Hardware Efficient Authentication based on Random Selection

Frederik Armknecht, Matthias Hamann, Matthias Krause

University of Mannheim
{armknecht,hamann,krause}@uni-mannheim.de

Abstract: Lightweight authentication protocols based on random selection were introduced as an alternative design paradigm besides the usage of lightweight block ciphers and the principle of adding based noise. However a comparatively large key length and the use of involved operations made a hardware-efficient implementation a challenging task. In this work we introduce the $(n, k, L)^{80}$-protocol, a variant of linear authentication protocols which overcomes these problems, and analyze its security against all currently known, relevant passive and active attacks. Moreover, we present an implementation of our protocol for FPGAs and ASICs using Verilog and discuss its efficiency w.r.t. generally accepted costs metrics. The respective numbers show that the $(n, k, L)^{80}$-protocol is a viable alternative to existing solutions and is, for example, well suited for the implementation on passive RFID tags.

1 Introduction

1.1 Lightweight Authentication Protocols

Devices of extremely limited computational power like (passive) radio frequency identification (RFID) tags are used in practice to a rapidly growing extent, a trend commonly referred to as ubiquitous computing. One of the major use-cases for such pervasive devices are authentication solutions, e.g., access control for buildings or cars, electronic passports or even human-implantable chips providing sensitive medical information about a person. Consequently, the search for lightweight authentication protocols became an important topic in cryptography during the last years with high relevance for academia and industry.

Today, one can distinguish three main approaches for constructing lightweight authentication protocols:

1. protocols which use lightweight block ciphers like PRESENT [BKL$^+$07], KATAN and KTANTAN [DDK09] as basic cryptographic operations,
2. protocols which employ the well-researched principle of adding biased noise to a secret linear function,
3. protocols which are based on the principle of random selection, being the most recent of all three paradigms.

Concerning approach 1.), it has to be stated that very convincing proposals for lightweight block ciphers as PRESENT, KATAN and KTANTAN do exist which have been analyzed in a large number of papers (e.g., [BKL$^+$07], [DDK09], [KMNP11], [Å11]). However such protocols are less flexible with respect to scalability than other approaches.

Concerning approach 2.), the security of these kinds of (HB-type) protocols w.r.t passive attackers can be reduced to the widely accepted hardness of the learning parity in the presence of noise (LPN) assumption. A severe drawback of these protocols is that presumably secure parameter combinations imply large amounts of transmitted data. Together with the small available bandwidth in RFID communication, this may add up to authentication times that are unacceptable for many applications. A further major problem is that almost all variants, i.e., [JW05, GRS08, BC08] were broken by active man-in-the-middle (MITM) attacks, e.g., see [GRS05, OOV08, FS09]. The only exception we are aware of are the proposals in [KPC$^+$11] and [HKL$^+$12], which are both based on modified variants of the original LPN problem. However, even the latter (more efficient) one of these is "targeting lightweight tags that are equipped with (small) CPUs" [HKL$^+$12] due to the fact that the protocol's computational complexity would violate common timing constraints on cheaper, less powerful hardware (which is targeted in this work).

Approach 3.), i.e., the principle of random selection, implies that the secret key K consists of a small collection of L linear mappings. The prover (e.g., an RFID tag) computes responses to challenges $a \in GF(2)^n$, $n \in \mathbb{N}$, by choosing one of these functions $f \in K$ and replying with $f(a')$, where a' depends on a in a way we are going to specify concretely as part of section 2. The first protocols of this kind were the CKK-protocols given in [CKo08]. Further protocols based on the principle of random selection include the F_f-protocols in [BKM$^+$09] and the Linear Protocols in [KS09]. The most important and still unbroken suggestion of the latter type is the $(n,k,L)^{++}$-protocol also given in [KS09]. It has been proved that the $(n,k,L)^{++}$-protocol is resistant w.r.t. to a wide family of active MITM attacks. Moreover, the security of $(n,k,L)^{++}$-protocols can be reduced to the complexity of the problem of learning unions of linear subspaces (LULS problem). In analogy to HB-type protocols, the security of Linear Protocols is thus based on the assumption that it is impossible to solve the LULS problem in a more efficient way. In [KH11] the best approach known so far for solving the LULS problem has been given. Its effort is dominated by the cost of inverting a matrix of size n^L.

1.2 Our Contribution

In previous works about $(n,k,L)^{++}$-protocols, two problems w.r.t. efficiency were left open for future research and prevented this type of protocol from being practically used so far: Firstly, the large key length resulting from the need to specify the afore-mentioned set of secret linear functions. Secondly, certain operations deemed necessary in order to achieve MITM-security were still too demanding in hardware.

The $(n,k,L)^{80}$-protocol introduced in this paper aims at solving both of these problems. In particular, we are able to reduce the key length to a feasible size of 80 bits and show that the security reductions presented in [KS09] and [KH11] still apply to a large extent. Moreover, all operations used in the $(n, k, L)^{80}$-protocol can be realized efficiently in hardware. In order to show this, we created an actual implementation for FPGAs and ASICs using the hardware description language Verilog and present the corresponding efficiency metrics, which indicate that the suggested protocol is a viable alternative to prevalent block cipher based constructions.

Table 1 in appendix A provides an overview of our results, which are explained in further detail as part of section 4. Most notably, the suggested protocol can be safely realized at costs below 1,300 GEs (Gate Equivalents) without succumbing to the attacks described in section 3 (or having been broken by other means). The in-depth description of our design in subsection 2.2 will explain why choosing smaller parameters reduces the area costs as well as the total number of needed clock cycles (despite an increase in rounds) while leaving the communication complexity unchanged.

2 A Proposal for a Hardware Efficient Linear Protocol

2.1 The $(n, k, L)^{++}$-Protocol

We first recall the definition of $(n, k, L)^{++}$-protocols as it was suggested in [KS09]. In the original specification, it is a one-round challenge-response authentication protocol, whose symmetric key consists of a small number L of injective linear functions $F_1, \ldots, F_L : \{0,1\}^n \longrightarrow \{0,1\}^{n+k}$. Based on theoretical considerations as well as experimental results (see also section 3), the following parameter sizes were suggested: $n = 128, k = 32, L = 8$. Figure 1 in appendix B depicts an instance of the $(n, k, L)^{++}$-protocol for a verifier Alice (RFID reader) and a prover Bob (RFID tag).

