Noname manuscript No.
(will be inserted by the editor)

Towards factoring in $SL(2,\mathbb{F}_{2^n})$

Christophe Petit[†]

the date of receipt and acceptance should be inserted later

Abstract The security of many cryptographic protocols relies on the hardness of some computational problems. Besides discrete logarithm or integer factorization, other problems are regularly proposed as potential hard problems. The *factorization problem in finite groups* is one of them. Given a finite group G, a set of generators $\mathcal{S}$ for this group and an element $g \in G$, the factorization problem asks for a "short" representation of g as a product of the generators. The problem is related to a famous conjecture of Babai on the diameter of Cayley graphs. It is also motivated by the preimage security of *Cayley hash functions*, a particular kind of cryptographic hash functions. The problem has been solved for a few particular generator sets, but essentially nothing is known for generic generator sets.

In this paper, we make significant steps towards a solution of the factorization problem in the group $G := SL(2,\mathbb{F}_{2^n})$, a particularly interesting group for cryptographic applications. To avoid considering all generator sets separately, we first give a new reduction tool that allows focusing on some generator sets with a "nice" special structure. We then identify classes of *trapdoor matrices* for these special generator sets, such that the factorization of a single one of these matrices would allow efficiently factoring any element in the group. Finally, we provide a heuristic *subexponential* time algorithm that can compute *subexponential* length factorizations of *any* element for any pair of generators of $SL(2,\mathbb{F}_{2^n})$.

Our results do not yet completely remove the factorization problem in $SL(2,\mathbb{F}_{2^n})$ from the list of potential hard problems useful for cryptography. However, we believe that each one of our individual results is a significant step towards a polynomial time algorithm for factoring in $SL(2,\mathbb{F}_{2^n})$.

1 Introduction

Cryptographic protocols use computational assumptions as axioms in their security proofs. Typical assumptions include the hardness of factoring large integers or computing discrete logarithms in some groups, but other assumptions are regularly proposed [40,27,31]. In this paper, we investigate the hardness of the f*actorization problem in finite groups*, in particular in the group $SL(2,\mathbb{F}_{2^n})$.

1.1 Notations

We denote by $\mathbb{F}_{2^n}$ the field with 2^n elements. Let $G := SL(2,\mathbb{F}_{2^n})$ be the group of 2×2 matrices with elements in $\mathbb{F}_{2^n}$ and determinant 1. For any $A := \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in G$, we write A' for the *transpose matrix* of A that is $\left(\begin{smallmatrix} a & c \\ b & d \end{smallmatrix}\right)$. We write $t(A) := a+d$ for the *trace* of A and $s(A) := b+c$. We write I for the identity matrix $\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right) \in G$. For any non-zero λ, we write $\Lambda(\lambda) := \left(\begin{smallmatrix} \lambda & 0 \\ 0 & \lambda^{-1} \end{smallmatrix}\right)$. For any w, we write $O(w) := \left(\begin{smallmatrix} w+1 & w \\ w & w+1 \end{smallmatrix}\right)$. We

Postdoctoral[†] Research Fellow of the Belgian Fund for Scientific Research (F.R.S.-FNRS) at Université catholique de Louvain (UCL), crypto group.

UCL[†] Crypto GroupUniversité catholique de Louvain
Place du levant 3
1348 Louvain-la-Neuve, Belgium
E-mail: christophe.petit@uclouvain.be

will see that these matrices are the orthogonal matrices of $SL(2,\mathbb{F}_{2^n})$. For any t, we write $E(t) := \left(\begin{smallmatrix} t & 1 \\ 1 & 0 \end{smallmatrix}\right)$ and we call these matrices *Euclidean matrices*. For any $w_1,...,w_k \in \mathbb{F}_{2^n}$, we write $< w_1,...,w_k >_+$ for the group additively generated by $w_1,...w_k$. For any $x \in \mathbb{F}_{2^n}$, we write $\sqrt{x}$ for the (unique) y such that $y^2 = x$. For any $a,b \in \mathbb{F}_{2^n}$ with $(a,b) \neq (0,0)$, we write $[a : b]$ for the *projective point* that is the equivalence class of $\left(\begin{smallmatrix} a & b \end{smallmatrix}\right)$ with respect to the equivalence relation $\left(\begin{smallmatrix} a_1 & b_1 \end{smallmatrix}\right) \sim \left(\begin{smallmatrix} a_2 & b_2 \end{smallmatrix}\right) \Leftrightarrow a_1b_2 = a_2b_1$. For any $w \in \mathbb{F}_{2^n}$, the *algebraic degree* of w is the minimal degree of all polynomials p with coefficients in $\mathbb{F}_2$ satisfying $p(w) = 0$.

