On Hardware-Oriented Message Authentication with Applications Towards RFID

Martin Ågren, Martin Hell, and Thomas Johansson
Dept. of Electrical and Information Technology, Lund University,
P.O. Box 118,
221 00 Lund, Sweden
Email: martin.agren@eit.lth.se

***Abstract*—We consider ultra light-weight constructions of message authentication in hardware applications like RFID. We propose a new type of constructions that will be less costly to implement in hardware, compared to any previous construction. These constructions are based on the framework of universal hash functions, Toeplitz matrices and ϵ-biased sample spaces. Some new theoretical results in this area are derived. The hardware-attractive new constructions come at the price of not being able to prove the exact substitution probability. The expected probability is examined both through theoretical methods as well as through simulation.**

***Keywords*-Message Authentication Codes; Universal Hash Functions; RFID**

I. Introduction

Message Authentication Codes (MAC:s) [7] is a class of keyed functions used in many different areas to ensure that a message has been sent by the true sender and received without having been altered during transmission. The sender produces a tag $\boldsymbol{t} = f(\boldsymbol{k}_{\text{MAC}}, \boldsymbol{m})$ for a message $\boldsymbol{m}$ using a key $\boldsymbol{k}_{\text{MAC}}$ and attaches it to the message. The receiver, who shares the key $\boldsymbol{k}_{\text{MAC}}$, can produce the tag for the received message and immediately decide whether the message can be regarded as authentic.

An active adversary tries to modify a transmitted message and its tag and hopes to get this accepted at the receiver side. We would like the probability that the attacker succeeds to be some very small value.

There are many ways to provide message authentication, e.g., using certain block cipher modes of operation, using a keyed hash function, or using constructions based on universal hashing [23], [21]. This paper focuses on the last approach, which is the usual choice when we assume that encryption is done through a stream cipher (unless there is some dedicated authentication inside the stream cipher). Typical examples would be the GCM mode and the UIA2 found in UMTS and LTE.

One major new application area in cryptology is light-weight cryptology. Many recent applications put very strong demands on parameters of cryptographic algorithms. In particular, we have in mind passive RFID applications, where the total gate count must be kept very low. No acceptable solutions using symmetric algorithms have yet been demonstrated in practice. Recently, a new RFID authentication protocol based on a stream cipher has been presented [3]. Motivated by this, we consider ultra light-weight constructions of message authentication in hardware applications like RFID.

We examine several known constructions and evaluate details around their hardware implementations. More specifically, we try to assess the cost of a simple implementation which uses no lookup tables or advanced techniques. We quantify the cost in terms of registers and gates.

We then propose a new type of constructions that will be less costly to implement in hardware, compared to any previous construction. This class is very promising with respect to security, practical implementation and performance.

Our constructions are based on the framework of universal hash functions, Toeplitz matrices and ϵ-biased sample spaces. Some new theoretical results in this area are derived. In particular, some results by Krawczyk [14] are corrected and extended.

The hardware-attractive new constructions come at the price of not being able to prove the exact substitution probability. Instead, the expected probability is examined both through theoretical methods as well as through simulation. The arguments point at a sufficiently low substitution probability for the proposed constructions.

The paper is organized as follows. In Section II we give the general setup for the problem we study as well as some basic definitions. Section III outlines a hardware evaluation of existing constructions. Section IV then derives the theoretical background for the new constructions, correcting and generalizing some previous results related to the use of Toeplitz matrices for authentication. In Section V we then give a new class of constructions and discuss various properties of this class. Section VI studies a specific instance, while in Section VII we conclude the paper.

II. Preliminaries

We are interested in a mechanism combining encryption and authentication in some packet-based communication system, or similar. We assume that the encryption is done

978-0-7695-4340-6/11 $26.00 © 2011 IEEE
DOI 10.1109/LightSec.2011.9

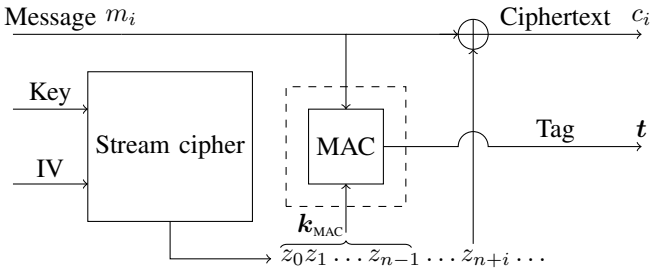

Figure 1. The setting we consider in this paper. The first n output bits from a stream cipher, initialized with a key and IV, form $\boldsymbol{k}_{\text{MAC}}$ which is used to initialize the authentication mechanism. The rest of the stream cipher bits are used to encrypt the message bits m_i, $i = 0, 1, \ldots$. The end result is a sequence of ciphertext bits c_i and the authentication tag $\boldsymbol{t}$.

through a modern stream cipher, using a secret key $\boldsymbol{k}$ and a public initial value (IV). The stream cipher would either be a dedicated one, or a suitable mode of operation for a block cipher. The key $\boldsymbol{k}_{\text{MAC}}$ used to produce the MAC is derived from $\boldsymbol{k}$ by using keystream before encryption starts.

To appreciate why this is done, consider a first approach that, after initializing the stream cipher, extracts whatever randomness the MAC construction needs to get started. Then, encrypts the message as usual and feeds the encrypted data into the authentication box. Finally, once the entire message has been processed, extracts some additional randomness from the keystream to finalize the authentication tag, e.g., by XOR-ing it with a one-time pad. However, using bits from the "end" of the keystream in this manner is not a very good idea if the message can have variable size – with a slightly shorter message, the key material previously used for encryption, would now be used for authentication. The security implications would be disastrous — it is crucial that the sender and receiver use the same one-time pad.

The solution is to extract all the key bits in $\boldsymbol{k}_{\text{MAC}}$ needed for the authentication *before* starting the encryption process. This either means that we need to store some of these key bits somewhere until we need them; or we use a construction that solves this in another way, e.g., putting them in a processing state. The message is processed bit by bit and does not need to be stored. Summarizing this, we present an overview of this approach in Figure 1. This is an adopted and standard approach, used in GCM, GSM, UMTS, etc. The remainder of this paper will focus on the dashed rectangle in Figure 1 which inputs a key and a message and outputs a tag. (One might also authenticate the cipherbits c_i instead of message bits m_i.)

