TS-HASH: A LIGHTWEIGHT CRYPTOGRAPHIC HASH FAMILY BASED ON GALOIS LFSRS

ITAY BOOKSTEIN AND BOAZ TSABAN

Abstract. We study a novel family of cryptographic hash functions based on Galois linear feedback shift registers (LFSRs), and identify initial guidelines for choosing secure parameters for this family. These hash functions are extremely simple, efficient, and suitable for implementation in constrained environments.

1. Introduction

We study a new family of cryptographic hash functions for constrained environments. This family is an instantiation of the following abstract construction. For a fixed natural number k, let $f_0, \ldots, f_{k-1}$ be actions on a set V, that is, functions from the set V into itself. Throughout this paper, we denote function composition by multiplication. Fix an initial value v_0 in V. A message to be hashed is coded as a sequence $(b_0, \ldots, b_{\ell-1})$ of elements in $\{0, \ldots, k-1\}$. The hash of the message is defined by

$$\begin{aligned} h(b_0, \ldots, b_{\ell-1}) &:= f_{b_{\ell-1}} f_{b_{\ell-2}} \cdots f_{b_1} f_{b_0}(v_0) = \\ &= f_{b_{\ell-1}}(f_{b_{\ell-2}}(\cdots f_{b_1}(f_{b_0}(v_0)) \cdots)). \end{aligned}$$

For bit by bit hashing, we let $k = 2$. Using larger k may improve the efficiency of the hash function, but this does not enhance the security of this hash function. Indeed, the attacker can choose to hash only strings of zeros and ones.

Cayley hash functions [9, 8] form an important special case of this scheme: There, the set V is a *group*, the initial element v_0 is the group identity element e, and each function f_i is defined by $f_i(g) := h_i \cdot g$, for a fixed group element h_i. In this case,

$$h(b_0, \ldots, b_{\ell-1}) = h_{b_{\ell-1}} \cdot h_{b_{\ell-2}} \cdots h_{b_1} \cdot h_{b_0},$$

an element of the group V.

The case $V = \mathrm{SL}_2(\mathbb{F})$, the group of 2×2 matrices with determinant 1 over a field of cardinality 2^n, is called SL_2-Hash. It was thoroughly studied [8, 1, 6, 3, 7, 4]. SL_2-Hash remains secure when the group elements are chosen at random [5, and references therein].

2020 *Mathematics Subject Classification.* 94A60,94A55,11C08.

Key words and phrases. Cryptography, Cryptanalysis, Cryptographic Hash Functions, Galois LFSRs, TS-Hash.

This research was supported by the BIU Center for Research in Applied Cryptography and Cyber Security in conjunction with the Israel National Cyber Directorate in the Prime Minister's Office, and by the Check Point Institute for Information Security.

This hash function is equally secure if we let V be the vector space $\mathbb{F}^2$, and the functions on V are defined by matrix multiplication on the left [6]. This motivates the search of additional useful instantiations of the abstract construction.

We consider *TS-Hash* [10], a particularly simple and efficient instantiation of the abstract construction. The definitions are provided in Section 2. Section 3 describes basic mathematical properties of this hash function, including a mild provable security result that cannot be established for ad-hoc hash functions such as the SHA family. Section 4 includes an initial cryptanalytic study of TS-Hash, showing that extremely degenerated feedback polynomials should not be used for this hash function to be secure.

2. TS-Hash

2.1. **General definition.** Fix a natural number k. Let $T_0, \ldots, T_{k-1}$ be linear transformations defined on a vector space V. Let $S\colon V \to V$ be a function such that, for each nonzero vector $v \in V$, the elements

$$T_0 S(v), \ldots, T_{k-1} S(v)$$

are all distinct. Then *TS-Hash* is the hash function defined by the abstract construction, using the functions $T_0 S, \ldots, T_{k-1} S$. Explicitly, we fix an initial nonzero vector $v_0 \in V$, and define

$$h(b_0, \ldots, b_{\ell-1}) = T_{b_{\ell-1}} S\, T_{b_{\ell-2}} S \cdots T_{b_1} S\, T_{b_0} S(v_0).$$

Remark 2.1. The condition on the function S must hold, at least with overwhelming probability, for the hash function to be collision-resistant. Indeed, assume that, with non-negligible probability, the hash value v of a random string $(b_0, \ldots, b_{\ell-1})$ has $T_i S(v) = T_j S(v)$, for some distinct i and j. Then we have a collision

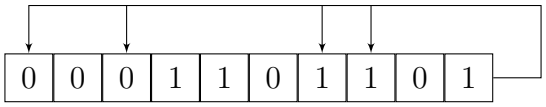
$$\begin{aligned} h(b_0, \ldots, b_{\ell-1}, i) &= \\ &= T_i S\, T_{b_{\ell-1}} S\, T_{b_{\ell-2}} S \cdots T_{b_1} S\, T_{b_0} S(v_0) = \\ &= T_i S(v) = T_j S(v) = \\ &= T_j S\, T_{b_{\ell-1}} S\, T_{b_{\ell-2}} S \cdots T_{b_1} S\, T_{b_0} S(v_0) = \\ &= h(b_0, \ldots, b_{\ell-1}, j). \end{aligned}$$

In some cases, the function S may be the identity function. Indeed, this is essentially the case in the projective version of SL_2-hash [6]. In the main concrete example considered in this paper, the function S is necessary, for the above reason.

