# Hard and easy Components of Collision Search in the Zémor-Tillich Hash Function: new Attacks and Reduced Variants with Equivalent Security

Christophe Petit[1*], Jean-Jacques Quisquater[1],
Jean-Pierre Tillich[2] and Gilles Zémor[3]

| 1 | 2 | 3 |
|---|---|---|
| UCL Crypto Group** | Equipe SECRET | Institut de Mathématiques de Bordeaux |
| Université catholique de Louvain | INRIA Rocquencourt | Université Bordeaux 1 |
| Place du levant 3 | | 351, cours de la Libération |
| 1348 Louvain-la-Neuve, Belgium | 78153 Le Chesnay, France | 33405 Talence, France |

e-mails: christophe.petit@uclouvain.be,jjq@uclouvain.be,
jean-pierre.tillich@inria.fr,Gilles.Zemor@math.u-bordeaux1.fr

**Abstract.** The Zémor-Tillich hash function has remained unbroken since its introduction at CRYPTO'94. We present the first *generic* collision and preimage attacks against this function, in the sense that the attacks work for any parameters of the function. Their complexity is the *cubic root* of the birthday bound; for the parameters initially suggested by Tillich and Zémor they are very close to being practical. Our attacks exploit a separation of the collision problem into an easy and a hard component. We subsequently present two variants of the Zémor-Tillich hash function with essentially the same collision resistance but reduced outputs of $2n$ and $n$ bits instead of the original $3n$ bits. Our second variant keeps only the hard component of the collision problem; for well-chosen parameters the best collision attack on it is the birthday attack.

## 1 Introduction

Since its introduction at CRYPTO'94, the Zémor-Tillich hash function has kept on appealing Cryptographers by its originality, its elegance, its simplicity and its security. The function computation can be parallelized and even the serial version is quite efficient as it only requires XOR, SHIFT and TEST operations. Uniform distribution of the outputs follows from a graph theoretical interpretation of the hash computation, and collision resistance is strictly equivalent to an interesting group theoretical problem [9].

A few publications have claimed attacks on the Zémor-Tillich hash function. However, a closer look at these papers reveals that the scheme has not been seriously threatened so far. Some of the claimed "attacks" are unpractical, creating

* Research Fellow of the Belgian Fund for Scientific Research (F.R.S.-FNRS) at Université catholique de Louvain (UCL).
** A member of BCRYPT network.

very long colliding messages [3]. Others are trapdoor attacks that can be avoided by fixing the parameters in an appropriate way [2,1,8]. A last, important class of attacks are subgroup attacks [8], damaging for particular parameters in a similar way as RSA algorithm can be insecure if the parameters are not correctly generated. For well-chosen parameters, the function has remained unbroken so far.

In this paper, we present new collision and preimage subgroup attacks against the Zémor-Tillich hash function. Unlike previous ones, our attacks are generic in the sense that they work for any parameters of the function. With a time complexity close to $2^{n/2}$, our attacks beat by far the birthday bound and ideal preimage complexities which are $2^{3n/2}$ and $2^{3n}$ for the Zémor-Tillich hash function. The attacks are practical up to $n \approx 120, 130$ that is very close to the parameter's lower bound $n \geq 130$ initially proposed by Zémor and Tillich. As the attacks include a birthday search in a reduced set of size $2^n$ they do not invalidate the scheme but rather suggest that the initial parameters were too small.

Our attacks exploit a separation of the collision problem into an easy and a hard component, and suggest that an output of $n$ bits should be extracted from the original $3n$ bits of Zémor-Tillich. We consequently present two reduced versions of Zémor-Tillich, the vectorial and projective versions with output sizes respectively $2n$ and $n$, and we show that their collision resistance is essentially equivalent to the collision resistance of the original Zémor-Tillich.

This paper is organized as follows: the Zémor-Tillich hash function is recalled in Section 2. In Section 3 we present a general result separating hard and easy components of the collision problem, then we apply this result in Section 4 to obtain a generic collision search algorithm with time complexity close to $2^{n/2}$ (while the birthday bound is $2^{3n/2}$). This collision algorithm is extended in Section 5 to a generic preimage attack with the same complexity (while the ideal bound would be $2^{3n}$), and memory free versions of these algorithms are given in Section 6. Finally, we introduce the vectorial and projective versions of Zémor-Tillich in Sections 7 and 8 and conclude the paper in Section 9.

## 2 The Zémor-Tillich Hash Function

Let $m = m_0 m_1 ... m_k$ be the bit string representation of a message $m$. Let $P_n(X)$ be an irreducible polynomial of degree $n$ (Tillich and Zémor suggested using $130 \leq n \leq 170$) and let us represent the field $\mathbb{F}_{2^n}$ by $\mathbb{F}_2[X]/(P_n(X))$. Let $A_0, A_1$ be the matrices of $G := SL(2, \mathbb{F}_{2^n})$ (the group of $2 \times 2$ matrices over $\mathbb{F}_{2^n}$ with unitary determinant) defined by

$$A_0 = \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix} \qquad A_1 = \begin{pmatrix} X & X+1 \\ 1 & 1 \end{pmatrix}$$

The Zémor-Tillich hash value of $m$ is defined as the matrix product [9]

$$h_{ZT}(m) := A_{m_0} A_{m_1} ... A_{m_k}.$$

As the group $SL(2, \mathbb{F}_{2^n})$ has size $2^n(2^{2n} - 1)$, the output size is roughly $3n$ bits if the matrices of $SL(2, \mathbb{F}_{2^n})$ are mapped to bitstrings.

## 3 Hard and Easy Components of Collision Search

The best attack so far against the Zémor-Tillich hash function has been the subgroup attack of Steinwandt et al. [8]. However, as this attack exploits subgroups of $SL(2, \mathbb{F}_{2^n})$ that are specific to composite degrees $n$ and particular polynomials $P_n(X)$, it can be simply prevented by choosing $n$ in an appropriate way.

In this section, we consider the generic subgroups of $SL(2, \mathbb{F}_{2^n})$ (subgroups existing for any parameter $n$), including the subgroups of diagonal or triangular matrices and the subgroups of matrices with a given left or right eigenvector. We show that finding elements of these subgroups together with their factorization is nearly as hard as finding collisions for the Zémor-Tillich hash function. As our reductions involve solving discrete logarithms in $\mathbb{F}_{2^n}^*$ we do not claim `ppt` (probabilistic polynomial time) reductions but reductions that are practical for the parameters initially suggested by Zémor and Tillich.

We start with an easy proposition that will simplify our proofs later.