When we study known constructions, we ignore the process of creating key bits $\boldsymbol{k}_{\text{MAC}}$. We consider the hardware cost for the MAC generation part, as marked in Figure 1. When assessing the costs for certain hardware primitives (gates), we have used cost figures found in several other papers, e.g., [11]; Flip flop 8, NAND2 1, XOR2 2.5, MUX2 4, MUX3 5, DEMUX3 5.

A. MAC:s and Universal Hash Functions

We need some definitions and standard results when constructing MAC:s from universal hash functions (or equivalently authentication codes [20]).

A Message Authentication Code is a family of functions which map messages to tags using keys. Some common MAC:s are HMAC [2], which iterates a hash function, and UMAC [4], which uses a family of universal hash functions. In fact, it is quite common to use universal hash functions in MAC constructions (GCM, UIA2), since they are proven secure in a very intuitive sense and are surprisingly often well-adapted for hardware implementations. Denote a family (set) H of hash functions mapping from set A to set B by (H, A, B). We will now give some basic definitions.

Definition 1: (H, A, B) is **ϵ-almost XOR universal** (ϵ-AXU) if $\forall x, x' \in A, x \neq x', y \in B$,

$$|\{h \in H : h(x) \oplus h(x') = y\}| \leq \epsilon \cdot |H|.$$

When constructing a MAC using an ϵ-AXU family, one part of the key is used to select a function $h \in H$ and the output of this function is XOR-ed with a second part of the key, used as a one-time pad, chosen randomly from B. The following definition will turn out to be related [18], [14].

Definition 2: Let S be a distribution on binary sequences $\boldsymbol{s} = (s_0, s_1, \ldots, s_{n-1})$ of length n. Then,

- S **passes the linear test $\boldsymbol{\alpha}$ with bias** ϵ if
$$\left| \frac{|\{\boldsymbol{s} \in S : \boldsymbol{\alpha} \cdot \boldsymbol{s} = 0\}|}{|S|} - \frac{1}{2} \right| \leq \epsilon,$$
where $\boldsymbol{\alpha} \in \mathbb{F}_2^n$, and $\boldsymbol{\alpha} \cdot \boldsymbol{s} = \sum_{i=0}^{n-1} \alpha_i s_i$.
- S is an **ϵ-biased distribution** if it passes all linear tests $\boldsymbol{\alpha} \neq \boldsymbol{0}$ with bias ϵ.

We usually consider S as a set of n-bit vectors, each one taken with the same probability $1/|S|$.

Here $\boldsymbol{s} = (s_0, s_1, \ldots, s_{n-1}) \in S$ is a random variable and so is $\boldsymbol{\alpha} \cdot \boldsymbol{s} \in \mathbb{F}_2$ for any $\boldsymbol{\alpha}$. If all such binary random variables are deviating at most ϵ from the uniform binary distribution, we refer to S as an ϵ-biased distribution. Clearly, selecting S as all 2^n binary vectors would give us a 0-biased distribution. The idea is to make S smaller but still keep ϵ small.

Traditionally, two attack success probabilities are of interest: that of insertion (impersonation) attacks and that of substitution attacks. It has been shown that the substitution attack is always the more powerful attack [12]. The attacker tries to replace a legitimate message-tag pair $(\boldsymbol{m}, \boldsymbol{t})$ with another pair $(\boldsymbol{m}', \boldsymbol{t}')$ of his or her choice and succeed with probability P_S. Relating this to our previous definitions; split the key as $\boldsymbol{k}_{\text{MAC}} = (\boldsymbol{k}_A, \boldsymbol{k}_B)$, where $\boldsymbol{k}_B$ has the tag size. Assume that $H = \{h_i\}$ is ϵ-almost XOR universal. Then if the tag is generated as $\boldsymbol{t} = h_{\boldsymbol{k}_A}(\boldsymbol{m}) \oplus \boldsymbol{k}_B$, we have $P_S \leq \epsilon$. Obviously, this works only for one message, and for the next one we assume that the key has received a new value.

B. Discrete Fourier Transforms of Boolean Functions

We define the basis vectors $\mathcal{X}_{\boldsymbol{\sigma}}(\boldsymbol{x}) = (-1)^{\boldsymbol{\sigma}\cdot\boldsymbol{x}}$ for $\boldsymbol{\sigma}, \boldsymbol{x} \in \mathbb{F}_2^n$ and, for a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, the transform

$$\hat{f}(\boldsymbol{\sigma}) = \frac{1}{2^n} \sum_{\boldsymbol{x}\in\mathbb{F}_2^n} f(\boldsymbol{x})\mathcal{X}_{\boldsymbol{\sigma}}(\boldsymbol{x}),\ \boldsymbol{\sigma} \in \mathbb{F}_2^n \tag{1}$$

The coefficients describe f uniquely, and the inverse is

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{\sigma}\in\mathbb{F}_2^n} \hat{f}(\boldsymbol{\sigma})\mathcal{X}_{\boldsymbol{\sigma}}(\boldsymbol{x}).$$

The interested reader will find a more detailed introduction in e.g., [15]. We end by defining $\mathrm{L}_1(f) = \sum_{\boldsymbol{\sigma}} |\hat{f}(\boldsymbol{\sigma})|$.

III. A Hardware Evaluation of Existing Constructions

We have started by revisiting some known hardware-efficient constructions. Due to space limitations, this part is to be found in the full version of this paper. We investigate the MAC generation in GCM; the UIA2 algorithm in UMTS; the WH construction (a hardware-oriented version of UMAC); Krawczyk's CRC construction; the LH and UH constructions by Sarkar; and the LFSR based Toeplitz construction.

The conclusion of this investigation is that the constructions need at least 4 but often 5 buffers (registers) of tag size w. Lesser number of buffers would require a costly multiplication in $\mathbb{F}_{2^l}$, where l can be the tag size w. The exception is Krawczyk's CRC construction which is very implementation efficient. But here the key bits need to select an irreducible polynomial, and this is very costly to implement. In the full version of this paper we show that it is not possible to modify this construction in a simple way.