2.2. **Implementing with Galois LFSRs.** A *Galois LFSR* is a linear-feedback shift register (LFSR) represented by a linear transformation $T\colon \mathbb{F}_2^n \to \mathbb{F}_2^n$ of the following type:

$$T(x_0, x_1, \ldots, x_{n-1}) := (0, x_0, \ldots, x_{n-2}) + x_{n-1} \cdot (c_0, c_1, \ldots, c_{n-1}),$$

where $c_0, \ldots, c_{n-1} \in \{0, 1\}$ are fixed. Thus, the state is shifted once, and if the discarded bit x_{n-1} is 1, the *mask vector* $(c_0, \ldots, c_{n-1})$ is XORed to the state. Equivalently, after the shift, the bit x_{n-1} is XORed to the places i where $c_i = 1$.



Galois LFSRs arise from Galois field extensions, as follows. Using the above notation, let

$$p(X) := c_0 + c_1 X + \cdots + c_{n-1} X^{n-1} + X^n,$$

the *feedback polynomial* of the tranformation T. Assume that the polynomial $p(X)$ is irreducible, and identify the vector space $\mathbb{F}_2^n$ with the Galois extension field $\mathbb{F}_2[X]/\langle p(X)\rangle$, viewed as a vector space spanned by $1, X, \ldots, X^n$. Then the induced linear transformation on the latter space is

$$q(X) \mapsto X \cdot q(X) \bmod p(X).$$

In the instantiation we consider, the linear transformations $T_0, \ldots, T_{k-1}$ are Galois LFSRs. In this case, there is a simple function S as required in the definition of TS-hash.

Definition 2.2. The *shift* of a nonzero vector in $\mathbb{F}_2^n$ is defined by

$$S(x_0, x_1, \ldots, x_{n-1}) := (0, \ldots, 0, x_0, x_1, \ldots, x_{n-m})$$

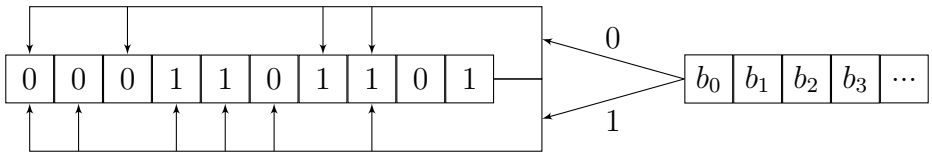
where m is minimal with $x_{n-m} \neq 0$. That is, the vector is shifted to the right, until the rightmost bit is 1.

The shift map S is nonlinear. It is equivalent to repeated application of a Galois LFSR, with any feedback polynomial, until the rightmost bit is 1.

Lemma 2.3. *Let T and R be distinct Galois LFSRs. Then $TS(v) \neq RS(v)$ for all nonzero vectors v.*

Proof. In $S(v)$, the rightmost bit is 1. Thus, in the application of T and R to $S(v)$, the (distinct) feedback masks are XORed to the one-bit shift of $S(v)$, and the results are distinct. □

Consider the case $k = 2$. The input to the hash function is a string of bits, and the function is illustrated by the following figure, keeping in mind that the shift function is applied prior to each application of the LFSRs.



Following is the C-like pseudocode of this hash function. Despite being very close to a true implementation, it is remarkably simple and short.

```
word Hash(bitstring b) {
    word mask[2] = {p0, p1};
    word v = v0;
    for (i = 0; i < length(b); i=i+1) {
        while ((v & 1) == 0) v >>= 1;    // v = S(v)
        v = (v >> 1) ^ mask[b[i]];       // v = T_{b[i]}(v)
    }
    return v;
}
```

On average, the shift function performs two conditional single-bit shifts per iteration. The application of a Galois LFSR to a *shifted* element only requires one shift and one XOR with the feedback mask.

On modern processors, the shift function can be implemented using a single CPU instruction, by finding the required shift amount m [12]. Knowing m, we can shift by $m+1$ (this includes the last shift in the pseudocode) in one CPU instruction, and then perform one memory access, and one XOR operation. This amounts to very few clock cycles per each hashed bit, if the entire inner state is stored in one CPU word. In general, the complexity grows proportionally to the number of CPU words needed to store the inner state.

Popular word-oriented hash functions, such as the SHA family, are more efficient than TS-Hash as instantiated here. There are obvious variations of this instantiation. For example, using $k = 256$, we read 8 bits of the input string on each step, and thus gain a speedup factor close to 8, at the price of 256 words of memory. Word-oriented LFSRs [11, 13] can also be used in word-oriented TS-Hash, especially if the state space is too large for one machine word. We leave the consideration of the general case for future studies, and consider here the most basic instantiation.

2.3. **The feedback polynomials and parameter choices.** For a Galois LFSR to be invertible, the bit c_0 in its feedback mask (which is the constant term in its feedback polynomial) must be 1. To have a chance that it has a maximal cycle covering all nonzero states, the feedback polynomial must be irreducible, but this is insufficient in general. Feedback polynomials providing maximal periods are called *primitive polynomials*, and there are efficient tests for the primitivity of a polynomial. Thus, we suggest choosing the initial vector v_0 uniformly at random among the nonzero vectors, and the feedback polynomials uniformly at random among the primitive ones. This is done by choosing random polynomials repeatedly, until a primitive one is obtained.

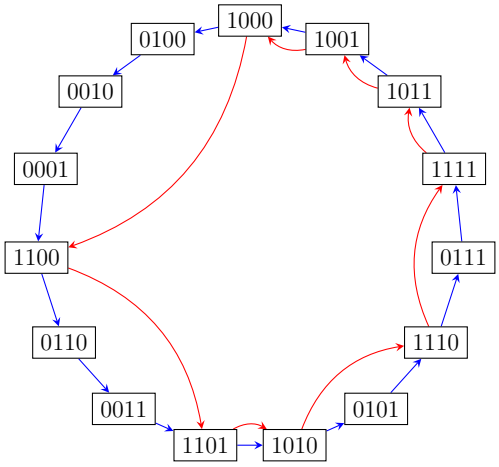
Degenerate polynomials should be avoided; we will treat this later. Here, it suffices to note that with overwhelming probability, for large vector lengths n (such as $n = 256$) random polynomials are not degenerate.

