Cryptanalysis of TS-Hash*

Aleksei Udovenko

SnT, University of Luxembourg
aleksei@affine.group

Abstract. This note presents attacks on the lightweight hash function TS-Hash proposed by Tsaban, including a polynomial-time preimage attack for short messages (at most $n/2$ bits), high-probability differentials, a general subexponential-time preimage attack, and linearization techniques.

Keywords: Cryptanalysis · Hash function · LFSR · Lightweight · Low-weight · Generalized birthday paradox · Linearization · Quadratic equations

1 Introduction

Recently, Bookstein and Tsaban [BT23a, BT23b] studied the lightweight hash function TS-Hash, proposed originally by Tsaban for a master's thesis project [Tsa17]. TS-Hash can be compactly described by the following C-style pseudocode:

```
1  word TSHash(bitstring m) {
2      word poly[2] = {p0, p1};
3      word s = s0;
4      for (i = 0; i < m.size(); i++) {
```

*The work was supported by the Luxembourg National Research Fund's (FNR) and the German Research Foundation's (DFG) joint project APLICA (C19/IS/13641232).

Implementations of main attacks from this paper are available at https://github.com/cryptolu/TS-Hash-Cryptanalysis.

```
5           while ((s & 1) == 0) s >>= 1;
6           s = (s >> 1) ^ poly[m[i]];
7       }
8       return s;
9   }
```

Essentially, the hash function is based on a *linear feedback shift register* (LFSR), with the feedback polynomial being selected by the message bit among two public polynomials p_0, p_1. The message bit is consumed only when the feedback polynomial is needed, that is, when the LFSR generates the value 1, which creates the source of nonlinearity.

2 Formulations of the TS-hash

2.1 Algebraic formulation of TS-Hash

We describe an algebraic reformulation of the scheme, by treating the LFSR with one of the polynomials as multiplication by the element $g = X$ in the finite field[1] $\mathbb{F}_{2^n} \simeq \mathbb{F}_2[X]/(p_0(X))$. Note that the endianness is reversed: coefficients of the lowest-degree monomials correspond to the most significant bits in a state word in the implementation above. In this reformulation, the change of the polynomial amounts to adding the element $h = p_1(x) - p_0(X) \ \in \mathbb{F}_{2^n}$ to the state after the step. The switching is controlled by the message bit m and a bit in the representation of $g \cdot s$, which can always be expressed as $\mathsf{Tr}(\alpha \cdot s)$ for some $\alpha \in \mathbb{F}_{2^n}$, where

$$\mathsf{Tr} : \mathbb{F}_{2^n} \to \mathbb{F}_2 : z \mapsto \sum_{i=0}^{n-1} z^{2^i}$$

is the finite field trace function. The corresponding constant α can be efficiently determined in the following way.

Proposition 1. *Let* $\alpha \in \mathbb{F}_{2^n} = \sum_{i=0}^{n-1} \alpha_i g^i, \alpha_i \in \mathbb{F}_2$ *be such that* $\mathsf{Tr}(g^i) = 0$ *for* $i \in \{0, \ldots, n-2\}$ *and* $\mathsf{Tr}(g^{n-1}) = 1$. *Then,* $(\alpha_0, \ldots, \alpha_{n-1})$ *is the unique solution to the matrix equation* $M \times \alpha = (0, \ldots, 0, 1)^T$, *where* $M_{i,j} = \mathsf{Tr}(g^{i+j})$, *i.e.,*

$$M = \begin{bmatrix} \mathsf{Tr}(g^0) & \mathsf{Tr}(g^1) & \cdots & \mathsf{Tr}(g^{n-1}) \\ \mathsf{Tr}(g^1) & \mathsf{Tr}(g^2) & \cdots & \mathsf{Tr}(g^n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathsf{Tr}(g^{n-1}) & \mathsf{Tr}(g^n) & \cdots & \mathsf{Tr}(g^{2n-1}) \end{bmatrix}.$$

We can now define the TS-Hash step function. It takes as input a state and a message bit, although the message bit is not always used.

