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

Erik-Oliver Blass, Anil Kurmus, Refik Molva, Guevara Noubir, and Abdullatif Shikfa

Abstract—In this paper, we present the design of the lightweight F_f family of privacy-preserving authentication protocols for RFID-systems. F_f results from a systematic design based on a new algebraic framework focusing on the security and privacy of RFID authentication protocols. F_f offers user-adjustable, strong authentication, and privacy against known algebraic attacks and recently popular SAT-solving attacks. In contrast to related work, F_f achieves these security properties without requiring an expensive cryptographic hash function. F_f is designed for a challenge-response protocol, where the tag sends random nonces and the results of HMAC-like computations of one of the nonces together with its secret key back to the reader. In this paper, the authentication and privacy of F_f is evaluated using analytical and experimental methods.

Index Terms—Lightweight RFID security, authentication, privacy, algebraic attacks, SAT-solving, LPN.

1 Introduction

Nowadays, Radio-Frequency-Identification (RFID) is used for a variety of applications, ranging from simple library borrowing systems and access-control to Supply-Chain-Management. The general setup consists of tiny, chip-like "tags" and "readers". Tags are wirelessly identified by the readers using some identification protocol executed by tags and readers. The pervasive use of RFID-systems, however, raises new security and privacy issues. Recently, it has been shown that currently deployed RFID-systems, e.g., London's underground ticketing system Oyster Card" or the keyless car entry system "KeeLoq" are insecure and allow fraudulent usage, cf., [1]. Further to fraudulent service usage or illegal access, the privacy of RFID users is also at risk, as RFID-tags can be wirelessly scanned and tracked. For RFID-systems to become widely accepted by industry and end-users, secure and privacy-preserving authentication protocols are thus required.

Di Pietro and Molva [2] introduced a privacy-preserving authentication protocol for RFID tags called "DPM". DPM introduces an iterative identification technique whereby the tag sends several randomized hash results of its secret key, and the reader identifies the tag by successively eliminating entries in its database that do not match the hash results. While one advantage of the DPM-protocol is reduced complexity on the reader-side during authentication, a major drawback of the DPM-protocol, and many others, lies in the requirement to evaluate a strong, cryptographic hash function, e.g., SHA-1, on the tag. Such cryptographic primitives are in general too "costly" for tags in most RFID-applications, as they alone already require $\approx$ 10,000 gate equivalents. To minimize production costs, tags are typically assumed to feature only around 1,000 to 10,000 gate equivalents available for the entire security protocol [3], [4], [5], [6], [7]. In addition, 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 [8], [9].

Inspired by the iterative identification technique introduced by DPM, we propose a family of new low-cost authentication protocols for RFID tags that provide key secrecy and privacy. The new family of protocols called F_f is secure against algebraic attacks, statistical attacks, LPN attacks [10], and the recently highlighted SAT-solving approach [11]. To reason about F_f's security, we present an universal, algebraic linearization attack-framework. We investigate this framework's theoretical and practical efficiency by exploiting weaknesses in DPM to compute secret key bits. Furthermore, the F_f family of protocols features major advantages over related approaches, as follows:

- *Low-cost tags:* As oppposed to a common assumption in related work, e.g., [2], [12], [13], [14], [15], [16], our scheme does not rely on complex cryptographic hash functions like SHA-1. This results in reduced hardware complexity and lower cost for tags. 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 and privacy can be assured with an arbitrary, user-adjustable level of security.
- *Stateless tags:* Contrary to related work, e.g., [12], [14], [15], [16], F_f does not require the existence of a nonvolatile state on tags, again resulting in cheaper tags.

- *E.-O. Blass, R. Molva, and A. Shikfa are with EURECOM, 2229 Route des cretes, BP 193, 06560 Sophia-Antipolis Cedex, France. E-mail: {Erik-Oliver.Blass, Refik.Molva, Abdullatif.Shikfa}@eurecom.fr.*
- *A. Kurmus is with the IBM Zurich Research Laboratory, Saeumerstrasse 4, 8803 Rueschlikon, Switzerland. E-mail: kur@zurich.ibm.com.*
- *G. Noubir is with the College of Computer and Information Science, West Village H (202), Northeastern University, 360 Huntington Ave., Boston, MA 02115. E-mail: noubir@ccs.neu.edu.*

Manuscript received 5 June 2009; revised 22 Dec. 2009; accepted 29 Dec. 2009; published online 6 Aug. 2010.
Recommended for acceptance by A. Juels.
For information on obtaining reprints of this article, please send e-mail to: tdsc@computer.org, and reference IEEECS Log Number TDSC-2009-06-0080.
Digital Object Identifier no. 10.1109/TDSC.2010.37.

1545-5971/11/$26.00 © 2011 IEEE Published by the IEEE Computer Society
Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

- *No false negatives:* 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+ [6] and variants, where this is only guaranteed within a certain probability.

This paper presents the F_f family of protocols in a "progressive" manner. First, we introduce the basic idea underlying the protocols, i.e., the round-based identification of tags in Section 3.1. Then, we present one instance of round-based identification using the generic F_f function in Section 3.2. A concrete implementation of F_f, called F_{f_Δ}, is presented in detail in Section 3.3. The remainder of the paper is dedicated to a detailed analysis of F_f and F_{f_Δ}. Section 4 discusses statistical properties, such as key equivalence, indistinguishability, and distribution of output. Section 5 analyzes algebraic properties with respect to specific features of F_f, such as key linearization and F_f's resistance against common algebraic attacks as well as SAT-solving and LPN attacks. Besides theoretical results, the paper evaluates the feasibility of attacks in an experimental setting in Section 5.2.2.

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 $K_{T_{\text{ID}}}$ with the reader. The reader stores n different tuples $(T_{\text{ID}}, K_{T_{\text{ID}}})$ 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 Di Pietro and Molva [2].

The setup, i.e., the simple application, used in this paper is a reader in front of locked door. The reader will unlock and open the door if and only if it can identify a tag $T_{\text{ID}} \in \mathbb{D}$ 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 [17]. They feature only a couple of thousands *Gate Equivalents* (GE) [6], [7], thus the implementation of complex hash functions like SHA-1 (requiring 10,641 GE [3]) 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. Therefore, 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 [18], 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* [19] phase, by drawing a tag into possession.

More formally, the non-narrow strong adversary of Vaudenay [18] has access to a set of oracles that can be called multiple times. He can do a DRAWTAG oracle-call. This will give him, out of the set of all possible tags, access to one tag T_{ID}, but temporarily anonymized to T_{vtag}. The oracle randomly chooses T_{ID} out of the total set of all tags. The adversary can only draw one tag at a time. Before he can draw another tag, he has to FREE the current tag. Note that with subsequent calls to DRAWTAG, the oracle might by chance choose the same tag T_{ID}. However, each time T_{vtag} given to the adversary might differ from each other.

A tag T_{vtag} that is drawn into quality time can be queried with a finite number of additional calls to oracles in any interleaved fashion. A call to LAUNCH initializes a new protocol run, i.e., resets and prepares the reader and T_{vtag} for a new execution of a protocol. Also, LAUNCH returns the first message, e.g., an initial nonce sent by the reader. With a call to SENDTAG, the adversary sends data to T_{vtag} and receives its response from the oracle. Similarly, with a call to SENDREADER, the adversary can send data to the reader and therewith finish a protocol run. Finally, if the adversary called SENDREADER, a query to the RESULT-oracle will tell the adversary, whether the reader accepted the data it received, apparently identified a tag, and opened the door, "1", or not, "0". With EXECUTE, a complete protocol instance is run through, and the adversary receives from the oracle all communication between T_{vtag} and the reader.

The adversary has access to the above oracles a finite, reasonable number of times. Reasonable means that the adversary cannot exceed more operations than typical "security margins". For example, he can query oracles $\ll 2^{64}$ times.

2.1.1 Tag Compromise and Destruction.

Theoretically, the adversary might also compromise tags, i.e., read-out their secrets, reprogram 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. If the adversary compromises (or steals) a tag, he can use this tag to open the door at a reader, and no RFID authentication protocol can protect against this. 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.

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

2.2.1 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 adversary cannot impersonate any legitimate tag at the reader. To successfully impersonate a tag, the adversary can access the above mentioned oracles a reasonable number of times.

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, and the adversary only relays communication. We call this *soundness* [20].

Similar, *completeness* means that if some valid tag $T_{\mathrm{ID}} \in \mathbb{D}$ has been returned as T_{vtag} after a call to DRAW, and the adversary simply relays data from LAUNCH, SENDTAG, and SENDREADER calls, the reader should never reject this tag, i.e., always authenticate T_{ID} and open the door.

Note that authentication, i.e., secure identification of a certain tag is a stronger requirement than completeness. Clearly, if a reader can identify a certain tag T_{ID}, it can decide whether $T_{\mathrm{ID}} \in \mathbb{D}$. Depending on the scenario, completeness might be sufficient. Yet, in this paper, F_f will provide both: completeness *and* soundness.

Also note that we do not focus on any form of *agreement* or *mutual* authentication. There is no application or need to do further communication besides protocol execution for authentication in the RFID-system.

