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

Erik-Oliver Blass[1] Anil Kurmus[1] Refik Molva[1] Guevara Noubir[2] Abdullatif Shikfa[1]

[1] Eurecom, Sophia Antipolis, France
[2] Northeastern University, Boston, USA

## Abstract

*In this paper, we present the design of the lightweight $F_f$ family of privacy-preserving authentication protocols for RFID-systems. $F_f$ is based on a new algebraic framework for reasoning about and analyzing this kind of authentication protocols. $F_f$ offers user-adjustable, strong authenticity and privacy against known algebraic and also recent SAT-solving attacks. In contrast to related work, $F_f$ achieves these two 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. In this paper, the authenticity and privacy of $F_f$ is evaluated using analytical and experimental methods.*

## 1. Introduction

Nowadays, Radio-Frequency-Identification (RFID) is used for a variety of applications, ranging from simple library borrowing systems, access-control, up to complete Supply-Chain-Management solutions. 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 in our daily life 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., [11, 12]. Similar to fraudulent usage of services or illegal access to cars, the privacy of RFID users is 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.

In [37], Di Pietro and Molva introduced a privacy-preserving authentication protocol for RFID tags called "DPM". This protocol provides tag anonymity and security against malicious impersonation of a tag. However, the DPM-protocol suffers from some weaknesses. Based on an algebraic approach, an adversary is able to compute $\frac{2}{3}$ of the secret key bits shared between reader and tag and also break the tag's privacy, cf., [40, 42].

While one advantage of the DPM-protocol is reduced complexity on the reader-side during authentication, a major drawback of the DPM-protocol lies in the requirement to evaluate a strong, cryptographic hash function, e.g., SHA-1/-2, on the tag. Such cryptographic primitives are in general to costly for tags in most RFID-applications, as they alone already require $\approx 10,000$ gate equivalents (NANDs), while tags are often assumed to feature around $1,000$ to $10,000$ gates available for the whole security protocol on the tag, cf., [6, 21, 22, 24, 25].

Inspired by the DPM-protocol, we propose a family of new low-cost authentication protocols for RFID tags that provide key secrecy and privacy. The protocols and underlying keyed-hashing functions are designed to withstand known and new efficient algebraic attacks, statistical attacks, LPN attacks, and recent SAT-solving based attacks. We introduce a new algebraic attack and show it's efficiency against existing protocols.
In conclusion, the major **contributions** of this paper are:

- First, we develop a new kind of algebraic, linearization attack that is especially effective against DPM [37], breaking key secrecy and privacy.
- We propose the new $F_f$ family of protocols which is designed such that it is secure against algebraic attacks, LPN attacks [27], and also against a new type of attack based on SAT-solving [3]. We present theoretical analysis as well as implementation and evaluation results to argue for $F_f$'s security.

- Compared to related work, e.g., [2, 35–37, 41, 46], there is no expensive cryptographic hash function like SHA-1 running on the tag required anymore: $F_f$ is extremely lightweight. Similarly, the reader does not need to be able to compute SHA-1, but can also be resource restricted, e.g., an embedded device. Nevertheless with $F_f$, authentication as well as privacy can be assured with an arbitrary, user-adjustable level of security.
- Contrary to, e.g., [2, 35, 36, 41], $F_f$ does not require the existence of a non-volatile state on the tag.
- Finally, the $F_f$-protocols are "unconditionally" complete, i.e., a valid, legitimate tag will *always* be identified by the reader as a valid tag – in contrast to, for example HB+ [24] and variants, where this is only guaranteed within a certain probability.

After introducing our system- and adversary model together with the definition of the two security goals authenticity and privacy in Section 2, Section 3 presents an overview of the $F_f$-family of privacy-preserving authentication protocols. In Section 4, we discuss the necessary properties of low-complexity keyed-hashing functions to at least guarantee key non-equivalence and other statistical properties. Section 5, we then discuss the algebraic properties of the considered hashing function, such as linearization, key expansion-compaction, and analysis of a new algebraic attack that performs very well against special instances of $F_f$, e.g., [37]. Finally, in Section 6, we propose a new, secure instance for $F_f$ and evaluate the algebraic and SAT-solving attacks on it.

## 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 $T_{\mathrm{ID}}$, i.e., $T_{\mathrm{ID}}$ is the unique name or ID of a tag. Each tag $T_{\mathrm{ID}}$ shares a different secret, e.g., a key $K_{T_{\mathrm{ID}}}$, with the reader. The reader stores $n$ different tuples $(T_{\mathrm{ID}}, K_{T_{\mathrm{ID}}})$ as entries in its database $\mathbb{D}, n = |\mathbb{D}|$. For better comparison, we set $n$ to a typical value, i.e., $n = 2^{16}$ as in [37].

The setup, i.e., the simple application, used in this paper is a reader in front of a closed (and locked) door. The reader will unlock and open the door, if and only if he can identify a tag $T_{\mathrm{ID}} \in \mathbb{D}$. The reader identifies a single tag with the usage of 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. We assume tags similar to EPC Global Gen. 2 Class 1 tags [18]. These read-only tags are assumed to be *passive*, without a battery. They draw their power to operate completely from the reader's electromagnetic field. Tags do not feature a state, i.e., some kind of non-volatile memory. However, we assume each tag has access to a source of (cryptographically) good randomness. For example, a tag could draw random data from its antenna by extrapolation of static-noise.

Today's tags feature only a couple of thousands *Gate Equivalents* [24, 25] (GE), thus the implementation of complex hash functions like SHA-1 (requiring $10,641$ GE [6]) is impossible.

Typically a tag is in the reader's communication range for only a short duration of time, e.g., while its user "swings" it close to the reader. We assume this to be $\approx 1$s. The overall amount of data exchanged between the tag and the reader is thus limited.

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. The reader is therefore not capable of doing complex computations, e.g., SHA-1 computations, on all elements of its database. As a result, strong hash functions are not available for RFID protocol design.

### 2.1. Adversary model

The adversary model in this paper follows the definitions of [44]: 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* [43] phase, by drawing a tag into his possession. During this phase, he can query the tag a finite, reasonable number of times, i.e., send messages to and receive answers from the tag. Reasonable means that the adversary can not exceed more operations than typical "security margins" [26]. For example, he can query only $\ll 2^{64}$ times. He can also send and receive messages to and from the reader a finite, reasonable number of times. The adversary can, however, neither

read out a tag's nor the reader memory. He can not compromise a tag or the reader.

More formally, the adversary executes an algorithm, and this algorithm has access to a set of oracles that can be called a finite number of times. He can do a DRAWTAG oracle-call. This will give him, out of the set of all possible tags, one temporarily anonymized tag ID $T_{vtag}$ of a tag $T_{\mathrm{ID}}$. The oracle randomly chooses $T_{\mathrm{ID}}$ 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 randomly choose the same tag $T_{\mathrm{ID}}$. 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. 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. 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. Finally, if a protocol instance is finished, i.e., the adversary called SENDREADER, a query to the RESULT-oracle will tell the adversary, if the reader accepted the data it received, apparently identified a tag, and opened the door.

With EXECUTE, a complete protocol instance is run through, and the adversary receives from the oracle all communication between $T_{vtag}$ and the reader.

This adversary is called *non-narrow weak* in [44].

**Tag Compromise and Destruction.** Instead of the above *non-narrow weak* adversary, we could also allow the adversary to compromise and destruct tags or to create fake tags. However, such a *strong* adversary would not have any advantage over the *weak* adversary in our setup: we assume tags to be stateless, and in the $F_f$ family of protocols, tags do not share any keys or secret information, even not partially. Consequently, there would be no gain in compromising or destroying tags for the adversary. So, even the $F_f$ is secure against a *strong* adversary, it is sufficient to focus only on the capabilities of a *weak* adversary in this paper.

## 2.2. Security Goals

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

**2.2.1. Authenticity.** The purpose of a tag is to authentically identify itself, and therewith the person carrying the tag, towards the reader using some communication protocol. Authenticity in this context means that, after execution of the communication protocol, i.e., one protocol run, the reader can be sure that a certain, specific tag took part in this communication protocol – and not a non-legitimate adversary. The reader should be sure with "overwhelming probability", the adversary should be able to fake authentication only with "negligible probability".

More formally speaking, the communication protocol should result in *recent aliveness*, with the reader being the *initiator* of the protocol, e.g., see definitions in [29, 43].

As in [44], the adversary will use the above oracles, put tags into quality time and query them. Nevertheless, he should not succeed in making the reader authenticate some tag $T_{\mathrm{ID}} \in \mathbb{D}$ in a protocol execution, without that $T_{\mathrm{ID}}$ takes part in this protocol execution. The probability of succeeding should be negligible for the adversary.

The above can also be called the *soundness* of an identification protocol [13]. Similar, *completeness* means that if some tag $T_{\mathrm{ID}} \in \mathbb{D}$ takes part in a protocol run with the reader, and the adversary does not modify or block the tag's message to the reader, the reader should never reject this tag, i.e., always open the door for a valid tag.

Please 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 $T_{\mathrm{ID}}$, 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.

Also note that we do not focus on any form of *agreement* authentication, i.e., we do not care about 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 authenticity aims at protecting the tag identification, privacy 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 tell whether $T_{vtag} = T_{vtag'}$ or not.

Following the more formal definitions of [44], the adversary draws a tag into quality time using the DRAWTAG-oracle. He receives $T_{vtag}$ and calls the above mentioned oracles a finite number of times. He uses FREE to release the tag. Finally, he does

this procedure a second time: draws $T_{vtag'}$, calls the oracles, and frees $T_{vtag'}$.

Now, the adversary has to decide whether $T_{vtag}$ was the same tag as $T_{vtag}$. If the probability of the adversary doing the right decision is negligible small, the protocol is private.

## 3. A Family of Authentic and Private RFID Protocols

The scope of this paper is a family of RFID protocols that allow for the identification of tags by readers in a privacy-preserving manner. In a generic, challenge-response based authentication protocol, a reader sends a nonce $N$ to a tag $T_{\mathrm{ID}}$, and the tag replies with $\{R, F(N, R, K_{T_{\mathrm{ID}}})\}$, see [5]. Here, $N$ is the reader's challenge for replay protection, $R$ is a random nonce by the tag for privacy protection, $K_{T_{\mathrm{ID}}}$ is the pairwise secret key between tag and reader, and $F$ is an HMAC-like ("keyed-hash") function.

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

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

### 3.1. Round-Based Identification

Figure 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 detection. In the second message, the tag $T_{\mathrm{ID}}$ replies with a set of a total of $q$ random numbers ($R_i$) and the results of a one-way function $w_i := F(R_i, N, K_{T_{\mathrm{ID}}})$ computed over each $R_i$, $N$, and the tag's secret identification key $K_{T_{\mathrm{ID}}}$. We call each pair $(R_i, w_i = F(R_i, N, K_{T_{\mathrm{ID}}}))$ sent 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 identity of the tag.