Since our Galois LFSRs are invertible, the leftmost bit of all hash values is 1. Thus, this bit can be discarded and the effective inner state and output size is $n - 1$ bits. This makes for a small constant factor in security estimations.

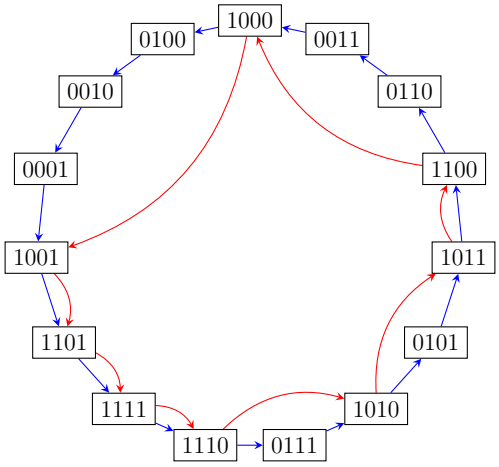
Since the feedback polynomials are irreducible, the parity of their mask is even. By the definition of the shift operator, in every step we remove from the state v one set bit, and change an even number of bits in the state. This flips the parity of the state. Thus, the parity of the hash value is determined by the parity of the length of the message. This is an interesting property, but if it has any effect on the effective output length (e.g., if only even-length messages are hashed), it reduces it by at most one additional bit. Taking everything into account, it seems that $n = 256$ is a secure choice for TS-Hash for all purposes.

3. Mathematical features

3.1. **Graph visualization.** TS-Hash may be viewed as a walk on a graph, where vertices represent states and edges represent the possible transitions between states via application of single iterations T_iS. This is a directed 2-regular graph. Let us first consider the graph corresponding to a single feedback polynomial, that is, hashes of messages of the form 0^ℓ or 1^ℓ:



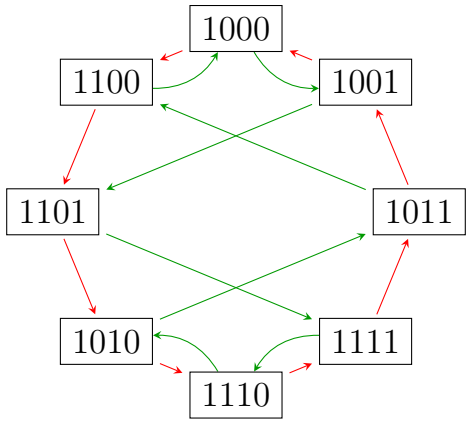
$p_0(X) = 1 + X + X^4 = 1100$



$p_1(X) = 1 + X^3 + X^4 = 1001$

The blue edges describe the action of the Galois LFSR T_i with the corresponding feedback polynomial. The red edges represent the action of the iteration function T_iS. Since both polynomials are primitive, the red subgraphs have the same vertices: These are all possible $2^{4-1} = 8$ state words with leftmost bit 1. TS-Hash takes a walk in the graph with the same vertices, but alternates between the red edges on the left cycle and those on the right cycle, according to the message input bits. This is described by the following graph, where the walk follows the red arrow for each input bit 0, and the green arrow for each input bit 1.

The graph of TS-Hash is the union of the red cycles from the above two figures. We show it here, with red edges for T_0S and green edges for T_1S:



The provided figures are just toy examples. For example, for large values of n, the combined graph is unlikely to have short cycles.

3.2. **Provable security.** In contrast to presently used efficient hash functions, TS-Hash is based on a clear, well studied mathematical concept, namely, Galois LFSR. As such, it has the potential to have some provable security guarantees. We demonstrate this with the following mild security guarantee: No subexponential collisions among "all 0" or "all 1" messages are possible. Such a guarantee is not available for ad-hoc hash functions such as the SHA family.

Proposition 3.1. For TS-Hash, there are no collisions of the form $h(0^\ell) = h(0^m)$, or $h(1^\ell) = h(1^m)$, for $\ell, m < 2^{n-1}$.

Proof. It suffices to consider 0-strings. We may assume that $\ell < m$. Since the feedback polynomial is primitive, the cycle of the corresponding Galois LFSR covers all $2^n - 1$ nonzero states, and the cycle of the map $T_0 S$ covers all 2^{n-1} states with leftmost bit 1. For a collision of the form $h(0^\ell) = h(0^m)$, we need a cycle of length $m - \ell$, and then $m - \ell \geq 2^{n-1}$. □

We expect that future explorations will reveal additional security guarantees.

3.3. **Symbolic Calculation.** Identify the state space with the degree $n-1$ polynomials, as the vector space spanned by the monomials $1, X, \ldots, X^{n-1}$. Let $p_0(X)$ and $p_1(X)$ be the feedback polynomials of T_0 and T_1, respectively. For each state $v(X)$, we have:

$$\begin{aligned} S(v(X)) &= X^{n-1-\deg v(X)} \cdot v(X) \\ T_i(v(X)) &= X \cdot v(X) \bmod p_i(X) \\ T_i S(v(X)) &= X \cdot X^{n-1-\deg v(X)} \cdot v(X) \bmod p_i(X) \\ &= X^{n-\deg v(X)} \cdot v(X) \bmod p_i(X) \end{aligned}$$

Definition 3.2. A polynomial expression $X^{m_1} + \ldots + X^{m_r}$ is *ordered* if $m_1 < \ldots < m_r$.