**Proposition 1**

*(a) Let* $\begin{pmatrix} a & b \end{pmatrix}, \begin{pmatrix} a' & b' \end{pmatrix} \in \mathbb{F}_{2^n}^2 \setminus \{\begin{pmatrix} 0 & 0 \end{pmatrix}\}$ *and* $M \in SL(2, \mathbb{F}_{2^n})$ *such that* $\begin{pmatrix} a & b \end{pmatrix} M = \begin{pmatrix} a' & b' \end{pmatrix}$. *Then there exists* $\epsilon \in \mathbb{F}_{2^n}$ *such that* $M = \begin{pmatrix} a^{-1} & b \\ 0 & a \end{pmatrix} \begin{pmatrix} a' & b' \\ 0 & a'^{-1} \end{pmatrix} + \epsilon \begin{pmatrix} b \\ a \end{pmatrix} \begin{pmatrix} a' & b' \end{pmatrix}$.

*(b) If* $M_1 = \begin{pmatrix} a_0^{-1} & b_0 \\ 0 & a_0 \end{pmatrix} \begin{pmatrix} a_1 & b_1 \\ 0 & a_1^{-1} \end{pmatrix} + \epsilon_1 \begin{pmatrix} b_0 \\ a_0 \end{pmatrix} \begin{pmatrix} a_1 & b_1 \end{pmatrix}$ *and* $M_2 = \begin{pmatrix} a_1^{-1} & b_1 \\ 0 & a_1 \end{pmatrix} \begin{pmatrix} a_2 & b_2 \\ 0 & a_2^{-1} \end{pmatrix} + \epsilon_2 \begin{pmatrix} b_1 \\ a_1 \end{pmatrix} \begin{pmatrix} a_2 & b_2 \end{pmatrix}$ *then* $M_1 M_2 = \begin{pmatrix} a_0^{-1} & b_0 \\ 0 & a_0 \end{pmatrix} \begin{pmatrix} a_2 & b_2 \\ 0 & a_2^{-1} \end{pmatrix} + (\epsilon_1 + \epsilon_2) \begin{pmatrix} b_0 \\ a_0 \end{pmatrix} \begin{pmatrix} a_2 & b_2 \end{pmatrix}$.

PROOF: Part (a) is implied by the two following observations:

- For $\epsilon = 0$ we have $\begin{pmatrix} a & b \end{pmatrix} \begin{pmatrix} a^{-1} & b \\ 0 & a \end{pmatrix} \begin{pmatrix} a' & b' \\ 0 & a'^{-1} \end{pmatrix} = \begin{pmatrix} a' & b' \end{pmatrix}$.
- If $M_1, M_2 \in SL(2, \mathbb{F}_{2^n})$ satisfy $(a, b)M_1 = (a, b)M_2 = (a', b')$ then $M_1 + M_2 = \epsilon \begin{pmatrix} b \\ a \end{pmatrix} \begin{pmatrix} a' & b' \end{pmatrix}$. Indeed, let $c, d$ such that $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is unitary and let $\begin{pmatrix} a' & b' \\ c_1 & d_1 \end{pmatrix} := \begin{pmatrix} a & b \\ c & d \end{pmatrix} M_1$ and $\begin{pmatrix} a' & b' \\ c_2 & d_2 \end{pmatrix} := \begin{pmatrix} a & b \\ c & d \end{pmatrix} M_2$. As $M_1, M_2$ and $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ are in $SL(2, \mathbb{F}_{2^n})$, we have $\det \begin{pmatrix} a' & b' \\ c_1 & d_1 \end{pmatrix} = \det \begin{pmatrix} a' & b' \\ c_2 & d_2 \end{pmatrix} = 1$. We get

$$\begin{aligned} M_1 + M_2 &= \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} \left[ \begin{pmatrix} a' & b' \\ c_1 & d_1 \end{pmatrix} + \begin{pmatrix} a' & b' \\ c_2 & d_2 \end{pmatrix} \right] = \begin{pmatrix} d & b \\ c & a \end{pmatrix} \begin{pmatrix} 0 & 0 \\ c_1 + c_2 & d_1 + d_2 \end{pmatrix} \\ &= \begin{pmatrix} b \\ a \end{pmatrix} \begin{pmatrix} c_1 + c_2 & d_1 + d_2 \end{pmatrix}. \end{aligned}$$

Moreover, as $\begin{pmatrix} c_1 + c_2 & d_1 + d_2 \end{pmatrix} \begin{pmatrix} b \\ a \end{pmatrix} = a(d_1 + d_2) + b(c_1 + c_2) = (ad_2 + bc_2) + (ad_1 + bc_1) = 0$, we get the result.

Part (b) is a straightforward computation. □

We now define the (generalized) representation problem in $\mathbb{F}_{2^n}^*$ and we show how it can be solved for small $n$ (and certainly if $n \leq 170$).

**Problem 1** Representation problem in $\mathbb{F}_{2^n}^*$*: Given $N$ (randomly chosen) elements $g_i \in \mathbb{F}_{2^n}^*$, find a factorization $\prod g_i^{e_i} = 1$ such that $\sum |e_i|$ is* not too large.
Generalized representation problem in $\mathbb{F}_{2^n}^*$*: Given $N$ (randomly chosen) elements $g_i \in \mathbb{F}_{2^n}^*$ and a (randomly chosen) element $g_0 \in \mathbb{F}_{2^n}^*$, find a factorization $\prod g_i^{e_i} = g_0$ such that $\sum |e_i|$ is* not too large.

**Proposition 2** *The (generalized) representation problem can be solved in groups $\mathbb{F}_{2^n}^*$ where the discrete logarithm problem can be solved.*

Proof: Let $g_i \in \mathbb{F}_{2^n}^*, i = 0, ...N$. Let $g$ a generator of $\mathbb{F}_{2^n}^*$, and let $\alpha_i$ be the discrete logarithms of $g_i$ with respect to base $g$. The representation problem amounts to solving the following problem: find $\{e_i\}$ such that $\sum e_i\alpha_i = \alpha_0 \bmod (2^n - 1)$ and $\sum |e_i|$ is *not too large*. A good solution to this problem can be computed with the LLL algorithm [4].□

If the exponents $\alpha_i$ are random numbers uniformly distributed in $[1, 2^n - 1]$ the smallest solution has expected size $\sum_i |e_i|$ about $N2^{n/N}$ (approximating that there is no collision, the sums $\sum e_i\alpha_i$ for $e_i \leq 2^{n/N}$ produce the $2^n - 1$ possible values). The LLL algorithm actually gives a solution such that $\sum |e_i|^2$ is close to optimal, but this is enough for our purposes. By the LLL approximation bound, the solution provided using LLL has a norm 2 smaller than $\sqrt{N}2^{n/N+N}$ which is subexponential for $N \approx \sqrt{n}$. In practice, LLL performs much better and in the analysis of our algorithms, we will approximate that the isze of the solution given by LLL algorithm is also about $N2^{n/N}$.