The value of n parameterizes the "size" of the problem. All our complexity estimates will be with respect to this value. For example, by a *polynomial length product* of a set $\mathcal{S} \subset G$ we mean a product of at most $p(n)$ elements of $\mathcal{S}$ for some polynomial p. In some estimates, we use $\mathcal{O}$ for the "big O" notation: given two functions f and g of n, we say that $f = \mathcal{O}(g)$ if there exist $N \in \mathbb{N}$ and $c \in \mathbb{Z}^+$ such that $n > N \Rightarrow f(n) \leq cg(n)$. When exact complexity estimates are given in the paper, they refer to the number of bit operations and not to the number of arithmetic operations in $\mathbb{F}_{2^n}$.

Given $\mathcal{S} := \{s_1,...,s_k\} \subset G$, we define *straight-line programs* of $\mathcal{S}$ as sequences of simple line codes of the form $M_i := s_i$, $i = 1,...,k$ followed by line codes of the form $M_i := M_jM_\ell$ where $j,\ell \leq i$ and $i = k+1,...,k'$. We call k' the *length* of this straight-line program. Straight-line programs allow returning some factorizations of very large lengths (non polynomial in n) in a compressed form that may have polynomial size in n [26].

1.2 Factoring in finite groups

The factorization problem in finite groups can be formulated as follows.

Problem 1 *Let G be a non-Abelian finite group and let $\mathcal{S} = \{s_1,...,s_k\}$ be a set of generators for this group. Let L be a positive integer. Find an algorithm that given any element $g \in G$, returns a word $m_1...m_\ell$ with $\ell \leq L$ and $m_i \in \{1,...,k\}$ such that*

$$\prod_{i=1}^{\ell} s_{m_i} = g.$$

In this paper, we focus on the group $G := SL(2,\mathbb{F}_{2^n})$. In a relaxation of Problem 1, we could also allow negative powers, meaning that we would be searching for factorizations of the form $\prod_{i=1}^{\ell} s_{m_i}^{e_i} = g$ where $e_i \in \{\pm 1\}$. This paper only considers the restrictive version.

The hardness of the problem clearly depends on L and $\log|G|$. Ideally, for a family of groups G with increasing size, we would like an algorithm running in time $T \leq p_1(\log|G|)$ and returning factorizations of length $L \leq p_2(\log|G|)$ where p_1, p_2 are polynomials. The mere *existence* of these *short* factorizations is not guaranteed *a priori*, but for simple groups it appears likely in the light of Babai's conjecture.

1.3 Babai's conjecture

A famous and long-standing conjecture of Babai [4,14] states that "short" factorizations (in the above sense) always exist for finite non-Abelian simple groups, whatever the choice of generator set. (Trivial counter-examples exist in the case of cyclic groups.) The original conjecture allowed negative powers, but a recent result of Babai [3] implies that the two versions of the conjecture are equivalent.

Babai's conjecture has recently drawn a lot of attention from the Mathematics community, after a breakthrough proof by Helfgott that the conjecture is true for the groups $SL(2,\mathbb{F}_p)$ when p is prime [14]. The conjecture (when allowing negative powers) has now been proved in large classes of groups including the groups $SL(2,\mathbb{F}_{2^n})$ considered in this paper [38,6].

