Algebraic Cryptanalysis and RFID Authentication

Carlos Cid[1], Loic Ferreira[2], Gordon Procter[1], and Matt J.B. Robshaw[3]

[1] Information Security Group, Royal Holloway University of London
Egham, TW20 OEX, UK
[2] Applied Cryptography Group, Orange Labs
38–40 rue de General Leclerc, 92794 Issy les Moulineaux, France
[3] Impinj, 701 N. 34th Street, Suite 300,
Seattle, WA 98103, USA

Abstract. The standardization group ISO/IEC SC31/WG7 is working on a set of *cryptographic suites* to provide security to wireless devices including UHF RFID tags. These cryptographic suites are presented as independent parts to a single standard ISO/IEC 29167. Within this multi-part standard 29167-15 is based around very simple operations and intended to provide tag, interrogator, and mutual authentication. Here we show that these proposals can be fully compromised using algebraic cryptanalytic techniques; the entire key can be recovered after eavesdropping on just four authentications.

1 Introduction

It is perhaps a sign of commercial maturity that standardization on cryptography for low-cost UHF RFID tags has begun. With such standards to hand tag, interrogator, and mutual authentication, along with secure tag-interrogator communications, can all be considered and securely implemented in the future. Increasingly referred to as RAIN RFID after the foundation of the *RA*dio *I*dentificatio*N* (RAIN) *Industry Alliance* [20], the technology will be widely deployed and is likely to become an integral part of the *Internet of Things (IoT)*. However, such tags pose a major challenge when deploying cryptography since they are very limited in terms of the space available in silicon and the power available for on-tag computation. By comparison HF RFID tags that we find in public transport ticketing and NFC applications are positively luxurious. It is within this context that ISO/IEC 29167-15 has been proposed; to provide security services to resource constrained devices.

One goal of this paper is to highlight the poor—in fact non-existent—security provided by ISO/IEC 29167-15. While the scheme can be compromised in many ways, this paper uses algebraic cryptanalysis to provide an elegant and efficient attack. But an arguably more important goal is to provide a warning of how even technically poor proposals can advance far into the standardization process. Given that there already exist many sound (and standardized) cryptographic designs for HF and UHF RFID tags, we hope our analysis will deter

S. Mangard, P. Schaumont (Eds.): RFIDsec 2015, LNCS 9440, pp. 104–121, 2015.
DOI: 10.1007/978-3-319-24837-0_7

standardization bodies and product developers from adopting schemes that have had little technical scrutiny.

1.1 The Standardization Landscape for UHF RFID

It may not be immediately apparent why yet another group within ISO/IEC is standardizing cryptographic mechanisms. To see why this important work is underway we need to understand the role of other standards in the field.

The commands that can be sent to a (standardized) UHF RFID tag are defined in two documents that have been published[1] by EPCglobal, part of GS1. The dominant standard covering all current large-scale deployments is known as Gen2v1 [8] and the final update to this standard was published in 2008. In 2013, however, the Gen2v2 standard was published [9] and as the version number implies, this extends the functionality of Gen2v1. The most significant and far-reaching additions are optional over-the-air commands that allow the deployment of security functionality.

Gen2v2 defines the over-the-air commands for UHF RFID but this is all it does. For instance, a command AUTHENTICATE is defined and this can be used to develop a solution for tag, interrogator, or mutual authentication. However all the security commands in Gen2v2, including AUTHENTICATE, have been deliberately designed to be both flexible and crypto-agnostic; they are completely independent of any specific cryptographic technology. By way of illustration, the format of the AUTHENTICATE command is given below, with the field descriptions, lengths, and possible values given by the three rows of the table. The `handle` and `CRC-16` are part of the communication protocol while `SenRep` and `IncRepLen` are application options. The most important fields for our purposes are marked $\star$ and their values are not defined by Gen2v2. The `CSI` field identifies the cryptographic algorithm/protocol while the `Length`/`Message` fields identify the cryptographic payload being carried by the command.

	`command`	`RFU`	`SenRep`	`IncRepLen`	`CSI`	`Length`	`Message`	`RN`	`CRC`
length	8	2	1	1	8	12	*variable*	16	16
value	$\texttt{d5}_{\text{x}}$	00_b	$0_b/1_b$	$0_b/1_b$	$\star$	$\star$	$\star$	`handle`	`CRC-16`

For the cryptographic technology itself we would likely turn to the usual sources. NIST standardizes cryptographic technologies such as the *Advanced Encyption Standard (AES)* [16]. Other cryptographic technologies have been standardized within ISO/IEC SC27 and some, such as PRESENT [2,14] and CRYPTOGPS [10,15], are explicitly targeted at constrained environments.

However we can see there is an implementation gap between the over-the-air commands and the cryptographic primitives. For example, the AUTHENTICATE command says nothing about how to achieve tag authentication using, say, a *challenge-response* authentication protocol. It doesn't even say what algorithms might be supported on the tag or interrogator. Similarly, the AES standard

[1] The Gen2v1 specifications are also standardized within ISO/IEC 18000-63 with Gen2v2 standardization underway as a revision.

(FIPS-197 [16]) doesn't tell us how to use the AES block cipher to perform tag authentication; instead FIPS-197 tells us how a 128-bit output is derived from a 128-bit input and a key. It is the goal of the work in ISO/IEC SC31/WG7, therefore, to provide a mapping between the cryptographic primitive and the generic over-the-air command; that is, to fill in the information marked $\star$ in the command above. This mapping is referred to as a *cryptographic suite* and the ISO/IEC 29167 standard consists of several parts, each describing a cryptographic suite and a solution. If one wishes to perform tag authentication using AES-128 then ISO/IEC 29167-10 is the cryptographic suite of interest. For tag authentication with PRESENT-80, Grain-128a or CRYPTOGPS, parts 29167-11, 29167-13, and 29167-17 are, respectively, the ones to use.

Many cryptographic suites in ISO/IEC 29167 are built on trusted or standardized primitives. Some, however, are built around new and immature proposals. Despite very negative comments during the development of 29167-15, the ISO/IEC voting structure is such that even a technically poor proposal can advance far through the standardization process. The current status of 29167-15 is unclear, though it may be moved to a *Technical Specification*. Technical specifications are sometimes used when there are irreconcilable differences of opinion and they provide the opportunity for public comment. After three years the work in the technical specification is then either abandoned or re-introduced to the standards process. This paper can be viewed, therefore, as input to this process and provides compelling support for the view that future work on this standard should be resisted.

2 Early Versions

One reason for the longevity of ISO/IEC 29167-15 is that patches have been applied at several stages during the process. All variants propose mechanisms for tag authentication, interrogator authentication, and mutual authentication. All variants are simple and built around the supposed difficulty of analyzing the combination of bitwise exclusive-or and integer addition, though first proposals were even simpler; see Table 1.