The authentication process is initiated by Alice who chooses uniformly and at random a challenge $a \in_U GF(2)^{\frac{n}{2}}$, $a \neq \mathbf{0}$, and sends it to the prover. Likewise, the prover chooses a random nonce $b \in_U GF(2)^{\frac{n}{2}}$, $b \neq \mathbf{0}$, of the same length, randomly picks one of the L secret linear functions $F_1, \ldots, F_L$, and responds $w = F_l\left(f\left(a, b\right)\right)$. The non-linear bijective connection function $f : GF\left(2^{\frac{n}{2}}\right)^* \times GF\left(2^{\frac{n}{2}}\right)^* \longrightarrow GF\left(2^{\frac{n}{2}}\right)^* \times GF\left(2^{\frac{n}{2}}\right)^*$, where $GF\left(2^{\frac{n}{2}}\right)^*$ denotes $GF\left(2^{\frac{n}{2}}\right) \setminus \{0\}$, is defined by $f\left(a, b\right) = \left(ab, ab^3\right)$. It is included for thwarting a certain class of man-in-the-middle (MITM) attacks (see subsection 3.3). In order to verify the prover's response, the reader Alice first checks whether w belongs to one of the L n-dimenensional subspaces $V_1, \ldots, V_L$ of $GF(2)^{n+k}$, which are the images of the corresponding injective linear functions $F_1, \ldots, F_L$. Given that $w \in V_l$ holds, Alice subsequently computes $(\tilde{a}, \tilde{b}) = f^{-1}\left(F_l^{-1}(w)\right)$. Finally, if $\tilde{a}$ equals the initial challenge a, Alice will accept the prover's valid response.

The security of this protocol with respect to passive attackers can be reduced to the hardness of the Learning Unions of Linear Subspaces (LULS) problem. In a nutshell, the LULS problem is to learn specifications of the subspaces $V_1, \ldots, V_L$ from independently and uniformly chosen random samples from $\bigcup_{l=1}^{L} V_l$. The complexity of this problem was studied in [KS09] for $L = 2$ and for general L in [KH11]. The best solving algorithm known so far is an algebraic attack approach which takes time proportional to the time needed to invert a regular matrix of size $O(n^L)$.

A crucial open problem of the original $(n, k, L)^{++}$-protocol was the apparently large key length. As each of the L secret n-dimensional, injective linear functions $F_1, \ldots, F_L : \{0,1\}^n \longrightarrow \{0,1\}^{n+k}$ can be expressed as a distinct $((n+k) \times n)$-matrix over $GF(2)$, in total $((n+k) \cdot n) \cdot L$ bits would need to be stored permanently, which is clearly infeasible for suggested parameter sizes like $n = 128$, $k = 32$ and $L = 8$.

Moreover, for such parameters sizes, even the "simple" non-linear connection function $f(a, b) = \left(ab, ab^3\right)$ induces a big computational overhead in the form of several multiplications over $GF\left(2^{\frac{n}{2}}\right)^*$. Analogously, lookup tables, e.g., in order to efficiently compute b^3 would become very expensive in terms of space.

2.2 The $(n, k, L)^{80}$-Protocol

In this section we introduce the new $(n, k, L)^{80}$-protocol to overcome the two problems mentioned above. In short, the basic ideas are summarized as follows:

- To shorten the key length, the linear functions are no longer randomly sampled and stored but are computed from a smaller seed.
- To lower the effort of the connection function, we replace it by several subfunctions which compute the same functionality but on a smaller domain.

One consequence of these modifications is that the protocol needs to be executed several times. In the following, we first explain all modifications in further detail and provide a description of the overall protocol afterwards.

Shortening the Key Length. The basic idea is to take a keystream generator G that uses a seed of length $m + M$ to (pseudorandomly) generate the $((n+k) \cdot n) \cdot L$ key bits characterizing the secret linear functions $F_1, \ldots, F_L$. In particular, we suppose that $L = 2^M$ for a small $M \in \mathbb{N}$ (e.g., $M = 4$) and represent each index l, $1 \leq l \leq L$, as an M-bit string $\tilde{l}$. Hence, given a secret symmetric session key $\kappa = (\kappa_1, \ldots, \kappa_m)$, the entries of the matrix corresponding to F_l are certain bits from the key stream produced by G on $(\kappa, \tilde{l})$. Striving for a lightweight construction, it might be tempting to employ a single linear feedback shift register (LFSR) as a simple bitstream generator G. However, we show in subsection 3.2 that allowing the matrices of $F_1, \ldots, F_L$ to be generated by a keystream of small linear complexity opens the door to an algebraic attack which is much more efficient than the afore-mentioned algorithm from [KH11].

Splitting the Connection Function. Another open problem was to reduce the cost introduced by the so-called connection function $f : GF\left(2^{\frac{n}{2}}\right)^* \times GF\left(2^{\frac{n}{2}}\right)^* \longrightarrow GF\left(2^{\frac{n}{2}}\right)^* \times GF\left(2^{\frac{n}{2}}\right)^*$, which is applied to the random values $a, b \in GF\left(2^{\frac{n}{2}}\right)$, $a, b \neq \mathbf{0}$, before they are fed into one of the L secret linear functions $F_1, \ldots, F_L$. Instead of using $f(a,b) = \left(ab, ab^3\right)$ as a connection function (and thus multiplications over $GF\left(2^{\frac{n}{2}}\right)$), in the new $(n,k,L)^{80}$-protocol, we compute $f(a,b) = \left(\left(a_1 b_1, a_1 b_1^3\right), \ldots, \left(a_{n/8} b_{n/8}, a_{n/8} b_{n/8}^3\right)\right)$ where $a_i, b_i \in GF\left(2^4\right)$, $a_i, b_i \neq 0$, are obtained by splitting a and b into blocks of 4 bits, respectively. The practical security implications of this modification, which reduces the number of valid challenge-nonce pairs (a,b) from $\left(2^{n/2}-1\right)^2$ to $\left(2^4-1\right)^{n/4}$, are mainly confined to the active attack discussed in section 3.3.