With this method, the representation problem in $\mathbb{F}_{2^n}^*$ can be solved if discrete logarithms can be computed, in particular the representation problem can be solved today for $n \leq 170$. The following result follows from Proposition 2.

**Proposition 3** *Let $n$ be such that discrete logarithms can be solved in $\mathbb{F}_{2^n}^*$. Let $\mathcal{D}, \mathcal{T}^{up}, \mathcal{T}^{low}, \mathcal{L}^v, \mathcal{R}^v \subset SL(2, \mathbb{F}_{2^n})$ be the subgroups of diagonal, upper and lower triangular matrices and the subgroup of matrices with left or right eigenvector $v$. If an attacker can compute $N$ random elements $M_i$ of one of these subgroups together with bit sequences $m_i$ of length at most $L$ hashing to these matrices, then he can also find a message $m$ such that $h_{ZT}(m) = I$. The message $m$ has expected size smaller than $NL2^{n/N}$ in the diagonal case and smaller than $NL2^{1+n/N}$ in the other cases.*

Proof: Clearly any diagonal matrix writes down as $D_i = \begin{pmatrix} a_i & 0 \\ 0 & a_i^{-1} \end{pmatrix}$ for some $a_i \in \mathbb{F}_{2^n}^*$. Let $\{e_i\}$ be a solution to the representation problem with respect to $\{a_i\}$, that is $\prod a_i^{e_i} = 1$. Construct $m$ as the concatenation of $e_1$ messages $m_1$, $e_2$ messages $m_2$, etc. (in any order). Then $h_{ZT}(m) = \prod D_i^{e_i} = \begin{pmatrix} \prod a_i^{e_i} & 0 \\ 0 & \prod a_i^{-e_i} \end{pmatrix} = I$.

Similarly, an upper triangular matrix $T_i$ writes down as $\begin{pmatrix} a_i & b_i \\ 0 & a_i^{-1} \end{pmatrix}$ for some $a_i \in \mathbb{F}_{2^n}^*, b_i \in \mathbb{F}_{2^n}$. Let $\{e_i\}$ be a solution to the representation problem with respect to $\{a_i\}$, that is $\prod a_i^{e_i} = 1$. Construct $m'$ as the concatenation of $e_1$ messages $m_1$, $e_2$ messages $m_2$, etc. (in any order) and $m = m'||m'$. Then $h_{ZT}(m') = \begin{pmatrix} 1 & b \\ 0 & 1^{-1} \end{pmatrix}$ for some $b \in \mathbb{F}_{2^n}$ and $h_{ZT}(m) = I$.

By definition each $M_i \in \mathcal{L}^{(a\ b)}$ satisfies $(a\ b)\, M_i = \lambda_i\, (a\ b)$ for some $\lambda_i \in \mathbb{F}_{2^n}^*$. Let $\{e_i\}$ be a solution to the representation problem with respect to $\{\lambda_i\}$, that is $\prod \lambda_i^{e_i} = 1$. Construct $m'$ as the concatenation of $e_1$ messages $m_1$, $e_2$ messages $m_2$, etc. (in any order) and $m = m'||m'$. Then $(a\ b)\, h_{ZT}(m') = (a\ b)$ which by Proposition 1 implies $h_{ZT}(m') = I + \epsilon \begin{pmatrix} b \\ a \end{pmatrix} (a\ b)$ hence $h_{ZT}(m) = I$.

The proof for $\mathcal{T}^{low}$ and $\mathcal{R}^v$ are similar and the claim on the message lengths follows from our analysis of the representation problem in $\mathbb{F}_{2^n}^*$. $\square$

The part of Proposition 3 concerning $\mathcal{L}^v$ and $\mathcal{R}^v$ has interesting graph interpretations that we give in Appendix A.

## 4 A New Generic Collision Attack

We now give an algorithm finding $N_2$ matrices $M_i$ such that $(1\ 0)\, M_i = \lambda_i\, (1\ 0)$ for some $\lambda_i \in \mathbb{F}_{2^n}^*$, and combining them as in Proposition 3 to find collisions for the Zémor-Tillich hash function.

We denote by $\mathbb{P}^1(\mathbb{F}_{2^n})$ the projective space of dimension 1 on $\mathbb{F}_{2^n}$, which is the set of equivalence classes of $\mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$ that results from identifying two vectors $(a_1\ b_1)$ and $(a_2\ b_2)$ if and only if $(a_2\ b_2) = \lambda\, (a_1\ b_1)$ for some $\lambda \in \mathbb{F}_{2^n}^*$. We denote by $[a : b]$ the projective point that is the equivalence class of a vector $(a\ b)$. To any message $m = m_1 m_2 ... m_k$ we associate two projective points $q(m), q_{-1}(m) \in \mathbb{P}^1(\mathbb{F}_{2^n})$ as follows. We define $(a(m)\ b(m)) := (1\ 0) \prod_{i=1}^{k} M_{m_i} = (1\ 0)\, h_{ZT}(m)$ and $(a'(m)\ b'(m)) := (1\ 0) \prod_{i=k}^{1} M_{m_i}^{-1} = (1\ 0)\, h_{ZT}^{-1}(m)$, then $q(m) := [a(m) : b(m)]$ and $q_{-1}(m) := [a'(m) : b'(m)]$.

Our algorithm first performs a birthday attack [11] to find collisions on the $q$ values as follows. Random messages $m$ and $m'$ of size $k > n/2$ are generated and stored together with $q(m)$ and $q(-m')$, until $m_1, m_2$ are found such that $q(m_1) = q_{-1}(m_2)$ (see Figure 1). As there are $2^n + 1$ points in $\mathbb{P}^1(\mathbb{F}_{2^n})$, the probability that $q(m_1) = q_{-1}(m_2)$ for some $m_1, m_2$ is $1 - \left(1 - \frac{2^{N_1}}{2^n+1}\right)^{2^{N_1}}$ after $2^{N_1}$ steps. In particular, after $2^{N_1} = 2^{n/2}$ steps we have a probability $1 - e^{-1} \approx 0.63$ to know a message $m := m_1||m_2$ of size $2k$ such that $(1\ 0)\, h_{ZT}(m) = \lambda\, (1\ 0)$ for some $\lambda \in \mathbb{F}_{2^n}^*$.