There are many problems with the proposal in Table 1 but the most pressing is that there is no security. The sole use of a single operator (in this case bitwise exclusive-or) gives a differential attack. By eavesdropping an attacker recovers $\mathtt{RI}$, $\mathtt{RT}$, $\mathtt{A}$, and $\mathtt{B}$. The adversary can then make a fake tag that fools a legitimate reader *without* knowing the secret key $\mathtt{K}$. The attack is outlined in Table 2 where variables in a subsequent run of the protocol are denoted *. To confirm that the fake tag is always accepted as genuine we observe that

$$\begin{aligned}
\text{REVERSE}(\mathtt{T}^*) &= \text{REVERSE}(\mathtt{SK}^* \oplus \mathtt{B} \oplus \text{REVERSE}(\mathtt{X}) \oplus \Delta_{SK}) \\
&= \text{REVERSE}(\mathtt{SK} \oplus \mathtt{B}) \oplus \mathtt{X} = \text{REVERSE}(\mathtt{SK} \oplus (\mathtt{SK} \oplus \mathtt{RCI})) \oplus \mathtt{X} \\
&= \mathtt{CI} \oplus \mathtt{X} = \mathtt{CI} \oplus \Delta_A \oplus \Delta_{SK} \\
&= (\mathtt{SK} \oplus \mathtt{A}) \oplus (\mathtt{A} \oplus \mathtt{A}^*) \oplus (\mathtt{SK} \oplus \mathtt{SK}^*) = \mathtt{A}^* \oplus \mathtt{SK}^* = \mathtt{CI}^*
\end{aligned}$$

A tag can be cloned after eavesdropping one legitimate authentication.

Table 1. The first version of the tag authentication scheme where all variables are 64 bits long

Interrogator (secret key K)		**Tag** (secret key K)
Choose random RI	$\xrightarrow{\texttt{RI}}$	
	$\xleftarrow{\texttt{RT}}$	Choose RT
$\texttt{SK} = \texttt{K} \oplus \texttt{RI} \oplus \texttt{RT}$		$\texttt{SK} = \texttt{K} \oplus \texttt{RI} \oplus \texttt{RT}$
Choose CI		
$\texttt{A} = \texttt{SK} \oplus \texttt{CI}$	$\xrightarrow{\texttt{A}}$	
		$\texttt{CI} = \texttt{SK} \oplus \texttt{A}$
		$\texttt{RCI} = \text{REVERSE}(\texttt{CI})$
	$\xleftarrow{\texttt{B}}$	$\texttt{B} = \texttt{SK} \oplus \texttt{RCI}$
$\texttt{T} = \texttt{SK} \oplus \texttt{B}$		
$\text{REVERSE}(\texttt{T}) \stackrel{?}{=} \texttt{CI}$		

Table 2. Fooling a legitimate reader during tag authentication. The attacker has eavesdropped on one run of the protocol in Table 1. The (changing) parameters in this second run are indicated using *.

Interrogator (secret key K)		**Fake Tag**
Choose random $\texttt{RI}^*$	$\xrightarrow{\texttt{RI}^*}$	
	$\xleftarrow{\texttt{RT}^*}$	Choose $\texttt{RT}^*$
$\texttt{SK}^* = \texttt{K} \oplus \texttt{RI}^* \oplus \texttt{RT}^*$		$\texttt{SK}^*$ is unknown
		$\Delta_I = \texttt{RI} \oplus \texttt{RI}^*$
		$\Delta_T = \texttt{RT} \oplus \texttt{RT}^*$
		Save $\Delta_{SK} = \Delta_I \oplus \Delta_T$
Choose $\texttt{CI}^*$		
$\texttt{A}^* = \texttt{SK}^* \oplus \texttt{CI}^*$	$\xrightarrow{\texttt{A}^*}$	
		$\texttt{A} \oplus \texttt{A}^* = \Delta_A$
		$\texttt{X} = \Delta_A \oplus \Delta_{SK}$
	$\xleftarrow{\texttt{B}^*}$	$\texttt{B}^* = \texttt{B} \oplus \text{REVERSE}(\texttt{X}) \oplus \Delta_{SK}$
$\texttt{T}^* = \texttt{SK}^* \oplus \texttt{B}^*$		
$\text{REVERSE}(\texttt{T}^*) \stackrel{?}{=} \texttt{CI}^*$		

Table 3. A second tag authentication scheme. The shared secret key $\mathtt{K}$ and all intermediate values are 64 bits long, $\oplus$ denotes bitwise exclusive-or, and $+$ denotes integer addition modulo 2^{64}.

Interrogator (secret key $\mathtt{K}$)		**Tag** (secret key $\mathtt{K}$)
Choose random $\mathtt{R}$		
$\mathtt{S} = \mathtt{R} \oplus \mathtt{K}$	$\xrightarrow{\mathtt{S}}$	$\mathtt{R} = \mathtt{S} \oplus \mathtt{K}$
		$\mathtt{A} = \mathtt{R} + \mathtt{0x55}\cdots\mathtt{55}$
$\mathtt{B} = \mathtt{T} \oplus \mathtt{K}$	$\xleftarrow{\mathtt{T}}$	$\mathtt{T} = \mathtt{A} \oplus \mathtt{K}$
$\mathtt{B} \stackrel{?}{=} \mathtt{R} + \mathtt{0x55}\cdots\mathtt{55}$		

After this inauspicious start a second proposal is illustrated in Table 3. Interrogator authentication is provided by reversing the roles of the two participants while mutual authentication is derived by interleaving two sessions that establish tag and interrogator authentication.

The weaknesses are immediately apparent and, as before, there are too many problems to list. It is sufficient to show that we can recover the key in a passive attack with high reliability. Indeed, suppose an attacker intercepts $\mathtt{S}$ and $\mathtt{T}$ from a legitimate authentication session. We then have that

$$\mathtt{S} \oplus \mathtt{T} = (\mathtt{R} + \mathtt{0x55}\cdots\mathtt{55}) \oplus \mathtt{R}.$$

The least significant bit of $\mathtt{S} \oplus \mathtt{T}$ is always set to 1 and further analysis of $\mathtt{S} \oplus \mathtt{T}$ is easy to make. For instance, the 2^{32} values of $\mathtt{R}$ for which $\mathtt{R} \wedge \mathtt{0x55}\cdots\mathtt{55} = \mathtt{0x00}\cdots\mathtt{00}$ will give $\mathtt{S} \oplus \mathtt{T} = \mathtt{0x55}\cdots\mathtt{55}$ and other observations based on $\mathtt{S} \oplus \mathtt{T}$ can be used to recover $\mathtt{R}$ and, via $\mathtt{S}$, the shared secret key $\mathtt{K}$.

