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, $\mathrm{GF}(2^{32})$ 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 $\mathrm{GF}(2^{32})$ instead of requiring low weight multiples with $\mathrm{GF}(2)$ 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 $\mathrm{GF}(2^{32})$, 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 $\mathrm{GF}(2^{32})$ 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 $\mathrm{GF}(2^{32})$, 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 `shifts`/`xors`, 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

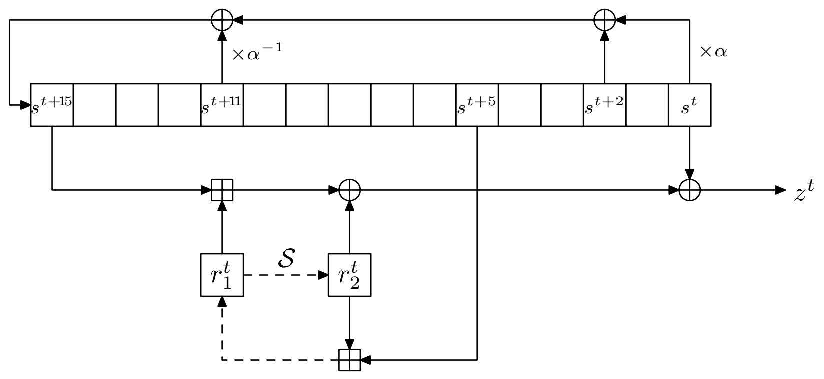

Fig. 1. SNOW 2.0

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

$$\begin{cases} r_2^{t+1} \overset{\text{def}}{=} \mathcal{S}\left(r_1^t\right) \ , \\ r_1^{t+1} \overset{\text{def}}{=} r_2^t \boxplus s^{t+5} \ , \\ \quad F^t \overset{\text{def}}{=} (r_1^t \boxplus s^{t+15}) \oplus r_2^t \ . \end{cases}$$

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

$$s^{t+16} \overset{\text{def}}{=} \alpha^{-1} s^{t+11} \oplus s^{t+2} \oplus \alpha s^t \oplus F^t \ .$$

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

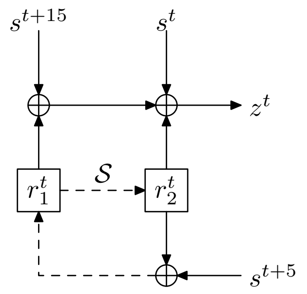

Fig. 2. a variant of SNOW 2.0

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}.

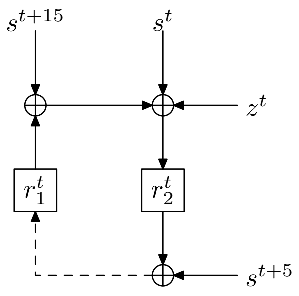

Fig. 3. tracking memory registers r_1 and r_2

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.