IV. The Generalized Toeplitz Construction – a Basis for New Constructions

Assume that k_i, $i = -w, 1-w, \ldots, L-2$ is a sequence of randomly chosen key bits. Let $\boldsymbol{t} = (t_0, \ldots, t_{w-1})$ be a bitvector of length w and $\boldsymbol{m} = (m_0, \ldots, m_{L-1})$ a bitvector of length L. A possible MAC construction could be

$$\begin{bmatrix} t_0 \\ t_1 \\ \vdots \\ t_{w-1} \end{bmatrix} = \begin{bmatrix} k_{-1} & \ldots & k_{L-2} \\ k_{-2} & \ldots & k_{L-3} \\ \vdots & \ddots & \vdots \\ k_{-w} & \ldots & k_{L-1-w} \end{bmatrix} \begin{bmatrix} m_0 \\ \vdots \\ m_{L-1} \end{bmatrix}. \tag{2}$$

An algorithmic interpretation is that we initialize $\boldsymbol{t} \leftarrow \boldsymbol{0}$. Introduce a window of size w to form

$$\boldsymbol{K}_0 = [k_{-w} \ldots k_{-1}], \ldots, \boldsymbol{K}_{L-1} = [k_{L-1-w}, \ldots k_{L-2}].$$

For each bit m_i, if it is zero we do nothing and if it is one we update $\boldsymbol{t} \leftarrow \boldsymbol{t} \oplus \boldsymbol{K}_i$. Note how we can use a register $\boldsymbol{K}$ to maintain a state which we shift and accumulate into the tag $\boldsymbol{t}$. We call this the register–accumulator design. It is a simple step to show that $P_S = 2^{-w}$.

The Toeplitz matrix above uses $L + w - 1$ truly random key bits $k_{-w}, \ldots, k_{L-2}$, which is much too costly. The next step is to remove the requirement for genuine randomness and instead use some bitstream generator (decreasing the key size). We then have the following theorem.

Theorem 1: A family of hash functions defined by a Toeplitz construction as above, where the bits $k_{-w} \ldots k_{L-2}$ determine the Toeplitz matrix, is $(2^{-w} + 2\epsilon)$-almost XOR universal over equal-length strings if the distribution of the bit sequences is ϵ-biased.

The original formulation [14] of this theorem, inspired by [1], [18], [13], was stated incorrectly as it lacked the factor 2 that goes with ϵ.[1] See Section IV-A for a discussion on the proof of this theorem.

One might want to extend the security to messages of variable length:

Theorem 2: Let $A = \mathbb{F}_2^L$ and let $A' = \mathbb{F}_2^1 \cup \mathbb{F}_2^2 \cup \ldots \cup \mathbb{F}_2^{L-1}$. Assume $H = \{h_i | i \in I\}$. Define $H' = \{h'_i | i \in I\}$ by $h'_i : A' \to B$, $h'_i(\boldsymbol{m}) = h_i(\boldsymbol{m}||1||0^{L-|\boldsymbol{m}|-1})$. If (H, A, B) is the $(2^{-w} + 2\epsilon)$-AXU construction in Theorem 1, then (H', A', B) is $(2^{-w} + 2\epsilon)$-AXU.

The crucial property of H is that appending zeros to the message does not change the tag. This property also allows h'_i to be calculated without invoking h_i on the zero-padded message.

All in all, if we assume the constructed keystream to be ϵ-biased, we can construct the MAC by using the above Toeplitz construction, on the message and a trailing 1, and XOR-ing with w keystream bits, getting $P_S \leq 2^{-w} + 2\epsilon$.

A. The Proofs of Theorem 1 and [14, Lemmas 11 and 12]

We need to give a proof of Theorem 1, and begin with some background. The idea of this proof, as carried out in [14], is that the success probability of replacing an $n - w + 1$-bit message and w-bit tag $(\boldsymbol{m}, \boldsymbol{t})$ with $(\boldsymbol{m} \oplus \boldsymbol{a}, \boldsymbol{t} \oplus \boldsymbol{b})$ is precisely the probability that $A \cdot \boldsymbol{s} = \boldsymbol{b}$ with $\boldsymbol{s} \in S$ taken from the ϵ-biased distribution and letting the matrix A of size $w \times n$ be constructed by letting row $i = 1, 2, \ldots, w$ be $\boldsymbol{\alpha}_i = 0^{w-i}\boldsymbol{a}0^{i-1}$, i.e., the rows in A are zero-padded shifted versions of $\boldsymbol{a}$. Note that A is of maximal rank (w) as long as $\boldsymbol{a} \neq \boldsymbol{0}$, which is true by definition in the attack scenario.

The proof uses Fourier analysis and an indicator function

$$f_{\boldsymbol{\alpha},\beta}(\boldsymbol{s}) = \begin{cases} 1, & \text{if } \boldsymbol{\alpha}\cdot\boldsymbol{s} = \beta, \\ 0, & \text{otherwise.} \end{cases}$$

A lemma then claims that $\mathrm{L}_1(f_{\boldsymbol{\alpha},\beta}) = 1$ for all l-bit vectors $\boldsymbol{\alpha}$ and bits β. The proof of this result is not included in [14], but is stated to be quite straightforward. Nonetheless, it is based upon an incorrectly stated property of Fourier coefficients. It can be shown that the result is not correct

[1] The result can actually be improved by changing the coefficient from 2 to $2(1 - 2^{-w})$, but for large w this change is negligible.

and needs to be adjusted to requiring $\boldsymbol{\alpha} \neq \mathbf{0}$. See the full version of this paper for the details.

More interestingly, we can change [14, Lemma 12] from an inequality to an equality. That is, we can study the matrix–vector version,

$$f_{A,\boldsymbol{b}}(s) = \begin{cases} 1, & \text{if } A \cdot \boldsymbol{s} = \boldsymbol{b}, \\ 0, & \text{otherwise.} \end{cases}$$

We then have

Lemma 1: $\mathrm{L}_1(f_{A,\boldsymbol{b}}) = 1$ for all A of full rank.

The proof uses some valuable information on the Fourier coefficients, and since we will use these insights later, we formalize them.

