Preimages for the Tillich-Zémor hash function

Christophe Petit[1]*, Jean-Jacques Quisquater[1]

UCL Crypto Group**
Université catholique de Louvain
Place du levant 3
1348 Louvain-la-Neuve, Belgium
e-mails: christophe.petit@uclouvain.be,jjq@uclouvain.be

Abstract. After 15 years of unsuccessful cryptanalysis attempts by the research community, Grassl et al. have recently broken the collision resistance property of the Tillich-Zémor hash function. In this paper, we extend their cryptanalytic work and consider the preimage resistance of the function.
We present two algorithms for computing preimages, each algorithm having its own advantages in terms of speed and preimage lengths. We produce theoretical and experimental evidence that both our algorithms are very efficient and succeed with a very large probability on the function parameters. Furthermore, for an important subset of these parameters, we provide a full proof that our second algorithm always succeeds in deterministic cubic time.
Our attacks definitely break the Tillich-Zémor hash function and show that it is not even one-way. Nevertheless, we point out that other hash functions based on a similar design may still be secure.

1 Introduction

The Tillich-Zémor hash function was one of the oldest unbroken cryptographic hash functions in the literature. It was proposed at CRYPTO'94, following the cryptanalysis of a related scheme of Zémor [15,16,14,12]. It received significant cryptanalytic attention over the years [3,5,1,11,10] but although closely related schemes had been completely broken [2,13,9], it remained essentially intact during 15 years.

In August 2009, Grassl et al. introduced a new and very elegant algorithm finding collisions for the Tillich-Zémor hash function [6]. The authors discovered a particular structure in the hash values of *palindromic* messages (messages such that their bitstring representation can be reversed without changing) and exploited their finding with a nice result of Mesirov and Sweet [8] on the Euclidean algorithm applied to polynomials in characteristic 2.

In this paper, we extend the work of Grassl et al. to the problem of finding preimages for the Tillich-Zémor hash function. We first show that a tiny modification of their algorithm actually provides a second preimage algorithm. Inspired by previous work on a similar hash function [9], we then reduce the problem of finding preimages to any hash value to the problem of precomputing preimages to a few hash values with certain characteristics. Finally, we provide two algorithms for this precomputing part.

Both our precomputing algorithms are very efficient and successful for random choices of the function parameters. Each algorithm has its own advantages resulting from different approaches. The first algorithm produces shorter preimages than the second one and it is therefore more interesting from a practical point of view. On the other hand, the second algorithm is deterministic and it is faster than the first one. It is also more interesting from a theoretical point of

* Research Fellow of the Belgian Fund for Scientific Research (F.R.S.-FNRS) at Université catholique de Louvain (UCL).
** Supported by the Interuniversity Attraction Pole (IAP) projet BCrypt.

view since we have a proof that it always succeeds in deterministic cubic time for an important subset of the function parameters.

The remainder of this paper is organized as follows. In Section 2 we introduce our notations, the Tillich-Zémor hash function, the essential of Grassl et al.'s algorithm, and we briefly sketch out our algorithms. In Section 3, we modify Grassl et al.'s algorithm into a second preimage algorithm. In Section 4, we reduce the preimage problem to a precomputation part. In Sections 5 and 6 we give our two precomputation algorithms. We conclude the paper in Section 7 with a discussion of our results and the security of Tillich-Zémor-like hash functions. Finally, we illustrate our algorithms with a toy example in Appendix A.

2 Preliminaries

2.1 The Tillich-Zémor hash function

Let n be a positive integer and let $p(X)$ be an irreducible polynomial of degree n over the field $\mathbb{F}_2$. Let A_0 and A_1 be the following two matrices

$$A_0 := \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix} \text{ and } A_1 := \begin{pmatrix} X & X+1 \\ 1 & 1 \end{pmatrix}$$

that have determinant 1. We call these matrices the *generators* of the Tillich-Zémor hash function. Let $m = m_1 m_2 ... m_k \in \{0,1\}^*$ be the bitstring representation of a message. The Tillich-Zémor hash value of m is defined as

$$H(m_1 m_2 ... m_k) := A_{m_1} A_{m_2} ... A_{m_k} \bmod p(X).$$

2.2 Notations

Let $K := \mathbb{F}_2[X]/(P(X)) \approx \mathbb{F}_{2^n}$. The images of the Tillich-Zémor hash functions are the matrices of the group $SL(2,K)$, that is the group of matrices with elements in K and determinant 1. Let

$$h(m_1 ... m_k) := A_{m_1} A_{m_2} ... A_{m_k}$$

be the Tillich-Zémor hash function without modular reduction. Its images are elements of $SL(2, \mathbb{F}_2[X])$. In this paper, we sometimes identify the elements of K to their unique representatives of degree smaller than n in $\mathbb{F}_2[X]$. To remove any ambiguity when it may appear, we use the symbol $=$ to mean an equality over $\mathbb{F}_2[X]$ and $\equiv$ to mean an equality over K. For $q(X) \in \mathbb{F}_2[X]$, we write q_i for the coefficient of the term of degree i of $q(X)$. Finally, if $m, m' \in \{0,1\}^*$ are two bitstrings, we write mm' for their concatenation.

2.3 Grassl et al.'s collision algorithm

Grassl et al. [6] first observed that two messages collide for the Tillich-Zémor hash function if and only if they collide for the following modified function

$$H'(m_1 ... m_k) := A'_{m_1} A'_{m_2} ... A'_{m_k} \bmod p(X)$$

where

$$A'_0 := A_0 = \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix} \text{ and } A'_1 := A_0^{-1} A_1 A_0 = \begin{pmatrix} X+1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Grassl et al. then observed the following property of palindromic messages. Let h' be the modified Tillich-Zémor hash function without reduction.

Proposition 1 [6] *Let $m \in \{0,1\}^{2k}$ be a palindrome of even length, say $m = m_k...m_1m_1...m_k$. Let $a_{(0)}, ..., a_{(k)}$ be the following polynomials*

$$a_{(i)} = \begin{cases} 1, & \text{if } i = 0; \\ X + m_1 + 1, & \text{if } i = 1; \\ (X + m_i)a_{(i-1)} + a_{(i-2)}, & \text{if } 1 < i \leq k. \end{cases}$$

Then $h'(m) = \left(\begin{smallmatrix} a^2 & b \\ b & d^2 \end{smallmatrix}\right)$ for $a = a_{(k)}$, $d = a_{(k-1)}$ and some $b \in \mathbb{F}_2[X]$. Moreover, $h'(0m0) + h'(1m1) = \left(\begin{smallmatrix} a^2 & a^2 \\ a^2 & 0 \end{smallmatrix}\right)$.

From Proposition 1, we see that the square roots of the upper left entries of $h'(m_1m_1)$, $h'(m_2m_1m_1m_2)$, $h'(m_3m_2m_1m_1m_2m_3)$, etc. satisfy a Euclidean algorithm sequence (in reverse order) where each quotient is either X or $X + 1$. Those sequences are often called *maximal length sequences for the Euclidean algorithm* or *maximal length Euclidean sequences*, and they have long been a topic of interest in number theory. Mesirov and Sweet [8] showed that, when $a \in \mathbb{F}_2[X]$ is irreducible, there exist exactly two polynomials d such that a, d are the first terms of a maximal length Euclidean sequence. They also provide an algorithm to compute them, which we will give below.

In their collision algorithm, Grassl et al. apply Mesirov and Sweet's algorithm to the irreducible polynomial $a = p(X)$ in order to recover d. The corresponding bit sequence $m_1...m_n$ can be recovered by applying the Euclidean algorithm to a and d. By Proposition 1, we have

$$h'(0m_n...m_1m_1...m_n0) = h'(1m_n...m_1m_1...m_n1) + \left(\begin{smallmatrix} a^2 & a^2 \\ a^2 & 0 \end{smallmatrix}\right)$$

hence

$$H'(0m_n...m_1m_1...m_n0) \equiv H'(1m_n...m_1m_1...m_n1).$$

2.4 Maximal length sequences in the Euclidean algorithm in $\mathbb{F}_2[X]$

The Mesirov and Sweet's algorithm, as described by Grassl et al., is the following one. To find a maximal length Euclidean sequence starting from a given polynomial $a(X)$ of degree k,

1. Construct a matrix $A \in \mathbb{F}_2^{(k+1)\times k}$ from the $k + 1$ polynomials

   $$g_0 = 1,$$
   $$g_i = X^{i-1} + X^{2i-1} + X^{2i} \bmod a(X), \qquad \text{for } i = 1, 2, ..., k,$$

   placing in the i^{th} row of A the coefficients $g_{i,0}, g_{i,1}, ..., g_{i,k-1}$ of the polynomial $g_i(X) = g_{i,0} + g_{i,1}X + ... + g_{i,k-1}X^{k-1}$.
2. Solve the linear system $Au^t = (1, 0, ..., 0, 1)^t$ where $u = (u_1, ..., u_k)$.
3. Compute $d(X)$ by multiplying $a(X)$ by $\sum_{i=1}^{k} u_i X^{-i}$ and taking only the non-negative powers.

