Two Attacks against the F_f RFID Protocol [⋆]

Olivier Billet[1] and Kaoutar Elkhiyaoui[2]

[1] Orange Labs, Issy-les-Moulineaux, France
[2] Institut Eurécom, Valbonne, France
billet@eurecom.fr, kaoutar.elkhiyaoui@eurecom.fr

Abstract. This paper investigates a new family of RFID protocols called F_f that grew out of a proposal made at ESORICS 2007. This family has the property of having highly efficient implementations and simultaneously providing some security arguments which shares some features with the HB protocol family. In this work, we exhibit links between the F_f protocol and the LPN problem, and demonstrate two attacks against the F_f family of protocols which run with a time complexity of about 2^{52} and 2^{38} respectively against the instance proposed by the designers that has a 512-bit secret key. Our two attacks have the nice property that they only require interactions with the tag alone and does not involve the reader.

1 Introduction

Radio Frequency IDentifiers (RFID) are tiny electronic tags attached to items that allow them to be identified in an automatic way, without requiring physical access or line of sight. The main incentive to their introduction has been the ease and simplification of the supply chain management, but RFID tags already found a great variety of applications: postal tracking, tickets in transportation networks, airline luggage tracking, counterfeits fighting… The economics behind the above mentioned use-cases requires that RFID tags can be built at a very low cost, which translates into very strong design constraints for security. In particular, the memory available is very limited and the overall number of gates must be lower than a few thousand for most of the applications.

These constraints explain why the first RFID tags basically only hold a unique identifier. This however, posed a security threat as the RFID tags entered more and more into the life of end users, attached to the items they carry around. To solve these security issues, several proposals have been made, with different trade-offs between security and efficiency. As an example, forward-privacy was reached at the expense of embedding hash functions [15], whereas several authors tried to reduce the resources needed by common cryptographic primitives as much as possible: a lightweight block cipher, PRESENT, was proposed in [4], a clever tweak of Rabin's mapping, SQUASH, was introduced in [17]. One line

[⋆] *This work has been supported in part by the French Government through the ANR project RFID-AP* of cryptographic designs that looked very promising is built around the problem of learning parity with noise (LPN) and was initiated by the introduction of the HB protocol [10]. But reaching high security requirements proved to be hard as shown by the cryptanalysis of the members of this family. The HB protocol is secure against passive attackers, but fails against a simple active attack. The HB^+ protocol introduced in [11] corrected this but succumbed a more subtle active attack [7]. Almost every other proposals were flawed in some way [8], and the most robust proposal to date might be [9].

Although the simplicity of protocols from the HB family and the fact that they build on the LPN hard problem make them very attractive, they have the main drawback of requiring quite long secret keys to be able to reach a given level of security. Some alternatives to the HB family have recently appeared. One of these was introduced by Chichoń, Klonowski, and Kutyłowski in [6] where the secret consists in the knowledge of linear subspaces, but this proposal has been recently broken [12]. Another recent proposal that shares some features with HB-like protocols is the F_f protocol recently proposed in [2] which aims for an implementation that fits about 2kGE for which best known attacks require a time complexity of more than 2^{130}.

In this work, we study the security of the F_f protocol, and exhibit two key-recovery attacks on it. For the parameters chosen by the authors (512-bit secret keys) our best attack runs in time 2^{38}. Moreover, our two attacks only require to query the tag and do not need to interact with the reader. In order to explain our attacks, we first expose the LPN problem and give the best known algorithms to solve it. We then briefly describe the F_f protocol which shares some features with the HB protocol together with its main underlying building block, the f function, very similar in spirit to a universal hash function family. After this preliminary descriptions, we explain the connexions between the F_f protocol and the LPN problem. We then proceed to a study of the f function that unveils some of its properties that we use in our attacks. The first of our attacks indeed directly relies on the particularities of the function f to lower the complexity of the LPN problem underlying the F_f protocol. Our second attack relies on the existence of collisions in the random number generator used in F_f to mount a low complexity key-recovery.

2 Learning Parity with Noise

We now describe the problem of learning parity with noise (hereafter the LPN problem). To this end, let us denote the scalar product of two vectors x and y of $\mathrm{GF}(2)^n$ by $x \cdot y$. The problem of recovering a binary vector $s \in \mathrm{GF}(2)^n$ given the parity of $a \cdot s$ for randomly chosen vectors a of $\mathrm{GF}(2)^n$ is easy: given any set $\{(a_i, a_i \cdot s)\}$ where the a_i span $\mathrm{GF}(2)^n$, the value of s can be found by Gaussian elimination. In the case of LPN, the problem consists in learning the parity in the presence of noise: given enough values $(a, a \cdot s \oplus \nu)$ where a is randomly chosen and $\Pr[\nu = 1] = \epsilon$, recover the value s. The LPN problem is much more difficult and the best currently known algorithms have a time complexity of $2^{\Theta(n/\log(n))}$.