Definition 1. Define the TS-Hash step function as

$$F : \mathbb{F}_{2^n} \times \mathbb{F}_2 \to \mathbb{F}_{2^n} : (s, m) \mapsto g \cdot s + h \cdot m \cdot \mathsf{Tr}(\alpha \cdot s),$$

where $g, h, \alpha \in \mathbb{F}_{2^n}$ are fixed such that $\mathsf{Tr}(\alpha \cdot g^{-1} \cdot h) = 0$. We say that a state s is *controlled* if $\mathsf{Tr}(\alpha \cdot g^{-1} \cdot s) = 1$.

Remark 1. $\mathsf{Tr}(\alpha \cdot s)$ corresponds to the coefficient of X^{n-1} in $s = s(X)$, while $\mathsf{Tr}(\alpha \cdot g^{-1} \cdot s)$ corresponds to the constant coefficient in $s = s(X)$. We know that this term in h is equal to zero, since it is equal to 1 in both p_0, p_1 by assumption that they are irreducible.

[1] For simplicity, we assume that $p_0(X)$ is irreducible and thus defines a finite field.

The notion "controlled" refers to the output state for convenience reasons: $s' = F(s, m)$ is controlled when it depends on m. The condition on trace of h implies that this property is dependent only on the state, since, for any m,

$$\mathsf{Tr}(\alpha \cdot g^{-1} \cdot s') = \mathsf{Tr}(\alpha \cdot s) + m \cdot \mathsf{Tr}(\alpha \cdot g^{-1} \cdot h) = \mathsf{Tr}(\alpha \cdot s).$$

Although in TS-Hash the message bits are consumed only at controlled states, we can assume that they are consumed at each step: this corresponds to injection of arbitrary message bits at uncontrolled positions.

Definition 2. The *extended* TS-Hash $H(m) : \mathbb{F}_2^N \to \mathbb{F}_2^n$ is defined as the final state s_N, where the initial state $s_0 \in \mathbb{F}_2^n$ is a public constant and intermediate states are given by

$$s_i = F(s_{i-1}, m_i), \quad 1 \le i \le N.$$

Definition 3. The *compressed* (original) TS-Hash $\hat{H}(\hat{m}) : \mathbb{F}_2^M \to \mathbb{F}_2^n$ is defined as s_N, where the initial state $s_0 \in \mathbb{F}_2^n$ is a public constant, $j_0 = 1$, intermediate values for $1 \le i \le N$ are given by

$$s_i = F(s_{i-1}, \hat{m}_{j_{i-1}}),$$
$$j_i = \begin{cases} j_{i-1}, & \text{if } s_i \text{ is not controlled,} \\ j_{i-1} + 1, & \text{if } s_i \text{ is controlled.} \end{cases}$$

and N is smallest integer such that $j_N = M$.

Remark 2. This definition matches the C-style pseudocode given in the beginning of the section.

2.2 Explicit algebraic expression of TS-Hash

Using the algebraic formulation of a TS-hash step, we can now derive an algebraic expression of the output state in terms of the extended input message bits.

Theorem 1. *Let N be a positive integer and let $s_N = H(m)$ for $m = (m_1, \ldots, m_N) \in \mathbb{F}_2^N$. Then,*

$$s_N = s_0 g^N + \sum_{\substack{I \subseteq \{1,\ldots,N\} \\ I \neq \emptyset}} \underbrace{\left(\prod_{i \in I} m_i\right)}_{\text{monomial in } m} \cdot \underbrace{hg^{N-\max(I)}}_{\text{constant}} \cdot \underbrace{\mathsf{Tr}(\alpha s_0 g^{\min(I)-1}) \cdot \prod_{j=2}^{|I|} \mathsf{Tr}(\alpha h g^{i_j - i_{j-1} - 1})}_{\text{control bits}} \quad (1)$$

where $i_1 < i_2 < \ldots < i_{|I|}$ are all the indices from I in sorted order.

Proof. The proof can be done by induction on N using the step expression $(s_{i-1}, m_i) \mapsto g \cdot s_{i-1} + h \cdot m_i \cdot \mathsf{Tr}(\alpha \cdot s_{i-1})$. □

This result indicates on the extreme sparseness of the algebraic normal form of the output state in terms of the extended message bits. Indeed, a degree-d monomial is controlled by a product of d bits (trace functions), and thus, by a rough estimation, it can be expected to have about $\sum_{i=0}^{N} \binom{N}{i} 2^{-i} = 1.5^N$ monomials. This set of possible monomials is very structured: only certain distances between consequtive variable indexes can occur. For example, if $\mathsf{Tr}(\alpha h g^2) = 0$, then, for all i, any present monomial not having variables m_{i+1}, m_{i+2} can never contain (be divisible by) $m_i m_{i+3}$. Furthermore, each of the n output bits only contains a subset of this set of monomials, depending on a coordinate of the constant $hg^{N-\max(I)}$.