For an alternative approach we simplify the notation by setting $\mathtt{X} = \mathtt{S} \oplus \mathtt{T}$ and $\mathtt{C} = \mathtt{0x55}\cdots\mathtt{55}$. This means that $\mathtt{X} = \mathtt{R} \oplus (\mathtt{R} + \mathtt{C})$ and considering this equation bit-by-bit gives, for bit position j with $j \geq 0$,

$$\mathtt{X}_j = \mathtt{R}_j \oplus ((\mathtt{R}_j + \mathtt{C}_j + \mathtt{c}_{j-1}) \bmod 2) = \mathtt{C}_j \oplus \mathtt{c}_{j-1}$$

where $\mathtt{c}_{j-1}$ denotes the carry given at bit $j-1$ generated within the integer addition $\mathtt{R} + \mathtt{C}$. Setting $\mathtt{c}_{-1} = 0$, the carry bit $\mathtt{c}_j$ for $j \geq 0$ is computed as:

$$\begin{aligned} \mathtt{c}_j &= \mathrm{MAJ}(\mathtt{R}_j,\ \mathtt{C}_j,\ \mathtt{c}_{j-1}) = (\mathtt{R}_j \wedge \mathtt{C}_j) \oplus (\mathtt{R}_j \wedge \mathtt{c}_{j-1}) \oplus (\mathtt{C}_j \wedge \mathtt{c}_{j-1}) \\ &= (\mathtt{R}_j \wedge (\mathtt{C}_j \oplus \mathtt{c}_{j-1})) \oplus (\mathtt{C}_j \wedge \mathtt{c}_{j-1}) \end{aligned}$$

where MAJ denotes the *majority function*. Hence, for $j \geq 0$,

$$\begin{aligned} \mathtt{X}_{j+1} &= \mathtt{C}_{j+1} \oplus (\mathtt{R}_j \wedge (\mathtt{C}_j \oplus \mathtt{c}_{j-1})) \oplus (\mathtt{C}_j \wedge \mathtt{c}_{j-1}) \\ &= \mathtt{C}_{j+1} \oplus (\mathtt{R}_j \wedge \mathtt{X}_j) \oplus (\mathtt{C}_j \wedge (\mathtt{X}_j \oplus \mathtt{C}_j)) \\ &= \mathtt{C}_{j+1} \oplus (\mathtt{R}_j \wedge \mathtt{X}_j) \oplus (\mathtt{C}_j \wedge (\mathtt{X}_j \oplus 1)) \end{aligned}$$

This means that, for $j \geq 0$, we have $\mathtt{R}_j \wedge \mathtt{X}_j = \mathtt{X}_{j+1} \oplus \mathtt{C}_{j+1} \oplus (\mathtt{C}_j \wedge (\mathtt{X}_j \oplus 1))$, which we write, setting $\mathtt{V}_j = \mathtt{R}_j \wedge \mathtt{X}_j$, as

$$\mathtt{V}_j = \mathtt{X}_{j+1} \oplus \mathtt{C}_{j+1} \oplus (\mathtt{C}_j \wedge (\mathtt{X}_j \oplus 1)) \text{ for } j \geq 0.$$

Looking at the expression for $\mathtt{V}_j$ we see that it consists entirely of arguments from $\mathtt{X}$, which is available to an eavesdropper, and $\mathtt{C}$ which is fixed and known. Thus if $\mathtt{X}_j = 1$ for bit j, which we expect half the time, then we can compute $\mathtt{R}_j$ directly and the corresponding bit of the shared secret $\mathtt{K}$ is given by

$$\mathtt{K}_j = \mathtt{R}_j \oplus \mathtt{X}_j.$$

Each bit $\mathtt{K}_j$, for $0 \leq j \leq 62$ can be recovered and we expect to be able to recover all but the most significant bit of $\mathtt{K}_j$ with two intercepted authentications. The work-effort is negligible. Note that this gives us two possible values for the full 64-bit shared secret $\mathtt{K}$. However these two keys are *equivalent*, that is they behave identically in the authentication protocol, and so they can both be used to impersonate a tag.

3 More Advanced Versions

Those not involved in ISO/IEC standardization may be somewhat mystified by what is happening with ISO/IEC 29167-15. Early versions were clearly weak and offered little promise. Yet voting was such that the scheme moved forward towards standardization anyway. Once we arrive at a *committee draft (CD)* the document should, in theory, be technically mature since each subsequent stage of the process, namely *draft international standard (DIS)* and *final draft international standard (FDIS)* provide little opportunity for technical modification before publication. Yet as we will show in this section, even advanced versions of these schemes were far from being technically mature and were, in fact, completely insecure.

3.1 Applying Algebraic Cryptanalysis

To repair earlier weaknesses a patched version was briefly proposed; see Table 4.

Conventional Observations. Again, we can immediately see that the least significant bit of $\mathtt{S} \oplus \mathtt{T}$ is always set to 0. It is easy to find other faults and by setting $\mathtt{C} = \mathtt{0x55} \cdots \mathtt{55}$ we have $\mathtt{S} = (\mathtt{RI} + \mathtt{C}) \oplus \mathtt{K}$ and $\mathtt{T} = (\mathtt{K} + \mathtt{C}) \oplus \mathtt{RI}$ so

$$\mathtt{S} \oplus \mathtt{K} = ((\mathtt{K} + \mathtt{C}) \oplus \mathtt{T}) + \mathtt{C}. \tag{1}$$

We can use Equation 1 as a distinguisher to check if a possible value for the key $\mathtt{K}$ is a correct candidate. This can be done in several ways, but we illustrate a byte-by-byte approach, first considering the least significant byte of $\mathtt{K}$. Suppose k, a, b are the least significant bytes of $\mathtt{K}$, $\mathtt{S}$, and $\mathtt{T}$ respectively. Then any x

Table 4. Another patched version of the tag authentication scheme. All variables are 64 bits long and $\mathtt{C} = \mathtt{0x55} \cdots \mathtt{55}$.

Interrogator (secret key K)		**Tag** (secret key K)
Choose RI		
$\mathtt{S} = (\mathtt{RI} + \mathtt{C}) \oplus \mathtt{K}$	$\xrightarrow{\mathtt{S}}$	$\mathtt{RI} = (\mathtt{S} \oplus \mathtt{K}) - \mathtt{C}$
$\mathtt{T} \oplus \mathtt{RI} \stackrel{?}{=} \mathtt{K} + \mathtt{C}$	$\xleftarrow{\mathtt{T}}$	$\mathtt{T} = (\mathtt{K} + \mathtt{C}) \oplus \mathtt{RI}$

satisfying $a \oplus x = ((x + \mathtt{0x55}) \oplus b) + \mathtt{0x55}$ is a good candidate for k. After eight to sixteen runs, one value should be predicted with close to 100% reliability and the least significant byte of the key is recovered. In *parallel* we can process other bytes of K in the same manner. There is a slight complication due to the possibility of a carry from one byte to another in the integer addition; at the same time the most significant bit of each byte might also need some attention. However, closer analysis when using particular values S and T can be used to avoid significant carry propagation. This allows us to filter incorrect values and the few key candidates that remain can be tested against the tag to find the right one. Passively eavesdropping on eight to sixteen authentication runs appears to be sufficient to recover each byte of the key K with good reliability. The work effort is negligible since all bytes can be treated in parallel.

Algebraic Cryptanalysis. Algebraic attacks are powerful techniques that have been successfully applied against several stream ciphers (*e.g.* [5,6]) and considered against block ciphers and other cryptographic primitives [3,4]. In algebraic attacks, one writes the entire cryptographic operation as a large set of multivariate polynomial equations and then uses equation-solving techniques to recover the value of some of the unknown variables (*e.g.* the encryption key). The scheme in ISO/IEC 29167-15 uses a set of very simple operations and algebraic cryptanalysis proves to be very effective. We describe our attack below.