Let us denote by $x \xleftarrow{\Delta} X$ the random choice of an element x from X according to the probability distribution Δ. We also denote by \$ the uniform distribution, by Ber_ϵ the Bernouilli distribution of parameter $\epsilon \in]0, \frac{1}{2}[$, that is $\Pr[\nu = 1] = \epsilon$ and $\Pr[\nu = 0] = 1 - \epsilon$ for $\nu \xleftarrow{\mathrm{Ber}_\epsilon} \mathrm{GF}(2)$. The LPN problem can be stated more formally as follows:

Definition 1 (LPN Problem). *Let s be a vector randomly chosen from GF$(2)^n$, $\epsilon \in]0, 1/2[$ be some noise parameter, and $\mathcal{O}_{s,\epsilon}$ be an oracle that outputs independent values according to the following distribution:*

$$\left\{a \xleftarrow{\$} \mathrm{GF}(2)^n; \nu \xleftarrow{\mathrm{Ber}_\epsilon} \mathrm{GF}(2) : (a, a \cdot s \oplus \nu)\right\}$$

An algorithm A such that

$$\Pr\left[s \xleftarrow{\$} \mathrm{GF}(2)^n : A^{\mathcal{O}_{s,\epsilon}}(1^n) = s\right] \geq \delta \ ,$$

running in time at most T using at most M memory and making at most q queries to oracle $\mathcal{O}_{s,\epsilon}$ is said to (q, T, M, δ)-solve the $\mathrm{LPN}_{n,\epsilon}$ *problem.*

The LPN problem can be reformulated as the problem of decoding a random linear code, which is well-known to be NP-complete [1]. Combined to the extreme simplicity of implementation of scalar products over $\mathrm{GF}(2)^n$, this hardness makes it a problem of choice for the design of cryptographic primitives. It has served, among other cryptographic uses, as main building block of various RFID protocols designs [10, 11, 9, 5].

As stated above, the best known algorithms have a complexity of $2^{\Theta(n/\log(n))}$. The first algorithm to reach this complexity has been proposed by Blum, Kalai, and Wasserman in [3] and uses ideas similar to that put into use in the generalized birthday paradox [20]. By introducing the Walsh-transform during the last step of the BKW algorithm, Levieil and Fouque were able in [13] to give a sensible improvement of the complexity. Typical values of the complexity of the LF algorithm and stated in terms of memory sorting are given in Table 1. Finally, both algorithms given above require $2^{\Theta(n/\log(n))}$ queries. As noted in [14], it is possible to lower this number of queries to $\Theta(n)$ by first generating very low-weight linear combinations of the original set of queries; the loss of independence does not seem to have a great impact in practice [13].

3 The F_f Family of Protocols

At ESORICS 2007, a new RFID protocol was proposed [16] that relies on a lightweight function called DPM in order to perform identification. DPM is a function of degree two in the secret key and is very weak as it only involves very few of the set of possible monomials of degree two. Even more problematic is the fact that an attacker is able to access the output of the DPM function for various inputs, leading to very simple algebraic attacks [19, 18].

Table 1. Complexity of the algorithm LF1 from [13] to solve an LPN problem over vectors of n bits and with an error probability of ϵ. The table should be read in the following way: it takes 2^{130} bytes of memory to solve LPN problem with $n = 512$ and a noise parameter of $\epsilon = .49$.

$\epsilon \backslash n$	128	192	256	512	640
0.0001	13	17	21	36	44
0.2500	34	41	55	89	109
0.4375	44	53	66	105	130
0.4900	55	67	88	130	162

In order to deal with these issues, a new family [2] of lightweight functions F_f was designed, and the RFID protocol was reworked. The rationale behind the design of this protocol is to minimize the workload on the tag. To this end, it uses a lightweight function with an output of very small size instead of the usual cryptographically strong hash functions. This, however, implies colliding outputs for a large number of the secret keys; the resulting ambiguity is resolved, as usual, by using a large number of interactions. The most interesting feature of this new protocol lies in the way it prevents an attacker from having direct access to F_f's output: instead of providing the reader (and thus, the attacker) with a lightweight function of a known random R and the secret key K, it manages to keep some level of uncertainty. This calls for a parallel with the HB family of protocols [10, 11, 9, 5] where at each pass, a very simple function is used—a scalar product between R and K, but some uncertainty is ensured by adding noise.