Lemma 2: $\hat{f}_{A,\boldsymbol{b}}(\boldsymbol{\sigma}) = \pm 2^{-w}$ for the 2^w values of $\boldsymbol{\sigma}$ that are linear combinations of rows in the full-rank matrix A and zero for all other $\boldsymbol{\sigma}$. The sign is positive precisely when the corresponding sum of bits in $\boldsymbol{b}$ is zero.

We prove these lemmas in the full version of the paper. Theorem 1 now follows by the arguments in [14], but we continue towards a modified proof here for completeness and some further insights.

First, let $\mu(\boldsymbol{s})$ be the distribution of sequences of length n and assume that there are 2^l distinct sequences, each occurring with probability 2^{-l}. Define $\epsilon_{\boldsymbol{\sigma}}$ as the bias ϵ for the linear test $\boldsymbol{\sigma}$, but with the sign kept. That is,

$$|\{\boldsymbol{s} \in S : \boldsymbol{\sigma} \cdot \boldsymbol{s} = 0\}| = 2^{l-1} + \epsilon_{\boldsymbol{\sigma}} 2^l.$$

We get

$$\begin{aligned} \hat{\mu}(\boldsymbol{\sigma}) &= 2^{-n-l} \sum_{\boldsymbol{s}:\mu(\boldsymbol{s})\neq 0} \mathcal{X}_{\boldsymbol{\sigma}}(\boldsymbol{s}) \\ &= 2^{-n-l} \left[(2^{l-1} + \epsilon_{\boldsymbol{\sigma}} 2^l)(+1) + (2^{l-1} - \epsilon_{\boldsymbol{\sigma}} 2^l)(-1) \right] \\ &= 2\epsilon_{\boldsymbol{\sigma}} 2^{-n} \end{aligned}$$

for all $\boldsymbol{\sigma} \neq \mathbf{0}$ (see also [15, pages 14-15]).

We state the following lemma which is found in the proofs of [14, Theorem 5] and [15, Lemma 4.5]:

Lemma 3:

$$|\operatorname{Prob}_S(A \cdot \boldsymbol{s} = \boldsymbol{b}) - 2^{-w}| = |2^n \sum_{\boldsymbol{\sigma}\neq \mathbf{0}} \hat{\mu}(\boldsymbol{\sigma}) \hat{f}_{A,\boldsymbol{b}}(\boldsymbol{\sigma})|$$

Proof (Theorem 1): Recall from Lemma 2 that for those $A, \boldsymbol{b}$ that we consider, $\hat{f}_{A,\boldsymbol{b}}(\boldsymbol{\sigma}) = \pm 2^{-w}$ for precisely 2^w values of $\boldsymbol{\sigma}$ and 0 otherwise. Since, by definition, $|\epsilon_{\boldsymbol{\sigma}}| \leq \epsilon$, we get

$$\begin{aligned} |\operatorname{Prob}_S(A \cdot \boldsymbol{s} = \boldsymbol{b}) - 2^{-w}| &= 2 \left| \sum_{\boldsymbol{\sigma}\neq \mathbf{0}} \epsilon_{\boldsymbol{\sigma}} \hat{f}_{A,\boldsymbol{b}}(\boldsymbol{\sigma}) \right| \\ &\leq 2 \sum_{\substack{\boldsymbol{\sigma}\neq \mathbf{0}: \\ \hat{f}_{A,\boldsymbol{b}}(\boldsymbol{\sigma})\neq 0}} \epsilon 2^{-w} < 2 \cdot 2^w \cdot \epsilon 2^{-w} = 2\epsilon. \end{aligned}$$

We might think that we have realized how to find the best attack probability and the best attack: For each $\boldsymbol{\sigma} \neq \mathbf{0}$, find $\epsilon_{\boldsymbol{\sigma}}$. Then choose $\hat{f}$ nonzero for the 2^w values of $\boldsymbol{\sigma}$ that have the largest absolute values for $\epsilon_{\boldsymbol{\sigma}}$. By also matching signs, we can get all products $\epsilon_{\boldsymbol{\sigma}} \hat{f}(\boldsymbol{\sigma}) \geq 0$ (or $\leq$) and arrive at

$$|\operatorname{Prob}_S(A \cdot \boldsymbol{s} = \boldsymbol{b}) - 2^{-w}| = 2|\sum_{\boldsymbol{\sigma}} \epsilon_{\boldsymbol{\sigma}} \hat{f}(\boldsymbol{\sigma})| = 2 \sum_{\boldsymbol{\sigma}} |\epsilon_{\boldsymbol{\sigma}}| 2^{-w}.$$

Having selected $\hat{f}$, we can invert and acquire f. But is it a realizable attack? Probably not. There are $\binom{2^n-1}{2^w} 2^{w-1}$ such transforms but only $(2^{n-w}-1)2^w < 2^n$ valid functions $f_{A,\boldsymbol{b}}$. Thus, it is highly unlikely that we have found a transform $\hat{f}$ that can be inverted into $f_{A,\boldsymbol{b}}$ on our form, giving $\boldsymbol{a}, \boldsymbol{b}$.

This means that we need to use other sets of $\epsilon_{\boldsymbol{\sigma}}$, e.g., many large values and some close to zero (where we can then allow $\hat{f}_{A,\boldsymbol{b}}$ to have any sign). This lowers the final sum, and thus also the success probability of the best attack.

We conclude: the bias ϵ gives a bound, but in order to have a realizable attack we need to use a collection of $\epsilon_{\boldsymbol{\sigma}}$ (most of which are likely lower than ϵ) and sum them in a sub-optimal way. That is, we cannot expect to be able to find $|\sum_{\boldsymbol{\sigma}} \epsilon_{\boldsymbol{\sigma}} \hat{f}_{A,\boldsymbol{b}}(\boldsymbol{\sigma})| = \epsilon$.

In Section VI, we will study a specific design and show how to turn a linear test with a large bias into an attack and a bound on the success probability.

V. A New Class of Constructions

From the analysis of existing authentication mechanisms, provided in the full version of this paper, we bring some experiences: either, 1) they are too keystream consuming, or 2) their hardware implementation is fairly resource consuming, or 3) we need "special randomness", i.e., random irreducible polynomials. We have also noted several properties which we would like to keep in mind when constructing new algorithms. First of all, the register–accumulator design is very promising. However, until now, it has always had at least one of the three deficiencies listed above. Secondly, we prefer not to halt the tag generation every now and then to load a batch of message and/or key material.

