The F_f-Family of Protocols for RFID-Privacy and Authentication

Erik-Oliver Blass[1], Anil Kurmus[1], Refik Molva[1], Guevara Noubir[2], and Abdullatif Shikfa[1]
[1]EURECOM, Sophia Antipolis, France and [2]Northeastern University, Boston, USA

Abstract. F_f is a family of lightweight and privacy-preserving authentication protocols for RFID-systems. Contrary to related work, F_f offers user-adjustable authentication and privacy without requiring a complex cryptographic hash function or non-volatile state on the tag. At the core of F_f is a lightweight keyed hash function that allows the reader to identify and authenticate a tag by iterative elimination of the entries in its database through a few rounds of verifications. The security of F_f is analyzed against algebraic and statistical attacks, attacks based on the LPN technique, and also with respect to recently highlighted SAT-solving approaches. The feasibility of the scheme is discussed through an estimation of the hardware cost and of the protocol performance based on a specific instance of the F_f family.

1 Introduction

Di Pietro and Molva [11] introduced a privacy-preserving authentication protocol for RFID tags called "DPM". However, the DPM-protocol suffers from some weaknesses: based on an algebraic approach, an adversary is able to compute $\frac{2}{3}$ of the secret key bits shared between reader and tag and also break the tag's privacy, cf., Soos [28], van Deursen et al. [31]. Inspired by the DPM-protocol, we propose a family of new low-cost authentication protocols for RFID tags that provide key secrecy and privacy. The protocols and underlying keyed-hashing functions are designed to withstand known attacks.
The major **contributions** of this paper are:

- We propose the new F_f family of protocols which is designed to be secure against algebraic attacks, statistical attacks, LPN attacks [21], and the recently highlighted SAT-solving approach [2]. To reason about F_f's security, we present a universal algebraic, linearization attack-framework.
- As oppposed to a common assumption in related work, e.g., [1, 11, 26, 27, 29, 34], F_f does not rely on complex cryptographic hash functions like SHA-1: F_f itself achieves the purpose of an extremely lightweight hash function. Similarly, the reader does not need to be able to compute SHA-1, but can also be resource restricted, e.g., an embedded device. Nevertheless with F_f, authentication as well as privacy can be assured with an arbitrary, user-adjustable level of security.
- Contrary to, e.g., [1, 26, 27, 29], F_f does not require the existence of a non-volatile state on tags.
- Finally, the F_f-protocols are complete, i.e., a valid, legitimate tag will *always* be identified by the reader as a valid tag – in contrast to, for example HB+ [19] and variants, where this is only guaranteed within a certain probability.

After introducing our system- and adversary model in Section 2, Section 3 presents an overview of the F_f-family of privacy-preserving authentication protocols as well as one example instance, F_{f_Δ}, that we propose. The rest of the paper will argue for the security of F_f and F_{f_Δ} in particular: Section 4 discusses statistical properties, Section 5 discusses algebraic properties of F_f including resistance against algebraic attacks, LPN attacks, and SAT-solving attacks.

2 System Model and Assumptions

An RFID-"system" consists of n tags and a single reader. For the sake of simplicity, we call a tag T_{ID}, i.e., T_{ID} is the unique name or ID of a tag. Each tag T_{ID} shares a different secret, e.g., a key $K_{T_{\mathrm{ID}}}$, with the reader. The reader stores n different tuples $(T_{\mathrm{ID}}, K_{T_{\mathrm{ID}}})$ as entries in its database $\mathbb{D}$, $n = |\mathbb{D}|$. For better comparison, we set n to a typical value, i.e., $n = 2^{16}$ as in [11].

The setup, i.e., the simple application, used in this paper is a reader in front of a closed (and locked) door. The reader will unlock and open the door, if and only if it can identify a tag $T_{\mathrm{ID}} \in \mathbb{D}$

using a communication protocol. As soon as a tag is within the reader's wireless communication range, the reader starts a protocol *run* with the tag. Here, a protocol run is a single execution of the protocol, i.e., a pass through one instance of the protocol. During the protocol run, the reader uses its database $\mathbb{D}$, to finally identify the tag's ID at the end of the protocol run.

RFID tags are severely restricted in terms of computational resources, cf., EPC Global Gen. 2 Class 1 tags [15]. They feature only a couple of thousands *Gate Equivalents* (GE) [19, 20], thus the implementation of complex hash functions like SHA-1 (requiring $10,641$ GE [5]) is impossible. Tags are read-only and assumed to be *passive*, without a battery.

As opposed to prior work, we also assume the reader to be resource restricted: in many real world application scenarios, a reader is neither permanently connected to a high-performance back-end system to forward a tag's reply for authentication to, nor does the reader feature a high-performance CPU. Instead, we assume the reader to be equipped with a microcontroller-based CPU. So, the reader is not capable of doing complex computations, e.g., SHA-1 computations, on *all elements of its database*. As a result, strong, cryptographic hash functions are not available for RFID protocol design.

2.1 Adversary model

The adversary model in this paper is based on the definitions of Vaudenay [32], referring in particular to the *non-narrow strong* adversary. We assume an active, man-in-the-middle-like adversary. The adversary can not only listen to all wireless communication between reader and tags, but also block, exchange, or modify ongoing communication. He can also temporarily put a tag into a *quality time* [30] phase, by drawing a tag into his possession. During this phase, he can query the tag a finite, reasonable number of times, i.e., send messages to and receive answers from the tag. Reasonable means that the adversary cannot exceed more operations than typical "security margins". For example, he can query only $\ll 2^{64}$ times. He can also send and receive messages to and from the reader a finite, reasonable number of times.

Theoretically, the adversary might also compromise tags, i.e., read-out their secrets, re-program them, and even destruct tags. However, we assume tags to be stateless, and in the F_f family of protocols, tags do not share any keys or secret information, not even partially. Consequently, there is no gain in compromising or destroying tags for the adversary. In the sequel of this paper, we will therefore not consider tag compromise or destruction.

2.2 Security Goals

In our RFID-system, we want to provide two security goals: (1) Authentication and (2) Privacy.

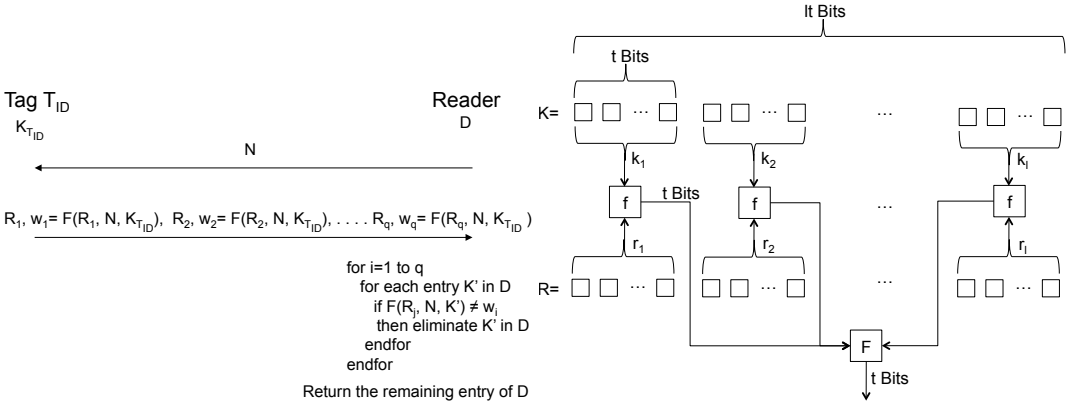
Authentication Authentication means that, after execution of the protocol, i.e., one protocol run, the reader can identify a legitimate tag with certainty. By corollary, authentication means that an intruder cannot impersonate any legitimate tag at the reader using the protocol. The adversary thus should succeed only with negligible probability in making the reader authenticate some tag $T_{\mathrm{ID}} \in \mathbb{D}$ in a protocol execution, without that T_{ID} takes part in this protocol execution. This can be called the *soundness* of authentication [10]. Similar, *completeness* means that if some tag $T_{\mathrm{ID}} \in \mathbb{D}$ takes part in a protocol run with the reader, and the adversary does not modify or block the tag's message to the reader, the reader should never reject this tag, i.e., always open the door for a valid tag.