Let $p_i(X) = 1 + X^{k_1} + \ldots + X^{k_s} + X^n, v(X) = X^{m_1} + \ldots + X^{m_r}$ be both ordered. We can calculate:

$$\begin{aligned} T_iS(v(X)) &= X^{n-\deg v(X)} \cdot v(X) \bmod p_i(X) \\ &= X^{n-m_r} \cdot (X^{m_1} + \ldots + X^{m_r}) \bmod p_i(X) \\ &= \left(X^{m_1+(n-m_r)} + \ldots + X^{m_{r-1}+(n-m_r)} + X^n\right) \bmod p_i(X) \\ &= \left(X^{m_1+(n-m_r)} + \ldots + X^{m_{r-1}+(n-m_r)}\right) + \left(1 + X^{k_1} + \ldots + X^{k_s}\right) \end{aligned}$$

The expressions in the parentheses are ordered, but the total sum is not.

4. Cryptanalysis

4.1. **Trinomials are weak keys.** Trinomials are particularly efficient in certain environments. However, we will show that they produce short collisions, at least for the simplest initial value v_0, and should thus be avoided.

Proposition 4.1. Let $v_0(X) = 1$, $p_0(X) = 1 + X^m + X^n$, and $p_1(X) = 1 + X^k + X^n$. Without loss of generality, assume that $0 < m < k < n$. Let $c = \lceil k/(n-m) \rceil$.

(1) If $c = k/(n-m)$, we have the following collisions of length $c+1$:

$$h(00^{c-1}0) = h(10^{c-1}1)$$
$$h(10^{c-1}0) = h(00^{c-1}1)$$

(2) If $c > k/(n-m)$, we have the following collisions of length $c+2$:

$$h(100^{c-1}0) = h(010^{c-1}1)$$
$$h(010^{c-1}0) = h(100^{c-1}1)$$

Proof. We will only prove the more general assertion (2); the proof for the remaining assertion (1) is similar. Assume $cm+k < cn$. By the minimality of c, for all $d < c$ we have $dm+k > dn$, and thus

$$m > (d-1)(n-m) + (n-k)$$

Calculating $h(100^{c-1}0) = h(10^{c+1})$ iteratively and using the above inequalities to order the expressions, we get:

$$\begin{aligned} T_1S(1) &= 1 + X^k \\ T_0S(T_1S(1)) &= 1 + X^{n-k} + X^m \\ (T_0S)^2(T_1S(1)) &= 1 + X^{n-m} + X^{(n-m)+(n-k)} + X^m \\ &\vdots \\ (T_0S)^{c-1}(T_1S(1)) &= 1 + X^{n-m} + X^{2(n-m)} + \ldots + X^{(c-2)(n-m)} + \\ &\quad + X^{(c-2)(n-m)+(n-k)} + X^m \end{aligned}$$

To determine the degree of the upcoming expression, we need to compare $(c-1)(n-m)+(n-k)$ to m:

$$\begin{aligned}(c-1)(n-m)+(n-k)=\\ =c(n-m)-(n-m)+(n-k)=c(n-m)+(m-k)>\\ >k+(m-k)=m\end{aligned}$$

Thus,

$$\begin{aligned}(T_0S)^c(T_1S(1))=\\ =1+X^{n-m}+\ldots+X^{(c-1)(n-m)}+X^m+X^{(c-1)(n-m)+(n-k)}.\end{aligned}$$

This expression may not be ordered; we do not know how m compares to $(c-1)(n-m)$. Finally,

$$\begin{aligned}h(10^{c+1})&=(T_0S)^{c+1}(T_1S(1))\\ &=X^{n-(c-1)(n-m)-(n-k)}\cdot(T_0S)^c(T_1S(1))\bmod p_0\\ &=X^{k-(c-1)(n-m)}\cdot(T_0S)^c(T_1S(1))\bmod p_0\\ &=1+X^{k-(c-1)(n-m)}+\ldots+X^{k-2(n-m)}+X^{k-(n-m)}+\\ &\quad X^k+X^{m+k-(c-1)(n-m)}+X^m\end{aligned}$$

We calculate $h(010^{c-1}1)$ iteratively, using the above inequalities to order the expressions:

$$\begin{aligned}T_0S(1)&=1+X^m\\ T_1S(T_0S(1))&=1+X^{n-m}+X^k\\ T_0S(T_1S(T_0S(1)))&=1+X^{n-k}+X^{(n-m)+(n-k)}+X^m\\ (T_0S)^2(T_1S(T_0S(1)))&=1+X^{n-m}+X^{(n-m)+(n-k)}+X^{2(n-m)+(n-k)}+\\ &\quad+X^m\\ &\vdots\\ (T_0S)^{c-2}(T_1S(T_0S(1)))&=1+X^{n-m}+\ldots+X^{(c-3)(n-m)}+\\ &\quad+X^{(c-3)(n-m)+(n-k)}+X^{(c-2)(n-m)+(n-k)}+X^m\\ (T_0S)^{c-1}(T_1S(T_0S(1)))&=1+X^{n-m}+\ldots+X^{(c-2)(n-m)}+\\ &\quad+X^{(c-2)(n-m)+(n-k)}+X^m+X^{(c-1)(n-m)+(n-k)}.\end{aligned}$$