2.2.2 Privacy

While authentication aims at protecting the tag identification, the privacy property of a protocol focuses on not revealing the identity of a tag to an adversary. Generally, an adversary should not be able to tell which tag exactly has been drawn into quality time. He must not find out the ID of some tag T_{vtag}. This can also be called anonymity. Furthermore, two subsequently drawn tags T_{vtag} and $T_{vtag'}$ must not be linked together: the adversary should not be able to decide whether $T_{vtag} = T_{vtag'}$.

If, after having access to and querying $T_{vtag}, T_{vtag'}$ one after another, the probability of the adversary doing the right decision is only negligibly higher than simple 50 percent guessing, we call the protocol private.

3 Privacy-Preserving Authentication

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 $K_{T_{\mathrm{ID}}}$ and 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 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.

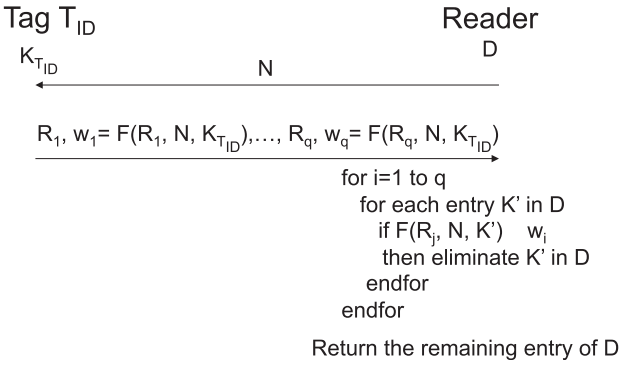

Fig. 1. Round-based tag identification.

3.1 Overview: Round-Based Identification

Fig. 1 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, $w_i := F(R_i, N, K_{T_{\mathrm{ID}}})$ is the result of a one-way function computed over R_i, N, and the tag's identification key is $K_{T_{\mathrm{ID}}}$. 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 all $\{1, \ldots, q\}$ 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 strong 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* (authentication), 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 percent 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,

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

reader and tag T_{ID} share not only one key $K_{T_{\text{ID}}}$, but also a second key $K'_{T_{\text{ID}}}$. Consequently, the reader stores tuples $(T_{\text{ID}}, \{K_{T_{\text{ID}}}, K'_{T_{\text{ID}}}\})$ in its database $\mathbb{D}$. (Parameters $\{d, l, t, q\}$ 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. $(R_1^1, R_1^2, \ldots, R_1^d)$,
      $F_f(K_{T_{\text{ID}}}, R_1^{a_1}) + F_f(K'_{T_{\text{ID}}}, N_1)$
   2. $(R_2^1, R_2^2, \ldots, R_2^d)$,
      $F_f(K_{T_{\text{ID}}}, R_2^{a_2}) + F_f(K'_{T_{\text{ID}}}, N_2)$
      $\ldots$
   q. $(R_q^1, R_q^2, \ldots, R_q^d)$,
      $F_f(K_{T_{\text{ID}}}, R_q^{a_q}) + F_f(K'_{T_{\text{ID}}}, N_q)$,

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

In every round i, a_i is chosen randomly by the tag. Therefore, 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 $F_f(K_{T_{\text{ID}}}, R_i^{a_i}) + F_f(K'_{T_{\text{ID}}}, N_i)$ along with the random values to the reader. Therefore, you can see that in the F_f protocol family, $w_i = F(R_i, N, K_{T_{\text{ID}}})$ of Fig. 1 is split into $w_i = F_f(K_{T_{\text{ID}}}, R_i^{a_i}) + F_f(K'_{T_{\text{ID}}}, N_i)$, and R_i of Fig. 1 is split into $(R_i^1, \ldots, R_i^d)$. Here, tag and reader derive N_i from the initial N_0 as explained later.

3.2.1 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. For each i, $1 \leq i \leq q$, the reader verifies all remaining keys as follows: for the jth remaining entry $(T_j, \{K_{T_j}, K'_{T_j}\}) \in \mathbb{D}$, he computes the equations:

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

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 one 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 by the reader.

3.2.2 Replay-Protection

The reason behind not simply sending $w_i = F_f(K_{T_{\text{ID}}}, R_i^{a_i})$, but $w_i = F_f(K_{T_{\text{ID}}}, R_i^{a_i}) + F_f(K'_{T_{\text{ID}}}, N_i)$ to the reader during round i is to protect against replay attacks. The reader expects w_i to depend on the original nonce N_0 sent at the beginning of the protocol run. Thus, the adversary cannot simply store the tag's response of a previous, successful protocol run using EXECUTE and replay the data during a subsequent run with SENDREADER.

3.2.3 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 answer he receives from SENDTAG and calling the SENDREADER oracle with modified data.

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 $q, d, l, and t$, this limit might be exceeded such that the tag cannot authenticate itself. To overcome both problems, we, therefore, derive subsequent R_i^j from previous R_i^j using a pseudorandom-number-generator PRNG. More formally, $R_i^j := \text{PRNG}(R_i^{j-1})$ for $j > 1$, and $R_i^1 := \text{PRNG}(R_{i-1}^d)$.

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 pseudorandom 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 F_f protocol message, T_{ID}'s reply, now 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 pseudorandom numbers.

Note. We do not care about the *secrecy* or *predictability* of the internal state of PRNG, but *only* require pseudorandom properties for the Rs for *statistical purposes*, as discussed in Sections 4 and 5. 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: $N_{i+1} := \text{PRNG}(N_i)$, with the original N_0 received from the reader. This also protects against chosen-plaintext attacks, where the adversary would choose N on subsequent calls to SENDTAG. Tag and reader will not accept $(0, \ldots, 0)$ for N_0 or R_0^d, avoiding trivial pseudorandomness.

3.2.4 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

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

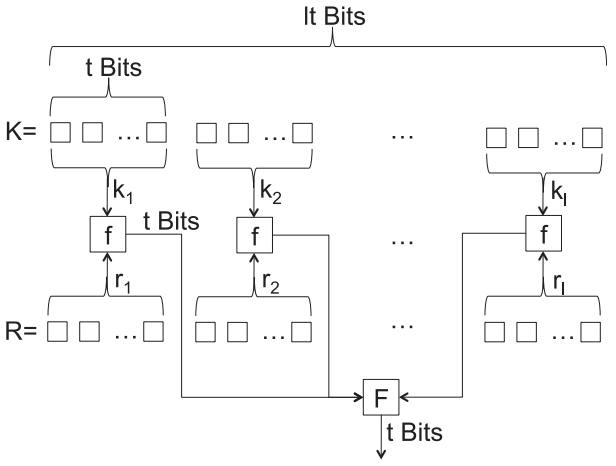

Fig. 2. Overview of $F_{f_\Delta}(K, R)$.

writing $k_{i,j}$ or $r_{i,j}$, we denote the jth bit of the ith key symbol or random symbol, respectively. These relations are shown in Fig. 2. 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. Therefore, f_Δ works on inputs $k_i, r_i \in GF(2^4)$, and outputs an element in $GF(2^4)$, $f_\Delta : GF(2^4) \times GF(2^4) \to GF(2^4)$. The output of f_Δ is represented as four 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:

$$\begin{aligned}
f_{\Delta_1}(k_i, r_i) &:= r_{i,1}k_{i,1} + r_{i,2}k_{i,2} + r_{i,3}k_{i,3} + r_{i,4}k_{i,4} \\
&\quad + r_{i,1}r_{i,2}k_{i,1}k_{i,2} + r_{i,2}r_{i,3}k_{i,2}k_{i,3} \\
&\quad + r_{i,3}r_{i,4}k_{i,3}k_{i,4}, \\
f_{\Delta_2}(k_i, r_i) &:= r_{i,4}k_{i,1} + r_{i,1}k_{i,2} + r_{i,2}k_{i,3} + r_{i,3}k_{i,4} \\
&\quad + r_{i,1}r_{i,3}k_{i,1}k_{i,3} + r_{i,2}r_{i,4}k_{i,2}k_{i,4} \\
&\quad + r_{i,1}r_{i,4}k_{i,1}k_{i,4}, \\
f_{\Delta_3}(k_i, r_i) &:= r_{i,3}k_{i,1} + r_{i,4}k_{i,2} + r_{i,1}k_{i,3} + r_{i,2}k_{i,4} \\
&\quad + r_{i,1}r_{i,2}k_{i,1}k_{i,4} + r_{i,2}r_{i,3}k_{i,2}k_{i,4} \\
&\quad + r_{i,3}r_{i,4}k_{i,1}k_{i,3}, \\
f_{\Delta_4}(k_i, r_i) &:= r_{i,2}k_{i,1} + r_{i,3}k_{i,2} + r_{i,4}k_{i,3} + r_{i,1}k_{i,4} \\
&\quad + r_{i,1}r_{i,3}k_{i,3}k_{i,4} + r_{i,2}r_{i,4}k_{i,2}k_{i,3} \\
&\quad + r_{i,1}r_{i,4}k_{i,1}k_{i,2}.
\end{aligned}$$

We choose f_Δ to be nonlinear 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 six additions (boolean XOR) are required. Using figures as stated by Batina et al. [21], one output bit can be implemented in 34.5 GE; therefore, 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 $N_i, 1 \le i \le q$ 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 ≈10,000 [3], 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 ≈2,000. Note that this gate count does not only include the F_{f_Δ} function, but also all storage for keys, nonces, and random numbers. 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 [22] between 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 sec. 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's security in general and F_{f_Δ}'s security in particular.