Privacy While authentication aims at assuring the security of tag identification, privacy focuses on hiding the identity of tags from the adversary. Generally, a passively eavesdropping adversary or an active adversary exchanging messages with a tag should not be able to identify it. It must not be able to find out the ID T_{ID} of any legitimate tag. Furthermore, the adversary must not be able to trace or link tags: if the adversary eavesdrops communication or exchanges messages with any two tags T_{ID} and $T_{\mathrm{ID}'}$ on two different occasions, he must not be able to tell whether $T_{\mathrm{ID}} = T_{\mathrm{ID}'}$.

3 A Family of Privacy-Preserving RFID Authentication Protocols

The scope of this paper is a family of RFID protocols that allow for the identification of tags by readers in a privacy-preserving manner. The basic idea behind the *family* or *framework* of protocols we focus on is described in the following. (1) A tag T_{ID} provides the reader with a series of *one-way results* computed over its key $K_{T_{\mathrm{ID}}}$. (2) The reader compares these one-way results with the entries of its database $\mathbb{D}$: using the key included in each entry, the reader identifies the entry in its database whose series of one-way results matches all the one-way results received by the tag.

The reason for such a setup is to keep the complexity for tag and reader low while still trying to make the reader quickly "converge" to a single entry in its database. Instead of one pass through the whole database $\mathbb{D}$ with a very expensive hash function, our scheme is based on multiple passes ("rounds") on a database of decreasing size with a lightweight hash function.



(a) Round-based tag identification (b) Computation of $F_f(K, R)$

Fig. 1. F_f protocol scheme

3.1 Overview: Round-Based Identification

Figure 1(a) depicts a typical message flow based on this protocol framework. In the first protocol message, the reader transmits the random challenge N as required for replay protection. In the second message, the tag T_{ID} replies with q pairs (R_i,w_i) whereby (R_i) is a random number and $w_i := F(R_i, N, K_{T_{\mathrm{ID}}})$ is the result of a one-way function computed over R_i, N, and the tag's secret identification key $K_{T_{\mathrm{ID}}}$. We call each pair $(R_i, w_i = F(R_i, N, K_{T_{\mathrm{ID}}}))$ a *round* in this context. In order to identify the tag, the reader computes $w_i' := F(R_i, N, K')$ using the identification key K' of tag T' included in each tag's entry of its database. The entry (T', K') in $\mathbb{D}$ for which the received values w_i and the one computed by the reader w_i' matches for $1 \leq i \leq q$ yields the tag's identity.

As it is the core of this family of protocols, function F has to fulfill some critical requirements:

1.) Efficiency: F must be less complex than a strong hash function, because if F were comparable to a hash function, there would not be an advantage over simple hash-based authentication protocols.

2.) Security and Privacy: even though F discloses some information about the secret key of the tag as all one-way hash functions do, retrieving the key and doing *impersonation* (authenticity), identifying a tag, or guessing the link between two different protocol runs (privacy) with the same tag should be practically infeasible.

3.) Identification Rate: the received value of F and the one computed by the reader should be different with a non-negligible probability for the entries in the reader's database that do not match the tag; in other words: if $K_{T_{\mathrm{ID}}} \neq K'$, then $F(R_i, N, K_{T_{\mathrm{ID}}}) \neq F(R_i, N, K')$ with a non-negligible probability. Ideally, this probability should be close to 50% to give good identification rate and to

protect the privacy of tags. If two tags reply to one query with the same outputs or different outputs in half of the cases, the adversary does not gain any information whether the tags are the same or not.

3.2 The F_f Protocol Family

Referring to the above overview on round-based identification, we now present F_f in more detail. With F_f, reader and tag T_{ID} share not only one key $K_{T_{\mathrm{ID}}}$, but also a second key $K'_{T_{\mathrm{ID}}}$. Consequently, the reader stores tuples $(T_{\mathrm{ID}}, \{K_{T_{\mathrm{ID}}}, K'_{T_{\mathrm{ID}}}\})$ in its database $\mathbb{D}$. (Parameters $\{d, l, t\}$ mentioned below are system security parameters and will be discussed later.)

1. Each protocol run, i.e., single execution of the protocol, starts with the reader sending a nonce.
   Reader $\rightarrow$ Tag: $N_0 \in GF(2^{lt})$
2. The tag (T_{ID}) replies with a single message that is split into q *rounds* as follows:
   Tag $\rightarrow$ Reader:
   1. $(R_1^1, R_1^2, \ldots, R_1^d)$,
      $F_f(K_{T_{\mathrm{ID}}}, R_1^{a_1}) + F_f(K'_{T_{\mathrm{ID}}}, N_1)$
   2. $(R_2^1, R_2^2, \ldots, R_2^d)$,
      $F_f(K_{T_{\mathrm{ID}}}, R_2^{a_2}) + F_f(K'_{T_{\mathrm{ID}}}, N_2)$

      $\ldots$
   q. $(R_q^1, R_q^2, \ldots, R_q^d)$,
      $F_f(K_{T_{\mathrm{ID}}}, R_q^{a_q}) + F_f(K'_{T_{\mathrm{ID}}}, N_q)$,

with $R_u^v, N_u, K_{T_{\mathrm{ID}}}, K'_{T_{\mathrm{ID}}} \in GF(2^{lt})$, $a_i \in \{1 \ldots d\}$, $q \in \mathbb{N}$, $F_f : GF(2^{lt}) \times GF(2^{lt}) \rightarrow GF(2^t)$.

In every round i, a_i is chosen randomly by the tag. So, T_{ID} sends in each round i, $1 \leq i \leq q$, not only one random value R_i, but each time d random values $R_i^1, \ldots, R_i^d$. Also per round, T_{ID} randomly selects one of these values, $R_i^{a_i}$, $1 \leq a_i \leq d$ and sends $F_f(K_{T_{\mathrm{ID}}}, R_i^{a_i}) + F_f(K'_{T_{\mathrm{ID}}}, N_i)$ along with the random values to the reader. So you can see that in the F_f protocol family, $w_i = F(R_i, N, K_{T_{\mathrm{ID}}})$ of Fig. 1(a) is split into $w_i = F_f(K_{T_{\mathrm{ID}}}, R_i^{a_i}) + F_f(K'_{T_{\mathrm{ID}}}, N_i)$, and R_i of Fig. 1(a) is split into $(R_i^1, \ldots, R_i^d)$.

Reader-Side Identification of a Tag After sending N_0, the reader receives a message from T_{ID} containing q tuples $((R_i^1, \ldots, R_i^d), w_i)$. Using these tuples, the reader "strikes out" keys in $\mathbb{D}$ to eventually reduce $\mathbb{D}$ to one single key, similar to Fig. 1(a). For each i, $1 \leq i \leq q$, the reader verifies all remaining keys as follows: for the j^{th} remaining entry $(T_j, \{K_{T_j}, K'_{T_j}\}) \in \mathbb{D}$, he computes the equations:

$$
\begin{aligned}
F_f(K_{T_j}, R_i^1) + F(K'_{T_j}, N_i) &\stackrel{?}{=} w_i \\
F_f(K_{T_j}, R_i^2) + F(K'_{T_j}, N_i) &\stackrel{?}{=} w_i \\
&\ldots \\
F_f(K_{T_j}, R_i^d) + F(K'_{T_j}, N_i) &\stackrel{?}{=} w_i
\end{aligned}
$$

If and only if *all* of the above equations are invalid, the entry $(T_j, \{K_{T_j}, K'_{T_j}\})$ is removed from $\mathbb{D}$ and the reader continues with the next round $i + 1$ and the reduced database. The idea is that after q-rounds, there will be only 1 tag remaining in $\mathbb{D}$. We call this kind of identification of a single tag *converging* to a single entry. You can already see that F_f provides completeness: for data sent from a valid tag, at least one equation will always hold. Therefore, a valid tag will never be removed from $\mathbb{D}$ and never be rejected from the reader.

The reason behind not simply sending $w_i = F_f(K_{T_{\mathrm{ID}}}, R_i^{a_i})$, but instad $w_i = F_f(K_{T_{\mathrm{ID}}}, R_i^{a_i}) + F_f(K'_{T_{\mathrm{ID}}}, N_i)$ to the reader during round i is to protect against replay attacks – the tag's reply depends on the initial nonce N_0 sent by the reader. In F_f, we derive all N_i from N_0 as explained in the sequel.