Interestingly, Problem 1 can be seen as an algorithmic version of Babai's conjecture. The algorithmic version is *a priori* harder: it does not only require proving the *existence* of short factorizations, but also finding an algorithmic method to compute them. The algorithmic version was mentioned as an open problem in a book by Lubotzky (Problem 8.1.2 in [25]). As far as we know, it has only been solved in a few papers for *particular* generator sets [5,19,21,35,41,20]. We point out that most of the proofs of Babai's conjecture that apply to *any* generator set are non constructive [14,15,38,6]. Problem 1 is therefore a widely open problem, in particular for the group $SL(2,\mathbb{F}_{2^n})$.

1.4 Cayley hash functions

Problem 1 is also related to the security of a particular kind of cryptographic hash functions called *Cayley hash functions* [49,32]. More precisely, the *preimage resistance* [28] of these functions would be affected by a solution to Problem 1.

The design of Cayley hash functions was introduced by Zémor [48]. The parameters he originally proposed in $SL(2, \mathbb{F}_p)$ were quickly broken and replaced by Tillich and Zémor [45] for a new generator set in the group $SL(2, \mathbb{F}_{2^n})$. The function drew the attention of the cryptography community for its elegant design and its appealing properties, in particular its natural parallelism. Some weaknesses were identified for subsets of the parameters [10,12,1,43] but no serious attack had been found for many years. The design was rediscovered more than ten years later [9,34] but the new parameters proposed were quickly broken [46,33]. Finally, the factorization problem for the Tillich-Zémor hash function was broken as well [13,35].

Therefore, all the hash functions proposed following the Cayley hash function design are now broken. However, as pointed out in [46,33,35,36], all these functions used parameters that were specially chosen for efficiency. Considering the very nice properties of Cayley hash functions, their security for generic parameters is a very interesting open problem.

This paper focuses on the case $G = SL(2, \mathbb{F}_{2^n})$. This group seems to be the most interesting one for cryptographic applications. First of all, a Cayley hash function built from an Abelian group would not be *collision resistant*, another very important requirement for hash functions. Likewise, the security of general linear groups $GL(m, K) \approx SL(m, K) \times \mathbb{F}_{2^n}^*$ would be similar to the security of their corresponding special linear groups $SL(m, K)$. Finally, taking $m = 2$ and $K = \mathbb{F}_{2^n}$ makes the group law more efficient in both software and hardware implementations, which is also very important for cryptographic applications.

1.5 Previous work on $SL(2, \mathbb{F}_{2^n})$

The factorization problem in $SL(2, \mathbb{F}_{2^n})$ was solved in [5] for appropriately chosen generator sets $\mathcal{S}$ containing 3 elements. It has also been recently solved by Petit and Quisquater [35] for the particular generators $A := \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix}, B := \begin{pmatrix} X & X+1 \\ 1 & 1 \end{pmatrix}$ that correspond to the Tillich-Zémor hash function [45]. In this paper, we focus on the two generators case, where $\mathcal{S} := \{A, B\}$ for two matrices $A, B \in SL(2, \mathbb{F}_{2^n})$ that generate the group. In this case, a heuristic attack for *generic* generator sets was given in [37]. The attack runs in time roughly $2^{n/2}$ and it produces factorizations of length roughly n^3.

The algorithm of Petit and Quisquater [35] first replaces the original Tillich-Zémor generators by the following slightly simpler ones $\tilde{A} := \begin{pmatrix} X & 1 \\ 1 & 0 \end{pmatrix}, \tilde{B} := \begin{pmatrix} X+1 & 1 \\ 1 & 0 \end{pmatrix}$. Interestingly, matrices of the form $\begin{pmatrix} t_i & 1 \\ 1 & 0 \end{pmatrix}$ can be linked to the famous Euclidean algorithm [13]. The reduction from one set of generators to the other one is obtained by *conjugating* both generators by A. In a second step, the algorithm uses a factorization of one matrix M of the form $\begin{pmatrix} 0 & b \\ c & d \end{pmatrix}$ to compute a factorization of any matrix in the group. The factorization of M as a product of $\tilde{A}$ and $\tilde{B}$ was provided by Grassl et al. using an algorithm of Mesirov and Sweet [13,29].