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

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

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

*3.) Identification Rate*: the received value of $F$ and the one computed by the reader should be different with a non-negligible probability for the entries in the reader's database that do not match the tag; in other words: if $K_{T_{\mathrm{ID}}} \neq K'$, then $F(R_i, N, K_{T_{\mathrm{ID}}}) \neq F(R_i, N, K')$ with a non-negligible probability. Ideally, this probability should be close to $50\%$ to give good identification rate and to protect the privacy of tags. If two tags reply to one query with the same outputs or different outputs in half of the cases, respectively, the adversary does not gain any information whether the tags are the same or not. Finally, a good identification rate does not only help the reader to eventually identify a tag, but also prevents the adversary to *impersonate* any tag in $\mathbb{D}$ by simply sending random data to the reader.

### 3.2. The $F_f$ Protocol Family

Based on the above "round-based" idea of protocols and inspired by [37], we now describe and concentrate

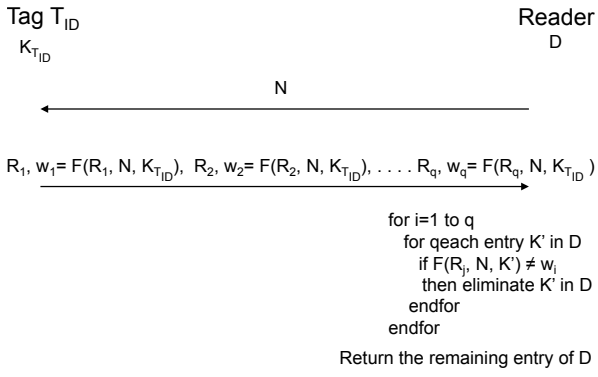



Figure 1. Round-based tag identification

on our $F_f$ family of private authentication protocols. Each tag $T_{\text{ID}}$ shares not only one key $K_{T_{\text{ID}}}$, but also a second secret key $K'_{T_{\text{ID}}}$ with the reader. Consequently, the reader stores tuples $(T_{\text{ID}}, \{K_{T_{\text{ID}}}, K'_{T_{\text{ID}}}\})$ in its database $\mathbb{D}$.

1) Each protocol run, i.e., single execution of the protocol, starts with the reader sending a nonce. Reader $\rightarrow$ Tag: $N_0 \in GF(2^{lmt})$
2) The tag ($T_{\text{ID}}$) replies with a single message. This message's
   content 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 $R_u^v, N_u, K_{T_{\text{ID}}}, K'_{T_{\text{ID}}} \in GF(2^{lmt})$, $a_i \in \{1 \ldots d\}$, $q \in \mathbb{N}$, $F_f : GF(2^{lmt}) \times GF(2^{lmt}) \rightarrow GF(2^t)$.

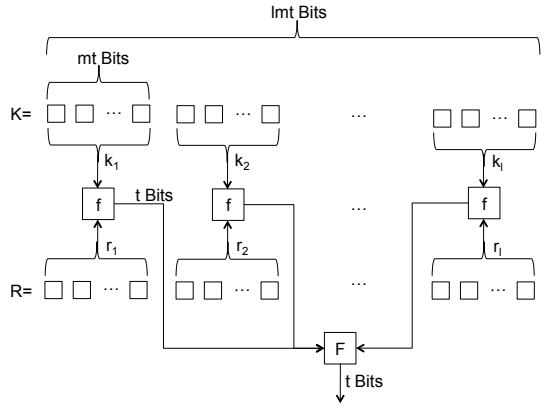



Figure 2. Overview of $l$, $m$, $t$, $F_f$, $f$, $K$, $k_i$, $R$, $r_i$

In every round $i$, $a_i$ is chosen randomly by the tag. So, $T_{\text{ID}}$ 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 in each round, $T_{\text{ID}}$ 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. So 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 $w_i = F_f(K_{T_{\text{ID}}}, R_i^{a_i}) + F_f(K'_{T_{\text{ID}}}, N_i)$, $R_i$ of Fig. 1 is split into $R_i^1, \ldots, R_i^d$.

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

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

If and only if *all* of the above equations are invalid, the entry $(T_j, \{K_{T_j}, K'_{T_j}\})$ is removed from $\mathbb{D}$ and the reader continues with the next round $i+1$ and the reduced database. The idea is that after $q$-rounds, there will be only 1 tag remaining in $\mathbb{D}$. We call this kind of identification of a single tag *converging* to a single entry.

You can already see that $F_f$ provides completeness: for data sent from a valid tag, at least one equation will always hold. Therefore, a valid tag will never be removed from $\mathbb{D}$ and never be rejected from the reader.

**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.

In $F_f$, we derive all $N_i$ from $N_0$ as explained later in Section 3.2.4.

**3.2.3. Relation between $F_f$ and $f$.** Furthermore in our family of protocols, $F_f$ is made of small fan-in functions $f$, $f : GF(2^{mt}) \times GF(2^{mt}) \rightarrow GF(2^t)$, 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 $(l \cdot m \cdot t)$ bits. Throughout this paper, we will group subsequent bits of a key or nonce into $l$ so called *symbols*, such that $K = (k_1, \ldots, k_l)$, $R = (r_1, \ldots, r_l)$. Each of the $l$ symbols consists of $(mt)$ bits. By writing $k_{i,j}$ or $r_{i,j}$, we denote the $j^{th}$ bit of the $i^{th}$ key symbol or random symbol, respectively.

These relations are shown in Fig. 2.

Parameters $\{d, l, m, t\}$ are system parameters and will be discussed later. Finally, real-world example

values for these parameters are presented in Section 6.

**3.2.4. 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 choosen-plaintext-attacks on the tag's key by modifying the answer he receives from SENDTAG and calling the SENDREADER oracle with the 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 $q, d, l, m, t$, this amount of data might be exceeded such that the tag can not authenticate itself.

To overcome both problems, we therefore propose to derive subsequent $R_i^j$ from previous $R_i^j$ using a pseudo-random-number-generator PRNG.

More formally, we propose to compute

$$\begin{aligned} R_i^j &:= \mathrm{PRNG}(R_i^{j-1}), j > 1 \\ R_i^1 &:= \mathrm{PRNG}(R_{i-1}^d). \end{aligned}$$

Now, the tag only needs to draw and send one single random number, $R_0^d$, to the reader. This reduces the opportunity for the adversary to precisely modify subsequent $R$s, as he is able to choose only the first random date $R_0^d$. Also, data volume that is wirelessly sent to the reader shrinks from $(qd) \cdot |R|$ bits to $|R|$ bits.

Still within the protocol, the tag computes all pseudo-random numbers $R_i^j, 1 \leq j \leq d$ for each round $i$ and then randomly, i.e., indeterministically, chooses one $a_i$ and computes $F_f(K, R_i^{a_i}) + F(K', N_i)$.

The second $F_f$ protocol message, from the tag to the reader, now looks as follows: $(R_0^d, F_f(K, R_1^{a_1}) + F(K', N_1), F_f(K, R_2^{a_2}) + F(K', N_2), \ldots, F_f(K, R_q^{a_q}) + F(K', N_q))$, where $a_i$ is chosen (really) randomly each time. With $R_0^d$, the reader computes all subsequent pseudo-random numbers and proceeds as in Section 3.2.1.

As we do not care about the secrecy of the internal state of PRNG, but only require (pseudo)-randomness for the $R$s, we use a cheap LFSR to derive them 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. Section 6 will argue about the costs of implementing an LFSR on a tag and why using an LFSR does not have an impact on security. As we only care about the (pseudo)-randomness of the $R$s, it is in the following Sections sufficient to only *assume* that each subsequent $R_i^j$ is still drawn truly indeterministically.

**Deriving $N_i$.** The above also holds for subsequent $N_i$s that are required for replay-protection: we derive

$$N_{i+1} := \mathrm{PRNG}(N_i),$$

with the original $N_0$ received from the reader.

Let us again assume for now that the $N_i$ are completely random, deriving from an LFSR is not a security issue.

For the following algebraic reasoning, let us also not consider $F_f(K, R_i^{a_i}) + F_f(K', N_i)$ in each round $i$, but focus only on $F_f(K, R_i^{a_i})$. For the discussion on statistical properties of $F_f$ and its strength against algebraic and SAT-attacks, this is sufficient.

# 4. Low-cost hash functions with good statistical properties

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

We will first focus on hashing functions with $t = 1$ bit output, as this is the case, e.g., for DPM [37], and security properties are much easier to derive. Then, in Section 4.2, we extend the required properties to the case of $t > 1$ bit output hashing functions.

## 4.1. $t = 1$ bit output hashing

Let $f$ be a small fan-in boolean function $f : GF(2^m) \times GF(2^m) \rightarrow GF(2)$, i.e., $t = 1$. Consider the class of 1 bit output hashing function $F_f$ built by XORing the outputs of $f$ as follows. For every key $K$ and random input $R$ of length $l > 1$ symbols (each of $m$ bits), $K, R \in GF(2^{lm})$:

$$F_f(K, R) = \sum_{i=1}^{l} f(k_i, r_i) \tag{1}$$

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

It turns out that it is sufficient that the elementary function $f$ satisfies two simple conditions to prevent the formation of equivalent keys in $F_f$. The required conditions, specified in Theorem 4.1, are that for any two key symbols $k_i$, $k_j$, there should exist at least one

random symbol $r$ for which they are not distinguishable with $f$, and there should exist at least one random symbol $r'$ for which they are distinguishable.

*Theorem 4.1:* A hashing function $F_f$ has no indistinguishable keys (no equivalence keys) $iff$ the underlying elementary hashing function $f$ satisfies conditions (2). More formally stated:

$$\begin{gathered}\forall K \neq K' \in GF(2^{lm}) \exists R \in GF(2^{lm}) \text{ s.t.} \\ F_f(K,R) \neq F_f(K',R) \\ \Leftrightarrow \\ \left\{ \begin{array}{c} \forall k_i \neq k_j \in GF(2^m) \\ \exists r \in GF(2^m) \text{ s.t. } f(k_i,r) = f(k_j,r) \\ \exists r' \in GF(2^m) \text{ s.t. } f(k_i,r') \neq f(k_j,r') \end{array} \right\} \quad (2)\end{gathered}$$

See Appendix A for the proof.

**No Anti-Equivalence.** Note that, as an additional property, (2) also implies the following:

$$\begin{gathered}\forall K \neq K' \in GF(2^{lm}) \exists R \in GF(2^{lm}) \text{ s.t.} \\ F_f(K,R) = F_f(K',R).\end{gathered}$$

This property is important from a privacy perspective: Otherwise, if two keys $K, K'$ give different output on all different $R$s, the adversary would get with calls to the SENDTAG oracle, $\forall R, F(K,R) \neq F(K',R)$, and a third key, $K''$, gives different output to $K$ on all $R$s, $\forall R, F(K,R) \neq F(K'',R)$, then the adversary would know that $\forall R, F(K',R) = F(K'',R)$. So, $K'$ and $K''$ are the same keys. This would break privacy.