3 Attacks on TS-Hash

The authors of TS-Hash observed that an all-zero message $m = 0^N$ can be used as a preimage for (some) values of the hash function by choosing an appropriate value of N, which amounts to computing a discrete logarithm in $\mathbb{F}_{2^n}$. Indeed, the hash value of $m = 0^N$ is simply $s_0 g^N$. This attack is not deemed meaningful as it would typically produce unrealistically long messages.

However, this attack idea can be generalized to deviate from the all-zero message. If we set $m_i = 1$ at a controlled state s_i, we have the new value

$$s'_i = s_i + h = s_0 g^i + h,$$

so that the new final state is

$$s'_N = (s_0 g^i + h) g^{N-i} = s_0 g^N + h g^{N-i}.$$

This naturally generalizes to multiple message bits set to 1.

Proposition 2. *Let $m \in \mathbb{F}_2^N$ be such that $m_i = 1$ if and only if $i \in \{e_1, \ldots, e_k\}$, where $1 \le e_1 < \ldots < e_k \le N$ and all states $s_{e_1}, \ldots, s_{e_k}$ are controlled. Then,*

$$H(m) = s_0 g^N + h(g^{N-e_1} + g^{N-e_2} + g^{N-e_3} + \ldots + g^{N-e_k}). \tag{2}$$

This representation suggests to choose the exponents e_i arbitrarily to match the desired hash value T. However, some of the respective states would likely happen to be uncontrolled, which invalidates the representation. There are several directions of handling this problem.

1. Since a compressed message bit only affects the controllability of states after absorbing this bit, flipping bits close to the end of the message often do not affect the set of controlled states, keeping the expression (2) valid. This leads to high-probability differentials.

2. Consider the preimage problem when it is guaranteed that a short message solution exists. In this case, the shortness guarantees a unique solution to the problem implying that the necessary states are controlled. The solution can be found using simple linear algebra (e.g., Gaussian elimination).

3. More general approach is to find a small set of exponents e_i satisfying (2) and hope for them to land on controlled states by chance, repeating the process until success.

4. Ensuring a small number of controlled states in (2) can be done by manipulating the control bits which are hidden in Proposition 2 but are given in Theorem 1 and Definition 1. This leads to a small improvement of the general attack.

These directions are explored in more details in the following subsections.

3.1 High-probability differentials

Throughout this section, the probability is taken over all possible values of a certain intermediate state (e.g., s_{N-i} for a small i), which are assumed to be uniformly distributed over sufficiently long message prefixes.

The first observation is a probability-1 differential that can be clearly observed from the TS-Hash pseudocode.

Proposition 3. *For all $\hat{m} \in \mathbb{F}_2^M$ it holds $\hat{H}(\hat{m}) + \hat{H}(\hat{m} + (0, \ldots, 0, 1)) = h$ with probability 1.*

For the second-to-last bit, the situation varies depending on h.

Proposition 4. *If* $\mathsf{Tr}(\alpha h) = 0$*, then, for all* $\hat{m} \in \mathbb{F}_2^M$*, it holds* $\hat{H}(\hat{m}) + \hat{H}(\hat{m} + (0, \ldots, 0, 1, 0)) = gh$ *with probability 1/2.*

For the third-to-last bit, an extra constraint on h leads to a similar result:

Proposition 5. *If* $\mathsf{Tr}(\alpha h) = \mathsf{Tr}(\alpha g h) = 0$*, then, for all* $\hat{m} \in \mathbb{F}_2^M$*, it holds* $\hat{H}(\hat{m}) + \hat{H}(\hat{m} + (0, \ldots, 0, 1, 0, 0)) = g^2 h$ *with probability 1/4.*

This idea can be easily generalized for farther bit-flips and output differences (depending on h).

3.2 Preimage attack on short messages

In this preimage attack, we require a promise that there exists a (compressed) message of length ℓ resulting in the target hash. Note that this is different from the second preimage attack, where the first message is given to the adversary. Since the length of the message depends on the representation (compressed/extended), this section works with both representations.