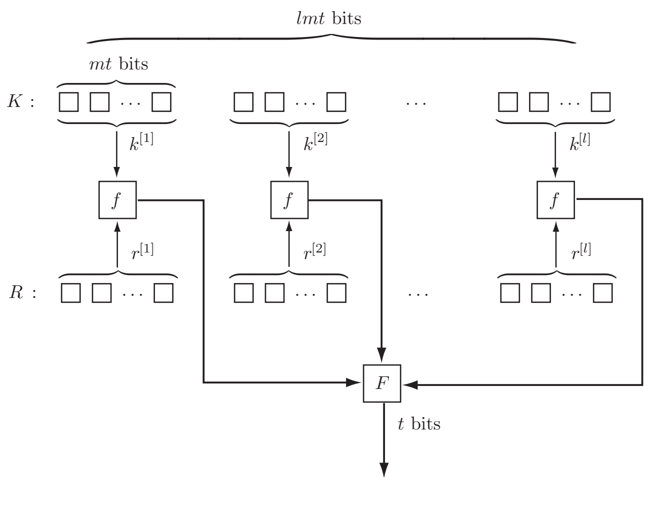

Fig. 1. The F_f function

Before describing the protocol that relies on it, let us first describe the F_f function family. The F_f function is built around a small fan-in function $f : \mathrm{GF}(2^{mt}) \times \mathrm{GF}(2^{mt}) \to \mathrm{GF}(2^t)$. The function f operates on the t-bit blocks of its mt-bit inputs. As described in Figure 1, F_f in turn operates on mt-bit blocks of lmt-bit inputs. Denoting by $x^{[i]}$ the i-th mt-bit block of any lmt-bit value X, we can define F_f as:

$$F_f(K, R) = \bigoplus_{i=1}^{l} f\left(k^{[i]}, r^{[i]}\right) \ .$$

We now turn to the description of the protocol itself. Each tag T_{ID} in the system is initialized with a pair of secret keys $(K_{\mathrm{ID}}, K'_{\mathrm{ID}})$ and the back-end system stores the corresponding tuples $(\mathrm{ID}, K_{\mathrm{ID}}, K'_{\mathrm{ID}})$ in its database. An execution of the protocol proceeds as follows:

- the reader sends a nonce $N \in \mathrm{GF}(2^{lmt})$ to the tag T_{ID};
- the tag T_{ID} replies with a seed ρ, and the following q values:

$$\begin{aligned} v_1 &= F_f(K_{\mathrm{ID}}, R_1^{a_1}) \oplus F_f(K'_{\mathrm{ID}}, N_1) \ , \\ v_2 &= F_f(K_{\mathrm{ID}}, R_2^{a_2}) \oplus F_f(K'_{\mathrm{ID}}, N_2) \ , \\ &\vdots \\ v_q &= F_f(K_{\mathrm{ID}}, R_q^{a_q}) \oplus F_f(K'_{\mathrm{ID}}, N_q) \ . \end{aligned}$$

The seed ρ is used to generate q sets $\{R_i^1, \ldots, R_i^d\}$ consisting of d random values computed by the tag using a simple LFSR. To generate the i-th value sent to the reader, the tag T_{ID} first secretly selects a number a_i in $\{1, \ldots, d\}$ and then computes

$$F_f(K_{\mathrm{ID}}, R_i^{a_i}) \oplus F_f(K'_{\mathrm{ID}}, N_i)$$

using the corresponding $R_i^{a_i}$, one out of the d random values from the i-th set. The rational behind generating the q sets of d randoms is to avoid sending them over the air and thus, to prevent an active attacker from tampering with them. (In a similar way, the tag uses the same LFSR to derive the values $N_1, \ldots, N_q$ from the nonce N.)

On the reader side, the answer of the tag is processed as follows. From the seed ρ, the reader first derives the q sets of d random values $\{R_i^1, \ldots, R_i^d\}$. Then, for each of the q received values v_i, the reader discards from its database every identity j such that:

$$\forall a \in \{1, \ldots, d\} \qquad F_f(K_j, R_i^a) \oplus F_f(K'_j, N_i) \neq v_i \ .$$

Obviously, a valid tag is never discarded as v_i is obtained at least when $a = a_i$. Additionally, if the f function is well balanced, the parameters d and t can be chosen in such a way that by increasing q, the probability of accepting invalid tag is negligible, see [2] for further details.

Since it is well known how to design identification protocols with cryptographically strong hash functions, the main advantage of the F_f protocol is to allow for highly compact implementations. As cryptographic hash functions and lightweight block ciphers currently respectively require around 7 kGE and 5 kGE, the F_f protocol targets implementations of size about 2 kGE. The practical set of parameters given in [2] is

lmt	l	m	t
256	64	1	4

d	q
8	60

and the function $f : \mathrm{GF}(2^4) \times \mathrm{GF}(2^4) \to \mathrm{GF}(2^4)$, $(r, k) \mapsto z$ is such that