**Example: DPM-Function.** The DPM-function $f_{\mathrm{DPM}}$ was introduced and defined in [37] as the majority function $M$ of a key symbol XOR random symbol:

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

Unfortunately, this function does not satisfy the first condition of Theorem 4.1. A key symbol, i.e., the "triplet" $(k_{i,1}, k_{i,2}, k_{i,3})$, is always hashed to the same output as its opposite value, i.e., the key symbol $(\overline{k_{i,1}}, \overline{k_{i,2}}, \overline{k_{i,3}})$. Therefore, the resulting $F_{f_{\mathrm{DPM}}}$ function, $F_{f_{\mathrm{DPM}}} := \sum_{i=1}^{l} f_{\mathrm{DPM}} f(k_i, r_i)$, has equivalence classes. One can note, that by inverting an even number of symbols of a key (of a total of $l$ symbols), we obtain an equivalent key for $F_{f_{\mathrm{DPM}}}$. Thus, the key-space in DPM is divided into $2^{2m+1}$ equivalence-classes, each of $2^{m-1}$ equivalent elements.

**4.1.2. 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 $50\%$: on the one hand, this helps the reader to identify a tag in its database much more quickly. On the other hand, it is important to have $F_f$ produce the same output for two different keys with $50\%$ for each $R$, to protect against above mentioned Anti-Equivalence classes.

In the following, we show that the XOR structure of the considered class of hashing functions allows an *exponential amplification* of statistical (in-)distinguishability. The size of the set of $R$s which do *not* distinguish between two keys $K, K'$ shrinks exponentially quickly as a function of the *Hamming distance* of the two keys, $\mathrm{HD}(K, K')$. This result can also be viewed as an extension of the piling-up lemma [31] and Yao's XOR lemma [28].

*Theorem 4.2:* The set of random values for which two keys are indistinguishable is bounded as follows.

$$\begin{gathered}\exists \delta \in [0, \tfrac{1}{2}) \forall k \neq k' \in GF(2^m) \text{ s.t.} \\ \tfrac{1}{2} - \delta \leq Pr[f(k,r) \neq f(k',r)] \leq \tfrac{1}{2} + \delta \\ \Rightarrow \\ \forall K \neq K' \in GF(2^{lm}) \\ \tfrac{1}{2} - (2\delta)^{\mathrm{HD}(K,K')} \leq Pr[F(K,R) \neq F(K',R)] \\ \leq \tfrac{1}{2} + (2\delta)^{\mathrm{HD}(K,K')}\end{gathered}$$

See Appendix A for the proof.

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

**Example** For an elementary hashing function $f$ with a bounded bias $\delta = \frac{1}{8}$ (meaning that two key symbols result in equal and different values in at least $\frac{3}{8}$ of the cases), key symbol size $m = 4$, and key length $l = 32$ symbols, we obtain that the probability that two random keys are correlated is bounded within the interval: $[\frac{1}{2} - \sum_{i=0}^{l} \binom{l}{i} (1-\frac{1}{2^4})^{l-i} (\frac{1}{2^4})^i (2\delta)^{l-i}, \frac{1}{2} + \sum_{i=0}^{l} \binom{l}{i} (1-\frac{1}{2^4})^{l-i} (\frac{1}{2^4})^i (2\delta)^{l-i}]$. This is obtained by summing the product of the probability that two random keys have exactly $i$ equal symbols and the correlation bound of Theorem 4.2 with Hamming distance $i$. The resulting bounding interval is: $[\frac{1}{2} - 2^{-56}, \frac{1}{2} + 2^{-56}]$.

**4.1.3. Balanced Output.** Balanced output is an important statistical property that the $F_f$ needs to satisfy. Even a slight imbalance in the output would allow an adversary to characterize a tag based on the probability distribution of it's output. A tag can be characterized

by $(p_0, p_1)$, where $p_i$ is the probability of outputting value $i$. A safe function $F_f$ should have equal values of $p_i = \frac{1}{2}$.

The family of $F_f$ that we are considering converges towards a balanced output as the key length is increased. This derives from a similar argument as used in Theorem 4.2. From this theorem, 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 $\delta_0$ on its bias: $Pr[f(k,r) = 0 \in [\frac{1}{2} - \delta_0, \frac{1}{2} + \delta_0]$ (and similarly for 1 output). Let $K_i = (k_i, \ldots, k_i)$ be the $l$-symbol repetition of $k_i$. Then, $Pr[F_f(K,R) = 0 \in [\frac{1}{2} - (\delta_0)^{HD(K,K_i)}, \frac{1}{2} + (\delta_0)^{HD(K,K_i)}]$ (and similarly for output 1).

## 4.2. $t > 1$ bit output hashing functions

The theorems of the previous section can be generalized to hashing functions built using the same XOR structure but with $t > 1$ bits of output. In this case, the conditions on the elementary hashing function $f$ can be relaxed and the indistinguishability bounds further tightened. This will allow for a better tag identification (faster convergence).

Consider the class of $t$ bit output hashing function built by XORing the outputs of an elementary $t$ bit hashing function $f : GF(2^{mt}) \times GF(2^{mt}) \rightarrow GF(2^t)$ as follows. For every key $K$ and random input $R$ of length $l > 1$ symbols (each of $mt$ bits), $K = (k_1, \cdots, k_l), R = (r_1, \cdots, r_l) \in GF(2^{lmt})$:

$$F_f(K,R) = \sum_{i=1}^{l} f(k_i, r_i) \tag{3}$$

*Theorem 4.3:* A hashing function $F_f$ has no indistinguishable keys (no equivalence keys) $iff$ the underlying $t$ bit elementary hashing function $f$ satisfies conditions (4). More formally stated:

$$\left\{ \begin{array}{c} \forall k_i \neq k_j \in GF(2^{mt}) \\ \exists h_1, h_2 \in GF(2^t), h_1 \neq h_2 \\ \exists r \in GF(2^{mt}) \text{ s.t. } f(k_i, r) + f(k_j, r) = h_1 \\ \exists r' \in GF(2^{mt}) \text{ s.t. } f(k_i, r') + f(k_j, r') = h_2 \end{array} \right\} \tag{4}$$
$$\Rightarrow$$
$$\forall K \neq K' \in GF(2^{lmt}) \exists R \in GF(2^{lmt}) \text{ s.t. } F(K,R) \neq F(K',R)$$

See Appendix A for the proof.

Note that in the case of $t$ bit output functions, there is no need to make sure there is "no anti-equivalence" as in the case of 1 bit output, cf., Section 4.1.1. Even if the adversary would observe always different outputs, $\forall R, F(K,R) \neq F(K',R), F(K,R) \neq F(K'',R)$, from SENDTAG calls, he could not say anything on $K' \stackrel{?}{=} K''$. As outputs are now elements of $GF(2^4)$, each output can have $2^4$ different choices – and not 2 anymore as in the case of $GF(2)$.

Finally, Theorem 4.2 can also be extended to $t$ bit output functions. Let $f_i$ denote the restriction of $f$ to its $i^{th}$ output bit $(1 \leq i \leq t)$.

*Theorem 4.4:* The set of random values for which two keys are indistinguishable is bounded as follows.

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

See Appendix A for the proof.

**4.2.1. Balanced Output.** Similarly to Section 4.1.3, if the $f$ provides sufficient balance for each of the bits, XORing over a large number of $f$ outputs allows the bias to converge to 0. So, to insure output balance with high probability, we give the following sufficient condition that should hold for a non-negligible fraction of the key symbols $k_i$: for a given key symbol $k_i$, the dimension of the vector space spanned by the elements $\{f(k_i, r) | r \in GF(2^t)\}$, is equal to $t$.

## 4.3. Convergence-Rate and Anti-Impersonation Security

As mentioned before, 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)

**4.3.1. Convergence and Completeness.** As of Section 4.2.1, the output of $F_f$ is balanced. So, for each tuple of random numbers $R_i^1, \ldots, R_i^d$ and hash output $F_f(K, R_i^{a_i})$ that is sent during each of the $q$ rounds of the protocol from the tag, the probability that another key $K' \neq K$ in the reader's database $\mathbb{D}$ is removed is $P_{\mathrm{remove}}(t,d)$:

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

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 $(1 - P_{\text{remove}}(t,d))^q$:

$$\begin{aligned} n' &= (n-1) \cdot (1 - P_{\text{remove}}(t,d))^q \\ &= (n-1) \cdot (1 - (1 - \frac{1}{2^t})^d)^q \end{aligned}$$

A key is still valid after $q$ rounds implies that it complies to all $(R_i^{a_i}, F_f(K, R_i^{a_i}))$ tuples so far. So, convergence of identification of a single tag increases exponentially with the number of rounds $q$ and the output bit-size $t$ of $F_f$. However, convergence decreases exponentially with increasing $d$.

Please note that with the $F_f$ protocols a valid tag sending data to the reader will *never* be removed from $\mathbb{D}$. Thus, $F_f$ is (unconditionally) *complete*.

In Section D.1, we will present evaluation results for convergence rate of one $F_f$ implementation proposal.

**4.3.2. Statistical Impersonation.** 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. If the adversary tries to impersonate any tag by sending random data with SENDREADER in each of the $q$ rounds, the probability that he successfully impersonates is called $P_{\text{success}}(t,d,q,n)$. We can compute $P_{\text{success}}(t,d,q,n)$ 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:

$$\begin{aligned} P_{\text{invalid}}(t,d,q): &= 1 - (1 - P_{\text{remove}}(t,d))^q \\ &= 1 - (1 - (1 - \frac{1}{2^t})^d)^q, \end{aligned}$$

and finally

$$P_{\text{success}}(t,d,q,n) := 1 - P_{\text{invalid}}(t,d,q)^n$$

So, $n$ and $d$ exponentially weaken statistical security against impersonation, while $q$ and $t$ exponentially strengthen it.

Please refer to Section 6.1.2 for evaluation results regarding impersonation security of one $F_f$ implementation proposal. Also note that $d$ increases computational complexity for the adversary to compute a tag's secret key. This will be discussed in detail in Section 5.2.

# 5. Authentication protocols with good algebraic properties

In this section, we present a framework for algebraic reasoning about the $F_f$ family of protocols.

We first show a class of algebraic attacks that can be applied to the DPM-protocol and, based on this, show why the $F_f$-family is generally able to withstand these attacks. The proposed attack relies on linearization techniques and key compaction.

In Section 5.1, we show that any $t \geq 1$ bit output keyed-hashing function $f$ can be linearized using a potentially larger, linearized key. A careful analysis of the dimension of the vector space generated with the coefficients of the linearized form allows a compaction of the key. This result can be generalized to the larger class of $F_f$ keyed-hashing functions. Therefore, any instance of $F_f$ is equivalent to an inner-product of a random-vector and a key-vector each of length at most $l$ symbols. In Appendix B, we present as an example, how this can be exploited as an algebraic attack and show its theoretic efficiency against the DPM-protocol [37]. Further practical evaluations, in particular on a "secure" instance of $F_f$, can be found in Section 6.2.

## 5.1. Key Linearization and "Expansion-Compaction"

We first show that small fan-in keyed hash functions can be *expanded-compacted*, i.e., first expanded into a linearized expression and then compacted in a smaller expression that captures all security properties of the original keyed hash function.

*Lemma 5.1:* Any function $f : GF(2^{mt}) \times GF(2^{mt}) \rightarrow GF(2^t)$ can be linearized into:

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

See Appendix A 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 $l(s-1)+1$.