Further Modification. On contrast to the (practically infeasible) $(n,k,L)^{++}$-protocol, it is necessary to run the $(n,k,L)^{80}$-protocol multiple times in order to obtain sufficient resistance w.r.t. certain MIMT attacks. The reason for this is twofold: Firstly, for efficiency reasons, the implementation outlined in section 4 uses challenge-nonce tuples of length $n = 64$ or smaller as compared to $n = 128$ suggested for $(n,k,L)^{++}$. Secondly, one has to compensate for the afore-mentioned decrease of valid inputs (a,b) resulting from splitting up a and b into blocks of size 4 bits each as part of the modified connection function. In subsection 3.3 we show that these modifications lead to an upper bound of $2^{-n/4}$ (e.g., 2^{16} for $n = 64$) for the success probability of a certain MITM attacker to convince an honest verifier to accept an illegitimate response. As this success probability is too large for practical applications, one has to run the protocol at least two times, which would, e.g., lead to an upper bound of $2^{-n/2}$ due to the fact that the rounds can be considered independent w.r.t. the details of this type of attack. As a final modification to the original $(n,k,L)^{++}$-protocol, we introduce a (publicly known) bit-wise permutation σ to the n-bit result of $f(a,b)$. Note that in terms of hardware efficiency, such a bit-wise permutation comes at practically no cost as it is realized simply through wires and does not involve any additional gates.

Protocol Description. The $(n,k,L)^{80}$-protocol proceeds according to the scheme described in subsection 2.1. Again, the process is initiated by the verifier Alice, who chooses some $a \in_U GF(2)^{\frac{n}{2}}$ uniformly and at random and sends it to the prover Bob. Bob then also randomly chooses some $b \in_U GF(2)^{\frac{n}{2}}$ and l, $1 \leq l \leq L$, and answers with

$$w = F_l\left(\sigma\left(f(a,b)\right)\right) = F_l\left(\sigma\left(\left(a_1 b_1, a_1 b_1^3\right), \ldots, \left(a_{n/8} b_{n/8}, a_{n/8} b_{n/8}^3\right)\right)\right)$$

as described previously. Remember that, in order to allow for inverting $f(a,b)$ as part of the verification, only challenges a and nonces b satisfying $a_i, b_i \neq 0$ for $i \in \left\{1..\frac{n}{8}\right\}$ are allowed by the protocol. The verification step of Alice is exactly the same as for the $(n,k,L)^{++}$-protocol, see subsection 2.1. Please note that while a might be known to an adversary eavesdropping on the communication between Alice and Bob, b is kept strictly secret by the prover and is only used to compute $\sigma\left(f(a,b)\right)$. We will denote $\sigma\left(f(a,b)\right) = (x_0, \ldots, x_{n-1}) = x$ during the following steps. Let us now consider an example where $L = 16$ (public known) and the prover Bob randomly and secretly chooses $l = 6$. Consequently, Bob would have to compute $F_6(x)$ and send the resulting $(n+k)$-bit string to the verifier Alice. As outlined previously, in order to achieve a feasible key length for the $(n, k, L)^{80}$-protocol, we deploy a keystream generator G with 80-bit initial state to specify the secret linear functions $F_1, \dots, F_L$. The common secret shared between the verifier Alice and the prover Bob comprises of $80-log_2(L) = 76$ bits $\kappa = (\kappa_1 \dots \kappa_{76})$. In order to derive a specification of F_6 (needed to compute $F_6\,(x)$), Bob concatenates the bit string $\tilde{l} = 0101$, the binary representation of the 6th function, and the 76 common secret bits, yielding the corresponding 80-bit seed $\kappa||\tilde{l}$ of G. The resulting keystream $z = z_0 z_1 z_2 \dots$ enters the computation of the $n + k$ bits $y_0 \dots y_{n+k-1}$ of $F_l(x)$, i.e. the final authentication token, as follows:

$$\begin{aligned} y_0 &= z_0 \cdot x_0 \oplus \dots \oplus z_{n-1} \oplus x_{n-1}, \\ &\dots \\ y_{n+k-1} &= z_{(n+k-1)\cdot n} \cdot x_0 \oplus \dots \oplus z_{(n+k-1)\cdot n+(n-1)} \oplus x_{n-1}. \end{aligned} \tag{1}$$

It should be noted that in the course of computing, e.g., y_0, only a one-bit-wide register is needed in hardware, i.e., firstly, $z_0 \cdot x_0$ is computed and stored, then $z_1 \cdot x_1$ is XORed, and so on, until $z_{n-1} \cdot x_{n-1}$ has been added and y_0 is finally ready to be transmitted to the verifier. This is an important property as registers are especially costly in terms of area and power consumption, so that their use should be restricted to an absolute minimum when designing lightweight cryptographic protocols. While we trade in clock cycles for a reduction of area (and thus power) in several parts of the $(n, k, L)^{80}$-protocol (see, e.g., the above paragraph), the hardware implementation outlined in section 4 also contains measures to reduce the time complexity where possible. Most notably, the block-wise evaluation of $f\,(a, b)$ can be performed in parallel to the initialization phase of the generator G without inducing any additional hardware cost. This allows to start computing the first token bit y_0 instantly once G (e.g., a self-shrinking generator based on an LFSR) is ready.

3 Security Analysis

3.1 General Remarks

In this section we analyze the security of the $(n, k, L)^{80}$-protocol, which is, as pointed out previously, in fact a variant of the $(n, k, L)^{++}$ authentication protocols where some modifications have been made for improving the hardware efficiency. In a nutshell, these modifications are (cf. Sec. 2):

- The linear functions F_l are not randomly chosen but generated from a common seed, using a bitstream generator G.
- The connection function has been broken down into several subfunctions which all realize in principle the same function, but restricted to a smaller domain.

Consequently, we investigate if and to what extent these modifications impact the security of the $(n, k, L)^{80}$-protocol in comparison to the security of the $(n, k, L)^{++}$-protocol.

With respect to the latter, we want to point out that the best attacks known so far against $(n, k, L)^{++}$-type protocols are a passive algebraic attack (cf. [KH11]) and an active MITM attack (cf. [KS09]).