$$\begin{aligned}
z_1 &= r_1k_1 \oplus r_2k_2 \oplus r_3k_3 \oplus r_4k_4 \oplus r_1r_2k_1k_2 \oplus r_2r_3k_2k_3 \oplus r_3r_4k_3k_4 \ , \\
z_2 &= r_4k_1 \oplus r_1k_2 \oplus r_2k_3 \oplus r_3k_4 \oplus r_1r_3k_1k_3 \oplus r_2r_4k_2k_4 \oplus r_1r_4k_1k_4 \ , \\
z_3 &= r_3k_1 \oplus r_4k_2 \oplus r_1k_3 \oplus r_2k_4 \oplus r_1r_2k_1k_4 \oplus r_2r_3k_2k_4 \oplus r_3r_4k_1k_3 \ , \\
z_4 &= r_2k_1 \oplus r_3k_2 \oplus r_4k_3 \oplus r_1k_4 \oplus r_1r_3k_3k_4 \oplus r_2r_4k_2k_3 \oplus r_1r_4k_1k_2 \ ,
\end{aligned} \tag{1}$$

where (r_1, r_2, r_3, r_4), (k_1, k_2, k_3, k_4), and (z_1, z_2, z_3, z_4) respectively stand for a representation of r, k, and z in $\mathrm{GF}(2)^4$. Let us also note the projection π_i from $\mathrm{GF}(2^4)$ to $\mathrm{GF}(2)$ that, according to this representation, sends any element of $\mathrm{GF}(2^4)$ to its i-th output bit: $\pi_i(z) = z_i$.

Our two attacks given below work for other values of t, but in order to ease the exposition, we focus on the choice of $t = 4$ made by the authors of F_f.

4 Preliminary Remarks for the Attacks

4.1 On the LPN problem underlying F_f.

If we discard the anti-replay nonce, the problem of recovering the key K in the F_f protocol can be seen as an LPN problem. Indeed, each of the q values v_i sent by the tag to the reader yields an equation involving some R among d possible values $R_i^1, \ldots, R_i^d$. Therefore, the attacker can always collect the equations $F_f(R_i^1, K) = v_i, \ldots, F_f(R_i^d, K) = v_i$ for $i = 1, \ldots, q$ and for several executions of the protocol. Each equation (which is defined over $\mathrm{GF}(2^t)$) can be projected over $\mathrm{GF}(2)$. Then, for each i and each execution, at least one of the d values R_i^j yields t correct boolean equations, whereas the other ones are uniformly wrong or false when F_f is well balanced, as requested by the design. Therefore, the probability that a boolean equation from the set collected by the attacker is true is $\frac{1}{d} + \frac{d-1}{d}\frac{1}{2}$. Now the equations contain terms of degree 2 in the key bits: for the parameters chosen by the authors ($t = 4$ and 256-bit keys), there are $6 \cdot 64$ monomials of degree exactly two, and $4 \cdot 64$ linear terms. Moreover, the choice of $d = 8$ yields a huge noise of $\epsilon = .4375$. Therefore, as stated in Table 1, a direct tentative to solve the LPN problem underlying F_f by linearization of the 640 monomials would have a complexity of 2^{130}.

4.2 Structure of the f function.

The f function at the core of the F_f protocol is strongly constrained, and consequently exhibits a quite specific structure. Indeed, for the protocol to be both complete and sound (i.e. reject invalid keys with overwhelming probability), the f function must be well balanced on its inputs. We now study the effect of the following function τ on the output values of f:

$$\tau : \mathrm{GF}(2^4) \rightarrow \mathrm{GF}(2) \ , \quad x \mapsto \pi_1(x) \oplus \pi_2(x) \oplus \pi_3(x) \oplus \pi_4(x) \ .$$

According to the definition (1) of f, we derive the following facts:

$$\begin{array}{ll}
\forall r \in \{\mathtt{0x0}, \mathtt{0xf}\} \ , & \tau\big(f(k,r)\big) = 0 \\
\forall r \in \{\mathtt{0x1}, \mathtt{0x2}, \mathtt{0x4}, \mathtt{0x8}\} \ , & \tau\big(f(k,r)\big) = k_1 \oplus k_2 \oplus k_3 \oplus k_4 \\
\forall r \in \{\mathtt{0x5}, \mathtt{0xc}\} \ , & \tau\big(f(k,r)\big) = k_1k_3 \oplus k_3k_4 \\
\forall r \in \{\mathtt{0x6}, \mathtt{0xa}\} \ , & \tau\big(f(k,r)\big) = k_2k_3 \oplus k_2k_4 \\
\forall r \in \{\mathtt{0x3}, \mathtt{0x9}\} \ , & \tau\big(f(k,r)\big) = k_1k_2 \oplus k_1k_4
\end{array}$$

and

$$\begin{aligned}
\tau\big(f(k,\mathtt{0xe})\big) &= (k_1 \oplus k_2 \oplus k_3 \oplus k_4) \oplus (k_1k_3 \oplus k_3k_4) \\
\tau\big(f(k,\mathtt{0xb})\big) &= (k_1 \oplus k_2 \oplus k_3 \oplus k_4) \oplus (k_2k_3 \oplus k_2k_4) \\
\tau\big(f(k,\mathtt{0xd})\big) &= (k_1 \oplus k_2 \oplus k_3 \oplus k_4) \oplus (k_1k_2 \oplus k_1k_4) \\
\tau\big(f(k,\mathtt{0x7})\big) &= (k_1 \oplus k_2 \oplus k_3 \oplus k_4) \oplus (k_1k_3 \oplus k_3k_4) \oplus (k_2k_3 \oplus k_2k_4) \oplus (k_1k_2 \oplus k_1k_4)
\end{aligned}$$