Let n be the size of the set of all variables in the protocol; in our case we have $n = 64$. We will assume that an attacker can eavesdrop on m runs of a uni-directional (tag or interrogator) authentication protocol. Since the mutual authentication protocol consists of two interleaved runs of a uni-directional protocol, observing m runs of the mutual authentication protocol will give identical results to $2m$ runs of the uni-directional protocol.

Without loss of generality we have implemented the attack on the protocol for uni-directional (tag or interrogator) authentication. Denote the value of S on the t^{th} run of the protocol by $\mathtt{S}^t$, similarly for T and RI. We use variables $\mathtt{p_i}$, $\mathtt{s_{t\_i}}$, $\mathtt{r_{t\_i}}$, $\mathtt{so_{t\_i}}$, $\mathtt{a_{t\_i}}$, and $\mathtt{b_{t\_i}}$ for $0 \leq i \leq 63$ and $0 \leq t < m$ assuming that all strings are written using big-endian convention with the most significant bit on the left, *i.e* $\mathtt{K} = \mathtt{p_{n-1}} \ldots \mathtt{p_1 p_0}$.

We represent the computation of $\mathtt{S}^t$ using the equations

$$\mathtt{s_{t\_i}} = \mathtt{p_i} + \mathtt{r_{t\_i}} + \mathtt{a_{t\_i}} + \mathtt{c_i}$$

for $0 \leq i < n$, $0 \leq t < m$ and where $\mathtt{a_{t\_i}}$ is the carry bit during the modular addition of $\mathtt{RI}^t$ and $\mathtt{C}$. This gives, for $1 \leq i \leq n$,

$$\begin{aligned}\mathtt{a_{t\_0}} &= 0\\ \mathtt{a_{t\_i}} &= \mathrm{MAJ}(\mathtt{r_{t\_(i-1)}}, \mathtt{c_{i-1}}, \mathtt{a_{t\_(i-1)}})\\ &= \mathtt{r_{t\_(i-1)}} * \mathtt{c_{i-1}} + \mathtt{r_{t\_(i-1)}} * \mathtt{a_{t\_(i-1)}} + \mathtt{a_{t\_(i-1)}} * \mathtt{c_{i-1}}\end{aligned}$$

Similarly the computation of $\mathtt{T}^t$ can be represented, for $1 \leq i < n$ and $0 \leq t < m$, as:

$$\begin{aligned}\mathtt{b_{t\_0}} &= 0\\ \mathtt{so_{t\_i}} &= \mathtt{p_i} + \mathtt{r_{t\_i}} + \mathtt{b_{t\_i}} + \mathtt{c_i}\\ \mathtt{b_{t\_i}} &= \mathtt{p_{i-1}} * \mathtt{c_{i-1}} + \mathtt{p_{i-1}} * \mathtt{b_{t\_(i-1)}} + \mathtt{b_{t\_(i-1)}} * \mathtt{c_{i-1}}\end{aligned}$$

where the $\mathtt{b}$ variables denote the carry bits. Finally, we define equations to represent the observed values of $\mathtt{S}^t$ and $\mathtt{T}^t$ and the defined value of $\mathtt{C}$; $\mathtt{s_{t\_i}} = \mathtt{S}_i^t$, $\mathtt{so_{t\_i}} = \mathtt{T}_i^t$, and $\mathtt{c_i} = \mathtt{C}_i$. Our system is presented as polynomials over the finite field $\mathbb{F}_2$, i.e. all variables and coefficients take values in $\{0, 1\}$. We may therefore include the equations of the form $x^2 = x$ for every variable x.

The complete set of equations can be summarized as follows:

$$\begin{cases}
\mathtt{s_{t_i}} = \mathtt{p_i} + \mathtt{r_{t_i}} + \mathtt{a_{t_i}} + \mathtt{c_i} & 0 \leq i < n & 0 \leq t < m\\
\mathtt{a_{t_(i+1)}} = \mathtt{r_{t_i}} * \mathtt{c_i} + \mathtt{r_{t_i}} * \mathtt{a_{t_i}} + \mathtt{a_{t_i}} * \mathtt{c_i} & 0 \leq i < n-1 & 0 \leq t < m\\
\mathtt{a_{t_0}} = 0 & 0 \leq t < m &\\
\mathtt{so_{t_i}} = \mathtt{p_i} + \mathtt{r_{t_i}} + \mathtt{b_{t_i}} + \mathtt{c_i} & 0 \leq i < n & 0 \leq t < m\\
\mathtt{b_{t_(i+1)}} = \mathtt{p_i} * \mathtt{c_i} + \mathtt{p_i} * \mathtt{b_{t_i}} + \mathtt{b_{t_i}} * \mathtt{c_i} & 0 \leq i < n-1 & 0 \leq t < m\\
\mathtt{b_{t_0}} = 0 & 0 \leq t < m &\\
\mathtt{s_{t_i}} = \mathtt{S}_i^t & 0 \leq i < n & 0 \leq t < m\\
\mathtt{so_{t_i}} = \mathtt{T}_i^t & 0 \leq i < n & 0 \leq t < m\\
\mathtt{c_i} = \mathtt{C}_i & 0 \leq i < n &\\
x^2 = x & \text{for all } x &
\end{cases}$$

This system of polynomial equations includes $5nm+2n$ variables and $11nm+3n$ equations of degree at most two. Of course this system can be greatly simplified, by substituting the variables that have a fixed value (e.g. $\mathtt{c_i}, \mathtt{a_{t\_0}}, \mathtt{b_{t\_0}}$), as well as the ones observed in the protocol runs ($\mathtt{s_{t\_i}}$ and $\mathtt{so_{t\_i}}$). This reduces the number of variables to $(3n-2)m+n$, and the number of equations to $(7n-4)m+n$. For the parameter values of relevance to ISO/IEC 29167-15, the entire system will consist therefore of $444m+64$ equations in $190m+64$ variables, which can be constructed for the very small values of m that are required to recover the key. We use Gröbner bases algorithms to solve this system [4].