Mesirov and Sweet showed in [8] that polynomials d such that a, d are the first terms of a maximal length sequence for the Euclidean algorithm, are in one-to-one correspondence with the solutions of the equation $Au^t = (1, 0, ..., 0, 1)^t$. Moreover, they proved that when a is irreducible, this equation has exactly two solutions.

Maximal length Euclidean sequences are closely connected to the matrices A'_0 and A'_1. If m_i and $a_{(i)}$ are as in Proposition 1, we have

$$\left(\begin{smallmatrix} a_{(1)} & a_{(0)} \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right) \left(\begin{smallmatrix} X+1+m_1 & 1 \\ 1 & 0 \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right) A'_{m_1}$$

(where $\overline{m_1} := 1 - m_1$) and for $1 < i \leq k$ we have

$$\begin{pmatrix} a_{(i)} & a_{(i-1)} \end{pmatrix} = \begin{pmatrix} a_{(i-1)} & a_{(i-2)} \end{pmatrix} \begin{pmatrix} X+m_i & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} a_{(i-1)} & a_{(i-2)} \end{pmatrix} A'_{m_i}.$$

Therefore, the first row of any product of A'_0 and A'_1 is the beginning of a maximal length Euclidean sequence. By induction, we have

$$\begin{pmatrix} a_{(k)} & a_{(k-1)} \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix} h'(\overline{m_1} m_2 ... m_k). \quad (1)$$

2.5 Ideas behind our algorithms

Before going into the details of our algorithms, we provide some intuition behind them. Let m be the palindromic message used in Grassl et al.'s collision attack. In Section 3, we first observe that the hash values of $m0$ and $0m$ have the form $L := \begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix}$ and $U := \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix}$ for some $\alpha \in \mathbb{F}_{2^n}$. Since L and U have order 2, we obtain an algorithm finding preimages to the identity matrix, hence a second preimage algorithm for the Tillich-Zémor hash function.

We then observe that the set of matrices $\mathcal{L} := \{L_\alpha := \begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix}, \alpha \in K\}$ forms an Abelian subgroup of $SL(2, K)$ that is isomorphic to the additive group $(K, +)$. Finding n matrices L_{α_i} (together with their preimages) such that the set $\{\alpha_i, i = 1, ..., n\}$ is a basis of K over $\mathbb{F}_2$, therefore suffices to generate the whole subgroup $\mathcal{L}$. The same holds for the set $\mathcal{U} := \{U_\alpha := \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix}, \alpha \in K\}$. Moreover, inspired by previous work on a similar function [9], we prove in Proposition 3 that any matrix of $SL(2, K)$ can be written as a small product of A_0 and matrices of the sets $\mathcal{L}$ and $\mathcal{U}$. At this point, it remains to obtain n matrices L_{α_i} and U_{α_i} generating $\mathcal{L}$ and $\mathcal{U}$. We solve this problem in two different ways and obtain two algorithms, each of them having its own advantages but both of them being very efficient and successful.

As observed above, Grassl et al.'s paper indirectly provides one matrix $L \in \mathcal{L}$ after applying Mesirov and Sweet's algorithm to $a = p$. In our first precomputing algorithm, we obtain n different matrices after applying Mesirov and Sweet's algorithm to $a_i = pp'_i$, where p'_i are randomly-chosen small degree polynomials. This idea is quite simple but proving its correctness requires solving two issues. First, Mesirov and Sweet only guarantee the success of their algorithm when applied to an irreducible polynomial. Second, the α_i values obtained by this way should not be restricted to any vectorial subspace of K. In this paper, we extend Mesirov and Sweet's result in Proposition 5 and then argue on the correctness of our algorithm.

In our second precomputing algorithm, we follow a different approach and obtain n matrices recursively from the first one. In particular, we exhibit a sequence of messages with increasing lengths hashing to matrices of the required form, such that the corresponding values α_i satisfy a very simple recurrence. By studying the elements of this recurrence, we identify a subset $I \subset \{1, ..., 2n\}$ such that the corresponding matrices $\{L_{\alpha_i}, i \in I\}$ generate the whole subgroup when n is prime. This important result is proved through Lemma 6, Lemma 7 and Proposition 9. Matrices $\{U_{\alpha_i}, i \in I\}$ and their preimages are recovered at the same time.

Due to the way it constructs matrices L_{α_i} and U_{α_i}, our second precomputing algorithm produces larger preimages than the first one. On the other hand, it is deterministic, faster than the first one, and it is guaranteed to always succeed when parameter n is prime.

3 Second preimages for Tillich-Zémor hash function

The following proposition constructs collisions with the void message from the palindromic messages used in Grassl et al.'s attack.

Proposition 2 *Let* $\begin{pmatrix} a^2 & b \\ b & d^2 \end{pmatrix} = H'(m)$ *with* $a \equiv 0$ *be the modified Tillich-Zémor hash value of some message* $m \in \{0, 1\}^*$. *Then*

$$H(0m0m) = H(1m1m) = H(m0m0) = H(m1m1) = I = H().$$

PROOF: We have $1 = \det(h'(m)) = a^2d^2 + b^2 \equiv b^2$ hence $b \equiv 1$. By a straightforward computation, we have $H'(0m) = \left(\begin{smallmatrix} 1 & X+d^2 \\ 0 & 1 \end{smallmatrix}\right)$, $H'(m0) = \left(\begin{smallmatrix} 1 & 0 \\ X+d^2 & 1 \end{smallmatrix}\right)$, $H'(1m) = \left(\begin{smallmatrix} 1 & X+1+d^2 \\ 0 & 1 \end{smallmatrix}\right)$, $H'(m1) = \left(\begin{smallmatrix} 1 & 0 \\ X+1+d^2 & 1 \end{smallmatrix}\right)$, and all these matrices have order 2. Finally, we observe that for any $\tilde{m} \in \{0,1\}^*$ such that $H'(\tilde{m}) = I$, we have $H(\tilde{m}) = A_0 H'(\tilde{m}) A_0^{-1} = A_0 A_0^{-1} = I$. □

The message m in Proposition 2 can be obtained by applying Mesirov and Sweet's algorithm to $a = p(X)$ as in Grassl et al.'s attack (see Proposition 1). We therefore obtain a message $\tilde{m}$ colliding with the void message for the Tillich-Zémor hash function. A second preimage algorithm is straightforwardly deduced, since for any $m \in \{0,1\}^*$ we have $H(m\tilde{m}) = H(m)$.

4 Preimage algorithm from a few precomputed preimages

For the remaining of the paper, we define $L_\alpha := \left(\begin{smallmatrix} 1 & 0 \\ \alpha & 1 \end{smallmatrix}\right)$ and $U_\beta := \left(\begin{smallmatrix} 1 & \beta \\ 0 & 1 \end{smallmatrix}\right)$ for any $\alpha, \beta \in K$. In [9], preimages for the LPS hash function (a function similar to Tillich-Zémor, with different matrix generators) were computed by decomposing any matrix into a product of generators and diagonal matrices, a subset of matrices for which computing preimages appeared to be easier. In the case of Tillich-Zémor, the proof of Proposition 2 suggests the following decomposition.

Proposition 3 *Given a preimage of length at most L for every matrix among a set $\mathcal{S} = \{L_{\alpha_i}, U_{\beta_i}, i = 1, ..., n\}$ where $\{\alpha_i, i = 1, ..., n\}$ and $\{\beta_i, i = 1, ..., n\}$ are two basis of K as a vector space over $\mathbb{F}_2$, there is a deterministic algorithm computing preimages of length at most $3nL + 5$ for the Tillich-Zémor hash function, in time $O(n^3)$.*

PROOF: We first observe that it is sufficient to have an algorithm with the same characteristics for the modified Tillich-Zémor hash function. Indeed, for any A, B, C, D with $AD + BC = 1$,

$$H(m) = \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right) \Leftrightarrow H'(m) = A_0^{-1} \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right) A_0.$$

Now, suppose we are given $M = \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right)$ with $\det(M) = 1$ and we want to find a preimage of M for the modified Tillich-Zémor function. If $B \neq 0$, it is easily checked that

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix} \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & \beta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix}^3 \begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix}$$

with

$$\begin{cases} \alpha = (DX + X + B)/(XB) \\ \beta = (B + X^3)/X^2 \\ \gamma = (X + B + X^2B + AX)/(XB) \end{cases}$$

while if $B = 0$, we have

$$\begin{pmatrix} A & 0 \\ C & D \end{pmatrix} = \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} C & D \\ A + CX & DX \end{pmatrix}$$

and $D \neq 0$, so we may apply the above decomposition to the last matrix.

