambda-i-2}(x_1))).$$

Thus $t(S, x_1, x_2)$ is distributed according to $D^{i+1}(S)$. In order to distinguish the output distribution of S from the uniform law, algorithm B calls algorithm A with inputs $(S, t(S, x_1, x_2))$ and returns the value returned by A. Thus

$$\begin{aligned} &|Pr_{S,x}(B(S, S(x)) = 1) - Pr_{S,x_1,x_2}(B(S, (x_1, x_2)) = 1)| \\ &= |\frac{1}{\lambda}\sum_{i=0}^{\lambda-1} p^i - \frac{1}{\lambda}\sum_{i=1}^{\lambda} p^i| = \frac{1}{\lambda}|p^0 - p^\lambda| \geq \frac{\epsilon}{\lambda}. \end{aligned}$$

Thus B distinguishes the output distribution of S from the uniform distribution with probability at least $\frac{\epsilon}{\lambda}$ in time $T + \lambda T_S$.

4.2 Distinguishing the Output of a Random Quadratic System Allows to Predict any Quadratic Equation

Now we prove that if there exists a distinguisher between $S(x)$ and a kn-bit random value such as the one considered in the above theorem, it can be converted into an algorithm that predicts the result of any quadratic polynomial (and in particular any linear polynomial).

Theorem 2. *Suppose there is an algorithm A that, given a randomly chosen known multivariate quadratic system S of kn equations in n unknowns, distinguishes $S(x)$, where x is an unknown random input value, from a random string of length kn with advantage at least ϵ and in time T. Then there is an algorithm B that, given a randomly chosen quadratic system S of kn equations in n unknowns, any n-bit to 1-bit quadratic form R, and $y = S(x)$ where x is a random input value, predicts $R(x)$ with success probability at least $\frac{1}{2} + \frac{\epsilon}{4}$ using at most $T' = T + 2T_S$ operations.*

Proof. We first show that there exists an algorithm A' which returns 1 on input $(S, S(x))$ with probability at least $\frac{1}{2} + \frac{\epsilon}{2}$ and returns 1 on input (S, u) for some random u with probability $\frac{1}{2}$: if the acceptance probability of A is larger (by at least ϵ) on an input $(S, S(x))$ than on a random input. Then it suffices to consider A' which on input (S, r) either returns $A(S, r)$ or draws a random value u and returns $1 - A(S, u)$ with probability $\frac{1}{2}$ for each case. In the opposite situation, it suffices to consider A' which on input (S, r) either returns $1 - A(S, r)$ or draws a random value and returns $A(S, u)$ with probability $\frac{1}{2}$ for each case.

Algorithm B works as follows. On input $S = (Q_1, \ldots Q_{kn})$, R and a kn-bit value y, B selects a random kn-bit vector $a = (a_1, \ldots, a_{kn})$ and a random bit b, which represents an hypothesis for $R(x)$. Then it computes for all i from 1 to kn the quadratic equation $P_i = Q_i + (a_i \cdot R)$. All the equations P_i form the quadratic system S'. Then B invokes the algorithm A' with input the new quadratic system S' and the value $y + (b \cdot a)$. Finally B returns what A' returns.

Now assume that $y = S(x)$ where x is an unknown random value. We have $\forall i, x, P_i(x) = Q_i(x) + (a_i \cdot R(x)) = y_i + (a_i \cdot R(x))$.

Suppose b is really equal to $R(x)$, then $S'(x) = y + (b \cdot a)$ so the distinguisher A' has been fed with the random quadratic system $S' = (P_1, \cdots, P_{kn})$ and $S'(x)$:

$$Pr_{S,x\in U_n}(B(S, S(x), R) = R(x)) = Pr_{S',x\in U_n}(A'(S', S'(x)) = 1) \geq \frac{1}{2} + \frac{\epsilon}{2}.$$

On the contrary, suppose b is not equal to $R(x)$, then $S'(x) = y + ((1 + b) \cdot a) = (y + (b \cdot a)) + a$. Thus there is an error of a on the value furnished to A' as compared with $S'(x)$. Because a is randomly chosen, we have:

$$\begin{aligned} Pr_{S,x\in U_n}(B(S, S(x), R) = R(x)) &= Pr_{S',x\in U_n}(A'(S', S'(x) + a) = 0) \\ &= Pr_{S',t\in U_{kn}}(A'(S', t) = 0) = \frac{1}{2} \end{aligned}$$

Thus we have:

$$Pr_{S,x\in U_n}(B(S, S(x), R) = R(x)) \geq \frac{1}{2}\left(\left(\frac{1}{2} + \frac{\epsilon}{2}\right) + \frac{1}{2}\right) = \frac{1}{2} + \frac{\epsilon}{4}$$

The total running time of B is at most $T + 2T_S$, since computing the kn P_i requires for each i to compute all the $\frac{n(n-1)}{2}$ monomials of Q_i and R, which does not cost more than two evaluations of the system for some entry.

4.3 A Linear Form is a Hard Core Bit for any One Way Function

Now we show that if for a fixed or random quadratic system S and more generally any fixed or random n-bit to m-bit function f there exists a predictor such as the one considered in the former theorem, i.e. a predictor allowing, given an n-bit to 1-bit linear form R, to predict $R(x)$ with a success probability (over all S and x values) strictly larger than $\frac{1}{2}$, then a preimage of $S(x)$ (resp. f(x)) can be efficiently computed, so that S (resp f) is not one way. This result is the Goldreich-Levin theorem [25] that we prove as to get a tight reduction. Before proving the theorem, which relates to the computation, given the image $S(x)$ or $f(x)$ for a random unknown value x and a random system S, of a list containing x, we first establish a lemma representing the technical core of the proof in which a fixed (unknown) value of x is considered. Our proofs are inspired by the simplified treatment of the original Goldreich-Levin proofs developed by Rackoff, Goldreich[18] and Bellare [2], and also by the proofs provided by Håstad and Näslund in their BMGL paper [29].

Lemma 1. *Let us denote by x a fixed unknown n-bit value and denote by f a fixed n-bit to m-bit function. Suppose there exists an algorithm B that given the value of $f(x)$ allows to predict the value of any linear equation R over n unknowns with probability $\frac{1}{2} + \epsilon$ over R, using at most T operations. Then there exists an algorithm C, which given $f(x)$ produces in time at most T' a list of at*

most $4n^2\epsilon^{-2}$ *values such that the probability that* x *appears in this list is at least* $1/2$.

$$T' = \frac{2n^2}{\epsilon^2}\left(T + \log\left(\frac{2n}{\epsilon^2}\right) + 2\right) + \frac{2n}{\epsilon^2}T_f$$

The proof of lemma 1 is given in the Appendix. Lemma 1 applies to a fixed x and a fixed system S (or a fixed n-bit to m-bit function f). However, the success probability of the predictor of Theorem 2 is taken over all (x, S) pairs for any linear form R. Consequently, we need a theorem allowing us to exploit the existence of such a predictor to show the applicability of the lemma to a non-negligible fraction of (x, S) pairs.

Theorem 3. *Suppose there exists an algorithm* B*, that given a randomly chosen quadratic system* S *of* m *quadratic equations, a randomly chosen* n*-bit to 1-bit quadratic form* R *and the image* $S(x)$ *of a randomly chosen (unknown)* n*-bit value* x*, predicts the value of* $R(x)$ *with probability at least* $\frac{1}{2} + \epsilon$ *over all possible (x, S, R) triplets using* T *operations. Then there is an algorithm* C*, which given the image* $S(x)$ *of a randomly chosen (unknown)* n*-bit value* x *produces a preimage of* $S(x)$ *with probability at least* $\epsilon/2$ *(over all possible values of* x *and* S*) in time* T'.

$$T' = \frac{8n^2}{\epsilon^2}\left(T + \log\left(\frac{8n}{\epsilon^2}\right) + 2\right) + \frac{8n}{\epsilon^2}T_f$$

Proof. The assumption about algorithm B can be written as

$$Pr_{(x,S,R)\in\{0,1\}^{n+mN+n}}\{B(S, S(x), R) = R(x)\} \geq \frac{1}{2} + \epsilon.$$

It results that for a fraction at least ϵ of all the (x, S) pairs one has

$$Pr_{R\in\{0,1\}^n}\{B(S, S(x), R) = R(x)\} \geq \frac{1}{2} + \frac{\epsilon}{2}.$$

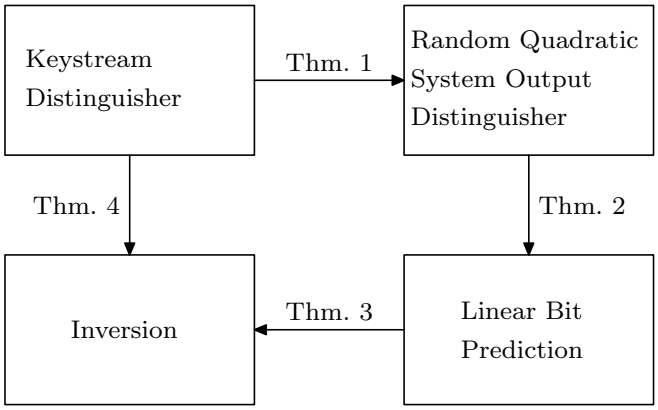
Otherwise, there would exist a fraction at least $1 - \epsilon$ of the (x, S) pairs which associated prediction probability over the R values would be strictly less than $\frac{1}{2} + \frac{\epsilon}{2}$, and therefore $Pr_{(x,S,R)\in\{0,1\}^{n+mN+n}}\{B(S, S(x), R) = R(x)\}$ would be upper bounded by $(1-\epsilon)(\frac{1}{2} + \frac{\epsilon}{2}) + \epsilon = \frac{1}{2} + \epsilon - \epsilon^2$, which contradicts the assumption about Algorithm B.

Thus for a fraction at least ϵ of all the (x, S) pairs the conditions of lemma 1 are met and algorithm C of the lemma provides a preimage of $S(x)$ with probability at least $1/2$.

4.4 A Security Proof for the Proposed PRNG

Now it is easy to see that if we sequentially apply theorems 1, 2, and 3, we obtain the following reduction theorem, which states that if, for a random system and a random initial value, the L-bit keystream sequence was distinguishable from a random L-bit sequence then there would exist an efficient algorithm allowing

to find a preimage of the image of a random n-bit input value by a random quadratic n-bit to m-bit system, which for suitably chosen values of n would contradict the assumptions made in Section 2 on the difficulty of solving MQ.



Theorem 4. *Let $L = \lambda(k-1)n$ be the number of keystream bits produced by in time λT_S using λ iterations of our construction. Suppose there exists an algorithm A that distinguishes the L-bit keystream sequence associated with a known randomly chosen system S and an unknown randomly chosen initial internal state $x \in \{0,1\}^n$ from a random L-bit sequence in time T with advantage ϵ. Then there exists an algorithm C, which given the image $S(x)$ of a randomly chosen (unknown) n-bit value x by a randomly chosen n-bit to m-bit quadratic system S produces a preimage of $S(x)$ with probability at least $\frac{\epsilon}{2^3\lambda}$ over all possible values of x and S in time upper bounded by T'.*

$$T' = \frac{2^7 n^2 \lambda^2}{\epsilon^2}\left(T + (\lambda+2)T_S + \log\left(\frac{2^7 n\lambda^2}{\epsilon^2}\right) + 2\right) + \frac{2^7 n\lambda^2}{\epsilon^2} T_S$$

Proof. Theorems 1 to 3 state that if an algorithm X exists, then another algorithm Y exists. In the case of Theorem 1, the resulting algorithm Y can be directly play the role of algorithm X in Theorem 2. In the case of Theorem 2, the resulting algorithm Y, named algorithm B, has the property

$$\forall R \in \{0,1\}^N Pr_{(x,S)\in\{0,1\}^{n+mN}}\{B(S, S(x), R) = R(x)\} \geq \frac{1}{2} + \frac{\epsilon}{4}$$

which implies

$$Pr_{(x,S,R)\in\{0,1\}^{n+mN+N}}\{B(S, S(x), R) = R(x)\} \geq \frac{1}{2} + \frac{\epsilon}{4}$$

Thus algorithm Y can play the role of algorithm X in Theorem 3, and if we compose the distinguishing probability and complexity expressions of the three concatenated theorems, we obtain the claimed distinguishing probability and complexity bounds.

Discussion: Theorem 4 above relates to the keystream generation part of QUAD, i.e. to the expansion of a randomly chosen initial state into the keystream

and does not include the key and IV loading for deriving the initial state. Moreover it does not guarantee the strength of a particular instance of QUAD associated with a fixed system S but (informally) it shows that if MQ is intractable then most instances of QUAD are secure.

4.5 Specifying the Parameter Values for QUAD

We now propose concrete parameters n, k, L, $|K|$ and $|IV|$ for our construction. We restrict ourselves to the $GF(2)$ case. We want to ensure a security level of at least 2^{80}. More precisely we want Theorem 4 to ensure that if for a random system and a random initial internal state value at the beginning of the keystream generation there exists a testing algorithm that allows us to distinguish an L-bit keystream produced by QUAD from a uniformly drawn keystream sequence with an advantage of more than $\epsilon = \frac{1}{100}$ in time less than $T = 2^{80}$ this would imply the existence of an inversion algorithm of non negligible success probability $\epsilon' = \frac{\epsilon}{2^3\lambda}$ allowing, given a random n-bit to kn-bit system of quadratic equations and the $S(x)$ image by S of a random input value x, to find a preimage by S of $S(x)$ in time T' lower by a factor of more than ϵ' than the best known inversion algorithms for the MQ problem, and thus result in the existence of a large set of weak instances of MQ.

Depending on the intended application of the stream cipher, the maximum keystream length L can vary from a few hundreds bits for a mobile phone application to up to 2^{40} bits. Consequently the allowed parameter values for n and k will also vary, since it is much more demanding to get a security argument for $L = 2^{40}$ bits than for $L = 1000$ bits. We will however retain the latter value $L = 2^{40}$ for a first estimate of the corresponding required value of n.

In her thesis, Magali Bardet [1] shows that the best Groebner basis algorithm to solve a system of kn equations in k unknowns has (in the case of a regular system) a complexity of $T(k,n) = \left(\binom{n+1}{D}\right)^{2.37}$, where D is close to $\left(-k + \frac{1}{2} + \frac{1}{2}\sqrt{2k^2 - 10k - 1 + 2(k+2)\sqrt{k(k+2)}}\right) n$. To obtain a contradiction, we need to have T' lower than $\epsilon' T(k,n)$. For $k = 2$ and with the previous values of $L = 2^{40}$, $T = 2^{80}$ and $\epsilon = \frac{1}{100}$, we get $\epsilon' = 2^{-42}$ and we need to have n greater than 350. For $n = 256$ and $k = 2$, we only get a contradiction if we produce less than $L = 2^{22} = 4$ Mbits of keystream for each key and IV pair.

Practical values For practical use of QUAD we recommend an internal state length of $n = 160$ bits and an expansion factor k of 2 and a maximum keystream length $L = 2^{40}$. We further recommend an IV length $|IV|$ of 80 bits. For such n, k and L values, we do not get a contradiction as for the former parameter values. However our proof reduction is not optimal, and we expect that these parameter values suffice to provide the desired security level of about 2^{80}.

If instead of the n-bit key length assumed (for simplicity of the security arguments) in sections 2 and 3, a keylength $|K|$ strictly lower than n is preferred in order for $|K|$ to better reflect the expected security level, we suggest the

following extension of the key loading method described in section 3: periodically repeat the $|K|$ bits of K to get an expanded key of length n, and apply the key and IV procedure of section 3 to this expanded key. We suggest, if this extended key loading method option is retained, to select a key length $|K| = 80$. Though the shorter key option weakens the security arguments of section 4 and can thus be considered less conservative than the full length $n = 160$-bit key, we are not aware of any major security weakness resulting from this option.

An indication of the advantages of the use of the MQ problem for constructing a provably secure stream cipher, in terms of the required internal state size, is given by a comparison with the fastest known provably secure stream cipher, namely a discrete log based construction proposed by Gennaro in [16] with internal state length $n = 3000$ bits (to be compared with the $n = 350$ and 256 internal state lengths derived above) and which produces 2775 bits per iteration and applies 335 modular multiplications of 3000-bit numbers at each iteration. Moreover the security argument of [16] does not assume the existence of a keystream sequence distinguishing algorithm in time $T = 2^{80}$ to get a contradiction, but only a distinguishing algorithm in time $T = 3.5 \cdot 10^{10} \simeq 2^{35}$. Another advantage of MQ is that MQ is NP-hard, whereas the Discrete Logarithm Problem is only in NP $\cap$ co-NP. Moreover the best known algorithm to solve the Discrete Logarithm problem are subexponential, while for MQ, those algorithm are exponential.

5 Cryptanalysis

In this section, we consider various attacks and verify whether they are applicable to our construction. We focus on security aspects not covered by the proof of security of the former section, e.g. the protection against resynchronization attacks provided by the key and IV loading mechanism.

Resistance against Algebraic Attacks: QUAD was designed to resist algebraic attack techniques. As a matter of fact, the key and IV loading and keystream generation mechanisms of QUAD are based upon the iteration of quadratic systems whose associated equations are conjectured to be computationally impossible to solve [4]. In more details, recovering the initial state x of the keystream generator from the whole keystream is more difficult than recovering x from $S(x)$, i.e. solving an intractable quadratic system of kn equations. As for the key and IV loading mechanism, it is possible to express any keystream block, as a set of $(k-1)n$ algebraic equations on the $|K| \geq 80$ key bits. However since the key and IV setup consists of $2|K|$ rounds of a quadratic function, this set consists of $(k-1)n$ equations of degree $|K|$ or nearly $|K|$ on the $|K|$ key bits. It is quite natural to conjecture that such a system is highly intractable.

Correlation Attacks and Distinguishing Attacks: we expect QUAD to be immune to such attacks except for extremely unlikely degenerate instances of the quadratic system S, for example if one of the n-bit to 1-bit quadratic

[4] except for a small fraction of degenerate instances of S, S_0 and S_1 whose occurrence is extremely unlikely if these systems are selected as described in section 4.5.

forms of S_{out} or a linear combination of these $(k-1)n$ quadratic forms has an exceptionally low rank and therefore (for even values of n) a detectable bias.

Time-Memory-Data Tradeoffs and other Generic Attacks: the internal state of our construction has a size n of at least 160 bits in order to resist against generic time-data tradeoff, which have a complexity of $2^{\frac{n}{2}}$.

Since QUAD is based upon the iteration of the quadratic system S_{it}, the keystream sequences it produces are ultimately periodic. Moreover, since S_{it} is not one to one, the order of magnitude of the period can be expected to be $2^{\frac{n}{2}}$ $(k-1)n$-bit keystream blocks. One of the consequences of specifying a maximal keystream length $L << 2^{\frac{n}{2}}$ (a typical order of magnitude is $L = 2^{40}$) is that the detection of short cycles is extremely unlikely.