Using a PRNG Sending $(R_i^1, \ldots, R_i^d)$ to the reader in every round i will generally give an adversary the opportunity to mount chosen-plaintext-attacks on the tag's key by modifying the message sent by the tag. Also, as a tag is in communication range only for a limited time, the amount of data that can be transferred is limited. Depending on system parameters q, d, l, t, this limit might be exceeded such that the tag can not authenticate itself. To overcome both problems, we therefore derive subsequent R_i^j from previous R_i^j using a pseudo-random-number-generator PRNG. More formally, we compute: $R_i^j := \mathrm{PRNG}(R_i^{j-1})$, if $j > 1$, and $R_i^1 := \mathrm{PRNG}(R_{i-1}^d)$. Now, the tag only needs to draw and send one single (real) random number, R_0^d, to the reader. This reduces the opportunity for the adversary to precisely modify subsequent Rs, as he is able to choose only the first random R_0^d. Also, data volume that is wirelessly sent to the reader shrinks from $(qd) \cdot |R|$ bits to $|R|$ bits. Still within the protocol, the tag computes all pseudo-random numbers $R_i^j, 1 \leq j \leq d$ for each round i and then (really) randomly, i.e., indeterministically, chooses one a_i and computes $w_i := F_f(K, R_i^{a_i}) + F(K', N_i)$.

In conclusion, the second protocol message, T_{ID}'s reply, looks as follows: $(R_0^d, w_1, w_2, \ldots, w_q)$. Using received R_0^d and PRNG, the reader can also compute the subsequent pseudo-random numbers.

Note: 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". Both, the tag and the reader will use R_0^d as the seed, the first internal state of the LFSR and derive subsequent R_i^j from it.

The above also holds for the N_i required for replay-protection: $N_{i+1} := \mathrm{PRNG}(N_i)$, with the original N_0 received from the reader. This also protects against chosen-plaintext attacks where the adversary would choose subsequent N.

Relation between F_f and f Furthermore in our family of protocols, F_f is made of small fan-in functions f, $f : GF(2^t) \times GF(2^t) \rightarrow GF(2^t)$, as follows: $F_f(K, R) = \sum_{i=1}^{l} f(k_i, r_i)$.

Here, "+" equals the XOR "$\oplus$". Generally, keys K and random nonces R (or N) are each of size $(l \cdot t)$ bits. Throughout this paper, we will group subsequent bits of a key or nonce into l so called *symbols*: $K = (k_1, \ldots, k_l)$, $R = (r_1, \ldots, r_l)$. Each of the l symbols consists of t bits. By writing $k_{i,j}$ or $r_{i,j}$, we denote the j^{th} bit of the i^{th} key symbol or random symbol, respectively. These relations are shown in Fig. 1(b). For the security and privacy reasoning in the following sections, let us also not consider $F_f(K, R_i^{a_i}) + F_f(K', N_i)$ in each round, but focus only on $F_f(K, R_i^{a_i})$. For the discussion on statistical and algebraic properties of F_f and its strength against attacks, this is sufficient.

3.3 Implementation Proposal: F_{f_Δ}

One suitable instance of the f function could be $f_\Delta(k_i, r_i)$ as proposed in the following. System parameters are $d = 8$, $t = 4$, $l = 64$, $q = 60$, and the LFSR for PRNG has an internal state of $\sigma = 64$ bits. The exchanged N_0 and R_0^d will be used as the LFSR's initial internal state. So, f_Δ works on inputs $k_i, r_i \in GF(2^4)$ and outputs an element in $GF(2^4)$, $f_\Delta : GF(2^4) \times GF(2^4) \rightarrow GF(2^4)$. The output of f_Δ is represented as 4 output bits, i.e., $f_\Delta(k_i, r_i) = \{f_{\Delta_1}, f_{\Delta_2}, f_{\Delta_3}, f_{\Delta_4}\}$. These output bits are separately defined as follows:

$$
\begin{aligned}
f_{\Delta_1}(k_i, r_i) := {} & r_{i,1}k_{i,1} + r_{i,2}k_{i,2} + r_{i,3}k_{i,3} + r_{i,4}k_{i,4} + r_{i,1}r_{i,2}k_{i,1}k_{i,2} \\
& + r_{i,2}r_{i,3}k_{i,2}k_{i,3} + r_{i,3}r_{i,4}k_{i,3}k_{i,4} \\
f_{\Delta_2}(k_i, r_i) := {} & r_{i,4}k_{i,1} + r_{i,1}k_{i,2} + r_{i,2}k_{i,3} + r_{i,3}k_{i,4} + r_{i,1}r_{i,3}k_{i,1}k_{i,3} \\
& + r_{i,2}r_{i,4}k_{i,2}k_{i,4} + r_{i,1}r_{i,4}k_{i,1}k_{i,4} \\
f_{\Delta_3}(k_i, r_i) := {} & r_{i,3}k_{i,1} + r_{i,4}k_{i,2} + r_{i,1}k_{i,3} + r_{i,2}k_{i,4} + r_{i,1}r_{i,2}k_{i,1}k_{i,4}
\end{aligned}
$$

$$f_{\Delta_4}(k_i, r_i) := \begin{array}{c} +r_{i,2}r_{i,3}k_{i,2}k_{i,4} + r_{i,3}r_{i,4}k_{i,1}k_{i,3} \\ r_{i,2}k_{i,1} + r_{i,3}k_{i,2} + r_{i,4}k_{i,3} + r_{i,1}k_{i,4} + r_{i,1}r_{i,3}k_{i,3}k_{i,4} \\ +r_{i,2}r_{i,4}k_{i,2}k_{i,3} + r_{i,1}r_{i,4}k_{i,1}k_{i,2} \end{array}$$

We choose f_Δ to be non-linear in both, the bits of the key symbol and the bits of the random symbol, and to hold all other required security properties as discussed in Sections 4 and 5. Also, computation of $f_\Delta(k_i, r_i)$ can be implemented quite efficiently: per output bit of f_Δ, 13 multiplications in $GF(2)$ (boolean AND) and 6 additions (boolean XOR) are required. Using figures as stated in [3], one output bit can be implemented in 34.5 GE, so f_Δ can be implemented using a total of 138 GE. To compute F_{f_Δ} out of the outputs of f_Δ, we need a 4 bit register to store temporary data, which comes in at 48 GE. The LFSR with $\sigma = 64$ bits of state requires 768 GE. We can get along with only one LFSR to derive the Rs and Ns, if we use the LFSR alternately. Therefore, we have to store both, the current R_i^d and $N_i, 1 \leq i \leq q$ of round i, in RAM. So, a total of 128 bits of RAM, i.e., 192 GE are required for this. Finally, K and K' need to be wired to f_Δ, which uses a total of 512 GE.

The above sums up to a total of $1,658$ GE. This is far less than current implementations of strong hash functions alone, e.g., SHA-1 with already $\approx 10,000$ [5], not even taking storage of the secret key into account. We clearly confirm that our computation of F_{f_Δ} with $1,658$ GE is naive, because one typically needs some kind of "multiplexing" logic around F_{f_Δ}, but we estimate the total gate count to be $\approx 2,000$ for F_{f_Δ}. In conclusion, the implementation of F_{f_Δ} is perfectly feasible with low cost RFID tags using current technology.

With an assumed data rate of ≈ 70 Kbps [12] between and EPC-class tag and a reader, the tag can send up to 70 kbit to the reader if it is in the reader's distance for ≈ 1 second. The tag's message to the reader consists of R_0^d and $w_1, \ldots, w_q$. With $\sigma = 64$, $|R_0^d| = 64$ bit, and $q \cdot t = 240$ bit are required for the w_i. In total, the tag needs to send 304 bit to the reader which, in conclusion, should be feasible for today's RFID communication.

The rest of this paper will argue for F_{f_Δ}'s security. To improve readability, we moved all mathematical proofs to Appendix 8.

4 Statistical Properties of F_f

In this section, we discuss the required properties of "low-cost" or "low-complexity" hash functions F_f to prevent classical statistical attacks and to prevent the formation of key *equivalence classes*, i.e., we discuss *key in-distinguishability* and *balanced output*.