Petit and Quisquater's algorithm seems very specific to the generators of the Tillich-Zémor hash function. The connection to the Euclidean algorithm requires generators of the form $\begin{pmatrix} t_i & 1 \\ 1 & 0 \end{pmatrix}$ and Mesirov and Sweet's algorithm can only be used for the matrices $\tilde{A}$ and $\tilde{B}$. Using the results of Lauder [22], the whole attack can be extended to a finite number of generator sets [36]. However, the hardness of the factorization problem is a widely open problem for generic (random) generator sets, even if the generators are constrained to belong to $\left\{ \begin{pmatrix} t_i & 1 \\ 1 & 0 \end{pmatrix} | t_i \in \mathbb{F}_{2^n} \right\}$.

1.6 Contributions of this paper

In this paper, we follow a three-step strategy for factoring in $G = SL(2, \mathbb{F}_{2^n})$:

1. We reduce generic generator sets to other simpler ones.
2. For some of these simpler generator sets, we identify classes of *trapdoor matrices*, such that factoring only one of these matrices would allow factoring any matrix in the group.
3. We factor one trapdoor matrix.

This approach is inspired by Petit and Quisquater's algorithm for the Tillich-Zémor parameters [35], that used a "trapdoor matrix" previously found by Grassl et al. [13].

The main contributions of our paper are a series of techniques to perform Steps 1 and 2 of the above program. Whereas there could be about 2^{6n} different pairs of generators to consider at first sight, our reductions allow focusing on some classes of about 2^n "simpler" generator sets, including symmetric matrices, diagonal matrices $\Lambda(\lambda)$, orthogonal matrices $O(w)$ or Euclidean matrices $E(t)$. For a *symmetric* generator set $\{A, A'\}$, we then show that the knowledge of a factorization of *any* single matrix with $\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$ as eigenvector allows factoring efficiently any matrix in the group. We also provide sufficient conditions for a reduction to Tillich-Zémor generators.

Using these reduction tools, we finally propose two new heuristic algorithms solving the factorization problem for *generic* generator sets of $SL(2, \mathbb{F}_{2^n})$. The second algorithm computes factorizations of *subexponential* lengths in *subexponential* time complexity (both complexity measures are taken with respect to the parameter n). Previous algorithms either focused on particular generator sets, had an exponential time complexity or produced exponential length factorizations.

1.7 Roadmap

The remaining of this paper is organized according to our three-step strategy for factoring in $SL(2, \mathbb{F}_{2^n})$. Section 2 provides reduction tools between various generator sets; Section 3 proves our results on trapdoor matrices; Section 4 describes and analyzes our new factorization algorithms; Section 5 concludes the paper.

2 Changing the generators

Since almost all pairs of elements in $SL(2, \mathbb{F}_{2^n})$ generate the whole group [24], we focus on generator sets $\mathcal{S} := \{A, B\}$ containing only two elements. There are about $|SL(2, \mathbb{F}_{2^n})|^2 \approx 2^{6n}$ such generator sets. Instead of solving a factorization problem for each generator set separately, we propose to reduce generic sets to other "simpler" ones.

At first sight, three simple reduction tools may naturally come to mind. First, we can use *substitutions*: clearly if we know a short factorization of A_2 and B_2 as products of A and B, then the factorization problem for the set $\{A, B\}$ reduces to the factorization problem for the set $\{A_2, B_2\}$. Second, we can use *field isomorphisms*: if A_2 and B_2 are the images of A and B through a field isomorphism of $\mathbb{F}_{2^n}$, then any product of A and B straightforwardly transforms into a product of A_2 and B_2 through the same isomorphism. Third, we can use *conjugations* as in [13,35]: for any $S \in GL(2, \mathbb{F}_{2^n})$, a factorization of $S^{-1}gS$ as a product of $S^{-1}AS$ and $S^{-1}BS$ directly leads to a factorization of g as a product of A and B.