Guess and Determine Attacks: the analysis of attacks of this type allows us to fix an upper bound on k. Let us assume that an adversary is able to guess p bits of the internal state. Then this adversary gets a system of $(k-1)n$ equations in the $(n-p)$ remaining internal state variables. If the number of monomials generated by these $n-p$ variables $n_p = \frac{1}{2}(n-p)(n-p+1)$ is close to $(k-1)n$, the adversary can linearize the system and recover the internal state. Solving $n_p = (k-1)n$ gives us a number $p_0 = n + \frac{1-\sqrt{1+8n(k-1)}}{2}$ such that for $p \geq p_0$ the linearization is possible. The complexity C of the resulting "attack" is about $2^{p_0}((k-1)n)^{\omega}$, where ω is between 2 and 3. If C is lower than $2^{|K|}$, then the attack is better than exhaustive search. Consequently, k has to be chosen such that C be larger than $2^{|K|}$. For instance for $n = 160$ and $|K| = 80$, $k < 21$ implies that $p_0 > 80$, and therefore $C >> 2^{80}$. More conservative (i.e. lower) values of k than the one given by this simple bound are of course recommended.

Unsurprisingly, the attack would become more efficient for unlikely degenerate instances of S, for instance if several quadratic forms of S could be all expressed as quadratic functions of substantially less than n linear combinations of the n state variables.

Resistance to Resynchronization Attacks with Chosen IVs: our proof does not cover the Key and IV setup but only the keystream generation. They provide a strong argument towards the conjecture that the keystream sequence resulting from any single known or chosen IV value cannot be distinguished from a random sequence, but do not provide guarantees regarding the independence of the sequences resulting from several chosen IVs and the resistance of QUAD against resynchronization attacks. However the following informal argument indicates that the key and IV setup construction of QUAD prevents such resynchronization attacks, or more generally any detectable statistical bias on the joint distribution of the keystream sequences resulting from the same key and several chosen IVs. Let us consider any t-tuple $(IV^1, \cdots, IV^t)$ of t distinct IV values and one randomly chosen n-bit initial state value before IV loading x. By applying the security proofs of section 4 to the $S = (S_0, S_1)$ system of $2n$ quadratic equations, the n-bit to $2n$-bit mapping S_0, S_1 is a strong pseudorandom generator. However, the key and IV loading consists of applying a tree-based construction proposed by Goldreich, Goldwasser and Micali [19] to this generator, so that we can expect the distribution of the $(x^1, \cdots, x^t)$ t-tuple of internal

state values resulting from the loading of x and IV^1 to IV^t to be indistinguishable from a t-tuple of random independent values. Moreover, the subsequent runnup rounds during which the keystream generator is run without outputting keystream bits provide an extra security margin, since only high degree functions of x^1 to x^t are available to an adversary instead of quadratic functions. If instead of the proposed key and IV setup the key and IV values the IV had been loaded into the initial state and an insufficient number of quadratic mappings had been applied to the initial state before activating the keystream generation, then chosen-IV attacks exploiting the higher degree differential properties of low degree functions could have been mounted.

Dual Ciphers: because of the structure of the QUAD equations, it is easy to find dual ciphers of QUAD, i.e. simple (e.g. linear) transformations f and g of the key K and the keystream as to ensure that for each triplet of quadratic systems (S, S_0, S_1) there exist quadratic systems (S', S'_0, S'_1) such that for any key K and any IV value IV, the keystream associated with $(f(K), IV, S', S'_0, S'_1)$ is the image by g of the keystream associated with $(f(K), IV, S, S_0, S_1)$. We do not expect this property to represent a security threat for QUAD.

6 Performance

In this Section we give performance results for our recommended version of QUAD, which has 160 bits of internal state, an expansion factor of 2 and a 80-bit key and IV length. On a Pentium IV clocked at 2.5GHz with 512 kByte of cache and using the Intel compiler, our recommended version of QUAD reaches a speed of 4347 cycles/byte (4.6 Mbit/s). On a Pentium 4 with 1MByte of cache, the same version reaches a speed of 2915 cycles/byte (5.7 Mbit/s). This cache effect is due to the fact that the quadratic system used contains more than 4 millions of binary coefficients, which requires around 1MByte to store. A version of QUAD running on an Opteron clocked at 2.1 GHz with a 64-bit architecture reaches the speed of 2176 cycles/byte (quite close from 1MByte/s). An optimised version of Blum Blum Shub's generator with an internal state of 1024 bits, which is far from the number of bits of the internal state required for proven security, reaches 30374 cycles/byte. In his paper[16], Gennaro claimed his discrete logarithm based generator to be twice faster for these parameters. We can therefore assume that this generator runs at about 15000 cycles/byte. Though QUAD is significantly slower than AES, which runs at 25 cycles/byte, it is much more efficient than other provably secure pseudo random generator. Moreover, implementations of QUAD with quadratic system over larger fields (e.g. $GF(16)$ or $GF(256)$) are much faster.

7 Conclusion

In this paper we introduced QUAD, a novel synchronous stream cipher based on MQ with a security proof in the concrete security model. Eventhough this construction relies on a mathematical problem and has a proof of security, its

internal state is of small size n and it extracts a small multiple of n bits at each round. A software implementation of our recommended version of QUAD reaches a speed of 4.6 Mb/s on a standard PC. This makes QUAD of great interest for applications where security is the main concern. We do not preclude that it might be possible to derive tighter bounds in some parts of the proof, which would allow us to further reduce the internal state size and increase the number of extracted bits.

We would like to thank Matt Robshaw and Olivier Billet for helpful comments.

References

1. Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie*. PhD thesis, Université Paris VI, 2004.
2. Mihir Bellare. The Goldreich-Levin Theorem. `http://www-cse.ucsd.edu/users/mihir/courses.html`, 1999.
3. Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.
4. Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.
5. Don Coppersmith, Shai Halevi, and Charanjit S. Jutla. Cryptanalysis of stream ciphers with linear masking. In Moti Yung, editor, *Advances in Cryptology – CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 515–532. Springer-Verlag, 2002.
6. Nicolas Courtois, Louis Goubin, Willi Meier, and Jean-Daniel Tacier. Solving underdefined systems of multivariate quadratic equations. In *Public Key Cryptography*, pages 211–227, 2002.
7. Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer-Verlag, 2000.
8. Nicolas Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer-Verlag, 2003.
9. Nicolas Courtois and Jacques Patarin. About the XL Algorithm over $GF(2)$. In Marc Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 141–157. Springer-Verlag, 2003.
10. Claus Diem. The XL-Algorithm and a Conjecture from Commutative Algebra. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 323–337. Springer-Verlag, 2004.
11. ECRYPT. eSTREAM: ECRYPT Stream Cipher Project, IST-2002-507932. Available at `http://www.ecrypt.eu.org/stream/`, Accessed September 29, 2005, 2005.
12. Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, Makoto Sugita, and Gwénolé Ars. Comparison Between XL and Grbner Basis Algorithms. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 338–353. Springer-Verlag, 2004.
13. Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In *EUROCRYPT*, pages 245–255, 1996.

14. Aviezri S. Fraenkel and Yaacov Yesha. Complexity of solving algebraic equations. *Inf. Process. Lett.*, 10(4/5):178–179, 1980.
15. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, chapter 7.2 Algebraic Equations over $GF(2)$. W H Freeman & Co, 1979.
16. Rosario Gennaro. An improved pseudo-random generator based on discrete log. In *CRYPTO*, pages 469–481, 2000.
17. Oded Goldreich. Three xor-lemmas an exposition. Technical report, Weizmann Institute of Science, Revohot, Israel, 1995.
18. Oded Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.
19. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
20. Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography. Available at `http://www-cse.ucsd.edu/users/mihir/courses.html`, 2001.
21. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudo-random generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.
22. Russel Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In D.S.Johnson, editor, *21th ACM Symposium on Theory of Computing – STOC '89*, pages 12–24. ACM Press, 1989.
23. Russel Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *Journal of Cryptology*, 9(4):199–216, 1996.
24. Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptology*, 9(4):199–216, 1996.
25. Leonid A. Levin and Oded Goldreich. A hard-core predicate for all one-way functions. In D. S. Johnson, editor, *21th ACM Symposium on Theory of Computing – STOC '89*, pages 25–32. ACM Press, 1989.
26. Rudolf Lidl and Haradl Niederreiter. *Finite Fields*. Cambride University Press, 1997.
27. National Institute of Standards and Technology. FIPS-197: Advanced Encryption Standard, November 2001. Available at `http://csrc.nist.gov/publications/fips/`.
28. Jacques Patarin and Louis Goubin. Asymmetric cryptography with s-boxes. In *ICICS*, pages 369–380, 1997.
29. Johan Håstad and Mats Näslund. Bmgl: Synchronous key-stream henerator with provable security. submitted to Nessie Project, 2000.
30. Andrew Yao. Theory and applications of trapdoor function. In *Foundations of Cryptography FOCS 1982*, 1982.

Appendix: Proof of lemma 1

We denote by L_i, $1 \leq i \leq n$ the n-bit to 1-bit linear forms defined by $L_i(x)=x_i$, where x is represented by the binary string $x_1x_2\cdots x_n$. The idea of the proof is to call algorithm B sufficiently many times to recover all the $x_i = L_i(x)$ one by one. To do so, we introduce a parameter t, whose order of magnitude is $\log n$ which will be specified later. We use t randomly chosen n-bit to 1-bit linear forms $R_1, \ldots, R_t$ to randomize our requests to algorithm B. For each $L_i(x)$ we want to

retrieve, we call algorithm B 2^t times, using the 2^t linear combinations $\bigoplus_j \alpha_j R_j$ of the R_k forms in order to randomize L_i. Suppose we know the t values for $R_j(x)$, then for any α we can also compute the value of $\bigoplus_j \alpha_j R_j(x)$ and add this value to $B(\bigoplus_j \alpha_j R_j \oplus L_i, f(x))$. We denote

$$C(i,\alpha) = B(\bigoplus_j \alpha_j R_j \oplus L_i, f(x)) \oplus \bigoplus_j \alpha_j R_j(x)$$

If we make a correct assumption on the t values $R_1(x)$ to $R_t(x)$ and if B returned the right value of $(\bigoplus_j \alpha_j R_j \oplus L_j)(x)$, then we have

$$\begin{aligned} C(i,\alpha) &= (\bigoplus_j \alpha_j R_j \oplus L_i)(x) \oplus \bigoplus_j \alpha_j R_j(x) \\ &= L_i(x) \oplus \bigoplus_j \alpha_j R_j(x) \oplus \bigoplus_j \alpha_j R_j(x) = L_i(x). \end{aligned}$$

For all the possible α values, we collect the vote $C(i,\alpha)$ for the value of $L_i(x)$. Since algorithm B is supposed to answer correctly most of the time, taking the majority of the votes $C(i,\alpha)$ will provide us with the value of $L_i(x)$ with a high probability if we assume that 2^t requests are enough. The counterpart of this technique is that we have to guess the real values of $R_j(x)$ for all j but since t is of logarithmic size this is achievable.

We now give a more formal proof with a small difference: we use fast Walsh transform computations to simultaneously compute the 2^t results of the votes on the $C(i,\alpha)$ values for all the 2^t possible t-tuples of assumptions $R_j(x)$, $1 \le j \le t$, instead of computing them independently.

Before we give the proof, we need to recall some results on the Walsh transform. Given a real function of t binary variables $g(x_1,\ldots,x_t)$, the Walsh transform of g is the real function of t binary variables $G = W(g)$ defined by

$$G(u_1,\ldots,u_t) = \sum_{x_1,\ldots,x_t \in \{0,1\}^t} f(x_1,\ldots,x_t)(-1)^{u_1x_1+\ldots+u_tx_t}$$

It is known that the time needed to compute the Walsh transform of a function of t binary variables is $t \cdot 2^t$.

Proof. The algorithm C works as follows : first it randomly selects t elements $R_1,\ldots,R_t$ of the n-dimensional vector space over $GF(2)$ of the n-bit to 1-bit linear forms.

Then for each $i = 1,\ldots,n$ it executes the following process: for all the 2^t possible $\alpha = (\alpha_1,\ldots\alpha_t)$ t-tuples $\in \{0,1\}^t$ store $(-1)^{B(\bigoplus_j \alpha_j r_j \oplus L_i, f(x))}$ in a table of size 2^t, say $(c_0,\ldots c_{2^t-1})$ (thus the coefficient associated with α is $c_{\sum_{j=0}^{t-1} \alpha_j \cdot 2^{j-1}}$). Then it applies the Walsh transform to this table (which represents a function of α. This gives 2^t numbers $(\beta_0^i,\ldots,\beta_{2^t-1}^i)$ such that

$$\begin{aligned} \beta_k^i &= \sum_\alpha (-1)^{B(\bigoplus_j \alpha_j R_j \oplus L_i, f(x))}(-1)^{<k,\alpha>} \\ &= |\{\alpha | C(i,\alpha) = 0\}| - |\{\alpha | C(i,\alpha) = 1\}| \end{aligned}$$

β_k^i is the difference of the number of 0 votes and 1 for $L_i(x)$ corresponding to the assumption that $R_j(x) = k_j$ for all j comprised between 1 and t. Consequently if β_k^i is positive, then C sets bit i of the n-bit candidate value C_k associated with the assumption k to $C_k^i = 0$, otherwise this bit is set to $C_k^i = 1$.

After this process has been completed for all the n values of i, one is left with a list of 2^t n-bit candidate values for x corresponding to each of the 2^t assumptions for $R_1(x)$ to $R_t(x)$. For each candidate value C_k, algorithm C then computes $f(C_k)$ and compares it to $f(x)$. If a match occurs, C keeps C_k in the list of at most 2^t candidate values for x it outputs, otherwise C_k is discarded from the list.

The total running time of algorithm C is $n2^t(T + t + 2) + 2^t T_f$ where T_f is the time needed to compute $f(y)$ for an n-bit value y.

Let us now upper bound the probability that algorithm C fails to select x in the list of pre-images of $f(x)$ it produces. Over the 2^t assumptions for $R_1(x)$ to $R_t(x)$, only the correct one is to be considered. The failure probability of C is upper bounded by the sum of the n probabilities p_i that the vote for $L_i(x)$ is incorrect and we have:

$$p_i = Pr\left\{|\{\alpha|C(i,\alpha) = L_i(x)\}| < \frac{2^t}{2}\right\}$$

$|\{\alpha|C(i,\alpha) = L_i(x)\}|$ is the sum of the 2^t pairwise independent 0-1 variables $C(i,\alpha) \oplus L_i(x) \oplus 1$ of average value $\mu_\alpha \geq \frac{1}{2} + \frac{\epsilon}{2}$ and variance $v_\alpha = \frac{1}{4} - \frac{\epsilon^2}{4}$. Thus p_i has average value $\mu = 2^t\left(\frac{1}{2} + \frac{\epsilon}{2}\right)$ and variance $\sigma^2 = 2^t\left(\frac{1}{4} - \frac{\epsilon^2}{4}\right)$. By applying Chebyshev's inequality, we have

$$\begin{aligned} p_i &= Pr\left\{\sum_\alpha C(i,\alpha) \oplus L_i(x) \oplus 1 < \frac{2^t}{2}\right\} \\ &= Pr\left\{\sum_\alpha C(i,\alpha) \oplus L_i(x) \oplus 1 - \mu < -\frac{2^t\epsilon}{2}\right\} \\ &\leq Pr\left\{\left|\sum_\alpha C(i,\alpha) \oplus L_i(x) \oplus 1 - \mu\right| > \frac{2^t\epsilon}{2}\right\} \leq \frac{\sigma^2}{(2^t\frac{\epsilon}{2})^2} \leq \frac{1}{2^t\epsilon^2} \end{aligned}$$

Thus the failure probability of C is upper bounded by $\frac{n}{2^t\epsilon^2}$. If we want to have a probability of success for algorithm C higher than $\frac{1}{2}$, then we have to choose t such that $2^t = \frac{n}{\epsilon^2}$. Finally the total complexity of algorithm C is given by

$$\frac{2n^2}{\epsilon^2}\left(T + \log(\frac{2n}{\epsilon^2}) + 2\right) + \frac{2n}{\epsilon^2}T_f$$

On the Security of IV Dependent Stream Ciphers

Côme Berbain and Henri Gilbert

France Télécom R&D
38–40, rue du Général Leclerc
92794 Issy les Moulineaux Cedex 9 — France
{firstname.lastname}@orange-ftgroup.com

Abstract. Almost all the existing stream ciphers are using two inputs: a secret key and an initial value (IV). However recent attacks indicate that designing a secure IV-dependent stream cipher and especially the key and IV setup component of such a cipher remains a difficult task. In this paper we first formally establish the security of a well known generic construction for deriving an IV-dependent stream cipher, namely the composition of a key and IV setup pseudo-random function (PRF) with a keystream generation pseudo-random number generator (PRNG). We then present a tree-based construction allowing to derive a IV-dependent stream cipher from a PRNG for a moderate cost that can be viewed as a subcase of the former generic construction. Finally we show that the recently proposed stream cipher QUAD [3] uses this tree-based construction and that consequently the security proof for QUAD's keystream generation part given in [3] can be extended to incorporate the key and IV setup.
Keywords: stream cipher, PRNG, IV setup, provable security

1 Introduction

Stream ciphers and block ciphers are the two most popular families of symmetric encryption algorithms. Unlike block ciphers, stream ciphers do not produce a key-dependent permutation over a large blocks space, but a key-dependent sequence of numbers over a small alphabet, typically the binary alphabet $\{0, 1\}$. To encrypt a plaintext sequence, each plaintext symbol is combined with the corresponding symbol of the keystream sequence using a group operation, usually the exclusive or operation over $\{0, 1\}$.

Nearly all stream ciphers specified recently use two inputs to generate a keystream sequence: a secret key and an additional parameter named the initial value (IV), that is generally not secret. The purpose of the initial value is to allow

[0] The work described in this paper has been supported by the European Commission through the IST Program under Contract IST-2002-507932 ECRYPT.

to derive several "independent" keystream sequences from one single key, and thus to provide a convenient method for encrypting several plaintext sequences under the same secret key, by "resynchronizing" the stream cipher each time with a new IV value. This represents an obvious practical advantage over formerly proposed stream ciphers which single input was the secret key. But on the other hand, the use of an IV input has considerable impacts on the cryptanalysis and on the formalization of the security requirements on stream ciphers.

As for cryptanalytic implications, the quite numerous attacks of IV-dependent stream ciphers published in the past years clearly indicate that IVs result in additional attack opportunities, and that the key and IV setup procedure still represents one of the less well understood aspects of stream ciphers design. As a matter of fact an adversary can compare the keystream sequences associated with several known, related or chosen IV values, and potentially derive information upon the corresponding internal state values that could not be derived from one single keystream sequence. This is illustrated by Fluhrer, Mantin, and Shamir's attack on the key and IV loading method of the RC4-based cipher used in certain WiFi systems [10], by Ekdahl and Johansson's cryptanalysis of the GSM cipher A5/1 [9], by Joux and Muller's differential known or chosen IV attacks on various ciphers [16, 17], by Daemen, Govaerts, and Vandewalle's and by Armknecht, Lano, and Preneel's resynchronization attacks [8, 1] or more recently by attacks against some of the eSTREAM candidates.

As for the implications of IVs on the formalization of security requirements on stream ciphers, they can be outlined as follows:

In the case of a stream cipher without IV, the requirements are conveniently captured by the theory of pseudo-random numbers generators which has been stemming from the seminal work of Shamir [18], Yao [19], Blum and Micali [6] in the early 80's. A stream cipher is considered secure if the associated key to keystream function is a pseudo-random number generator (PRNG), i.e. an input-expanding function allowing to expand a short seed (the key) into a strictly longer output (the keystream) in such a way that if the secret input seed is uniformly distributed, then the probability distribution of the corresponding output is computationally indistinguishable with non negligible probability from the uniform distribution.