Key Equivalence Two keys are said to be equivalent, if they can never be distinguished when hashed with any random input. To guarantee that an RFID tag can be uniquely identified and cannot be impersonated with any other tag, it is important to guarantee the non-existence of equivalence classes of keys with respect to F_f.

Theorem 1. *A hashing function F_f has no indistinguishable keys (no equivalent keys) if the underlying t bit elementary hashing function f satisfies conditions (1). More formally stated:*

$$\left\{ \begin{array}{c} \forall k_i \neq k_j \in GF(2^t) \\ \exists h_1, h_2 \in GF(2^t), h_1 \neq h_2 \\ \exists r \in GF(2^t) \textit{ s.t. } f(k_i, r) + f(k_j, r) = h_1 \\ \exists r' \in GF(2^t) \textit{ s.t. } f(k_i, r') + f(k_j, r') = h_2 \end{array} \right\} \Rightarrow \tag{1}$$

$$\forall K \neq K' \in GF(2^{lt}) \exists R \in GF(2^{lt}) \textit{ s.t.}$$
$$F(K,R) \neq F(K',R)$$

See Appendix 8 for the proof.

The f_Δ-function holds the property above. Consider any two key symbols $k \neq k' \in GF(2^4)$ of larger keys $K \neq K'$, then we will show that there exists at least one pair of random numbers r, r' such that $f_\Delta(k,r)+f_\Delta(k',r) = h_1$, $f_\Delta(k,r')+f_\Delta(k',r') = h_2$, and $h_1 \neq h_2$. If $k \neq k'$, then they differ in at least one bit, i.e., the i^{th} bit. Consider r to be $r := (0,0,0,0)$, all bits of r are zero. Looking only at the i^{th} output bit f_{Δ_i} the following equation holds, $f_{\Delta_i}(k,r)+f_{\Delta_i}(k',r) = h_{1_i} = 0$. The i^{th} bit of h_1 is 0. Consider r' to be $r'_1 := 1, r'_j := 0, j \neq 1$, so the first bit of r' is 1, the rest is 0. Looking only at the i^{th} output bit f_{Δ_i} the following equation holds, $f_{\Delta_i}(k,r') + f_{\Delta_i}(k',r') = h_{2_i} = k_i + k'_i = 1 \neq h_{1_i}$. So, $h_1 \neq h_2$. In conclusion, there are no equivalent keys in F_{f_Δ}.

Probability of (In-)Distinguishability It is quite important that the probability for which two different keys can be distinguished from each other with any R is close to 50%: on the one hand, this helps the reader to identify a tag in its database much more quickly. On the other hand, it is important to have F_f produce the same output for two different keys with 50% for each R, to protect tags' privacy: the adversary must not be able to distinguish between two tags. Let f_i denote the restriction of f to its i^{th} output bit $(1 \leq i \leq t)$, HD denotes the Hamming distance.

Theorem 2. *The set of random values for which two keys are indistinguishable is bounded as follows.*

$$\exists \delta \in [0, \tfrac{1}{2}) \forall k \neq k' \in GF(2^t)$$
$$\tfrac{1}{2} - \delta \leq Pr[f_{1\leq i\leq t}(k,r) \neq f_{1\leq i\leq t}(k',r)] \leq \tfrac{1}{2} + \delta$$
$$\Rightarrow$$
$$\forall K \neq K' \in GF(2^{lt})$$
$$\tfrac{1}{2} - (2\delta)^{t\cdot \mathrm{HD}(K,K')} \leq Pr[F(K,R) \neq F(K',R)] \leq \tfrac{1}{2} + (2\delta)^{t\cdot \mathrm{HD}(K,K')}$$

See Appendix 8 for the proof.

In conclusion, this means that with a sufficient key-length, the probability of F_f to have a different output between two keys or not is bound to 50% for any R, regardless of f. This allows the reader to eventually distinguish between two tags and "converge" during its identification process to a single tag, providing completeness.

Function f_Δ holds the left hand side of the implication in Theorem 2: looking at each output bit i, we computed the bias of output f_{Δ_i} over all 16 possible inputs to be $\leq 25\%$. So, $\delta = 0.25$. In F_{f_Δ}, with two completely random keys K, K', each of size $lt = 256$ bit, $t = 4$, the term $(2\delta)^{t\cdot \mathrm{HD}(K,K')}$ becomes negligible small, 2^{-512}, so $Pr[F(K,R) \neq F(K',R)]$ is very close to 50%.

Balanced Output Balanced output is an important statistical property that F_f needs to satisfy. Even a slight imbalance or bias in the output allows an adversary to characterize a tag based on the probability distribution of it's output. A tag can be characterized by $p_i, 1 \leq i \leq t$, i.e., the probability of outputting value 1 (or 0) at output bit i. A secure function F_f should have equal values of $p_i = \frac{1}{2}$. The family of F_f that we are considering converges towards a balanced output as the key length is increased. Similar to Yao's XOR-Lemma [18], if the underlying f provides sufficient balance for each of the bits, XORing over a large number of f outputs allows the bias to converge exponentially to 0. So, to insure output balance with high probability, we give the following sufficient condition that should hold for a non-negligible fraction of the key symbols k_i: for a given key symbol k_i, the dimension of the vector space spanned by the elements $\{f(k_i,r) | r \in GF(2^t)\}$, is equal to t.

For f_Δ, consider the 4 key symbols k_i, $k_1 = (1,0,0,0)$, $k_2 = (0,1,0,0)$, $k_3 = (0,0,1,0)$, $k_4 = (0,0,0,1)$ – they make of 25% of all key symbols. We construct the 4 random symbols $r'_{k_i}, r''_{k_i}, r'''_{k_i}, r''''_{k_i}$: with r'_{k_i}, all bits are 0, and the i^{th} bit is 1. With r''_{k_i}, all bits are 0, and the $(i^{th}-1)$ mod 4 bit is 1. With r'''_{k_i}, all bits are 0, and the $(i^{th}-2)$ mod 4 bit is 1. With r''''_{k_i}, all bits are 0, and the $(i^{th}-3)$ mod 4 bit is 1. Consequently, $\{f_\Delta(k_i, r'_{k_i}), f_\Delta(k_i, r''_{k_i}), f_\Delta(k_i, r'''_{k_i}), f_\Delta(k_i, r''''_{k_i})\}$ gives, $\{(1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1)\}$, a basis spanning a $t = 4$-dimensional vector-space. With $l = 64$, F_{f_Δ}'s output is balanced.

As the random R are chosen by the tag, *indistinguishability* and *balanced output* provide **privacy** against statistical attacks: the adversary cannot statistically distinguish or characterize any two tags.

Convergence-Rate and Anti-Impersonation Security A mandatory property of F_f should be to allow the reader to *converge* to a single key in his database quickly, generally accept valid tags (*completeness*), but still prevent an adversary to do an *impersonation* attack.

As shown before, the output of F_f is balanced. So, for each tuple of random numbers $R_i^1, \ldots, R_i^d$ and hash output $F_f(K, R_i^{a_i})$ that is sent during each of the q rounds of the protocol from the tag, the probability that a key $K' \neq K$ in the reader's database $\mathbb{D}$ is removed is: $P_{\text{remove}}(t,d) := (\frac{2^t-1}{2^t})^d$.

With the original size of the database $n = |\mathbb{D}|$, the number of invalid keys $K' \neq K$ that are still *valid* after q rounds shrinks to n', i.e., every invalid key is still valid with $(1 - P_{\text{remove}}(t,d))^q$. So, $n' = (n-1)\cdot(1-P_{\text{remove}}(t,d))^q$ and therewith $n' = (n-1)\cdot(1-(1-\frac{1}{2^t})^d)^q$. With F_{f_Δ}, $n = 2^{16}$ as in [11], $n' \approx 2^{-60}$, so all invalid keys have been removed after $q = 60$ rounds. Note that with the F_f protocols a valid tag sending data to the reader will *never* be removed from $\mathbb{D}$. Thus, F_f is *complete*.