where, in order to save space, an element (z_1, z_2, z_3, z_4) of $\mathrm{GF}(2)^4$ is denoted by the corresponding nibble '$\mathtt{0x}z_1z_2z_3z_4$'. Therefore, $\tau\big(f(k,r)\big)$ is always a linear combination of the four bits c_1, c_2, c_3, and c_4 defined as

$$\begin{aligned}
c_1 &= k_1 \oplus k_2 \oplus k_3 \oplus k_4 \ , \\
c_2 &= k_1k_3 \oplus k_3k_4 \ , \\
c_3 &= k_2k_3 \oplus k_2k_4 \ , \\
c_4 &= k_1k_2 \oplus k_1k_4 \ .
\end{aligned}$$

Our two attacks against the F_f protocol both lead to a step where we have to solve for the values of $c_1, \ldots, c_4$ instead of the values of $k_1, \ldots, k_4$. Although the underlying mapping that sends k to c is not one-to-one, we will show that k can be derived from the knowledge of c and a few interactions with the system, this for a very low computational complexity.

Direct implications for the F_f protocol. It is interesting to note that the structural property of f uncovered by τ reveals a whole set of weak keys. Indeed, it is easy to check that: $\forall r, \forall k, \ \tau(f(r,k)) = \tau(f(k,r))$. As an example, if K is such that for all i, $k^{[i]} \in \{\mathtt{0x0}, \mathtt{0xf}\}$ then $\forall r, \ \tau(f(r,k)) = 0$, a property that can be easily distinguished. Also, if $k^{[i]} \in \{\mathtt{0x1}, \mathtt{0x2}, \mathtt{0x4}, \mathtt{0x8}, \mathtt{0x0}, \mathtt{0xf}\}$ for all i,

$\tau(f(r,k))$ is a linear combination of r, for any r; again, this can be distinguished. There are 2^{64} keys of the first type and $6^{64} \simeq 2^{165}$ keys of the second type.

Also, the symmetry of f with respect to r and k shows that there is a very large class of randoms N such that $F_f(K', N) = 0$. This fact can be used by an attacker to get information about $F_f(K,R)$ directly instead of through $F_f(K,R) \oplus F_f(K', N)$.

5 LPN Solving Attack

In the description of our first attack, we make use of the following property that was exhibited at the end of Section 4.2: while the tag answers with a value $F_f(K_{\text{ID}}, R_i^{a_i}) \oplus F_f(K'_{\text{ID}}, N_i)$ to the reader, the attacker—when simulating a reader—is able to choose "nonces" N (such as $N = 0$ for instance) so that $F_f(K'_{\text{ID}}, N) = 0$ for any K'_{ID}. In the following, we therefore assume without loss of generality that the tag directly answers with $F_f(K_{\text{ID}}, R_i^{a_i})$, and thus, that the attacker's goal is to recover the part K_{ID} of the tag's secret key. (We also note that once K_{ID} has been recovered, it is immediate to additionally recover K'_{ID} as the answers of the tag become deterministic in the bits of K'_{ID} and the solving complexity of a simple linearisation is negligible compared to the complexity of the rest of the attack.)

5.1 The LPN problem through τ

As we have seen in Section 4.1, it is possible to put the F_f protocol into the framework of the LPN problem. However, the protocol parameters have been chosen to escape a straightforward attack. In order to lower the complexity, we take advantage of the properties of f exhibited in Section 4.2. This requires to consider the LPN problem associated with $\tau \circ f$ instead of with f.

Let us recall that during an execution of the protocol, the tag sends q values v_i defined over GF(2^4) as $v_i = F_f(K_{\text{ID}}, R_i^{a_i})$. An attacker who collects equations of the type $\tau(v_i) = \tau\big(F_f(K_{\text{ID}}, R^a)\big)$ for every possible $a \in \{1, \ldots, d\}$ will get noisy equations on the bits of K_{ID}. What is exactly the corresponding noise ϵ? The probability that the above boolean equation is true is 1 in the case where $a = a_i$ and $\frac{1}{2}$ otherwise, so that $1 - \epsilon = \frac{1}{8} + \frac{7}{8}\frac{1}{2}$, that is $\epsilon = 0.4375$.