In the case of an IV-dependent stream cipher, no as unanimously accepted formalization of the security requirements has emerged so far. However, most cryptologists would probably agree that a sufficient security condition is that the associated function generator which maps the secret key onto the IV to keystream function be a pseudo-random function generator (PRF), i.e. a random function generator indistinguishable with non negligible probability from a perfect random function generator. To quote an example, this is the condition Halevi, Coppersmith, and Jutla are using to express the security requirements on the IV-dependent stream cipher Scream [15]. We will briefly discuss the validity of this PRF-based formalization in Section 3 hereafter, and conclude that it indeed captures the most natural generalization to IV-dependent stream ciphers of the well accepted (PRNG based) formalization of IV less stream ciphers.

One might argue that since constructing a secure PRF can be expected to be more demanding than constructing a secure PRNG and nearly as difficult as constructing a block cipher, introducing IVs in stream ciphers looses all performance advantages of stream ciphers over block ciphers and requires the same kind of techniques than designing a block cipher. This is however not necessarily the case, as will be shown in the sequel.

The purpose of this paper is twofold. Firstly, to clarify the security requirements upon an IV-dependent stream cipher (Section 3) and to identify sufficient conditions on its key and IV setup and key generation parts in order for the whole stream cipher to be secure (Section 4). Secondly, to propose a practical construction allowing to meet these conditions (Section 5), and therefore to derive an IV-dependent stream cipher with a provable security argument. Finally we show that as an application of this construction the security arguments of QUAD can be extended in order to include the key and IV setup (Section 6). An overview of our main results is given in Section 2.

2 Outline of our results

For all the stream ciphers we consider in this paper, the keystream derivation is split, as in nearly all existing IV-dependent stream ciphers, into the two following separate phases, according to the generic construction illustrated in Figure 1:

- (1) **Key and IV setup:** an m-bit initial state value is derived from the key and IV value.
- (2) **Keystream generation:** the keystream is derived from the m-bit initial state obtained in the key and IV-setup phase. For that purpose, the m-bit initial state is taken as the seed input of a number generator [1].

We formally establish, in Section 4, the validity of the following "folklore" belief implicitly invoked in the security argumentation of several existing IV-dependent stream ciphers [5, 2]: if the family $\{F_K\}$ of IV to initial state functions parametrized by the key K is a PRF and if the number generator g is a PRNG, then the family $\{G_K = g \circ F_K\}$ of IV to keystream sequence functions is a PRF. This provides useful sufficient conditions in order for the IV-dependent stream cipher resulting from the generic construction of Figure 1 to be secure. Our security proof relies upon a simple "composition theorem". A specific construction directly suggested by the former security results might consist of using a trusted block cipher for the IV-setup, as done for instance in the stream cipher candidates LEX [5] and SOSEMANUK [2], both selected as focus ciphers for

[1] Though our constructions are potentially applicable to any number generator with a sufficiently long input size, they are mainly intended for number generators based upon the iterated invocation of a finite state machine (FSM). The keystream sequence generation procedure of nearly all existing stream ciphers has this specific structure, i.e. uses the state transition function of a FSM to update an m-bit internal state and derive at each iteration a t-bit keystream portion by means of a fixed output function.

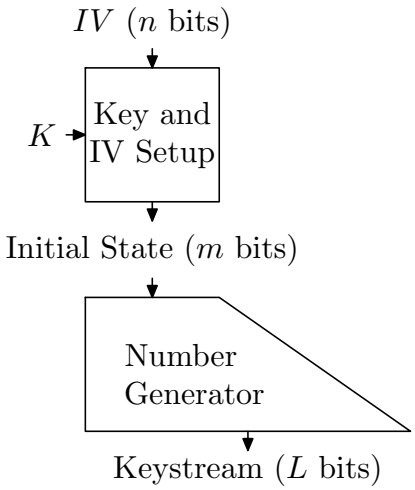


Fig. 1. IV-dependent stream cipher: (generic construction)

third evaluation phase of the stream cipher initiative eSTREAM of the European network ECRYPT. One may however argue that this construction results in a lack of design unity when (unlike in LEX) the keystream generation does not reuse the trusted block cipher used for the key and IV setup.

We propose, in Section 5 hereafter, another specific construction also supported by former security results, which has the additional advantage that it better preserves the design unity of stream ciphers. This construction consists of applying the so called tree-based construction proposed by Goldreich, Goldwasser, and Micali in [13] for deriving the PRF needed for the key and IV setup from any n-bit to $2n$-bit PRNG. This PRNG can be essentially the same as the one used in the keystream generation phase. The later option allows to even better preserve the unity of the design, and to achieve substantial savings in the hardware and software implementation complexity of the stream cipher, since the key and IV setup and the keystream generation are then using the same computational ingredients.

Last of all, we focus in Section 6 on a particular stream cipher where the tree-based construction of Section 5 is applied in the key and IV setup, namely the recently proposed stream cipher QUAD [3]. We show that the partial proof of security of [3] (which gives some evidence that the keystream generation part of QUAD is secure) can be extended to incorporate the key and IV setup. This allows to reduce the security of the whole stream cipher to the difficulty of solving a random multivariate quadratic system.

3 Security Model

3.1 Basic Security Notions

We first recall definitions of advantages for distinguishing a number generator from a perfect random generator and a function generator from a perfect random function generator, and the notions of Pseudo-Random Number Generator

(PRNG) and Pseudo-Random Function (PRF). All the security definitions used throughout this paper relate to the concrete (non asymptotic) security model. In the sequel, when we state that a value u is randomly chosen in a set U, we implicitly mean that u is drawn according to the uniform law over U.

Single-query distinguisher for a number generator: let us consider a number generator $g : \{0,1\}^n \longrightarrow \{0,1\}^L$ with input and output lengths $L > n$, used to expand an n-bit secret random seed into an L-bit sequence. A distinguisher in time t for g is a probabilistic testing algorithm A which when input with an L-bit string outputs either 0 or 1 with time complexity at most t. We define the advantage of A for distinguishing g from a perfect random generator as

$$\mathbf{Adv}_g^{prng}(A) = \left|\Pr_{x\in\{0,1\}^n}(A(g(x)) = 1) - \Pr_{y\in\{0,1\}^L}(A(y) = 1)\right|,$$

where the probabilities are not only taken over the value of an unknown randomly chosen $x \in \{0,1\}^n$ (resp. of a randomly chosen $y \in \{0,1\}^L$), as explicitly stated in the above formula, but also over the random choices of the probabilistic algorithm A.

We define the advantage for distinguishing the function g in time t as

$$\mathbf{Adv}_g^{prng}(t) = \max_A\{\mathbf{Adv}_g^{prng}(A)\},$$

where the maximum is taken over all testing algorithms of time complexity at most t.

A function g is said to be a PRNG if $\mathbf{Adv}_g^{prng}(t)$ is negligible (for example less than 2^{-40}) for values of t strictly lower than a fixed threshold (for example 2^{80} or 2^{128}). The definition of a PRNG is therefore dependent upon thresholds reflecting the current perception of an acceptably secure number generator.

Multiple-query distinguisher for a number generator: let us still consider a function g from n bits to L bits. A q-query distinguisher in time t for g is a probabilistic testing algorithm A which when input with a q-tuple of L-bit words outputs either 0 or 1 with time complexity at most t. We define the advantage of A for distinguishing g from a perfect random generator as

$$\mathbf{Adv}_g^{prng}(A) = \left|\Pr(A(g(x_1),\ldots,g(x_q)) = 1) - \Pr(A(y_1,\ldots,y_q) = 1)\right|,$$

where the probabilities are taken over the q-tuples of n-bit values x_i (resp. of L-bit values y_i) and on the random choices of the probabilistic algorithm A. We also define the advantage for distinguishing the function g in time t with q queries as

$$\mathbf{Adv}_g^{prng}(t,q) = \max_A\{\mathbf{Adv}_g^{prng}(A)\},$$

where the maximum is taken over all testing algorithms A having time-complexity at most t and using q inputs.

Distinguisher for a function generator: let us now consider a function generator, i.e. a family $F = \{f_K\}$ of $\{0,1\}^n \longrightarrow \{0,1\}^m$ functions indexed by a key K randomly chosen from $\{0,1\}^k$. A distinguisher in time t with q queries for F is a probabilistic testing algorithm A^f able to query an n-bit to m-bit oracle function f up to q times. Such an algorithm allows to distinguish a randomly chosen function f_K of F from a perfect random function f^* randomly chosen in the set $F^*_{n,m}$ of all $\{0,1\}^n \longrightarrow \{0,1\}^m$ functions with a distinguishing advantage

$$\mathbf{Adv}^{prf}_F(A) = \left|\Pr(A^{f_K} = 1) - \Pr(A^{f^*} = 1)\right|,$$

where the probabilities are taken over $K \in \{0,1\}^k$ (resp $f^* \in F^*_{n,m}$) and over the random choices of A. We define the advantage for distinguishing the family F in time t with q queries as

$$\mathbf{Adv}^{prf}_F(t,q) = \max_A\{\mathbf{Adv}^{prf}_F(A)\},$$

where the maximum is taken over all testing algorithms A working in time at most t and capable to query an n-bit to m-bit oracle function up to q times.

A family of functions $F = \{f_K\}$ is said to be a PRF if $\mathbf{Adv}^{prf}_F(t,q)$ is negligible for values of t and q strictly lower than the respective threshold (for example 2^{80} or 2^{128} for t and 2^{40} for q).

3.2 Security Requirements for an IV-dependent Stream Ciphers

Let us consider an IV-dependent stream cipher, i.e. a family $G = \{g_K\}_{K\in\{0,1\}^k}$ of IV to keystream functions $g_K : \{0,1\}^n \longmapsto \{0,1\}^L$, where k is the size of the key, n is the size of the IVs and L is the maximum number of keystream bits that can be produced for a given IV.

Such an IV-dependent stream cipher can be viewed as a special number generator, allowing to expand a k-bit secret seed onto an exponentially long sequence of 2^n L-bit keystream words $\{Z_{IV} = g_K(IV)\}_{IV\in\{0,1\}^n}$, with the additional property that while this sequence is too long to be entirely accessed in a sequential manner, it can be directly accessed, i.e. that for any value of $IV \in \{0,1\}^n$ the computational cost for accessing the L-bit subsequence Z_{IV} is constant.

Fig. 2. Exponentially long sequence with direct access associated to an IV-dependent stream cipher g

This observation can be used in order to try to generalize the well accepted formalization of the security requirements on an IV-less stream cipher by means

of a PRNG in a natural manner. An IV-less stream cipher is considered secure if and only if no testing algorithm, when given access either to a Λ-bit output of the generator corresponding to a random secret input or to a random Λ-bit sequence, can distinguish both situations in time less than a sufficiently large threshold (say 2^{80}) with a non-negligible advantage. It can be reasonably argued that in the case of an IV-dependent stream cipher, the most natural generalization of the above security definition is to require that no testing algorithm, when given a sufficiently large number q (say for instance $\min(2^{80}, 2^n)$) of direct accesses to L-bit subsequences of a sequence $\{Z_{IV}\}$ associated with a random unknown key K or to a uniformly drawn sequence of 2^n L-bit subsequences, can distinguish both situations in time less than a sufficiently large threshold (say 2^{80}) with a non-negligible advantage.

But it is easy to see that both the sequence $\{Z_{IV}\}$ and the uniformly drawn sequence of $2^n \cdot L$ bits can be viewed as n-bit to L-bit functions, and that direct accesses to these sequences can be viewed as oracle queries to these functions. Therefore the requirements formulated above are strictly equivalent to saying that the function family $G = \{g_K\}_{K \in \{0,1\}^k}$ is a PRF.

In other words, we have given some evidence that an IV-dependent stream cipher $G = \{g_K\}$ for $K \in \{0,1\}^k$ with $g_K : \{0,1\}^n \longrightarrow \{0,1\}^L$ can be viewed as a family of functions, and can be considered secure if and only if it is a PRF , i.e. sufficiently indistinguishable from a perfect random function.

Related key attacks, which relevance, when it comes to security requirements on symmetric ciphers, is a controversial issue [4], are not covered by our security model.

4 Security of the Generic Construction

4.1 A Simple Composition Theorem

In this Section, we define the composition G of a family of function F and a function g, relate the indistinguishability of G to the one of F and g, and show that this composition theorem results in a secure construction allowing to derive a secure IV-dependent stream cipher from a PRF and a PRNG.

Definition 1. *The composition $G = g \circ F$ of an n-bit to m-bit family of functions $F = \{f_K\}$ and of an m-bit to L-bit function g is the n-bit to L-bit family of functions*

$$G = \{g \circ f_K\}.$$

Theorem 1. *Let us consider a PRF $F = \{f_K\}$ where $f_K : \{0,1\}^n \longrightarrow \{0,1\}^m$ and a PRNG $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ that produces L bits in time T_g^L. The advantage in time t with q queries of $G = g \circ F = \{g \circ f_K\}$ can be upper bounded as follows*

$$\mathbf{Adv}_G^{prf}(t,q) \leq \mathbf{Adv}_F^{prf}(t + qT_g^L) + q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

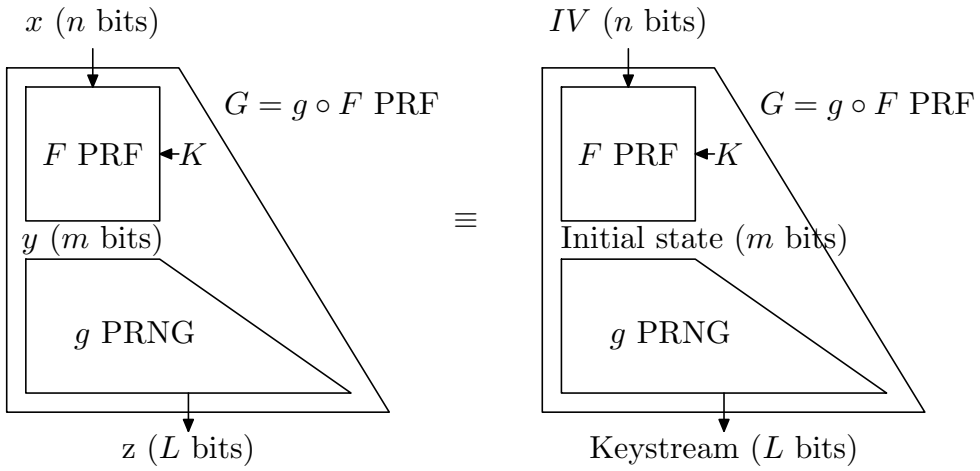


Fig. 3. Composing a PRF and a PRNG gives a PRF

In order to prove Theorem 1 we first establish a useful lemma which relates the single-query and multiple-queries advantages of any PRNG.

Lemma 1. *Let $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ be a PRNG which can be computed in time T_g^L. The q-query advantage for distinguishing g in time t is related to the single-query advantage for distinguishing g by the inequality*

$$\mathbf{Adv}_g^{prng}(t,q) \leq q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

A proof of the above lemma is given in Appendix 1; this proof is similar to the one of a proposition relating the single-sample indistinguishability and the multiple-sample indistinguishability of polynomial-time constructible ensembles established by Goldreich and Krawczyk [14, 12]. Using this lemma, we can prove Theorem 1. Our proof is given in Appendix 2. Theorem 1 is illustrated on the left part of Figure 3.

Application to the security of IV-dependent stream ciphers. A direct application of Theorem 1 is depicted on the right part of Figure 3. As said in Section 3, an IV-dependent stream cipher can be considered secure if and only if the IV to keystream function g_K parametrized by the key is a PRF. Theorem 1 implies that this is indeed the case, i.e. that the stream cipher is secure if:
1) the n-bit to m-bit IV to initial state function parametrized by the key representing the IV setup of a stream cipher is a PRF;
2) the m-bit to L-bit initial state to keystream function is a PRNG;
3) the upper bounds on the advantage for distinguishing $\{g_K\}$ given by Theorem 1 guarantee a sufficient resistance against attacks.

From now on, we consider the following stream cipher design problem. Let us assume that a trusted number generator g allowing to expand an m-bit initial state into an L-bit sequence is available. We are now faced with the issue of constructing a key and IV setup PRF in order to compose this PRF with g to get a secure IV-dependent stream cipher.

A straightforward construction for such a PRF, directly suggested by the former security results, would consist of using a trusted block cipher with a sufficient block-length to fit the initial state length m of g. As a matter of fact it is usual to conjecture that the family of key dependent encryption permutations associated with a secure block cipher represents both a pseudo-random family of permutations (PRP) and a PRF. The value of m must be typically at least 160 bits in the frequent case where g has a finite state automaton structure, in order to avoid generic time-memory trade-offs. This suggests that one could for instance use the variant of Rijndael with a 256-bit block size, or a truncated instance of this cipher if m is smaller than 256, or an appropriate "one to many blocks" mode of operation of any block cipher [11] for initial state sizes larger than 256 bits. This would allow to accommodate keys and IVs of size up to one block. Such an approach can certainly be considered more conservative than the key and IV setup procedure of many existing stream ciphers. On the other hand, one may argue that it results in a strong performance penalty for the encryption of short messages and an increased implementation complexity, and in a lack of design unity if the trusted number generator g does not reuse the same ingredients as the trusted block cipher.

5 A Tree Based Stream Cipher Construction

Is this Section we present a key and IV setup procedure derived from the Tree Based Construction introduced by Goldreich, Goldwasser, and Micali in [13]. This construction allows to derive a PRF from a PRNG and to relate their securities. Though initially introduced for theoretical purposes (namely show in the asymptotic model that the existence of a PRNG implies the existence of a PRF) it is also of practical interest since it allows, as shown here, to build a stream cipher from two number generators: the Tree Based Construction is used to transform the first number generator into an efficiently computable key and IV setup; the second number generator is initialized with the value given by the key and IV setup, and generates the keystream. These two number generators can advantageously be the same. Thus it becomes possible to build an IV-dependent stream cipher from one single number generator.

5.1 The Tree Based Construction

The Tree Based Construction allows to derive a PRF F^g from a PRNG g. Let us consider a PRNG $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ where $L \geq 2m$ and let us denote the L bit image of $y \in \{0,1\}^m$ by $g(y) = z_0 z_1 \dots z_{L-1}$. We derive from g two functions $g_0 : y \in \{0,1\}^m \longmapsto z_0, \dots, z_{m-1}$ and $g_1 : y \in \{0,1\}^m \longmapsto z_m, \dots, z_{2m-1}$ from m bits to m bits which on input $y \in \{0,1\}^m$ respectively produce the first and the second m-bit string generated by g when input with y.

The PRF F^g is the family of functions $\{f_y\}_{y \in \{0,1\}^m}$ where

$$\begin{aligned} f_y : \{0,1\}^n &\longrightarrow \{0,1\}^m \\ (x_1, x_2, \dots, x_n) &\longmapsto f_y(x_1, x_2, \dots, x_n) = g_{x_n} \circ g_{x_{n-1}} \dots \circ g_{x_1}(y) \end{aligned}$$

This construction is illustrated on Figure 4: the input bits determine a path trough the binary tree leading to the output of the function.