Since $\{\alpha_i, i = 1, ..., n\}$ and $\{\beta_i, i = 1, ..., n\}$ are two basis of K, we may write $\alpha = \sum_{i \in I_\alpha} \alpha_i$, $\beta = \sum_{i \in I_\beta} \beta_i$ and $\gamma = \sum_{i \in I_\gamma} \alpha_i$, for some $I_\alpha, I_\beta, I_\gamma \subset \{1, ..., n\}$. Moreover, those decompositions can be recovered in time $O(n^3)$ by solving three corresponding linear systems over $\mathbb{F}_2$. Finally, we observe that for any $I \subset \{1, ..., n\}$, we have

$$\begin{pmatrix} 1 & 0 \\ \sum_{i \in I} \alpha_i & 1 \end{pmatrix} = \prod_{i \in I} \begin{pmatrix} 1 & 0 \\ \alpha_i & 1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & \sum_{i \in I} \beta_i \\ 0 & 1 \end{pmatrix} = \prod_{i \in I} \begin{pmatrix} 1 & \beta_i \\ 0 & 1 \end{pmatrix}.$$

Putting together what we have seen so far, we obtain a decomposition of any matrix into at most 5 matrices A_0' and $3n$ matrices from the set $\mathcal{S}$. A preimage is obtained by concatenating

the preimages of the corresponding matrices, and the maximal length of this preimage follows. □

Let $\mu_L, \mu_U : \{0,1\}^* \to \{0,1\}^*$ be two transformations on bitstrings defined as follows:

$$\mu_L(m_1 m_2 ... m_k) = m_k ... m_2 \overline{m_1 m_1} m_2 ... m_k 0;$$
$$\mu_U(m_1 m_2 ... m_k) = 0 m_k ... m_2 \overline{m_1 m_1} m_2 ... m_k.$$

Lemma 4 *Let $m \in \{0,1\}^*$ such that $(1\ 0)\, H'(m) = (0\ q)$ for some $q \in K$. Then*

$$H'(\mu_L(m)) = \begin{pmatrix} 1 & 0 \\ X+q^2 & 1 \end{pmatrix} \quad \text{and} \quad H'(\mu_U(m)) = \begin{pmatrix} 1 & X+q^2 \\ 0 & 1 \end{pmatrix}.$$

PROOF: Let $m = m_1 ... m_k$ be the bitwise representation of m. According to Equation 1 and Proposition 1 we have $H'(m_k ... m_2 \bar{m_1} \bar{m_1} m_2 ... m_k) = \begin{pmatrix} 0 & b \\ b & q^2 \end{pmatrix}$. Moreover $b = 1$ since the determinant of any hash value is 1. Multiplying left and right by A_0 we obtain the result. □

Proposition 3 reduces the preimage problem to the problem of precomputing the preimages of some set of matrices. Lemma 4 further reduces the precomputation to find n messages $m_i, i = 1, ..., n$ such that

$$(1\ 0)\, H'(m_i) = (0\ q_i) \text{ for some } q_i \in K \tag{2}$$

and $\{q_i^2 + X \bmod p, i = 1, ..., n\}$ is a basis of K. In Sections 5 and 6, we give two algorithms for finding these messages.

5 First precomputing algorithm

As observed in Section 3, we can obtain one message satisfying Equation 2 by applying Mesirov and Sweet's algorithm to $a = p$. In order to obtain more messages satisfying the equation, a natural idea is to apply the algorithm to a small multiple of p. This leads us to the following algorithm.

1. Take R large enough.
2. Construct a set $T = \{\alpha_i, i = 1, ..., n\}$ containing elements of K that are linearly independent over $\mathbb{F}_2$, as well as preimages to L_{α_i} and U_{α_i}. To this aim, start from an empty set T, then until the set contains n elements:
   (a) Generate a random irreducible polynomial p' of degree R.
   (b) Construct a matrix A by applying the first step of Mesirov and Sweet's algorithm to $a = pp'$.
   (c) If $Au^t = (1, 0, ..., 0, 1)^t$ has solutions, compute $\alpha := d^2 + X$ where d is obtained by completing Mesirov and Sweet's algorithm.
   (d) Check whether α is independent of the elements of T; if it is, add it to the list and compute the corresponding preimages.

The algorithm is conceptually simple but it is not a trivial task to prove its correctness. First, Mesirov and Sweet only guarantee the success of their algorithm for irreducible polynomials while in Step 2(b) we apply it to a more general polynomial. Second, the above algorithm succeeds only if it is possible to generate n independent α_i values in Step 2(d). This last condition seems particularly hard to prove given our current understanding of maximal length Euclidean sequences in $\mathbb{F}_{2^n}$.

In Section 5.1 below, we extend Mesirov and Sweet's result and argue that in Step 2(c) of our algorithm, the system $Au^t = (1, 0, ..., 0, 1)^t$ has solutions with a probability ρ at least 1/2 on average. In Section 5.2, we provide intuitive arguments and experimental evidence showing that if R is $O(\log n)$, the loop of Step 2 must only be repeated $O(n)$ times. Since each loop

requires solving at most two linear systems of size n, we expect the whole algorithm to run in probabilistic $O(n^4)$ time.

All the messages constructed by this algorithm have length exactly $R+n$. These messages can be used in Lemma 4 and Proposition 3 to compute preimages of length $O(n^2+nR) \approx O(n^2)$ for any matrix.

5.1 Mesirov and Sweet's algorithm for $a = pp'$

In this section, we argue that for a parameter p chosen at random, the probability that $Au^t = (1,0,...,0,1)^t$ has solutions in Step 2(c) of our algorithm is $\rho \approx 1/2$. Let us consider the arithmetic sequence

$$1 + p + \ell X(X+1)p, \qquad \ell \in \mathbb{F}_2[X], \qquad \deg(\ell) \leq R-1.$$

For any $\ell \in \mathbb{F}_2[X]$, let $N(\ell)$ be the number of distinct irreducible polynomials of degree R in the factorization of $1+p+\ell X(X+1)p$. Let $N_0 := \sum_{\deg(\ell)\leq R-2} N(\ell)$ and $N_1 := \sum_{\deg(\ell)=R-1} N(\ell)$. Although a priori there may exist some R and p such that $N_0 \approx 0$, for most values R and random polynomials p it seems reasonable to expect $N_0 \approx N_1$. In the following, we show that the probability that $Au^t = (1,0,...,0,1)^t$ has solutions is at least $\frac{N_0}{N_0+N_1}$, hence for a parameter p chosen at random we expect to have ρ at least equal to $1/2$.

By simple linear algebra the system $Au^t = (1,0,...,0,1)^t$ has solutions if both 1) the first row of A is a linear combination of its last row and some other rows and 2) the last row of A is not a linear combination of its middle rows. In other terms,

1. There exists $v = (v_0,...,v_{n+R}) \in \mathbb{F}_2^{(n+R+1)\times 1}$ such that $vA = 0$ and $v_0 = v_{n+R} = 1$.
2. For any $w = (w_0,...,w_{n+R}) \in \mathbb{F}_2^{(n+R+1)\times 1}$ such that $wA = 0$ and $w_0 = 0$, we have $w_{n+R} = 0$.

Following Mesirov and Sweet [8], let us consider the following equations in the (polynomial) variables $r(X)$ and $s(X)$:

$$r(X) + Xr(X)^2 + X^2 r(X)^2 = 1 \bmod a(X), \tag{3}$$
$$s(X) + Xs(X)^2 + X^2 s(X)^2 = 0 \bmod a(X). \tag{4}$$

Remembering the definition of A, we see that $Au^t = (1,0,...,0,1)^t$ has solutions if

1. For some $r(X) = r_{n+R-1}X^{n+R-1} + ... + r_1 X + r_0$ solution to Equation 3, we have $r_{n+R-1} = 1$.
2. For any $s(X) = s_{n+R-1}X^{n+R-1} + ... + s_1 X + s_0$ solution to Equation 4, we have $s_{n+R-1} = 0$.