3.2 Impact of Using a Generator G

In this subsection we investigate the security impact if the linear functions F_l are not randomly chosen but derived from a bitstream generated by a generator G. To this end, in appendix C, we demonstrate that if G is *weak* (more precisely, where the generated bitstream exhibits a small linear complexity) the whole protocol becomes vulnerable to the passive algebraic attack from [KH11]. This shows the necessity for stronger generators. In fact, we will argue now, using a standard hybrid argument, that using G does not imply any significant change in security if G produces a pseudorandom bitstream (as it is commonly expected from secure keystream generators).

Security Reduction for Pseudorandom-Bit-Generators G. Next we consider the case that G is instantiated by a bitstream generator which produces a bitstream (z_i) of pseudorandom bits given a seed $\alpha \in GF(2)^\ell$. More precisely, let $q = ((n + k) \cdot n) \cdot L$ be the number of bits that characterize the secret linear functions $F_1, \ldots, F_L$. For simplicity, we assume that the first q outputs of G eventually define the linear functions. Now, let G be a (q, t, ε)-secure pseudorandom bit generator and let $\vec{z} = (z_0, \ldots, z_{q-1})$ be a bitstring of length q. This means that for any algorithm D which accepts q bits input and which runs in time t, it holds

$$\begin{aligned} &|\Pr\left(1 \leftarrow D(\vec{z}) | \vec{z} \leftarrow G(\alpha), \alpha \in_U GF(2)^\ell\right) \\ &\qquad -\Pr\left(1 \leftarrow D(\vec{z}) | \vec{z} \in_U GF(2)^q\right)| \leq \varepsilon. \end{aligned} \tag{2}$$

Using a standard argument, one can show that the success probability of any attacker A against the protocol using G deviates at most by ε from the success probability if the linear functions are characterized by uniformly and independently sampled bits. More precisely, let A denote any attacker against the $(n, k, L)^{80}$-protocol which runs in time t at most. We define a corresponding security experiment Exp_A which is equal to 1 if A has been successful. Moreover, we consider two games. In $\mathsf{Game\ 0}$, the linear functions F_l have been determined by the output of G based on a secret seed, while in $\mathsf{Game\ 1}$, they are characterized by independently and uniformly sampled bits. The latter corresponds to a situation where the linear functions are randomly chosen, as suggested in the context of $(n, k, L)^{++}$-protocols. It follows from (2) that

$$|\Pr\left(\mathsf{Exp}_A = 1 | \mathsf{Game\ 0}\right) - \Pr\left(\mathsf{Exp}_A = 1 | \mathsf{Game\ 1}\right)| \leq \varepsilon. \tag{3}$$

Otherwise A could be used directly as a distinguisher for telling apart random bits from outputs of G, hence violating (2). Summing up, if G is a (q, t, ε)-secure pseudorandom bit generator for a sufficiently small value ε, we can practically restrict to the case that the linear functions are randomly chosen. In particular, using generator G yields at most a negligible difference w.r.t. the security against the passive algebraic attack (cf. [KH11]) and the active MITM attack (cf. [KS09]) in comparison to the $(n,k,L)^{++}$-protocols.

Of course the parameters need to be chosen carefully. The choice of n provides infeasibility of exhaustive key search, while the choice of L has to guarantee resistance against the algebraic attack approach summarized in subsection 3.2. The choice of the parameter k, in turn, must ensure that the following probabilities are negligibly small:

1.) the probability that one of the functions F_l, $l = 1, \ldots, L$, is not injective,

2.) the probability that a random vector $w \in GF(2)^{n+k}$ falls into $\bigcup_{l=1}^{L} V_l$,

3.) the probability that a random vector $w \in V_l$ falls into $V_l \cap V_k$ for some $k \neq l$,

4.) and the probability that there is a pair of secret subspaces V_l, V_k, $1 \leq l \neq k \leq L$, such that $\dim(V_l \oplus V_k) < n + k$.

For an estimation of the corresponding probabilities (for randomly chosen linear functions) see [KS09] and [KH11].

3.3 Impact of Splitting the Connection Function

In this section, we investigate any impact on the security caused by splitting the connection function. As the algebraic attack [KH11] (see Sec. 3.2 for a summary) is independent of the connection function, the resistance against this attack remains unchanged. However, as we elaborate below, the situation is different for the active MITM attack explained in [KS09]. This MITM attack has been called (x, y)-equality attack and was used to break, e.g., the CKK^2-protocol by Cichoń, Klonowski and Kutyłowski. We show that splitting the connection function implies an (for the attacker better) upper bound of about $2^{-n/4}$ for the success probability of this kind of attack against $(n,k,L)^{80}$-protocols. One consequence is that for the parameters suggested in section 4 (e.g., $n = 64$, $k = 32$, $L = 16$), a reasonable level of security can be reached by running the protocol a few times (e.g., four independently executed rounds would reduce the upper bound to $2^{-16 \cdot 4}$ if $n = 64$). The aim of an (x, y)-equality attacker Eve is to generate two messages $w \neq w' \in \mathrm{GF}(2)^{n+k}$ and to efficiently test by MITM-access to the protocol if w and $w \oplus w'$ belong to the same linear subspace V_l for some $l \in [L]$. As shown in [KS09], such an attack can be used to efficiently compute specifications of the subspaces $V_1, \ldots, V_L$. Eve works in three phases:

1. Send a message $y \in \mathrm{GF}(2)^N$ to Bob and receive $w' = F_l(f(y, b'))$.
2. Observe a challenge $a \in \mathrm{GF}(2)^N$ sent by Alice.
3. Compute a value $x = x(y, w', a) \in \mathrm{GF}(2)^N$, send it to Bob, receive the message $w = F_r(f(x, b))$ and send $w \oplus w'$ to Alice.

The success probability of the attack is given by the probability that Alice accepts $w \oplus w'$ if $l = r$.