5.2 Lowering the complexity of the LPN problem

In order to lower the complexity of the above LPN attack, we seek to lower the number of unknowns involved, as this is the parameter having the strongest impact on the complexity. We can achieve a 25% reduction of the number of unknowns using the following fact stated in Section 4.2:

$$\begin{aligned}
&\Pr_r\big[\tau(f(k,r)) = 0\big] = \tfrac{2}{16}\ , && \Pr_r\big[\tau(f(k,r)) = c_1\big] = \tfrac{4}{16} \\
&\Pr_r\big[\tau(f(k,r)) = c_2\big] = \tfrac{2}{16}\ , && \Pr_r\big[\tau(f(k,r)) = c_3\big] = \tfrac{2}{16}\ , \\
&\Pr_r\big[\tau(f(k,r)) = c_1 \oplus c_2\big] = \tfrac{1}{16}\ , && \Pr_r\big[\tau(f(k,r)) = c_1 \oplus c_3\big] = \tfrac{1}{16}\ .
\end{aligned}$$

Indeed, the above values show that the probability over the randoms r that a 4-bit block contribution $f(k, r)$ to $F_f(K, R)$ only involves c_1, c_2, and c_3 instead of all four unknowns $c_1, \ldots, c_4$ is equal to $\mu = \frac{12}{16} \simeq 2^{-0.415}$. In order to lower the number of unknowns involved in the LPN problem from $4 \cdot 64$ bits to $3 \cdot 64$ bits, the attacker seeks a seed ρ such that at least one value among the qd randoms $R_1^1, \ldots, R_1^d, R_2^1, \ldots, R_q^d$ has all its 4-bit blocks of the requested form. This happens with probability $1 - (1 - \mu^{64})^{qd} \simeq qd\mu^{64}$. Thus, about $2^{17.6}$ interactions with the tag will give one boolean equation to solve the underlying LPN problem with noise $\epsilon = 0.4375$ on 192 unknowns. As explained in Section 2, it is enough to collect $4 \cdot 192$ equations to produce the number of samples by considering all linear combinations of weight four, yielding a total number of interactions with the tag lower than 2^{28}. As shown in Table 1, the cost for solving the LPN problem becomes about 2^{53}—to be compared to the complexity of 2^{130} of the original LPN problem.

Once the values of $c_1, \ldots, c_3$ are known, the attacker derives a new set of equations by interacting with the tag, this time removing the constraints on the initial random seed ρ chosen by the tag. This provides the attacker with a set of equations in the four values $c_1, \ldots, c_4$ for each 4-bit block in which the attacker substitutes the value of $c_1, \ldots, c_3$ just recovered: this yields another LPN problem with 64 unknowns the complexity of which is negligible compared to the complexity of the previous LPN problem. After this step, the attacker knows $c_1, \ldots, c_4$ for every 4-bit block of the key K.

5.3 Recovering the key K

At this point, the attacker gained knowledge of $c_1, \ldots, c_4$ for each of the 64 blocks of 4 bits, and thus is able to predict the value $\tau\big(f(R, K)\big)$ for any R. However, there still remains to get the value of the bits of K to be able to predict $F_f(R, K)$ and as explained in Section 4.2, the mapping from (k_1, k_2, k_3, k_4) to (c_1, c_2, c_3, c_4) is not one-to-one.

One possibility for the attacker to overcome this issue is to use her knowledge of $c_1, \ldots, c_4$ for each 4 bits block that allows her to predict with absolute certainty the value $\tau(F_f(R, K))$ *for any* R. Therefore, the attacker enters a few additional interactions with the tag (once again using nonces N_0 such that $F_f(N_0, K') = 0$). For each of the q values $v_i = F_f(R_i^a, K)$ returned by the tag during an interaction, the attacker computes the d values $b_j = \tau(F_f(R_i^j, K))$ for $j = 1, \ldots, d$. If exactly one of $\{b_1, \ldots, b_8\}$, say b_{j_0}, is equal to $\tau(v_i)$, then we know that necessarily $a = j_0$. This yields an exact equation over GF(2^4), namely $F_f(R^{j_0}, K) = v_i$, involving all the bits of K. As this event happens only when the $d - 1$ values b_j where $j \neq a$ are equal to $v_i \oplus 1$, it occurs with probability $\frac{1}{2}^{d-1}$. In order to collect N exact equations on the key bits, the attacker needs $N \frac{1}{q} 2^{d-1}$ interactions with tag, which is lower than 2^{10} for the parameters chosen by the authors of F_f (these parameters yields $N = 640$ different monomials in the key bits). The resulting system can then be solved with a complexity of $N^3 \simeq 2^{26}$.

6 A Resynchronization Attack

Contrary to the previous attack which recovers K, our second attack aims to recover K': even without the knowledge of K, the attacker is able to replay any valid execution of the protocol (this includes traces obtained when the attacker takes the role of the reader) by removing the contribution involving K' and an incorrect nonce from the trace and incorporating the correct value involving K' and the nonce challenged by the reader.