This collision search is repeated until $N_2$ distinct messages $m_i$ are found such that $(1\ 0)\, h_{ZT}(m_i) = \lambda_i\, (1\ 0)$ for some $\lambda_i \in \mathbb{F}_{2^n}^*$. To guarantee that the collisions found are all distinct, we may perform each collision search with a different length $k > n/2$, or choose $k$ slightly larger than $n/2 + \log_2(N_2)$, say $k = n/2 + \log_2(N_2) + 10$.

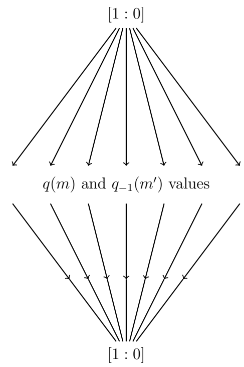


**Fig. 1.** Collision search on $q$ values.

The next step of the algorithm combines the messages $m_i$ to get a collision for the Zémor-Tillich hash function. As in the proof of Proposition 3, we compute a solution $\{e_i\}$ to the representation problem in $\mathbb{F}_{2^n}^*$ with respect to the $\lambda_i$, that is $\prod \lambda_i^{e_i} = 1$. From this solution, we finally construct a message $m'$ as the concatenation of each message $m_i$ repeated $e_i$ times (in any order), and a message $m = m'||m'$ that collides with the void message as shown in the proof of Proposition 3.

To analyze this attack, suppose that the $N_2$ collision searches are done with $k = n/2+1, ..., n/2+N_2$ and that the algorithm described in Section 3 is used to solve the representation problem. The expected size of the collision is then bounded by $(n/2+N_2)N_2 2^{n/N_2+2}$, the memory requirement is $2^{n/2+1}n$ and the time complexity is $N_2 2^{n/2+1}t + t_{REP}$ where $t$ is the time needed to compute one $q$ value and $t_{REP}$ is the time needed to solve the representation problem. In particular for $n = 130$ and $N_2 = 16$, this attack produces a collision to the void message of size about $2^{18}$ in time $2^{69}t$ and memory requirements $2^{69}$. The memory requirements will be removed in Section 6 by using distinguished points techniques [6].

## 5 A New Generic Preimage Attack

We now extend our ideas to a preimage attack. Interestingly, this attack has essentially the same complexity as the collision attack.

Suppose we want to find a preimage to a matrix $M = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, that is a message $m = m_1...m_k$ such that $M = h_{ZT}(m) = \prod M_{m_i}$. As we showed in previous section, random messages $m_i$ of size $L > n$ such that $\left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right) h_{ZT}(m_i) = \lambda_i \left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right)$ for some $\lambda_i \in \mathbb{F}_{2^n}^*$ can be found with memory $n2^{n/2+1}$ and time $2^{n/2+1}t$. Similarly, random messages $m_i, i = 0, ...N_2$ of size $L > n$ satisfying $\left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right) h_{ZT}(m_0) =$

$\lambda_0 \left( a\ b \right)$ and $\left( a\ b \right) h_{ZT}(m_i) = \lambda_i \left( a\ b \right), i > 0$ for some $\lambda_i \in \mathbb{F}_{2^n}^*$ can also be found with the same time and memory complexities.

Solving a (generalized) representation problem, we can compute $\{e_i\}$ such that $\prod \lambda_i^{e_i} = \lambda_0$, hence we can compute a message $m_0'$ of size $N_2 L 2^{n/N_2}$ and a matrix $M_0 := h_{ZT}(m_0')$ such that $\left( 1\ 0 \right) M_0 = \left( a\ b \right)$. Similarly, from $N_3$ different solutions to the representation problem $\prod \lambda_i^{e_i} = 1$ we get $N_3$ messages $m_i'$ of size $N_2 L 2^{n/N_2}$ such that $\left( a\ b \right) h_{ZT}(m_i') = \left( a\ b \right)$. Let $\left( c'\ d' \right) := \left( 0\ 1 \right) h_{ZT}(m_0')$. As $ad' + bc' = ad + bc = 1$, we have $a(d+d') + b(c+c') = 0$, that is $\left( c+c'\ d+d' \right) = \delta_0 \left( a\ b \right)$ for some $\delta_0 \in \mathbb{F}_{2^n}$.

According to Proposition 1, for all $i > 0$ there exists $\delta_i \in \mathbb{F}_{2^n}$ such that $h_{ZT}(m_i') = \left( \begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix} \right) + \delta_i \left( \begin{smallmatrix} b \\ a \end{smallmatrix} \right) \left( a\ b \right)$; moreover we have $h_{ZT}(m_{i_1}') h_{ZT}(m_{i_2}') = \left( \begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix} \right) + (\delta_{i_1} + \delta_{i_2}) \left( \begin{smallmatrix} b \\ a \end{smallmatrix} \right) \left( a\ b \right)$. Suppose the $\delta_i$ values generate $\mathbb{F}_{2^n}/\mathbb{F}_2$, which is very likely if $N_3$ is shortly bigger than $n$, say $N_3 = n + 10$. Then by solving a binary linear system, we can write $\delta_0 = \sum_{i \in I} \delta_i$ for some $I \subset \{1, ..., N_3\}$ of size $\leq n$ and hence $M_1 := \prod_{i \in I} h_{ZT}(m_i') = \left( \begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix} \right) + \delta_0 \left( \begin{smallmatrix} b \\ a \end{smallmatrix} \right) \left( a\ b \right)$. Finally, we have $M_0 M_1 = \left( \begin{smallmatrix} a & b \\ c' & d' \end{smallmatrix} \right) [\left( \begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix} \right) + \delta_0 \left( \begin{smallmatrix} b \\ a \end{smallmatrix} \right) \left( a\ b \right)] = \left( \begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \right)$.

This shows that any message made of $m_0'$ concatenated with any concatenation of the messages $m_i', i \in I$, is a preimage to $\left( \begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \right)$. The collision size is about bounded by $N_3(n/2 + N_2) N_2 2^{n/N_2 + 2}$, that is $12n^2(n + 10)$ if $N_2 = n$ and $N_3 = n + 10$. The memory requirement of this attack is $2^{n/2+1} n$ and the time complexity is $N_2 2^{n/2+1} t + t_{REP}$ where $t$ is the time needed to compute one $q$ value and $t_{REP}$ is the time needed to solve the representation problem (note that finding $N_3$ solutions to a representation problem essentially requires the same time as finding one solution because both times are essentially determined by the computation of the discrete logarithms). As for our collision attack, the memory requirements can be removed by using distinguished points techniques.