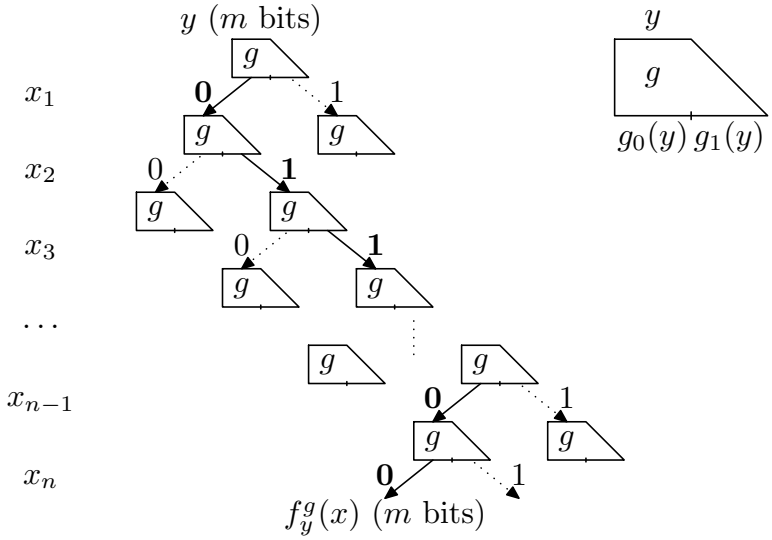


Fig. 4. Tree Based Construction

Theorem 2. *Let $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ be a PRNG which generates $L \geq 2m$ outputs bits and produces its $2m$ first output bits in time T_g^{2m} and let $F^g = \{f_y\}_{y\in\{0,1\}^m}$ be the family of n-bit to m-bit functions derived from g by the Tree Based Construction. The (t,q) advantage of PRF F^g is related to the single-query advantage of PRNG g by the following inequality:*

$$\mathbf{Adv}_{F^g}^{prf}(t,q) \leq nq\mathbf{Adv}_g^{prng}(t + q(n+1)T_g^{2m}).$$

A proof of Theorem 2 is given in Appendix 3. This proof is essentially the same as the security proof of the Tree Based Construction due to Goldreich, Goldwasser, and Micali[13], a detailed version of which is given by Goldreich in [12], up to the fact that we consider the concrete security model instead of the asymptotic (polynomial time indistinguishability) security model.

5.2 Resulting Stream Cipher Construction

To build an IV-dependent stream cipher from a m-bit to L-bit PRNG g representing an IV-less stream cipher, we apply the Tree Based Construction to a m to L'-bit PRNG g' ($L' \geq 2m$) typically equal to g itself, in order to derive the key and IV setup function, which produces the initial state of g, following the construction of Figure 4. For a key K and an IV IV, the value $f_K^{g'}(IV)$ is the initial state of g. Thanks to Theorem 1 and since the Tree Based Construction provides a PRF, the resulting stream cipher is also a PRF. The security of the final stream cipher only depends on the security of the PRNG.

In case K is smaller than m bits, we need to extend K to a value randomly chosen in $\{0,1\}^m$. In order to achieve this we can use an additional PRNG $h : \{0,1\}^k \longrightarrow \{0,1\}^m$. The proof for the Tree Based Construction can easily be extended and we have:

$$\mathbf{Adv}^{prf}_{F^g}(t,q) \leq nq\mathbf{Adv}^{prng}_{g'}(t + q(n+1)T^{2m}_{g'}) + q\mathbf{Adv}^{prng}_{h}(t + q(n+1)T^{m}_{h}).$$

Theorem 3. *Let $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ be a PRNG which generates L outputs bits in time T^L_g and $g' : \{0,1\}^m \longrightarrow \{0,1\}^{2m}$ be a PRNG. We can define a stream cipher $G = \{G_K\} = g \circ F^{g'}$, where $F^{g'}$ is derived from g' using the Tree Based Construction*

$$G_K(IV) = g \circ f^{g'}_K(IV)$$

Moreover G is a PRF and we have:

$$\mathbf{Adv}^{prf}_{G}(t,q) \leq nq\mathbf{Adv}^{prng}_{g'}(t + qT^L_g + q(n+1)T^{2m}_{g'}) + q\mathbf{Adv}^{prng}_{g}(t + qT^L_g)$$

If g and g' are equal, then we have

$$\mathbf{Adv}^{prf}_{G}(t,q) \leq q(n+1)\mathbf{Adv}^{prng}_{g}(t + q(n+2)T^L_g)$$

Proof. To prove this result we only have to use Theorem 1 and Theorem 2.

The above key and IV setup construction is of practical interest: suppose we have a trusted PRNG, for example the Shrinking Generator [7], we can build an IV-dependent stream cipher based on this PRNG without introducing any additional feature for a moderate computational cost.

5.3 Efficiency Considerations

Let us assume for instance that we want to build from a PRNG g of initial state length 160 bits a stream cipher with a 160-bit key, a 80-bit IV, and a target security of 2^{80}, using the previously described construction. Then the time required to compute the key and IV setup with the above construction is the time required by g to produce 3200 bytes. Considering a very fast PRNG, running at 5 cycles per byte, the key and IV setup requires about 16000 cycles on a standard PC. This is slower than using a block cipher like AES, which would require about 1000 cycles. Therefore for software applications where resynchronization has to be done frequently, this construction is not at all efficient and using a block cipher for the key and IV setup should be considered. However for applications where the keystream generated for a single IV is very long compared to these 3200 bytes our construction can be competitive.

Now in the case of hardware applications, the above construction can be of real interest, in order to minimize the hardware complexity since it uses a single PRNG for the key and IV setup and the keystream generation. Then only a few additional gates are required to implement the key and IV setup. If a block cipher were used instead, then the total number of gates required to implement the stream cipher would be much higher than the number of gates required for a PRNG.

6 Application to the QUAD Stream Cipher

The stream cipher QUAD is a practical stream cipher with some provable security which was introduced [3] by Berbain, Gilbert, and Patarin at Eurocrypt 2006. The provable security argument relates, in the $GF(2)$ case, the indistinguishability of the keystream generated by QUAD to the conjectured hardness of solving random quadratic systems. QUAD iterates a one way function, namely a quadratic system, upon an internal state and extracts a certain number of bits of this step at each iteration.

The keystream generation makes use of two systems $S_{\text{in}} = (Q_1, \ldots, Q_m)$ and $S_{\text{out}} = (Q_{m+1}, \ldots, Q_{km})$ of multivariate quadratic equations both sharing the same m unknowns over $\text{GF}(q)$, typically $GF(2)$ as described on Fig. 5. The first system S_{in} is used to update the internal state and thus contains m equations, whereas the second system S_{out} produces the keystream and contains $(k-1)m$ equations. As explained in [3], the quadratic systems S_{in} and S_{out}, though randomly generated, are both publicly known.

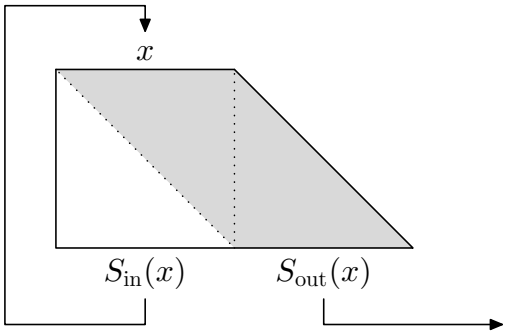


Fig. 5. Stream cipher QUAD

Given an internal state $x = (x_1, \ldots, x_m)$, the keystream generation amounts to iterating the following steps:

- compute $\big(S_{\text{in}}(x), S_{\text{out}}(x)\big) = \big(Q_1(x), \ldots, Q_{km}(x)\big)$, from the internal state x;
- output the sequence $S_{\text{out}}(x) = \big(Q_{m+1}(x), \ldots, Q_{km}(x)\big)$ of $(k-1)m$ keystream elements of $\text{GF}(q)$;
- update the internal state x with the sequence $S_{\text{in}}(x) = \big(Q_1(x), \ldots, Q_m(x)\big)$.

Before generating any keystream the internal state x needs to be initialized, with the key K and the initialization vector IV, which are respectively represented by a sequence of $GF(q)$ elements of length $|K|$ and a binary sequence of $\{0, 1\}$ values of length $|IV|$. We will assume in the sequel that $|K|$ is equal to m. The initialization is done as follows: two publicly known chosen multivariate quadratic systems S_0 and S_1 of m equations over m unknowns are used. The initial state is filled with the key. Then for each of the $|IV|$ bits IV_1 to $IV_{|IV|}$ of the IV value the internal state x is updated as follows: if $IV_i = 0$, x is replaced

by the $GF(q)^m$ value $S_0(x)$; if $IV_i = 1$, x is replaced by the $GF(q)^m$ value $S_1(x)$. Finally the cipher is clocked m additional times as described before, but without outputting the keystream.

We now extend the partial proof over GF(2) given in [3] to incorporate the key and IV Setup. We denote by $S = (S_{\text{in}}||S_{\text{out}})$ the randomly chosen system of km equations on m variables and $S' = (S_0||S_1)$ the randomly chosen system of $2m$ equations in m variables. We also denote by $g^S : \{0,1\}^m \longrightarrow \{0,1\}^L$ and $g^{S'} : \{0,1\}^m \longrightarrow \{0,1\}^{2m}$ the corresponding PRNGs.

The key and IV setup proposed in [3] can be divided into two parts: in the first part the Tree Based Construction for $g^{S'}$ is applied and the value $y = f_K^{g^{S'}}(IV)$ is computed. Then in the second part, y is used to initialize g^S which is clocked m times but the corresponding $(k-1)m^2$ bits of keystream are not used. The value of the internal state after these m clocks is used to produce L bits of keystream. The security of the first part is related to the security of $g^{S'}$ thanks to Theorem 2.

$$\mathbf{Adv}^{prf}_{F^{g^{S'}}}(t,q) \leq 2q\mathbf{Adv}^{prng}_{g^{S'}}(t + q(n+2)T^{2m}_{g^{S'}})$$

The security of the PRNG g_{real} which starts by running m clocks like g^S without producing any keystream to reflect the runup of QUAD and then produces L bits of keystream as g^S does, is related to the security of generator $\tilde{g}^S : \{0,1\}^m \longrightarrow \{0,1\}^{L+(k-1)m^2}$ which iterates S to produce $L + (k-1)m^2$ bits, since g_{real} produces the same keystream as $\tilde{g}^S$ up to the fact that the first $(k-1)m^2$ bits of $g^{S'}$ are discarded. Consequently a distinguisher on g_{real} is also a distinguisher for $\tilde{g}^S$. Thus the advantage of g_{real} is upper-bounded by the advantage of $\tilde{g}^S$.

$$\mathbf{Adv}^{prng}_{g_{real}}(t) \leq \mathbf{Adv}^{prng}_{\tilde{g}^S}(t + T^{L+(k-1)m^2}_{\tilde{g}^S})$$

Finally the security of the stream cipher is related to the securities of $\tilde{g}^S$ and of $g^{S'}$ by the composition theorem of Section 4.

$$\mathbf{Adv}^{prf}_{\text{QUAD}}(t,q) \leq 2q\mathbf{Adv}^{prng}_{g^{S'}}(t+q(n+3)T^L_{g^{S'}})+q\mathbf{Adv}^{prng}_{\tilde{g}^S}(t+qT^L_{\tilde{g}^S}+T^{L+(k-1)m^2}_{\tilde{g}^S})$$

Those two generators are based on the iteration of a randomly chosen quadratic system of km equations in m variables (with $k = 2$ for S'). We can use the main result of [3], which relates the security of this kind of PRNG to the difficulty of inverting a randomly chosen multivariate quadratic system to say that the security of QUAD as a stream cipher is related to the difficulty of the MQ Problem, i.e. an adversary able to distinguish QUAD from a PRF in time t with q queries is then able to construct a MQ solver:

$$\begin{aligned}\mathbf{Adv}^{prf}_{\text{QUAD}}(t,q) &\leq 3q\mathbf{Adv}^{prng}_{g^S}(t + qT^{L+(k-1)m^2}_{\tilde{g}^S}) \\ &\leq \mathbf{Adv}^{MQinversion}(t'[t + qT^{L+(k-1)m^2}_{\tilde{g}^S}]),\end{aligned}$$

where $t'[t + qT^{L+(k-1)m^2}_{\tilde{g}^S}]$ is the running time of the MQ inverter algorithm given by the main reduction theorem of [3], which is recalled in Appendix.

7 Conclusion

In this paper we investigated security issues arising for IV-dependent stream ciphers. We confirmed the "folklore" belief that the composition of a key and IV setup PRF and a key generation PRNG provides a secure stream cipher, which furnishes a proof that initializing a PRNG with a block cipher is secure provided that the block cipher's block length is sufficiently large. Moreover we described a practical construction that allows to derive an IV-dependent stream cipher from a PRNG (or equivalently an IV-less stream cipher) for a moderate additional cost. This construction is quite simple and does not require additional components. Finally we showed an application of this provably secure construction to the stream cipher QUAD, by incorporating the key and IV setup in the security proof given by the authors of QUAD. The resulting extended proof relates the security of the whole stream cipher (not only the keystream generation part) to the conjectured intractability of the MQ problem.

References

1. Frederik Armknecht, Joseph Lano, and Bart Preneel. Extending the Resynchronization Attack. In Helena Handschuh and Anwar Hasan, editors, *Selected Areas in Cryptography – SAC 2004*, volume 3357 of *Lecture Notes in Computer Science*, page 19. Springer-Verlag, 2004.
2. Côme Berbain, Olivier Billet, Anne Canteaut, Nicolas Courtois, Henri Gilbert, Louis Goubin, Aline Gouget, Louis Granboulan, Cédric Lauradoux, Marine Minier, Thomas Pornin, and Hervé Sibert. Sosemanuk, a Fast Software-Oriented Stream Cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/001, 2005. http://www.ecrypt.eu.org/stream.
3. Côme Berbain, Henri Gilbert, and Jacques Patarin. QUAD: a Practical Stream Cipher with Provable Security. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, Lecture Notes in Computer Science. Springer-Verlag, 2006.
4. Daniel J. Bernstein. Related-key attacks: Who cares? eSTREAM, ECRYPT Stream Cipher Project, 2006. http://www.ecrypt.eu.org/stream/phorum.
5. Alex Biryukov. A new 128 bit key Stream Cipher : LEX. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/001, 2005. http://www.ecrypt.eu.org/stream.
6. Manuel Blum and Silvio Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM J. Comput.*, 13(4):850–864, 1984.
7. Don Coppersmith, Hugo Krawczyk, and Yishay Mansour. The Shrinking Generator. In Douglas Robert Stinson, editor, *Advances in Cryptology – CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 22–39. Springer-Verlag, 1993.
8. Joan Daemen, Ren Govaerts, and Joos Vandewalle. Resynchronization Weaknesses in Synchronous Stream Ciphers. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 159–167. Springer-Verlag, 1993.
9. Patrik Ekdahl and Thomas Johansson. Another Attack on A5/1. *IEEE Transactions on Information Theory*, 49(1):284–289, 2003.
10. Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. In *Selected Areas in Cryptography*, pages 1–24, 2001.

11. Henri Gilbert. The Security of "One-Block-to-Many" Modes of Operation. In Thomas Johansson, editor, *Fast Software Encryption – FSE 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 376–395. Springer-Verlag, 2003.
12. Oded Goldreich. *The Foundations of Cryptography - Volume 1.* Cambridge University Press, 2001.
13. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *J. ACM*, 33(4):792–807, 1986.
14. Oded Goldreich and Hugo Krawczyk. Sparse Pseudorandom Distributions. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 113–127. Springer-Verlag, 1989.
15. Shai Halevi, Don Coppersmith, and Charanjit S. Jutla. Scream: A Software-Efficient Stream Cipher. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encrytion – FSE 2002*, volume 2365 of *Lecture Notes in Computer Science*, pages 195–209. Springer-Verlag, 2002.
16. Antoine Joux and Frédéric Muller. A Chosen IV Attack Against Turing. In Mitsuru Matsui and Robert Zuccherato, editors, *Selected Areas in Cryptography – SAC 2003*, volume 3006 of *Lecture Notes in Computer Science*, pages 194–207. Springer-Verlag, 2003.
17. Frédéric Muller. Differential Attacks against the Helix Stream Cipher. In Willi Meier and Roy Bimal, editors, *Fast Software Encryption – FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, page 0. Springer-Verlag, 2004.
18. Adi Shamir. On the Generation of Cryptographically Strong Pseudo-Random Sequences. In *ICALP*, pages 544–550, 1981.
19. Andrew Yao. Theory and Applications of Trapdoor Function. In *Foundations of Cryptography FOCS 1982*, 1982.

Appendix

Proof of Lemma 1

Lemma 1. Let us consider a PRNG $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ which can be computed in time T_g^L. Then we have

$$\mathbf{Adv}_g^{prng}(t,q) \leq q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

Proof. Suppose there is an algorithm A that distinguishes q L-bit keystream sequence produced by g from q unknown randomly chosen initial internal state $x_i \in \{0,1\}^m$ from q random L-bit sequences in time t with advantage ϵ. Then we are going to build an algorithm B that distinguishes $g(x)$ corresponding to an unknown random input x, from a random value of size L in time $t' = t + qT_g^L$ with advantage $\frac{\epsilon}{q}$.

We introduce the hybrid probability distributions D^i over $\{0,1\}^L$ for any $0 \leq i \leq q$ respectively associated with the random variables

$$Z^i = (g(x_1), g(x_2), \ldots, g(x_i), r_{i+1}, \ldots, r_q)$$

where the r_j and x_i are random independent uniformly distributed values of $\{0,1\}^L$ and $\{0,1\}^m$ respectively. Consequently D^q is the distribution of the q

L-bit keystream produced by g and D^0 is the distribution of q L-bit uniformly distributed values of $\{0,1\}^L$.

We denote by p^i the probability that A accepts a random qL-bit sequence distributed according to D^i.

We have supposed that algorithm A distinguishes between D^0 and D^q with advantage ϵ, in other words that $|p^0 - p^q| \geq \epsilon$.