Recall the equation (2). Assume that the length N of the extended message is known. Then, we can represent the unknown set of exponents e_i by unknown binary coefficients $\lambda_i \in \mathbb{F}_2$:

$$H(m) = s_0 g^N + h \cdot \left(\sum_{i=0}^{N} \lambda_i g^i \right),$$

This gives a linear equation system over $\mathbb{F}_2$ with n equations on the variables $\lambda_i \in \mathbb{F}_2$. Assuming g is a generator, its minimal polynomial has degree n and the first n powers g^i are linearly independent. Thus, the set of coefficients λ_i (equivalent to the set of exponents e_i) is uniquely determined if $N \leq n$, or has 2^t candidates if $N = n + t$. Since the extended message is on average twice as large as the compressed message, this condition is satisfied for most compressed messages of length $\ell \leq n/2$. We conclude that hash digests of most half-block messages can be attacked in polynomial time.

Extending the attack The attack can be combined with an exhaustive search of a prefix (or a suffix) of the message. Assume that we exhaustively check the first p compressed message bits. We can now apply the linear algebraic attack on the $(\ell - p)$-bit message, assuming $2(\ell - p)$-bit extended representation. The rank-n system then yields $2^{2\ell - 2p - n}$ candidate solutions, totaling in $2^{2\ell - p - n}$ solutions when counting the 2^p guessed prefixes. This computation suggests that p should be maximized, as long as the number of candidate solutions is not negligible. Indeed, an extra compressed message bit costs 1-bit guess in the prefix guess or 2-bit (4-wise) increase in the number of candidates in the linear system due to the extension. Thus, a $(p + n/2)$-bit compressed message can be recovered in time $\mathcal{O}(2^p n^c)$ for some constant c depending on the computational model. In the extreme, for an arbitrary target value we can expect to have an n-bit compressed preimage, leading to the attack cost $\tilde{\mathcal{O}}(2^{n/2})$, comparable to the general meet-in-the-middle approach proposed in [BT23b].

The attack naturally applies to preimages of longer message with a *known* prefix and/or suffix, such that the unknown middle part is of small length.

3.3 Generalized birthday general preimage attack

Consider now the general preimage attack for arbitrary hash digest values. In this section, we only consider the extended representation. Recall the polynomial equation for TS-Hash:

$$H(m) = s_0 g^N + h(g^{N-e_1} + g^{N-e_1-e_2} + g^{N-e_1-e_2-e_3} + \ldots + g^{N-e_1-\ldots-e_k}),$$

which applies when states s_{e_i} are controlled. Although we can not ensure that the chosen states are controlled, we can hope that it happens by chance. Each bit is controlled with probability $1/2$, so that for k monomials we need about 2^k "random" attempts to succeed. We thus arrive at the problem of representing $\frac{H(m)-s_0 g^N}{h}$ by a very sparse polynomial in g with binary coefficients. As the degree of the polynomial defines the message length, it has to be bounded too.

Problem 1. *Given $g, t \in \mathbb{F}_{2^n}$, find a polynomial $q(x) \in \mathbb{F}_2[x]$ of low degree and few nonzero coefficients, such that $q(g) = t$.*

It follows that a preimage attack on TS-Hash can be reduced to finding about 2^k solutions with k nonzero coefficients to the above problem. Note that a given polynomial may be tested efficiently (to have all necessary bits controlled) by computing only the relevant states: one such jump requires computing a power of g using fast exponentiation. This is in contrast to direct evaluation of all N states of TS-Hash, which can be slow for long messages (arising from a possibly high degree of the polynomial q).

In particular, Problem 1 can be solved by the *generalized birthday algorithm* by Wagner [Wag02].

Generalized birthday procedure Consider the list $L_0 = \{g^0, g^1, \ldots, g^{2^{t+1}-1}\}$. Using a hash-table on t (say) least-significant bits, we can find all pairs (i, j) such that $g_i + g_j$ has t least significant bits equal to zero. We can expect $\binom{|L|}{2} 2^{-t} \approx 2^{t+1}$ such pairs. Denote the new list by L_1. By repeating the procedure on the next t bits on pairs from L_1, we get a new list L_2 of the same size with $2t$ chosen bits equal to zero. After $d = \left\lceil \frac{n}{t} \right\rceil$ iterations, we obtain a list L_d containing zero vectors, obtained as sums of 2^d monomials g^i. Complexity of this procedure is $\mathcal{O}\left(d2^t\right) = \mathcal{O}\left(d2^{n/d}\right)$ hash-map and field operations.