The starting point of our second attack to recover K' is the internal generator that produces the random numbers $R_1^1, \ldots, R_1^d, \ldots, R_q^1, \ldots, R_q^d$ of an execution of the protocol. As the goal of F_f is to fit under the 3kGE limit, this number generator was chosen with a 64-bit internal state. As this generator does not directly manipulate the key bits, the designers claimed that the uniformity of its output is the only constraint, and that the generator does not need to be cryptographically secure [2]:

"*We do not care about the secrecy or predictability of the internal state of PRNG, but only require (pseudo-)random properties for the Rs for statistical purposes as discussed in the next sections. Therefore, we can safely use a cheap LFSR to derive R with good enough randomness.*"

The authors therefore chose to implement it as an LFSR, but our attack only relies on its reduced entropy; it remains valid with any other pseudo-random generator with a 64-bit internal state.

6.1 Deriving noisy information on K'

The main idea of the attack to recover information about K' is to find collisions on the random seed ρ used to generate the randoms $R_1^1, \ldots, R_d^1, \ldots, R_d^q$. Indeed, the set of d randoms $\{R_i^1, \ldots, R_i^d\}$ used at the i-th round of one execution of the protocol will be identical for any two traces for which the random seeds ρ collide. As the generator producing the R_i^j has an internal state of 64 bits, it requires 2^{32} interactions with a tag to find such a collision on the seeds ρ.

Therefore, the attacker first chooses two nonces N_1 and N_2 and challenges the tag with each of these nonces 2^{32} times so that the seeds ρ will collide for an execution of the protocol involving the nonce N_1 and another execution involving the nonce N_2 about once. This way, the attacker is able to collect values $v_i^{(1)} = F_f(K, R_i^{a_{i,1}}) \oplus F_f(K', N_1)$ and $v_i^{(2)} = F_f(K, R_i^{a_{i,2}}) \oplus F_f(K', N_2)$ for $i = 1, \ldots, q$. In order to get information on K' alone, the attacker hopes that $F_f(K, R_i^{a_{i,1}}) = F_f(K, R_i^{a_{i,2}})$ so that:

$$v_i^{(1)} \oplus v_i^{(2)} = F_f(K', N_1) \oplus F_f(K', N_2) \ .$$

When the seeds ρ collide however, this equation only holds when $a_{i,1} = a_{i,2}$ or when $a_{i,1} \neq a_{i,2}$ but $F_f(K, R_i^{a_{i,1}}) = F_f(K, R_i^{a_{i,2}})$ over GF(2^t). There are d^2 possible couples $(R_i^{a_{i,1}}, R_i^{a_{i,2}})$ and the first case happens with probability $\frac{d}{d^2}$ while the second one happens with probability $\frac{1}{2^t}\left(1 - \frac{d}{d^2}\right)$ since F_f is well balanced and the R_i are randomly chosen.

6.2 Decreasing the noise and solving for K'

A major issue with the approach described above is that the equations on K' collected by the attacker are very noisy—projected over GF(2) they are true with probability $\frac{1}{8} + \frac{7}{8}\frac{1}{2} = \frac{1}{2} + \frac{1}{16}$ for the parameters chosen by the authors. As explained earlier, the number of monomials in the bits of K' that occur in $F_f(K', N)$ is 640, and Table 1 shows that trying to solve the corresponding LPN problem requires a complexity of 2^{130}.

One possibility to overcome this issue is to decrease the noise affecting the collected equation. In contrast with what happened for our first attack, it is possible to get several noisy samples of *the same equation*. Therefore, by finding several collisions on ρ, the value $v_i^{(1)} \oplus v_i^{(2)}$ obtained by the attacker is more likely to be equal to $F_f(K', N_1) \oplus F_f(K', N_2)$ than to any other value. With enough collisions, the attacker is thus able to recover the value $F_f(K', N_1) \oplus F_f(K', N_2)$ by voting for the value that appears the most often. As the analysis of such a strategy is a little bit involved over GF(2^t), we instead project each collected equation over GF$(2)^t$ and consider each of the t boolean equations independently: this yields a very lose upper-bound for the complexity of our attack.

Let us determine the probability that the majority vote for N versions of a boolean equation is correct. Recall that each boolean equation collected by the attacker is true with probability $\frac{1}{2} + \epsilon$ where $\epsilon = \frac{1}{16}$. Therefore, let us assume that the constant member is a random variable b distributed according to the probabilities $\Pr[b = 0] = \frac{1}{2} + \epsilon$ and $\Pr[b = 1] = \frac{1}{2} - \epsilon$. If we denote by b_i the constant member for the i-th version of the boolean equation, the mean value of the random variable $B = \bigoplus_{i=1}^{N} b_i$ is $(\frac{1}{2} - \epsilon)N$ and so the majority vote is wrong when $B > \frac{N}{2}$. The Chernoff bound shows that:

$$\Pr\left[B > \tfrac{N}{2}\right] < e^{-N\epsilon^2(1+2\epsilon)^{-1}} \quad .$$

To make the probability of getting a wrong equation become η, the attacker has to perform a majority vote on $N = -2\epsilon^2 \ln(\eta)$ samples. To solve the linearized system of 640 monomials, we need to get 640 correct equations, which happens with probability $(1 - \eta)^{640}$. As there are $q = 60$ rounds in one execution of the protocol, the attacker needs $\tilde{N} = 2^{32} N \frac{1}{q} (1 - \eta)^{-640}$ interactions with the tag to get a linearized system in the 640 monomials which is correct with probability greater than $\frac{1}{2}$. For the parameters chosen by the authors, setting $N = 4096$ leads to an error probability $\eta = 0.00018$, and thus to a total number of interactions with the tag of $\tilde{N} = 2^{38.4}$. The complexity to solve the linearized system is less than 2^{28} and thus negligible compared to the above complexity.