## 6 Memory-Free Versions of Our Attacks

The attacks of Sections 4 and 5 require storing two databases of about $2^{n/2}$ projective points in $\mathbb{P}^1(\mathbb{F}_{2^n})$ and their corresponding messages. We now remove the memory requirements by using distinguished points techniques [6].

Let $\alpha : \mathbb{P}^1(\mathbb{F}_{2^n}) \to \{0,1\}^k$ and $\beta : \mathbb{P}^1(\mathbb{F}_{2^n}) \to \{0,1\}$ be two "pseudorandom functions" and let $\varphi : \mathbb{P}^1(\mathbb{F}_{2^n}) \to \mathbb{P}^1(\mathbb{F}_{2^n})$ be defined by

$$p \to \varphi(p) = \begin{cases} q(\alpha(p)) & \text{if } \beta(p) = 0 \\ q_{-1}(\alpha(p)) & \text{if } \beta(p) = 1, \end{cases}$$

where $k > n$ is arbitrarily chosen and $q$ and $q_{-1}$ are defined as in Section 4.

The iterates $q_0, \varphi(q_0), \varphi(\varphi(q_0)), ...$ of $\varphi$ on $q_0$ all belong to the finite domain $\mathbb{P}^1(\mathbb{F}_{2^n})$ so at some point iterating $\varphi$ will produce a collision (see Figure 2), that is two points $p_1$ and $p_2$ such that $\varphi(p_1) = \varphi(p_2) = c$. If the behavior of $\varphi$ is sufficiently random then $\beta(p_1) \neq \beta(p_2)$ with a probability $1/2$, in which case

$\alpha(p_1)$ and $\alpha(p_2)$ can be combined to produce a message $m$ of size $2k$ such that $\begin{pmatrix} 1 & 0 \end{pmatrix} h_{ZT}(m) = \lambda \begin{pmatrix} 1 & 0 \end{pmatrix}$ for some $\lambda \in \mathbb{F}_{2^n}^*$.

The functions $\alpha$ and $\beta$ do not need to be "pseudorandom" in the strong cryptographic meaning, but only "sufficiently pseudorandom" for the above analysis to hold.

Now that the problem of finding a collision on the $q$ values has been translated in the problem of detecting a cycle in the iterates of $\varphi$, we can remove the memory requirements by standard techniques. We recall here the method of *distinguished points*; other methods are described in [7]. Let $\mathcal{D}_d := \{q = [a : b] \in \mathbb{P}^1(\mathbb{F}_{2^n}) | b \neq 0, lsb_d(a/b) = 0^d\}$ be sets of $2^{n-d}$ distinguished $q$ values such that their $d$ last bits are all 0. During the collision search, we only store the $q$ values that belong to $\mathcal{D}$ and only look for collisions on these particular $q$ values. Finding a collision $c'$ on distinguished points requires $2^{d-1}$ additional steps in average but the memory is reduced to $2^{n/2-d}$; if $d = n/2 - 10$ the time overhead is negligible and the memory requirements are very small (see Figure 3).

From the two distinguished points $p_1'$ and $p_2'$ that precede $c'$ in the iterates of $\varphi$, we can recover the points $p_1$ and $p_2$ that produce the actual collision $c$ as follows. Iterate again $\varphi$ on $p_1'$ and $p_2'$ and store only distinguished points but this time with $d = n/2 - 20$. After about $2^{n/2-10}$ steps on each side (and a small memory of about $2^{11}$) a collision $c''$ and preceding distinguished points $p_1''$ and $p_2''$ are found that are closer to the actual collision $c, p_1, p_2$. Iterating again from $p_1''$ and $p_2''$ with a larger distinguished-point set, we finally get the actual collision with small time overhead and small memory.

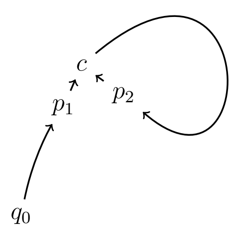


**Fig. 2.** Iterating $\varphi$ from some initial point $q_0$, we eventually get a collision $c$

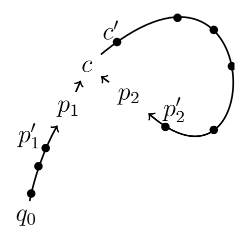


**Fig. 3.** Collision graph with markers on the distinguished points. The average distance between two distinguished points is $2^d$. The average length of the path is $2^{n/2}$. Finding a collision on a distinguished point requires essentially the same time as finding a general collision, as soon as $2^d << 2^{n/2}$.

With this method instead of the trivial collision search steps, our collision and preimage attacks require negligible memory and essentially the same time complexity. As the output of Zémor-Tillich is about $3n$ bits, these attacks are far better than birthday and optimal preimage bounds. In the following sections, we introduce two variants of Zémor-Tillich with reduced output sizes respectively $2n$ and $n$ bits, and we show that these variants are essentially as secure as the original Zémor-Tillich for sufficiently small parameters including the parameters initially suggested in [9].

## 7 Vectorial Version of Zémor-Tillich

Our first variant $h_{ZT}^{vec}$ is simply the first row of Zémor-Tillich, that is $h_{ZT}^{vec}(m) := \left( \begin{smallmatrix} a & b \end{smallmatrix} \right)$ if $h_{ZT}(m) = \left( \begin{smallmatrix} a & b \\ c & d \end{smallmatrix} \right)$. This variant was introduced in [5] by Petit et al. but without a proof of its equivalence to the original function. Alternatively, we may parameterize the function $h_{ZT}^{vec}$ by an initial vector $\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right) \neq \left( \begin{smallmatrix} 0 & 0 \end{smallmatrix} \right)$ as $h_{ZT}^{vec,\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right)}(m) := \left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right) h_{ZT}(m)$. Clearly, the output has $2n$ bits.

Finding a collision for this variant corresponds to finding two messages $m$ and $m'$ such that $\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right) h_{ZT}(m) = \left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right) h_{ZT}(m')$, in particular it is enough to find one message $m$ such that $\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right) h_{ZT}(m) = \left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right)$ (we call such a collision a *cyclic* collision). Finding a preimage to a vector $\left( \begin{smallmatrix} a & b \end{smallmatrix} \right)$ is finding a message $m$ such that $\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right) h_{ZT}(m) = \left( \begin{smallmatrix} a & b \end{smallmatrix} \right)$.

The following proposition shows that $h_{ZT}^{vec}$ is collision resistant if and only if the original function $h_{ZT}$ is collision resistant.