Although we obtain a polynomial evaluating to zero at g, this approach can be easily adapted to makes the sum equal to any predetermined value instead of zero. This requires maintaining lists T_i with ti least significant bits matching the target value. These lists can be merged from T_{i-1} and L_{i-1}. Thus, the total number of list merges is only doubled.

In order to mount the preimage attack on TS-Hash, we need 2^{2^d} such solutions (since we have 2^d monomials). The total complexity is thus $\mathcal{O}\left(2^{2^d} n^c + d2^{n/d}\right)$, where n^c corresponds to the complexity of testing a candidate polynomial in g. Setting $d = \log n - \log\log n$, we get complexity

$$\mathcal{O}\left(2^{\frac{n}{\log n}} n^c + (\log n) 2^{\frac{n}{\log n - \log\log n}}\right) = \mathcal{O}\left((\log n) 2^{\frac{n}{\log n - \log\log n}}\right).$$

More concretely, for $n = 128$, we can use $d = 5$ to have lists of length 2^{32} over 5 levels and testing 2^{32} final candidate solutions, which is practical.

The attack was implemented and verified using SageMath [Sag23] on $n = 80$ taking about 2 minutes.

Second-preimage attack Solving Problem 1 with $t = 0$ allows to find a collision for the TS-Hash function. This problem is equivalent to finding a low-weight multiple of the minimal polynomial of g, which is $p_0(X)$; it was actively studied in relations to attacks on stream ciphers [Sie86, MS88, CJM02, Aim21].

3.4 Prefix/suffix linearization technique

We will now try to linearize the TS-hash function. Recall the expression from Theorem 1:

$$s_N = s_0 g^N + \sum_{\substack{I \subseteq \{1,\dots,N\} \\ I \neq \emptyset}} \underbrace{\left(\prod_{i \in I} m_i\right)}_{\text{monomial in } m} \cdot \underbrace{hg^{N-\max(I)}}_{\text{constant}} \cdot \underbrace{\mathsf{Tr}(\alpha s_0 g^{\min(I)-1}) \cdot \prod_{j=2}^{|I|} \mathsf{Tr}(\alpha h g^{i_j - i_{j-1} - 1})}_{\text{control bit}}.$$

The goal is, for monomials m^I with $|I| \geq 2$, to force the corresponding control bits to be equal to zero. This can be done by fixing certain message bits to zero. Then, s_N will become a linear function of the remaining (non-fixed) message bits. The total control bit for each monomial consists of two part.

Steps between variable indexes Consider a monomial of degree at least 2. Its control bit is equal to zero in particular when $\prod_{j=2}^{|I|} \mathsf{Tr}(\alpha h g^{i_j - i_{j-1} - 1}) = 0$. Thus, we can choose steps between variables to ensure this condition. If we keep message bits $m_{i_1}, m_{i_2}, \dots, m_{i_k}$, $i_1 < i_2 < \dots < i_k$, then for all $j < j'$ we need that $s = i'_j - i_j$ satisfies $\mathsf{Tr}(\alpha h g^{s-1}) = 0$, otherwise the monomial $m_\ell m_{\ell+s}$ will be present at some offset ℓ. Therefore, we want to minimize the number of unique distances between active message variables. It is easy to see that this is achieved when variables are placed at a fixed step s from each other: $m_\ell, m_{\ell+s}, \dots, m_{\ell+(k-1)s}$. The set of distances between positions is then simply $s, 2s, \dots, (k-1)s$. A random choice s can be expected to satisfy the respective constraint ($\mathsf{Tr}(\alpha h g^{is-1}) = 0$ for $i \in \{1, \dots, k-1\}$) with probability 2^{-k+1}. Therefore, we can expect to find one after about 2^{k-1} trials, and it should be about $(k-1)$-bit long.