With our ultra light-weight setting in mind, we have investigated several new constructions, all of which use the register–accumulator paradigm. This allows the final one-time pad to be added before the actual accumulating, i.e., before we process the message. This is a very desirable property from a hardware and security perspective, since it allows us to use fewer registers without sacrificing the security of the tag.

A. A New Class of Accumulator–Register Constructions

Any state-machine generating a sequence that is internally stored in a shift register is suitable to be used with the register–accumulator paradigm. One could, for example, use the keystream from the stream cipher TRIVIUM [5]. The keystream is not directly stored internally, but quite close, as the output is formed as the XOR of a few internal bits. To ensure parallelism in TRIVIUM, such a stored bit

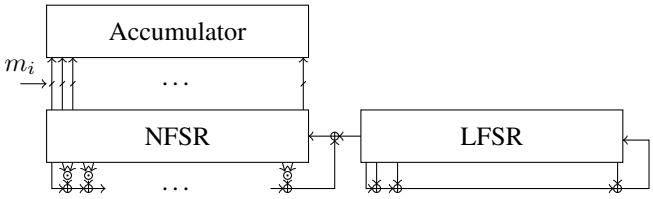

Figure 2. A hardware implementation of the suggested construction. The top register is the accumulator, where the contents of the left register is added iff the message bit is a one. The contents of the NFSR and LFSR are updated with each clock cycle.

never changes value within 64 register shifts after having been calculated. Since the keystream is generated through a simple XOR of six bits, this means that XOR-ing six 64-bit sections of the internal state, one has a logical register which can be accumulated as previously into a 64-bit tag. Theorem 1 then relates the security of this construction to the bias of TRIVIUM's keystream. Finding any large bias would mean a strong distinguishing attack on TRIVIUM, so we can be satisfied from a security point of view.

However, we consider it overkill to use a mechanism as strong and expensive as TRIVIUM.[2] We want a more hardware-efficient construction where the keystream might not be cryptographically secure, but still it generates an ϵ-biased distribution, where ϵ is low. This is sufficient for a good protection against a substitution attack.

We propose the following construction idea and note the similarities with the stream cipher Grain [11], [10].

As before, we use an accumulator that is updated (XOR-ed) precisely when the bit-to-authenticate is 1. The keystream that we use to update the accumulator comes from a nonlinear finite state machine (FSM). The FSM consists of a nonlinear shift register (NFSR) serving as a filter, and a linear feedback shift register (LFSR) to feed the NFSR. An implementation is outlined in Figure 2.

Formalizing this construction, we denote by n_i, $i = 0, 1, \ldots$ the keystream used in the Toeplitz construction. This keystream is the output of the NFSR, and the initial state of the NFSR is $(n_0, \ldots, n_{w-1})$. The LFSR has initial state $(l_0, \ldots, l_{v-1})$ and generates an LFSR sequence $l_j = \sum_{i=1}^{v} c_i l_{j-i}$, for $j \geq v$. Finally, the NFSR sequence is obtained as

$$n_i = f(n_{i-w}, \ldots, n_{i-1}) \oplus l_{i-w},$$

for $i \geq w$, where f is some nonlinear function. We only allow the construction to authenticate messages of at most L bits, where the security parameter L will be studied later.

Let us briefly examine the hardware cost of this construction. Assume that both registers are of size w, i.e., the initial states are determined by $2w$ bits. An implementation as outlined in Figure 2 requires three registers of length w and some minor combinatorics. An approximate gate count for $w = 64$ gives around 1800. This corresponds to $3w$ flip flops, w XOR:s, $3w/2$ AND:s (for a specific choice of f) and a small number of multiplexers, where the feedback functions have been estimated at w XOR:s and $w/2$ AND:s. This is much cheaper than any previous construction.[3]

In order to quantify the security of this construction, we turn to Theorem 1 and consider the keystream sequence n_i leaving the NFSR. If this sequence is random-looking enough in our sense, i.e., ϵ is low, the construction is secure enough. Thus, it is irrelevant whether the construction could be used as a stream cipher (it can't, since observing just $w+v$ bits leaving the NFSR, we can trivially reconstruct the entire initial state), or whether it suits any other cryptographic applications. We intend to use it as a MAC, and all that matters is that it is a very efficient construction which is secure in this very application.

B. On the Problem of Finding the Bias

In order to use Theorem 1, we need to find (a bound on) the bias ϵ. Finding the actual value of the bias is difficult, but we can at least argue around it. If there is a linear approximation f' of the function f with bias ϵ', i.e.,

$$f'(n_{i-w}, \ldots, n_{i-1}) = d_0 + \sum_{j=1}^{w} d_j n_{i-j}, \quad d_j \in \mathbb{F}_2,$$

$$\left| \frac{|\{\mathbf{n} \in \mathbb{F}_{2^w} : f'(\mathbf{n}) = f(\mathbf{n})\}|}{2^w} - \frac{1}{2} \right| = \epsilon',$$

we can use a linear test $\boldsymbol{\alpha}$, chosen by keeping the taps of the LFSR in mind, to achieve $\epsilon = 2^{c-1}(\epsilon')^c$ [16], where $c-1$ is the number of taps in the LFSR or, equivalently, c is the weight of the LFSR polynomial. This means that if we have the bias ϵ' for the best linear approximation of f, and can rewrite the LFSR into an up-to-L-degree, weight-three multiple of the feedback polynomial [17], [19], [8], [22], there is a linear test with a bias of $2^2(\epsilon')^3$.

With $L \leq 2^{w/2}$, we should expect there to be no weight-3 multiple of the LFSR polynomial of degree L or less [8].