Finally,

$$\begin{aligned}
h(010^{c-1}1) &= T_1S((T_0S)^{c-1}(T_1S(T_0S(1)))) \\
&= X^{n-(c-1)(n-m)-(n-k)} \cdot (T_0S)^{c-1}(T_1S(T_0S(1))) \bmod p_1 \\
&= X^{k-(c-1)(n-m)} \cdot (T_0S)^{c-1}(T_1S(T_0S(1))) \bmod p_1 \\
&= 1 + X^{k-(c-1)(n-m)} + \ldots + X^{k-2(n-m)} + X^{k-(n-m)} + \\
&\quad + X^m + X^{m+k-(c-1)(n-m)} + X^k \\
&= h(10^{c+1})
\end{aligned}$$

We calculate $h(010^{c-1}0)$ and $h(100^{c-1}1)$ using the results of previous calculations:

$$\begin{aligned}
h(010^{c-1}0) &= T_0S((T_0S)^{c-1}(T_1S(T_0S(1)))) \\
&= X^{k-(c-1)(n-m)} \cdot (T_0S)^{c-1}(T_1S(T_0S(1))) \bmod p_0 \\
&= 1 + X^{k-(c-1)(n-m)} + \ldots + X^{k-2(n-m)} + X^{k-(n-m)} + \\
&\quad + X^m + X^{m+k-(c-1)(n-m)} + X^m \\
&= 1 + X^{k-(c-1)(n-m)} + \ldots + X^{k-2(n-m)} + X^{k-(n-m)} + \\
&\quad + X^{m+k-(c-1)(n-m)}
\end{aligned}$$

$$\begin{aligned}
h(100^{c-1}1) &= T_1S((T_0S)^{c}(T_1S(1))) \\
&= X^{k-(c-1)(n-m)} \cdot (T_0S)^{c}(T_1S(1)) \bmod p_1 \\
&= 1 + X^{k-(c-1)(n-m)} + \ldots + X^{k-2(n-m)} + X^{k-(n-m)} + \\
&\quad + X^k + X^{m+k-(c-1)(n-m)} + X^k \\
&= 1 + X^{k-(c-1)(n-m)} + \ldots + X^{k-2(n-m)} + X^{k-(n-m)} + \\
&\quad + X^{m+k-(c-1)(n-m)}.
\end{aligned}$$

The expressions are indeed equal. □

To maximize c, we want m and k to be as large as possible. But even if we set $m = n - 2$ and $k = n - 1$, we have $c \approx n/2$, which is the birthday bound. These choices for m and k will rarely yield primitive trinomials, however, so in practice we end up with collision lengths that are much shorter.

We stress, though, that the exact length of the collisions is not the issue here, but rather that they are short and can be found efficiently.

4.2. **Polynomials with small monomials are weak.** The result in Section 4.1 parially extends to a special class of polynomials.

Proposition 4.2. Let $v_0(X) = 1$, and let

$$p_0(X) = 1 + X^{m_1} + \ldots + X^{m_r} + X^n,$$
$$p_1(X) = 1 + X^{k_1} + \ldots + X^{k_s} + X^n$$

be ordered. If $m_r + k_s < n$, we have the following collisions of length $\max\{r+2, s+2\}$:

$$h(01^r0) = h(10^s1);$$
$$h(01^r1) = h(10^s0).$$

The proof of this proposition, which is similar to that of Proposition 4.1, is included in Section 5. To avoid this weakness, we must require that $m_r + k_s \geq n$. This will be the case by default, with overwhelming probability, if the primitive polynomials are chosen at randoms.

It remains open to what extent the weaknesses exhibited here for degenerate feedback polynomials with initial value $v_0(X) = 1$ remain when using a random initial value.

4.3. **Discrete logarithm attacks.** Assume that a message m has hash value v, whose logarithm in base X, in the field $\mathbb{F}_{2^n} = \mathbb{F}_2[X]/\langle p_0(X)\rangle$ is small. In other words, there is a small number l with $X^l = v$ in the field. Computing this discrete logarithm l is feasible, since the characteristic of the field $\mathbb{F}_{2^n}$ is small. Using this, we can find a number $k \leq l$ with $h(0^k) = v$, thus mounting a second-preimage attack. A similar assertion holds for the other polynomial, $p_1(X)$.

However, the probability that a hash value of a long message has small index should be negligible. The above argument implies that if we are provided with $h(0^k)$ or $h(1^k)$ for *small* k, we can compute k and thus recover the input message. But this is true for every hash function, since in this scenario k can be found by exhaustive search, hashing all values $h(0)$, $h(1)$, $h(0^2)$, $h(1^2)$, ..., until the value v is attained. Thus, this is not a threat.

4.4. **Preimage attack at the birthday bound.** For each state v and each i, we can calculate $(T_iS)^{-1}(v)$ by XORing the feedback mask, shifting to the left, setting the rightmost bit to 1, and shifting to the left until the leftmost bit is 1. In this case, preimages of hash values can be found in complexity $O(2^{n/2})$.

Proposition 4.3. Let h be the abstract hash function defined by

$$h(b_0, \ldots, b_{\ell-1}) := f_{b_{\ell-1}} f_{b_{\ell-2}} \cdots f_{b_1} f_{b_0}(v_0),$$

with all functions $f_i \colon V \to V$ efficiently invertible. Given a hash value $h(b_0, \ldots, b_{\ell-1})$, we can compute a preimage $(c_0, \ldots, c_{m-1})$ in complexity $O(\sqrt{|V|})$.

Proof. We will find a minimal length string with the same hash value, using the meet-in-the-middle approach. Let

$$v := h(b_0, \ldots, b_{\ell-1}) = f_{b_{\ell-1}} f_{b_{\ell-2}} \cdots f_{b_1} f_{b_0}(v_0).$$

Initialize to sets $L := \{v\}$ and $R := \{v_0\}$. Repeat the following procedure until an element added to one set already belongs to the other set:

(1) For each $u \in R$ and each $i = 0, \ldots, k-1$, add $f_i(u)$ to the set R.
(2) For each $u \in L$ and each $i = 0, \ldots, k-1$, add $f_i^{-1}(u)$ to the set L.