First active variable's offset The second contributor to the control bit is the term $\mathsf{Tr}(\alpha s_0 g^{\min(I)-1})$. It only depends on the smallest index of a variable involved in the monomial. Since we aim to only allow (and enforce) linear monomials, it has to be equal to 1. Otherwise, the chosen active bits would simply become inactive. In order to keep all k chosen bits active, it must hold

$$\mathsf{Tr}(\alpha s_0 g^{\ell+is-1}) = 1 \quad \forall i \{0, \dots, k-1\}.$$

A good such offset ℓ can be found again by random sampling, since all these constraints are satisfied with probability 2^{-k}. Note that this sampling is done after the first step, and thus the complexity is added (and not multiplied).

We conclude with the final linearization procedure:

1. choose parameter k;
2. sample random $s \geq 1$ until $\mathsf{Tr}(\alpha h g^{is-1}) = 0$ for all $i \in \{1, \dots, k-1\}$;
3. sample random $\ell \geq 1$ until $\mathsf{Tr}(\alpha s_0 g^{\ell+is-1}) = 1$ for all $i \in \{0, \dots, k-1\}$.

Both sampling steps can be expected to work in time $\mathcal{O}\left(2^k\right)$ finite field operations.

Theorem 2. *Let k, s, ℓ satisfy the conditions above and let $m \in \mathbb{F}_2^{\ell+ks}$ be such that $m_{\ell+is} = x_{i+1}$ for $i \in \{1, \dots, k\}$ and $m_j = 0$ in all other positions, where $x_i \in \mathbb{F}_2$ are variables. Then, $H(m)$ is a linear function in $x_1, \dots, x_k$ and is non-degenerate in each variable.*

Note that this result does not imply that the TS-Hash itself is linearized: compressed messages corresponding to various choices of $x_1, \dots, x_k$ would typically even have different size. However, this is a useful tool for cryptanalytic purposes, since any extended message (e.g., a preimage solution) always has a corresponding compressed message.

Suffix linearization This technique can be modified to work in the decryption direction.

3.5 Linearization with quadratic constraints

Recall the TS-hash step function from Definition 1:

$$F : (s, m) \mapsto g \cdot s + h \cdot m \cdot \mathsf{Tr}(\alpha \cdot s).$$

Assume that the current state s is a linear function of some previous message bits:

$$s = \beta_1 m_1 + \beta_2 m_2 + \ldots + \beta_k m_k,$$

where $\beta_i \in \mathbb{F}_2^n$ are constants. This can be achieved for example using the prefix linearization technique described previously. Our goal is to linearize the next state

$$s' = F(s, m_{k+1}) = g \cdot s + h \cdot m_{k+1} \cdot \mathsf{Tr}(\alpha \cdot s).$$

This can be done by adding a linear constraint $m_{k+1} = c$ or $\mathsf{Tr}(\alpha \cdot s) = c$ for a constant $c \in \mathbb{F}_2$. However, we would not gain a new degree of freedom unless $\mathsf{Tr}(\alpha \cdot s) = c$ holds by the construction of s.

Instead, we can use a quadratic constraint

$$(m_{k+1} + 1)(\mathsf{Tr}(\alpha \cdot s) + 1) = 0, \tag{3}$$

which implies

$$m_{k+1}\mathsf{Tr}(\alpha s) = (m_{k+1} + 1)(\mathsf{Tr}(\alpha s) + 1) + m_{k+1} + \mathsf{Tr}(\alpha s) + 1 = m_{k+1} + \mathsf{Tr}(\alpha s) + 1,$$

and so

$$s' = g \cdot s + h \cdot m_{k+1} + h \cdot \mathsf{Tr}(\alpha s) + h,$$

which is now a linear non-degenerate function of $m_1, \ldots, m_{k+1}$ under the constraint (3).

Theorem 3. *There exist at most $M - 1$ quadratic constraints of the form*

$$\langle \beta_i, (1, m_1, \ldots, m_i) \rangle \cdot (m_{i+1} + 1) = 0, \quad 1 \le i \le M - 1,\ \beta_i \in \mathbb{F}_2^{i+1}$$

such that the extended TS-hash function $H(m)$ is linear in m for all messages m satisfying the constraints. Furthermore, the linear representation of H and the constraints can be computed in polynomial time.

This result can be used to mount preimage attack in two ways. The advantage of this approach compared to the generalized birthday attack is that it can bin principle be mounted with short messages for arbitrary target digests.