Note. DPM [2] can be seen as one special *instance* of F_f, i.e., $F_{f_{\text{DPM}}}$, with $d = 1, t = 3,$ and $l = 39$. However, all three output bits of f_{DPM} are the same, or simply: f_{DPM} can be seen as an output to $GF(2)$ instead of $GF(2^3)$. More precisely, f_{DPM} computes the *majority* of the bits of $k_i + r_i$:

$$\begin{aligned}
f_{\text{DPM}}(k_i, r_i) = (k_{i,1} + r_{i,1})(k_{i,2} + r_{i,2}) + (k_{i,1} + r_{i,1})(k_{i,3} + r_{i,3}) \\
+ (k_{i,2} + r_{i,2})(k_{i,3} + r_{i,3}).
\end{aligned}$$

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 protect tags' privacy: we discuss *key indistinguishability* and *balanced output*. For the sake of readability, we moved all mathematical proofs to Appendix A1.

4.1 Key Equivalence

Two keys K, K' 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 "distinguished" from other tags, it is important to guarantee the nonexistence of equivalence classes of keys with respect to F_f.

Theorem 4.1. *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:*

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

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

See Appendix A1 for the proof.

F_{f_Δ}**:** The f_Δ-function holds the aforementioned property. 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 ith bit. Consider r to be $r := (0, 0, 0, 0)$, all bits are zero. Looking only at the ith output bit f_{Δ_i}, the following equation holds, $f_{\Delta_i}(k, r) + f_{\Delta_i}(k', r) = h_{1_i} = 0$. The ith 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 ith output bit f_{Δ_i}, the following equation holds, $f_{\Delta_i}(k, r') + f_{\Delta_i}(k', r') = h_{2_i} = k_i + k'_i = 1 \neq h_{1_i}$. Therefore, $h_1 \neq h_2$. In conclusion, there are no equivalent keys in F_{f_Δ}.

4.2 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 percent: 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 percent 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 ith output bit $(1 \leq i \leq t)$ and HD denote the Hamming distance.

Theorem 4.2. *The set of random values for which two keys are indistinguishable is bounded as follows: f_i denotes the ith output bit of f.*

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

Note that the probability on the left-hand side of the implication is computed over r, while the right-hand side probability is computed over R.

See Appendix A 1 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 percent, for any R. This allows the reader to eventually distinguish between two tags and "converge" during its identification process to a single tag, providing completeness.

F_{f_Δ}**:** Function f_Δ holds the left-hand side of the implication in Theorem 4.2: looking at each output bit i, we computed the bias of output f_{Δ_i} over all 16 possible inputs to be $\leq 25\%$. Therefore, $\delta = 0.25$. In F_{f_Δ}, with two completely random keys K and K', each of size $lt = 256$ bit, $t = 4$, the term $(2\delta)^{t \cdot \mathrm{HD}(K, K')}$ becomes negligibly small, 2^{-512}; therefore, $Pr[F(K, R) \neq F(K', R)]$ is very close to 50 percent.

4.3 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 toward a balanced output, as the key length is increased. This derives from a similar argument as used in Theorem 4.2. First, consider, 1 bit output. From Theorem 4.2, it is easy to see that there is at most one key symbol k_i that leads to a constant output $f(k_i, r)$ independently of the random input. Any other key k should have a bound δ_0 on its bias: $Pr[f(k, r) = 0 \in [\frac{1}{2} - \delta_0, \frac{1}{2} + \delta_0]$ (and similarly for output 1). Let $K_i = (k_i, \ldots, k_i)$ be the l symbol repetition of k_i. Then, $Pr[F_f(K, R) = 0 \in [\frac{1}{2} - (\delta_0)^{\mathrm{HD}(K, K_i)}, \frac{1}{2} + (\delta_0)^{\mathrm{HD}(K, K_i)}]$ (and similarly for output 1). For $t > 1$, if f provides sufficient balance and independence for each of the bits, XORing over a large number of f outputs allows the bias to converge to 0. To insure output balance with high probability, we require that the following sufficient condition should hold for a non-negligible fraction of the key symbols k_i: for any 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.

F_{f_Δ}**:** Consider the four key symbols k_i, $k_1 = (1, 0, 0, 0)$, $k_2 = (0, 1, 0, 0)$, $k_3 = (0, 0, 1, 0)$, and $k_4 = (0, 0, 0, 1)$—they make of 25 percent of all 16 possible key symbols. We construct the four random symbols $r'_{k_i}, r''_{k_i}, r'''_{k_i}$, and 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, $\{f_\Delta(k_i, r'_{k_i}), f_\Delta(k_i, r''_{k_i}), f_\Delta(k_i, r'''_{k_i})$, and $f_\Delta(k_i, r''''_{k_i})\}$ give $\{(1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0)$, and $(0, 0, 0, 1)\}$, a basis spanning a $t = 4$-dimensional vector-space. With $l = 64$, F_{f_Δ}'s output is balanced.

4.4 Convergence-Rate and Anti-Impersonation

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. Therefore, 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: $P_{\mathrm{remove}}(t, d) := (\frac{2^t - 1}{2^t})^d$.

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 $(1 - P_{\mathrm{remove}}(t, d))^q$. Therefore, $n' = (n - 1) \cdot (1 - P_{\mathrm{remove}}(t, d))^q$ and therewith $n' = (n - 1) \cdot (1 - (1 - \frac{1}{2^t})^d)^q$.

Note that with the F_f protocols, a valid tag will *never* be removed from $\mathbb{D}$. Thus, F_f is *complete*.

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

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 with calls to SENDREADER and just sending random data. The probability that he successfully impersonates any tag, $P_{\text{success}}(t,d,q,n)$, 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 $P_{\text{success}}(t,d,q,n) := 1-P_{\text{invalid}}(t,d,q)^n$.

F_{f_Δ}: With F_{f_Δ}, $t=4, d=8, q=60, n=2^{16}$, so all *invalid* keys have been removed after $q=60$ rounds with high probability. Also, $P_{\text{success}} \approx 2^{-64}$, 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. The efficiency of this framework is demonstrated by successfully breaking privacy and computing secret key bits of the DPM protocol. We also use this framework to indicate why the F_f family is able to withstand typical algebraic linearization attacks. Additionally, we discuss why recent SAT-solving and LPN attacks are ineffective against F_f. We present theoretical *and* practical results.

5.1 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 capturing all security properties of the original keyed hash function.

Lemma 5.1. *Any function $f: GF(2^t)\times GF(2^t)\to GF(2^t)$ can be linearized into:*

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

See Appendix A 1 for the proof.

Lemma 5.1 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 $l(s-1)+1$ (if one $u_{j_0}(r)$ is a constant independent of r then the contribution from all the F_f subkeys can be grouped resulting in a dimension $l(s-1)+1$).

Theorem 5.2. *Let $F_f(K,R)$ be a t-bit keyed-hashing function $(GF(2^t)\times GF(2^t))^l \to GF(2^t)$ defined as $F_f(K,R)=\sum_{i=1}^{l} f(k_i,r_i)$, where $f: GF(2^t)\times GF(2^t)\to GF(2^t)$, 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. *$F_f(K,R)=\sum_{j=1}^{L} ER_jEK_j$, where $L=ls$ or $L=l(s-1)+1$, $s\le 2^t, ER_j=u(r_j)\in GF(2^t), EK_j=v(k_j)\in GF(2^t)$,*
2. *the vectors $(ER_1,\ldots,ER_L)$ generate a vector space of dimension L, and*
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 A1 for the proof.

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

5.2 Attacks on F_f Protocols

As of Theorem 5.2, 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 two 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 types of equations:

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

(Note that with multiple calls to SENDTAG, the adversary will usually get more than q rounds of output and derive more than q equations.)

The set of equations (2) holds because for any round i, at least one of the $(K\cdot R_i^j-w_1)$ is equal to zero. Any equation that an adversary can obtain from observing the tag-reader communication can be derived from (2): we are operating in a finite field; therefore for each round i, $(K\cdot R_i^1-w_i)\cdot(K\cdot R_i^2-w_i)\cdots(K\cdot R_i^d-w_i)=0$ implies that at least one $(K\cdot R_i^j)$ is equal to w_i, which completely captures all the information leaked by the authentication round and allows to derive any equations the adversary can setup.

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:

$$\sum_{\substack{1\le j\le d\\ 1\le c_1\le c_2\le\cdots\le c_j\le L}} C_{i,c_1c_2\cdots c_j}k_{c_1}k_{c_2}\cdots k_{c_j}=(w_i)^d.$$

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 the vector of $(w_i)^d$. By 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 5.3 bounds the total number of possible monomials U. Here, U represents the number of columns (unknowns) in Γ.