Since X does not divide $a = pp'$, it must divide $a+1$. The polynomial $(a+1)/X$ has degree $n+R-1$ and is a solution to Equation 3; it therefore satisfies the first condition. Equation 4 has four solutions s_{00}, s_{01}, s_{10} and s_{11} characterized as follows:

$$\begin{cases} s_{00} = 0 \bmod p, \\ s_{00} = 0 \bmod p'; \end{cases} \qquad \begin{cases} s_{10} = 0 \bmod p, \\ 1 + X(X+1)s_{10} = 0 \bmod p'; \end{cases}$$

$$\begin{cases} 1 + X(X+1)s_{01} = 0 \bmod p, \\ s_{01} = 0 \bmod p'; \end{cases} \qquad \begin{cases} 1 + X(X+1)s_{11} = 0 \bmod p, \\ 1 + X(X+1)s_{11} = 0 \bmod p'. \end{cases}$$

Clearly, $s_{00} = 0 \bmod a$. Since $X+1$ does not divide $a = pp'$, it must divide $a+1$. The polynomial $(a+1)/X(X+1)$ has degree $n+R-2$ and is a solution to Equation 4. Reducing it modulo p and p', we see that $(a+1)/X(X+1) = s_{11}$. As Equation 4 is homogeneous, its solutions form a vector space, hence $s_{01} = s_{10} + s_{11}$. Since the coefficient $n+R-1$ of s_{11} is zero, the coefficient $N+R-1$ of s_{01} and s_{10} are equal. Using Chinese Remainder Theorem, we have

$$s_{10} = \left[(X(X+1)p')^{-1} \bmod p \right] p'$$

hence the coefficient $N+R-1$ of s_{10} is equal to the coefficient $N-1$ of $(X(X+1)p')^{-1} \bmod p$. We have proved the following proposition that extends Mesirov and Sweet's result on irreducible polynomials to polynomials with two distinct nonlinear irreducible factors:

Proposition 5 *Let p, p' be nonlinear irreducible polynomials and let $a = pp'$. If*

$$\deg\left([X(X+1)p']^{-1} \bmod p\right) \leq \deg(p) - 2.$$

then the corresponding Mesirov and Sweet system $Au^t = (1, 0, ..., 0, 1)^t$ has solutions.

Let $y := [X(X+1)p']^{-1} \bmod p$. By definition, we have

$$yX(X+1)p' = 1 + kp$$

for some unique $k \in \mathbb{F}_2[X]$. As $\deg(y) \leq N-1$, we have $\deg(k) \leq R+1$. We easily see that $X(X+1)$ divides $1+kp$ if and only if $k = 1+X(X+1)\ell$ for some $\ell \in \mathbb{F}_2[X]$ with $\deg(\ell) \leq R-1$. Therefore for any p' (randomly generated in our algorithm), there exists exactly one polynomial ℓ with $\deg(\ell) \leq R-1$ such that

$$yX(X+1)p' = 1 + p + X(X+1)\ell p.$$

If $\ell \neq 0$, then by Proposition 5, the system $Au^t = (1, 0, ..., 0, 1)^t$ has solutions if $\deg(\ell) = \deg(y) + R - \deg(p) \leq R-2$. For $\ell = 0$ we have $\deg(y) = \deg(p) - 2 - R$ which satisfies the condition of Proposition 5. Defining N_0 and N_1 as above, N_0 (respectively N_1) is precisely the number of irreducible polynomials p' of degree R such that the corresponding polynomial ℓ satisfies $\deg(\ell) \leq R-2$ (respectively $\deg(\ell) = R-1$). We finally obtain $\rho \geq \frac{N_0}{N_0+N_1}$.

Remark. It is possible to remove the irreducibility condition on p' in Step 2(a) of our algorithm. However, the probability that Mesirov and Sweet system $Au^t = (1, 0, ..., 0, 1)^t$ has solutions in Step 2(c) is maximal when p' is irreducible, as can be seen by further extending Proposition 5.

5.2 Correctness of the algorithm

Even after finding many different α values in Step 2(c), our algorithm could still fail if all these values belonged to a vector subspace of K. However, the algorithm will always succeed if the following hypothesis holds.

Hypothesis 1 *The polynomials d resulting from applying the Mesirov and Sweet's algorithm to $a = pp'$ (with p and p' irreducible, p fixed and p' random), form a set without any particular structure. In particular, this set can be thought of as a random set of polynomials of degree $n+R-1$.*

If each d value constructed can be considered as a random polynomial of degree $n+R-1$, then the corresponding value $\alpha := d^2 + X \bmod p$ can be considered as a random polynomial of degree at most $n-1$. In that case, every new value α is (very) likely to be independent from the previous one: if there are already i independent elements in T, the new value will be independent from the previous ones with a probability $1 - 2^{i-n}$. After generating a little more than n polynomials d, our algorithm will be likely to succeed in finding n linearly independent α values. By the analogue of the prime number theorem for irreducible polynomials over a finite field, the number of irreducible polynomials p' of degree R is roughly $2^R/R$. By our analysis in Section 5.1, the Mesirov and Sweet's system has solutions for at least one half of the corresponding polynomials $a = pp'$. Therefore, we must have

$$\frac{2^R}{R} \geq 2n. \tag{5}$$

Assuming Hypothesis 1 is correct, taking a value $O(\log n)$ for R in Step 1 should be sufficient to guarantee the success of our algorithm.

Hypothesis 1 seems very likely to hold. Of course, there is a strong relationship between two polynomials a and d such that all their partial quotients are X or $X+1$. However, this relation does not seem to restrict d to any vector subspace of K, even if a is only chosen among the multiples of p. Today, the "simplest" relation known to hold between two consecutive terms of a maximal length Euclidean sequence is precisely given by the Mesirov and Sweet's algorithm, and it does not seem to impose such a restriction.

A theoretical analysis of our algorithm based on firmer grounds than Hypothesis 1 would be very valuable, but given current understanding of maximal length Euclidean sequences in $\mathbb{F}_{2^n}$, we believe that it is far from reach. To prove the efficiency of our algorithm, we therefore complete our arguments with experimental results. In our experiments, we took p the shortest irreducible polynomial of degree n, that is the polynomial $p = X^n + ... + p_0$ for which $\sum p_i 2^i$ is minimal. For some values of n between 5 and 2039, we tried different values of R until we found one that was large enough. The results are given in Figure 1.

n	R	n	R	n	R
5	5	67	11	257	13
7	7	73	11	277	13
11	8	83	11	307	13
13	9	97	11	331	14
17	9	103	12	353	14
19	9	109	12	379	14
23	9	127	12	401	14
29	9	137	12	449	14
31	10	149	12	499	14
37	10	167	13	607	15
41	10	191	13	1021	15
47	10	211	13	2039	16
59	11	233	13		

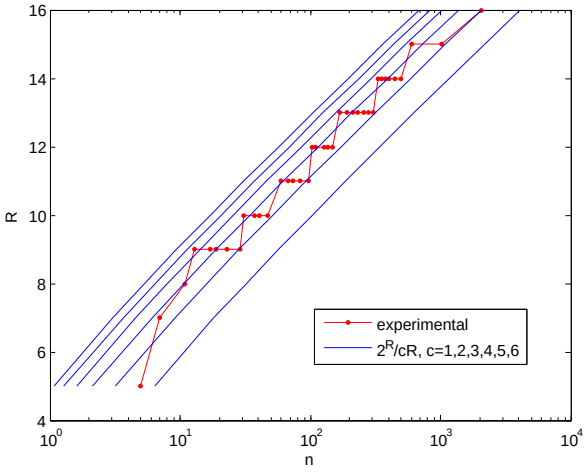

Fig. 1. Minimal value R for different values n and the "shortest" polynomials of degree n. The points on the staircase-like curve are experimental results. The other curves are $n = \frac{2^R}{cR}$ for $c = 1, 2, 3, 4, 5, 6$.

The algorithm always succeeded with a value R satisfying

$$\frac{2^R}{R} \geq 5n,$$

that is slightly larger than predicted by Equation 5 but still consistent with the expected $R = O(\log n)$. When $n \geq 17$, it always succeeded as long as $\frac{2^R}{R} \geq 4n$. The last points of Figure 1 for $n = 1021$ and $n = 2039$ are very close to the bound of Equation 5. The results confirm the analysis performed in this section and in the previous one. In Appendix B, we describe additional experiments performed on randomly chosen polynomials. All the results obtained are also consistent with our analysis.

6 Second precomputing algorithm

Our first precomputing algorithm is very efficient both in theory and in practice, but its correctness must likely rely on some *ad hoc* hypothesis. In this section, we provide an alternative algorithm precomputing messages of length bounded by n^2, resulting in preimages of length $O(n^3)$ after applying Lemma 4 and Proposition 3. On the one hand this second algorithm is worse than the first one since it produces larger preimages, but on the other hand it is deterministic and it runs in time $O(n^3)$, much faster than the first one. More importantly from a theoretical point of view, we provide a proof that it always succeeds when n is prime, and strong evidence that it has a very large probability of success when n is reasonably large and the polynomial p is chosen at random.