Finding preimages by solving the quadratic equation system The direct approach is to simply solve the system of quadratic constraints and linear equations arising from equating s_M with the target digest value). The simple structure of the constraint system may potentially lead to efficient methods. This avenue is left for future research.

Probabilistic preimage solving A rather straightforward technique is to sample a random solution to the linear part of the system (which is under-defined for $M > n$), and hope to satisfy the quadratic constraints by chance. Indeed, each quadratic constraint can be expected to be satisfied with probability $3/4$ due to its shape, leading to probability $(3/4)^{M-1}$ of the partial solution to be correct. It follows that it is more efficient to choose $M = n$ and to randomize the target preimage by doing random reverse steps from the original target.

The attack can be combined with the prefix/suffix linearization: these bits simply provide extra degrees of freedom without adding quadratic constraints. Recall that k-dimensional linearization can be done in time $\mathcal{O}\left(2^k\right)$, both for the prefix and the suffix. Then, the $n-2k$ missing dimensions have to be covered with quadratic constraints, costing a factor 4/3 each. This leads to complexity $\tilde{\mathcal{O}}\left(2^k + (4/3)^{n-2k}\right)$, which is minimized when the two parts are balanced, namely when $k = \frac{n}{2+\log_{4/3} 2} \approx 0.227n$, leading to final attack complexity $\tilde{\mathcal{O}}\left(2^{0.227n}\right)$, which is significantly better than the basic meet-in-the-middle complexity $\tilde{\mathcal{O}}\left(2^{0.5n}\right)$, while asymptotically worse than the generalized birthday complexity $\tilde{\mathcal{O}}\left(2^{\frac{n}{\log n - \log\log n}}\right)$.

4 Discussion

As described in [BT23b], the TS-Hash is an instance of the so-called Cayley / group-theoretic hash functions [TZ94]. Given linear bijections $T_0, T_1 : \mathbb{F}_2^n \to \mathbb{F}_2^n$, a nonlinear bijection $S : \mathbb{F}_2^n \to \mathbb{F}_2^n$ and a starting state $s_0 \in \mathbb{F}_2^n$, a hash function h can be defined such that for a message $m = (m_1, m_2, \ldots, m_M) \in \mathbb{F}_2^M$ it is given by

$$h(m) = T_{m_M} \circ S \circ \ldots \circ T_{m_2} \circ S \circ T_{m_1} \circ S(s_0).$$

In particular, the TS-Hash is given by the "shift" function $S(s||0) = S(0||s), S(s||1) = (s||1)$, and the T_0, T_1 being multiplications by X in $\mathbb{F}_2[X]/(p_0(X))$ and $\mathbb{F}_2[X]/(p_1(X))$ respectively.

This structure can in principle be secure, since the modern sponge-based hashing can be viewed as an example it (with single-bit message injection). However, this requires a sufficiently strong nonlinear mixing map S, which is not the case for TS-Hash. Furthermore, there is a lot of interaction between the functions S, T_0, T_1: the functions S and T_0 "almost" commute, and T_1 is equal to T_0 with a constant addition.

4.1 Countermeasures

The rest of the section discusses potential countermeasures.

Output truncation Typical sponge-based hash functions output only part of the final state ("squeezing"). Truncating the TS-hash function could lead to increase in security against some of the attacks, for a fixed output length: the meet-in-the-middle attack [BT23b] requires full-state collision, so that its complexity $\mathcal{O}\left(2^{n/2}\right)$ for n-bit state can be worse than generic exhaustive preimage search for an output size less than $n/2$. However, this does would not affect the short message attack and the generalized birthday attack.

External padding of the message A natural idea for protecting the hash function against short message attacks is to pad it with fixed bits. However, a prefix padding does not add any security since it only adjusts the starting state. A suffix padding is similarly useless on itself, but it can be more useful in combination with truncation: the map

$$P_\ell = (T_1 \circ S \circ T_0 \circ S)^\ell$$

corresponding to padding the message suffix with $(0, 1, 0, 1, \ldots)$ for a sufficiently large ℓ (for example, $\ell = 2n$) may provide a good source of nonlinearity to thwart the linearity-based attacks (e.g., the short-message and the generalized birthday attacks). This requires to increase the state size n sufficiently to secure against the generalized birthday and similar attacks, and truncating the output state (after processing the padding) to prevent the inversion of the padding function.