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

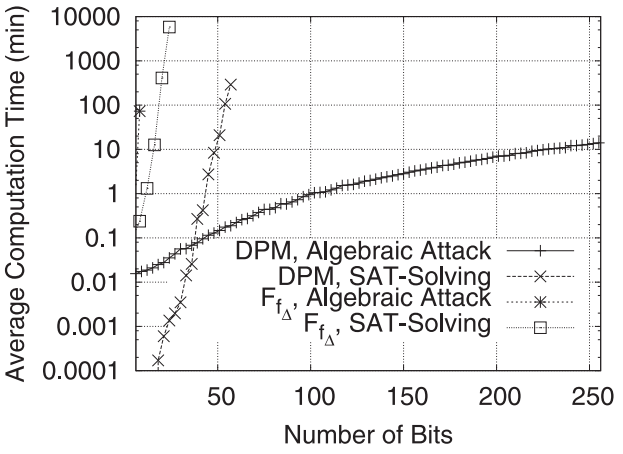

Fig. 3. Time to break privacy of DPM and F_{f_Δ}.

Theorem 5.3.

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

See Appendix A1 for the proof.

Corollary 5.4. *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 exponential computational effort to compute the tag's key.*

See Appendix A1 for the proof.

5.2.1 Algebraic and SAT-Solving Attacks Against F_{f_Δ}

For better readability, we dedicate Appendix A2 to analyze and explain all the details of F_{f_Δ}'s security with respect to algebraic attacks and also against SAT-solving attacks, and compare with DPM's security. Here, we will only give a conclusive summary: with $U \approx 2^{55}$, F_{f_Δ} offers 165-bit security against algebraic attacks. Also, SAT-solving [11] turns out to be useless against F_{f_Δ}, as the systems of equations derivable by the adversary are very "dense".

5.2.2 Experimental Evaluation

To back up our claims, we also experimentally evaluated algebraic and SAT-solving attacks against F_{f_Δ} and for comparison against F_{DPM}. Fig. 3 shows the results of algebraic and SAT-based attacks on $F_{f_{\text{DPM}}}$ (DPM) and F_{f_Δ} using an Intel Xeon, 1.86-GHz, 16-GB RAM (RAM is not an issue during the attacks, even with MiniSat). For the algebraic attack, which is mainly Gaussian-Elimination, we use MatLab, the SAT-attack uses MiniSat. The *x*-axis shows the number of key bits. The *y*-axis shows the required time (in minutes) to *break privacy*, i.e., an adversary can decide $T_{vtag} \stackrel{?}{=} T_{vtag'}$. Per sampling point, there are 20 different instances. Relative standard deviation is $<20\%$ with the $f$algebraic attack and, due to its indeterministic nature, $<80\%$ with MiniSat.

SAT Parameters: For reproducibility of our results, this paragraph lists the SAT-parameters used, see Appendix A 2.3. The optimal grouping threshold p is $p = 77\%$, the optimal cutting number is four. We also shuffle [11] each input to MiniSat 16 times, to get good performance results. Interestingly, neither guessing of variables, nor elimination of single variables have a positive impact on performance. We can only explain this by pointing at the very dense system of equations we have in $F_{f_{\text{DPM}}}$ and F_{f_Δ}, e.g., "random matrices" with $\beta \approx 50\%$, while Bard et al. [11] and Courtois and Bard [23] assume density of equations to be very low with $\beta \le 1\%$. We also *overdefine* the system of equations as proposed by Bard et al. [11], i.e., provide more than $|K| = Lt$ equations. MiniSat performs best with $2 \cdot Lt$ equations.

$F_{f_{\text{DPM}}}$: Even with 256 key bits, you can compute an (equivalent) key and break privacy of DPM in a couple of minutes using the algebraic attack. MiniSat, however, does not perform well: already with key sizes >57 bits, MiniSat does not finish in a "reasonable" amount of time (>2 h). 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. This was also mentioned by Bard et al. [11]. With DPM, sparsity/density is $\beta \approx 50\%$.

To compute all secret key bits, and not only an equivalent key, there is also the additional SHA-1 brute-forcing step of Section A2.1.3.1 (see Appendix) necessary to finally compute the "right" key. On average, 2^{l-2} SHA-1 computations are required for this step. With 117 key bits and $l = 39$, 2^{37} SHA-1 operations are necessary. On our machine running OpenSSL 0.9.8b, 2^{37} SHA-1 HMAC-operations $h(K|R_1|N|K)$ with a total $4 \cdot 117 = 468$ bits of input can be executed in around two days. This shows that computing the secret key is totally feasible with the DPM-protocol, allowing not only to break privacy, but also do impersonation.

F_{f_Δ}: As shown in Fig. 3, with key sizes >24 bits, MiniSat did not finish in any reasonable amount of time (>2 h). For the algebraic attack with 12 bits of key size, $U \approx 2^{19}$ linearized equations have to be solved with 2^{19} monomials—this does not finish in any reasonable amount of time. For a 256-bit key, there will be $U \approx 2^{55}$ monomials and equations in matrix Γ. The computational complexity of inverting this matrix is $\approx 2^{55 \cdot 3} = 2^{165}$. We claim that this will render attacks against the key to break authenticity or privacy infeasible.

Note: The setup $F_{f_\Delta}(K, R_i^{a_i}) + F_{f_\Delta}(K', N_i)$ used in this paper does not increase security against attacks on K compared to simply using $F_{f_\Delta}(K, R_i^{a_i})$. As already mentioned in Section 3.1, adding $F_{f_\Delta}(K', N_i)$ serves for replay protection only. An adversary could repeatedly use SENDTAG to send the same initial N to T_{vtag} during subsequent protocol runs. As a result, all $F_{f_\Delta}(K', N_i), 1 \le i \le q$ can be taken as constant, and do not increase the number of monomials in Γ.

5.2.3 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 [6], and then use an efficient method to compute key bits: sending d random numbers R and *one* output(-bit) of $F_f(K, R_i^{a_i})$ 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. However, we are convinced that by carefully choosing an appropriate key size (lt) *and* picking a nonlinear function f

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

such as f_Δ will make attacks, as proposed by Levieil and Fouque [10], 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 nonlinear f first. This will introduce new monomials such that the key size "virtually" increases—in the case of F_f, if you rewrite a nonlinear 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 nonlinear f, this key size can become much higher than $|K| = (lt)$, making LPN-attacks infeasible.

F_{f_Δ}: With F_{f_Δ}, the LPN-bias $\epsilon = 1 - (\frac{1}{d} + \frac{1}{2}\frac{d-1}{8}) \approx 44\%$. Linearization of f_Δ will introduce three new monomials per key symbol, so $|K|$ 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., [10]. Therefore, we are convinced that F_{f_Δ} is secure against LPN attacks.

6 Security Implications

We will briefly summarize the last section' implications on authentication and privacy.

6.1 Authentication

In conclusion, it is impossible for the adversary to fake authentication and impersonate as a valid tag. We have shown in Section 5 that the adversary cannot compute K, K' with known attacks, such as algebraic attacks, SAT-attacks, and LPN-attacks. To still successfully impersonate any tag, the adversary would need to respond to a random reader's nonce N with $q = 60$ valid $F_{f_\Delta}(K, R_i^{a_i}) + F_{f_\Delta}(K', N_i)$ pairs. Section 4 has shown that this is statistically impossible.

6.2 Privacy

As the adversary cannot compute K, he has to analyze the output of SENDTAG oracle calls to characterize tags. Here, the two properties, *indistinguishability* and *balanced output*, of Section 4 provide privacy against statistical attacks: the adversary is not able to statistically distinguish or characterize any two tags.

Still, the adversary might try to exploit algebraic properties of the SENDTAG output he sees. After querying two tags T_{vtag} and $T_{vtag'}$, he has gathered two sets of equations S_1 and S_2 of the form of (2) in Section 5.2. Assume that within each set there are λ linearly independent equations, respectively. If the adversary mixes equations between S_1 and S_2, and the resulting matrix contains results in a contradiction (i.e., the left-hand side of one equation is a linear combination of other equations, but the right-hand side does not match), he knows $T_{vtag} \neq T_{vtag'}$, otherwise he cannot conclude anything.

A necessary, but not sufficient condition for this is to find at least one linear independent equation in the mix of S_1 and S_2, which is unlikely: given that the maximum achievable rank of matrices S_1 and S_2 derived from observing the protocol execution is extremely high, cf., Section 5.3, then the probability that the $2 \cdot \lambda$ equations will be linearly independent is extremely high. For the computationally bound adversary, to be able to compute a linear dependent equation in a $2 \cdot \lambda \times U$ matrix, $2 \cdot \lambda \ll U$. Yet, already a $(n-5) \times n$ matrix has high probability rank $(n-5)$, so the $2 \cdot \lambda \times U$ matrix has rank $2 \cdot \lambda$ with high probability, see [24]. Therefore, the adversary cannot reach a conclusion on the relation between S_1 and S_2.

7 Related Work

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

For example, Tsudik [12] 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. Therefore, in addition to an HMAC, a nonvolatile state is required on the tag which, in many scenarios, might not be feasible or simply too expensive for a tag. This protocol is prone against DoS-attacks and has been shown to leak privacy [7].

Weis et al. [13] use 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., [14].

Using a tree-based setup, Molnar and Wagner [14] distribute $O(\log n)$ 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 [15]. 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, Avoine et al. [15] propose the OSK/AO protocol using hash-chains, also proposed by Ohkubo et al. [16]. Yet, OSK/AO is known to leak privacy [7], requires an expensive hash function, and a state on the tag.