**Proposition 4** *If there exists a* `ppt` *(probabilistic polynomial time) algorithm that for randomly chosen starting vectors* $\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right) \neq \left( \begin{smallmatrix} 0 & 0 \end{smallmatrix} \right)$ *finds a collision on* $h_{ZT}^{vec,\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right)}$*, then there exists a* `ppt` *algorithm finding collisions for the original Zémor-Tillich function.*

Proof: Given a `ppt` algorithm $A^{vec}$ finding collisions for the vectorial version, we build a `ppt` algorithm $A^{mat}$ finding collisions for the original matrix version. The algorithm $A^{mat}$ first picks a random matrix $M_0 := \left( \begin{smallmatrix} a_0 & b_0 \\ c_0 & d_0 \end{smallmatrix} \right) \in SL(2, \mathbb{F}_{2^n})$ and runs $A^{vec}$ on $(a_0, b_0)$ to get two messages $m_{10}$ and $m_{11}$ corresponding to matrices $M_{10}$ and $M_{11}$ such that $(a_0, b_0)M_{10} = (a_0, b_0)M_{11} = (a_1, b_1)$. Without loss of generality, we can assume that $(a_1, b_1)$ is randomly uniformly distributed (otherwise we may just append the same randomly chosen sequence of bits to both messages). Algorithm $A^{mat}$ then calls again $A^{vec}$ on $(a_1, b_1)$ to get two matrices $M_{20}$ and $M_{21}$, etc. It repeats this operation $n + 1$ times.

Let $v_i := \left( \begin{smallmatrix} a_i & b_i \end{smallmatrix} \right)$ and $\widetilde{v_i} := \left( \begin{smallmatrix} b_i \\ a_i \end{smallmatrix} \right)$. According to Proposition 1(a), the matrices $M_{ij}$ write down as

$$M_{ij} = \begin{pmatrix} a_{i-1}^{-1} & b_{i-1} \\ 0 & a_{i-1} \end{pmatrix} \begin{pmatrix} a_i & b_i \\ 0 & a_i^{-1} \end{pmatrix} + \epsilon_{ij} \widetilde{v_{i-1}} v_i$$

for some $\epsilon_{ij} \in \mathbb{F}_{2^n}$. Applying Proposition 1(b) recursively, for any $e = e_1...e_{n+1} \in \{0,1\}^{n+1}$, we have

$$\prod_{i=1}^{n+1} M_{ie_i} = \begin{pmatrix} a_0^{-1} & b_0 \\ 0 & a_0 \end{pmatrix} \begin{pmatrix} a_{n+1} & b_{n+1} \\ 0 & a_{n+1}^{-1} \end{pmatrix} + \left( \sum_{i=1}^{n+1} \epsilon_{ie_i} \right) \widetilde{v_0} v_{n+1}.$$

For $1 \leq i \leq n+1$, let $\epsilon_i := \epsilon_{i0} + \epsilon_{i1}$. Seeing each $\epsilon_i$ as a binary vector of length $n$ over $\mathbb{F}_2$, these vectors are linearly dependent. Moreover, finding a subset $I$ of $\{1,...,n+1\}$ such that $\sum_{i \in I} \epsilon_i = 0$ simply amounts to invert a binary linear system, which is cubic in $n+1$.

We now conclude the description of $A^{mat}$. After computing $I \subset \{1,...,n+1\}$ such that $\sum_{i \in I} \epsilon_i = 0$, the algorithm $A^{mat}$ returns $m = m_{10}||m_{20}||...||m_{n+1,0}$ and $m' = m_{1e_1}||m_{2e_2}||...||m_{n+1,e_{n+1}}$ where $e_i = 1$ if and only if $i \in I$. By the discussion above, it is clear that

$$h_{ZT}^{mat}(m) = h_{ZT}^{mat}(m') = \begin{pmatrix} a_0^{-1} & b_0 \\ 0 & a_0 \end{pmatrix} \begin{pmatrix} a_{n+1} & b_{n+1} \\ 0 & a_{n+1}^{-1} \end{pmatrix} + \left( \sum_{i=1}^{n+1} \epsilon_{i0} \right) \widetilde{v_0} v_{n+1}.$$

□

The reduction of Proposition 4 is polynomial but not completely tight: the algorithm $A^{mat}$ runs $n+1$ times the algorithm $A^{vec}$. Note that if instead of $A^{vec}$ we have an algorithm $A'^{vec}$ returning a message $m$ corresponding to a *cycle* for the vectorial version, then the message $m||m$ is a collision for the matrix version. Indeed, if $\left( \begin{smallmatrix} a & b \end{smallmatrix} \right) M = \left( \begin{smallmatrix} a & b \end{smallmatrix} \right)$ Proposition 1(a) shows that $M$ writes down as $M = \left( \begin{smallmatrix} a^{-1} & b \\ 0 & a \end{smallmatrix} \right) \left( \begin{smallmatrix} a & b \\ 0 & a^{-1} \end{smallmatrix} \right) + \epsilon \left( \begin{smallmatrix} b \\ a \end{smallmatrix} \right) \left( \begin{smallmatrix} a & b \end{smallmatrix} \right) = I + \epsilon \left( \begin{smallmatrix} b \\ a \end{smallmatrix} \right) \left( \begin{smallmatrix} a & b \end{smallmatrix} \right)$ hence $M^2 = I$.

## 8 Projective Version of Zémor-Tillich

Our second variant $h_{ZT}^{proj,\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right)}$ exploits even further Proposition 3. We define

$$h_{ZT}^{proj,\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right)} := [a : b]$$

where $\left( \begin{smallmatrix} a & b \end{smallmatrix} \right) := h_{ZT}^{vec,\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right)}(m)$ and $[a : b] \in \mathbb{P}^1(\mathbb{F}_{2^n})$. Finding a collision for $h_{ZT}^{proj,\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right)}$ is finding two messages $m$ and $m'$ such that $\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right) h_{ZT}(m) = \lambda \left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right) h_{ZT}(m')$ for some $\lambda$, in particular it is enough to find a *cyclic* collision which is a message $m$ such that $\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right)$ is a left eigenvector of $h_{ZT}(m)$.

The output of $h_{ZT}^{proj,\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right)}$ is very close to $n$ bits. For the parameters suggested by Tillich and Zémor, its collision resistance is equivalent to the collision resistance of the original function.