The connection function of the $(n, k, L)^{80}$-protocol yields provable security against (x, y)-equality attacks. From now on we identify $\{0, 1\}^4$ with the finite field $K = \mathrm{GF}(2^4)$ and denote by $+, \cdot$ the addition an multiplication in K. Let the function value $f(a, b)$ for all $a, b \in \{0, 1\}^{n/2}$ be defined by $f(a, b) = \left(\left(a_1 b_1, a_1 b_1^3\right), \ldots, \left(a_{n/8} b_{n/8}, a_{n/8} b_{n/8}^3\right)\right)$, where $a_i, b_i \in K$, $i = 1, \ldots, n/8$, are obtained by partitioning a and b into blocks of 4 bits, respectively. Note that, according to the specification of the $(n, k, L)^{80}$-protocol (see subsection 2.2), the prover Bob will only reply to challenges a (and choose nonces b) which satisfy $a_i, b_i \neq 0$ for all $i = 1, \ldots, n/8$. Thus, Alice accepts a message w with $F_l^{-1}(w) = \left((u_1, v_1), \ldots, \left(u_{n/8}, v_{n/8}\right)\right)$ in inner state $a \in (K^*)^{n/8}$ if for all $i = 1, \ldots, n/8$ it holds that $\left(a_i^{-1} u_i\right)^3 = a_i^{-1} v_i$, which is equivalent to $u_i^3 = a_i^2 v_i$.

Theorem 1 *The success probability of an (x, y)-equality attack against the $(n, k, L)^{80}$-protocol is at most* $0.2^{n/8}$.

Proof: See Appendix D.

4 Hardware Efficiency

Considered Metric. In order to assess the efficiency of our hardware implementation and to allow for comparing the results with other cryptographic protocols, generally accepted metrics are needed. Most authors consider *area*, *throughput* and *power consumption* the most important factors and, depending on the nature of their protocol, focus on one of them with respect to optimization (which is usually a trade-off). In the case of the suggested $(n, k, L)^{80}$-protocol, we will focus on area for the following reasons. Clearly, the throughput of our protocol is dominated by the speed of the keystream generator G (see, e.g., equation 1 in subsection 2.2). Given that G produces one bit every c clock cycles, computing the $n + k$ token bits $y_0 \ldots y_{n+k-1}$ takes $c \cdot n \cdot (n + k)$ clock cycles (after the initialization phase). Hence, for reasonably small values of c (e.g., $c = 4$ on average for the self-shrinking generator used in our implementation), the bottleneck w.r.t. to timing is not the speed at which the token is generated but rather the extremely limited transmission bandwidth of (passive) RFID tags. With respect to power consumption, the low clock rates of, e.g., 100 KHz in the context of RFID applications, lead to a situation where the static part of the power consumption becomes dominant. As this, in turn, can be decreased directly by minimizing the number of needed gates, it emphasizes our approach to focus on a low area footprint. In terms of hardware design, measuring the area size is a complicated task. First of all, one needs to distinguish between two main target platforms for our authentication protocol: *Field Programmable Gate Arrays* (FPGAs) and *Application-specific Integrated Circuits* (ASICs). As the names already suggest, FPGAs are integrated circuits designed to be configured by a customer or a designer after manufacturing, whereas ASICs are integrated circuits customized for a particular use, rather than intended for general-purpose use. Both worlds have rather different ideas of area, which will be explained, as needed, in subsections 4.1 and 4.2, respectively, along with the specific target devices and tools used.

Chosen Keystream Generator. The protocol can be instantiated with any secure and hardware efficient keystream generator. For our implementation we decided to use the self-shrinking generator [MS94] on top of a mere MLLFSR (a maximal length LFSR, i.e., an LFSR with a primitive feedback polynomial). While only few additional gates are needed to implement the logic of the self-shrinking generator as compared to a simple MLLFSR (see section 4.2), the security benefit is enormous. The best currently known attacks against self-shrinking generators are a time memory attack by Mihaljevic (1996) [Mih96] and an OBDD-attack by Krause (2001) [ZKL01]. However, we do not see how to use these attacks in order to realize a non-trivial attack against the $(n, k, L)^{80}$-protocol. In particular, the fact that no algebraic attacks are known makes the self-shrinking generator seem especially suited for our context.

General Remark. Before presenting our implementation results for FPGAs and ASICs in the following two sections, we would like to share our impression that despite the multitude of allegedly lightweight authentication protocols which have been suggested so far (see, e.g., [JW05], [BC08], [GRS08] or, more recently, [KPC$^+$11]), none of the respective works contains any of the cost metrics listed above. In contrast, newly introduced lightweight block ciphers like PRESENT [BKL$^+$07] or KATAN [DDK09] always come with an extensive assessment of their real-world hardware cost. This is why in subsections 4.1 and 4.2, we compare the numbers of $(n, k, L)^{80}$ rather with those of PRESENT, assuming its use as part of the following simple authentication scheme: Both parties share the encryption/decryption key of PRESENT as a common secret and in order to prove his identity, the prover needs to correctly encrypt a random nonce provided the verifier. However, we hope that our hardware results presented in this paper will encourage other designers of lightweight authentication protocols to also go trough the process of actually implementing their schemes in order to allow for easier efficiency comparison in the future.

4.1 The $(n, k, L)^{80}$-prover on FPGAs

In order to allow for an easy comparison on FPGAs, we implemented our authentication protocol for the *Spartan3 XC3S400* (Package FG456, Speed -5) from Xilinx [Xil13], using Verilog and their ISE Design Suite 14.1 for synthesis. Please refer to table 1 in subsection 1.2 for a concise overview of the corresponding implementation results. Clearly, while actually aimed at ASICs, the area footprint of the $(n, k, L)^{80}$-protocol is also very moderate on FPGAs, e.g., it amounts to 139 FFs (Flip Flops) and 177 4-input LUTs (Look-Up Tables) in the case of $n = 32$ and $k = 16$. This compares to 152 FFs and 253 LUTs given in [Pos09] for the encryption unit of PRESENT-80 on the same platform and an *espresso*-optimized [UoC94] S-box. Without this latter optimization, the respective numbers are 154 FFs and 350 LUTs.

Overall, these numbers suggest, that our preliminary implementation of the $(n, k, L)^{80}$-protocol is already a viable alternative to the optimized code of PRESENT when it comes to authentication schemes. The subsequent section will even reinforce this impression.