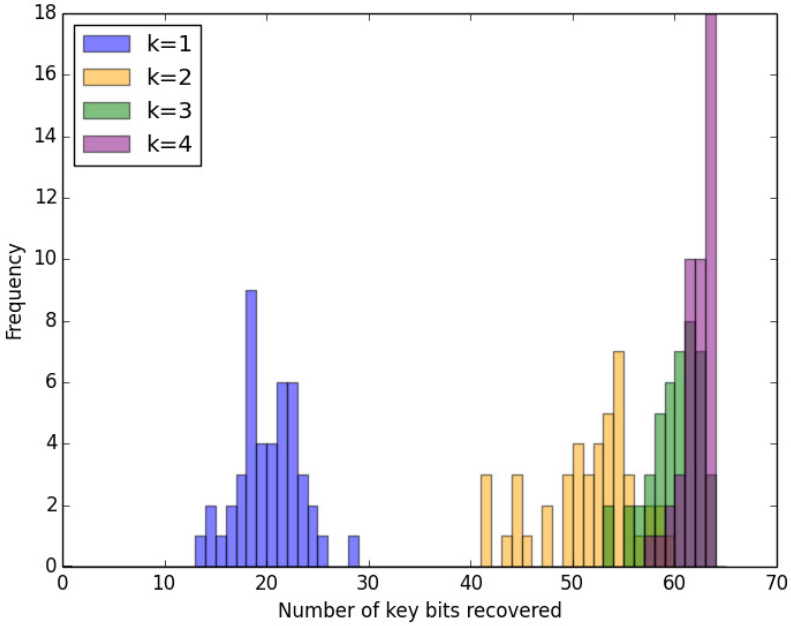

Fig. 1. Results from experiments on the scheme of Section 3.1. The number of protocol runs is given by k in this figure and each set of experiments was repeated 45 independent times.

Results. The average number of key bits recovered and the average time required to solve the system of equations are given in the following table.

Number of protocol runs	1	2	3	4
Average number of key bits recovered	19.7	51.1	59.4	61.7
Average run time (s)	8	83	113	255

The attack was implemented on *SageMathCloud* [21] and timed using Python's `timeit` function; any set-up time is assumed to be pre-computed or amortized over many protocol runs. Figure 1 provides a visualization of the number of key bits recovered after observing m runs of the uni-directional authentication protocol and performing 45 sets of experiments. The values $(\mathtt{K}, \mathtt{RI}^1)$ were chosen randomly for each trial when $m = 1$. As the number of protocol runs was increased $\mathtt{K}$ remains unchanged but fresh random choices were used for $\mathtt{RI}^2$, $\mathtt{RI}^3$, and $\mathtt{RI}^4$ as would be expected in a real-life implementation.

Our experiments suggest that after witnessing four uni-directional runs of the protocol – or just two mutual authentication runs – the attacker would be able to recover 62 out of 64 bits of the secret key in around 84% of the time. While the entirety of the key can often be recovered, we conjecture that the "missing" bits that occur from time-to-time are neutral bits; the values of these bits cannot be

determined by that particular instance of the equation system. This is a feature to many cryptanalytic techniques and is often exhibited in the most significant bits of operations such as integer addition. While it might be an interesting exercise to provide an exact explanation of this phenomenon, it is not relevant to the essential message of our cryptanalysis.

3.2 The Most Advanced Version

The fourth iteration to be described in this paper is the mutual authentication protocol outlined in Table 5. As in previous versions tag and interrogator authentication are derived from the relevant halves of the full authentication protocol. Several changes have been made to this latest proposal to complicate the task of the cryptanalyst. The use of the (unknown) bitwise rotation seems to prevent the attacker from aligning bits in the challenge and response in a trivial way. For example a single bit change in the challenge will change the Hamming Weight of the candidate RI derived by the tag; this would result in different rotation amounts being used for the computation of the final response. Despite these complications it is straightforward to compromise the scheme. Our initial analysis suggested that even with the rotation operation, the scheme can be compromised using conventional cryptanalysis after intercepting around 32 uni-directional authentication runs. However, as demonstrated in the previous section, it is more elegant and efficient to use algebraic cryptanalysis against the scheme. This technique allows us again to recover the shared secret key K after eavesdropping on as few as four authentication runs.

Algebraic Cryptanalysis. As before, we need to set up a system of multivariate polynomial equations. The system used to describe this latest scheme is similar to that of Section 3.1. However it is helpful to introduce some additional variables to take account of the rotation: $\mathtt{m}_{\mathtt{t\_i}}$ corresponds to $\mathtt{K}'_i$ in the t^{th} protocol run and $\mathtt{n}_{\mathtt{t\_i}}$ corresponds to $\mathtt{RI}'_i$ in the t^{th} protocol run. In truth these variables have been introduced to improve the exposition of the attack. It would be straightforward to work without them if there were significant advantage in doing so.

In this variant of the scheme we need to take account of the unknown rotation amount. The simplest way to do this is to guess the rotation amount, and to solve each set of equations that arise for each guess. To include this within the equation system we introduce an array rot_guess where rot_guess$[t]$ is a guess for the Hamming weight of $\mathtt{RI}^t$.

The complete set of equations is summarized overleaf and the system includes $7nm + 2n$ variables and $15nm + 3n$ equations of degree at most two. Again, this system can be greatly simplified by substituting fixed/known value variables, as well as redundant ones, to a system with $(3n-2)m+n$ variables, and $(7n-4)m+n$ equations. For the parameter values of relevance to ISO/IEC 29167-15, the entire system consists of $444m + 64$ equations in $190m + 64$ variables, which can be constructed for the small values of m that are required to recover the key.

Table 5. The last version of the mutual authentication scheme. The *Hamming weight* of $\mathtt{A}$ is denoted $\mathrm{HW}(\mathtt{A})$ while $\mathtt{A} \lll w$ (*resp.* $\mathtt{A} \ggg w$) denotes a left (*resp.* right) bitwise rotation of $\mathtt{A}$ by w bits. The constant $\mathtt{C}$ takes the value $\mathtt{0x55}\cdots\mathtt{55}$.

Interrogator (secret key K)		**Tag** (secret key K)
Choose RI		
$\mathtt{S} = (\mathtt{RI} + \mathtt{C}) \oplus \mathtt{K}$	$\xrightarrow{\mathtt{S}}$	$\mathtt{RI} = (\mathtt{S} \oplus \mathtt{K}) - \mathtt{C}$
		$w_i = \mathrm{HW}(\mathtt{RI})$
		$\mathtt{K}' = \mathtt{K} \lll w_i$
		$\mathtt{RI}' = \mathtt{RI} \lll w_i$
		$\mathtt{T} = (\mathtt{K}' + \mathtt{C}) \oplus \mathtt{RI}'$
		Choose RT
$w_i = \mathrm{HW}(\mathtt{RI})$	$\xleftarrow{\mathtt{T},\mathtt{U}}$	$\mathtt{U} = (\mathtt{RT} + \mathtt{C}) \oplus \mathtt{K}$
$\mathtt{K}' = \mathtt{K} \lll w_i$		
$\mathtt{RI}' = \mathtt{RI} \lll w_i$		
$(\mathtt{T} \oplus \mathtt{RI}') \stackrel{?}{=} (\mathtt{K}' + \mathtt{C})$		
tag authenticated		
$\mathtt{RT} = (\mathtt{U} \oplus \mathtt{K}) - \mathtt{C}$		
$w_t = \mathrm{HW}(\mathtt{RT})$		
$\mathtt{K}'' = \mathtt{K} \lll w_t$		
$\mathtt{RT}'' = \mathtt{RT} \lll w_t$		
$\mathtt{V} = (\mathtt{K}'' + \mathtt{C}) \oplus \mathtt{RT}''$	$\xrightarrow{\mathtt{V}}$	$w_t = \mathrm{HW}(\mathtt{RT})$
		$\mathtt{K}'' = \mathtt{K} \lll w_t$
		$\mathtt{RT}'' = \mathtt{RT} \lll w_t$
		$(\mathtt{V} \oplus \mathtt{RT}'') \stackrel{?}{=} (\mathtt{K}'' + \mathtt{C})$
		interrogator authenticated