*Theorem 5.2:* Let $F_f(K,R)$ be a $t$ bit keyed-hashing function $(GF(2^{mt}) \times GF(2^{mt}))^l \rightarrow GF(2^t)$ defined as $F_f(K,R) = \sum_{i=1}^{l} f(k_i, r_i)$ where $f : GF(2^{mt}) \times GF(2^{mt}) \rightarrow GF(2^t)$ and $K = (k_1, \ldots, k_l), R = (r_1, \ldots, r_l) \in (GF(2^{mt}))^l$. $F_f(K,R)$ can be linearized into:

1) $F_f(K,R) = \sum_{j=1}^{L} ER_j EK_j$ where $L = ls$ or $L = l(s-1)+1$, ($s < 2^{mt}$, $ER_j = u(r_j) \in GF(2^t), EK_j = v(k_j) \in GF(2^t)$,
2) the vectors $(ER_1, \cdots, ER_L)$ generate a vector space of dimension $L$,

3) it is sufficient for an adversary to know the expanded key $(EK_1, \cdots, EK_L)$ to compute $F_f(K, R)$.

See Appendix A for the proof.

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

## 5.2. Algebraic attacks on d-choice low cost authentication 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 $K, R \in GF(2^{Lmt})$.

Let $K, R$ be to such vectors of $L$ symbols of $(mt)$ bits each. $K$ denotes a key and $R$ a random input. $(K \cdot R) \in GF(2^t)$ denotes the dot product. Following the introduction of $F_f$ in Section 3, consider the following protocol P:

**Algorithm 1**: Protocol P

**for** round $i = 1$ *to* $q$ **do**
- $R_i^1, R_i^2, ..., R_i^d \in_{\text{random}} GF(2^{lmt})$;
- $a_i \in_{\text{random}} [1, \ldots, d]$;
- $O_i = K \cdot R_i^{a_i}$;
- send$(R_i^1, R_i^2, \ldots, R_i^d, O_i)$;

The attack is now as follows: an adversary drawing one tag into quality-time can derive the following and only the following equations (because we are operating in a field):

$$\begin{array}{rcccl} (K \cdot R_1^1 - O_1) \cdot (K \cdot R_1^2 - O_1) & \cdots & (K \cdot R_1^d - O_1) & = & 0 \\ (K \cdot R_2^1 - O_2) \cdot (K \cdot R_2^2 - O_2) & \cdots & (K \cdot R_2^d - O_2) & = & 0 \\ \vdots & & \vdots & & \\ (K \cdot R_q^1 - O_q) \cdot (K \cdot R_q^2 - O_q) & \cdots & (K \cdot R_q^d - O_q) & = & 0 \end{array} \tag{5}$$

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

Generally, the adversary can compute $K$ by solving this system of equations. However, we will now show that with a careful choice of $l, m, t, d$ solving this system of equations becomes computationally infeasible.

Each equation of system (5) can be expanded in a sum of monomials of degree at most $d$. Each equation in round $i$ can be rewritten as:

$$\sum_{\substack{1 \le j \le d \\ 1 \le c_1 \le c_2 \le \cdots \le c_j \le l}} C_{i,c_1 c_2 \cdots c_j} k_{c_1} k_{c_2} \cdots k_{c_j} = (O_i)^d \tag{6}$$

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 $\Gamma$ the matrix of coefficients $C_{i,c_*}$, and $O$ is the vector of $(O_i)^d$. Ordering the linearized monomials according to a lexicographic order and renaming their vector as $Y$, we obtain the following equation:

$$\Gamma \cdot Y = O \tag{7}$$

To get the key bits $K$, the adversary computes $Y$ by inverting matrix $\Gamma$. The complexity of inversion depends on the number of (linearized) monomials. Theorem 5.3 bounds the total number of possible monomials $U$. $U$ represents the number of columns in matrix $\Gamma$.

*Theorem 5.3:*

$$\sum_{j=1}^{d} \binom{l}{j} \le U \le \sum_{j=1}^{d} \binom{l+j-1}{j} \tag{8}$$

See Appendix A 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 an exponential computational effort to compute the tag's key.

See Appendix A for the proof.

In conclusion, security of an $F_f$-instance against algebraic attacks rises exponentially with rising $d$.

However, with a small $d$ and small key sizes $lmt$, an adversary can use the above attack to easily compute all secret key bits. Thus, we applied this attack to the DPM-protocol which uses $d = 1$, $lmt = 117$. Due to space restrictions, you can find all the details of attacking DPM in Appendix B. We also experimentally evaluated this attack on DPM and a *secure* implementation proposal for $F_f$. The results of the evaluation can be found in Section 6.2.1.

## 5.3. Learning Parity with Noise (LPN)

One could argue that the adversary might look at each of the $t$ output bits of $F_f$, try to reduce this to a problem similar to Learning Parity with Noise (LPN [24]) and use an efficient method to compute key bits.

In principle, this is true, as sending $d$ random numbers $R$ and one output(-bit) $F_f(K, R_i^{a_i})$ on one of these $R$s in each round is similar to sending $R$s as well as output bits that are flipped with a certain bias. Yet, we are convinced that by carefully choosing an appropriate key size $lmt$ *and* picking a non-linear function $f$ will make attacks as in [27] infeasible: generally, the time- and memory complexity of these attacks rise with $2^{O(\frac{|K|}{\log |K|})}$, $|K|$ being the key size. However to

apply LPN-based attacks, the adversary will have to linearize a non-linear $f$ first. This will introduce new monomials such that the key size "virtually" increases – in the case of $F_f$, if you rewrite a non-linear $f$ as a linear one, $f(k,r) = \sum_{j=1}^{s} u_j(r)v_j(k)$, as shown above, key size $|K|$ will become $|K| = l \cdot s, s < 2^{mt}$. Given a non-linear $f$, this key size can become much higher than $|K| = (lmt)$, which makes LPN-attacks infeasible.

## 6. Implementation Proposal

One suitable instance of the $f$ function could be $f_\Delta(k_i, r_i)$ as proposed in the following. The *system parameters* are $d = 8$, $t = 4$, $m = 1$, $l = 64$, $q = 60$. The LFSR has an internal state of $\sigma = 64$ bits. This means it has to be initially seeded with a truly random 64 bit value – the initially exchanged $N_0$ and $R_0^d$ of Section 3.2.4 will be used as the LFSR's internal state. So, $f_\Delta$ works on inputs $k_i, r_i \in GF(2^4)$ and outputs an element in $GF(2^4)$, $f_\Delta : GF(2^4) \times GF(2^4) \rightarrow GF(2^4)$. The output of $f_\Delta$ is represented as 4 output bits, i.e., $f_\Delta(k_i, r_i) = \{f_{\Delta_1}, f_{\Delta_2}, f_{\Delta_3}, f_{\Delta_4}\}$. These output bits are separately defined as follows:

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

Again, $k_{i,j}, r_{i,j}$ denote the $j^{th}$ bit of the $i^{th}$ key-/random symbol. As you can see, $f_\Delta$ is non-linear in both, the bits of the key symbol and the bits of the output symbol.

Computation of $f_\Delta(k_i, r_i)$ can be implemented quite efficiently: per output bit of $f_\Delta$, 13 multiplications in $GF(2)$ (boolean AND) and 6 additions (boolean XOR) are required. Using figures as stated in [4], one output bit can be implemented in 34.5 NAND-gates, so $f_\Delta$ can be implemented using a total of 138 gates, which is very cheap.

To compute $F_{f_\Delta}$ out of the outputs of $f_\Delta$, we need a 4 bit register to store temporary data, which comes in at 48 NAND-gates. The LFSR with $\sigma = 64$ bits of state requires 768 NAND-gates. We can get along with only one LFSR to derive the $R$s and $N$s, if we use the LFSR alternately. Therefore, we have to store both, the current $R_i^d$ and $N_i, 1 \leq i \leq q$ of round $i$, in RAM. So, a total of 128 bits of RAM, i.e., 192 NAND-gates are required for this. Finally, $K$ and $K'$ need to be wired to $f_\Delta$, which uses a total of 512 gates.

The above, including gates for storage of the keys, sums up to a total of $1,658$ gates. This is far less than current implementations of strong hash functions alone, e.g., SHA-1 with already $\approx 10,000$ [6], not even taking storage of the secret key into account. We clearly confirm that our computation of $F_{f_\Delta}$ with $1,658$ gates is naive, because you typically need some kind of "multiplexing" logic around $F_{f_\Delta}$, but we estimate the total gate count to be $\approx 2,000$ for $F_{f_\Delta}$. In conclusion, $F_{f_\Delta}$ is feasible to implement on today's RFID-tags.

### 6.1. Properties of $F_{f_\Delta}$

Before we present the attack evaluation on first the DPM-protocol and finally the $F_{f_\Delta}$ implementation proposal, we show that $f_\Delta$ holds the statistical properties as of Section 4.2.

**6.1.1. Distinguishability and Balanced ouput.** The $f_\Delta$-function proposed above holds both properties of (4), Theorem 4.3, in Section 4.2. Consider any two key symbols $k \neq k' \in GF(2^4)$ of larger keys $K \neq K'$, then we will show that there exists at least one pair of random numbers $r, r'$ such that $f_\Delta(k, r) + f_\Delta(k', r) = h_1$, $f_\Delta(k, r') + f_\Delta(k', r') = h_2$, and $h_1 \neq h_2$.

- If $k \neq k'$, then they differ in at least one bit, i.e., the $i^{th}$ bit. Consider $r$ to be $r := (0,0,0,0)$, all bits of $r$ are zero. Looking only at the $i^{th}$ output bit $f_{\Delta_i}$ the following equation holds, $f_{\Delta_i}(k, r) + f_{\Delta_i}(k', r) = h_{1_i} = 0$. The $i^{th}$ bit of $h_1$ is 0.
- Consider $r'$ to be $r'_1 := 1, r'_j := 0, j \neq 1$, so the first bit of $r'$ is 1, the rest is 0. Looking only at the $i^{th}$ output bit $f_{\Delta_i}$ the following equation holds, $f_{\Delta_i}(k, r') + f_{\Delta_i}(k', r') = h_{2_i} = k_i + k'_i = 1 \neq h_{1_i}$.
  So, $h_1 \neq h_2$.

In conclusion $f_\Delta$ holds both properties of Theorem 4.3, and keys can generally be distinguished with $F_{f_\Delta}$ – there are no equivalence classes in $F_{f_\Delta}$.

To ensure a balanced output as of Section 4.2.1, we show that for a non-negligible fraction of key symbols

$k$, $f_\Delta(k,r)$ spans a vector space of dimension 4. Consider only the 4 key symbols $(1,0,0,0)$, $(0,1,0,0)$, $(0,0,1,0)$, $(0,0,0,1)$ – they make of $25\%$ of all key symbols. For such a key symbol $k$, all bits besides the $i^{th}$ are set to 0. We construct the 4 random symbols $r', r'', r''', r''''$: with $r'$, all bits are 0, and the $i^{th}$ bit is 1. With $r''$, all bits are 0, and the $(i^{th}-1)\ mod\ 4$ bit is 1. With $r'''$, all bits are 0, and the $(i^{th}-2)\ mod\ 4$ bit is 1. With $r''''$, all bits are 0, and the $(i^{th}-3)\ mod\ 4$ bit is 1. Consequently, $\{f(k,r'),\ f(k,r''),\ f(k,r'''),\ f(k,r'''')\}$ gives, $\{(1,0,0,0),\ (0,1,0,0),\ (0,0,1,0),\ (0,0,0,1)\}$ , a basis for a 4-dimensional vector-space.