Algorithm B works as follows : on input $y \in \{0,1\}^L$ it selects randomly an i such that $1 \leq i \leq q$ and constructs the vector

$$Z(y) = (g(x_1), g(x_2), \ldots, g(x_{i-1}), y, r_{i+1}, r_{i+2}, \ldots, r_q)$$

with x_i and r_i randomly chosen values. If y is distributed accordingly to the output distribution of g, i.e. $y = g(x)$ for a uniformly distributed value of x, then

$$Z(y) = (g(x_1), g(x_2), \ldots, g(x_{i-1}), g(x), r_{i+1}, r_{i+2}, \ldots, r_q)$$

is distributed according to D^i. Now if y is distributed according to the uniform distribution, then

$$Z(y) = (g(x_1), g(x_2), \ldots, g(x_{i-1}), y, r_{i+1}, r_{i+2}, \ldots, r_q).$$

Thus $Z(y)$ is distributed according to D^{i-1}. In order to distinguish the output distribution of g from the uniform law, algorithm B calls algorithm A with inputs $Z(y)$ and returns the value returned by A. Thus

$$\begin{aligned} &\big|\Pr_x(B(g(x)) = 1) - \Pr_y(B(y) = 1)\big| \\ &= \big|\frac{1}{q}\sum_{i=0}^{q-1} p^i - \frac{1}{q}\sum_{i=1}^{q} p^i\big| = \frac{1}{q}|p^0 - p^q| \geq \frac{\epsilon}{q}. \end{aligned}$$

Thus B distinguishes the output distribution of g from the uniform distribution with probability at least $\frac{\epsilon}{q}$ in time $t + qT_g^L$. Consequently we have:

$$\begin{aligned} \mathbf{Adv}_g^{prng}(A) &= q\mathbf{Adv}_g^{prng}(B) \\ &\leq q\mathbf{Adv}_g^{prng}(t + qT_g^L) \end{aligned}$$

which gives us the final result:

$$\mathbf{Adv}_g^{prng}(t,q) \leq q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

Proof of Theorem 1

Theorem 1. Let us consider $F = \{f_K\}$ where $f_K : \{0,1\}^n \longrightarrow \{0,1\}^m$ a PRF and $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ a PRNG that produces L bits in time T_g^L. The advantage in time t with q queries of $G = g \circ F = \{g \circ f_K\}$ can be upper bounded as follows

$$\mathbf{Adv}_G^{prf}(t,q) \leq \mathbf{Adv}_F^{prf}(t + qT_g^L) + q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

Proof. We want to upper bound the advantage of an algorithm A that in time t with q requests distinguishes a random instance $g_K = g \circ f_K$ of $G = g \circ F$ from a perfect random function $g^* \in F^*_{n,L}$. We can write the advantage of A as

$$\mathbf{Adv}^{prf}_G(A) = \big|\Pr(A^{g_K} = 1) - \Pr(A^{g^*} = 1)\big|.$$

A is making at most q distinct queries to an oracle function instantiated by g_K, resp. g^*. In order to upper bound $\mathbf{Adv}^{prf}_G(A)$, we consider the intermediate situation where the oracle function is neither g_k nor g^*, but a random instance $g \circ f^*$ of the composition of a perfect random function $f^* \in F^*_{n,m}$ and g. Due to the triangular inequality, we have

$$\begin{aligned}\mathbf{Adv}^{prf}_G(A) \leq &\big|\Pr(A^{g_K} = 1) - \Pr(A^{g\circ f^*} = 1)\big| \\ &+ \big|\Pr(A^{g\circ f^*} = 1) - \Pr(A^{g^*} = 1)\big|.\end{aligned}$$

We denote the first and the second absolute values of the right expression by δ_1 and δ_2.

Let us first upper bound δ_2. It is easy to see that instantiating the oracle function of A with $g \circ f^*$ (resp. g^*) amounts to answering the up to q distinct oracle queries of A with a q-tuple $(g(y_1), \ldots, g(y_q))$ of L-bit values, where the q-tuple $(y_1, \ldots, y_q)$ is a randomly drawn from $\{0,1\}^{mq}$, resp. with a q-tuple $(z_1, \ldots, z_q)$ of answers randomly drawn from $\{0,1\}^{Lq}$. In both case, the q-tuple of oracle answers is independent of the up to q distinct values of the oracle queries. Using this fact we can derive an algorithm B that distinguishes q values $(g(x_1), \ldots, g(x_q))$ from q random values of $\{0,1\}^L$. Algorithm B works as follows: on input $(y_1, \ldots, y_q)$ it runs algorithm A. Consequently it has to answer A's n-bit oracle queries x_i with L-bit responses. On each distinct query x_i, B simply answers y_i. When A halts, B halts also and returns the output of A. We can easily see that $\Pr(B(g(x_1), \ldots g(x_q)) = 1) = \Pr(A^{g\circ f^*} = 1)$ and that $\Pr(B(y_1, \ldots y_q) = 1) = \Pr(A^{g^*} = 1)$. Consequently we have

$$\delta_2 = \mathbf{Adv}^{prng}_g(B) \leq \mathbf{Adv}^{prng}_g(t, q).$$

Lemma 1 now provides:

$$\delta_2 \leq q\mathbf{Adv}^{prng}_g(t + qT^L_g).$$

In order to upper-bound $\mathbf{Adv}^{prf}_G(A)$ we still have to upper bound δ_1. This can be done by deriving from algorithm A an algorithm C that is able by invoking A one single time to distinguish a random instance of the n-bit to m-bit PRF F from a perfect random function $f^* \in F^*_{n,m}$ with an advantage also equal to δ_1.

Algorithm C has access to an n-bit to m-bit oracle function f. C works as follows: first it invokes algorithm A. Consequently it has to answer A's n-bit oracle queries x_i with L-bit responses. For such a query, C queries its own oracle function f with the same query value x_i, gets an m-bit answer $y_i = f(x_i)$, computes the L-bit value $g(y_i)$, and answers this value to algorithm A. When A halts, C halts as well and outputs the same output as A. Therefore

we have $\Pr(C^{f_K} = 1) = \Pr(A^{g \circ f_K} = 1) = \Pr(A^{g_K} = 1)$ and $\Pr(C^{f*} = 1) = \Pr(A^{g \circ f*} = 1)$. This implies that $\left|\Pr(C^{f_K} = 1) - \Pr(C^{f*} = 1)\right|$ is equal to $\left|\Pr(A^{g_K} = 1) - \Pr(A^{g \circ f^*} = 1)\right|$, i.e.

$$\mathbf{Adv}_F^{prf}(C) = \delta_1.$$

Furthermore the time required for algorithm C is equal to the time required for algorithm A plus q times the time of computing g. Therefore $\mathbf{Adv}_F^{prf}(C)$ is upper-bounded by $\mathbf{Adv}_F^{prf}(t + qT_g^L, q)$, i.e. $\delta_1 \leq \mathbf{Adv}_F^{prf}(t + qT_g^L, q)$. Finally we have for any A

$$\mathbf{Adv}_G^{prf}(A) \leq \delta_1 + \delta_2 \leq \mathbf{Adv}_F^{prf}(t + qT_g^L, q) + q\mathbf{Adv}_g^{prng}(t + qT_g^L).$$

Consequently

$$\mathbf{Adv}_G^{prf}(t, q) \leq \mathbf{Adv}_F^{prf}(t + qT_g^L, q) + q\mathbf{Adv}_g^{prng}(t + qT_g^L).\square$$

Proof of Theorem 2

Theorem 2. Let $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ be a PRNG which generates $L \geq 2m$ outputs bits and produces its $2m$ first output bits in time T_g^{2m} and let $F^g = \{f_y\}_{y \in \{0,1\}^m}$ be the family of n-bit to m-bit functions derived from g by the Tree Based Construction. The (t, q) advantage of PRF F^g is related to the single-query advantage of PRNG g by the following inequality:

$$\mathbf{Adv}_{F^g}^{prf}(t, q) \leq nq\mathbf{Adv}_g^{prng}(t + q(n+1)T_g^{2m}).$$

Proof. First we define, for $0 \leq i \leq n$, a family F_i^g of $\{0,1\}^n \longrightarrow \{0,1\}^m$ functions; each F_i^g can be viewed as an intermediate PRF between F^g and the set $F_{n,m}^*$ of perfect random n-bits to m-bit functions.

- $F_0^g = \{f_{y_0}^g\}_{y_0 \in \{0,1\}^n}$ where $f_{y_0}^g : (x_1, \ldots, x_n) \longmapsto g_{x_n} \circ \ldots \circ g_{x_1}(y_0)$.

- $F_1^g = \{f_{y_0,y_1}^g\}_{(y_0,y_1) \in \{0,1\}^{2n}}$
  where $f_{y_0,y_1}^g : (x_1, \ldots, x_n) \longmapsto g_{x_n} \circ \ldots \circ g_{x_2}(y_{x_1})$.

- $F_i^g = \{f_{y_0,y_1,\ldots,y_{2^i-1}}^g\}_{y_0,y_1,\ldots,y_{2^i-1} \in \{0,1\}^{2^i n}}$
  where $f_{y_0,\ldots,y_{2^i-1}}^g : (x_1, \ldots, x_n) \longmapsto g_{x_n} \circ \ldots \circ g_{x_{i+1}}(y_{x_1 \ldots x_i})$
  (in the former expression $y_{x_1,\ldots x_i}$ represents $y_{\sum_{t=1}^i x_i 2^{i-1}}$)

- $F_n^g = \{f_{y_0,y_1,\ldots,y_{2^n-1}}^g\}_{y_0,y_1,\ldots,y_{2^n-1} \in \{0,1\}^{2^n}}$
  where $f_{y_0,\ldots,y_{2^n-1}}^g : (x_1, \ldots, x_n) \longmapsto (y_{x_1 \ldots x_n})$.

It is easy to see that F_0^g is equal to F^g, and that F_n^g is the set $F_{n,m}^*$ of all n-bit to m-bit functions.

Let us consider any (t, q) distinguishing algorithm A for F^g, i.e. a testing algorithm capable to query an n-bit to m-bit oracle function up to q times, and let us denote its distinguishing probability by

$$\epsilon = \left|\Pr_{f \in F^g}(A^f = 1) - \Pr_{f \in F^*_{n,m}}(A^f = 1)\right|.$$

We denote $\Pr_{f \in F^g_i}(A^f = 1)$ by p_i. Thus we have

$$\epsilon = |p_0 - p_n|.$$

We now construct a q-query distinguisher B for g, which when input with a q-tuple $(z_1, \ldots, z_q)$ of 2m-bit words is using one invocation of algorithm A to output either 0 or 1. In order to processes an input q-tuple $(z_1, \ldots, z_q)$, B first randomly draws an integer i comprised between 0 and $n-1$, and then inputs (z_1, z_q) to a testing algorithm B_i, and outputs B_i's binary output. Each testing algorithm B_i is defined as follows: B_i invokes algorithm A, and computes the answers to the up to q distinct n-bit oracle queries of A. For that purpose, B_i uses its random generation capability to simulate an auxiliary random function $\alpha : \{0,1\}^i \longrightarrow \{1, q\}$ that is initially undetermined. At each novel n bit oracle query $x^j = (x^j_1, \ldots, x^j_n)$ of A, algorithm B_i uses:

- the bits x^j_1 to x^j_i to determine a $2m$-bit value z_k as follows: B_i first checks if α is defined on point $(x^j_1, \ldots, x^j_i)$. If not, it selects randomly a value in $\{1, q\}$, affects it to $\alpha(x^j_1, \ldots x^j_i)$ and stores the new point of α. Otherwise B_i simply read the previously stored value. In both case, we denote by k the obtained value of $\alpha(x^j_1, \ldots x^j_i)$; k is used to select the k-th input z_k from B_i's input $(z_1, \ldots, z_q)$;
- the bit x^j_{i+1} to select an m-bit word y equal to the substring of the m left bits of z_k if $x^j_{i+1} = 0$, and of the m right bits of z_k if $x^j_{i+1} = 1$;
- the bits x^j_{i+2} to x^j_n to compute A^s L-bit oracle response $g_{x^j_n} \circ \ldots \circ g_{x^j_{i+2}}(y)$.

Finally when A halts, B_i halts also and returns A's binary output.
It is not too difficult to see that:

- if B_i's input is $(g(a_1), \ldots, g(a_q))$, where $(a_1, \ldots, a_q)$ is a randomly drawn q-tuple of m-bit, then A's oracle queries and response pairs have exactly the same probability distribution as if A were run with an n-bit to m-bit oracle function f randomly drawn from the family F^g_i:

$$\Pr(B_i((g(a_1), \ldots, g(a_q)) = 1) = \Pr_{f \in F^g_i}(A^f = 1) = p_i.$$

- if B_i's input is is a randomly drawn q-tuple $(z_1, \ldots, z_q)$ of $2m$-bit values, then A's oracle queries and response pairs have exactly the same probability distribution as if A was run with an n-bit to m-bit oracle function f randomly drawn from the family F^g_{i+1}:

$$\Pr(B_i(z_1, \ldots, z_q) = 1) = \Pr_{f \in F^g_{i+1}} = (A^f = 1) = p_{i+1}.$$

The above equalities imply:

$$\begin{aligned}&\big|\Pr(B((g(a_1),\ldots,g(a_q))=1)-\Pr(B(z_1,\ldots,z_q)=1)\big|\\&=\big|\frac{1}{n}\sum_{i=0}^{n-1}p_i-\frac{1}{n}\sum_{i=1}^{n}p_i\big|=\frac{1}{n}|p_0-p_n|=\frac{\epsilon}{n}.\end{aligned}$$

In other words:

$$\mathbf{Adv}_g^{prng}(B)=\frac{1}{n}\mathbf{Adv}_{F^g}^{prf}(A).$$

However, algorithm B requires at most $t+qnT_g^{2m}$, where t is the time required by A and $T_{g'}^{2m}$ is the time required by g' to produce $2m$ bits. Therefore for any A we have

$$\mathbf{Adv}_{F^g}^{prf}(A)\leq n\mathbf{Adv}_g^{prng}(t+qnT_g^{2m},q).$$

Finally, since $\mathbf{Adv}_g^{prng}(t+qnT_g^{2m},q)\leq q\mathbf{Adv}_g^{prng}(t+q(n+1)T_g^{2m})$ due to Lemma 1, we obtain

$$\mathbf{Adv}_{F^g}^{prf}(t,q)\leq qn\mathbf{Adv}_g^{prng}(t+q(n+1)T_g^{2m}).\square$$

Main Reduction Theorem of [3]

Theorem 4. *Let $L=\lambda(k-1)n$ be the number of keystream bits produced by in time λT_S using λ iterations of our construction. Suppose there exists an algorithm A that distinguishes the L-bit keystream sequence associated with a known randomly chosen system S and an unknown randomly chosen initial internal state $x\in\{0,1\}^n$ from a random L-bit sequence in time T with advantage ϵ. Then there exists an algorithm C, which given the image $S(x)$ of a randomly chosen (unknown) n-bit value x by a randomly chosen n-bit to m-bit quadratic system S produces a preimage of $S(x)$ with probability at least $\frac{\epsilon}{2^3\lambda}$ over all possible values of x and S in time upper bounded by T'.*

$$T'=\frac{2^7n^2\lambda^2}{\epsilon^2}\left(T+(\lambda+2)T_S+\log\left(\frac{2^7n\lambda^2}{\epsilon^2}\right)+2\right)+\frac{2^7n\lambda^2}{\epsilon^2}T_S$$

QUAD: Overview and Recent Developments

David Arditti[1], Côme Berbain[1], Olivier Billet[1],
Henri Gilbert[1], and Jacques Patarin[2]

[1] France Telecom Research and Development,
38-40 rue du Général Leclerc, F-92794 Issy-les-Moulineaux, France.
firstname.lastname@orange-ftgroup.com
[2] Université de Versailles,
45 avenue des Etats-Unis, F-78035 Versailles cedex, France.
jacques.patarin@prism.uvsq.fr

Abstract. We give an outline of the specification and provable security features of the QUAD stream cipher proposed at Eurocrypt 2006 [5]. The cipher relies on the iteration of a multivariate system of quadratic equations over a finite field, typically GF(2) or a small extension. In the binary case, the security of the keystream generation can be related, in the concrete security model, to the conjectured intractability of the MQ problem of solving a random system of m equations in n unknowns. We show that this security reduction can be extended to incorporate the key and IV setup and provide a security argument related to the whole stream cipher. We also briefly address software and hardware performance issues and show that if one is willing to pseudorandomly generate the systems of quadratic polynomials underlying the cipher, this leads to suprisingly inexpensive hardware implementations of QUAD.

Key words: MQ problem, stream cipher, provable security, Gröbner basis computation

1 Introduction

Symmetric ciphers can be broadly classified into two main families of encryption algorithms: block ciphers and stream ciphers. Unlike block ciphers, stream ciphers do not produce a key-dependent permutation over a large block space, but a key-dependent sequence of numbers over a small alphabet, typically the binary alphabet $\{0, 1\}$. To encrypt a plaintext sequence, each plaintext symbol is combined with the corresponding symbol of the keystream sequence by using a group operation, usually the exclusive or operation over $\{0, 1\}$. Nearly all stream ciphers specified recently use two inputs to generate a keystream sequence: a secret key and an additional parameter named initial value (IV) that is generally not secret. The use of IVs allows to derive several independent keystream sequences from one single key by resynchronizing the stream cipher each time with a new IV.

The current status of stream ciphers design is characterized by a considerable discrepancy between theory and practice.

On the theoretical side, seminal work by Shamir[31], Blum and Micali [7], Yao [33], Levin and Goldreich [21] in the early 80's produced the well founded theory of pseudo-random generators, which represents one of the major achievements in the area of provable security. A pseudo-random number generator (PRNG) can be viewed as an IV-less stream cipher. It expands a short seed, e.g. a key, into a larger bit string in such a way that if the input seed is secret and randomly drawn, then the resulting output is computationally indistinguishable from a perfect random sequence. The research effort on security proofs for PRNGs has not only led to remarkable generic results, e.g. the proof by Impagliazzo, Levin, Luby and Håstad [24] that a secure PRNG can be constructed based upon any one way function (OWF). It has also led to "provably secure" PRNG constructions exploiting the conjectured one-wayness of specific permutation or function f, which generally rely on the iteration of f and the extraction of a few bits at each iteration. The first construction of this type was introduced by Blum and Micali [7]. Its security reduction relates the security of the PRNG to the one-wayness of exponentiation modulo a prime number. The construction proposed by L. Blum, M. Blum and M. Shub [6] exploits the conjectured intractability of quadratic residuosity modulo Blum integers. Alexi, Chor, Goldreich and Schnorr proposed a construction with security that relies upon the RSA assumption. Impagliazzo and Naor [27] proposed a construction relying on the difficulty of the subset sum problem. More recently, some efficiency improvements were recently achieved, either by decreasing the state length as in Fisher and Stern's construction [16] based on the intractability of the syndrome decoding problem, or by increasing the number of bits extracted at each iteration of f as illustrated in Gennaro's construction based on the intractability of the discrete logarithm problem and Boneh, Halevi, Howgrave-Graham's construction [8] and in Steinfeld, Pieprzyk and Wang's construction [32] respectively based on the conjectured pseudo-randomness and one-wayness of RSA with small inputs. However, current provably secure PRNGs are still generally regarded as too complex and inefficient to provide really practical stream ciphers. The lack, for these various algorithms, of an extra IV parameter, represents an additional drawback.

On the practical side, extremely efficient stream ciphers have been proposed, which either allow like SCREAM [23], RC4 [30], SNOW 2.0 [14]and its UMTS variant SNOW 3G) much faster software encryption than existing block ciphers such as AES or require much lower computing resources for hardware implementations or both, like the GRAIN [26] and TRIVIUM [9] candidates to the ongoing European initiative eSTREAM [13]. However, the design of secure stream ciphers is not currently as well understood as the design of secure block ciphers. The state of the art of the cryptanalysis of stream ciphers has evolved significantly over the last ten years with the development of attack techniques such as algebraic attacks, fast correlation and linear masking attacks, resynchronization attacks against IV-dependent stream ciphers, and it turns out that

many recent proposals still suffer from security weaknesses. This is illustrated by the fact that more than one third of the 34 candidate algorithms submitted to the eSTREAM stream ciphers evaluation project have already been shown to be insecure.