7 Conclusion

In this paper we studied the connections between the F_f RFID protocol and the LPN problem. We showed several properties of the f function underlying the F_f protocol and described two key-recovery attacks that build on these properties. In our attacks, the adversary only requires interactions with the tag and does not need to interact with the reader.

References

1. E. Berlekamp, R. McEliece, and H. van Tilborg. On the inherent intractability of certain coding problems. *Information Theory, IEEE Transactions on*, 24(3):384–386, May 1978.
2. Erik-Oliver Blaß, Anil Kurmus, Refik Molva, Guevara Noubir, and Abdullatif Shikfa. The F_f-family of protocols for RFID-privacy and authentication. In *Conference on RFID Security*, Leuven, Belgium, July 2009.
3. Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.
4. 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.
5. Julien Bringer, Hervé Chabanne, and Emmanuelle Dottax. HB^{++}: a Lightweight Authentication Protocol Secure against Some Attacks. In *Security, Privacy and Trust in Pervasive and Ubiquitous Computing—SecPerU 2006*, pages 28–33. IEEE Computer Society, 2006.
6. Jacek Cichon, Marek Klonowski, and Miroslaw Kutylowski. Privacy Protection for RFID with Hidden Subset Identifiers. In Jadwiga Indulska, Donald J. Patterson, Tom Rodden, and Max Ott, editors, *Pervasive 2008*, volume 5013 of *Lecture Notes in Computer Science*, pages 298–314. Springer, 2008.
7. H. Gilbert, M. Robshaw, and H. Sibert. Active attack against HB^{+}: a provably secure lightweight authentication protocol. *Electronics Letters*, 41(21):1169–1170, Oct. 2005.
8. Henri Gilbert, Matthew J. B. Robshaw, and Yannick Seurin. Good Variants of HB^{+} Are Hard to Find. In Gene Tsudik, editor, *Financial Cryptography and Data Security—FC 2008*, volume 5143 of *Lecture Notes in Computer Science*, pages 156–170. Springer, 2008.
9. Henri Gilbert, Matthew J.B. Robshaw, and Yannick Seurin. HB$^{\#}$: Increasing the Security and Efficiency of HB^{+}. In Nigel P. Smart, editor, *Advances in Cryptology—EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 361–378. Springer, 2008.
10. Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In Colin Boyd, editor, *Advances in Cryptology—ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 52–66. Springer, 2001.
11. 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.
12. Matthias Krause and Dirk Stegemann. More on the Security of Linear RFID Authentication Protocols. In Jr. Michael J. Jacobson, Vincent Rijmen, and Rei Safavi-Naini, editors, *SAC 2009*, volume 5867 of *Lecture Notes in Computer Science*. Springer, 2009.
13. 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.
14. Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In Chandra Chekuri, Klaus Jansen,

José D. P. Rolim, and Luca Trevisan, editors, *APPROX-RANDOM 2005*, volume 3624 of *Lecture Notes in Computer Science*, pages 378–389. Springer, 2005.

15. Miyako Ohkubo, Koutarou Suzuki, and Shingo Kinoshita. Efficient hash-chain based RFID privacy protection scheme. In *Ubiquitous Computing – Privacy Workshop*, 2004.
16. Roberto Di Pietro and Refik Molva. Information Confinement, Privacy, and Security in RFID Systems. In Joachim Biskup and Javier Lopez, editors, *Computer Security—ESORICS 2007*, volume 4734 of *Lecture Notes in Computer Science*, pages 187–202. Springer, 2008.
17. 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*, volume 5086 of *Lecture Notes in Computer Science*, pages 144–157. Springer, 2008.
18. Mate Soos. Analysing the Molva and Di Pietro Private RFID Authentication Scheme. In *Conference on RFID Security*, Budapest, Hungary, July 2008.
19. Ton van Deursen, Sjouke Mauw, and Sasa Radomirovic. Untraceability of RFID Protocols. In Jose Antonio Onieva, Damien Sauveron, Serge Chaumette, Dieter Gollmann, and Constantinos Markantonakis, editors, *Information Security Theory and Practices—WISTP 2008*, volume 5019 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2008.
20. David Wagner. A generalized birthday problem. In Moti Yung, editor, *Advances in Cryptology—CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.