Using the above system parameters for $F_{f_\Delta}$, and according to [27], the LPN-bias $\epsilon = 1 - (\frac{1}{d} + \frac{1}{2}\frac{d-1}{8}) \approx 44\%$. Linearization of $f_\Delta$ will introduce 3 new monomials per key symbol, such that $|K|$ rises to $64 \cdot 4 + 3 = 448$ bits. This will result in a time- and memory complexity between $\gg 2^{66}$ and $\ll 2^{130}$, cf., [27].

Due to space restrictions, and as the security of $F_{f_\Delta}$ is of primary interest here, we moved the evaluation of $F_{f_\Delta}$'s convergence rate and the required amount of data sent to the reader to Appendix D.

**6.1.2. Statistical Impersonation Security.** Simply converging to a single key is not yet sufficient from a security perspective. As of Section 4.3.2, the adversary might try to randomly send data to the reader and try to get accepted as one valid tag. Figure 3 therefore shows the probability $P_{\text{success}}$ that the adversary can successfully impersonate himself as at least one tag to the reader using a database size $n = 2^{16}$. For different $t$, the x-axis varies the number of rounds $q$. For comparison, Fig. 4.3.2 shows statistical impersonation security of DPM [37] with $d = 1$. You can see that with $t = 4$, higher impersonation security is achieved compared to DPM. Generally with rising $t$, impersonation becomes more difficult for the adversary, however, at the cost of more computations on the tag and the reader for the increased $t$. Rising $t$ will also imply more data to be exchanged between tag and reader. More on this in Appendix D.2.

To keep probability of statistical impersonation low, e.g., $P_{\text{success}} \approx 2^{-64}$, we therefore choose $q = 60$.

**6.1.3. LFSR Security.** We do not care about the secrecy of the LFSR's internal state, but require the LFSR only to protect $F_{f_\Delta}$ against chosen-plaintext attacks. The period of the LFSR must be long enough to produce $(lmt)\cdot d \cdot q = 122880$ bits of pseudo-random output: in each of the $q$ rounds, $d$ random numbers are required, each of $lmt$ bits. We assume an LFSR with a public, characteristic, primitive polynomial over

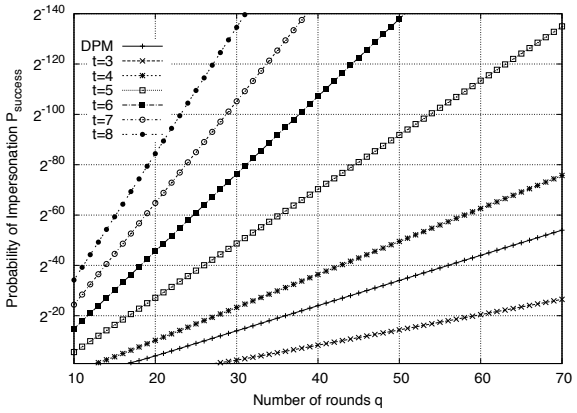



Figure 3. Impersonation security, $n = 2^{16}, d = 8$

$GF(2^{64})$ to produce a maximum-length period. An LFSR with $\sigma = \log_2 122880 + 1 \approx 17$ bits of internal state is already sufficient to produce such a period. However, to evade chosen-plaintext-exploits based on birthday-paradox, we set $\sigma = 64$. Together with $d = 8$, this will give sufficient security.

The reader will not accept $R_0^d = (0 \ldots 0)$, i.e., all $64$ bits of $R_0^d$ being zero as initial state of the LFSR to avoid trivial pseudo-randomness. If the tag draw such an LFSR, which is unlikely as it happens with probability $2^{-64}$, it will draw another random number. Likewise, the tag will not accept $N_0 = (0 \ldots 0)$ from the reader.

## 6.2. Attack Evaluation

Based on the above procedure, we implemented both, the algebraic attack as well as SAT-solving of 1.) the DPM protocol and 2.) $F_{f_\Delta}$ as proposed above.

**6.2.1. Attack on DPM.** The algebraic attack on DPM could be implemented in a straightforward fashion by setting up a system of linear equations as described in Section B and solving this using MatLab.

**SAT-Solving.** Recently, a lot of attention has been drawn to the use of SAT-solvers in cryptography, cf. [3, 10–12, 16, 17, 30, 32–34]. Therefore, we also used a SAT-solver, MiniSat [19], to attack DPM and $F_{f_\Delta}$. Basically, the idea of using a SAT-solver is to the convert equations that the adversary can set up to a Conjunctive Normal Form (CNF) and then use a SAT-solver to solve the CNF. Due to space restrictions, the details of applying SAT-solvers as well as an overview of the important SAT-parameters which appeared to have the best impact on performance can be found in Appendix C.

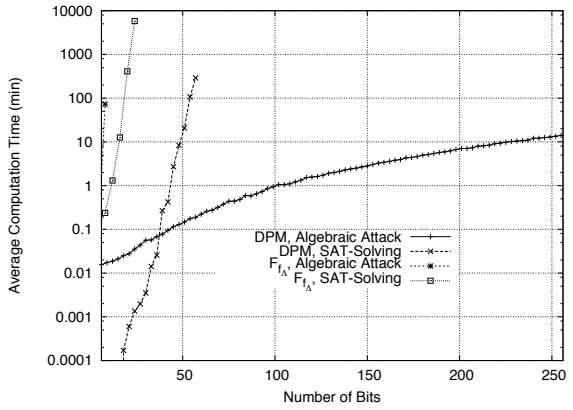



Figure 4. Time to break DPM and $F_{f_\Delta}$

Figure 4 shows the results of the attacks on DPM-privacy using an Intel Xeon, 1.86GHz, 16GByte of RAM (RAM was never an issue during the attacks, even with MiniSat). You can see the number of key bits on the x-axis and the required time (in minutes) to break privacy, i.e., an equivalent key, on the y-axis. Per sampling point, we ran 20 different instances. Relative standard deviation was $< 20\%$ with the algebraic attack and, due to its indeterministic nature, up to $80\%$ with MiniSat.

Even with $lmt = 256$ key bits, we could compute an equivalent key in a couple of minutes using the algebraic attack. MiniSat, however, appears to be inferior than our algebraic attack. With key sizes $> 57$ bits, MiniSat did not finish in a "reasonable" amount of time.

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, cf., [3] – as with DPM (sparsity/density $\beta \approx 50\%$).

In the case of the DPM-protocol, there is also the additional SHA-1 brute-forcing step of Section B.4 necessary to finally compute the right key (and not only an equivalent one). For deriving this "right" key $K$, based on one equivalent key $K'$ computed as above, the adversary has to try all possible $2^{l-1}$ equivalent keys of $K'$ – $K$ will be among those equivalent keys. So, $2^{l-2}$ SHA-1 computations on average are required for this step. With $lmt = 117$ key bits, $l = 39$, and $2^{37}$ 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 2 days. This shows that computing the secret key is totally feasible with the DPM-protocol, allowing not only to break privacy, but also impersonation.

**6.2.2. Attack on $F_{f_\Delta}$.** Both, the algebraic attack and SAT-solving was applied to $F_{f_\Delta}$. Results are also shown in Fig. 4. With key sizes $> 24$ bits, MiniSat did not finish in any reasonable amount of time. For the algebraic attack, we used Maple to set up the system of linear equations and solved this with MatLab. With 12 bits of key size, Maple had to generate $\approx 2^{12}$ linearized equations with a total number of $\approx 2^{12}$ monomials – it did not finish in any reasonable amount of time. For a 256 bit key, there will be $\gg 2^{32}$ monomials and equations in matrix $\Gamma$, cf., Section 5.2. The computational complexity of inverting this matrix is $\approx 2^{32 \cdot 3} = 2^{96}$. We claim that this will render attacks against the key to break authenticity or privacy infeasible.

While there might be optimizations to the algebraic attack, we believe they will not improve the computational complexity by a significant amount.

## 7. Related Work

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

For example in [41], the tags just sends the HMAC of the reader's challenge, keyed with the pairwise secret key, back to the reader. To protect against replay attacks, challenges need to be of ascending order, otherwise the tag rejects the challenge. So in addition to an HMAC, a non-volatile state is required on the tag which, in many scenarios, might not be feasible or simply too expensive for a tag. This protocol is also prone against DoS-attacks and has been shown to leak privacy, see [25].

The protocol of [46] uses a strong and expensive hash function and an HMAC-like computation for identification of a tag. This, however, does not protect against replay attacks from the adversary: as there is no nonce from the reader involved in the protocol, an adversary receives always the same response on subsequent SENDTAG calls with the same tag. This helps the adversary to identify a single tag breaking privacy, cf., [35].

Using a tree-based setup, [35] distributes $O(\log n)$ secret keys to each tag. This authentication with $O(\log n)$ complexity, i.e., "walking down" the tree of secrets until one tag is uniquely defined. Yet, besides requiring a complex hash function, the amount of memory required on a tag for this scheme might be

infeasible in many scenarios. Also, privacy of this scheme is weak, as shown in [2]. 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, [2] proposes the OSK/AO protocol using hash-chains, an idea originally proposed in [36]. Yet, OSK/AO is also known to leak privacy, cf., [25], requires an expensive hash function and a state on the tag.

With the HB+ protocol of [24], the tag XORs a biased "noise-bit" to the response before sending the response to the reader. The reader can then compute the tag's original response by solving the Learning Parity with Noise (LPN) problem. Yet, this scheme and also many variants are known to be insecure or leak privacy, cf., [23, 27]. Also note that with HB+ and all variants based on LPN-schemes [45], there will always be a potentially non-negligible probability that a valid tag gets rejected by the reader – HB+ is not complete.

There has been also some work on algebraic attacks against the DPM-protocol. In [42], an algebraic technique to compute $2/3$ of all key bits is proposed, [40] independently discovered the $2^{2m+1}$ equivalence classes in DPM. These papers can bee seen as special cases of the algebraic reasoning we are providing 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, which applies to all variants of DPM-like privacy-preserving authentication protocols.

## 8. Conclusion

In this paper, we presented the $F_f$ family of privacy-preserving authentication protocols together with an algebraic framework to indicate its security. $F_f$ uses a simple, round-based setup, where the tags sends the results of evaluating random numbers using with 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. $F_f$ 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 96 bit against algebraic attacks. Also, our experiments indicate that SAT-solving attacks are computationally infeasible. Generally, $F_f$ offers arbitrary, user-adjustable levels of soundness and identification rate, and even unconditional completeness.

## References


[1] Automated Reasoning Group UCLA. Rsat homepage, 2008. http://reasoning.cs.ucla.edu/rsat/.

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

[3] 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. In *ECRYPT workshop on Tools for Cryptanalysis*, Krakow, Poland, 2007. http://eprint.iacr.org/2007/024/.

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

[5] M. Burmester, B. de Medeiros, and R. Motta. Robust, anonymous rfid authentication with constant key-lookup. In *Proceedings of ACM Symposium on Information, Computer and Communications Security*, pages 283–291, Tokyo, Japan, 2008. ISBN 978-1-59593-979-1.