An element u in the intersection can be expressed (by storing the computation details) as

$$f_{j_t}^{-1} \cdots f_{j_1}^{-1}(v) = u = f_{i_m} \cdots f_{i_1}(v_0),$$

and then we have

$$v = f_{j_1} \cdots f_{j_t} f_{i_m} \cdots f_{i_1}(v_0) = h(j_1, \ldots, j_t, i_m, \ldots, i_1),$$

as required. □

It should be possible to reduce the space complexity substantially [2], but not the time complexity. This cryptanalysis is a serious drawback of TS-Hash if $2^{n/2}$ is a feasible complexity. For $n = 256$ or greater, this should not be considered a threat.

Ideally, if a cryptographic hash function has output size about 2^n, preimages should not be possible to find in time less than the order of 2^n. One way to achieve this goal is to double the state space and output only half of it as the hash value. The price is doubling the running time of the hash function. There may be additional, more effficient ways, but this is beyond the scope of the present paper.

4.5. **Length extension attacks.** We briefly point out that, since the basic design of TS-Hash outputs the entire inner state, this hash is not secure against length-extension attacks. In implementations where length-extension attacks are a threat, we recommend the same solution as in the end of Section 4.4: Double the size of the inner state, and output just half of the bits. Here too, more efficient solutions may be possible.

4.6. **Side-channel attacks.** In implementation where the operations `FindFirstSet`, `VariableShift`, `MemoryLoadFromL1Cache`, and `XOR` have constant latency, and the array of feedback polynomials fit in the L1 Cache, the algorithm processes every bit (or block) of input in constant time, so the duration of the entire computation leaks only the message's length. Even in other scenarios, it is unclear how timing attacks can proceed. However, this hash family is still in its initial study, and immunization against side channel attacks is too early to consider at present. When time is ripe, we believe that adaptations of the standard counter-measures would be applicable for TS-Hash.

5. Proof of Proposition 4.2

We begin by computing the common prefixes to both collisions, $h(01^r) = (T_1S)^r(T_0S(1))$ and $h(10^s) = (T_0S)^s(T_1S(1))$. First, let us compute $h(01^r)$:

$$\begin{aligned} T_0S(1) &= 1 + X^{m_1} + \ldots + X^{m_r} \\ T_1S(T_0S(1)) &= 1 + X^{k_1} + \ldots + X^{k_s} + \\ &\quad + X^{n-m_r} + X^{n-m_r+m_1} + \ldots + X^{n-m_r+m_{r-1}}. \end{aligned}$$

We note that $n - m_r > k_s$ and $m_{r-1} > 0$, so $n - m_r + m_{r-1} > k_s$. Thus, in the next step we multiply by $X^{m_r - m_{r-1}}$:

$$\begin{aligned}(T_1S)^2(T_0S(1)) = 1 &+ X^{k_1} + \ldots + X^{k_s} + \\ &+ X^{m_r - m_{r-1}} + X^{k_1 + m_r - m_{r-1}} + \ldots + X^{k_s + m_r - m_{r-1}} + \\ &+ X^{n - m_{r-1}} + X^{n - m_{r-1} + m_1} + \ldots + X^{n - m_{r-1} + m_{r-2}}.\end{aligned}$$

It can be shown that $n - m_{r-1} + m_{r-2} > k_s + m_r - m_{r-1}$, so in the next step we multiply by $X^{m_{r-1} - m_{r-2}}$ (and then subtract the modulus). In general, in step i we multiply by $X^{m_{r-(i-2)} - m_{r-(i-1)}}$. Iterating this r times we get:

$$\begin{aligned}(T_1S)^r(T_0S(1)) = 1 &+ X^{k_1} + \ldots + X^{k_s} + \\ &+ X^{m_2 - m_1} + X^{k_1 + m_2 - m_1} + \ldots + X^{k_s + m_2 - m_1} + \\ &+ \ldots + \\ &+ X^{m_r - m_1} + X^{k_1 + m_r - m_1} + \ldots + X^{k_s + m_r - m_1} + \\ &+ X^{n - m_1}.\end{aligned}$$

Now, let us compute $h(10^s)$:

$$\begin{aligned}T_1S(1) &= 1 + X^{k_1} + \ldots + X^{k_s} \\ T_0S(T_1S(1)) &= 1 + X^{m_1} + \ldots + X^{m_r} + \\ &\quad X^{n - k_s} + X^{n - k_s + k_1} + \ldots + X^{n - k_s + k_{s-1}}.\end{aligned}$$

We note that $n - k_s > m_r$ and $k_{s-1} > 0$, so $n - k_s + k_{s-1} > m_r$. Thus, in the next step we multiply by $X^{k_s - k_{s-1}}$:

$$\begin{aligned}(T_0S)^2(T_1S(1)) = 1 &+ X^{m_1} + \ldots + X^{m_r} + \\ &+ X^{k_s - k_{s-1}} + X^{m_1 + k_s - k_{s-1}} + \ldots + X^{m_r + k_s - k_{s-1}} + \\ &+ X^{n - k_{s-1}} + X^{n - k_{s-1} + k_1} + \ldots + X^{n - k_{s-1} + k_{s-2}}.\end{aligned}$$