The main design objective of QUAD was to contribute to reducing the discrepancy between practical stream ciphers and provably secure PRNG constructions depicted above by specifying a practical stream cipher with provable security arguments. Instead of relying upon the conjectured intractability of number theoretical problems -e.g. the the factoring and discrete logarithm problems- like most provably secure PRNG constructions proposed so far, QUAD relies upon the conjectured intractability of the MQ problem of solving a multivariate system of m quadratic equations in n unknowns over a finite field $\mathrm{GF}(q)$, e.g. $\mathrm{GF}(2)$, which is known to be NP-hard and conjectured to be intractable in terms of average complexity even for extremely compact instances provided that the $\frac{m}{n}$ ratio is sufficiently close to 1. Thus QUAD belongs to the promising and fast expanding family of multivariate cryptographic algorithms. Moreover, unlike asymmetric multivariate algorithms relying upon the intractability of the MQ problem proposed so far, e.g. HFE or UOV, QUAD can be based on a random instance of MQ without any embedded trapdoor since QUAD is symmetric cipher and the computations performed at the sending and receiving side do not require any inversion of a MQ instance. In other words, QUAD's security relies more directly upon the intractability of the MQ problem than the one of asymmetric multivariate algorithms.

This paper is organized as follows. We first summarize the status of the MQ problem (Section 2) and recall basic security definitions in a concrete (non asymptotic) security model (Section 3). We then describe the QUAD algorithm (Section 4). We show that in the $\mathrm{GF}(2)$ case, the security of the QUAD's keystream generator can be provably related to the conjectured intractability of the MQ problem (Section 5). We show how to extend this proof, also in the $\mathrm{GF}(2)$ case, as to incorporate the key and IV setup to get a security reduction for the whole cipher (Section 6). Finally, we address software and hardware implementation issues (Section 7).

2 Multivariate quadratic systems

We consider a finite field $\mathrm{GF}(q)$. A multivariate quadratic equation (or equivalently a multivariate quadratic polynomial) in n variables over $\mathrm{GF}(q)$ is a polynomial of degree at most 2 in $\mathrm{GF}(q)[x_1, \ldots, x_n]$ which can be written as

$$Q(x) = \sum_{1 \leq i \leq j \leq n} \alpha_{i,j} x_i x_j + \sum_{1 \leq i \leq n} \beta_i x_i + \gamma,$$

with coefficients $\alpha_{i,j}$, β_i, and γ in $\mathrm{GF}(q)$. In the particular case $q = 2$, which is the one most often considered in the sequel, monomials $x_i x_i$ and x_i are equal.

It is easy to see that the set $\mathcal{Q}$ of multivariate quadratic polynomials in n variables is an N-dimensional vector space over $\mathrm{GF}(q)$, where $N = \frac{1}{2}n(n+3)+1$ if $q \neq 2$ and $N = \frac{1}{2}n(n+1)+1$ if $q = 2$. A basis of this vector space is given by the $N-1$ distinct monomial functions of degree one or two, and the non-null constant polynomial. Any element of $\mathcal{Q}$ can be represented by the N-tuple of its $\mathrm{GF}(q)$ coefficients in this basis. Throughout the rest of this paper, by a randomly chosen quadratic polynomial in n unknowns we mean the quadratic polynomial represented in the above basis by a uniformly and independently drawn N-tuple of $\mathrm{GF}(q)$ coefficients.

A multivariate quadratic system S of m quadratic equations in n variables over $\mathrm{GF}(q)$ consist of a set $(Q_1, \ldots, Q_m)$ of m quadratic polynomials in n variables over $\mathrm{GF}(q)$. In the sequel, by a randomly chosen system of m quadratic poynomials in n unknowns, we mean m independently and randomly chosen quadratic polynomials. Such a system is represented by mN coefficients uniformly and independently drawn from $\mathrm{GF}(q)$.

We define the problem of solving multivariate quadratic systems (MQ problem) as follows: given a multivariate quadratic system $S = (Q_1, \ldots, Q_m)$, of m quadratic equations over $\mathrm{GF}(q)$, find a value $x \in \mathrm{GF}(q)^n$, if any, such that $Q_i(x) = 0$ for all $1 \leq i \leq m$.

Depending on the respective values of n and m, instances of MQ can be either easy or very difficult to solve. For $m = 1$ the number of solutions is known [28] and it is quite easy to find one solution. When m is significantly smaller than n, that is for an underdefined quadratic system, finding a solution is much easier than the exhaustive search on the number of variables [10]. In the opposite situation of an overdefined system ($m > n$) containing nearly N linearly independent quadratic equations solving an MQ problem is easy by linearization. The total complexity is then only $O(n^6)$. However, for general values of m and n the MQ problem is known to be NP-hard, even when restricted to quadratic equations over $\mathrm{GF}(2)$ (see [18, 17]) or over any finite field (see [29]).

Moreover, what makes the MQ problem particularly well suited for cryptographic applications is that it is conjectured to be very difficult not only asymptotically and in worst case, but already for small suitably selected values of m and n and in terms of the average complexity of solving a random instance. The problem seems to be the most difficult when m is close to n. For $m = n$ and $q = 2$ the complexity of the best known solving algorithms is $2^{n-O(\sqrt{(}n))}$ and thus rather close to the 2^n complexity of exhaustive search, and totally out of reach of existing computers when n is larger than 100. Even when $q = 2$ and $m = kn$, where $k > 1$ is small enough compared with $\frac{n}{2}$, the best known algorithms XL [11] and improved variants of Buchbergers's Gröbner basis computation algorithm such as Faugère's F4 and F5 algorithms [15] are exponential in n for a randomly chosen quadratic system. Much research has been dedicated in the past years to the above problem [12, 11]. Bardet's Ph.D. thesis [2] provides an accurate analysis of the complexity of the most efficient algorithm computing Gröbner basis known to solve a random system of $m = kn$ equations in n unknowns.

3 The stream cipher QUAD

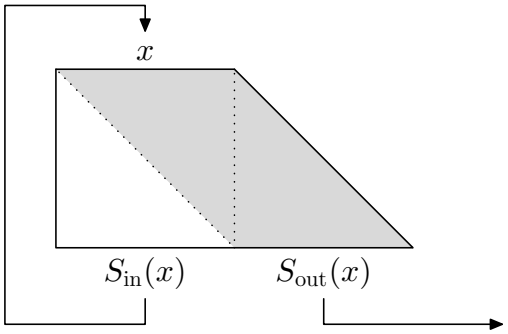
This section describes the stream cipher QUAD, which specification was first published in [5]. $S = (Q_1, \ldots, Q_{kn})$ denotes a multivariate quadratic system of kn randomly chosen equations in n variables over $GF(q)$, and S_0 and S_1 denote two (k times smaller) additional multivariate systems of n randomly chosen equations in n variables over $GF(q)$. S, S_0 and S_1 are fixed and publicly known. During the key and IV loading and the keystream generation, the internal register state is an n-tuple $x = (x_1, \ldots, x_n)$of $GF(q)$ values.

3.1 Keystream generation and encryption

The keystream generation process simply consists in iterating the three following steps in order to produce $(k-1)n$ $GF(q)$ keystream values at each iteration.

- **Step 1:** compute the kn-tuple of $GF(q)$ values $S(x) = (Q_1(x), \ldots, Q_{kn}(x))$ where x is the current value of the internal state;
- **Step 2:** output the keystream sequence $S_{out}(x) = (Q_{n+1}(x), \ldots, Q_{kn}(x))$ of $(k-1)n$ $GF(q)$ values
- **Step 3:** update the internal state x with the sequence of the n first generated $GF(q)$ values $S_{in}(x) = (Q_1(x), \ldots, Q_n(x))$

The maximal length of the keystream sequence that can be generated with a single (key,IV) pair is set to L $GF(q)$ symbols. In order to encrypt a plaintext of length $l \leq L$ $GF(q)$ symbols, each of the first l $GF(q)$ values of the keystream sequence is added (using the $GF(q)$ addition) with the corresponding plaintext value.



3.2 Key and IV setup

Before generating any keystream we need to initialize the internal state x, with the key K and the initialization vector IV, which are respectively represented by a sequence of $GF(q)$ elements of length $|K|$ and a binary sequence of $\{0, 1\}$ values of length $|IV|$. We assume for the time being, for simplicity of the subsequent proofs that $|K|$ is chosen exactly equal to n.

The initialization is done as follows: we first set the internal state value x to the $GF(q)^n$ value K. Then for each of the $|IV|$ bits IV_1 to $IV_{|IV|}$ of the IV

value, the internal state x is updated as follows: if $IV_i = 0$, x is replaced by the $GF(q)^n$ value $S_0(x)$; otherwise, x is replaced by the $GF(q)^n$ value $S_1(x)$. These $|IV|$ steps provide a key and IV dependent internal state value x. We then clock the cipher n additional times as described in section 3.1, but without outputting the keystream. After this preliminary runup phase, the keystream is generated as described in section 3.1.

4 Basic security notions

We first recall definitions of advantages for distinguishing a number generator from a perfect random generator and a function generator from a perfect random function generator, and the notions of Pseudo Random Number Generator (PRNG) and Pseudo Random Function (PRF). All the security definitions used throughout this paper relate to the concrete (non asymptotic) security model. In the sequel, when we state that a value u is randomly chosen in a set U, we implicitly mean that u is drawn according to the uniform law over U.

Single-query distinguisher for a number generator: let us consider a number generator $g : \{0,1\}^n \longrightarrow \{0,1\}^L$ with input and output lengths $L > n$, used to expand an n-bit secret random seed into an L-bit sequence. A distinguisher in time t for g is a probabilistic testing algorithm A which when input with an L-bit string outputs either 0 or 1 with time complexity at most t. We define the advantage of A for distinguishing g from a perfect random generator as

$$\mathbf{Adv}_g^{prng}(A) = \left|\Pr_{x\in\{0,1\}^n}(A(g(x)) = 1) - \Pr_{y\in\{0,1\}^L}(A(y) = 1)\right|,$$

where the probabilities are not only taken over the value of an unknown randomly chosen $x \in \{0,1\}^n$ (resp. of a randomly chosen $y \in \{0,1\}^L$), as explicitly stated in the above formula, but also over the random choices of the probabilistic algorithm A.

We define the advantage for distinguishing the function g in time t as

$$\mathbf{Adv}_g^{prng}(t) = \max_A\{\mathbf{Adv}_g^{prng}(A)\},$$

where the maximum is taken over all testing algorithms of time complexity at most t.

Pseudo Random Generator (PRNG): a function g is said to be a PRNG if $\mathbf{Adv}_g^{prng}(t)$ is negligible (for example less than 2^{-40}) for values of t strictly lower than a fixed threshold (for example 2^{80} or 2^{128}). The definition of a PRNG is therefore dependent upon thresholds reflecting the current perception of an acceptably secure number generator.

Distinguisher for a function generator: let us now consider a function generator, i.e. a family $F = \{f_K\}$ of $\{0,1\}^n \longrightarrow \{0,1\}^m$ functions indexed by a key K randomly chosen from $\{0,1\}^k$. A distinguisher in time t with q queries for F is a probabilistic testing algorithm A^f capable to query an n-bit to m-bit oracle function f up to q times. Such an algorithm allows to distinguish a randomly chosen function f_K of F from a perfect random function f^* randomly chosen in the set $F^*_{n,m}$ of all $\{0,1\}^n \longrightarrow \{0,1\}^m$ functions with a distinguishing advantage

$$\mathbf{Adv}_F^{prf}(A) = \left|\Pr(A^{f_K} = 1) - \Pr(A^{f^*} = 1)\right|,$$

where the probabilities are taken over $K \in \{0,1\}^k$ (resp $f^* \in F^*_{n,m}$) and over the random choices of A. We define the advantage for distinguishing the family F in time t with q queries as

$$\mathbf{Adv}_F^{prf}(t,q) = \max_A\{\mathbf{Adv}_F^{prf}(A)\},$$

where the maximum is taken over all testing algorithms A working in time at most t and capable to query an n-bit to m-bit oracle function up to q times.

Pseudo Random Function (PRF): a family of functions $F = \{f_K\}$ is said to be a PRF if $\mathbf{Adv}_F^{prf}(t,q)$ is negligible for values of t and q strictly lower than the respective threshold (for example 2^{80} or 2^{128} for t and 2^{40} for q).

5 Security of the keystream generation

We now give an outline of the security reduction relating, in the GF(2) case, the PRNG-indistiguishability of the keystream generation part of QUAD to the conjectured intractability of the MQ problem. This security reduction is expressed by Theorem 4 hereafter. The details of the proof of Theorem 4 are given in [5]. In this paper, we only describe the structure of this proof, which is divided in three parts.

5.1 Part 1: distinguishing the keystream allows to distinguish the output of a random quadratic function

In the first part (Theorem 1), we prove that if the L-bit keystream sequence associated with a known fixed or randomly chosen system S of $m = kn$ quadratic equations and an unknown randomly chosen initial internal state $x \in \{0,1\}^n$ is distinguishable from the L-bit output of a perfectly uniform generator, then for a known random quadratic system S of $m = kn$ equations and an unknown randomly chosen input value $x \in \{0,1\}^n$, $S(x)$ is distinguishable from a random kn bit word. Though we consider a randomly chosen system S because we need distinguishing properties related to a random system for the sequel, the property we prove would also hold if we considered a fixed system S. Our proof is inspired by the proof given in [22] that a similar result holds for the generator based on iteration of any fixed n-bit to m-bit function, where $m > n$, but provides a tighter bound for the advantage than [22].

Theorem 1. *Let $L = \lambda(k-1)n$ be the number of keystream bits produced in time λT_S using λ iterations of our construction. Suppose there is an algorithm A that distinguishes the L-bit keystream sequence associated with a known randomly chosen system S and an unknown randomly chosen initial internal state $x \in \{0,1\}^n$ from a random L-bit sequence in time T with advantage ϵ. Then there exists an algorithm B that for a randomly chosen S distinguishes $S(x)$ corresponding to an unknown random input x, from a random value of size kn in time $T' = T + \lambda T_S$ with advantage $\frac{\epsilon}{\lambda}$.*

5.2 Part 2: distinguishing the output of a random quadratic function allows to predict any linear function of its input

In the second part (Theorem 2), we prove that if for a known randomly chosen quadratic system S and an unknown randomly chosen x, there exists a distinguisher allowing to distinguish $S(x)$ from a random kn bit word such as the one considered in Theorem 1 above, then it can be converted into an algorithm allowing, for any n-bit to 1-bit quadratic function R, in particular any linear form R, to predict $R(x)$ for a randomly chosen n bit value x better than at random given $S(x)$.

Theorem 2. *Suppose there is an algorithm A that, given a randomly chosen known multivariate quadratic system S of kn equations in n unknowns, distinguishes $S(x)$, where x is an unknown random input value, from a random string of length kn with advantage at least ϵ and in time T. Then there is an algorithm B that, given a randomly chosen quadratic system S of kn equations in n unknowns, any n-bit to 1-bit quadratic form R, and $y = S(x)$ where x is a random input value, predicts $R(x)$ with success probability at least $\frac{1}{2} + \frac{\epsilon}{4}$ using at most $T' = T + 2T_S$ operations.*

5.3 Part 3: predicting any linear function of the input of a quadratic function allows to invert it

In the third part (Theorem 3), we show that if for a fixed or random quadratic system S and more generally any fixed or random n-bit to m-bit function f there exists a predictor such as the one considered in the former theorem, i.e. a predictor allowing, given an n-bit to 1-bit linear form R, to predict $R(x)$ with a success probability (over all S and x values) strictly larger than $\frac{1}{2}$, then a preimage of $S(x)$ (resp. f(x)) can be efficiently computed, so that S (resp f) is not one way. This part is essentially a proof of Goldreich-Levin's theorem ([21]), in which a fast Walsh transform computation is used to get a tighter reduction. In order to proof Theorem 3, which relates to the computation, given the image $S(x)$ or $f(x)$ for a random unknown value x and a random system S, of a list containing x, we first establish a lemma representing the technical core of the proof, in which a fixed (unknown) value of x is considered. Our proofs are inspired by the simplified treatment of the original Goldreich-Levin proofs developed by Rackoff, Goldreich in [19] and Bellare in [3], and also by the proofs provided by Håstad and Näslund in [25].

Lemma 1. *Let us denote by x a fixed unknown n-bit value and denote by f a fixed n-bit to m-bit function. Suppose there exists an algorithm B that given the value of $f(x)$ allows to predict the value of any linear equation R over n unknowns with probability $\frac{1}{2}+\epsilon$ over R, using at most T operations. Then there exists an algorithm C, which given $f(x)$ produces in time at most T' a list of at most $4n^2\epsilon^{-2}$ values such that the probability that x appears in this list is at least $1/2$.*

$$T' = \frac{2n^2}{\epsilon^2}\left(T + \log\left(\frac{2n}{\epsilon^2}\right) + 2\right) + \frac{2n}{\epsilon^2}T_f$$

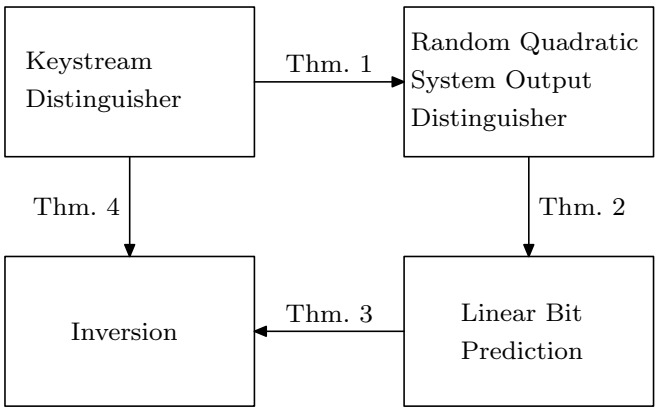
This Lemma applies to a fixed x and a fixed system S (or a fixed n-bit to m-bit function f). However, the success probability of the predictor of Theorem 2 is taken over all (x, S) pairs for any linear form R. Consequently, we need a theorem allowing us to exploit the existence of such a predictor to show the applicability of the lemma to a non-negligible fraction of (x, S) pairs.

Theorem 3. *Suppose there exists an algorithm B, that given a randomly chosen quadratic system S of m quadratic equations, a randomly chosen n-bit to 1-bit quadratic form R and the image $S(x)$ of a randomly chosen (unknown) n-bit value x, predicts the value of $R(x)$ with probability at least $\frac{1}{2}+\epsilon$ over all possible (x, S, R) triplets using T operations. Then there is an algorithm C, which given the image $S(x)$ of a randomly chosen (unknown) n-bit value x produces a preimage of $S(x)$ with probability at least $\epsilon/2$ (over all possible values of x and S) in time T'.*

$$T' = \frac{8n^2}{\epsilon^2}\left(T + \log\left(\frac{8n}{\epsilon^2}\right) + 2\right) + \frac{8n}{\epsilon^2}T_f$$

5.4 Security proof for the keystream generation

Now it is easy to see that if we sequentially apply theorems 1, 2, and 3, we obtain the following reduction theorem, which states that if, for a random system and a random initial value, the L-bit keystream sequence was distinguishable from a random L-bit sequence then there would exist an efficient algorithm allowing to find a preimage of the image of a random n-bit input value by a random quadratic n-bit to m-bit system, which for suitably chosen values of n would contradict the assumptions made in Section 2 on the difficulty of solving MQ.