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

[7] S. Cook. The complexity of theorem-proving procedures. In *Proceedings of ACM Symposium on Theory of Computing*, pages 151–158, Shaker Heights, USA, 1971.

[8] S. Cook and D. Mitchell. *Satisfiability Problem: Theory and Applications*, chapter Finding Hard Instances of the Satisfiability Problem: A Survey, pages 1–17. American Mathematical Society, 1997. ISBN 0821804790.

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

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

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

[12] N. Courtois, K. Nohl, and S. O'Neil. Algebraic attacks on the crypto-1 stream cipher in mifare classic and oyster cards, 2008. http://eprint.iacr.org/2008/166.pdf.

[13] I. Damgård and M. Østergaard. Rfid security: Tradeoffs between security and efficiency. In

*Proceedings of RSA Conference*, pages 318–332, San Francisco, USA, 2006. http://eprint.iacr.org/2006/234.pdf.

[14] DIMACS. Satisfiability suggested format, 1993. http://www.satlib.org/Benchmarks/SAT/satformat.ps.

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

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

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

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

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

[20] N. En and N. Srensson. The minisat page, 2008. http://minisat.se/MiniSat.html.

[21] M. Feldhofer and C. Rechberger. A case against currently used hash functions in rfid protocols. In *Proceedings of OTM 2006 Workshops*, pages 372–381, Montpellier, France, 2006. ISBN 978-3-540-48269-7.

[22] M. Feldhofer and J. Wolkerstorfer. Strong crypto for rfid tags – a comparison of low-power hardware implementations. In *Proceedings of International Symposium on Circuits and Systems*, pages 1839–1842, New Orleans, USA, 2007. ISBN 1-4244-0921-7.

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

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

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

[26] A. Lenstra and E. Verheul. Selecting cryptographic key sizes. In *Proceedings of International Workshop on Practice and Theory in Public Key Cryptography*, pages 446–465, Melbourne, Australien, 2000. Springer Verlag. ISBN 3-540-66967-1.

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

[28] L. Levin. One-way functions and pseudorandom generators. In *Proceedings of the seventeenth annual ACM symposium on Theory of computing*, pages 363–365, Providence, USA, 1985. ISBN 0-89791-151-2.

[29] G. Lowe. A hierarchy of authentication specifications. In *Proceedings of Computer Security Foundations Workshop*, pages 31–43, Rockport, USA, 1997.

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

[31] M. Matsui. Linear cryptanalysis method for des cipher. In *Proceedings of Workshop on the theory and application of cryptographic techniques on Advances in cryptology, EUROCRYPT*, pages 386–397, Lofthus, Norway, 1994. ISBN 3-540-57600-2.

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

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

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

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

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

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

[38] SAT Competitions. The international sat competitions web page, 2008. http://www.satcompetition.org/.

[39] C. Sinz. Sat-race, 2008. http://www-sr.informatik.uni-tuebingen.de/sat-race-2008/results.html.

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

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

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

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

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

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

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

# Appendix A. Proofs of Sections 4 and 5

**Theorem 4.1**

*Proof:* Done separately for "$\Leftarrow$" and "$\Rightarrow$".

1.) "$\Leftarrow$": Proof by induction over $l > 1$.

For the **induction basis**, $l = 2$, we take $K \neq K'$. This implies that at least one symbol of $K$ is different from its corresponding symbol in $K'$. So, w.l.o.g., we assume $k_1 \neq k'_1$. Now, as of (2), $\exists r_1 : f(k_1, r_1) = f(k'_1, r_1)$ and $\exists r'_1 : f(k_1, r'_1) \neq f(k'_1, r'_1)$. Then, for all possible random symbols $r_2$ either $F_f(K, r_1 r_2) \neq F_f(K, r_1 r_2)$ or $F_f(K, r'_1 r_2) \neq F_f(K', r'_1 r_2)$ which proves the claim for $l = 2$.

As the **induction hypothesis**, the theorem is true for any $l \leq n$ where $n > 1$.

**Inductive Step:** We consider $K \neq K'$ each of size $l = n + 1$ symbols. Since $K \neq K'$, there is at least one symbol of $K$ which is different from its corresponding symbol in $K'$. For convenience, and w.l.o.g., we assume it is the first one, $k_1 \neq k'_1$. As of (2), this implies

$$\exists r_1 \in GF(2^m) \text{ such that } f(k_1, r_1) = f(k'_1, r_1)$$

$$\exists r'_1 \in GF(2^m) \text{ such that } f(k_1, r'_1) \neq f(k'_1, r'_1)$$

There are now two cases:

- either $(k_2 \dots k_l) = (k'_2 \dots k'_l)$, then
$$\forall (r_2, \dots, r_l) \in GF(2^{mn}) : \\ F_f(k_2, \dots, k_l, r_2, \dots, r_l) \\ = F_f(k'_2, \dots, k'_l, r_2, \dots, r_l)$$
which implies that
$$\forall (r_2, \dots, r_l) \in GF(2^{mn}) \\ F_f(K, R) \neq F_f(K', R),$$
where $R = (r'_1, r_2, \dots, r_l)$,
- or $(k_2, \dots, k_n) \neq (k'_2, \dots, k'_n)$, then we can apply the induction hypothesis on this $n$-length keys to get:
$$\exists (r_2, \dots r_l) \in GF(2^{mn}) \\ F_f(k_2, \dots, k_l, r_2, \dots r_l) \neq F_f(k'_2, \dots, k'_l, r_2, \dots r_l)$$
which implies that
$$F_f(K, R) \neq F_f(K', R) \text{ where } R = (r_1, r_2 \dots r_l).$$

2.) "$\Rightarrow$": We prove by contradiction. Assume, (2) does *not* hold. We will construct two keys $K_v, K'_v$ that will violate the first part of the theorem's equivalence. Assuming (2) not to hold means that there exists $k_1 \neq k'_1 \in GF(2^m)$ such that either

$$\forall r \in GF(2^m), f(k_1, r) = f(k'_1, r)$$

or

$$\forall r \in GF(2^m), f(k_1, r) \neq f(k'_1, r).$$

We separately present $K_v, K'_v$ for each case:

- $\forall r \in GF(2^m), f(k_1, r) = f(k'_1, r)$,
we define $K_v := (k_1, \dots k_1)$ and $K'_v := (k'_1, \dots k'_1)$. Since $k_1 \neq k'_1$, $K_v \neq K'_v$, but $\forall R \in GF(2^{lm}) : F_f(K_v, R) = F_f(K'_v, R)$.
- $\forall r \in GF(2^m), f(k_1, r) \neq f(k'_1, r)$,
we define $K_v := (k_1, k_1, k_3, \dots, k_l)$ and $K'_v := (k'_1, k'_1, k'_3, \dots k'_l)$ where $k_{\geq 3}, k'_{\geq 3}$ can be any symbol in $GF(2^m)$. Since $k_1 \neq k'_1$, $K_v \neq K'_v$ but since for all $r$, $r'$ in $GF(2^m)$, $f(k_1, r) + f(k_1, r') = f(k'_1, r) + f(k'_1, r')$, then $\forall R \in GF(2^{lm}) : F_f(K_v, R) = F_f(K'_v, R)$.

□

**Theorem 4.2**

*Proof:* By induction over $l$.

**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 $Pr[F(K,R) \neq F(K',R)] \in [\frac{1}{2} - \delta, \frac{1}{2} + \delta] \subseteq \frac{1}{2} - 2\delta, \frac{1}{2} + 2\delta]$.

**Inductive Step:** Let the **induction hypothesis** be that the theorem is true for $l$, and let $K, K'$ be two keys of length $l+1$ symbols. Denote by $K^l$ (resp. $K'^l$), the subkeys $(k_1, \ldots, k_l)$ and $(k'_1, \ldots, k'_l)$, respectively.

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

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

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

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

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

Since,

$$\epsilon_1 \in [-\delta, \delta], \epsilon_2 \in [-(2\delta)^{\mathrm{HD}(K^l,K'^l)}, (2\delta)^{\mathrm{HD}(K^l,K'^l)}],$$

and $\mathrm{HD}(K,K') = \mathrm{HD}(K^l,K'^l) + 1$, then we obtain the final result:

$$Pr[F(K,R) \neq F(K',R)]$$
$$\in$$
$$[\tfrac{1}{2} - (2\delta)^{\mathrm{HD}(K,K')}, \tfrac{1}{2} + (2\delta)^{\mathrm{HD}(K,K')}]$$

□

**Theorem 4.3**

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

$$\forall r_i \in GF(2^{mt}), \forall r_j \in GF(2^{mt}) \quad \left\{ \begin{array}{c} f(k_i,r_i) + f(k'_i,r_i) = \\ \sum_{j\neq i} f(k_j,r_j) + \sum_{j\neq i} f(k'_j,r_j) \end{array} \right\} \qquad (9)$$

By construction, we know that for function $f$, **if** $k_i \neq k'_i$, then $\exists h_1, h_2 \in GF(2^t), h_1 \neq h_2$, and $r, r' \in GF(2^{mt})$ such that, $f(k_i,r) + f(k'_i,r) = h_1$ and $f(k_i,r') + f(k'_i,r') = h_2$. However, from Equation 9, this implies $\sum_{j\neq i} f(k_j,r_j) + \sum_{j\neq i} f(k'_j,r_j)$ being equal to $h_1$ and $h_2$ simultaneously, which is impossible. Therefore, $k_i$ has to be equal to $k'_i$. □

**Theorem 4.4**

*Proof:* The proof is a straightforward generalization of Theorem 4.2, since it is sufficient that $F(K,R)$ differs from $F(K',R)$ on at least one out of $t$ bits of output. Note that a tighter bound on the distinguishability probability can be obtained, if we consider the output as a single symbol of $t$ bits. □

**Lemma 5.1**

*Proof:* Let us view $k \in GF(2^{mt})$ as a vector $(k_1, \cdots k_m)$ over $GF(2^t)$. The function $f(k,r)$ can be interpolated into a multivariate polynomial of degree $m(2^t-1)$ in the $k_i$. In the case of $t = 1$, this is the same as the Algebraic Normal Form (ANF) of $f$. One way to make it explicit would be to use Lagrange interpolation.

The Lagrange interpolation gives a sum of polynomials each of degree $m(2^t-1)$ in $k_i$. Developing the polynomial into a sum of monomials in the $k_i$, we obtain:

$$f(k,r) = \sum_{0\leq i_1,i_2,\cdots,i_m,i'_1,i'_2,\cdots,i'_m \leq 2^t-1} C_{i_1,i_2,\cdots,i_m,i'_1,i'_2,\cdots,i'_m} \cdot r_1^{i'_1} r_2^{i'_2} \cdots r_m^{i'_m} k_1^{i_1} k_2^{i_2} \cdots k_m^{i_m},$$

where $C_{i_1 i_2 \cdots i_m, i'_1, i'_2, \cdots, i'_m}$ is uniquely defined by function $f$.