In can be shown that $n - k_{s-1} + k_{s-2} > m_r + k_s - k_{s-1}$, so in the next step we multiply by $X^{k_{s-1} - k_{s-2}}$ (and then subtract the modulus). In general, in step i we multiply by $X^{k_{s-(i-2)} - k_{s-(i-1)}}$. Iterating this s times we get:

$$\begin{aligned}(T_0S)^s(T_1S(1)) = 1 &+ X^{m_1} + \ldots + X^{m_r} + \\ &+ X^{k_2 - k_1} + X^{m_1 + k_2 - k_1} + \ldots + X^{m_r + k_2 - k_1} + \\ &+ \ldots + \\ &+ X^{k_s - k_1} + X^{m_1 + k_s - k_1} + \ldots + X^{m_r + k_s - k_1} + \\ &+ X^{n - k_1}.\end{aligned}$$

Now we can compute $h(01^r1)$ and $h(10^s0)$ and observe that they are indeed equal:

$$\begin{aligned}(T_1S)^{r+1}(T_0S(1)) = 1 &+ X^{k_1} + \ldots + X^{k_s} + \\ &+ X^{m_1} + X^{k_1+m_1} + \ldots + X^{k_s+m_1} + \\ &+ X^{m_2} + X^{k_1+m_2} + \ldots + X^{k_s+m_2} + \\ &+ \ldots + \\ &+ X^{m_r} + X^{k_1+m_r} + \ldots + X^{k_s+m_r} \\ (T_0S)^{s+1}(T_1S(1)) = 1 &+ X^{m_1} + \ldots + X^{m_r} + \\ &+ X^{k_1} + X^{m_1+k_1} + \ldots + X^{m_r+k_1} + \\ &+ X^{k_2} + X^{m_1+k_2} + \ldots + X^{m_r+k_2} + \\ &+ \ldots + \\ &+ X^{k_s} + X^{m_1+k_s} + \ldots + X^{m_r+k_s}.\end{aligned}$$

The computation for $h(01^r0)$ and $h(10^s1)$ is similar. This completes the proof.

6. Connection Between Reciprocal Trinomials

There is a strong connection between the behavior of T_pS with a primitive *trinomial* $p(X) = X^n + X^m + 1$ and the behavior of T_qS the *reciprocal trinomial* $q(X) = X^n + X^{n-m} + 1$ of p. To describe it, we will need to define a new term first.

Definition 6.1. Let $r(X)$ be a state polynomial (constant term equal to 1). Let us order it (to have $n > m_1 > m_2 > \ldots > m_{k-1} > m_k > 0$):

$$\begin{aligned}r(X) &= X^{m_1} + X^{m_2} + \ldots + X^{m_{k-1}} + X^{m_k} + 1 \\ &= X^{m_1} + X^{m_2} + \ldots + X^{m_{k-1}} + X^{m_k} + X^0\end{aligned}$$

Define its *reflection* as follows:

$$\begin{aligned}\mathrm{reflect}(r) &= X^{m_1-m_1} + X^{m_1-m_2} + \ldots + X^{m_1-m_{k-1}} + X^{m_1-m_k} + X^{m_1-0} \\ &= X^{m_1} + X^{m_1-m_k} + X^{m_1-m_{k-1}} + \ldots + X^{m_1-m_2} + 1\end{aligned}$$

The function reflect is an invertible involution on the state space. We also have $\deg(r) = \deg(\mathrm{reflect}(r))$ for all r.

Proposition 6.2. The following identity holds:

$$\mathrm{reflect} \circ T_qS \circ \mathrm{reflect} \circ T_pS = \mathrm{id}\,.$$

In other words, the reflection function conjugates T_qS to the inverse of T_pS.

Proof. Let $p(X), q(X), r(X)$ be as defined. We know that $\deg(r) = m_1$. We calculate:

$$\begin{aligned} T_pS(r) &= X^{n-m_1} \cdot (X^{m_1} + X^{m_2} + \ldots + X^{m_k} + 1) \bmod p \\ &= (X^n + X^{n-m_1+m_2} + \ldots + X^{n-m_1+m_k} + X^{n-m_1}) \bmod p \\ &= X^{n-m_1+m_2} + \ldots + X^{n-m_1+m_k} + X^{n-m_1} + X^m + 1. \end{aligned}$$

This expression is possibly not ordered because X^m may be out of place. Regardless, define $M = \deg(T_pS(r)) = \deg(\text{reflect}(T_pS(r)))$. Applying reflection, we get

$$\begin{aligned} &\text{reflect}(T_pS(r)) = \\ &\quad = X^{M-n+m_1-m_2} + \ldots + X^{M-n+m_1-m_k} + X^{M-n+m_1} + X^{M-m} + X^M. \end{aligned}$$

This expression still has degree M. Apply T_qS:

$$\begin{aligned} T_qS(\text{reflect}(T_pS(r))) = X^{n-M} \cdot (&X^{M-n+m_1-m_2} + \ldots + X^{M-n+m_1-m_k} + \\ &+ X^{M-n+m_1} + X^{M-m} + X^M) \bmod q. \end{aligned}$$

Simplifying, we have

$$\begin{aligned} &T_qS(\text{reflect}(T_pS(r))) = \\ &\quad = (X^{m_1-m_2} + \ldots + +X^{m_1-m_k} + X^{m_1} + X^{n-m} + X^n) \bmod q \\ &\quad = X^{m_1-m_2} + \ldots + X^{m_1-m_k} + X^{m_1} + 1 \\ &\quad = X^{m_1} + X^{m_1-m_k} + \ldots + X^{m_1-m_2} + 1. \end{aligned}$$

Finally, applying reflect again, we get

$$\text{reflect}(T_qS(\text{reflect}(T_pS(r)))) = X^{m_1} + X^{m_2} + \ldots + X^{m_k} + 1,$$

which is equal to r. □

A graph-theoretic interpretation of this property is that the graph of TS-Hash with two trinomial feedback polynomials and the graph of TS-Hash with the reciprocals of these trinomials are naturally isomorphic: the vertices are reflected and the edges are reversed.