**Proposition 5** *If there exists an algorithm that finds collisions on $h_{ZT}^{proj,\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right)}$, there exists an algorithm that finds collisions on $h_{ZT}^{vec,\left( \begin{smallmatrix} a_0 & b_0 \end{smallmatrix} \right)}$, assuming that for*

*some $n' > n$ it is feasible to compute $n'$ discrete logarithms in $\mathbb{F}_{2^n}^*$ and one subset sum problem of size $n'$.*
*If we denote by $t^{proj}$, $t^{DL}$ and $t^{SS}(n')$ the times needed respectively to find collisions on the projective version, to solve one discrete logarithm problem in $\mathbb{F}_{2^n}^*$ and to solve a subset sum problem of size $n'$, collisions on the vectorial version can be found in time $n'(t^{proj} + t^{DL}) + t^{KN}(n')$.*

PROOF: Given an algorithm $A^{proj}$ finding collisions for the projective version, we build an algorithm $A^{proj}$ finding collisions for the vectorial version. Receiving an initial vector $v_0 = (a_0, b_0)$, $A^{vec}$ forwards it to $A^{proj}$ and receives two messages $m_{10}, m_{11}$. To the two messages correspond two vectors $(a_{10}, b_{10})$ and $(a_{11}, b_{11}) = \lambda_1(a_{10}, b_{10})$ for some $\lambda_1$. The algorithm $A^{vec}$ computes the discrete logarithm $d_1$ of $\lambda_1$ with respect to some generator $g$ of $\mathbb{F}_{2^n}^*$. The algorithm $A^{vec}$ then runs $A^{proj}$ on the projective point $(a_{10}, b_{10})$ and computes $d_2$ similarly, etc.

After $n'$ steps, the algorithm $A^{vec}$ computes a subset $I \subset \{1, ..., n'\}$ such that $\sum_{i \in I} d_i = 0 \bmod 2^n - 1$. By concatenating the paths $m_{ie_i}$ where $e_i = 1$ if $i \in I$ and $e_i = 0$ otherwise, algorithm $A^{vec}$ produces a collision with the message $m_{10}||...||m_{n'0}$ for the vectorial version. The output is correct because both messages lead to the vector $\left(\prod_{i \in I} \lambda_i\right)(a_{n'0}, b_{n'0}) = g^{\sum_{i \in I} d_i}(a_{n'0}, b_{n'0}) = (a_{n'0}, b_{n'0})$.

The claim on the running time follows straightforwardly. □

The best choice for $n'$ depends on the exact values of $t^{proj}$, $t^{DL}$ and $t^{SS}(n')$. Solving discrete logarithms problems is believed to be hard but is definitely feasible in $\mathbb{F}_{2^n}^*$ if $n < 170$. Computing $I \subset \{1, ..., n'\}$ such that $\sum_{i \in I} d_i = 0 \bmod 2^n - 1$ is related to the subset sum problem which is NP-hard but usually easy in average. For the parameters proposed by Zémor-Tillich, lattice reduction algorithms like LLL will probably succeed in performing the reduction. Another method is to use Wagner's "$k$-lists" algorithm [10] for solving the subset sum problem. This algorithm can solve the subset sum problem in time and space $k2^{n/(1+\log k)}$ which for $k \approx \sqrt{n}$ is roughly $2^{2\sqrt{n}}$ which is about $2^{26}$ for $n = 170$. The drawback with this method is that $n'$ must also increase to $2^{2\sqrt{n}}$ hence the discrete logarithm costs increase and the quality of the reduction decreases.

Assuming the existence of an algorithm $A'^{proj}$ computing *cyclic* collisions on the projective version (messages $m_i$ such that $(a_0, b_0)h^{mat}(m_i) = \lambda_i(a_0, b_0)$ for some $\lambda_i$) the reduction slightly improves. Indeed, $A^{vec}$ must only compute a *small integer* solution $(x_1, ..., x_{n'})$ to $\sum_i x_i d_i = 0 \bmod 2^n - 1$ instead of a *binary* solution. The reduction algorithm still has to compute discrete logarithm problems but it must not solve any subset sum problem.

## 9 Conclusion

We have given new algorithms for computing collisions for the Zémor-Tillich hash function in a time equal to the cubic root of the birthday bound. Our attacks are the first generic ones in the sense that unlike previous attacks they work for any parameters $n$ and $P_n(X)$ of the function. Moreover, they are very close to being practical for the parameters $n \in [130, 170]$ initially suggested in [9].

Interestingly, we could extend our collision attacks to new preimage attacks with the same complexity due to the inherent possibility of "meet-in-the-middle" attacks in Zémor-Tillich and the fact that our collision attacks use a subgroup structure that preserves this possibility.

Our attacks exploit a separation of the collision problem into an easy and a hard component, and suggest that the output of Zémor-Tillich should be of $n$ bits rather than $3n$ bits. We have consequently introduced two variants of this function, the vectorial and the projective versions, with reduced output sizes of respectively $2n$ and $n$ bits. We have proved that the original function is collision resistant if and only if the vectorial variant and ((for small $n$) if and only if the projective variant are collision resistant.

*Acknowledgements* We thank Nicolas Veyrat-Charvillon and Giacomo Demeulenaer for interesting discussions related to this paper. We thank Martijn Stam for a remark on the Zémor-Tillich hash function that motivated the algorithm of Section 5. We thank Phong Nguyen for pointing us out reference [10]. Finally, we would like to thank an anonymous referee of CT-RSA for his very useful comments that considerably improved the 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. 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.
3. 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.
4. H. W. J. L. L. Lenstra, A. K.; Lenstra. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(5):515–534, 1982.
5. C. Petit, N. Veyrat-Charvillon, and J.-J. Quisquater. Efficiency and Pseudo-Randomness of a Variant of Zémor-Tillich Hash Function. In *IEEE International Conference on Electronics, Circuits, and Systems, ICECS2008*, 2008.
6. J.-J. Quisquater and J.-P. Delescaille. How easy is collision search? application to des (extended summary). In *EUROCRYPT*, pages 429–434, 1989.
7. A. Shamir. Random graphs in cryptography. Invited talk at Asiacrypt 2006, 2006.
8. 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.
9. 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.
10. D. Wagner. A generalized birthday problem. In M. Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.

11. G. Yuval. How to swindle Rabin. *Cryptologia*, 3:187–189, 1979.

## A Graphical Interpretation of Proposition 3

The part of Proposition 3 concerning $\mathcal{L}^v$ and $\mathcal{R}^v$ has interesting graph interpretations. To the Zémor-Tillich hash function is associated a Cayley graph $\mathcal{ZT}$, in which each vertex corresponds to a matrix $M \in SL(2, \mathbb{F}_{2^n})$ and each edge to a couple $(M_1, M_2) \in SL(2, \mathbb{F}_{2^n})^2$ such that $M_2 = M_1 A_0$ or $M_2 = M_1 A_1$ [9].