Now consider $V$, the space generated by vectors

$$(C_{i_1 i_2 \cdots i_m, i'_1, i'_2, \cdots, i'_m} r_1^{i'_1} r_2^{i'_2} \cdots r_m^{i'_m}),$$
$$0 \leq i_1, i_2, \cdots, i_m, i'_1, i'_2, \cdots, i'_m \leq 2^t - 1$$

whose coordinates are the coefficients of monomials with non-zero degree. $V$ has $2^{mt} - 1$ coordinates. Let $s < 2^{mt}$ be the dimension of $V$ and $(u_1(r), \cdots, u_s(r))$ be a basis. Rewriting the $C_{i_1 i_2 \cdots i_m, i'_1, i'_2, \cdots, i'_m} r_1^{i'_1} r_2^{i'_2} \cdots r_m^{i'_m}$ as a linear combina-

tion of the basis $u_i(r)$, the polynomial interpolation of $f(k,r)$ can be compacted into a sum of $s$ terms:

$$f(k,r) = \sum_{j=1}^{s} u_j(r)v_j(k) \tag{10}$$

where $v_j(k)$ results from the linear combination of the monomials in 10.

By definition of the basis $(u_i(r))_{1\leq i\leq s}$, the vectors $(u_1(r), \cdots, u_s(r))$ generate a vector space of dimension $s$. Finally using Equation (10), it is clear knowledge of the key $(k_1, \ldots, k_m)$ is equivalent to knowledge of its expanded form $(v_1(k), \ldots, v_s(k))$. □

**Theorem 5.2**

*Proof:* 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 $v_j(k) = c$ is a constant function of $k$, (2) all function $v_j(k)$ are non-constants. These are the only two possible cases, because if two (or more) functions $v_j$, and $v_{j'}$ were constants, then:

$$\begin{aligned} u_j(r)v_j(k) + u_{j'}(r)v_{j'}(k) &= [u_j(r)c + u_{j'}(r)c'] \\ &= u''_j(r) \cdot v''(k) \end{aligned}$$

where $v''(k)$ is a constant function of the key, and more importantly, the $f$ would have been compacted to $(s-1)$. Therefore, at most one $v_j$ function is a constant.

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

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

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

In case (2),

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

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

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

**Theorem 5.3**

*Proof:*

- The number of monomials of degree $j, 1 \leq j \leq d$, where each key symbol is used at most once, e.g., $k_1 \cdots k_j$, is $\binom{l}{j}$. This number can already be reached in a field where the maximum order of an element is 1: for example in $GF(2)$. Thus, this is a lower bound.
- The number of monomials of degree $j, 1 \leq j \leq d$, where each key symbol is used at most $j$ times, e.g., $k_1$ is used 3 times in $k_1^3 \cdot k_2 \cdots k_{j-2}$, is $\binom{l+j-1}{j}$. In the general case, this is an upper bound.

In conclusion, we derive these bounds on $U$:

$$\sum_{j=1}^{d} \binom{l}{j} \leq U \leq \sum_{j=1}^{d} \binom{l+j-1}{j} \tag{11}$$

□

**Corollary 5.4**

*Proof:* A lower bound on $\binom{l}{d}$ is given by:

$$\binom{l}{d} = \prod_{i=0}^{d-1} \frac{l-i}{d-i} \geq \left(\frac{l}{d}\right)^d \tag{12}$$

Because:

$$\begin{gathered} \forall i, d, l : 0 \leq i \leq d-1 < l \\ \Rightarrow \\ (l-i) \cdot d \geq (d-i) \cdot l \end{gathered}$$

Inequalities (11) and (12) imply $U \geq \sum_{j=1}^{d} \binom{l}{j} \geq \sum_{j=1}^{d} \left(\frac{l}{j}\right)^j$, hence the number of monomials of the system rises exponentially with $d$. On a side note, inverting matrix $\Gamma$ will require $U$ linearly independent observations. Assuming that the random input values $R_i^j$ lead to a random matrix $\Gamma$, then using only a small number of extra equations, e.g., less than 5, we can obtain with overwhelming probability a maximum rank matrix $\Gamma$, cf., [9]. □

## Appendix B. Example: Attacking the DPM-Function

In the following, we will show that DPM-protocol as introduced in [37] will fail against the algebraic attack of Section 5.1. The adversary will be able to compute a tag's expanded-compacted key, which is sufficient to spoil authenticity, soundness, and privacy of the protocol.

First, we will "expand-compact" $f_{\mathrm{DPM}}$ of Section B.1. This results in an expression for the expanded-compacted version of the tag's original key. Sections B.1 and B.4 describe, how an adversary can finally compute this expanded-compacted key for one tag based only on the observations he makes during quality-time.

### B.1. Expanding-Compacting $f_{\mathrm{DPM}}$

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

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

All operations are in GF(2), and $M$ is the majority function. Informally, $F_{f_{\mathrm{DPM}}}$ can be described as the XOR of all the majorities of 3 key bits XOR 3 random bits, respectively.

The coefficients of the $f_{\mathrm{DPM}}(k_i, r_i)$ polynomial with unknowns $(k_{i,1}, k_{i,2}, k_{i,3})$ form vectors that generate 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

$$
\begin{gathered}
(r_{i,2}+r_{i,3})k_{i,1} + (r_{i,1}+r_{i,3})k_{i,2} + (r_{i,1}+r_{i,2})k_{i,3} \\
= \\
(r_{i,2}+r_{i,3})(k_{i,1}+k_{i,3}) + (r_{i,1}+r_{i,3})(k_{i,2}+k_{i,3}).
\end{gathered}
$$

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

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

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

Actual computation of one tag's expanded-compacted key can now be done by setting up and solving a system of linear equations as described in Section B.2.

### B.2. Computing Equivalent Keys

Generally, the DPM-protocol can be seen as an instance of the $F_f$ family of protocols with $d = 1, t = 1, m = 1$: 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 $A_i = ((\alpha_i)_1 = (r_i)_1 \oplus k_1, \dots, (\alpha_i)_l = (r_i)_l \oplus k_l) \in GF(2^{lmt})$, $(\alpha_i)_j \in GF(2^{mt})$ 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. The authors of [37] propose a key size of 117 bits for good security, i.e., $t = 1, m = 3, l = 39$.

First note as of Section 4.1.1 that there are indistinguishable, i.e., equivalent keys with $f_{\mathrm{DPM}}$ and $F_{f_{\mathrm{DPM}}}$. 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. Independently, [42] comes to the same conclusion, such that [42] can be viewed as a special case of the generic algebraic reasoning on the $F_f$ type of protocols that we do throughout Section 5.

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. After observing $w$ pairs, he generates the following system of equations with $K$ the *unknown* key bits:

$$
\begin{aligned}
F_{f_{\mathrm{DPM}}}(K, A_1) &= b_1 \\
F_{f_{\mathrm{DPM}}}(K, A_2) &= b_2 \\
&\dots \\
F_{f_{\mathrm{DPM}}}(K, A_w) &= b_w.
\end{aligned}
$$

The adversary receives only $q$ pairs per protocol run. As $w$ is going to be much larger than $q$, the adversary has to call LAUNCH and SENDTAG multiple times to get a total of $w$ pairs from the same tag.

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

For the $i^{th}$th observation $(A_i, b_i)$, the adversary has

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

Now, the adversary solves the following system of linear equations

$$\mathbb{M} \cdot EK = B.$$

$\mathbb{M}$ is the matrix of coefficients of $EK$. As of Theorem 5.2, $\mathbb{M}$ can only have rank $L = (2l + 1)$. Therefore, the adversary collects $w := L = (2l + 1)$ linear independent observations to make up a $(2l + 1) \times (2l + 1)$ matrix $\mathbb{M}$, with $j = \{1 \dots l\}$,

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

So, 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 [9]. The adversary queries the tag only a little bit more than $L$ times, e.g., $w := (L + 5)$ 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 efficiently, with polynomial complexity, using simple Gaussian elimination. This results in computing:
*(I)* $(k_{1,1} + k_{1,3}), (k_{1,2} + k_{1,3}), \dots, (k_{l,1} + k_{l,3}), (k_{l,2} + k_{l,3})$,
*(II)* $\sum_{j=1}^{l} M(k_{j,1}, k_{j,2}, k_{j,3})$.

**Equivalent Key.** Finally, to compute an *equivalent* key $K'$ of $T_{vtag}$'s original key, cf., Section 4.1.1, the adversary sets the first $(l - 1)mt$ third key bits of every key symbol to 0: $\{k_{1,3} := 0, k_{2,3} := 0, \dots, k_{l-1,3} := 1\}$. This yields key bits $\{k_{1,1}, k_{1,2}, k_{2,1}, k_{2,2}, \dots, k_{l,1}, k_{l,2}\}$.

Now all key bits, besides the last key bit $k_{l,3}$, are known. This last one is set to

- either 1, *iff* $\sum_{j=1}^{l-1} M(k_{j,1}, k_{j,2}) + M(k_{l,1}, k_{l,2}, 1) = \sum_{j=1}^{l} M(k_{j,1}, k_{j,2}, k_{j,3}) = (II)$,
- or 0, *iff* $\sum_{j=1}^{l-1} M(k_{j,1}, k_{j,2}) + M(k_{l,1}, k_{l,2}, 0) = (II)$

As you can see, the above attack against $f_{\mathrm{DPM}}$ does not only allow the adversary to compute the expanded-compacted key $EK$ of a tag's original key $K$, but even one equivalent key $K'$ of $K$. So, $EK_i = k'_i$ for $K' = k'_1, \dots, k'_l$.

Please refer to Section 6.2 for practical evaluation results, i.e., timing results, indicating the effectiveness

of the above attack, even for large key sizes.

### B.3. Breaking Privacy

According to the privacy definition of Section 2.2.2, the adversary should not have a higher, more than negligible probability of linking subsequent protocol runs of two tags to one tag compared to simply guessing whether it was the same tag or not, i.e., $50\%$.

The adversary calls DRAWTAG to receive one tag $T_{\text{vtag}}$ for quality time. During this quality time, he computes an equivalent key $K$ for $T_{\text{vtag}}$ using the above attack. Afterwards, he calls FREE $T_{\text{vtag}}$ to free this tag, he again calls DRAWTAG to receive a second tag $T_{\text{vtag}'}$ for quality time and computes an equivalent key $K'$ for $T_{\text{vtag}'}$. The adversary can now decide, whether $T_{\text{vtag}}$ and $T_{\text{vtag}'}$ are the same tags or not as follows.

If keys $K$ and $K'$ are not-equivalent, i.e., by inverting an even number of 3 bit symbols, he knows for sure (100%) that $T_{\text{vtag}}$ was not the same tag as $T_{\text{vtag}'}$. If the keys are equivalent, he guesses with $50\%$ whether $T_{\text{vtag}}$ is the same as $T_{\text{vtag}'}$.

With a key size of $(lmt)$ bits, $m = 1, t = 3$, there are $2^{l-1}$ equivalent keys in each of the $2^{2l+1}$ equivalence *classes* of the DPM-function. So, in $n\frac{2^{l-1}}{2^{3l}}$ out of $n$ cases, $T_{\text{vtag}'}$ gives a key $K'$ which is equivalent to $K$. In these cases, the adversary wins with $50\%$. In the remaining $n\frac{2^{3l}-2^{l-1}}{2^{3l}}$ out of $n$ cases, $T_{\text{vtag}'}$ is a non-equivalent tag, and the adversary wins with $100\%$.