$$
\begin{cases}
\mathtt{s_{t\_i}} = \mathtt{p_i} + \mathtt{r_{t\_i}} + \mathtt{a_{t\_i}} + \mathtt{c_i} & 0 \le i < n & 0 \le t < m \\
\mathtt{a_{t\_(i+1)}} = \mathtt{r_{t\_i}} * \mathtt{c_i} + \mathtt{r_{t\_i}} * \mathtt{a_{t\_i}} + \mathtt{a_{t\_i}} * \mathtt{c_i} & 0 \le i < n-1 & 0 \le t < m \\
\mathtt{a_{t\_0}} = 0 & 0 \le t < m & \\
\mathtt{m_{t\_i}} = \mathtt{p_{(i+rot\_guess[t]\%n)}} & 0 \le i < n & 0 \le t < m \\
\mathtt{n_{t\_i}} = \mathtt{r_{t\_(i+rot\_guess[t]\%n)}} & 0 \le i < n & 0 \le t < m \\
\mathtt{so_{t\_i}} = \mathtt{m_{t\_i}} + \mathtt{n_{t\_i}} + \mathtt{b_{t\_i}} + \mathtt{c_i} & 0 \le i < n & 0 \le t < m \\
\mathtt{b_{t\_(i+1)}} = \mathtt{m_{t\_i}} * \mathtt{c_i} + \mathtt{m_{t\_i}} * \mathtt{b_{t\_i}} + \mathtt{b_{t\_i}} * \mathtt{c_i} & 0 \le i < n-1 & 0 \le t < m \\
\mathtt{b_{t\_0}} = 0 & 0 \le t < m & \\
\mathtt{s_{t\_i}} = \mathtt{S}_i^t & 0 \le i < n & 0 \le t < m \\
\mathtt{so_{t\_i}} = \mathtt{T}_i^t & 0 \le i < n & 0 \le t < m \\
\mathtt{c_i} = \mathtt{C}_i & 0 \le i < n & \\
x^2 = x & \forall x &
\end{cases}
$$

Guessing the Rotation Amount. The equation-system depends on rot_guess, a guess for the Hamming weight of RI. We expect that a correct guess for the Hamming weight of RI will yield a system of equations that is easily solved to reveal many key bits. The values RI are random and so assuming that they are generated uniformly the random variable HW(RI) will be distributed according to a binomial distribution with parameters $(64, \frac{1}{2})$. It is therefore straightforward to compute the probability that a randomly chosen RI has a particular Hamming weight. Of particular relevance to our attack is the fact that a substantial fraction of all possible RI have a Hamming weight lying within a small range.

x	32	33	34	35	36
$\Pr_{\mathtt{RI}}[\mathrm{HW}(\mathtt{RI}) = x]$	0.10	0.10	0.09	0.08	0.06

This means that the probability HW(RI) lies in the interval $[32 - \delta, 32 + \delta]$ is:

δ	0	1	2	3	4
$\Pr_{\mathtt{RI}}[\mathrm{HW}(\mathtt{RI}) \in [32 - \delta, 32 + \delta]]$	0.10	0.29	0.46	0.62	0.74

Now assume an attacker who eavesdrops on one uni-directional authentication session. He could simply run the equation solving algorithm three times with the guesses of 31, 32, and 33 for the rotation amount. With a probability close to 30% one of these guesses would be correct. Alternatively, he could elect to further try the values 28, 29, 30, 34, 35, and 36 which would require nine runs of the equation solving algorithm. The probability that the eavesdropped session is covered by one of these nine guesses is close to 74%. This has been confirmed by experiments.

It turns out that the polynomial system solving algorithm is a good method to verify whether the guessed rotation amount is correct. Empirically it seems that selecting the wrong rotation makes the system inconsistent, *i.e.* there will be no valid solution and this is quickly detected by the Gröbner bases algorithm. Of course it cannot be ruled out that cases exist when an incorrect guess of rotation results in a system for which solutions exist (corresponding to an incorrect key). However this does not appear to be common; in over 20 experiments we never recovered false solutions for any $\delta \leq 5$. Of course, even if they did occur, false alarms could easily be filtered out using further intercepted authentication attempts or even a forgery attempt.

Results. The average number of key bits recovered and the average time required to solve the system of equations are given in the following table.

Number of protocol runs	1	2	3	4
Average number of key bits recovered	21.3	51.4	60.6	63.2
Average run time (s)	10	53	87	193

Again, the attack was implemented on *SageMathCloud* [21] and timed using Python's `timeit` function; any set-up time is assumed to be pre-computed or amortized over many protocol runs. Figure 2 provides a visualization of the

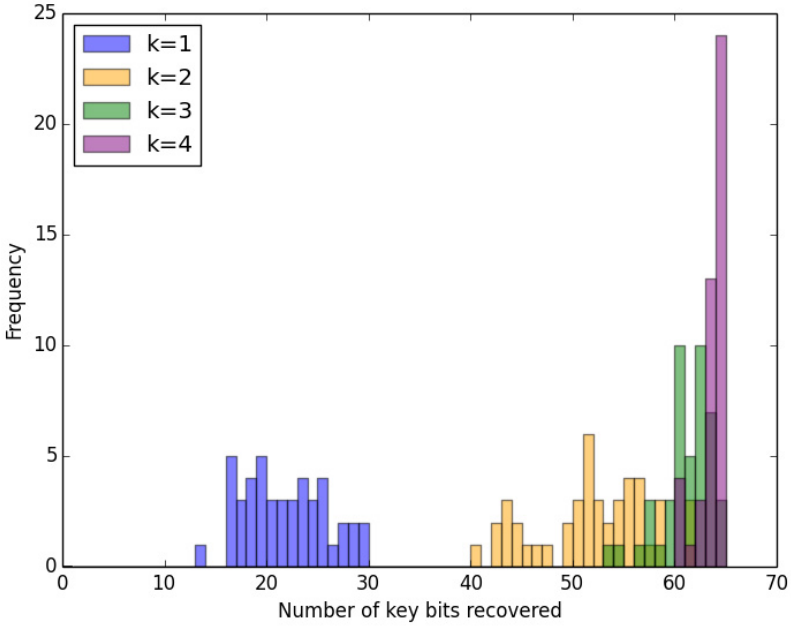

Fig. 2. Results from experiments on the scheme of Section 3.2. The number of protocol runs is given by k in this figure and 45 random instances were generated.

number of key bits recovered after observing m runs of the uni-directional authentication protocol over all 45 sets of experiments. The values $(\mathtt{K}, \mathtt{RI}^1)$ were chosen randomly for each trial when $m = 1$. As the number of protocol runs was increased $\mathtt{K}$ remains unchanged but fresh random choices were used for $\mathtt{RI}^2$, $\mathtt{RI}^3$, and $\mathtt{RI}^4$ as would be expected in a real-life implementation. These results assume a correct guess for $\mathrm{HW}(\mathtt{RI}^j)$, as discussed in the previous section.