With the HB+ protocol by Juels and Weis [6], 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 [10], [25]. Also note that with HB+ and all variants based on LPN-schemes [26], there will always be a potentially nonnegligible probability that a valid tag gets rejected by the reader—HB+ is not complete.

In [9], an algebraic technique to compute $\frac{2}{3}$ of all key bits is proposed. Soos [8] independently discovered the 2^{2m+1} equivalence classes in DPM. These papers can be seen as special cases of the algebraic reasoning provided in this paper to indicate the security of F_f. The paper at hand is a generalization of the above papers, additionally discussing security against SAT-solving and LPN applying to all variants of DPM-like privacy-preserving authentication protocols.

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

8 Conclusion

This paper presented the F_f family of privacy-preserving authentication protocols. F_f uses a simple, round-based setup, where 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 165 bit against algebraic attacks. Also, experiments indicate that SAT-solving attacks are computationally infeasible. Generally, F_f offers arbitrary, user-adjustable levels of soundness, identification rate, and even completeness.

Appendix

A1 Proofs of Sections 4 and 5

For additional information and annotations, please refer to the technical report [27] published in conjunction with this paper.

Proof of Theorem 4.1. 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 ith symbols k_i and k'_i, we can rewrite $F(K,R) = F(K',R)$:

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

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 (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. □

Proof of Theorem 4.2. We first consider 1-bit output functions ($t = 1$). 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 and K' be two keys of length $l+1$ symbols. Denote by K^l (respectively 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,

$$\exists \epsilon_2 \in \left[-(2\delta)^{\mathrm{HD}(K^l,K'^l)}, (2\delta)^{\mathrm{HD}(K^l,K'^l)}\right]$$

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):

$$\begin{aligned} Pr[F(K,R) \neq F(K',R)] = Pr[F(K^l,R^l) + f(k_{l+1}, r_{l+1}) \\ \neq F(K'^l,R) + f(k'_{l+1}, r'_{l+1})]. \end{aligned}$$

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

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

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. □

Proof of Lemma 5.1. 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 the following:

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

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

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

This is basically the dot product

$$\left(\sum_{0\leq i'\leq 2^t-1} C_{i,i'} r^{i'} \right)_{i=0,\ldots,2^t-1} \cdot (k^i)^T_{i=0,\ldots,2^t-1}.$$

Now consider E, the space generated by vectors $V = (\sum_{0\leq i'\leq 2^t-1} C_{i,i'} r^{i'})_{i=0,\ldots,2^t-1}$, 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 $(u_1(r), \ldots, u_s(r))$ be a basis. Therefore, V can be rewritten as a linear combination of the basis vector defined by a matrix A, such that: $V = (u_1(r), \ldots, u_s(r)) \cdot A$. We can then reformulate $f(k,r)$ as follows:

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

$$f(k,r) = V \cdot (k^i)_{i=0,\ldots,2^t-1}^T = [(u_1(r),\ldots,u_s(r)) \cdot A] \cdot (k^i)_{i=0,\ldots,2^t-1}^T = (u_1(r),\ldots,u_s(r)) \cdot [(k^i)_{i=0,\ldots,2^t-1} \cdot A^T]^T. \quad (6)$$

Let $(v_1(k),\ldots,v_s(k))$ denote $[(k^i)_{i=0,\ldots,2^t-1} \cdot A^T]$ and replacing it in (6), we then conclude that $f(k,r)$ can be compacted into a sum of s terms as follows:

$$f(k,r) = \sum_{j=1}^{s} u_j(r)v_j(k), \quad (7)$$

where $v_j(k)$ results from the linear combination of the monomials in (5). Finally, using (7), it is clear that knowledge of the key k is equivalent to knowledge of its expanded form $(v_1(k),\ldots,v_s(k))$. □

Proof of Theorem 5.2. From Lemma 5.1, 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 $u_j(r) = c$ is a constant function of r and 2) all function $u_j(r)$ are nonconstants. These are the only two possible cases, because if two (or more) functions u_j and $u_{j'}$ were constants, then, $u_j(r)v_j(k) + u_{j'}(r)v_{j'}(k) = [cv_j(k) + c'v_{j'}(k)] = u''_j(r) \cdot v''(k)$, where $u''(r)$ is a constant function of the key, and more importantly, the f would have been compacted to $(s-1)$. Therefore, at most one u_j function is a constant.

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

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

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 $ER_{l(s-1)+1} = 1$; $EK_{l(s-1)+1} = c \cdot \sum_{i=1}^{l} v_s(k_i)$.

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. □

Proof of Theorem 5.3. The number of monomials of degree $j, 1 \le j \le d$, where each key symbol is used at most once, e.g., $k_1,\ldots,k_j$, is $\binom{L}{j}$. This number can already be reached in a field, where the maximum order of an element is 1, e.g., in $GF(2)$. Thus, this is a lower bound.

The number of monomials of degree $j, 1 \le j \le d$, where each key symbol is used at most j times, e.g., k_1 is used three times in

$$k_1^3 \cdot k_2,\ldots,k_{j-2}, \text{ is } \binom{L+j-1}{j}.$$

In the general case, this is an upper bound. In conclusion, we derive these bounds on

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

□

Proof of Corollary 5.4. A lower bound on $\binom{L}{d}$ is given by: $\binom{L}{d} = \prod_{i=0}^{d-1}\frac{L-i}{d-i} \ge (\frac{L}{d})^d$ (because $\forall i,d,L : 0 \le i \le d-1 < L \Rightarrow (L-i)\cdot d \ge (d-i)\cdot L$). This inequality and (3) imply $U \ge \sum_{j=1}^{d}\binom{L}{j} \ge \sum_{j=1}^{d}(\frac{L}{j})^j$, 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 five, we can obtain with overwhelming probability a maximum rank matrix Γ [24]. □

A2 Applying Attacks to DPM and F_{f_Δ}

We attack DPM and F_{f_Δ} using our general algebraic framework and SAT-attacks for three purposes.

1. To show the effectiveness of this framework and algebraic attacks against "weak" instances of F_f, in particular $F_{f_{\text{DPM}}}$ ("DPM").
2. To indicate why "good" instances of F_f, in particular F_{f_Δ}, are secure.
3. Indicate why SAT-solving attacks are infeasible of breaking "good" instances of F_f, in particular F_{f_Δ}.

For better understanding, the following sections will focus on applying the attacks on $F_{f_{\text{DPM}}}$ first. Afterward, extending the attack to F_{f_Δ} and showing its computational infeasibility is straightforward.

A2.1 Algebraic Attacks on $F_{f_{\text{DPM}}}$

We will show that the adversary will be able to compute a tag's expanded-compacted key, which is sufficient to spoil authenticity, soundness, and privacy of $F_{f_{\text{DPM}}}$.

A2.1.1 Expanding-Compacting f_{DPM}

$F_{f_{\text{DPM}}}(K,R)$ is used during authentication, with

$$f_{\text{DPM}}(k_i,r_i) = M(k_{i,1}+r_{i,1}, k_{i,2}+r_{i,2}, k_{i,3}+r_{i,3}) = (r_{i,2}+r_{i,3})k_{i,1} + (r_{i,1}+r_{i,3})k_{i,2} + (r_{i,1}+r_{i,2})k_{i,3} + k_{i,1}k_{i,2} + k_{i,1}k_{i,3} + k_{i,2}k_{i,3} + r_{i,1}r_{i,2} + r_{i,1}r_{i,3} + r_{i,2}r_{i,3} = (r_{i,2}+r_{i,3})k_{i,1} + (r_{i,1}+r_{i,3})k_{i,2} + (r_{i,1}+r_{i,2})k_{i,3} + M(k_{i,1},k_{i,2},k_{i,3}) + M(r_{i,1},r_{i,2},r_{i,3}).$$

As mentioned in Section 3.3, $k_i, r_i \in GF(2^3)$, but f_{DPM} outputs in GF(2). M is the majority function. The coefficients of the $f_{\text{DPM}}(k_i,r_i)$ polynomial with unknowns $(k_{i,1},k_{i,2},k_{i,3})$ form vectors generating a space of dimension 3: coefficient $(r_{i,1}+r_{i,2})$ of $k_{i,3}$ is a linear combination of coefficients $(r_{i,2}+r_{i,3})$ of $k_{i,1}$ and $(r_{i,1}+r_{i,3})$ of $k_{i,2}$: $(r_{i,1}+r_{i,2}) = (r_{i,2}+r_{i,3}) + (r_{i,1}+r_{i,3})$ and thus

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

$$(r_{i,2}+r_{i,3})k_{i,1}+(r_{i,1}+r_{i,3})k_{i,2}+(r_{i,1}+r_{i,2})k_{i,3}$$
$$=(r_{i,2}+r_{i,3})(k_{i,1}+k_{i,3})+(r_{i,1}+r_{i,3})(k_{i,2}+k_{i,3}).$$

One of the coefficients, $M(k_{i,1},k_{i,2},k_{i,3})$, is independent of $r_{i,1},r_{i,2},r_{i,3}$ and, therefore, always equal to 1. Finally, $M(r_{i,1},r_{i,2},r_{i,3})$ is a constant term independent from $(k_{i,1},k_{i,2},k_{i,3})$. Therefore, the dimension of the space generated by the coefficients of

$$F_{f_{\mathrm{DPM}}}(K,R):=\sum_{i=1}^{l} f_{\mathrm{DPM}}(k_i,r_i)=\sum_{i=1}^{L} ER_i\cdot EK_i,$$

with

$$\begin{aligned}
ER_1\cdot EK_1&:=(r_{1,2}+r_{1,3})(k_{1,1}+k_{1,3})\\
ER_2\cdot EK_2&:=(r_{1,1}+r_{1,3})(k_{1,2}+k_{1,3})\\
&\cdots\\
ER_{2l-1}\cdot EK_{2l-1}&:=(r_{l,2}+r_{l,3})(k_{l,1}+k_{l,3})\\
ER_{2l}\cdot EK_{2l}&:=(r_{l,1}+r_{l,3})(k_{l,2}+k_{l,3})\\
ER_{2l+1}\cdot EK_{2l+1}&:=1\cdot\sum_{i=1}^{l} M(k_{i,1},k_{i,2},k_{i,3})
\end{aligned}$$

is $L=(2l+1)$.