The adversary might try to *randomly* impersonate at least one tag, e.g., to open a door, by randomly impersonating any valid key in the database. The probability that he successfully impersonates, $P_{\text{success}}(t,d,q,n)$, can be computed by using the probability $P_{\text{invalid}}(t,d,q)$ that one key in $\mathbb{D}$ is *not* valid after q rounds of sending random data, i.e., it fails on at least one round: $P_{\text{invalid}}(t,d,q) := 1-(1-P_{\text{remove}}(t,d))^q = 1-(1-(1-\frac{1}{2^t})^d)^q$, and finally $P_{\text{success}}(t,d,q,n) := 1-P_{\text{invalid}}(t,d,q)^n$. With F_{f_Δ}, $P_{\text{success}} \approx 2^{-64}$, so statistical impersonation is unlikely.

5 Algebraic Properties of F_f

In this section, we present a framework for algebraic reasoning about the F_f family of protocols. We use this to show why the F_f-family can withstand typical algebraic linearization attacks.

First, we show that any t bit output keyed-hashing function f can be linearized using a potentially larger, linearized key. A careful analysis of the dimension of the vector space generated with the coefficients of the linearized form allows a compaction of the key. This result can be generalized to the larger class of F_f keyed-hashing functions. Therefore, any instance of F_f is equivalent to an inner-product of a random-vector and a key-vector each of at most l symbols length.

Key Linearization and "Expansion-Compaction" Small fan-in keyed hash functions can be *expanded-compacted*, i.e., first expanded into a linearized expression and then compacted in a smaller expression that captures all security properties of the original keyed hash function.

Theorem 3. *Let $F_f(K,R)$ be a t bit keyed-hashing function $(GF(2^t)\times GF(2^t))^l \rightarrow GF(2^t)$ defined as $F_f(K,R) = \sum_{i=1}^{l} f(k_i, r_i)$ where $f : GF(2^t)\times GF(2^t) \rightarrow GF(2^t)$, and $K = (k_1, \ldots, k_l)$, $R = (r_1, \ldots, r_l) \in (GF(2^t))^l$. $F_f(K,R)$ can be linearized into:*

1. *$F_f(K,R) = \sum_{j=1}^{L} ER_j EK_j$ where $L = ls$ or $L = l(s-1)+1$, $s < 2^t$, $ER_j = u(r_j) \in GF(2^t), EK_j = v(k_j) \in GF(2^t)$,*

2. *the vectors* $(ER_1, \ldots, ER_L)$ *generate a vector space of dimension* L*,*
3. *it is sufficient for an adversary to know the expanded key* $(EK_1, \ldots, EK_L)$ *to compute* $F_f(K, R)$*.*

See Appendix 8 for the proof.

$(EK_1, \ldots, EK_L)$ is called the *expanded-compacted* key of a tag's original secret key K.

By its design, looking at and expanding-compacting any output bit f_{Δ_i} adds three new monomials per key symbol which results in $s = 4 + 3 = 7$ and $L = l \cdot s = 448$.

Algebraic attacks on F_f authentication protocols As of Theorem 3, we now can (without loss of generality) focus on *linearized* F_f functions, i.e., *any* function can be rewritten as the dot product $K \cdot R$, $F_f(K, R) = K \cdot R$, with $K, R \in GF(2^{Lt})$. Let K, R be to such vectors of L symbols of t bits each. K denotes a key and R a random input. $(K \cdot R) \in GF(2^t)$ denotes the dot product. Algebraic attacks now work as follows: an adversary eavesdropping to communication can derive the following type *and only* the following type of equations, because we are operating in a finite field:

$$\begin{array}{c} (K \cdot R_1^1 - w_1) \cdot (K \cdot R_1^2 - w_1) \cdots (K \cdot R_1^d - w_1) = 0 \\ (K \cdot R_2^1 - w_2) \cdot (K \cdot R_2^2 - w_2) \cdots (K \cdot R_2^d - w_2) = 0 \\ \vdots \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ (K \cdot R_q^1 - w_q) \cdot (K \cdot R_q^2 - w_q) \cdots (K \cdot R_q^d - w_q) = 0 \end{array} \tag{2}$$

(Note that listening to more than one protocol run or by initiating communication with the tag multiple times, the adversary can get more than q rounds of output and derive more than q equations.)

Generally, the adversary can compute K by solving this system of equations. However, we will now show that with a careful choice of L, t, d solving this system of equations becomes computationally infeasible. Each equation of system (2) can be expanded in a sum of monomials of degree at most d. Each equation in round i can be rewritten as: $\sum_{\substack{1 \le j \le d \\ 1 \le c_1 \le c_2 \le \cdots \le c_j \le l}} C_{i,c_1 c_2 \cdots c_j} k_{c_1} k_{c_2} \cdots k_{c_j} = (w_i)^d$

The adversary linearizes monomials of degree > 1, by substituting each with a new variable, i.e., a new monomial of degree 1. We now call Γ the matrix of coefficients C_{i,c_*}, and W is the vector of $(w_i)^d$. Ordering the linearized monomials according to a lexicographic order and renaming their vector as Y, we obtain the following equation: $\Gamma \cdot Y = W$

To get the key bits K, the adversary computes Y by inverting matrix Γ. The complexity of inversion depends on the number of (linearized) monomials. Theorem 4 bounds the total number of possible monomials U. U represents the number of columns (unknowns) in matrix Γ.

Theorem 4.

$$\sum_{j=1}^{d} \binom{L}{j} \le U \le \sum_{j=1}^{d} \binom{L+j-1}{j} \tag{3}$$

See Appendix 8 for the proof.

Corollary 1. *As long as* $d \le L/2$*,* U *increases exponentially with* d*. Therefore, an adversary needs an exponential number (in* d*) of equations and spends an exponential computational effort to compute the tag's key.*

See Appendix 8 for the proof.

Applying parameters of F_{f_Δ}, $l = 448, d = 8$, results in a system of linear equations with $U \ge 2^{55}$ and around 2^{132} computational complexity for matrix inversion ($\approx O(U^{2.4})$) and rank computation. F_{f_Δ} therefore has 132 bits of security against impersonation through matrix inversion as well as privacy attacks through rank computation (as that could allow an attacker to decide if two sets of

equations are from the same tag). So, F_{f_Δ} is safe against algebraic attacks. While there might be optimizations to such algebraic attacks, we believe they will not improve the computational complexity by a significant amount.

Learning Parity with Noise (LPN) The adversary might look at each of the t output bits of F_f, generate a Learning Parity with Noise Problem [19], and then use an efficient method to compute key bits: sending d random numbers R and *one* output(-bit) $F_f(K, R_i^{a_i})$ on one of these Rs in each round is similar to sending Rs as well as output bits that are randomly flipped with a certain bias. Yet, we are convinced that by carefully choosing an appropriate key size (lt) *and* picking a non-linear function f such as f_Δ will make attacks as in Levieil and Fouque [21] infeasible for the following reason: generally, the time- and memory complexity of these attacks rise with $2^{O(\frac{|K|}{\log|K|})}$, $|K|$ being the key size. However to apply LPN-based attacks, the adversary will have to linearize a non-linear f first. This will introduce new monomials such that the key size "virtually" increases – in the case of F_f, if you rewrite a non-linear f as a linear one, $f(k,r) = \sum_{j=1}^{s} u_j(r)v_j(k)$, as shown above, key size $|K|$ will become $|K| = l \cdot s, s < 2^t$. Given a non-linear f, this key size can become much higher than $|K| = (lt)$, which makes LPN-attacks infeasible.

With F_{f_Δ}, the LPN-bias is $\epsilon = 1 - (\frac{1}{d} + \frac{1}{2}\frac{d-1}{8}) \approx 44\%$. Linearization of f_Δ will introduce 3 new monomials per key symbol, such that $|K|$ rises to $64 \cdot (4+3) = 448$ bits. This will result in a time- and memory complexity between $\gg 2^{66}$ and $\ll 2^{130}$, cf., Levieil and Fouque [21]. So, F_{f_Δ} is secure against LPN attacks.

SAT-Solving Recently, a lot of attention has been drawn to the use of SAT-solvers in cryptography [2, 7–9, 13, 14, 22–25]. Therefore, we also attacked F_{f_Δ} using the SAT-solver MiniSat [16]. Due to space restrictions, we only present a summary of our findings, refer to the technical report [4] for full details. The basic idea of using a SAT-solver is to convert the equations an adversary can set up to Conjunctive Normal Form (CNF) and then use a SAT-solver to solve the CNF. We carefully implemented the attack by means of above publications and used the following parameters for *best performance results*. For a detailed explanation refer to Bard et al. [2], Eibach et al. [13, 14], McDonald et al. [23, 24]. The optimal *grouping* threshold was 77%, *cutting* number was 4, *shuffling* number was 16, we *overdefined* the system of equations by twice the number of unknowns.