4.2 The $(n, k, L)^{80}$-prover on ASICs

ASICs are a typical component in the context of RFID applications. They are (ex ante) tailored to a very specific need and subsequently produced in large quantities, allowing for low unit cost and making them perfectly suitable for pervasive devices like RFID tags. In the field of ASICs, area is usually measured in μm^2. However, as area requirements in μm^2 strongly depend on the used standard cell library (and, thus, the fabrication technology), it is common to use a metric called Gate Equivalents (GEs) instead. In short, one GE is equivalent to the area of a two-input drive-strength-one NAND gate. This at least allows for a rough comparison of area requirements derived using different technologies. As in the case of FPGAs, we again chose a technology closely related to the one which was used to derive the corresponding results for PRESENT-80 in [Pos09] to allow for a fair comparison. Synthesis and analysis was performed using *Cadence Encounter RTL Compiler RC11.24* [Cad13] and the employed technology library was *UMCL18G212T3*.

For the given technology and the parameter choice $(n, k, L) = (32, 16, 32)$, an ASIC implementation of the $(n, k, L)^{80}$-prover requires 1281 GEs. This is well below 2000 GEs, commonly referred to as the maximum area available on an RFID device for cryptographic purposes. In comparison, according to [Pos09], implementing PRESENT-80 in a fast round-based manner takes 1570 GEs, which is about the same size as the 1565 GEs needed to realize the $(n, k, L)^{80}$-prover for large parameters, i.e., $(n, k, L) = (64, 32, 16)$.

5 Conclusion

We introduced the $(n, k, L)^{80}$ authentication protocols, which are a modification of the already investigated $(n, k, L)^{++}$-protocol made in order to improve hardware efficiency. Our implementations confirm the suitability of our protocols for use cases which demand for low hardware size, e.g., RFID systems, making them interesting for practice. Moreover, the fact that the security of these protocols relies on a different paradigm than the alternative approaches based on block ciphers or the LPN problem, i.e., the random selection of secret functions, makes this kind of protocols likewise interesting for the cryptography community. One major modification is that the internal linear functions are generated by a bitstream generator G in order to save memory. Our analysis shows that, while using a single publicly known LFSR renders the protocol insecure, deploying a secure pseudorandom bit generator is sufficient. However, it remains an open question whether other, intermediate approaches, e.g., using an NLFSR or possibly keeping the LFSR-specifications secret, might be viable alternatives. In general, given that the underlying problem is relatively new, its hardness and possible connections to other problems need to be investigated further. Moreover, despite the popularity of lightweight authentication protocols, it turns out that only few actual implementations exist, which, in addition, commonly shift the problem (and cost) of generating random bits on the prover's side to a higher level of the tag's hardware (as we do in this work, too). This aspect represents an important next step towards a better understanding and comparison of existing design approaches.

References

[BC08] J. Bringer and H. Chabanne. Trusted-HB: A low cost version of HB$^+$ secure against a man-in-the-middle attack. *IEEE Trans. Inform. Theor.*, 54:4339–4342, 2008.

[BKL$^+$07] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. H. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *Proceedings of Cryptographic Hardware and Embedded Systems (CHES) 2007*, volume 4727 of *LNCS*, pages 450–466. Springer, 2007.

[BKM$^+$09] E.-O. Blass, A. Kurmus, R. Molva, G. Noubir, and A. Shikfa. The F_f-Family of Protocols for RFID-Privacy and Authentication. In *5th Workshop on RFID Security, RFID-Sec'09*, 2009.

[Cad13] Cadence. Encounter RTL Compiler, 2013. `http://www.cadence.com/products/ld/rtl_compiler/`.

[CKo08] J. Cichoń, M. Klonowski, and M. Kutyłowski. Privacy Protection for RFID with Hidden Subset Identifiers. In *Proceedings of Pervasive 2008*, volume 5013 of *LNCS*, pages 298–314. Springer, 2008.

[DDK09] C. De Cannière, O. Dunkelman, and M. Knežević. KATAN and KTANTAN – A Family of Small and Efficient Hardware-Oriented Block Ciphers. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems (CHES) 2009*, volume 5747 of *LNCS*, pages 272–288. Springer, 2009.

[FS09] D. Frumkin and A. Shamir. Untrusted-HB: Security Vulnerabilities of Trusted-HB. Cryptology ePrint Archive, Report 2009/044, 2009. `http://eprint.iacr.org`.

[GRS05] H. Gilbert, M. J. B. Robshaw, and H. Sibert. Active Attack against HB$^+$: A provable secure lightweight authentication protocol. *Electronic Letters*, 41:1169–1170, 2005.

[GRS08] H. Gilbert, M. J. B. Robshaw, and Y. Seurin. HB$^\#$: Increasing the Security and Efficiency of HB$^+$. In *Proceedings of Eurocrypt 2008*, volume 4965 of *LNCS*, pages 361–378, 2008.

[HKL$^+$12] Stefan Heyse, Eike Kiltz, Vadim Lyubashevsky, Christof Paar, and Krzysztof Pietrzak. Lapin: An Efficient Authentication Protocol Based on Ring-LPN. In Anne Canteaut, editor, *Fast Software Encryption*, volume 7549 of *Lecture Notes in Computer Science*, pages 346–365. Springer Berlin Heidelberg, 2012.

[JW05] A. Juels and S. A. Weis. Authenticating Pervasive Devices with Human Protocols. In *Proceedings of Crypto 2005*, volume 3621 of *LNCS*, pages 293–308. Springer, 2005.

[KH11] M. Krause and M. Hamann. The Cryptographic Power of Random Selection. In *Proceedings of SAC 2011*, volume 7118 of *LNCS*, pages 134–150. Springer, 2011.

[KMNP11] S. Knellwolf, W. Meier, and M. Naya-Plasencia. Conditional Differential Cryptanalysis of Trivium and KATAN. In *Proceedings of SAC 2011*, volume 7118 of *LNCS*, pages 200–212. Springer, 2011.

[KPC$^+$11] E. Kiltz, Krzysztof Pietrzak, David Cash, Abhishek Jain, and Daniele Venturi. Efficient authentication from hard learning problems. In *Proceedings of Eurocrypt 2011*, volume 6632 of *LNCS*, pages 7–26. Springer, 2011.

[KS09] M. Krause and D. Stegemann. More on the Security of Linear RFID Authentication Protocols. In *Proceedings of SAC 2009*, volume 5867 of *LNCS*, pages 182–196. Springer, 2009.