The Mesirov and Sweet's algorithm applied to p is guaranteed to succeed since p is irreducible. It provides a polynomial q of degree $n-1$ such that p and q are the first terms of a maximal length Euclidean sequence. By Equation 1, we have $\left(\begin{smallmatrix} p & q \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right) h'(\tilde{m}_1)$ for some $\tilde{m}_1 \in \{0,1\}^n$ that can be recovered with the Euclidean algorithm. For $i > 1$, let

$$\tilde{m}_i := \tilde{m}_{i-1} 0 \tilde{m}_1. \tag{6}$$

We first show that the messages $\tilde{m}_i$ satisfy the requirements of Lemma 4.

Lemma 6 *For any $i \geq 0$, $\left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right) H'(\tilde{m}_i) \equiv \left(\begin{smallmatrix} 0 & q^i \end{smallmatrix}\right)$.*

Proof: For $i = 1$ the result is trivial. Moreover, assuming the property is satisfied for some i, it is also satisfied for $i+1$ since

$$\begin{aligned} \left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right) H'(\tilde{m}_{i+1}) &= \left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right) H'(\tilde{m}_i) A'_0 H'(\tilde{m}_1) \\ &\equiv \left(\begin{smallmatrix} 0 & q^i \end{smallmatrix}\right) \left(\begin{smallmatrix} X & 1 \\ 1 & 0 \end{smallmatrix}\right) \left(\begin{smallmatrix} 0 & q \\ \dots & \dots \end{smallmatrix}\right) \equiv \left(\begin{smallmatrix} q^i & 0 \end{smallmatrix}\right) \left(\begin{smallmatrix} 0 & q \\ \dots & \dots \end{smallmatrix}\right) \equiv \left(\begin{smallmatrix} 0 & q^{i+1} \end{smallmatrix}\right). \quad \square \end{aligned}$$

To apply Lemma 4 and Proposition 3, we must find a subset of $\{q^{2i} + X \bmod p, i \geq 1\}$ that is a basis of K. We start by studying the sets $S_j := \{q^{2(j+i)} \bmod p, i = 1, ..., n\}$.

Lemma 7 *If n is prime, then for any $j \geq 0$, the set $S_j := \{q^{2(j+i)} \bmod p, i = 1, ..., n\}$ is a basis of K over $\mathbb{F}_2$.*

Proof: It suffices to show that each S_j is a free set of K. Let us assume by contradiction that there exists $(b_1, ..., b_n) \in \mathbb{F}_2^n \setminus \{(0, ..., 0)\}$ such that $0 = \sum_{i=1}^{n} b_i q^{2(j+i)}$. Since q is a polynomial of degree $n-1$, $q \neq 0, 1$ hence $q^2 \neq 0, 1$. The degree of the minimal polynomial of q^2 must divide n, hence if n is prime it is either 1 or n. Since $q^2 \neq 0, 1$ the minimal polynomial degree has degree larger than 1 and hence it has degree n. Now, let us define the polynomial $f : y \to f(y) = \sum_{i=0}^{n-1} b_{i-1} y^i$. We have $\deg(f) < n$. We also have $q^{2(j+1)} f(q^2) = 0$ hence $f(q^2) = 0$. Therefore f must be a multiple of the minimal polynomial of q^2, which brings us to a contradiction. $\square$

The proof of Lemma 7 does not work if n is not prime since the degree of the minimal polynomial of q^2 may then be a nontrivial divisor of n. However, even if n is not prime, S_j is a basis for all j if and only if S_j is a basis for some j. If n_1 is a divisor of n, the probability that a random element of K has a minimal polynomial with degree dividing n_1 is $2^{n_1 - n}$, that is very small for reasonably large values of n (Tillich and Zémor suggested $n > 130$). Of course, q^2 is not a random element of K since q is one of the two polynomials making p and q the first terms of a maximal length Euclidean sequence. However, the best link known today between p and q is the Mesirov and Sweet's algorithm, and when p is chosen at random this algorithm does not seem to influence the degree of the minimal polynomial of q. We have confirmed this intuition

experimentally: for large n and random p, S_0 was always a basis, while for very small values of n like $n = 4$ it was not always (although most often) the case.[1]

Let $T_j := \{q^{2(i+j)} + X \bmod p, i \geq 1\}$. We now show that if S_0 is a basis over K, then there exists $j \leq n$ such that T_j is a basis of K. To this aim we first need the following lemma.

Lemma 8 *Let $\{\beta_i, i = 1, ..., n\}$ be a basis of K over $\mathbb{F}_2$ and let $e_i \in \mathbb{F}_2, i = 1, ..., n$ be such that $X = \sum_i e_i\beta_i$. Then $\{\alpha_i := \beta_i + X, i = 1, ..., n\}$ is a basis of K over $\mathbb{F}_2$ if and only if $\sum_i e_i = 0$.*

PROOF: We can write $X = \sum_i e_i\alpha_i + (\sum_i e_i)X$. If $\sum_i e_i = 1$, then $0 = \sum_i e_i\alpha_i$ hence $\{\alpha_i, i = 1, ..., n\}$ is not a free set of elements. On the other hand, if $\sum_i e_i = 0$ then $X = \sum_i e_i\alpha_i$. Let $\alpha \in K$ and let $b_i \in \mathbb{F}_2, i = 1, ..., n$ be such that $\alpha = \sum_i b_i\beta_i$. Let $a_i := b_i + (\sum_i b_i)\, e_i$. We have $\alpha = \sum_i b_i\alpha_i + (\sum_i b_i)X = \sum_i a_i\alpha_i$ hence $\{\alpha_i, i = 1, ..., n\}$ is a generating set. □

We are now ready to prove the following result.

Proposition 9 *If S_0 is a basis of K over $\mathbb{F}_2$ (in particular, if n is prime), then there exists $j < n$ such that T_j is a basis of K over $\mathbb{F}_2$.*

PROOF: We give a constructive proof of the result. Since S_0 is a basis of K over $\mathbb{F}_2$, we know that S_j is a basis of K over $\mathbb{F}_2$ for all $j \geq 0$. Let $e_i \in \mathbb{F}_2, i = 1, ..., n$ such that $X = \sum_{i=1}^{n} e_i q^{2i}$. If $\sum_i e_i = 0$, then according to Lemma 8 we are done with $j = 0$. Otherwise, let j be the smallest index i such that $e_i = 1$. Since $q^2 \mod p$ belongs to K, the degree n' of its minimal polynomial p' divides n. (This polynomial can be easily computed, but it is not needed by the algorithm.) By definition, we have $p'_0 + p'_1 q^2 + ... + p'_{n-1} q^{2(n'-1)} + q^{2n'} = 0 \bmod p$. Since p' is irreducible, we have $p'_0 = 1$ (otherwise $X|p'$) and $\sum_{i=0}^{n'} p'_i = 1$ (otherwise $X + 1|p'$). Let us define $e_i := 0$ for $i > n$ and $p'_i := 0$ for $i > n'$. We have

$$X \equiv \sum_{i=1}^{n} e_i q^{2i} \equiv \sum_{i=1}^{n} e_i q^{2i} + q^{2j}\left(1 + p'_1 q^2 + ... + p'_{n'-1} q^{2(n'-1)} + q^{2n'}\right) \equiv \sum_{i=j+1}^{j+n} (e_i + p'_{i-j}) q^{2i}.$$

Moreover, $\sum_{i=j+1}^{j+n}(e_i + p'_{i-j}) = (1 + \sum_{i=1}^{n} e_i) + \left(1 + \sum_{i=0}^{n'} p'_i\right) = 0$. Together with Lemma 8, this shows that T_j is a basis of K over $\mathbb{F}_2$. □

In short, our second algorithm is as follows

1. Apply Mesirov and Sweet's algorithm to p; get q and $\tilde{m}_1$.
2. If n is not prime, check whether $S_0 = \{q^2, q^4, ..., q^{2n}\}$ is a basis of K over $\mathbb{F}_2$. If it is not, abort.
3. Decompose X in the basis S_0.
4. Determine j as in the proof of Proposition 9.
5. Compute $\tilde{m}_{j+1}, ..., \tilde{m}_{j+n}$ using Equation 6 and apply Lemma 4.

Step 1 and Step 3 both require solving a linear system of size $n \times n$ over $\mathbb{F}_2$ which can be done in time $O(n^3)$; the remaining steps are comparatively fast. The length of $\tilde{m}_i$ is $i(n+1)-1$ and we have $i \leq j + n$ and $j < n$. After application of the mappings μ_L and μ_U, we may apply Proposition 3 with $L = 4n^2 + 2n + 1$, resulting in preimages of length $O(n^3)$ for any matrix. The algorithm is guaranteed to succeed if n is prime, and as argued above when n is composite but reasonably large it will likely succeed with a very large probability.