Computation of a tag's expanded-compacted key is done by setting up and solving a system of linear equations the following.

A2.1.2 Computing Keys in DPM

With DPM, in each round of the q rounds of identification, the tag sends the pair $A_i=(R_i\oplus K)$, $b_i=F_{f_{\mathrm{DPM}}}(K,A_i)$, with

$$\begin{aligned}
A_i&=((\alpha_i)_1=(r_i)_1\oplus k_1,\ldots,(\alpha_i)_l\\
&=(r_i)_l\oplus k_l)\in GF(2^{lt}),(\alpha_i)_j\in GF(2^t)
\end{aligned}$$

to the reader that can therewith identify the tag's key K step by step. Note that there is no replay-protection in DPM during these identification rounds. Di Pietro and Molva [2] propose a key size of 117 bits for good security.

First, note that there are equivalent keys with f_{DPM} and, therefore, $F_{f_{\mathrm{DPM}}}$, the output of $f_{\mathrm{DPM}}(k_i,r_i)$ is the opposite of the output of all bits of k_i inverted, i.e.,

$$f_{\mathrm{DPM}}(k_i,r_i)=\overline{f_{\mathrm{DPM}}(\overline{k_i},r_i)}.$$

Thus, if a key K has an even number of symbols k_i whose bits are inverted compared to another key K', then K and K' are equivalent, $\forall R, F_{f_{\mathrm{DPM}}}(K,R)=F_{f_{\mathrm{DPM}}}(K',R)$. As a result, the adversary can never compute the "whole" key, but only $\frac{2}{3}$ of the key bits. However, we will see later that the remaining key bits can be easily brute-forced.

A2.1.2.1 Setting up and Solving Equations: The adversary calls LAUNCH multiple times during quality time to initiate new protocol runs with some randomly drawn tag T_{vtag}. To behave in compliance with the protocol, he gives T_{vtag} some nonce N in each protocol run by calling SENDTAG. The oracle returns with the T_{vtag}'s reply, i.e., q pairs $\{A_i,b_i\}$ from each protocol round. Now, the adversary sets up a system of equations using these pairs with K, the *unknown* key bits as follows:

$$\begin{aligned}
F_{f_{\mathrm{DPM}}}(K,A_1)&=b_1\\
F_{f_{\mathrm{DPM}}}(K,A_2)&=b_2\\
&\cdots\\
F_{f_{\mathrm{DPM}}}(K,A_j)&=b_j.
\end{aligned}$$

This system of equations can be simplified to a system of linear equations, using the linearization and expanding-compaction technique as described above. Therefore, for one observation (A_i,b_i), $F_{f_{\mathrm{DPM}}}(K,A_i)=\sum_{j=1}^{l}(ER_i)_jEK_j=b_i$ holds. Here, $EK=EK_1,\ldots,EK_L$ is the expanded-compacted key of the T_{vtag}'s original key K. For the ith observation (A_i,b_i), the adversary has

$$\begin{gathered}
\textstyle\sum_{j=1}^{L}(ER_i)_jEK_j=\\
((\alpha_i)_{1,2}+(\alpha_i)_{1,3})\cdot(k_{1,1}+k_{1,3})\\
+((\alpha_i)_{1,1}+(\alpha_i)_{1,3})\cdot(k_{1,2}+k_{1,3})\\
+1\cdot M(k_{1,1},k_{1,2},k_{1,3})+M((\alpha_i)_{1,1},(\alpha_i)_{1,2},(\alpha_i)_{1,3})\\
+\\
((\alpha_i)_{2,2}+(\alpha_i)_{2,3})\cdot(k_{2,1}+k_{2,3})\\
+((\alpha_i)_{2,1}+(\alpha_i)_{2,3})\cdot(k_{2,2}+k_{2,3})\\
+1\cdot M(k_{2,1},k_{2,2},k_{2,3})+M((\alpha_i)_{2,1},(\alpha_i)_{2,2},(\alpha_i)_{2,3})\\
+\cdots+\\
((\alpha_i)_{l,2}+(\alpha_i)_{l,3})\cdot(k_{l,1}+k_{l,3})\\
+((\alpha_i)_{l,1}+(\alpha_i)_{l,3})\cdot(k_{l,2}+k_{l,3})\\
+1\cdot M(k_{l,1},k_{l,2},k_{l,3})+M((\alpha_i)_{l,1},(\alpha_i)_{l,2},(\alpha_i)_{l,3})\\
=\\
\textstyle\sum_{j=1}^{l}((\alpha_i)_{j,1}+(\alpha_i)_{j,3})(k_{j,1}+k_{j,3})\\
\textstyle+\sum_{j=1}^{l}((\alpha_i)_{j,1}+(\alpha_i)_{j,3})(k_{j,2}+k_{j,3})\\
\textstyle+1\cdot\sum_{j=1}^{l}M(k_{j,1},k_{j,2},k_{j,3})\\
\textstyle+\sum_{j=1}^{l}M((\alpha_i)_{j,1},(\alpha_i)_{j,2},(\alpha_i)_{j,3})=b_i.
\end{gathered}$$

Now, the adversary solves the following system of linear equations $\mathbb{M}\cdot EK=B$. Here, $\mathbb{M}$ is the matrix of coefficients of EK. As of Theorem 5.2 (and Theorem 5.3 as $d=1$ with $F_{f_{\mathrm{DPM}}}$), $\mathbb{M}$ can only have rank $L=(2l+1)$. Therefore, the adversary collects $L=(2l+1)$ linear independent observations to make up a $(2l+1)\times(2l+1)$ matrix $\mathbb{M}$, with $j=\{1,\ldots,l\}$

$$\begin{aligned}
\mathbb{M}_{i,2(j-1)+1}&:=(\alpha_i)_{j,2}+(\alpha_i)_{j,3},\\
\mathbb{M}_{i,2(j-1)+2}&:=(\alpha_i)_{j,1}+(\alpha_i)_{j,3}\\
\mathbb{M}_{i,2l+1}&:=1.
\end{aligned}$$

Therefore, column $\mathbb{M}_{i,2(j-1)+1}$ represents $(k_{i,1}+k_{i,3})$, $\mathbb{M}_{i,2(j-1)+2}$ represents $(k_{i,2}+k_{i,3})$, and $\mathbb{M}_{i,2l+1}$ represents $\sum_{j=1}^{l}M(k_{j,1},k_{j,2},k_{j,3})$. Finding and guaranteeing a total of $L=(2l+1)$ linear independent observations turns out to be simple: See the proof of Corollary 5.4 and Cooper [24]. Using SENDTAG, the adversary queries the tag only a little bit more than L times, e.g., $(L+5)$ times, to get L linear independent observations with overwhelming probability. B is a vector of dimension $(2l+1)$,

$$B_i:=b_i+\sum_{j=1}^{l}M((\alpha_i)_{j,1},(\alpha_i)_{j,2},(\alpha_i)_{j,3}),1\le i\le 2l+1.$$

This system of linear equations can be solved using simple Gaussian elimination. This results in computing $(k_{1,1}+k_{1,3}),(k_{1,2}+k_{1,3}),\ldots,(k_{l,1}+k_{l,3}),(k_{l,2}+k_{l,3})$ and $\mathbb{B}=\sum_{j=1}^{l}M(k_{j,1},k_{j,2},k_{j,3})$.

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

A2.1.2.2 Computing an Equivalent Key: Finally, to compute an *equivalent* key K' of T_{vtag}'s original key, the adversary sets the first $(l-1)t$ third key bits of every key symbol to 0: $\{k_{1,3} := 0, k_{2,3} := 0, \ldots, k_{l-1,3} := 1\}$. This yields key bits $\{k_{1,1}, k_{1,2}, k_{2,1}, k_{2,2}, \ldots, k_{l,1}, k_{l,2}\}$. Now all key bits, besides the last key bit $k_{l,3}$, are known. This last bit is set to *either* 1, *iff* $\sum_{j=1}^{l-1} M(k_{j,1}, k_{j,2}) + M(k_{l,1}, k_{l,2}, 1) = \sum_{j=1}^{l} M(k_{j,1}, k_{j,2}, k_{j,3}) = \mathbb{B}$, *or* 0, *iff* $\sum_{j=1}^{l-1} M(k_{j,1}, k_{j,2}) + M(k_{l,1}, k_{l,2}, 0) = \mathbb{B}$.