However, there might be a (catastrophically) higher bias. In order to make ϵ' very small, we could try to use a bent function that uses as many n_i's as possible, e.g., $f(n_{i-w}, \ldots, n_{i-1}) = n_{i-w} \oplus \sum_{j=1}^{w/2-1} n_{i-w+2j-1} n_{i-w+2j}$. We note that this structure allows a very efficient choice of $\boldsymbol{\alpha}$ in Definition 2, which leads to a very large bias. For example, with $w = v = 32$, f as above and the low-weight linear feedback chosen as $l_i = l_{i-32} \oplus l_{i-31} \oplus l_{i-29} \oplus l_{i-1}$, there is a $2w$ + 3-bit $\boldsymbol{\alpha}$ for which the distribution on

[2] We might consider reusing the encrypting implementation of (e.g.,) Trivium, by using equally many bits for encryption and authentication, but this means the overall speed is reduced to half.

[3] The number of AND:s could e.g., correspond to pairwise multiplication of all NFSR bits. The number of XOR:s could e.g., correspond to adding those pair-wise products and using half(!) the LFSR bits in the feedback. We consider these approximations to be on the high side.

the sequences $n_0, n_1, \ldots, n_{2w+2}$ passes the linear test $\boldsymbol{\alpha}$ with bias precisely 2^{-6} (see Section VI). Changing the linear feedback function to one that contains more nonzero coefficients that are uniformly distributed, e.g., $l_i = l_{i-32} \oplus l_{i-28} \oplus l_{i-23} \oplus l_{i-20} \oplus l_{i-17} \oplus l_{i-12} \oplus l_{i-8} \oplus l_{i-4}$, the $\boldsymbol{\alpha}$ selected in a similar way gives a bias of 2^{-19}, but multiples of this LFSR polynomial might have a lower weight, making it possible to find a better linear test. In Section VI, we will see that creating an attack from these linear tests will not necessarily lead to attack success probabilities on the order of the biases.

One might want to avoid such highly structured feedback functions where two NFSR-states close in time give rise to similar expressions for the feedback bit. One could e.g., place the bits used nonlinearly at distances that determine a full positive difference set.

C. The Bias From Another Point of View

Denote the entire initial state by

$$(s_0, \ldots, s_{w+v-1}) = (n_0, \ldots, n_{w-1}, l_0, \ldots, l_{v-1}).$$

Each bit n_i, $i \geq 0$ can be described as a function of the initial state: $n_i = f_i(\mathbf{s})$. If the feedback function is highly irregular, we would expect the expressions for the different functions f_i to be highly different. Finding a linear combination of f_i's with a high bias seems difficult if not impossible.

We can use coding theory tools to formulate the problem in another direction. For each possible initial state, output L bits from the generator. Let the sequences form the columns of a matrix. We have now constructed the generator matrix of a code and the problem of finding the bias has been transferred into that of finding a codeword (i.e., a linear combination of rows) with a very high or low weight (high bias). By adding an all-one row to the matrix, this problem reduces to that of finding a low-weight codeword.

We argue that computing this characteristic of the code is generally difficult, which suggests that so is finding a linear test with high bias in this case. A general version of this problem is known to be NP complete.

The best practical algorithm for finding minimum-weight codewords in a linear code without using its structure is due to Canteaut and Chabaud [6]. This algorithm can be used on smaller instances, but will not be successful in finding minimum-weight codewords if the initial state is too large.

D. Numerical Results on Smaller Instances

As exhaustively searching for the bias is out of the question for any practical tag size, we have instead carried out experiments on smaller instances of our construction. We have done as follows. For each sequence length $L \in \{1, 2, \ldots, L_{\max}\}$ we have studied each $\boldsymbol{\alpha} \neq \mathbf{0}$ consisting of L bits. For each $\boldsymbol{\alpha}$ in turn, we have loaded each key $\boldsymbol{k}_A$ and generated the sequence $\boldsymbol{s}_{\boldsymbol{k}_A}$. Then, we have simply calculated the bias as in Definition 2. Naturally, with larger values of L there is a tendency for growing bias. See Figure 3 for the results.

We have used $w = v$ and tags of two different sizes: 4 and 6 bits. The linear feedback polynomials have been chosen primitive. The nonlinear feedbacks have been created by functions with high nonlinearity, and are not repeated here. With the tags of size four, the construction gave rise to a cycle set that contains some smaller cycles.

To judge the quality of the obtained biases, we have compared with the LFSR-based Toeplitz construction [13], which is featured in more detail in the full version of this paper. Briefly put, it uses the register–accumulator approach in (2), where the sequence $k_{-w'}, \ldots$ is produced by an LFSR. When the tag is of length $w' = r/2$, the key consists of a w'-bit initial state and a degree-w' irreducible feedback polynomial. Since there are only $\frac{2^{w'}}{w'}$ such polynomials, the entire key can be determined using $w' + \log\left(\frac{2^{w'}}{w'}\right) = 1 + r - \log r$ bits. Thus, if our construction uses $2w$ random bits, we allow the LFSR construction to use a slightly larger r, namely the one that solves

$$2w = 1 + r - \log r.$$

E.g., for $2w = 12$, we have $r \approx 14.9$. From [13], we know the LFSR construction to have $\epsilon = \frac{L}{2^{w'}}$. These values have been plotted in Figure 3.

We can e.g., note that the bias of our proposed construction is below that of the LFSR construction on equal-length strings. Do keep in mind that we allow the LFSR construction to use slightly larger registers in order to compare for a fixed amount of randomness. The gate count is thus a factor $\frac{14.9}{12} \approx 25\%$ larger than in our construction. (This completely ignores the issue of creating and/or verifying the special randomness needed — irreducible polynomials.)

As an additional comparison, we have derived the bias for a random number generator[4] (using the key as seed). We have tested several nonlinear feedback functions and can conclude that the higher the nonlinearity, the better the behaviour, bias-wise. We do not repeat all results here as the figure would appear quite busy. This intuitive relation between nonlinearity and bias gives crucial help in designing larger constructions.

E. Key Reuse

To make things clear: no security promises can be made if we reuse the key (or the key and IV that we initialize the stream cipher with).

For one thing, the zero-message reveals the one time pad. Knowing the one time pad, the message $(1, 0, 0, \ldots)$ reveals the initial state of the NFSR. Finally, the message corresponding to the value 2^w allows us to trivially reconstruct the initial LFSR state. Thus, three chosen messages are enough

[4]We chose to use the GNU Scientific Library's implementation of the Tausworthe random number generator (`gsl_rng_taus2`).