[1] The Tillich-Zémor hash function is *a priori* even weaker when n is not prime, due to the subgroup structure of $SL(2, K)$ [11].

7 Discussion

In this paper, we presented very efficient algorithms computing preimages for the Tillich-Zémor hash function. We first gave a second preimage algorithm. Then we reduced the problem of finding preimages for the Tillich-Zémor hash function to the problem of precomputing a few preimages with certain properties. Subsequently, we gave two algorithms for the precomputing part:

1. The first algorithm produces messages of length $O(n)$, resulting in generic preimages of length $O(n^2)$. We provided theoretical and experimental evidence that it runs in probabilistic time $O(n^4)$ and succeeds with a very large probability on the function parameters.
2. The second algorithm produces messages of length $O(n^2)$, resulting in generic preimages of length $O(n^3)$. We gave a proof that it always succeeds when n is prime, and arguments that it succeeds for most polynomials p when n is not necessarily prime but is reasonably large. The algorithm runs in deterministic time $O(n^3)$.

Since the size of $SL(2, \mathbb{F}_{2^n})$ is about 2^{3n}, it seems reasonable to conjecture that preimages of length $3n$ exist for any matrix. However, even if this conjecture is true, it is not clear whether there exists an efficient algorithm computing preimages of this length. We leave that question as an interesting open problem. From a practical point of view, our first algorithm is the most interesting one since it produces shorter messages. On the other hand, the second algorithm is more appealing from a theoretical point of view. In particular, it provides a constructive proof that the Cayley graphs corresponding to the Tillich-Zémor hash function satisfy Babai's conjecture (that is, that they have a polylogarithmic diameter [7]), at least when we restrict n to the prime values. Besides those important results, in the argumentation for our first algorithm we provided an extension of Mesirov and Sweet's result [8] and a connection to some arithmetic sequence of polynomials, both of which are of independent interest.

Grassl et al.'s attack found collisions for the Tillich-Zémor hash function. In this paper, we showed that the function is not even one-way. The attacks also break its vectorial and projective variants [10]. Our work puts a final end to the story of the Tillich-Zémor hash function, one of the oldest and most elegant hash functions in the literature. Similar hash functions that are using the same design (replacing the group $SL(2, K)$ and the generators A_0, A_1 by other groups and generators) have also been cryptanalyzed recently [14,13,9].

Nevertheless, we point out that these particular attacks do not invalidate the generic design. The key tool in the cryptanalysis of Tillich-Zémor hash function is Mesirov and Sweet's algorithm which is specific to quotients X and $X + 1$ in the Euclidean algorithm. At the current state of knowledge, collision and preimage resistances are recovered if we replace the matrices A_0 and A_1 by $B_0 := \left(\begin{smallmatrix} X^2 & 1 \\ 1 & 0 \end{smallmatrix}\right)$ and $B_1 := \left(\begin{smallmatrix} X+1 & 1 \\ 1 & 0 \end{smallmatrix}\right)$. Moreover, since current attacks do not allow controlling the form of the collisions and preimages, security might also be recovered by introducing some simple redundancy in the messages. It might even be sufficient to replace A_0 by A_0^2 or A_0^3. More generally, similar hash functions can be constructed from other non-Abelian groups and generators.

The generic design of the Tillich-Zémor hash function has many advantages over traditional hash functions like SHA: it has inherent parallelism, potentially efficient implementations in a wide range of contexts and a security equivalent to some concise mathematical problems [4]. For these reasons, we do not recommend to give it up but on the contrary, we suggest the community to look for secure and insecure instances. In the same way as many RSA instances can be insecure, especially when they are optimized for efficiency, we believe that the particularly efficient Tillich-Zémor hash function may be an unfortunately insecure instance of a more generally sound design.

Acknowledgements We are grateful to Gilles Zémor for pointing us an error in an early version of Proposition 5. We also thank Sylvie Baudine, François Koeune and François-Xavier Standaert for their help in improving this paper.

References

1. K. S. Abdukhalikov and C. Kim. On the security of the hashing scheme based on SL2. In *FSE '98: Proceedings of the 5th International Workshop on Fast Software Encryption*, pages 93–102, London, UK, 1998. Springer-Verlag.
2. D. Charles, E. Goren, and K. Lauter. Cryptographic hash functions from expander graphs. *J. Cryptology*, 22(1):93–113, 2009.
3. C. Charnes and J. Pieprzyk. Attacking the SL2 hashing scheme. In *ASIACRYPT '94: Proceedings of the 4th International Conference on the Theory and Applications of Cryptology*, pages 322–330, London, UK, 1995. Springer-Verlag.
4. G. de Meulenaer, C. Petit, and J.-J. Quisquater. Hardware implementations of a variant of Zémor-Tillich hash function: Can a provably secure hash function be very efficient? Preprint, 2009.
5. W. Geiselmann. A note on the hash function of Tillich and Zémor. In D. Gollmann, editor, *Fast Software Encryption*, volume 1039 of *Lecture Notes in Computer Science*, pages 51–52. Springer, 1996.
6. M. Grassl, I. Ilic, S. Magliveras, and R. Steinwandt. Cryptanalysis of the Tillich-Zémor hash function. Cryptology ePrint Archive, Report 2009/376, 2009. `http://eprint.iacr.org/`.
7. H. A. Helfgott. Growth and generation in $SL_2(Z/pZ)$, 2005.
8. J. P. Mesirov and M. M. Sweet. Continued fraction expansions of rational expressions with irreducible denominators in characteristic 2. *Journal of Number Theory*, 27:144–148, 1987.
9. C. Petit, K. Lauter, and J.-J. Quisquater. Full cryptanalysis of LPS and Morgenstern hash functions. In R. Ostrovsky, R. D. Prisco, and I. Visconti, editors, *SCN*, volume 5229 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 2008.
10. C. Petit, J.-J. Quisquater, J.-P. Tillich, and G. Zémor. Hard and easy components of collision search in the zémor-tillich hash function: New attacks and reduced variants with equivalent security. In M. Fischlin, editor, *CT-RSA*, volume 5473 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2009.
11. R. Steinwandt, M. Grassl, W. Geiselmann, and T. Beth. Weaknesses in the $SL_2(\mathbb{F}_{2^n})$ hashing scheme. In *Proceedings of Advances in Cryptology - CRYPTO 2000: 20th Annual International Cryptology Conference*, 2000.
12. J.-P. Tillich and G. Zémor. Hashing with SL_2. In Y. Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 40–49. Springer, 1994.
13. J.-P. Tillich and G. Zémor. Collisions for the LPS expander graph hash function. In N. P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 254–269. Springer, 2008.
14. J.-P. Tillich and G. Zémor. Group-theoretic hash functions. In *Proceedings of the First French-Israeli Workshop on Algebraic Coding*, pages 90–110, London, UK, 1993. Springer-Verlag.
15. G. Zémor. Hash functions and graphs with large girths. In D. W. Davies, editor, *EUROCRYPT*, volume 547 of *Lecture Notes in Computer Science*, pages 508–511. Springer, 1991.
16. G. Zémor. Hash functions and Cayley graphs. *Des. Codes Cryptography*, 4(4):381–394, 1994.

A Can we also compute preimages? A toy example.

In this section, we provide toy examples of our algorithms. We use $n = 11$ and the "smallest" irreducible polynomial of degree 11 which is $p(X) = X^{11} + X^2 + 1$.

The representation of the sentence "*Grassl et al. have shown how to find collisions. But can we also compute preimages?*" in ASCII is 47 72 61 73 73 6c 20 65 74 20 61 6c 2e 20 68 61 76 65 20 73 68 6f 77 6e 20 68 6f 77 20 74 6f 20 66 69 6e 64 20 63 6f 6c 6c 69 73 69 6f 6e 73 2e 20 42 75 74 20 63 61 6e 20 77 65 20 61 6c 73 6f 20 63 6f 6d 70 75 74 65 20 70 72 65 69 6d 61 67 65 73 3f to which corresponds the message m_{text} =01000111 01110010 01100001 01110011 01110011 01101100 00100000 01100101 01110100 00100000 01100001 01101100 00101110 00100000 01101000 01100001 01110110 01100101 00100000 01110011 01101000 01101111 01110111 01101110 00100000 01101000 01101111 01110111 00100000 01110100 01101111 00100000 01100110 01101001 01101110 01100100 00100000 01100011 01101111 01101100 01101100 01101001 01110011 01101001 01101111 01101110 01110011 00101110 00100000 01000010 01110101 01110100 00100000 01100011 01100001 01101110 00100000 01110111 01100101 00100000 01100001 01101100 01110011 01101111

00100000 01100011 01101111 01101101 01110000 01110101 01110100 01100101 00100000 01110000 01110010 01100101 01101001 01101101 01100001 01100111 01100101 01110011 00111111.The Tillich-Zémor hash value of this message for polynomial $p(X)$ is

$$h = \begin{pmatrix} X^7+X^5+X^4+X^2+1 & X^8+X^2+X+1 \\ X^9+X^8+X^7+X^5+X^4+X^3+X & X^9+X^6+X^5+X^4+X^2+X \end{pmatrix}.$$

We compute other preimages of this matrix using the algorithms of this paper.