Therefore, this attack allows the adversary to compute an equivalent key K' of a tag's original key K.

A2.1.3 Breaking Privacy

According to the privacy definition of Section 2.2.2, the adversary should not have a more than negligible probability over guessing (50 percent) of linking subsequent protocol runs of two tags to one tag.

The adversary calls DRAWTAG to receive one tag T_{vtag} for quality time. During this quality time, he computes an equivalent key K for T_{vtag} using the above attack. Afterward, he calls FREE T_{vtag} to free this tag, he again calls DRAWTAG to receive a second tag $T_{vtag'}$ for quality time and computes an equivalent key K' for $T_{vtag'}$. If keys K and K' are not equivalent, i.e., by inverting an even number of 3-bit symbols, he knows for sure (100 percent) that T_{vtag} was not the same tag as $T_{vtag'}$. If the keys are equivalent, he guesses with 50 percent whether T_{vtag} is the same as $T_{vtag'}$.

There are 2^{l-1} equivalent keys in each of the 2^{2l+1} equivalence *classes* of $F_{f_{\text{DPM}}}$. Therefore, in $n\frac{2^{l-1}}{2^{3l}}$ out of n cases, $T_{vtag'}$ gives a key K' which is equivalent to K. In these cases, the adversary wins with 50 percent. In the remaining $n\frac{2^{3l}-2^{l-1}}{2^{3l}}$ out of n cases, $T_{vtag'}$ is a nonequivalent tag, and the adversary wins with 100 percent. More formally,

$$P_{\text{win}} = \frac{n\frac{2^{l-1}}{2^{3l}}}{n} \cdot 50\% + \frac{n\frac{2^{3l}-2^{l-1}}{2^{3l}}}{n} \cdot 100\% = 1 - \frac{1}{2^{2l+2}}.$$

For $l \geq 1$, this is much better than a guessing adversary with 50 percent. With $l = 39$, as proposed by Di Pietro and Molva [2], $P_{\text{win}} = 1 - \frac{1}{2^{80}} \approx 100\%$.

A2.1.3.1 Finding the "Right" Key: In the original DPM-protocol proposal, there is an additional step required after q identifications rounds with $F_{f_{\text{DPM}}}$ to protect against replay attacks. The HMAC $h(K|R_1|N|K)$ is sent from the tag to the reader, with N being the nonce received from the tag, R_1 the first random number of the tag, and h a strong hash function (e.g., SHA-1). The reader compares $h(K|R_1|N|K)$ with the HMAC computed using the key(s) he identified in his database during the preceding q rounds. Therefore, if the adversary computes only an equivalent key K' of the tag's original key K, the reader's verification of above HMAC fails, and the adversary will be rejected, RESULT will be 0.

Still, the adversary can easily brute-force K using K': two keys $K \neq K'$ are equivalent for $F_{f_{\text{DPM}}}$, if an even number of key symbols is inverted from K to K'. There are 2^{l-1} keys equivalent to K, including K itself. To find out the correct K, the adversary has to additionally do 2^{l-2} SHA-1 computations on average for all equivalent keys K'_{equiv} of K'. Each of the generated equivalent keys K'_{equiv} of K' is hashed with $h(K'_{\text{equiv}}|R_1|N|K'_{\text{equiv}})$ and presented to the reader with a call to SENDREADER, until RESULT $= 1$, i.e., if $K'_{\text{equiv}} = K$.

Conclusion. Breaking privacy with $F_{f_{\text{DPM}}}$ is equal to solving a system linear equations with $U = L = (2l+1)$ unknowns and has $\approx L^3$ computational complexity. With the originally proposed 117-bit key size ($l = 39$), computational complexity is $\approx 2^{19}$. Experimental results on breaking privacy and computing key-bits of F_{f_Δ} are presented in Section 5.2.2.

A2.2 Algebraic Attacks on F_{f_Δ}

The same attacks can be applied against F_{f_Δ}: every output bit of $f_\Delta(k_i, r_i)$ captures all random and key bits $r_{i,j}, k_{i,j}, 1 \leq j \leq 4$. Therefore, for simplicity, the adversary can set up equations only by looking at one, w.l.o.g., the first output bit $F_{f_{\Delta_1}}$. This will be sufficient for him to compute all key bits.

By its design, expanding-compacting f_{Δ_1} adds three new monomials per key symbol which results in $s = 7$ and $L = l \cdot s = 448$. Using the attack setup of Section 5.2 with $d = 8$, this results in a system of linear equations with U unknowns, $\sum_{j=1}^{8} \binom{64.7}{j} \approx 2^{55} \leq U \leq \sum_{j=1}^{8} \binom{64.7+j-1}{j} \approx 2^{55}$. The adversary needs to set up 2^{55} equations and thus requires $\approx 2^{165}$ computational complexity for matrix inversion to compute key bits. F_{f_Δ}, therefore, has 165 bits of security against algebraic impersonation. Therefore, F_{f_Δ} is safe against algebraic attacks. While there might be optimizations to such attacks, we believe they will not significantly improve the computational complexity.

A2.3 SAT-Solving

Recently, a lot of attention has been drawn to the use of SAT-solvers in cryptography [1], [11], [23], [28], [29], [31], [32]. In theory, the problem of solving a system of multivariate equations is transformed into an equivalent boolean Satisfiability (SAT) problem. If there is a satisfying solution for the SAT-problem's variables, then this is also a solution for the variable assignments of the original system of multivariate equation. Therefore, the idea is to convert ANF-equations into boolean Conjunctive Normal Form (CNF) and then use a SAT-solver.

The anticipated benefit of transforming multivariate algebraic equations to a SAT-problem for cryptography is to get a "good" instance of a SAT-problem that can be solved by an indeterministic SAT-solver quickly. Therefore, we also attacked $F_{f_{\text{DPM}}}$ and F_{f_Δ} using the SAT-solver MiniSat [33].

The sequel describes all main theoretical properties of our SAT-solving attacks on $F_{f_{\text{DPM}}}$ and F_{f_Δ}, in particular details required for the conversion to CNF. For further technical details, refer to the list of papers cited. Sections 5.2.2 presents experimental evaluation results.

Linearization. Based on plain and cipher text observations, an adversary can potentially make, linearized equations in ANF are set up using a simple substitution technique [11]. All ANF equations are of the form $\sum_i u_i = 1$, for all linearized monomials u_i that have coefficients 1. The set $\mathbb{S}$ of left-hand-side expressions of above equations will be converted to CNF.

Grouping. Before the actual conversion of $\mathbb{S}$, it is suggested [28] to replace frequently reappearing groups of

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

monomials in $\mathbb{S}$ by introducing new variables. If there is a group of monomials $u_i + \cdots + u_j$ in many expressions of $\mathbb{S}$, this group is substituted in $\mathbb{S}$ by a new variable t, and the expression $\{1 + t + u_i + \cdots + u_j\}$ is added to $\mathbb{S}$. We implemented this by replacing groups of monomials appearing more often than a certain percentage p in the equations. The results in an increased *sparsity* of expressions. Sparse expressions, consisting only of a small amount of monomials, reduce CNF clause length, number of clauses, and SAT-solving time.

Elimination of Variables. It is also suggested to replace some of the variables in $\mathbb{S}$ by their definition [11]. If there is an expression $\{u_i + u_{i+1} + \cdots + u_j\} \in \mathbb{S}$, then, e.g., u_i is replaced inside every other expression in $\mathbb{S}$ by $1 + u_{i+1} + \cdots + u_j$. This effectively eliminates u_i from SAT-solving speeding up the solving process. However, as this *reduces* the sparsity of the expression and *might* have negative effects on computation time, cf., Section 5.2.2.

Guessing. Experiments [29], [31] indicate that it is worthwhile to systematically bute-force ("guess") some of the variables, i.e., key bits, before using SAT-solvers. With lt being the total key size, the adversary randomly picks $(u \ll lt)$ key bits $k_i, \ldots, k_{i+u-1}$. The adversary iterates over all possible 2^u assignments for these key bits. If the SAT-solver returns *unsatisfiable* for an assignment, the adversary knows that this assignment was wrong and proceeds with the next iteration until the right solution is found. It is expected that 2^u invocations of a SAT-solver with $(lt - t)$ key bits is faster than one invocation with lt key bits.

Cutting. Finally, $\mathbb{S}$ is converted to a boolean CNF, using the convention $1 \equiv \mathrm{True}$, $0 \equiv \mathrm{False}$. As the conversion of XORs in ANF expressions has exponential complexity in terms of CNF clause length, ANF expressions are typically *cut* before conversion: an expression $u_1 + u_2 + \cdots + u_k + u_{k+1} + \cdots + u_a$ is split into multiple expressions $\{u_1 + u_2 + \cdots + t_1,\ 1 + t_1 + u_{k+1} + \cdots + u_{2k-2},\ 1 + t_2 + u_{2k-1} + \cdots\}$, where each expression consists of at most k monomials, and new variables t_i are introduced [11]. Here, k is called the *cutting number*.

Please refer to Section 5.2.2 for practical evaluation results of algebraic and SAT-attacks.