We now construct the graphs $\mathcal{ZT}^{vec}$ and $\mathcal{ZT}^{proj}$ as follows. For $\mathcal{ZT}^{vec}$, associate a vertex to each row vector $\begin{pmatrix} a & b \end{pmatrix} \in \mathbb{F}_{2^n}^{1\times 2} \backslash \{\begin{pmatrix} 0 & 0 \end{pmatrix}\}$ and an edge to each couple of such vectors $(\begin{pmatrix} a_1 & b_1 \end{pmatrix}, \begin{pmatrix} a_2 & b_2 \end{pmatrix})$ satisfying $\begin{pmatrix} a_2 & b_2 \end{pmatrix} = \begin{pmatrix} a_1 & b_1 \end{pmatrix} A_0$ or $\begin{pmatrix} a_2 & b_2 \end{pmatrix} = \begin{pmatrix} a_1 & b_1 \end{pmatrix} A_1$. Alternatively, the graph $\mathcal{ZT}^{vec}$ can be constructed from the graph $\mathcal{ZT}$ by identifying two vertices $M_1 = \begin{pmatrix} a_1 & b_1 \\ c_1 & d_1 \end{pmatrix}$ and $M_2 = \begin{pmatrix} a_2 & b_2 \\ c_2 & d_2 \end{pmatrix}$ when $\begin{pmatrix} a_1 & b_1 \end{pmatrix} = \begin{pmatrix} a_2 & b_2 \end{pmatrix}$. An example of such a graph is shown in Figure 4.

Similarly, we associate a vertex of $\mathcal{ZT}^{proj}$ to each projective point $q_i = [a_i : b_i] \in \mathbb{P}^1(\mathbb{F}_{2^n})$ and an edge to each couple $(q_1, q_2)$ such that $\lambda \begin{pmatrix} a_2 & b_2 \end{pmatrix} = \begin{pmatrix} a_1 & b_1 \end{pmatrix} A_0$ or $\lambda \begin{pmatrix} a_2 & b_2 \end{pmatrix} = \begin{pmatrix} a_1 & b_1 \end{pmatrix} A_1$ for some $\lambda \in \mathbb{F}_{2^n}^*$. Alternatively, the graph $\mathcal{ZT}^{proj}$ may be constructed from the graph $\mathcal{ZT}^{vec}$ by identifying two vertices $\begin{pmatrix} a_1 & b_1 \end{pmatrix}$ and $\begin{pmatrix} a_2 & b_2 \end{pmatrix}$ when $\begin{pmatrix} a_1 & b_1 \end{pmatrix} = \lambda \begin{pmatrix} a_2 & b_2 \end{pmatrix}$ for some $\lambda \in \mathbb{F}_{2^n}^*$.

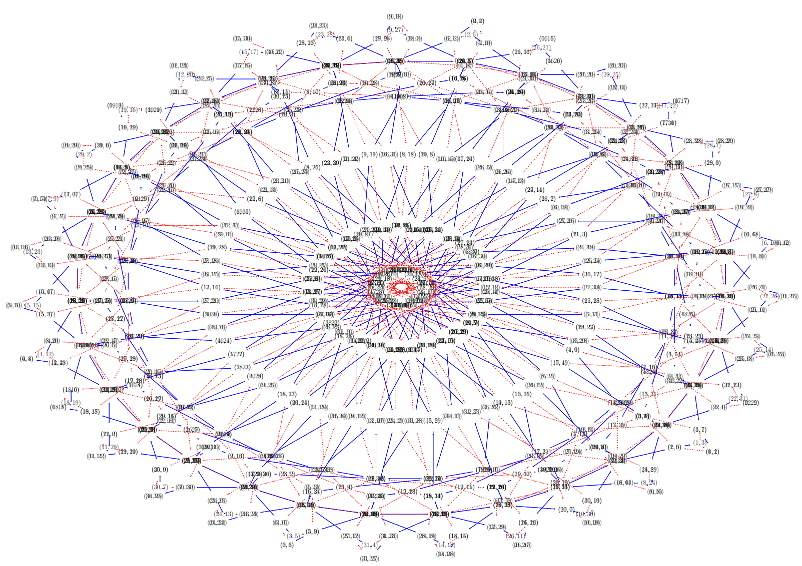

**Fig. 4.** $\mathcal{ZT}^{vec}$ graph for parameter $P_5(X) = X^5 + X^2 + 1$. The vertices are labeled by matrices. Red dotted (resp. blue solid) arrows correspond to multiplication by matrix $A_0$ (resp. $A_1$). Each polynomial $\sum a_i X^i$ is written as $\sum a_i 2^i$.

Finding a cycle in $\mathcal{ZT}^{vec}$ is just as hard as finding a cycle in $\mathcal{ZT}$ because if $\left(\begin{smallmatrix} a & b \end{smallmatrix}\right) M = \left(\begin{smallmatrix} a & b \end{smallmatrix}\right)$ then $M^2 = I$. The radial symmetry in the graph $\mathcal{ZT}^{vec}$ (Figure 4) is not surprising as it reflects the relation $\left(\begin{smallmatrix} a & b \end{smallmatrix}\right) A_i = \left(\begin{smallmatrix} a' & b' \end{smallmatrix}\right) \Leftrightarrow \left[\lambda \left(\begin{smallmatrix} a & b \end{smallmatrix}\right)\right] A_i = \left[\lambda \left(\begin{smallmatrix} a' & b' \end{smallmatrix}\right)\right]$: multiplying each vertex of $\mathcal{ZT}^{vec}$ by a constant $\lambda$ is equivalent to a rotation of the graph.

Roughly, a vertex in the graph $\mathcal{ZT}^{vec}$ can be characterized by a radial and an angular position. A cycle in the graph $\mathcal{ZT}^{proj}$ induces a path in the graph $\mathcal{ZT}^{vec}$ from a vertex to another vertex with the same radial coordinate, but not necessarily the same angular coordinate. Clearly, different such paths can be combined to give a cycle in the graph $\mathcal{ZT}^{vec}$. According to Proposition 3 and its proof, this can be done if the discrete logarithm problem, hence the representation problem, can be solved in $\mathbb{F}_{2^n}^*$.

A cycle in $\mathcal{ZT}^{vec}$ induces cycles in both radial and angular coordinates. Proposition 3 means that solving the angular part of the representation problem is easy once the radial part can be solved to produce various points with the same radius.