We start with the second preimage algorithm of Section 3. The Mesirov and Sweet algorithm applied to p gives $d(X) = X^{10} + X^8 + X^7 + X^6 + X^5 + X^2 + X + 1$ and the message $m = 01011000010$. The message

$$\tilde{m} = \mu_L(m)\mu_L(m) = [(101111001011)(11011000010)0][(101111001011)(11011000010)0]$$

is a preimage of the identity matrix, hence $H(m_{text}\tilde{m}) = H(m_{text}) = h$.

We now apply our preimage algorithms to h. We first change the generators and we look for a preimage of

$$h' = \begin{pmatrix} X^{10}+X^8+X & X^9+X^8+X^7+X^5+X^4+X^3+X \\ X^9+X^6+X^5+X^3+X^2 & X^{10}+Xy+X^8+X^7+X^6+1 \end{pmatrix}$$

for the modified Tillich-Zémor hash function. We first write

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \alpha & 1 \end{pmatrix} \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & \beta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix}^3 \begin{pmatrix} 1 & 0 \\ \gamma & 1 \end{pmatrix}$$

with

$$\begin{cases} \alpha = X^{10} + X^9 + X^8 + X^7 + X^6 + X^5 + X^4 + 1 \\ \beta = X^{10} + X^7 + X^6 + X^5 + X^3 + X^2 + X \\ \gamma = X^8 + X^4 + 1 \end{cases}.$$

Now, we illustrate our first precomputing algorithm. We take $R = 10$ and generate random polynomials p' of degree R. We apply Mesirov and Sweet's algorithm to $a = pp'$. If successful, we obtain a d value and compute the corresponding α value. If this α value is linearly independent (over $\mathbb{F}_{2^n}/\mathbb{F}_2$) with the previous values, we keep it and compute m such that $\begin{pmatrix} a & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \end{pmatrix} H'(m)$. The following table summarizes the results obtained.

p'	d	α	$\perp?$	m
$X^{10}+X^9+X^8+X^7+X^6+X^5+X^4+X^3+1$	$X^{20}+X^{19}+X^{15}+X^{14}+X^{13}+X^{11}+X^{10}+X^9+X^5+X^4+X$	$X^8+X^7+X^6+X^5+X^4+X^3+X^2$	yes	m_1 =00110100100010010100
$X^{10}+X^6+X^5+X^3+X^2+X+1$	-	-	-	-
$X^{10}+X^9+X^7+X^6+X^5+X^4+X^3+X^2+1$	$X^{20}+X^{18}+X^{16}+X^{15}+X^{14}+X^{12}+X^9+X^7+X^5+X^3+X$	$X^{10}+X^9+X^8+X^7+X^6+X^5+X^4+1$	yes	m_2 =10111110100101110100
$X^{10}+X^9+X^5+X^4+X^2+X+1$	$X^{20}+X^{19}+X^{18}+X^{15}+X^6+X^5+X^3+X^2+X+1$	$X^{10}+X^9+X^8+X^7+X^6+X^3+X+1$	yes	m_3 =01001011111010111100
$X^{10}+X^9+X^7+X^5+X^2+X+1$	-	-	-	-
$X^{10}+X^9+X^8+X^5+1$	$X^{20}+X^{19}+X^{17}+X^{16}+X^{14}+X^{13}+X^9+X^6+X^5+X^4+X^2+1$	$X^{10}+X^9+X^6+X^5+X^4+X^3+X^2+X+1$	yes	m_4 =00101001111010111001001
$X^{10}+X^9+X^5+X+1$	-	-	-	-
$X^{10}+X^9+X^8+X^6+X^4+X^3+1$	-	-	-	-
$X^{10}+X^7+X^5+X^3+X^2+X+1$	-	-	-	-
$X^{10}+X^8+X^4+X^3+1$	-	-	-	-
$X^{10}+X^9+X^8+X^4+1$	-	-	-	-
$X^{10}+X^8+X^3+X^2+1$	$X^{20}+X^{17}+X^{14}+X^{10}+X^8+X^7+X^3+X+1$	$X^{10}+X^7+X^5+X^4+1$	yes	m_5 =01011101011101011110
$X^{10}+X^9+X^7+X^5+X^2+X+1$	-	-	-	-
$X^{10}+X^4+X^3+X^2+1$	-	-	-	-
$X^{10}+X^9+X^8+X^6+X^5+X^4+X^3+X^2+1$	$X^{20}+X^{15}+X^{13}+X^{10}+X^9+X^8+X^7+X^5+X^3+X$	$X^{10}+X^9+X^7+X^6$	yes	m_6 =10010110100111001010
$X^{10}+X^7+X^6+X^2+1$	-	-	-	-
$X^{10}+X^8+X^4+X^3+1$	-	-	-	-
$X^{10}+X^7+1$	$X^{20}+X^{18}+X^{16}+X^{14}+X^{12}+X^{11}+X^{10}+X^7+X^3+X^2+1$	$X^9+X^6+X^5+X^4+X^3+X$	yes	m_7 =01001000101110001 0001
$X^{10}+X^9+X^7+X^5+X^4+X^3+X^2+X+1$	-	-	-	-
$X^{10}+X^8+X^3+X^2+1$	$X^{20}+X^{17}+X^{14}+X^{10}+X^8+X^7+X^3+X+1$	$X^{10}+X^7+X^5+X^4+1$	no	
$X^{10}+X^6+X^5+X^3+X^2+X+1$	-	-	-	-
$X^{10}+X^8+X^5+X+1$	$X^{20}+X^{19}+X^{18}+X^{17}+X^{16}+X^{14}+X^{11}+X^{10}+X^8+X^4+X$	$X^8+X^7+X^6+X^5+X^3+X^2$	yes	m_8 =11111011001010111 1001
$X^{10}+X^7+X^4+X^3+1$	-	-	-	-
$X^{10}+X^9+X^7+X^5+X^4+X^3+X^2+X+1$	-	-	-	-
$X^{10}+X^9+X^8+X^7+1$	$X^{20}+X^{17}+X^{15}+X^{14}+X^{13}+X^{12}+X^9+X^8+X^7+X^5+X^4+X^2+1$	$X^8+X^5+X^4+X^3+X^2+X$	yes	m_9 =10000101001001000 1001
$X^{10}+X^9+X^8+X^5+X^4+X^2+1$	-	-	-	-
$X^{10}+X^9+X^7+X^6+X^4+X+1$	$X^{20}+X^{18}+X^{16}+X^{11}+X^9+X^8+X^5+X^4+X+1$	$X^9+X^8+X^5+X^2+X+1$	yes	m_{10} =10100001111100000 1010
$X^{10}+X^9+X^8+X^6+X^5+X^4+X^3+X^2+1$	$X^{20}+X^{15}+X^{13}+X^{10}+X^9+X^8+X^7+X^5+X^3+X$	$X^{10}+X^9+X^7+X^6$	no	
$X^{10}+X^7+X^6+X^4+X^2+X+1$	-	-	-	-
$X^{10}+X^7+X^6+X^5+X^2+X+1$	-	-	-	-
$X^{10}+X^9+X^8+X^6+X^2+X+1$	-	-	-	-
$X^{10}+X^9+X^8+X^7+1$	$X^{20}+X^{17}+X^{15}+X^{14}+X^{13}+X^{12}+X^9+X^8+X^7+X^5+X^4+X^2+1$	$X^8+X^5+X^4+X^3+X^2+X$	no	
$X^{10}+X^7+X^6+X^5+X^2+X+1$	-	-	-	-
$X^{10}+X^9+X^8+X^7+X^6+X^2+1$	-	-	-	-
$X^{10}+X^8+X^7+X^3+X^2+X+1$	$X^{20}+X^{17}+X^{16}+X^{15}+X^{14}+X^{13}+X^{11}+X^{10}+X^9+X^7+X$	$X^9+X^7+X^6+X^5+X^4+X+1$	yes	m_{11} =00001101101011100 0101

Writing α, β, γ in the basis obtained, we get $\alpha = \alpha_2$, $\beta = \alpha_2 + \alpha_8 + \alpha_{11}$ and $\gamma = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5 + \alpha_9 + \alpha_{10}$. Finally we obtain one preimage of h as

$$m' = m_\alpha 0 m_\beta 000 m_\gamma$$

where

$$\begin{aligned} m_\alpha &= \mu_L(m_2), \\ m_\beta &= \mu_U(m_2)\mu_U(m_8)\mu_U(m_{11}), \\ m_\gamma &= \mu_L(m_1)\mu_L(m_2)\mu_L(m_3)\mu_L(m_4)\mu_L(m_5)\mu_L(m_9)\mu_L(m_{10}). \end{aligned}$$

(Note that the terms composing m_β and m_γ can be permuted arbitrarily.)