Remark 3. Alternation between 0 and 1 ensures that neither $\mathbb{F}_2[X]/(p_0(X))$ nor $\mathbb{F}_2[X]/(p_1(X))$ representations lead to a small number of possible linearizations. For example, padding with an all-1 suffix would make the padding map equivalent to the multiplication by $X^{\ell\pm\varepsilon}$ in the second field, for some small value of ε.

Note that securing the internal state against the generalized birthday attack requires state size more than 512 to achieve 128-bit security, which makes the hash function much less attractive, compared to the original variant.

Internal padding of the message Another countermeasure aiming to increase the confusion of absorbing the message bits is to inject an internal padding (constant bits) between the consecutive message bits. Similarly to the external padding, alternating 0-1 bits are a good candidate. In other words, replace the step function $T_{m_i} \circ S$ by

$$P_{\ell'} \circ T_{m_i} \circ S \;=\; (T_1 \circ S \circ T_0 \circ S) \;\circ\; (T_1 \circ S \circ T_0 \circ S) \;\circ\; \ldots \circ T_{m_i} \circ S$$

for some small number ℓ', for example $\ell' = 8$. This countermeasure makes it much more difficult to control states. For example, in the generalize birthday attack, the cost of controlling one summand g^{e_i} increases significantly: the attacker has now to match both g^{e_i} and additional ℓ' summands arising from the internal padding, decreasing the probability from $1/2$ to $1/2^{\ell'}$.

4.2 Conclusion

This section outlined two countermeasures against the proposed attacks. Concrete evaluation of these countermeasures as well as choices of suitable parameters is left as a future work.

It is an interesting open problem whether there exists an instantiation of TS-Hash with countermeasures that has still interesting properties, such as being lightweight in code size and having a small state, or being extendable (i.e., the digest is equal to the full internal state, allowing incremental hashing). The latter property however is not preserved under the truncation countermeasure.

Acknowledgements

The work was supported by the Luxembourg National Research Fund's (FNR) and the German Research Foundation's (DFG) joint project APLICA (C19/IS/13641232). The author thanks Alex Biryukov for discussions about early stream ciphers.

References

[Aim21] Laila El Aimani. A new approach for finding low-weight polynomial multiples. *IACR Cryptol. ePrint Arch.*, 2021:586, 2021. 6

[BT23a] Itay Bookstein and Boaz Tsaban. TS-Hash: a lightweight cryptographic hash family based on Galois LFSRs. Cryptology ePrint Archive, Paper 2023/179, 2023. https://eprint.iacr.org/2023/179. 1

[BT23b] Itay Bookstein and Boaz Tsaban. TS-Hash: a lightweight cryptographic hash family based on Galois LFSRs. *Mathematical Cryptology*, 3(1):96–106, Aug. 2023. 1, 5, 9

[CJM02] Philippe Chose, Antoine Joux, and Michel Mitton. Fast correlation attacks: An algorithmic point of view. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 209–221. Springer, Heidelberg, April / May 2002. 6

[MS88] Willi Meier and Othmar Staffelbach. Fast correlation attacks on stream ciphers (extended abstract). In C. G. Günther, editor, *EUROCRYPT'88*, volume 330 of *LNCS*, pages 301–314. Springer, Heidelberg, May 1988. 6

[Sag23] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 10.0)*, 2023. https://www.sagemath.org. 6

[Sie86] Thomas Siegenthaler. Cryptanalysts representation of nonlinearly filtered ML-sequences. In Franz Pichler, editor, *EUROCRYPT'85*, volume 219 of *LNCS*, pages 103–110. Springer, Heidelberg, April 1986. 6

[Tsa17] Boaz Tsaban. [Examples of research problems for a master's thesis in mathematical aspects of cyber security and cryptography]. http://u.cs.biu.ac.il/~tsaban/CyberIntro.html, 2017. Accessed at https://web.archive.org/web/20200205204442/http://u.cs.biu.ac.il:80/~tsaban/CyberIntro.html. 1

[TZ94] Jean-Pierre Tillich and Gilles Zémor. Group-theoretic hash functions. In G. Cohen, S. Litsyn, A. Lobstein, and G. Zémor, editors, *Algebraic Coding*, pages 90–110, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg. 9

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, Heidelberg, August 2002. 6