[Mih96] M. Mihaljević. A faster cryptanalysis of the self-shrinking generator. In *Information Security and Privacy*, volume 1172 of *LNCS*, pages 182–189. Springer, 1996.

[MS94] Willi Meier and Othmar Staffelbach. The Self-Shrinking Generator. In *Advances in Cryptology - EUROCRYPT '94, Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, May 9-12, 1994, Proceedings*, volume 950 of *Lecture Notes in Computer Science*, pages 205–214. Springer, 1994.

[OOV08] K. Ouafi, R. Overbeck, and S. Vaudenay. On the Security of HB$^{\#}$ against a Man-in-the-middle Attack. In *Proceedings of Asiacrypt 2008*, volume 5350 of *LNCS*, pages 108–124. Springer, 2008.

[Pos09] Axel York Poschmann. Lightweight Cryptography: Cryptographic Engineering for a Pervasive World, 2009.

[Å11] M. Ågren. Some Instant- and Practical-Time Related-Key Attacks on KTANTAN32/48/64. In *Proceedings of SAC 2011*, volume 7118 of *LNCS*, pages 213–229. Springer, 2011.

[UoC94] Berkeley University of California. Espresso, 1994. `http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/index.htm`.

[Xil13] Xilinx. Programmable Devices and Design Resources, 2013. `http://www.xilinx.com/`.

[ZKL01] E. Zenner, Matthias Krause, and Stefan Lucks. Improved Cryptanalysis of the Self-Shrinking Generator. In *Information Security and Privacy*, volume 2119 of *LNCS*, pages 21–35. Springer, 2001.

A Implementation Results

Parameters				FPGA		ASIC		Comm. (Tag)
				Slice-		GEs		Bits
n	k	L	Rnds.	FFs	LUTs	(μm^2)	Clk.	IN/OUT
64	32	16	2	175	205	1,565 (15,147)	31,210	64/192
32	16	32	4	139	177	1,281 (12,402)	11,540	64/192

Table 1: An overview of the results of our hardware implementation for FPGAs (Xilinx Spartan3 XC3S400) and ASICs. *Clk.* denotes the total number of clock cycles needed on the prover's side to perform a full authentication consisting of multiple rounds. Due to the nature of the self-shrinking generator used in our implementation, the timing values in the respective column may vary slightly for different keys. (see section 2 for an in-depth explanation of the given parameters and section 4 for further details relating hardware costs)

B The $(n, k, L)^{++}$-protocol

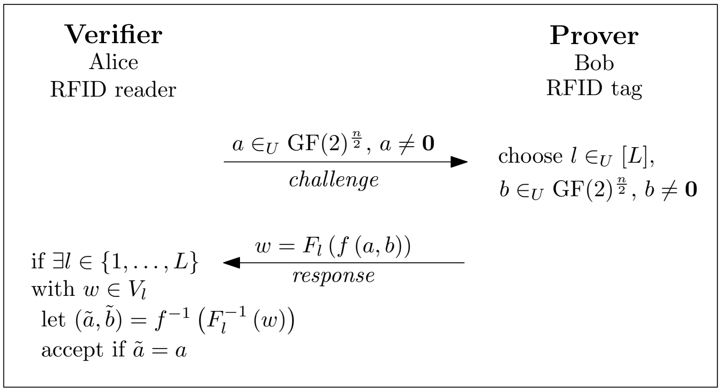
Verifier
Alice
RFID reader
Prover
Bob
RFID tag
$a \in_U \mathrm{GF}(2)^{\frac{n}{2}}$, $a \neq \mathbf{0}$
challenge
choose $l \in_U [L]$,
$b \in_U \mathrm{GF}(2)^{\frac{n}{2}}$, $b \neq \mathbf{0}$
$w = F_l\left(f\left(a, b\right)\right)$
response
if $\exists l \in \{1, \ldots, L\}$
with $w \in V_l$
let $(\tilde{a}, \tilde{b}) = f^{-1}\left(F_l^{-1}\left(w\right)\right)$
accept if $\tilde{a} = a$

Figure 1: An instance of the $(n, k, L)^{++}$-protocol (cf. [KS09]).

C Algebraic Attack for Weak Generators.

In the following, we present an efficient algebraic attack if the generator G produces a bitstream with a short, known linear complexity. For simplicity, we consider the case that

G is realized by an MLLFSR (a maximal length LFSR, i.e., an LFSR with a primitive feedback polynomial). Observe however that the same attack works against any G which produces a bitstream with low, known linear span.

We first recall the Passive Algebraic Attack against $(n, k, L)^{++}$-type protocols presented in [KH11]. Building on this, we then show that generating the function matrices F_l by a single LFSR is not a good idea as this allows for a much more efficient attack. Due to the pseudorandomness assumption regarding the generator G formulated above, we assume that it is not possible to algebraically attack the $(n, k, L)^{80}$-protocol in a significantly more efficient way than the original $(n, k, L)^{++}$-protocol.

Let $F_1, \dots, F_L : GF(2)^n \longrightarrow GF(2)^{n+k}$ denote the secret key consisting of L injective $GF(2)$-linear mappings, where k, n, L are appropriately chosen. During a passive key recovery attack the attacker tries to compute specifications of these function on the basis of pairs (x, y), where x is randomly and uniformly chosen from $GF(2)^n$ and it holds that $y = F_l(x)$ for some index l, which is randomly and uniformly chosen from $\{1, \dots, L\}$.

The passive attack described in [KH11] is based on choosing appropriate parameters λ, μ such that $\lambda \cdot \mu = n + k$, considering the secret functions F_l as vectors of μ component functions mapping from $GF(2)^n$ into $GF(2)^\lambda$, identifying $GF(2)^\lambda$ with the finite field $K = GF(2^\lambda)$, and computing the component functions by means of the following algebraic attack approach:

Suppose we are given secret functions $f_1, \dots, f_L : K^n \longrightarrow K$ and we want to compute specifications of these functions on the basis of known plaintext pairs (x, y), where x is randomly and uniformly chosen from $\{0, 1\}^n \subseteq K^n$ and it holds that $y = f_l(x)$ for some secret index l, which is randomly and uniformly chosen from $\{1, \dots, L\}$.