Finally, we use our second precomputing algorithm to find yet another preimage. Let $\tilde{m_1} =$ 01011000010 be the message obtained by applying Mesirov and Sweet's algorithm to $a = p$. The corresponding q value is $q = d = X^{10} + X^8 + X^7 + X^6 + X^5 + X^2 + X + 1$. We recursively define

$$\tilde{m_i} := \tilde{m_{i-1}} 0 \tilde{m_1}.$$

Since n is prime, we know that $S_0 := \{q^2, q^4, \cdot, q^{1+n}\}$ is a basis of $\mathbb{F}_{2^n}/\mathbb{F}_2$. We have $X = q^2 + q^4 + q^6 + q^8 + q^{12} + q^{18} + q^{22}$ so $T_j := \{q^{2(i+j)} + X, i = 1, ..., n\}$ is a basis for $j = 1$. Let $\alpha_i = q^{2(j+1)} + X$ for $i = 1, ..., n$.

Writing α, β, γ in this basis, we get $\alpha = \alpha_1 + \alpha_4 + \alpha_6 + \alpha_8 + \alpha_9 + \alpha_{10} + \alpha_{11}$, $\beta = \alpha_1 + \alpha_2 + \alpha_4 + \alpha_6 + \alpha_7 + \alpha_8 + \alpha_9 + \alpha_{10} + \alpha_{11}$ and $\gamma = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_7 + \alpha_9 + \alpha_{11}$. Finally we compute a preimage of h as

$$m' = m_\alpha 0 m_\beta 000 m_\gamma$$

where

$$\begin{aligned} m_\alpha &= \mu_L(m_1)\mu_L(m_4)\mu_L(m_6)\mu_L(m_8)\mu_L(m_9)\mu_L(m_{10})\mu_L(m_{11}), \\ m_\beta &= \mu_U(m_1)\mu_U(m_2)\mu_U(m_4)\mu_U(m_6)\mu_U(m_7)\mu_U(m_8)\mu_U(m_9)\mu_U(m_{10})\mu_U(m_{11}), \\ m_\gamma &= \mu_L(m_1)\mu_L(m_2)\mu_L(m_3)\mu_L(m_4)\mu_L(m_7)\mu_L(m_9)\mu_L(m_{11}). \end{aligned}$$

B Further experimental results on our first precomputing algorithm

Since it is unlikely that we can give a better theoretical analysis of our first precomputing algorithm, we provide additional experimental results in this section.

The results shown in Figure 1 were obtained with each "shortest" polynomial of degree n. Figure 2 shows similar results for randomly chosen polynomials. The algorithm always succeeds with a value R satisfying $\frac{2^R}{R} \geq 6n$. For reasonably large n, it always succeeds when $\frac{2^R}{R} \geq 4n$. When n increases the points get closer to the bound $\frac{2^R}{R} \geq 2n$. Some of the points of Figure 1 and Figure 2 differ, but the differences are small and they only appear for small n.

n	R	n	R	n	R
5	7	67	11	257	13
7	7	73	11	277	13
11	9	83	11	307	13
13	9	97	12	331	14
17	9	103	12	353	14
19	9	109	12	379	14
23	9	127	12	401	14
29	10	137	12	449	14
31	9	149	12	499	14
37	10	167	12	607	15
41	10	191	13	1021	15
47	11	211	13	2039	16
59	11	233	13		

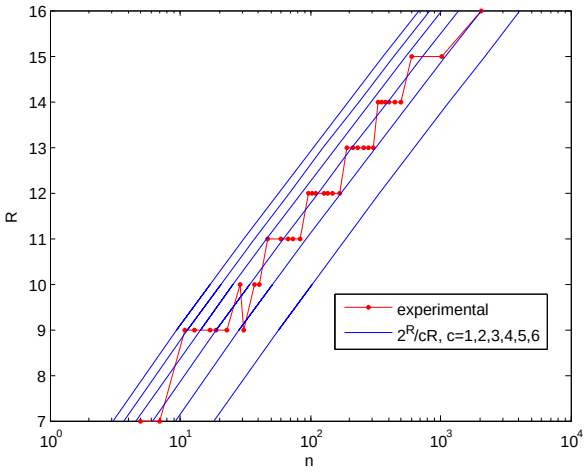

Fig. 2. Minimal value R for different values n and random polynomials. The points on the staircase-like curve are experimental results. The other curves are $n = \frac{2^R}{cR}$ for $c = 1, 2, 3, 4, 5, 6$.

To study these differences in more details, we generated twenty random polynomials of degrees $n = 11$, $n = 47$ and $n = 127$. For each polynomial, we recorded the shortest value of R for which the algorithm succeeded. (We started with short R values and increased R when 1000 α values linearly dependent with the previous ones were found). The results are presented in Table 1. The minimal value R for the success of the algorithm does not depend very much on the parameter p but only on its degree. Moreover, this dependence seems to disappear for reasonably large degrees n. Therefore, the algorithm is likely to succeed for any p as long as $\frac{2^R}{R} \geq 4n$ (for reasonably large n values).

Table 1. Variability of the minimal R needed for the algorithm when the polynomial p is randomly chosen. For each degree n and each R value, the table indicates how many polynomials p among the 20 polynomials generated required at least a randomness R.

$n = 11$		$n = 47$		$n = 127$	
$R = 7$	3/20	$R = 10$	11/20	$R = 12$	20/20
$R = 8$	14/20	$R = 11$	9/20		
$R = 9$	3/20				

Choosing R close to the minimal value will improve the efficiency of the algorithm since it performs linear algebra on vectors of length $N + R$. On the other hand, if R is chosen too close to the minimum, the algorithm will have to generate more polynomials p' before getting n independent elements α_i. In Table 2, it seems more efficient to choose $R = 10$ instead of $R = 8$ for the polynomial $p(X) = X^{11} + X^2 + 1$, and $R = 12$ instead of $R = 10$ for the polynomial $p(X) = X^{47} + X^5 + 1$. The reason appears clearly in the toy example of the previous section, presented in Table 1: in this example, all the dependencies obtained between the α values do actually come from the fact that the same polynomials are generated various times.

Table 2. Number of iterations needed to obtain a basis of K, for various R and the polynomials $p(X) = X^{11} + X^2 + 1$ and $p(X) = X^{47} + X^5 + 1$. The first row gives the total number of polynomials generated, the second row gives the number of times the Mesirov and Sweet's system had no solution, and the last row gives the number of times a new α value was linearly dependent on the previous ones. For each value R, we performed the experiment three times.

$p(X) = X^{11} + X^2 + 1$																		
	$R = 8$			$R = 9$			$R = 10$			$R = 15$			$R = 20$			$R = 50$		
# pol. generated	48	36	25	41	61	46	28	24	38	35	23	19	27	26	17	25	28	22
# no sol. MS	21	9	6	28	44	30	15	8	22	22	11	8	14	11	6	11	13	10
# dependencies	16	16	8	2	6	5	2	5	5	2	1	0	2	4	0	3	4	1

$p(X) = X^{47} + X^5 + 1$																		
	$R = 10$			$R = 11$			$R = 12$			$R = 15$			$R = 20$			$R = 50$		
# pol. generated	211	293	209	119	110	110	100	131	101	94	107	93	89	98	87	104	105	111
# no sol. MS	103	150	103	47	46	46	44	73	53	44	55	44	40	50	38	54	54	61
# dependencies	61	96	59	25	17	17	9	11	1	3	5	2	2	1	2	3	4	3

To conclude this section, we observe that in the experiments of Table 2, the Mesirov and Sweet's system had solutions respectively $49,21\%$ and $51,36\%$ of times for the polynomials $p(X) = X^{11} + X^2 + 1$ and $p(X) = X^{47} + X^5 + 1$. This confirms the analysis of Section 5.1.