Witnessing four uni-directional runs, or two mutual authentication runs, of the scheme in Section 3.2 and guessing the rotation amount correctly gives a probability of approximately 85% for recovering all but two bits of the key. However different attack strategies are possible.

Note that the attack using a single observed run is much faster than the attack with four observed runs, although it would recover a smaller proportion of the secret key bits. Moreover, as discussed above, when attempting to solve the resulting system of equations it is straightforward to recognize when an incorrect rotation has been guessed. Thus an efficient way to perform the full attack is as follows: we try the attack on four individual single authentications with different guesses to $\mathrm{HW}(\mathtt{RI}^t)$. Once we have identified the correct rotation for four runs of the protocol we apply the four-run attack that simultaneously uses the information from all four authentications to recover almost all bits of the key with high probability. For example, using this method, one could mount the full

attack to recover all except two bits of the key with probability of approximately 69%, taking around 30 minutes to run on *SageMathCloud*.

4 Results and Discussion

Over the course of this paper we have seen several incremental versions of a tag, reader, and mutual authentication scheme. We have shown that none offer any substantive security. The results are summarized here, with most of the results in this paper being confirmed by implementations.

Version	Type of Attack	Net Result	# Authentications
Table 1	Passive	Tag cloning	1
Table 3	Passive	Key recovery	2
Table 4	Passive	Key recovery	8–16
Table 4	Passive (Algebraic)	Key recovery	4
Table 5	Passive (Algebraic)	Key recovery	4

We have concentrated on uni-directional authentication but mutual authentication consists of two inter-leaved versions of a tag and interrogator authentication. This means that all attacks will also apply to mutual authentication, but often with less effort since twice as much information is leaked during each protocol run. For all but the first variant the long term `K` key can be recovered.

For completeness we note that there have also been a proposal for a method to provide a secure channel, but first versions of the encryption method were wholly insecure. It might also be noted that there is some ambiguity at times as to whether protocol variables should be considered 64-bit values or 65-bit values. However this is likely to be a case of poor notation since (i) 65-bit values make little mathematical sense and (ii) they carry a significant implementation penalty. Even if we set these objections aside, the discussion quickly becomes academic since our attacks apply in a similar fashion either way.

We note that there will be many other opportunities for the attacker; we happily recognize that our application of algebraic techniques has been straightforward. A little more analysis might yield more efficient attacks, both algebraic and non-algebraic. However, in our view the point is already made and there is little to be gained by adding other incremental attacks to the mix. Those interested at looking at things more will find other work in a similar vein by other authors, *e.g.* [11], as well as other simple proposals [18,19]. Our code is available via `www.isg.rhul.ac.uk/~ccid/publications/iso-iec-29167-15.htm`.

5 Conclusion

The schemes described in this document are intrinsically weak due to the simple operations used by the tag and the interrogator. Future versions, if based on identical principles, are unlikely to provide additional security (see Appendix A).

In fairness, it should be noted that the combination of exclusive-or, integer addition, and bitwise rotation *can* be a good basis for the design of a secure primitive. So-called ARX designs are rightly popular and they featured prominently in the NIST SHA-3 initiative [17]. However these are typically multi-round algorithms while the most advanced variant of the schemes under analysis has probably just achieved a "single round" of computational complexity (if one can make the analogy).

The project editors for 29167-15 motivate their use of the simplest operations by stating that this will result in a low-area solution. However this is misguided. The bulk of the area for an implementation comes from the cryptographic state which is governed by the size of the variables. So even though ISO/IEC 29167-15 uses simple operations it doesn't lead to a dramatic implementation advantage. More importantly, there are already very good cryptographic solutions for RAIN RFID tags that provide good security; the AES is one option while PRESENT [14,13] or Grain128a [1] provide different implementation trade-offs.

One goal of the paper was to demonstrate that the simple authentication schemes being considered for standardization are flawed; this helps to emphasize the contributions cryptographers can make to a variety of ISO/IEC initiatives [7]. But a second, more important, goal was to stress that cryptography for RFID does not need to be bad cryptography. The state of the art is such that well-studied standardized schemes are available and these can be deployed in even the most demanding environments.

Acknowledgements. The computations in this work were carried out using Sage [21] and *SageMathCloud*, which is supported by National Science Foundation Grant No. DMS-0821725. The figures in this work were generated using Matplotlib [12].

References

1. Ågren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: A New Version of Grain-128 with Optional Authentication. International Journal of Wireless and Mobile Computing 5(1), 48–59 (2011)
2. Bogdanov, A.A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: An ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
3. Cid, C., Murphy, S., Robshaw, M.J.B.: Algebraic Aspects of the Advanced Encryption Standard. Springer (2006)
4. Cid, C., Weinmann, R.P.: Block ciphers: algebraic cryptanalysis and Groebner bases. In: Groebner Bases, Coding, and Cryptography, pp. 307–327. Springer (2009)
5. Courtois, N.T.: Cryptanalysis of sfinks. In: Won, D.H., Kim, S. (eds.) ICISC 2005. LNCS, vol. 3935, pp. 261–269. Springer, Heidelberg (2006)
6. Courtois, N., Meier, W.: Algebraic attacks on stream ciphers with linear feedback. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 345–359. Springer, Heidelberg (2003)

7. Degabriele, J.P., Fehr, V., Fischlin, M., Gagliardoni, T., Günther, F., Azzurra Marson, G., Mittelbach, A., Paterson, K.G.: Unpicking PLAID - A Cryptographic Analysis of an ISO-standards-track Authentication Protocol. Cryptology ePrint Archive, Report 2014/728 (2014). http://eprint.iacr.org/
8. EPCglobal. EPC Radio Frequency Identity Protocols, Generation 2 UHF RFID. Specification for RFID Air Interface Protocol for Communications at 860 MHz – 960 MHz Version 1.2.0. Available via. http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2
9. EPCglobal. EPC Radio Frequency Identity Protocols, Generation 2 UHF RFID. Specification for RFID Air Interface Protocol for Communications at 860 MHz – 960 MHz Version 2.0.0. Available via. www.gs1.org/gsmp/kc/epcglobal/uhfc1g2
10. Girault, M., Poupard, G., Stern, J.: On the Fly Authentication and Signature Schemes Based on Groups of Unknown Order. Journal of Cryptology 19(4), 463–488 (2006)
11. Han, D.: Gröbner Basis Attacks on Lightweight RFID Authentication Protocols. Journal of Information Processing Systems 7(4), 691–706 (2011)
12. Hunter, J.D.: Matplotlib: A 2D graphics environment. Computing in Science & Engineering 9(3), 90–95 (2007)
13. ISO/IEC 29167-11:2014 – Information technology – Automatic identification and data capture techniques – Part 11: Crypto suite PRESENT-80 security services for air interface communications
14. ISO/IEC 29192-2:2011 – Information technology – Security techniques – Lightweight cryptography – Part 2: Block ciphers
15. ISO/IEC 29192-4:2013 – Information technology – Security techniques – Lightweight Cryptography – Part 4: Asymmetric Techniques
16. National Institute of Standards and Technology. FIPS 197: Advanced Encryption Standard, November 2001
17. National Institute of Standards and Technology. SHA-3 competition, Available via. csrc.nist.gov/groups/ST/hash/sha-3/index.html
18. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: M^2AP: A minimalist mutual-authentication protocol for low-cost RFID tags. In: Ma, J., Jin, H., Yang, L.T., Tsai, J.J.-P. (eds.) UIC 2006. LNCS, vol. 4159, pp. 912–923. Springer, Heidelberg (2006)
19. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: EMAP: An efficient mutual-authentication protocol for low-cost RFID tags. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4277, pp. 352–361. Springer, Heidelberg (2006)
20. RAIN RFID. Available via. http://www.rainrfid.org
21. Stein, W.A., et al.: Sage Mathematics Software (Version 6.3), The Sage Development Team (2014). http://www.sagemath.org