In conclusion, MiniSat was unable to compute any key bits: already with a key size of 24 bits, MiniSat did not stop in any "reasonable" amount of time (2 hours) on an Intel Xeon at 1.87GHz. We infer from this that SAT-solving appears not to be appropriate for computing keys of HMAC-like systems where the system of equations is very *dense*, as already mentioned by Bard et al. [2]. With F_{f_Δ}, due to its balanced output, sparsity/density is around $\beta \approx 50\%$. For the same reason, *guessing* variables did not improve performance. In conclusion, F_{f_Δ} is secure against SAT-solving attacks.

6 Related Work

Many recently proposed solutions for RFID-authentication and -privacy require usage of a strong, but expensive cryptographic hash function on the tag. Also, most of these protocols have been shown to be insecure or leak privacy.

For example in [29], the tag just sends the HMAC of the reader's challenge, keyed with the pairwise secret key, back to the reader. To protect against replay attacks, challenges need to be of ascending order, otherwise the tag rejects the challenge. So in addition to an HMAC, a non-volatile state is required on the tag which, in many scenarios, might not be feasible or simply too expensive for a tag. This protocol is also prone against DoS-attacks and has been shown to leak privacy, see [20].

The protocol of [34] uses a strong and expensive hash function and an HMAC-like computation for identification of a tag. This, however, does not protect against replay attacks from the adversary: as there is no nonce from the reader involved in the protocol, an adversary receives always the same response on subsequent interactions with the same tag. This helps the adversary to identify a single tag breaking privacy, cf., [26].

Using a tree-based setup, [26] distributes $O(\log n)$ secret keys to each tag. This authentication with $O(\log n)$ complexity, i.e., "walking down" the tree of secrets until one tag is uniquely defined. Yet, besides requiring a complex hash function, the amount of memory required on a tag for this scheme might be infeasible in many scenarios. Also, privacy of this scheme is weak, as shown in [1]. Finally in contrast to F_f this scheme is not secure against tag compromise, as tags share some of the secrets of other tags. To overcome these weaknesses, [1] proposes the OSK/AO protocol using hash-chains, an idea originally proposed in [27]. Yet, OSK/AO is also known to leak privacy, cf., [20], requires an expensive hash function and a state on the tag.

With the HB+ protocol of [19], the tag XORs a biased "noise-bit" to the response before sending the response to the reader. The reader can then compute the tag's original response by solving the Learning Parity with Noise (LPN) problem. Yet, this scheme and also many variants are known to be insecure or leak privacy, cf., [17, 21]. Also note that with HB+ and all variants based on LPN-schemes [33], there will always be a potentially non-negligible probability that a valid tag gets rejected by the reader – HB+ is not complete.

7 Conclusion

This paper presented the F_f family of privacy-preserving authentication protocols. F_f uses a simple, round-based setup, where the tags send the results of evaluating random numbers using small fan-in functions to the reader. The main advantage of F_f is its extreme low cost: compared to related work, it does not require a cryptographically strong, expensive hash function. One sample instance F_{f_Δ} can be implemented on a tag using less than $2,000$ gates, yet offering 64 bit security against statistical impersonation attacks, $\gg 66$ bit against LPN, and 132 bit against algebraic attacks. Also, experiments indicate that SAT-solving attacks are computationally infeasible. Generally, F_f offers arbitrary, user-adjustable levels of soundness and identification rate, and even completeness.

Acknowledgments: The authors wish to thank Olivier Billet for pointing at and discussing Learning Parity with Noise attacks.

References

[1] G. Avoine, E. Dysli, and P. Oechslin. Reducing time complexity in rfid systems. In *Proceedings of Selected Areas in Cryptography*, pages 291–306, Kingston, Canada, 2005. ISBN 978-3-540-33108-7.

[2] G.V. Bard, N.T. Courtois, and C. Jefferson. Efficient methods for conversion and solution of sparse systems of low-degree multivariate polynomials over gf(2) via sat-solvers. In *ECRYPT workshop on Tools for Cryptanalysis*, Krakow, Poland, 2007. `http://eprint.iacr.org/2007/024/`.

[3] L. Batina, J. Lano, N. Mentens, B. Preneel, I. Verbauwhede, and S.B. Oers. Energy, performance, area versus security trade-offs for stream ciphers. In *Proceedings of ECRYPT Workshop, SASC – The State of the Art of Stream Ciphers*, pages 302–310, Brugge, Belgium, 2004.

[4] E.-O. Blass, A. Kurmus, Refik Molva, Guevara Noubir, and Abdullatif Shikfa. The F_f-family of protocols for rfid-privacy and authentication. Cryptology ePrint Archive, Report 2008/476, 2008. `http://eprint.iacr.org/2008/476.pdf`.

[5] Y. Choi, M. Kim, T. Kim, and H. Kim. Low power implementation of sha-1 algorithm for rfid system. In *Proceedings of Tenth International Symposium on Consumer Electronics*, pages 1–5, St. Petersburg, Russia, 2006. ISBN 1-4244-0216-6.

[6] Colin Cooper. On the rank of random matrices. *Random Structures and Algorithms*, 16(2):209–232, 2000. ISSN 1042-9832.

[7] N.T. Courtois and G.V. Bard. Algebraic cryptanalysis of the data encryption standard. In *Lecture Notes in Computer Science, Cryptography and Coding*, pages 152–169, 2007. ISBN 978-3-540-77271-2.

[8] N.T. Courtois, G.V. Bard, and David Wagner. Algebraic and slide attacks on keeloq. In *Fast Software Encryption*, Luxembourg City, Luxembourg, 2008. http://eprint.iacr.org/2007/062.

[9] N.T. Courtois, K. Nohl, and S. O'Neil. Algebraic attacks on the crypto-1 stream cipher in mifare classic and oyster cards, 2008. http://eprint.iacr.org/2008/166.pdf.

[10] I. Damgård and M. Østergaard. Rfid security: Tradeoffs between security and efficiency. In *Proceedings of RSA Conference*, pages 318–332, San Francisco, USA, 2006. http://eprint.iacr.org/2006/234.pdf.

[11] R. Di Pietro and R. Molva. Information confinement, privacy, and security in rfid systems. In *Lecture Notes in Computer Science, Volume 4734*, pages 187–202, 2007. ISBN 978-3-540-74834-2.

[12] D.M. Dobkin. *The RF in Rfid: Passive UHF Rfid in Practice: Passive UHF RFID in Practice*. Elsevier, 2007. ISBN 0750682094.

[13] T. Eibach, E. Pilz, and S. Steck. Comparing and optomising two generic attacks on bivium. SASC 2008 – The State of the Art of Stream Ciphers, 2008. http://www.ecrypt.eu.org/stvl/sasc2008/SASCRecord.zip.

[14] T. Eibach, E. Pilz, and G. Vlkel. Attacking bivium using sat solvers. In *International Conference on Theory and Applications of Satisfiability Testing, SAT 2008*, pages 63–76, Guangzhou, China, 2008. ISBN 978-3-540-79718-0.

[15] EPCglobal. Epcglobal standards and technology, 2008. http://www.epcglobalinc.org/standards/.

[16] N. En and N. Srensson. An extensible sat-solver. In *Proceedings of Theory and Applications of Satisfiability Testing*, pages 502–518, Santa Margherita Ligure, Italy, 2004. ISBN 978-3-540-20851-8.

[17] H. Gilbert, M. Robshaw, and H. Sibert. Active attack against hb+: a provably secure lightweight authentication protocol. *IEE Electronic Letters*, 41(21):1169–1170, 2005. ISSN 0013-5194.

[18] O. Goldreich, N. Nisan, and A. Wigderson. On yao's xor-lemma. Technical report, Electronic Colloquium on Computational Complexity, 1995. TR95–050, http://www.wisdom.weizmann.ac.il/~oded/PS/yao.ps.

[19] A. Juels and S. Weis. Authenticating pervasive devices with human protocols. In *Proceedings of Annual International Cryptography Conference*, pages 293–308, Santa Barbara, USA, 2005. ISBN 3-540-28114-2.