Theorem 4. *Let $L = \lambda(k-1)n$ be the number of keystream bits produced by in time λT_S using λ iterations of our construction. Suppose there exists an algorithm A that distinguishes the L-bit keystream sequence associated with a known randomly chosen system S and an unknown randomly chosen initial internal state $x \in \{0,1\}^n$ from a random L-bit sequence in time T with advantage ϵ. Then there exists an algorithm C, which given the image $S(x)$ of a randomly chosen (unknown) n-bit value x by a randomly chosen n-bit to m-bit quadratic system S produces a preimage of $S(x)$ with probability at least $\frac{\epsilon}{2^3\lambda}$ over all possible values of x and S in time upper bounded by T'.*

$$T' = \frac{2^7 n^2 \lambda^2}{\epsilon^2}\left(T + (\lambda+2)T_S + \log\left(\frac{2^7 n \lambda^2}{\epsilon^2}\right) + 2\right) + \frac{2^7 n \lambda^2}{\epsilon^2} T_S$$

Theorem 4 above relates to the keystream generation part of QUAD only, not to the key and IV setup computation for deriving the initial state. Moreover it does not guarantee the strength of a particular instance of QUAD associated with a fixed system S but (informally) it shows that for suitably chosen parameter values if MQ is intractable then most instances of QUAD are secure.

5.5 Specifying Parameter Values for QUAD

We now propose concrete parameters n, k, and L for our construction. We still restrict ourselves to the $GF(2)$ case. We want to ensure a security level of at least 2^{80}. More precisely we want Theorem 4 to ensure that if for a random system and a random initial internal state value at the beginning of the keystream generation there exists a testing algorithm that allows us to distinguish an L-bit keystream produced by QUAD from a uniformly drawn keystream sequence with an advantage of more than $\epsilon = \frac{1}{100}$ in time less than $T = 2^{80}$, this would imply the existence of an inversion algorithm of non negligible success probability $\epsilon' = \frac{\epsilon}{2^3\lambda}$ allowing, given a random n-bit to kn-bit system of quadratic equations and the $S(x)$ image by S of a random input value x, to find a preimage by S of $S(x)$ in time T' lower by a substantial factor, say $\frac{1}{\epsilon'}$, than the best known inversion algorithms for the MQ problem, and thus result in the existence of a large set of weak instances of MQ.

Depending on the intended application of the pseudorandom number generator, the maximum keystream length L can vary from a few hundreds bits for a mobile phone application to up to 2^{40} bits. Consequently the allowed parameter values for n and k will also vary, since it is much more demanding to get a security argument for $L = 2^{40}$ bits than for $L = 1000$ bits. We will however retain the latter value $L = 2^{40}$ for a first estimate of the corresponding required value of n.

In her thesis, Magali Bardet [2] shows that the best Groebner basis algorithm to solve a system of kn equations in k unknowns has (in the case of a regular system) a complexity of $T(k,n) = \left(\binom{n+1}{D}\right)^{2.37}$, where D is close to $\left(-k + \frac{1}{2} + \frac{1}{2}\sqrt{2k^2 - 10k - 1 + 2(k+2)\sqrt{k(k+2)}}\right) n$. To obtain a contradiction, we need to have T' lower than $\epsilon' T(k,n)$. For $k = 2$ and with the previous

values of $L = 2^{40}$, $T = 2^{80}$ and $\epsilon = \frac{1}{100}$, we get $\epsilon' = 2^{-42}$ and we need to have n greater than 350. For $n = 256$ and $k = 2$, we only get a contradiction if we produce less than $L = 2^{22} = 4$ Mbits of keystream.

Parameter values recommended in practice: for QUAD over GF(2), we recommend in practice an internal state length of $n = 160$ bits and an expansion factor k of 2 and a maximum keystream length $L = 2^{40}$. For such n, k and L values, the former concrete security reduction is not applicable, i.e. we do not get a contradiction as for the former parameter values. However our proof reduction is not optimal, and we conjecture that these parameter values suffice to provide the desired security level of at least 2^{80}.

6 Extending the security proof to the Key and IV setup

The security proof of the former section only relates to the keystream generation part of QUAD. We now extend this proof to include the key and IV setup. Our aim is to relate the indistinguishability of the QUAD cipher, more precisely of the family of IV to keystream functions indexed by the key K associated with QUAD, to the conjectured intractability of the MQ problem. For that purpose, we view QUAD as the composition of two functions, and provide security proofs for these functions.

- A keyed **initial state derivation function**, which consists of the initial phase of the key and IV setup, i.e. the derivation of the initial state before the runup phase. This part can be viewed as a family of IV to initial state functions indexed by the key K. We will show that for suitably chosen parameter sizes, this family of functions can be expected to be a PRF.
- An unkeyed **initial state to keystream function**, which consists of the runup phase of the key and IV setup followed by the keystream generation. We will show that for suitably chosen parameter sizes, this function can be expected to be a PRNG.

A simple composition theorem (stating essentially that the composition of a PRF and a PRNG with fitting output and input lengths is a PRF) allows then to derive a security reduction for the whole cipher.

6.1 PRF-indistiguishability of the initial state derivation function of QUAD

The key observation allowing to establish that if the MQ problem is intractable then the initial state derivation function of QUAD is a secure is that this function results from applying the Tree Based Construction introduced by Goldreich, Goldwasser, and Micali in [20] to the quadratic n-bit to $2n$-bit function $S' = (S_0, S_1)$. Indeed, the IV bits determine a path leading from the key to the initial state in the binary tree induced by S' and thus plays exactly the role of the input bits in the Tree Based Construction.

The Tree Based Construction is a generic construction allowing to derive a PRF from a PRNG. It can be defined as follows. Let us consider a PRNG $g : \{0,1\}^m \longrightarrow \{0,1\}^{2m}$ and let us denote the $2m$-bit image of $y \in \{0,1\}^m$ by $g(y) = z_0 z_1 \dots z_{2m-1}$. We derive from g two m-bit to m-bit functions $g_0 : y \in \{0,1\}^m \longmapsto z_0, \dots, z_{m-1}$ and $g_1 : y \in \{0,1\}^m \longmapsto z_m, \dots, z_{2m-1}$.

The PRF F^g is the family of functions $\{f_y\}_{y \in \{0,1\}^m}$ where

$$\begin{aligned} f_y : \{0,1\}^n &\longrightarrow \{0,1\}^m \\ (x_1, x_2, \dots, x_n) &\longmapsto f_y(x_1, x_2, \dots, x_n) = g_{x_n} \circ g_{x_{n-1}} \dots \circ g_{x_1}(y) \end{aligned}$$

This construction is illustrated on Figure 4.

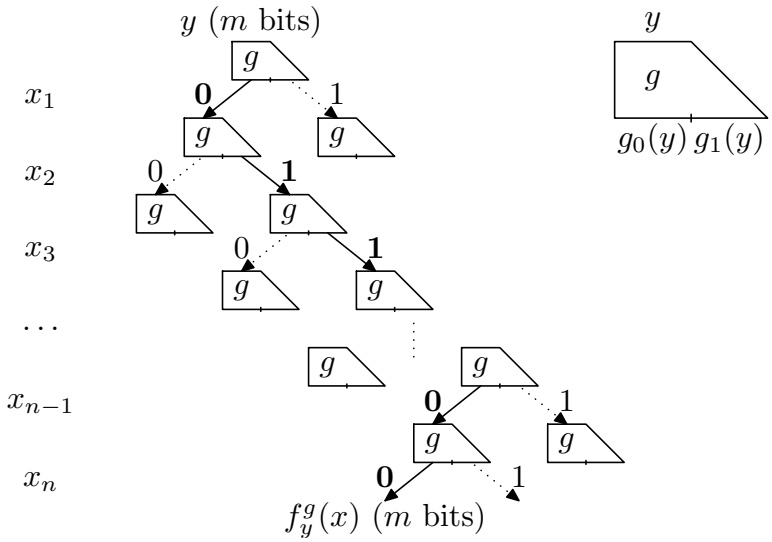


Fig. 1. Tree Based Construction

The following Theorem relates the PRF-advantage for distinguishing F^g to the PRNG-advantage for distinguishing g. The proof is essentially the same as the security proof for the Tree Based Construction given by Goldreich in [19], up to the fact that we consider the concrete security model instead of the asymptotic polynomial time indistinguishability model.

Theorem 5. *Let $g : \{0,1\}^m \longrightarrow \{0,1\}^{2m}$ be a number generator which generates $2m$ outputs bits in time T_g^{2m} and let $F^g = \{f_y\}_{y \in \{0,1\}^m}$ be the family of n-bit to m-bit functions derived from g by the Tree Based Construction. The (t,q) PRF advantage of F^g is related to the single-query PRNG advantage of g by the following inequality*

$$\mathbf{Adv}_{F^g}^{prf}(t,q) \leq nq\mathbf{Adv}_g^{prng}(t + q(n+1)T_g^{2m}).$$

The application of Theorem 5 to the initial state derivation of QUAD is straightforward. We denote by $g^{S'} : \{0,1\}^n \longrightarrow \{0,1\}^{2n}$ the n-bit to $2n$-bit

function associated with S', and by $F^{g^{S'}}$ the resulting family of initial state derivation functions. Theorem 5 allows to relate the PRF-advantage for distinguishing $F^{g^{S'}}$ to the PRNG-advantage for distinguising $g^{S'}$ by the inequality

$$\mathbf{Adv}^{prf}_{F^{g^{S'}}}(t,q) \leq 2q\mathbf{Adv}^{prng}_{g^{S'}}(t + q(|IV|+2)T^{2n}_{g^{S'}})$$

Moreover, Theorem 4 above allows to relate the PRNG-advantage for distinguish $g^{S'}$ to the hardness of inverting MQ.

6.2 PRNG-indistinguishability of the initial state to keystream function

Let us denote by $g^S : \{0,1\}^n \longrightarrow \{0,1\}^L$ the keystream generation function induced by the iteration of the quadratic function S. The security of the number generator g_{real} which starts by running n clocks like g^S without producing any keystream to reflect the runup of QUAD and then produces L bits of keystream in the same way as g^S is related to the security of $\tilde{g}^S : \{0,1\}^n \longrightarrow \{0,1\}^{L+(k-1)n^2}$ which iterates S to produce $L + (k-1)n^2$ bits, since g_{real} produces the same keystream as $\tilde{g}^S$ up to the fact that the first $(k-1)n^2$ bits of $g^{S'}$ are discarded. Consequently a distinguisher for g_{real} is also a distinguisher for $\tilde{g}^S$. Thus the advantage of g_{real} is upper-bounded by the advantage of $\tilde{g}^S$.

$$\mathbf{Adv}^{prng}_{g_{real}}(t) \leq \mathbf{Adv}^{prng}_{\tilde{g}^S}(t + T^{L+(k-1)n^2}_{\tilde{g}^S})$$

6.3 PRF-indistinguishability of the whole cipher

Now a simlpe composition theorem (Theorem 6 hereafter) allows to derive from the two former results a security reduction related to the whole cipher (Theorem 7 hereafter). We define the composition G of a family of function F and a function g, and relate the PRF-indistinguishability of G to the PRF-indistinguishability of F and the PRNG-indistinguishability of g.

Definition 1. *The composition $G = g \circ F$ of an n-bit to m-bit family of functions $F = \{f_K\}$ and of an m-bit to L-bit function g is the n-bit to L-bit family of functions*

$$G = \{g \circ f_K\}.$$

Theorem 6. *Let us consider $F = \{f_K\}$ where $f_K : \{0,1\}^n \longrightarrow \{0,1\}^m$ a functions family and $g : \{0,1\}^m \longrightarrow \{0,1\}^L$ a number generator that produces L bits in time T^L_g. The (t,q) advantage of $G = g \circ F = \{g \circ f_K\}$ can be upper bounded as follows*

$$\mathbf{Adv}^{prf}_G(t,q) \leq \mathbf{Adv}^{prf}_F(t + qT^L_g) + q\mathbf{Adv}^{prng}_g(t + qT^L_g).$$

The stream cipher QUAD results from the composition of the function family $F^{g^{S'}}$ and the number generator g_{real}. Due to the composition Theorem 6, the

security of QUAD can be related to the security of $F^{g^{S'}}$ and g_{real} which can in turn, as established in the former sections, be related to the security $g^{S'}$ and $\tilde{g}^S$. We get the inequality:

$$\mathbf{Adv}^{prf}_{\mathrm{QUAD}}(t,q) \leq 2q\mathbf{Adv}^{prng}_{g^{S'}}(t+q(|IV|+3)T^L_{g^{S'}})+q\mathbf{Adv}^{prng}_{\tilde{g}^S}(t+qT^L_{\tilde{g}^S}+T^{L+(k-1)n^2}_{\tilde{g}^S})$$

The initial state derivation and initial state to keystream functions of QUAD are based on the iteration of a randomly chosen quadratic system of km equations in m variables (with $k = 2$ for S'). If S' and S are equal (or S' consists of the $2n$ first polynomials of S) we have:

$$\mathbf{Adv}^{prf}_{\mathrm{QUAD}}(t,q) \leq 3q\mathbf{Adv}^{prng}_{g^S}(t+q(|IV|+3)T^{L+(k-1)n^2}_{\tilde{g}^S})$$

We can now use the results of Theorem 4 to relate the security of the whole stream cipher QUAD to the difficulty of the MQ problem, i.e. show that if there exists an adversary capable to distinguish QUAD from a perfect random function in time t with q queries then there exists a MQ solver. We estimate the time to compute a quadratic equation in n variables to n^2 and the time to compute a system of kn equations in n unknowns to kn^3.

Theorem 7. *Let us denote* $\lambda = \frac{L+(k-1)n^2}{(k-1)n}$. *Suppose there exists an algorithm A which distinguishes the stream cipher QUAD producing L keystream bits for each of the* $2^{|IV|}$ *IVs from a perfect random function in time T, with q queries and a PRF-advantage* ϵ. *Then there exists an algorithm B which given the image* $S(x)$ *of a randomly (unknown) n-bit value x by a randomly chosen n-bit to kn-bit quadratic system S produces a preimage of* $S(x)$ *with probability at least* $\frac{\epsilon}{3\cdot 2^3 q\lambda}$ *over all possible values of x and S in time upper bounded by* T'.

$$T' = \frac{9\cdot 2^7 n^2\lambda^2 q^2}{\epsilon^2}\left(T + q(|IV|+2)\lambda n^3 + (\lambda+4)kn^2 + \log\left(\frac{9\cdot 2^7 n\lambda^2 q^2}{\epsilon^2}\right)+2\right)$$

Now we have extended the security proof of QUAD to include the key and IV setup we can, as done after the security proof of the keystream generation, propose parameter values for n, k, L, $|K|$ and $|IV|$ allowing to get a concrete security reduction for whole stream cipher. We restrict ourselves to the $GF(2)$ case. We want to ensure a security level of at least 2^{80}. More precisely, we want Theorem 7 to ensure that the existence of an algorithm allowing, for a randomly chosen system S, to distinguish the IV to keystream function induced by the stream cipher QUAD and a random key from a random function with q queries and an advantage of more than $\epsilon = \frac{1}{100}$ in time less than $T = 2^{80}$ would imply the existence of an inversion algorithm of non negligible success probability $\epsilon' = \frac{\epsilon}{3\cdot 2^3\lambda q}$ allowing, given a random n-bit to kn-bit system of quadratic equations and the image $S(x)$ of a random input value x, to find a preimage of $S(x)$ by S in time T' substantially lower, by a factor of more than ϵ', than the best known inversion algorithms for the MQ problem, and thus the existence of a large set of weak instances of MQ.

For $k = 2$ and with the previous values of $L = 2^{40}$, $q = 2^{40}$, $T = 2^{80}$ and $\epsilon = \frac{1}{100}$, we get $\epsilon' = 2^{-78}$ and we need to have n greater than 760. For $n = 512$ and $k = 2$, we only get a contradiction if we produce less than $L = 2^{21}$ bits of keystream and allow up to $q = 2^{30}$ queries. These values of n are higher than those for the keystream generation. However our proofs are not perfectly tight and 760 bits is still quite low compared to the size stream ciphers based on discrete log or RSA would require.

7 Software and Hardware Implementation of QUAD

7.1 Software Implementation

Implementing QUAD in software essentially amounts to computing a system of quadratic functions in n variables over GF(q). This holds for the the key and IV setup and the runup and keystream generation phases of QUAD, the main differences between the two phases being that the number of quadratic functions one has to compute at each iteration are n and $m > n$ respectively. There are two main steps in the computation of a system of m quadratic functions $Q_1, \cdots, Q_m$ in n variables $x_1, \cdots, x_n$: firstly a monomials generation step which consists of computing the $N = \frac{1}{2}n(n+3)+1$ ($(q \neq 2)$ case) or $N = \frac{1}{2}n(n+1)+1$ monomials of degree 0, 1 or 2. Secondly a polynomials computation step which can be viewed as the computation of the product of the vector of the N monomials produced at the first step by the $m \times N$ matrix Q which rows are the coefficients of the m quadratic functions $Q_1, \cdots, Q_m$. Various techniques allowing to efficiently implement these two steps are described in [4]. In the case of a system over a small extension of GF(2), e.g. GF(2^4), the use of bitslicing techniques allows to speed up the first step by computing several monomials in parallel, whereas the use of lookup tables containing, for each column vector of Q and each scalar coefficient $a \in$ GF(q), the product of the column vector by a allows to speed up the second step.

name	**vendor**	**processor**	**frequency**	**L2 cache**
M1	Intel	Pentium 4	2505 MHz	512 kB
M2	Intel	Pentium M	1862 MHz	2048 kB
M3	Intel	Xeon	2784 MHz	512 kB
M4	AMD	Opteron	2197 MHz	1024 kB
M5	AMD	AMD64	1790 MHz	512 kB
M6	AMD	Athlon XP	2162 MHz	512 kB
M7	Power PC	G3	900 MHz	512 kB

Implementations of QUAD in C were produced for the two sets of parameters described hereafter, and a modified version of the eSTREAM Testing Framework made by C. de Cannière [13] was used to evaluate the performance of these implementations on various platforms listed in the table below when

compiled with different compilers and compiling options. We mostly used compilers `gcc-4`, `gcc-3.4`, `gcc-3.3`, and `gcc-2.95`, although Intel's `icc` compiler was also supported.

The two parameters sets we considered are the following:

- The GF(2) version of QUAD with a 160-bit state in which a system of 320 equations in 160 variables is iterated recommended in Section 6. Although $n = 160$ is not enough in order for the security reduction of QUAD to give any formal reduction argument, i.e. any contradiction with the conjectured intractability of MQ, we believe that this is in practice a rather conservative instance of QUAD.
- The GF(16) version of QUAD with a 160-bit state in which a system of 80 equations in 40 variables is iterated. Though there is no evidence so far that existing methods result in an attack of complexity less that 2^{80} against the keystream generator, the underlying MQ problem of solving a system of 80 equations in 40 variables can be solved in substantially less than 2^{80} GF(16) operations. Therefore, this is a much less conservative version of QUAD than the former one that we do not recommend for use in applications with strong security requirements. Although it does not make much sense to compare the performance of two instances which do not offer the same security level, the higher throughput achieved with this version suggests that one may expect some performance improvements when implementing QUAD on a small extension of GF(2) rather than GF(2).

The associated performance figures (in cycles per byte) are given in the two tables hereafter. The orders of magnitudes of the encryption speeds obtained for fastest implementations of the GF(2) instance and the GF(16) instance are 8 Mbit/s and 24 Mbit/s.