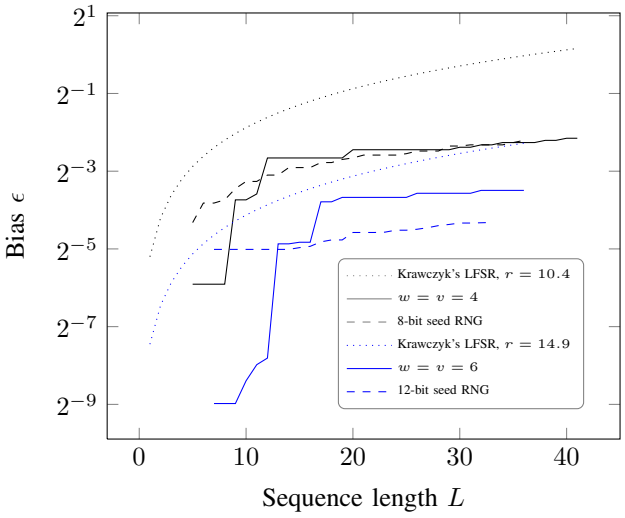

Figure 3. The bias as it develops for growing sequence lengths. Two different tag sizes w have been studied (corresponding to state-sizes $2w$) and their respective biases are plotted with solid lines. The dotted lines give the biases for the LFSR construction using equal amounts of randomness. Dashed lines show the behaviour of random number generators. (Lower curves are better.)

to get a full key recovery (or, for the stream cipher, the first $2w + v$ bits in the keystream).

The fact that this attack is outside the security model does not, as Handschuh and Preneel point out [9], make it uninteresting. On the contrary, it highlights the necessity of always using fresh keys for the authentication — i.e., a fresh IV for the stream cipher.

VI. An Example of a Specific Construction and the Best Attack on it

We define a specific 32-bit construction and analyze it. We use the nonlinear feedback function

$$f(n_{i-w}, \ldots, n_{i-1}) = n_{i-w} \oplus \sum_{j=1}^{w/2-1} n_{i-w+2j-1} n_{i-w+2j}$$

and the linear feedback specified by $l_i = l_{i-32} \oplus l_{i-31} \oplus l_{i-29} \oplus l_{i-1}$ (see Figure 2). Note that $w = v = 32$. The NFSR bits are calculated as $n_i = n_{i-32} \oplus N_{i-31}^{i-2} \oplus l_{i-32}$ for $i > 31$, where N_i^j is the sum of pairwise multiplications of the bits $n_i, \ldots, n_j$. By exploiting the fact that $N_i^j \oplus N_{i+2}^{j+2} = n_i n_{i+1} \oplus n_{j+1} n_{j+2}$, and that certain linear combinations of LFSR bits disappear, we can find that

$$\begin{aligned} &n_0 \oplus n_1 \oplus n_2 \oplus n_5 \oplus n_{31} \oplus n_{32} \oplus n_{33} \oplus n_{34} \oplus n_{32} \oplus n_{33} \\ &\oplus n_{34} \oplus n_{37} \oplus n_{63} \oplus n_{64} \oplus n_{65} \oplus n_{66} \oplus n_1 n_2 \oplus n_2 n_3 \\ &\oplus n_4 n_5 \oplus n_{31} n_{32} \oplus n_{33} n_{34} \oplus n_{34} n_{35} \oplus n_{62} n_{63} \oplus n_{63} n_{64} \end{aligned}$$

is zero. By cancelling common terms and using common factors to reduce the number of multiplications, we get

$$\begin{aligned} &n_0 \oplus n_1 \oplus n_2 \oplus n_5 \oplus n_{31} \oplus n_{37} \oplus n_{63} \oplus n_{64} \oplus n_{65} \oplus n_{66} \\ &\oplus n_2(n_1 \oplus n_3) \oplus n_4 n_5 \oplus n_{31} n_{32} \\ &\oplus n_{34}(n_{33} \oplus n_{35}) \oplus n_{63}(n_{62} \oplus n_{64}) = 0 \end{aligned} \quad (3)$$

as a relation between the keystream bits. There are five bit multiplications, and by replacing e.g., $n_4 n_5$ by n_5, we get a sum of bits with a bias of 2^{-6}! There are some choices to make, but one can e.g., try to choose cancelling bits and obtain a linear sum of bits $0, 1, 34, 37, 64, 65, 66$.

We construct an attack from a 67-bit vector $\boldsymbol{a}$ which is 1 precisely in these bits. The details of Theorem 1, as analyzed in Section IV-A, suggest that we choose to replace $\boldsymbol{m}$ by $\boldsymbol{m} \oplus \boldsymbol{a}$ and $\boldsymbol{t}$ with $\boldsymbol{t} \oplus \boldsymbol{b}$, where we still need to select $\boldsymbol{b}$.

The sum of various $\epsilon_{\boldsymbol{\sigma}}$ runs over precisely those nonzero $\boldsymbol{\sigma}$ that are linear combinations of shifted versions of $\boldsymbol{\alpha}_1 = 0 \ldots 0\boldsymbol{a}$. Considering the cyclic nature of the sequences that leave the NFSR, we realize that the linear tests $\boldsymbol{\alpha}_1$ and $\boldsymbol{\alpha}_2$ have the same bias, and that similarly $\epsilon_{\boldsymbol{\alpha}_1 \oplus \boldsymbol{\alpha}_2} = \epsilon_{\boldsymbol{\alpha}_2 \oplus \boldsymbol{\alpha}_3}$.

Recall that $\boldsymbol{b} = (b_0, b_1, \ldots, b_{w-1})$ and if we want all $\hat{f}_{A,\boldsymbol{b}}(\boldsymbol{\alpha}_i)$ to have the same sign, so that we add up all contributions from $\epsilon_{\boldsymbol{\alpha}_i}$, by Lemma 2, we need to set all b_i equal, so we choose $\boldsymbol{b} = \boldsymbol{0}$. Note that we have w values of $\epsilon_{\boldsymbol{\alpha}_1} = \epsilon$ and that if we assume that all other $\epsilon_{\boldsymbol{\sigma}}$ add up to zero, we get a success probability of $2^{-w} + 2|\sum_{\boldsymbol{\sigma}} \epsilon_{\boldsymbol{\sigma}} \hat{f}_{A,\boldsymbol{b}}(\boldsymbol{\sigma})| = 2^{-w} + 2w\epsilon 2^{-w} = 2^{-32} + 2^{-32}$. Thus, we need to consider the other terms as well, and (as an attacker) hope that they add up to something significant. (The same result can be obtained without Fourier analysis, by assuming a statistically independent behaviour for various shifts of the keystream sequence when used with the linear test as the message.)