[20] A. Juels and S.A. Weis. Defining strong privacy for rfid. In *PerCom Workshops*, pages 342–347, White Plains, USA, 2007. ISBN 978-0-7695-2788-8.

[21] E. Levieil and P.-A. Fouque. An improved lpn algorithm. In *Proceedings of Conference on Security and cryptography for networks*, pages 348–359, Maiori, Italy, 2006. ISBN 3-540-38080-9.

[22] F. Massacci. Using walk-sat and rel-sat for cryptographic key search. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 290–295, 1999. ISBN 1-55860-613-0.

[23] C. McDonald, C. Charnes, and J. Pieprzyk. Attacking bivium with minisat, 2007. http://www.ecrypt.eu.org/stream/papersdir/2007/040.pdf.

[24] C. McDonald, C. Charnes, and J. Pieprzyk. An algebraic analysis of trivium ciphers based on the boolean satisfiability problem, 2007. http://eprint.iacr.org/2007/129.

[25] I. Mironov and L. Zhang. Applications of sat solvers to cryptanalysis of hash functions. In *Proceedings of International Conference of Theory and Applications of Satisfiability Testing – SAT 2006*, pages 102–115, Seattle, USA, 2006. ISBN 3-540-37206-7, http://eprint.iacr.org/2006/254.

[26] D. Molnar and D. Wagner. Privacy and security in library rfid: issues, practices, and architectures. In *Proceedings of Conference on Computer and Communications Security*, pages 210–219, Washington, USA, 2004. ISBN 1-58113-961-6.

[27] M. Ohkubo, K. Suzuki, and S. Kinoshita. Cryptographic approach to privacy-friendly tags. In *Proceedings of RFID Privacy Workshop*, Cambridge, USA, 2003. http://www.rfidprivacy.us/2003/agenda.php.

[28] M. Soos. Analysing the molva and di pietro private rfid authentication scheme. In *RFIDSec*, Budapest, Hungary, 2008. http://events.iaik.tugraz.at/RFIDSec08/.

[29] G. Tsudik. Ya-trap: yet another trivial rfid authentication protocol. In *Proceedings of International Conference on Pervasive Computing and Communications Workshops*, Pisa, Italy, 2006. ISBN 0-7695-2520-2.

[30] T. van Deursen and S. Radomirovic. Attacks on rfid protocols, 2008. http://eprint.iacr.org/2008/310.

[31] T. van Deursen, S. Mauw, and S. Radomirovic. Untraceability of rfid protocols. In *Proceedings of 2nd Workshop on Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks*, pages 1–15, Seville, Spain, 2008. ISBN 978-3-540-79965-8.

[32] S. Vaudenay. On privacy models for rfid. In *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security*, pages 68–87, Kuching, Malaysia, 2007. ISBN 978-3-540-76899-9.

[33] S.A. Weis. Hb+ protocol information page, 2008. http://saweis.net/hbplus.shtml.

[34] S.A. Weis, S.E. Sarma, R.L. Rivest, and D.W. Engels. Security and privacy aspects of low-cost radio frequency identification systems. In *Security in Pervasive Computing*, pages 201–212, Boppard, Germany, 2003. ISBN 3-540-20887-9.

8 Proofs of Sections 4 and 5

Theorem 1

Proof. Assume that there are two keys, K, $K' \in GF(2^{lt})$ such that $\forall R \in GF(2^{lt})$, $F(K,R) = F(K',R)$, we will show that $K = K'$. Looking at the i^{th} symbols k_i and k'_i, we can rewrite $F(K,R) = F(K',R)$:

$$\forall r_i \in GF(2^t), \forall r_j \in GF(2^t)$$
$$\left\{ \begin{array}{c} f(k_i,r_i) + f(k'_i,r_i) = \\ \sum_{j\neq i} f(k_j,r_j) + \sum_{j\neq i} f(k'_j,r_j) \end{array} \right\} \quad (4)$$

By construction, we know that for function f, **if** $k_i \neq k'_i$, then $\exists h_1, h_2, r, r' \in GF(2^t), h_1 \neq h_2$ such that $f(k_i,r) + f(k'_i,r) = h_1$ and $f(k_i,r') + f(k'_i,r') = h_2$. However, from Equation 4, this implies $\sum_{j\neq i} f(k_j,r_j) + \sum_{j\neq i} f(k'_j,r_j)$ being equal to h_1 and h_2 simultaneously, which is impossible. Therefore, k_i has to be equal to k'_i.

Theorem 2

Proof. We first consider functions with one bit output ($t = 1$). The proof is by induction on l.

Case $t = 1$**, Induction Basis (**$l = 1$**):** In this case, $K = k_1$ and $K' = k'_1$, $F(K,R) = f(k,r)$, and $F(K',R) = f(k',r)$. If $k_1 = k'_1$, the interval bounding the distinguishability probability becomes $[-\frac{1}{2}, \frac{3}{2}]$, which is always true. If $k_1 \neq k'_1$, $\mathrm{HD}(K,K')$ is equal to 1, and the theorem hypothesis gives $Pr[F(K,R) \neq F(K',R)] \in [\frac{1}{2} - \delta, \frac{1}{2} + \delta] \subseteq [\frac{1}{2} - 2\delta, \frac{1}{2} + 2\delta]$.

Case $t = 1$**, Inductive Step:** Let the **induction hypothesis** be that the theorem is true for l, and let K, K' be two keys of length $l+1$ symbols. Denote by K^l (resp. K'^l), the subkeys $(k_1, \ldots, k_l)$ and $(k'_1, \ldots, k'_l)$, respectively.

If $k_{l+1} = k'_{l+1}$**:** then $\mathrm{HD}(K,K') = \mathrm{HD}(K^l, K'^l)$ and $Pr[F(K,R) \neq F(K',R)] = Pr[F(K^l,R) \neq F(K'^l,R)]$. Since, from the induction hypothesis $Pr[F(K^l,R^l) \neq F(K'^l,R^l)] \in [\frac{1}{2} - (2\delta)^{\mathrm{HD}(K^l,K'^l)}, \frac{1}{2} + (2\delta)^{\mathrm{HD}(K^l,K'^l)}]$, then it is trivial that the desired property is true for $l+1$.

If $k_{l+1} \neq k'_{l+1}$**:** from the initial assumption about the function f, $\exists \epsilon_1 \in [-\delta, \delta]$ such that $Pr[f(k_{l+1}, r_{l+1}) \neq f(k'^{l+1}, r_{l+1})] = \frac{1}{2} + \epsilon_1$. From the induction hypothesis, $\exists \epsilon_2 \in [-(2\delta)^{\mathrm{HD}(K^l,K'^l)}, (2\delta)^{\mathrm{HD}(K^l,K'^l)}]$ such that $Pr[F(K^l,R^l) \neq F(K'^l,R^l)] = \frac{1}{2} + \epsilon_2$. We also have (from the definition of F): $Pr[F(K,R) \neq F(K',R)] = Pr[F(K^l,R^l) + f(k_{l+1},r_{l+1}) \neq F(K'^l,R) + f(k'_{l+1},r'_{l+1})]$.

Because F and f functions take only 0 and 1 values, and the r_i are independent random variables, the following holds:

$$Pr[F(K,R) \neq F(K',R)] =$$
$$Pr[F(K^l,R^l) \neq F(K'^l,R)] \cdot Pr[f(k_{l+1},r_{l+1}) = f(k'_{l+1},r_{l+1})] + Pr[F(K^l,R^l) = F(K'^l,R)]$$
$$\cdot Pr[f(k_{l+1},r_{l+1}) \neq f(k'_{l+1},r_{l+1})]$$
$$= (\tfrac{1}{2} + \epsilon_2)(\tfrac{1}{2} - \epsilon_1) + (\tfrac{1}{2} - \epsilon_2)(\tfrac{1}{2} + \epsilon_1) = \tfrac{1}{2} - 2\epsilon_1\epsilon_2$$

Since, $\epsilon_1 \in [-\delta, \delta]$, $\epsilon_2 \in [-(2\delta)^{\mathrm{HD}(K^l,K'^l)}, (2\delta)^{\mathrm{HD}(K^l,K'^l)}]$, and $\mathrm{HD}(K,K') = \mathrm{HD}(K^l,K'^l)+1$, then we obtain the final result: $Pr[F(K,R) \neq F(K',R)] \in [\frac{1}{2} - (2\delta)^{\mathrm{HD}(K,K')}, \frac{1}{2} + (2\delta)^{\mathrm{HD}(K,K')}]$.