Table 1. GF(2) case: $n = 160$, $t = 2$ - speed in cycles/byte

version	M1	M2	M3	M4	M5	M6	M7
32 bit	7057	3746	4600	**2930**	3205	4866	4983
64 bit				**2081**	2636		

Table 2. GF(16) case: $n = 40$, $t = 2$ - speed in cycles/byte

version	M1	M2	M3	M4	M5	M6	M7
32 bit	1906	1204	1849	1003	990	1257	**874**
64 bit				**745**	885		

7.2 Hardware Performance

It might seem at first glance that QUAD is not well fit for hardware implementations because it is impractical to manage a large random system of quadratic equations in hardware. We show in this section that however, if one is willing to generate the fixed multivariate quadratic function S iterated by QUAD pseudo-randomly rather than randomly by means of a simple non linear number generator, this results in surprisingly good hardware performance: about 3500 Gate Equivalents (GE) for the smallest implementation reported here. Though one can argue that the security reduction relating the indistinguishability of the QUAD output to the intractability of a random MQ instance can no longer be invoked in such a setting and that moreover we are considering smaller parameter sizes than those needed to get a strong formal security argument, we think that the existence of a strong link between the security of QUAD and the one of the underlying MQ problem still provides a partial security argument in this setting. We implemented binary version of QUAD with state lengths of $n = 128$, 160 and 256 bits (for an intended security level of approximately $2^{\frac{n}{2}}$ on a Xilinx Virtex4 FPGA. For each state length, we developed two main implementations realizing distinct trade-offs between area and throughput.

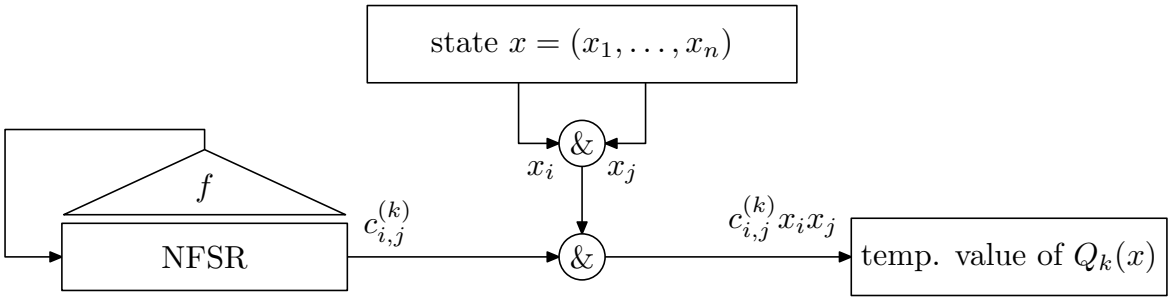


Fig. 2. The low area design $QUAD^{low}$

In the first implementation, named $QUAD^{low}$, area is minimized but the throughput is rather low. The second implementation, named $QUAD^{medium}$, achieves a much higher throughput at the expense of a moderate area increase. In the $QUAD^{low}$ implementation, we performed all binary operations sequentially in order to save area, as depicted in Figure 2. The implementation involves two main components. The first component is a linear feedback shift register (NFSR) which provides the sequence of binary coefficients for $Q_1, \cdots, Q_m$, where $m = 2n$. In our implementation, we used one of the NFSRs of the Achterbahn stream cipher proposal, of length 31 and period $2^{31} - 1$. The second component contains the current state x and produces the associated sequence of momomials $x_i x_j$. At each step, the AND product of the current coefficient and the current monomial is computed, and accumulated in a temporary output value. At the

end of the computation, the obtained final output value is partly fed back to the state memory, and partly output as keystream bits.

Table 3. Low area implementation

Version	**128 bits**	**160 bits**	**256 bits**
Flip/Flops	66	68	68
4 input LUTs	153	169	181
Slices	85	92	97
Gate Equiv. (GE)	2961	3694	4611
Max. Freq. (MHz)	267	244	243
Throughput (Kbps)	16.1	9.5	3.7

Table 4. Medium area implementation with an improved throughput/area ratio

Version	**128 bits**	**160 bits**	**256 bits**
Flip/Flops	350	418	613
4 input LUTs	781	970	1471
Slices	406	509	763
GE	8117	10184	14959
Max. Freq. (MHz)	262	269	260
Throughput (Mbps)	4.1	3.3	2.0

In order to improve the throughput, the $QUAD^{medium}$, implementation simultaneously computes for each pair $(i, j) \in \{1, \cdots, n\} \times \{1, \cdots, m\}$, the $2n$-bit word of the coefficients of monomial $x_i x_j$ in $Q_1, \cdots, Q_m$. For that purpose, the NFSR component of $QUAD^{low}$ was replaced by a more expensive finite state machine (FSM), in which we used the S-boxes of Serpent [1] to produce $2n$ bits at each iteration. Tables 3 and 4 provide detailed performance figures for $QUAD^{low}$, and $QUAD^{medium}$. One can see that for the 160-bit versions, the orders of magnitude of the gate counts are 3500 GE and 10000 GE, whereas the obtained throughputs are approximately 10 Kbit/s and 3.3 Mbit/s.

8 Conclusion

QUAD is a practical stream cipher whose security is provably related, for suitable parameter values, to the conjectured intractability of the MQ problem. QUAD seems well suited for three main kinds of environments:

– **software platforms**, e.g. on PCs, for applications where the use of a cipher of unusually strong security arguments matters and where a throughput of a few Mbit/s is sufficient;

- **embedded devices with hardware encryption capabilities**, e.g. mobile stations. The area-throughput trade-off given by $QUAD^{medium}$ seems best suited for this use case. Due to the fact that for suitable parameter values, the quadratic function iterated in QUAD is expected to be strongly one-way, modes of operation of QUAD allowing to provide other security functionalities than mere encryption, in particular authentication and key agreement, are easy to define.
- **lightweight devices with highly limited computation capabilities such as RFID tags**. On such environments, QUAD can be used to provide encryption and even more advanced security functionnalities such as untraceable identification with forward security. The most compact hardware implementation of QUAD seems to be best suited for this environment.

References

1. Rosss Anderson, Eli Biham, and Lars Ramkilde Knudsen. Serpent: A flexible block cipher with maximum assurance. In *Proceedings of the First Advanced Encryption Standard Conference*, 1998.
2. Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie*. PhD thesis, Université Paris VI, 2004.
3. Mihir Bellare. The Goldreich-Levin Theorem. `http://www-cse.ucsd.edu/users/mihir/courses.html`, 1999.
4. Côme Berbain, Olivier Billet, and Henri Gilbert. Efficient implementations of multivariate quadratic systems. In Selected Areas in Cryptography – SAC 2006, To appear in Lecture Notes in Computer Science. Springer-Verlag, 2006.
5. Côme Berbain, Henri Gilbert, and Jacques Patarin. QUAD: a Practical Stream Cipher with Provable Security. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, Lecture Notes in Computer Science. Springer-Verlag, 2006.
6. Lenore Blum, Manuel Blum, and Mike Shub. A simple unpredictable pseudo-random number generator. *SIAM J. Comput.*, 15(2):364–383, 1986.
7. Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. Comput.*, 13(4):850–864, 1984.
8. Dan Boneh, Shai Halevi, and Nick Howgrave-Graham. The modular inversion hidden number problem. In *ASIACRYPT*, pages 36–51, 2001.
9. Christophe De Cannière and Bart Preneel. Trivium: Specifications. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/001, 2005. Available at `http://www.ecrypt.eu.org/stream`.
10. Nicolas Courtois, Louis Goubin, Willi Meier, and Jean-Daniel Tacier. Solving underdefined systems of multivariate quadratic equations. In *Public Key Cryptography*, pages 211–227, 2002.
11. Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 392–407. Springer-Verlag, 2000.
12. Nicolas Courtois and Jacques Patarin. About the XL Algorithm over $GF(2)$. In Marc Joye, editor, *Topics in Cryptology – CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 141–157. Springer-Verlag, 2003.
13. ECRYPT. eSTREAM: ECRYPT Stream Cipher Project, IST-2002-507932. Available at `http://www.ecrypt.eu.org/stream/`, Accessed September 29, 2005, 2005.

14. Patrik Ekdahl and Thomas Johansson. A new version of the stream cipher SNOW. In Kaisa Nyberg and Howard M. Heys, editors, *Proceedings of Selected Areas in Cryptography – SAC'02*, number 2595, 2002.
15. Jean-Charles Faugère, Hideki Imai, Mitsuru Kawazoe, Makoto Sugita, and Gwénolé Ars. Comparison Between XL and Grbner Basis Algorithms. In Pil Joong Lee, editor, *Advances in Cryptology – ASIACRYPT 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 338–353. Springer-Verlag, 2004.
16. Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In *EUROCRYPT*, pages 245–255, 1996.
17. Aviezri S. Fraenkel and Yaacov Yesha. Complexity of solving algebraic equations. *Inf. Process. Lett.*, 10(4/5):178–179, 1980.
18. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, chapter 7.2 Algebraic Equations over $GF(2)$. W H Freeman & Co, 1979.
19. Oded Goldreich. *The Foundations of Cryptography - Volume 1.* Cambridge University Press, 2001.
20. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *J. ACM*, 33(4):792–807, 1986.
21. Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In D. S. Johnson, editor, *21th ACM Symposium on Theory of Computing – STOC '89*, pages 25–32. ACM Press, 1989.
22. Shafi Goldwasser and Mihir Bellare. Lecture notes on cryptography. Available at `http://www-cse.ucsd.edu/users/mihir/courses.html`, 2001.
23. Shai Halevi, Don Coppersmith, and Charanjit S. Jutla. Scream: A software-efficient stream cipher. In Joan Daemen and Vincent Rijmen, editors, *Proceedings of Fast Software Encryption – FSE'02*, number 2365, pages 195–209, 2002.
24. Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.
25. Johan Håstad and Mats Näslund. BMGL: Synchronous key-stream henerator with provable security. submitted to Nessie Project, 2000.
26. M. Hell, T. Johansson, and W. Meier. Grain - A Stream Cipher for Constrained Environments. ECRYPT Stream Cipher Project Report 2005/001, 2005. `http://www.ecrypt.eu.org/stream`.
27. Russell Impagliazzo and Moni Naor. Efficient cryptographic schemes provably as secure as subset sum. *J. Cryptology*, 9(4):199–216, 1996.
28. Rudolf Lidl and Haradl Niederreiter. *Finite Fields.* Cambride University Press, 1997.
29. Jacques Patarin and Louis Goubin. Asymmetric cryptography with s-boxes. In *ICICS*, pages 369–380, 1997.
30. Ronald Rivest. The RC4 enycryption algorithm. RSA Security Inc., March 1992.
31. Adi Shamir. On the Generation of Cryptographically Strong Pseudo-Random Sequences. In *ICALP*, pages 544–550, 1981.
32. Ron Steinfeld, Josef Pieprzyk, and Huaxiong Wang. On the provable security of an efficient rsa-based pseudorandom generator. In *ASIACRYPT*, pages 194–209, 2006.
33. Andrew Yao. Theory and Applications of Trapdoor Function. In *Foundations of Cryptography FOCS 1982*, 1982.

Liste de publications en cryptographie

[1] H. Gilbert and G. Chassé. A Statistical Attack of the FEAL-8 Cryptosystem. Advances in Cryptology - CRYPTO' 90. Lecture Notes in Computer Science, vol. 537, pp. 22-33, Springer, 1990.

[2] A. Tardy and H. Gilbert. A Known Plaintext Attack of FEAL-4 and FEAL-6. Advances in Cryptology - CRYPTO' 91. Lecture Notes in Computer Science, vol. 576, pp. 172-182, Springer, 1991.

[3] T. Baritaud, H. Gilbert, and M. Girault. FFT Hashing is not Collision-free. Advances in Cryptology - EUROCRYPT' 92. Lecture Notes in Computer Science, vol. 658, pp. 35-44, Springer, 1992.

[4] T. Baritaud, M. Campana, P. Chauvaud, and H. Gilbert. On the Security of the Permuted Kernel Identification Scheme. Advances in Cryptology - CRYPTO' 92. Lecture Notes in Computer Science, vol. 740, pp. 305-311, Springer, 1992.

[5] H. Gilbert and P. Chauvaud. A Chosen Plaintext Attack of the 16-round Khufu Cryptosystem. Advances in Cryptology - CRYPTO' 94. Lecture Notes in Computer Science, vol. 839, pp. 359-368, Springer, 1994.

[6] H. Gilbert. Cryptanalyse statistique des algorithmes de chiffrement. Thèse de doctorat, Université Paris-Sud, 14 mars 1997.

[7] H. Handschuh and H. Gilbert. χ^2 Cryptanalysis of the SEAL Encryption Algorithm. Proceedings of FSE 1997. Lecture Notes in Computer Science, vol. 1267, pp. 1-12, Springer, 1997.

[8] H. Gilbert, D. Gupta, J.J. Quisquater, and A. Odlyzko. Attacks on Shamir's 'RSA for paranoids'. Information Processing Letters 68 (1998).

[9] H. Gilbert. Techniques for Low Cost Authentication and Data Authentication. Proceedings of CARDIS' 98. Lecture Notes in Computer Science, vol. 1820, pp. 183-192, Springer, 2000.

[10] O. Baudron, H. Gilbert, L.Granboulan, H. Handschuh, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, and S. Vaudenay. Report on the AES candidates. Second Advanced Encryption Standard Conference, 1999.

[11] H. Gilbert, H. Handschuh, A. Joux, and S. Vaudenay. A Statistical Attack on RC6. Proceedings of FSE 2000. Lecture Notes in Computer Science, vol. 1978, pp. 64-74, Springer, 2000.

[12] M. Minier and H. Gilbert. Stochastic Cryptanalysis of Crypton. Proceedings of FSE 2000. Lecture Notes in Computer Science, vol. 1978, pp. 121-133, Springer, 2000.

[13] H. Gilbert and M. Minier. A collision attack on 7 rounds of Rijndael. The Third AES Candidates Conference (AES3), pp. 230-241, 2000.

[14] H. Gilbert and M. Minier. New Results on the Pseudorandomness of some Block cipher constructions. Proceedings of FSE 2001. Lecture Notes in Computer Science, vol. 2355, pp. 248-266, Springer, 2001.

[15] C. Debaert and H. Gilbert. The RIPEMD^L and RIPEMD^R Improved Versions of MD4 are not Collision Free. Proceedings of FSE 2001. Lecture Notes in Computer Science, vol. 2355, pp. 52-65, Springer, 2001.

[16] M. Minier and H. Gilbert. Cryptanalysis of SFLASH, Proceedings of EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 288- 298, Springer, 2002.

[17] H. Gilbert. The Security of "One-Block-to-Many" Modes of Operation. Proceedings of FSE 2003. Lecture Notes in Computer Science, vol. 2887, pp. 376-395, Springer, 2003.

[18] H. Gilbert and H. Handschuh. Security Analysis of SHA-256 and Sisters. Proceedings of SAC 2003. Lecture Notes in Computer Science, vol. 2887, pp. 376-395, Springer, 2004.

[19] O. Billet and H. Gilbert, A Traceable Bockcipher. Proceedings of ASIACRYPT 2003. Lecture Notes in Computer Science, vol. 2894, pp. 331-346, Springer, 2003.

[20] O. Billet, H. Gilbert, and C. Ech-Chatbi. Cryptanalysis of a WhiteBox AES Implementation. Proceedings of SAC 2004. Lecture Notes in Computer Science, vol. 3357, pp. 227-240, Springer, 2004.

[21] O. Billet and H. Gilbert. Resistance of SNOW 2.0 against Algebraic Attacks. Proceedings of SASC 2004, October 2004.

[22] O. Billet and H. Gilbert. A Potential Weakness of SNOW 2.0. Proceedings of RSA-CT 2005. Lecture Notes in Computer Science, vol. 3376, pp. 19-28, Springer, 2005.

[23] H. Gilbert and H. Handschuh (editors). Fast Software Encryption : 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005. Revised Selected Papers, Lecture Notes in Computer Science, vol. 3557, Springer, 2005.

[24] H. Gilbert, M. Robshaw, and H. Sibert. An Active Attack Against HB$^+$ - A Provably Secure Lightweight Authentication Protocol. Electronics Letters, 13 October

2005 – Volume 41, Issue 21, pp. 1169-1170.

[25] C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert. Sosemanuk, a Fast Software-oriented Stream Cipher. Proceedings of SKEW - Symmetric Key Encryption Workshop, Network of Excellence in Cryptology ECRYPT, May 2005.

[26] C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert. Decim v2. Proceedings of SASC - ECRYPT Worshop on Stream Ciphers, February 2006.

[27] C. Berbain, H. Gilbert, and J. Patarin. QUAD : A Practical Stream Cipher with Provable Security. Advances in Cryptology - EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 3357, pp. 227-240, Springer, 2006.

[28] C. Berbain, H. Gilbert, and A. Maximov. Cryptanalysis of Grain. Proceedings of FSE 2006. Lecture Notes in Computer Science, vol. 4004, pp. 109-128, Springer, 2006.

[29] O. Billet and H. Gilbert. Cryptanalysis of Rainbow. Proceedings of SCN 2006. Lecture Notes in Computer Science, vol. 4116, pp. 336-347, Springer, 2006.

[30] C. Cid, H. Gilbert, and T. Johansson. Cryptanalysis of Pomaranch. IEE Proceedings – Information Security, Vol. 153, pp. 51-53, 2006.

[31] C. Berbain, O. Billet, and H. Gilbert. Efficient Implementations of Multivariate Quadratic Systems. Proceedings of SAC 2006. Lecture Notes in Computer Science, vol. 4356, pp. 174-187, Springer, 2006.

[32] T. Peyrin, H. Gilbert, F. Muller, and M. J. B. Robshaw. Combining Compression Functions and Block Cipher-Based Hash Functions. Proceedings of ASIACRYPT 2006. Lecture Notes in Computer Science, vol. 4284, pp. 315-331, Springer, 2006.

[33] D. Arditti, C. Berbain, O. Billet, and H. Gilbert. Compact FPGA implementations of QUAD. Proceedings of ASIACCS 2007, pp. 347-349, ACM 2007.

[34] D. Arditti, C. Berbain, O. Billet, H. Gilbert, and J. Patarin. QUAD : Overview and Recent Developments. In Symmetric Cryptography Seminar, Dagstuhl Seminar Proceedings n°07021, IBFI, Germany.

[35] C. Berbain and H. Gilbert. On the Security of IV Dependent Stream Ciphers. Proceedings of FSE 2007. Lecture Notes in Computer Science, vol. 4593, pp. 254-273, Springer, 2007.

[36] H. Gilbert, M. Robshaw, and Y. Seurin. Good Variants of HB^+ are Hard to Find. Proceedings of Financial Crtyptography – FC 2008. Lecture Notes in Computer

Science, Springer (to appear).

[37] H. Gilbert, M. Robshaw, and Y. Seurin. HB$^\sharp$: Increasing the Security and Efficiency of HB$^+$. Advances in Cryptology - EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 361-378, Springer, 2008.

[38] H. Gilbert, M. Robshaw, and Y. Seurin. How to Encrypt with the LPN Problem. Proceedings of ICALP 2008. Lecture Notes in Computer Science. Springer 2008 (to appear).