We were done if we could compute the values $x_{i,l} = f_l(e_i)$ for $i = 1, \dots, n$ and $l = 1, \dots, L$, where $e_i \in K^n$ denotes the standard vector having one at position i and zero at all other positions.

Note that each known plaintext pair (x, y) yields a degree-L equation in the $x_{i,l}$-variables of the form

$$\prod_{l=1}^{L} \left(\bigoplus_{i \in I} x_{i,l} \oplus y \right) = 0,$$

where $x = \bigoplus_{i \in I} e_i$.

In [KH11] it is shown that systems built of degree-L equations of this kind can be solved by a nontrivial application of the technique of linearization, which implies to solve a system of linear equations over $O(n^L)$ variables.

We analyze now the case that the $((n + k) \cdot n) \cdot L$ key bits characterizing the secret linear functions $F_1, \dots, F_L$ are generated by one MLLFSR of length $m + M$, where $L = 2^M$. Remember that the secret symmetric key $\kappa = (\kappa_1, \dots, \kappa_m)$ and the M random bits $l_1, \dots, l_M$ forming the binary representations of the indexes $l \in \{1, \dots, L = 2^M\}$ serve as the initial state of the LFSR.

We show in the following that this construction opens the door to an algebraic attack allowing to compute the secret key bits much more efficiently as compared to the general case described in [KH11].

For demonstrating this we consider the algebraic attack of [KH11] against general linear protocols described above and suppose that λ is chosen by the attacker such that $\lambda = M + 1$. Our construction implies that each bit of the function matrices of $F_1, \ldots, F_L$, and consequently each bit of the secret K-elements $x_{i,l}$, is the output of a publicly known $GF(2)$-linear mapping in the k-bits and the random l-bits.

Hence, the secret K-elements $x_{i,l}$ can be written as

$$x_{i,l} = \bigoplus_{s=1}^{m} c_{i,s} k_s \oplus \bigoplus_{t=1}^{M} C_{i,t} l_t,$$

where $\tilde{l} = (l_1, \ldots, l_M)$ and the vectors $c_{i,s}, C_{i,t} \in GF(2)^\lambda$ are publicly known. Thus, each known plaintext pair (x, y), $x = \bigoplus_{i \in I} e_i$, translates into the statement that

$$\bigoplus_{s=1}^{m} \left(\bigoplus_{i \in I} c_{i,s} \right) k_s \in W(y),$$

where the set $W(y) \subseteq GF(2)^\lambda$ is defined by $W(y) = \{y \oplus C_{I,1}, \ldots, C_{I,L}\}$ and for each $\tilde{l} = (l_1, \ldots, l_M)$ representing an element of $\{1, \ldots, L\}$ it holds that

$$C_{I,l} = \bigoplus_{t=1}^{M} \left(\bigoplus_{i \in I} C_{i,t} \right) l_t.$$

Now we can compute a nonzero Boolean function $g : \{0,1\}^\lambda \longrightarrow \{0,1\}$ which annihilates $W(y)$. This is possible as $W(y)$ is a proper subset of $\{0,1\}^\lambda$ due to $|W(y)| \leq 2^M = 2^{\lambda-1}$.

More concretely, we compute a square free polynomial $p = p(z_1, \ldots, z_\lambda)$ which yields g. This can be done by solving a system of at most L $GF(2)$-linear equations in at most 2^λ variables corresponding to the square free monomials over $z_1, \ldots, z_\lambda$. As M and λ are small numbers in practice, this is feasible. Note that the degree of p is at most λ.

Consequently, the known plaintext pair (x, y) yields the following nonlinear equation in the key bits:

$$p\left(\bigoplus_{s=1}^{m} \left(\bigoplus_{i \in I} c_{i,s} \right) k_s \right) = 0.$$

The degree of this equation is at most $\lambda = \log_2(L) + 1$, which is much smaller than L, the degree of the algebraic attack for the general case.

D Proof of Theorem 1

Proof: For given $y, a \in (K^*)^{n/8}$, Eve has to choose an element $x \in (K^*)^{n/8}$ such that

$$w + w' = F_l\left((u_1, v_1), \ldots, (u_{n/8}, v_{n/8})\right)$$

will be accepted by Alice in inner state a, where $w = F_l(f(x,b))$ and $w' = F_l(f(y,b'))$ for some $l \in [L]$, and $b, b' \in (K^*)^{n/8}$. Note that Eve has no information about b, b', and that $u_i = x_i b_i + y_i b'_i$ and $v_i = x_i b_i^3 + y_i b'^3_i$ for $i = 1, \ldots, n/8$.

Consequently, Eve's choice for the value x has to satisfy

$$(x_i b_i + y_i b'_i)^3 = a_i^2 \left(x_i b_i^3 + y_i b'^3_i\right)$$

for all $i = 1, \ldots, n/8$.

This is equivalent to

$$(x_i + y_i c_i)^3 = a_i^2 \left(x_i + y_i c_i^3\right),$$

where $c_i = b'_i \left(b_i^{-1}\right)$, which, in turn, is equivalent to $P_i(x_i, c_i) = 0$, where the polynomial $P_i(x_i, d_i)$ is for all $d_i \in K^*$ and $i = 1, \ldots, n/8$ defined as

$$P_i(x_i, d_i) = x_i^3 + (y_i d_i) x_i^2 + \left(y_i^2 d_i^2 + a_i^2\right) x_i + d_i^3 \left(y_i^3 + y_i a_i^2\right).$$

Note that there are $|K^*| = 15$ different polynomials of type $P(x_i, d_i)$ with respect to the variable x_i (look at the coefficient $y_i d_i$ of x_i^2).

For all $x_i \in K^*$ let $P_i(x_i) = \{d_i \mid P(x_i, d_i) = 0\}$. $P_i(x_i, d_i)$ is a polynomial of degree 3 also in the unknown d_i, which implies $|P_i(x_i)| \leq 3$ for all $x_i \in K^*$.

Eve has to choose an $x \in (K^*)^{n/8}$ that satisfies $c_i \in P_i(x_i)$ for all $i = 1, \ldots, n/8$. Since she does not have any information about $c_1, \ldots, c_{n/8}$, her success probability is bounded from above by

$$\prod_{i=1}^{n/8} \frac{3}{15} = 0.2^{n/8}.$$

This concludes the proof. □