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 $l > 1$, this becomes much better than a guessing adversary with $50\%$. In [37], the authors propose to use $l = 39$.

### B.4. Finding the "Right" Key

After computation of the expanded-compacted key for an $F_f$ instance, the user can successfully impersonate the tag or break privacy as explained in the previous section.

However, in the original DPM-protocol [37], there is an additional step required after $q$ identifications rounds with $F_{f_{\text{DPM}}}$ to protect against replay attacks. This is basically an 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 secure hash function, e.g., SHA-1. The reader compares the HMAC $h(K|R_1|N|K)$ with the HMAC computed using the key(s) he finally identified in his database during the preceding $q$ rounds. So, if the adversary computes only an equivalent key $K'$ to the tag's original key $K$, but not $K$ itself, the verification of above HMACs will fail for the reader, and the adversary's key will be rejected, RESULT will be 0.

However, the adversary can easily brute-force $K$ using $K'$: As of Section 4.1.1, two keys $K \neq K'$ are equivalent for $F_{f_{\text{DPM}}}$, if an even number of key symbols ("triplets") is inverted from $K$ to $K'$. Therefore, there are a total of $2^{l-1}$ keys equivalent to $K$ (including $K$ itself). To find out the correct $K$, the adversary has to additionally do a total of $2^{l-2}$ SHA-1 computations on average for all equivalent keys of $K'$. Each of generated equivalent key $K'_{\text{equiv}}$ 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. The adversary does this until RESULT$= 1$, i.e., if $K'_{\text{equiv}} = K$.

These SHA-1 computations during quality time have to be taken into account and have also been evaluated in Section 6.2.

## Appendix C. SAT-Solving

By using SAT-solvers, 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. So, the idea is to convert ANF-equations into boolean Conjunctive Normal Form (CNF) and then use a SAT-solver.

Although the k-SAT-problem with $k > 2$ is NP-complete, it does not mean that on average some instances cannot be computed quickly [7, 8]. Thus, there has been much research and competitions on fast, indeterministic SAT-solvers, cf., SAT-Race [39] or SAT-Competition [38]. 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.

### C.1. Conversion

In the following, we will present some of the important details of the conversion process and summarize its main issues. For further information, refer to the list of papers cited above.

**Linearization.** First of all, based on the plaintext and ciphertext observations an adversary can potentially make, linearized equations in ANF are set up.

Once more, linearization is important, because otherwise conversion of high-degree monomials into CNF would result in exponentially growing clause-length, which is believed to extremely increase SAT-solving computation time [3]. All resulting ANF equations are rewritten such that they are of the form $\sum_i u_i = 1$, for all linearized monomials $u_i$ that have coefficients $1 \in GF(2)$. The set $\mathbb{S}$ of left-hand-side algebraic expressions of the above equations will be converted to CNF.

**Grouping.** Before the actual conversion of $\mathbb{S}$, it is however suggested in, e.g., [33], to replace frequently re-appearing groups of monomials in $\mathbb{S}$ by introducing new variables. So in general, 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 result of this replacement is to increase the *sparsity* of expressions. Sparse expression, i.e., consisting only of a small amount of monomials, reduce CNF clause-length, total number of clauses, and therewith SAT-solving time.

**Elimination of variables.** It is also suggested to replace some of the variables in $\mathbb{S}$ by their definition [3]. 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 $\mathbb{S}$, where it occurs, by $1 + u_{i+1} + \cdots + u_j$. Therewith, $u_i$ is effectively eliminated from SAT-solving which should speed up the solving process. However, this *reduces* the sparsity of the expression and results in more and longer CNF-clauses. As a result, this *might* also have negative effects on computation time and has therefore to be evaluated, cf., Section 6.2.

**Guessing.** Experiments in [16, 32] indicate that it is worthwhile to systematically bute-force some of the variables, i.e., key bits, before using SAT-solvers. This is often called "guessing", which is misleading, as it has not much to do with a random choice of variable assignments. With $lmt$ being the total key size, the adversary randomly picks ($u \ll lmt$) key bits $k_i, \ldots, k_{i+u-1}$. The adversary iterates over all possible $2^u$ assignments for the key bits. In each iteration, the adversary adds $\{1 + k_i + a_i, \cdots, 1 + k_{i+u-1} + a_{i+u-1}\}$, with his current assignments $a_i, \cdots, a_{i+u-1} \in GF(2)$, to the set of algebraic expressions $\mathbb{S}$. 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 can be expected that $2^u$ invocations of a SAT-solver with $(lmt - t)$ key bits is faster than 1 invocation of a SAT-solver with $lmt$ key bits.

**Cutting.** Eventually, $\mathbb{S}$ is converted to a boolean CNF, using the convention $1 \equiv \text{True}$, $0 \equiv \text{False}$. As the conversion of XORs in ANF expressions has exponential complexity in terms of resulting CNF clause-length, the 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 [3]. Here, $k$ is called the *cutting number*. Without cutting, clauses would rise to $2^{n-1}$ literals [17], if $n$ is the number of monomials in an expression in $\mathbb{S}$ which quickly becomes a memory and computation problem.

Finally, as today's SAT-solvers, e.g., MiniSat [19, 20] or RSAT [1] require a special input format, DIMACS format [14], CNF-clauses have to be converted into this format.

This concludes the description of SAT-solving's theoretical background. Section 6.2 presents practical evaluation results on attacks against DPM as well as another, more secure instance of the $F_f$-family and compare the performance of SAT-attacks with those of the new algebraic attack.

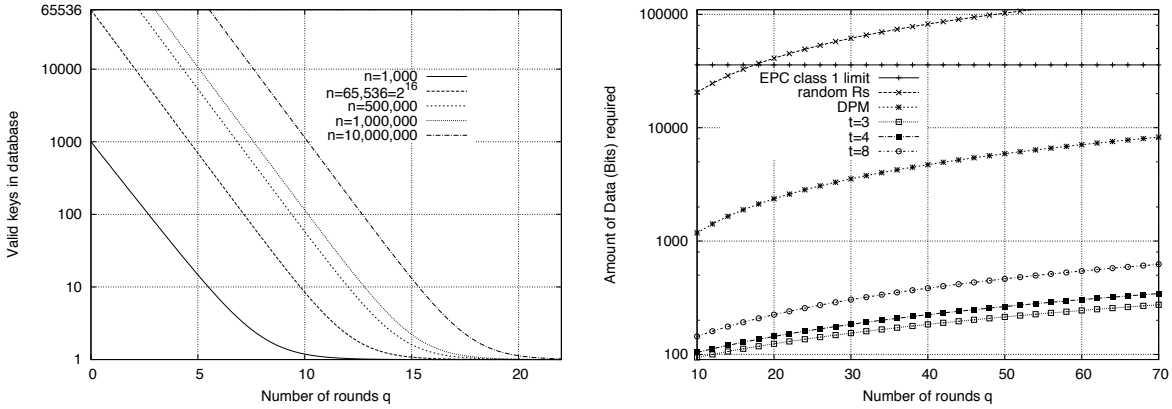
### C.2. Parameters used for attacking DPM

The optimal grouping threshold $p$ was $p = 77\%$, the optimal cutting number was $4$. We also shuffled [3] each DIMACS input to MiniSat 16 times, to get good performance results. Interestingly, neither guessing of variables, nor elimination of single variables had a positive impact on performance. We can only explain this by pointing at the very dense system of equations we have in DPM, e.g., "random matrices" with $\beta \approx 50\%$, while [3, 10] assumes density of equations to be very low with $\beta \leq 1\%$. We also *overdefined* the system of equations as proposed in [3], i.e., provided more than $lmt$ equations. MiniSat performed best with $2 \cdot lmt$ equations.

## Appendix D. $F_{f_\Delta}$ properties

### D.1. Convergence Rate

With $d = 8, t = 4$, we get $P_{\text{remove}}(4, 8) \approx 60\%$. Figure 5(a) shows convergence of $F_{f_\Delta}$ with various database sizes $n$. The x-axis varies the number of rounds $q$ within $F_{f_\Delta}$, the y-axis shows the number




(a) Convergence of $F_{f_\Delta}$, $t = 4$, varying $n$

(b) Amount of data for one protocol-run

Figure 5. Convergence and communication overhead for $F_f$ family of protocols, $d = 8$

of valid keys after $q$ rounds. So for example, with $n = 2^{16}$, the reader converges to a single entry in its database already after $q \approx 17$ rounds with a valid tag. This means that the reader has on average identified a valid tag after 17 rounds – resulting in completeness for $F_{f_\Delta}$. Even with huge databases, for example $10^7$ tags, $F_{f_\Delta}$ converges quite quickly in $q = 22$ rounds. As you can see, the reader on average identifies a tag exponentially fast in the number of rounds $q$.

Again, please note again that within this process a valid tag will never be removed. Although, it required some rounds to converge, $F_{f_\Delta}$ is always *complete*, contrary to related work, cf., HB+ [45] based protocols.

## D.2. Communication Overhead for $F_{f_\Delta}$

The amount of data that can be transferred during one protocol run is limited. In general, data rates between tag and reader vary and depend on many factors. We assume the data rate between tag and reader to be $\approx 70$ Kbps of, cf., [15], and the tag being in communication distance for $\approx 1$ second. So, the tag can at most transfer 70 Kbit to the reader. Without deterministic derivation of the $R$s, however, a lot of random data is sent from the tag: in round $i$, the tag sends $\{R_i^1, \ldots, R_i^d\}$, each of them $(lmt)$ bits, and it sends one $F_f(R_i^j, K)$ which is an additional $t$ bits.

Figure 5(b) shows the amount of data that has to be sent for one protocol-run with $q$ rounds. The horizontal represent today's EPC class-1 data volume limit of 70 Kbit and should not be crossed. Otherwise, an authentication would take longer as 1 second, i.e., a "user" carrying the RFID-tag would need to hold his tag closely to the reader for a couple of seconds.

As this might be unrealistic in many scenarios, where users are supposed to quickly swipe their tags closely to the reader, sending $d$ random numbers in each round can not be afforded. You can see the required amount of data for $d = 8$ random numbers sent per round in Figure 5(b) entitled with "random Rs". This curve represents $F_{f_\Delta}$ as above using $t = 4, lmt = 256$. You can see that already for $q \approx 17$ there is too much data required to be sent during one second of authentication, but $t = 4, q = 17$ does not provide much security as shown above, cf., Fig. 4.3.2.

The other curves, $t = 3, 4, 8$ represent the amount of volume transferred between tag and reader when using deterministic derivation of the $R$s, e.g., with the LFSR. As you can see, even with $q = 60$ rounds as proposed and $t = 4$, $F_{f_\Delta}$ requires less data volume than the EPC class-1 limit, by orders of magnitude. In conclusion, the amount of data sent with $F_{f_\Delta}$ is feasible in today's RFID-systems.

To put things into perspective, Fig. 5(b) also shows the amount of data required for DPM (here, $d = 1$, 117 key bits). DPM requires more data than $F_{f_\Delta}$ for any number of rounds $q$.