7. Conclusions and open problems

This is the first study of a cryptographic hash function proposed, but hitherto unpublished officially, by the second named author. It may be viewed as a particularly simple and efficient instatiation of a general scheme motivated by noncommutative mathematical structures. Indeed, the code for this function is surprisingly short and clear.

Based on the well studied notion of primitive Galois LFSRs, this function is likely to have security guarantees that are not provable for other hash functions. We provided here one, very simple and initial, guarantee of this kind.

This hash function is easy to secure against theoretical threats, such as preimage search in time less than exhaustive search, as well as against more practical threats such as lengh extention attacks, as demonstrated above.

Our initial cryptanalysis indicates that excessive greediness regarding efficiency may come at the cost of security: Using only trinomials as feedback polynomials, or using polynomials with low coefficients only, is insecure—at least when the initial vector is $v_0(X) = 1$. The following problems remain open.

Problem 7.1. Let $v_0(X)$ be a uniformly random initial state, and $p_0(X)$, and $p_1(X)$ be trinomials. Is there, with high probability, a collision of subexponential length?

Problem 7.2. Let $v_0(X)$ be a uniformly random initial state, and $p_0(X)$, and $p_1(X)$ be polynomials with small monomials, in the sense of Proposition 4.2. Is there an efficiently computable collision of subexponential length?

Problem 7.3. Let $v_0(X) = 1$, $p_0(X) = 1 + X^m + X^n$ be a trinomial, and $p_1(X) = 1 + X^{k_1} + X^{k_2} + X^{k_3} + X^n$ be a polynomial with $m + k_3 \geq n$. Is there an efficiently computable collision of subexponential length?

Acknowlegments. We thank Professor Gilles Zémor for interesting discussions regarding matrix-based Cayley hash functions and their comparison to TS-Hash. We are indebted to the referees of an earlier submission of this paper for comments that helped us clarify a number of important issues.

References

[1] D. Charles, K. Lauter, and E. Goren. "Cryptographic Hash Functions from Expander Graphs". In: *Journal of Cryptology* 22.1 (Jan. 2009), pp. 93–113. ISSN: 1432-1378. DOI: 10.1007/s00145-007-9002-x. URL: https://doi.org/10.1007/s00145-007-9002-x.

[2] I. Dinur et al. "Dissection: A New Paradigm for Solving Bicomposite Search Problems". In: *Communications of the ACM* 57.10 (Sept. 2014), pp. 98–105. ISSN: 0001-0782. DOI: 10.1145/2661434. URL: http://doi.acm.org/10.1145/2661434.

[3] M. Grassl et al. "Cryptanalysis of the Tillich–Zémor Hash Function". In: *Journal of Cryptology* 24.1 (Mar. 2010), pp. 148–156. DOI: 10.1007/s00145-010-9063-0. URL: https://doi.org/10.1007/s00145-010-9063-0.

[4] C. Mullan and B. Tsaban. "SL2 homomorphic hash functions: Worst case to average case reduction and short collision search". In: *Designs, Codes and Cryptography* 81.1 (2016), pp. 83–107.

[5] C. Petit and J.-J. Quisquater. "Rubik's for Cryptographers". In: *Notices of the American Mathematical Society* 60.6 (2013), pp. 733–739.

[6] C. Petit et al. "Hard and easy components of collision search in the Zémor–Tillich Hash function: new attacks and reduced variants with equivalent security". In: *CT-RSA*. Ed. by M. Fischlin. Vol. 5473. Lecture Notes in Computer Science. Springer, 2009, pp. 182–194. ISBN: 978-3-642-00862-7. DOI: 10.1007/978-3-642-00862-7_12.

[7] V. Shpilrain and B. Sosnovski. "Compositions of linear functions and applications to hashing". In: *Groups Complexity Cryptology* 8.2 (Nov. 2016), pp. 155–161. DOI: 10.1515/gcc-2016-0016. URL: https://doi.org/10.1515/gcc-2016-0016.

[8] J. Tillich and G. Zémor. "Hashing with SL_2". In: *CRYPTO*. Ed. by Y. Desmedt. Vol. 839. Lecture Notes in Computer Science. Springer, 1994, pp. 40–49.
[9] J.-P. Tillich and G. Zémor. "Group-Theoretic Hash Functions". In: *Algebraic Coding*. Ed. by G. Cohen et al. Vol. 781. Lecture Notes in Computer Science. Springer, 1994, pp. 90–110.
[10] B. Tsaban. *TS-hash*. `http://u.cs.biu.ac.il/~tsaban/CyberIntro.html`. 2017.
[11] B. Tsaban and U. Vishne. "Efficient Linear Feedback Shift Registers with Maximal Period". In: *Finite Fields and Their Applications* 8 (Apr. 2003), pp. 256–267.
[12] Wikipedia. *Find first set*. `https://en.wikipedia.org/w/index.php?title=Find_first_set&oldid=813631699`. 2017.
[13] G. Zeng, K. He, and W. Han. "A trinomial type of σ-LFSR oriented toward software implementation". In: *Science in China Series F: Information Sciences* 50.3 (June 2007), pp. 359–372.

Department of Mathematics, Bar-Ilan University, Ramat Gan, Israel
Email address: `ibookstein@gmail.com`

Department of Mathematics, Bar-Ilan University, Ramat Gan, Israel
Email address: `tsaban@math.biu.ac.il`
URL: `https://u.cs.biu.ac.il/~tsaban/`