Note that $\sum \epsilon_{\boldsymbol{\sigma}} = \epsilon_{\boldsymbol{\alpha}_1} + \ldots + \epsilon_{\boldsymbol{\alpha}_1 \oplus \ldots \oplus \boldsymbol{\alpha}_w}$. Why would e.g., the last term — the linear combination of all shifts of $\boldsymbol{\alpha}$ — have a bias even remotely close to 2^{-6}? Following this thought, we have used a computer to bound the biases for these linear combinations, not by summing the shifted approximations, but by summing the shifted exact expressions (3) and then approximating, for each sum getting the optimal approximation. In this way, we are able to bound the success probability of our attack to approximately $P_S < 2^{-32} + 2^{-27}$. Indeed, running the attack on a large number of random keys suggests that the success probability is slightly less than this value.

To say something about the behaviour for longer messages, we might use the results from Section V-D to heuristically bound the bias by $\frac{L}{2^{32}}$ and get $P_S \leq 2^{-32} + \frac{L}{2^{31}}$.

There might be other values of $\boldsymbol{a}$ that give a higher success probability (for some $\boldsymbol{b}$), but considering that we have found a linear test that exploits the structure of the construction, and that we have then derived an attack using insights gained from an expression that gives the exact success probability, it is hard to imagine how the success probability of another attack could be drastically better than the one presented here.

VII. Conclusions

We have proposed a new type of MAC construction, significantly cheaper in hardware compared to any known construction. The substitution probability is expected to be low, as it is bounded by the bias of the output stream and we demonstrate how this bound can be expected to be far from tight.

This class of constructions appears very promising and it would be interesting to find hardware-efficient designs where we can derive the bias explicitly, so that we can get a better understanding of the substitution probability.

Acknowledgment

This work was supported by the Swedish Foundation for Strategic Research (SSF) through its Strategic Center for High Speed Wireless Communication at Lund.

References

[1] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple construction of almost k-wise independent random variables. *Annual IEEE Symposium on Foundations of Computer Science*, pages 544–553, 1990.

[2] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology—CRYPTO'96*, pages 1–15. Springer-Verlag, 1996.

[3] O. Billet, J. Etrog, and H. Gilbert. Lightweight privacy preserving authentication for RFID using a stream cipher. In S. Hong and T. Iwata, editors, *Fast Software Encryption 2010*, volume 6147 of *Lecture Notes in Computer Science*, pages 55–78. Springer-Verlag, 2010.

[4] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. UMAC: Fast and secure message authentication. In *Advances in Cryptology—CRYPTO'99*, pages 215–233. Springer-Verlag, 1999.

[5] C. De Cannière and B. Preneel. Trivium. In M. Robshaw and O. Billet, editors, *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 244–266. Springer-Verlag, 2008.

[6] A. Canteaut and F. Chabaud. A new algorithm for finding minimum-weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 44:367–378, 1998.

[7] E. N. Gilbert, F. J. MacWilliams, and N. J. A. Sloane. Codes which detect deception. *Bell Systems Technical Journal*, 53(3):405–424, 1974.

[8] J. D. Golić. Computation of low-weight parity-check polynomials. *Electronic Letters*, 32(21):1981–1982, October 1996.

[9] H. Handschuh and B. Preneel. Key-recovery attacks on universal hash function based MAC algorithms. In D. Wagner, editor, *Advances in Cryptology—CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 144–161. Springer-Verlag, 2008.

[10] M. Hell, T. Johansson, A. Maximov, and W. Meier. A Stream Cipher Proposal: Grain-128. In *International Symposium on Information Theory—ISIT 2006*. IEEE, 2006.

[11] M. Hell, T. Johansson, and W. Meier. Grain - a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing, Special Issue on Security of Computer Network and Mobile Systems.*, 2(1):86–93, 2006.

[12] T. Johansson, G. Kabatianskii, and B. Smeets. On the relation between A-codes and codes correcting independent errors. In T. Helleseth, editor, *Advances in Cryptology—EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 1994.

[13] H. Krawczyk. LFSR-based hashing and authentication. In *Advances in Cryptology—CRYPTO'94*, pages 129–139. Springer-Verlag, 1994.

[14] H. Krawczyk. New hash functions for message authentication. In *Advances in Cryptology—EUROCRYPT'95*, pages 301–310. Springer-Verlag, 1995.

[15] E. Kushilevitz and Y. Mansour. Learning decision trees using the fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

[16] M. Matsui. Linear cryptanalysis method for DES cipher. In T. Helleseth, editor, *Advances in Cryptology—EUROCRYPT'93*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1994.

[17] W. Meier and O. Staffelbach. Fast correlation attacks on certain stream ciphers. *Journal of Cryptology*, 1(3):159–176, 1989.

[18] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM Journal on Computing*, 22(4):838–856, 1993.

[19] W.T. Penzhorn and G.J. Kühn. Computation of low-weight parity checks for correlation attacks on stream ciphers. In C. Boyd, editor, *Cryptography and Coding - 5th IMA Conference*, volume 1025 of *Lecture Notes in Computer Science*, pages 74–83. Springer-Verlag, 1995.

[20] G. J. Simmons. A survey of information authentication. In G. J. Simmons, editor, *Contemporary Cryptology, The Science of Information Integrity*, pages 379–419. IEEE Press, 1992.

[21] D. R. Stinson. Universal hashing and authentication codes. In J. Feigenbaum, editor, *Advances in Cryptology—CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 74–85. Springer-Verlag, 1992.

[22] D. Wagner. A generalized birthday problem. In M. Yung, editor, *Advances in Cryptology—CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer-Verlag, 2002.

[23] M. N. Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.