Appendix A: Obvious Variants Are Not Secure

It is our belief that the entire rationale for ISO/IEC 29167-15 is misguided. With the hope of discouraging further patches we pro-actively anticipate some modifications that might be made with the hope of increasing security. We show that none of the obvious variants provide a significantly increased level of security.

One variant would be to increase the size of all parameters, using a secret key of size $n = 64 + n'$ bits. The intention would be to increase the size of the equations systems since algebraic cryptanalytic schemes do not scale well. However this is particularly ineffective for the scheme of Table 4 since one can simply discard the n' high bits of the transmitted values and run precisely the same attack, recovering many of the lower 64 bits. We could then guess the value of the carry bit $\mathtt{a\_64}$ which would allow the remaining bits to be attacked independently. We view this kind of attack as "slicing" the problem and we will return to this below.

We can apply a similar technique to remove any advantage from increased parameters in the scheme of Table 5. The rotation $r = \mathrm{HW}(\mathtt{RI})$ will, with high probability, lie in an interval $(\frac{n}{2} - \delta, \frac{n}{2} + \delta)$ for small δ. This means we can consider a subset of equations consisting of $\mathtt{S}_0^j \ldots \mathtt{S}_{s-1}^j$ and $\mathtt{T}_r^j \ldots \mathtt{T}_{r+s-1}^j$ (and the corresponding $\mathtt{p_i}$, $\mathtt{r\_i}$, $\mathtt{a\_i}$, $\mathtt{b\_i}$, *etc.*) for some value of the rotation r. This attack requires us to guess the values of the rotations and, additionally, we must guess $\mathtt{b\_r}$. But this does not significantly affect the complexity of the attack. (In this case we do not need to guess $\mathtt{a\_0}$, however had we considered a subset of equations not including $\mathtt{S}_0^j$ we would have needed to guess another carry bit.) From two runs of the uni-directional authentication the attacker guesses the value of four bits and, for the correct rotation amount, the number of key bits recovered when using a single 16-bit or 24-bit slice in experiments is shown in Figure 3.

One strategy that may improve the attack would be to first attack a small number of bits (via the slicing attack) for several guessed values of $\mathrm{HW}(\mathtt{RI}^j)$. Then we could carry out a full attack against multiple protocol runs (simultaneously) once the correct rotation values have been identified. This way one could recover almost all the key bits without having to run the full attack many times.

An alternative modification to the protocol might be to change the value of the constant $\mathtt{C}$. We implemented an attack assuming that two runs of the uni-directional authentication protocol were observed. We implemented a 16-bit slice attack and repeated the whole set of experiments for 256 different values of $\mathtt{C}$. The 256 $\mathtt{C}$ values were built up as a single byte pattern repeated eight times. The number of key bits that was recovered in our attacks for different $\mathtt{C}$ is illustrated in Figure 4. Zero is the only value for $\mathtt{C}$ that leaked no key bits. However $00 \cdots 00$ is a particularly bad choice for $\mathtt{C}$ since this makes $\mathtt{S} = \mathtt{T}$ and forgeries are trivial. Every other choice of $\mathtt{C}$ leads to at least four key bits out of 16 being recovered. This suggests that changing the value of $\mathtt{C}$ is unlikely to improve the security of the proposed protocol.

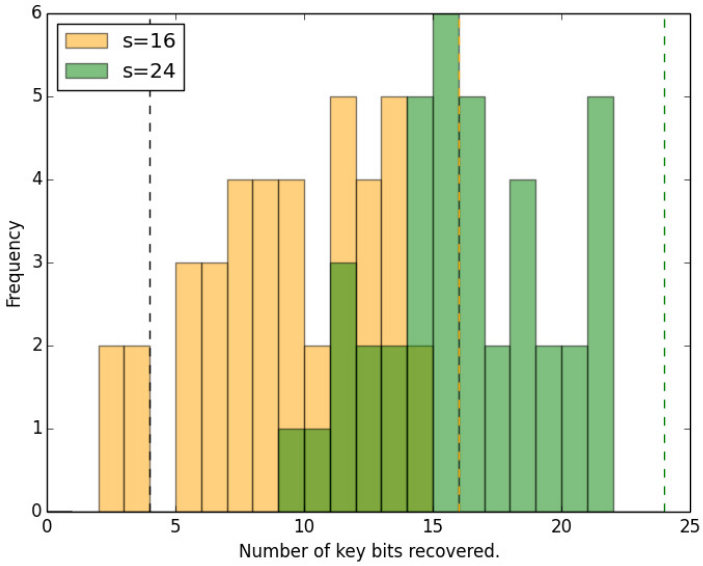

Fig. 3. Implementing the "slicing" attack against the scheme of Table 5 with slices of size s. Dashed lines (green and yellow) highlight the size of the slice used, which is an upper bound on the number of key bits that can be recovered within that slice. In both cases two runs of the uni-directional authentication scheme were used and the value of the four bits were guessed (highlighted by the black, dashed line).

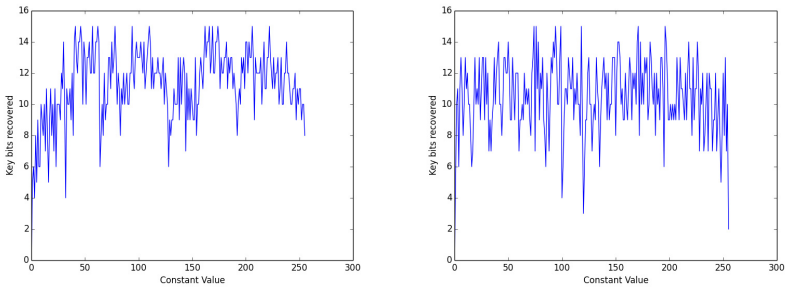

Fig. 4. A set of variants of the scheme in Table 5 were tested where the constant `C` is varied to take 256 different values (see text). Experiments were run over two instances of the uni-directional authentication scheme and the left and right charts give the results for two randomly generated keys.