Acknowledgments

The authors wish to thank Olivier Billet for pointing at and discussing LPN attacks.

References

[1] N. Courtois, K. Nohl, and S. O'Neil, "Algebraic Attacks on the Crypto-1 Stream Cipher in Mifare Classic and Oyster Cards," http://eprint.iacr.org/2008/166.pdf, 2008.

[2] R. Di Pietro and R. Molva, "Information Confinement, Privacy, and Security in RFID Systems," *Lecture Notes in Computer Science*, pp. 187-202, Springer, 2007.

[3] Y. Choi, M. Kim, T. Kim, and H. Kim, "Low Power Implementation of SHA-1 Algorithm for RFID System," *Proc. 10th Int'l Symp. Consumer Electronics*, pp. 1-5, 2006.

[4] M. Feldhofer and C. Rechberger, "A Case Against Currently Used Hash Functions in RFID Protocols," *Proc. OTM Conf.*, pp. 372-381, 2006.

[5] M. Feldhofer and J. Wolkerstorfer, "Strong Crypto for Rfid Tags—A Comparison of Low-Power Hardware Implementations," *Proc. Int'l Symp. Circuits and Systems*, pp. 1839-1842, 2007.

[6] A. Juels and S. Weis, "Authenticating Pervasive Devices with Human Protocols," *Proc. 25th Ann. Int'l Cryptology Conf.*, pp. 293-308, 2005.

[7] A. Juels and S. Weis, "Defining Strong Privacy for RFID," *Proc. IEEE Pervasive Computing Comm. Workshops*, pp. 342-347, 2007.

[8] M. Soos, "Analysing the Molva and Di Pietro Private Rfid Authentication Scheme," *RFIDSec*, http://events.iaik.tugraz.at/RFIDSec08/, 2008.

[9] T. van Deursen, S. Mauw, and S. Radomirovic, "Untraceability of Rfid Protocols," *Proc. Second Workshop Information Security Theory and Practices. Smart Devices, Convergence and Next Generation Networks*, pp. 1-15, 2008.

[10] E. Levieil, P.-A. Fouque, "An Improved LPN Algorithm," *Proc. Conf. Security and Cryptography Networks*, pp. 348-359, 2006.

[11] G. Bard, N. Courtois, and C. Jefferson, "Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials Over gf(2) via Sat-Solvers," *Proc. European Network Excellence Cryptology Workshop*, http://eprint.iacr.org/2007/024/, 2007.

[12] G. Tsudik, "Ya-Trap: Yet Another Trivial RFID Authentication Protocol," *Proc. Int'l Conf. Pervasive Computing and Comm. Workshops*, 2006.

[13] S. Weis, S. Sarma, R. Rivest, D. Engels, "Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems" *Proc. Security in Pervasive Computing Conf.*, pp. 201-212, 2003.

[14] D. Molnar, D. Wagner, "Privacy and Security in Library RFID: Issues, Practices, and Architectures" *Proc. Conf. Computer and Comm. Security*, pp. 210-219, 2004.

[15] G. Avoine, E. Dysli, P. Oechslin, "Reducing Time Complexity in RFID Systems" *Proc. Workshop Selected Areas in Cryptography*, pp. 291-306, 2005.

[16] M. Ohkubo, K. Suzuki, and S. Kinoshita, "Cryptographic Approach to Privacy-Friendly Tagsss," *Proc. Radio-Frequency Identification Privacy Workshop*, http://www.rfidprivacy.us/2003/agenda.php, 2003.

[17] EPCglobal, "Epcglobal Standards and Technology," http://www.epcglobalinc.org/standards/, 2008.

[18] S. Vaudenay, "On Privacy Models for RFID," *Proc. Int'l Conf. Theory and Application of Cryptology and Information Security*, pp. 68-87, 2007.

[19] T. van Deursen and S. Radomirovic, "Attacks on RFID Protocols," http://eprint.iacr.org/2008/310, 2008.

[20] I. Damgård, M. Østergaard, "Rfid Security: Tradeoffs Between Security and Efficiency," *Proc. RSA Conf.*, http://eprint.iacr.org/2006/234.pdf, pp. 318-332, 2006.

[21] L. Batina, J. Lano, N. Mentens, B. Preneel, I. Verbauwhede, S. Oers, "Energy, Performance, Area Versus Security Trade-Offs for Stream Ciphers," *Proc. European Network of Excellence in Cryptogoy (ECRYPT) Workshop, The State of the Art of Stream Ciphers (SASC)*, pp. 302-310, 2004.

[22] D. Dobkin, *The RF in Rfid: Passive UHF Rfid in Practice*. Elsevier, 2007.

[23] N. Courtois and G. Bard, "Algebraic Cryptanalysis of the Data Encryption Standard," *Lecture Notes in Computer Science, Cryptography and Coding*, pp. 152-169, Springer, 2007.

[24] C. Cooper, "On the Rank of Random Matrices," *Random Structures and Algorithms*, vol. 16, no. 2, pp. 209-232, 2000.

[25] H. Gilbert, M. Robshaw, and H. Sibert, "Active Attack Against Hb+: A Provably Secure Lightweight Authentication Protocol," *IEEE Electronic Letters*, vol. 41, no. 21, pp. 1169-1170, Oct. 2005.

[26] S. Weis, "Hb+ Protocol Information Page," http://saweis.net/hbplus.shtml, 2008.

[27] E.-O. Blass, A. Kurmus, R. Molva, G. Noubir, and A. Shikfa, "The f_f-Family of Protocols for Rfid-Privacy and Authentication," http://eprint.iacr.org/2008/476.pdf, 2008.

[28] C. McDonald, C. Charnes, and J. Pieprzyk, "Attacking Bivium with Minisat," http://www.ecrypt.eu.org/stream/papersdir/2007/040.pdf, 2007.

[29] C. McDonald, C. Charnes, and J. Pieprzyk, "An Algebraic Analysis of Trivium Ciphers Based on the Boolean Satisfiability Problem," http://eprint.iacr.org/2007/129, 2007.

[30] I. Mironov, L. Zhang, "Applications of Sat Solvers to Cryptanalysis of Hash Functions," *Proc. Int'l Conf. Theory and Applications of Satisfiability Testing (SAT '06)*, pp. 102-115, 2006.

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.

[31] T. Eibach, E. Pilz, and S. Steck, "Comparing and Optomising Two Generic Attacks on Bivium," *Proc. State of the Art of Stream Ciphers Workshop (SASC '08)*, http://www.ecrypt.eu.org/stvl/sasc2008, 2008.
[32] T. Eibach, E. Pilz, and G. Voelkel, "Attacking Bivium Using Sat Solvers." *Proc. Int'l Conf. Theory and Applications of Satisfiability Testing (SAT '08)*, pp. 63-76, 2008.
[33] N. Eèn, N. Sörensson, "An eXtensible Sat-Solver," *Theory and Applications of Satisfiability Testing*, pp. 502-518, Santa Margherita Ligure, 2004.

Erik-Oliver Blass received the diploma and doctoral degrees in computer science from the Institut für Telematik at the Universität Karlsruhe (TH). He is a postdoctoral fellow at the EURECOM's Network and Security team. His current work focuses on security and privacy aspects of RFID systems, in particular, basic communication protocols between "tags" and "readers". In his previous work, he investigated new security aspects of wireless sensor networks, where extremely resource-restricted nodes can only achieve "better-than-nothing" security guarantees.

Anil Kurmus is a TELECOM ParisTech engineer (2009). Being interested in all kinds of computer and communication security problems, his major focus currently lies on system security, and he is now working at IBM Research, Zurich, on practical aspects of cloud storage security. At IBM, he completed his thesis on the subject of access control for key management systems. Before graduation, he was an intern at EURECOM on RFID security and privacy. He recently discovered the first real-world attack for the TLS renegotiation vulnerability.

Refik Molva is a professor at EURECOM. His research interests are the design and evaluation of protocols for security and privacy in self-organizing systems. He was program chair or general chair for security conferences, such as ESORICS, RAID, SecureComm, IEEE ICC, and security workshops. He is an area editor for the *Computer Networks Journal*, the *Computer Communications* Magazine, and the *Pervasive and Mobile Computing Journal*. He worked in the Zurich Research Laboratory of IBM as one of the key designers of the KryptoKnight security system.

Guevara Noubir received the PhD degree in computer science from the Swiss Federal Institute of Technology in Lausanne (1996). He researches both theoretical and practical aspects of secure and robust wireless communication systems. He joined Northeastern University in 2001 and is now an associate professor of computer science. During 1997 and 2000, he was a senior researcher at CSEM SA (Switzerland), where he led several research projects on wireless communication systems. He is a recipient of the US National Science Foundation (NSF) CAREER Award.

Abdullatif Shikfa is an alumni of the Ecole Polytechnique (MSc, 2004), of the University of Nice Sophia Antipolis (MSc 2005), and of the ENST-EURECOM (MSc 2006). He is a PhD candidate and research engineer working at EURECOM in the Network and Security team on the EU project Haggle. His current research focuses on applied cryptography and security protocols in challenging environments, such as mobile opportunistic networks or RFIDs.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

Authorized licensed use limited to: Monash University. Downloaded on October 26,2020 at 05:08:49 UTC from IEEE Xplore. Restrictions apply.