Case $t > 1$**.** The generalization to $t > 1$ is straightforward since it is sufficient that $F(K,R)$ differs from $F(K',R)$ on at least one out of t bits of output. With t output bits, the bounding interval tightens by a power of t. Note that a tighter bound on the distinguishability probability can be obtained, if we consider the output as a single symbol of t bits.

Theorem 3

To prove Theorem 3, we first prove the following Lemma 1.

Lemma 1. *Any function $f : GF(2^t) \times GF(2^t) \rightarrow GF(2^t)$ can be linearized into:*

1. *$f(k,r) = \sum_{j=1}^{s} u_j(r)v_j(k)$ where $s \leq 2^t$, and $u_j(r)$, $v_j(k)$ are polynomials from $GF(2^t) \rightarrow GF(2^t)$,*
2. *the vectors $(u_1(r), \cdots, u_s(r))$ generate a vector space of dimension s,*
3. *it is sufficient for an adversary to know the linearized key $(v_1(k), \cdots, v_s(k))$ to compute $f(k,r)$.*

Proof. The function $f(k,r)$ can be interpolated into a multivariate polynomial of degree $(2^t - 1)$ in k and r. In case of $t = 1$, this is the same as the Algebraic Normal Form (ANF) of f. One way to make it explicit would be by Lagrange interpolation. Developing the polynomial into a sum of monomials in k and r, we obtain:

$$f(k,r) = \sum_{0 \leq i,i' \leq 2^t - 1} C_{i,i'} \cdot r^{i'} k^i \tag{5}$$

where $C_{i,i'}$ is uniquely defined by function f. This expression can be rewritten as:

$$f(k,r) = \sum_{0 \leq i \leq 2^t - 1} \Big(\sum_{0 \leq i' \leq 2^t - 1} C_{i,i'} r^{i'} \Big) \cdot k^i \tag{6}$$

This is basically the dot product $(\sum_{0 \leq i' \leq 2^t - 1} C_{i,i'} r^{i'})_{i=0 \cdots 2^t - 1} \cdot (k^i)^T_{i=0 \cdots 2^t - 1}$.

Now consider E, the space generated by vectors $V = (\sum_{0 \leq i' \leq 2^t - 1} C_{i,i'} r^{i'})_{i=0 \cdots 2^t - 1}$, whose coordinates are the coefficients of monomials k^i. V has 2^t coordinates. Let $s \leq 2^t$ be the dimension of E and $(u_1(r), \cdots, u_s(r))$ be a basis. Therefore, V can be rewritten as a linear combination of the basis vector defined by a matrix A s.t.: $V = (u_1(r), \cdots, u_s(r)) \cdot A$. We can then reformulate $f(k,r)$ as follows:

$$\begin{aligned} f(k,r) &= V \cdot (k^i)^T_{i=0 \cdots 2^t - 1} \\ &= [(u_1(r), \cdots, u_s(r)) \cdot A] \cdot (k^i)^T_{i=0 \cdots 2^t - 1} \\ &= (u_1(r), \cdots, u_s(r)) \cdot [(k^i)_{i=0 \cdots 2^t - 1} \cdot A^T]^T \end{aligned} \tag{7}$$

Let $(v_1(k), \cdots v_s(k))$ denote $[(k^i)_{i=0 \cdots 2^t - 1} \cdot A^T]$ and replacing it in (7), we then conclude that $f(k,r)$ can be compacted into a sum of s terms:

$$f(k,r) = \sum_{j=1}^{s} u_j(r)v_j(k) \tag{8}$$

where $v_j(k)$ results from the linear combination of the monomials in (5). Finally using Equation (8), it is clear knowledge of the key k is equivalent to knowledge of its expanded form $(v_1(k), \ldots, v_s(k))$.

Lemma 1 can be generalized to F_f functions composed of l functions f as follows. Note that now the dimension of the vector space V becomes (ls) or $l(s-1)+1$. Therewith, we now proof Theorem 3.

Proof. From Lemma 1, we have: $f(k,r) = \sum_{j=1}^{s} u_j(r)v_j(k)$. For a given f function, we consider the following two cases: (1) one function $v_j(k) = c$ is a constant function of k, (2) all function $v_j(k)$ are non-constants. These are the only two possible cases, because if two (or more) functions v_j, and $v_{j'}$ were constants, then: $u_j(r)v_j(k) + u_{j'}(r)v_{j'}(k) = [u_j(r)c + u_{j'}(r)c'] = u''_j(r) \cdot v''(k)$, where $v''(k)$ is a constant function of the key, and more importantly, the f would have been compacted to $(s-1)$. Therefore, at most one v_j function is a constant.

In case (1), w.l.o.g, we can assume that function $v_s(k) = c$ is the constant function. Thus,

$$F_f(K,R) = \sum_{i=1}^{l} f(k_i, r_i) = \sum_{i=1}^{l}\sum_{j=1}^{s} u_j(r_i)v_j(k_i) = \sum_{i=1}^{l}\sum_{j=1}^{s-1} u_j(r_i)v_j(k_i) + \sum_{i=1}^{l} u_s(r_i) \cdot c.$$

We can write $F_f(K,R) = \sum_{h=1}^{L} ER_h EK_h$, where $L = l(s-1)+1$, $ER_h = u_j(r_i)$ and $EK_h = v_j(k_i)$, for $h = (i-1)(s-1)+j$, and $ER_{l(s-1)+1} = \sum_{i=1}^{l} u_s(r_i) \cdot c$; $EK_{l(s-1)+1} = 1$.

In case (2),

$$F_f(K,R) = \sum_{i=1}^{l} f(k_i, r_i) = \sum_{i=1}^{l}\sum_{j=1}^{s} u_j(r_i)v_j(k_i)$$

We can write $F_f(K,R) = \sum_{h=1}^{L} ER_h EK_h$, where $L = ls$, $ER_h = u_j(r_i)$ and $EK_h = v_j(k_i)$, for $h = (i-1)s + j$.

In both cases, from the properties of $u_i(r_i)$, we can deduce that $(ER_1, \ldots, ER_L)$ generate a vector space of dimension L. Finally, knowing the values of vector $(EK_1, \ldots, EK_L)$, would allow an adversary to compute function F_f, because the ER_h are public and depend only on the publicly known random input.

Theorem 4

Proof. The number of monomials of degree $j, 1 \leq j \leq d$, where each key symbol is used at most once, e.g., $k_1 \cdots k_j$, is $\binom{L}{j}$. This number can already be reached in a field where the maximum order of an element is 1: for example in $GF(2)$. Thus, this is a lower bound.

The number of monomials of degree $j, 1 \leq j \leq d$, where each key symbol is used at most j times, e.g., k_1 is used 3 times in $k_1^3 \cdot k_2 \cdots k_{j-2}$, is $\binom{L+j-1}{j}$. In the general case, this is an upper bound.

In conclusion, we derive these bounds on U:

$$\sum_{j=1}^{d} \binom{L}{j} \leq U \leq \sum_{j=1}^{d} \binom{L+j-1}{j} \tag{9}$$

Corollary 1

Proof. A lower bound on $\binom{L}{d}$ is given by:

$$\binom{L}{d} = \prod_{i=0}^{d-1} \frac{L-i}{d-i} \geq \left(\frac{L}{d}\right)^d \tag{10}$$

(Because: $\forall i, d, L : 0 \leq i \leq d-1 < L$ implies $(L-i) \cdot d \geq (d-i) \cdot L$.)

Inequalities (9) and (10) imply $U \geq \sum_{j=1}^{d} \binom{L}{j} \geq \sum_{j=1}^{d} \left(\frac{L}{j}\right)^j$, hence the number of monomials of the system rises exponentially with d. On a side note, inverting matrix Γ will require U linearly independent observations. Assuming that the random input values R_i^j lead to a random matrix Γ, then using only a small number of extra equations, e.g., less than 5, we can obtain with overwhelming probability a maximum rank matrix Γ, cf., [6].