In this section, we introduce new, more elaborate reduction tools from generic generator sets to some "small" classes of generators, parameterized by a single parameter. In Section 2.1, we show that $\mathcal{S}$ can be replaced by a *symmetric* generator set $\tilde{\mathcal{S}} := \{\tilde{A}, \tilde{A}'\}$ as long as AB and BA generate the group. In Sections 2.2 and 2.3, we show that this new set can often in turn be replaced by a new set containing a symmetric matrix and orthogonal matrices.

2.1 Symmetric generators

We first replace the generator set $\mathcal{S}$ by another set $\tilde{\mathcal{S}} = \{\tilde{A}, \tilde{B}\}$ where $\tilde{A} = \tilde{B}'$. For any matrices A, B, the matrices AB and BA have the same trace.

Lemma 1 *Let $A, B \in G$. Then $t(AB) = t(BA)$.*

Given two matrices with the same trace that generate the whole group, we can conjugate these two matrices to obtain two new matrices $\tilde{A}$ and $\tilde{B}$ such that $\tilde{A} = \tilde{B}'$.

Lemma 2 *Let A, B generate G such that $t(A) = t(B)$. Then there exists $S \in GL(2, \mathbb{F}_{2^n})$ such that $(S^{-1}AS)' = S^{-1}BS$.*

PROOF: Let $a_1, b_1, c_1, a_2, b_2, c_2, t \in \mathbb{F}_{2^n}$ such that $A = \left(\begin{smallmatrix} a_1 & b_1 \\ c_1 & a_1+t \end{smallmatrix}\right)$ and $B = \left(\begin{smallmatrix} a_2 & b_2 \\ c_2 & a_2+t \end{smallmatrix}\right)$. For any $S = \left(\begin{smallmatrix} w & x \\ y & z \end{smallmatrix}\right)$ with $wz + xy = 1$, we have

$$\begin{pmatrix} z & x \\ y & w \end{pmatrix} \begin{pmatrix} a_i & b_i \\ c_i & a_i + t \end{pmatrix} \begin{pmatrix} w & x \\ y & z \end{pmatrix} = \begin{pmatrix} c_iwx + a_iwz + a_ixy + txy + b_iyz & c_ix^2 + txz + b_iz^2 \\ c_iw^2 + twy + b_iy^2 & c_iwx + a_iwz + twz + a_ixy + b_iyz \end{pmatrix}.$$

The conditions $(S^{-1}AS)' = S^{-1}BS$ and $\det(S) = 1$ lead to the algebraic system

$$\begin{cases} (c_1 + c_2)wx + (a_1 + a_2)(wz + xy) + (b_1 + b_2)yz = 0 \\ c_2w^2 + twy + c_1x^2 + txz + b_2y^2 + b_1z^2 = 0 \\ c_1w^2 + twy + c_2x^2 + txz + b_1y^2 + b_2z^2 = 0 \\ wz + xy = 1 \end{cases}$$

or

$$\begin{cases} (c_1 + c_2)wx + (b_1 + b_2)yz = (a_1 + a_2) \\ (c_1 + c_2)(w^2 + x^2) + (b_1 + b_2)(y^2 + z^2) = 0 \\ c_1w^2 + twy + c_2x^2 + txz + b_1y^2 + b_2z^2 = 0 \\ wz + xy = 1 \end{cases}$$

If $b_1 + b_2 \neq 0$, we first conjugate both A and B by

$$S_1 := \begin{pmatrix} 1 & 1 \\ 1 + \sqrt{\frac{c_1+c_2}{b_1+b_2}} & \sqrt{\frac{c_1+c_2}{b_1+b_2}} \end{pmatrix}.$$

This way we obtain two new matrices $\hat{A} = \left(\begin{smallmatrix} \hat{a}_1 & \hat{b}_1 \\ \hat{c}_1 & \hat{a}_1+t \end{smallmatrix}\right)$ and $\hat{B} = \left(\begin{smallmatrix} \hat{a}_2 & \hat{b}_2 \\ \hat{c}_2 & \hat{a}_2+t \end{smallmatrix}\right)$ such that $\hat{b}_1 = \hat{b}_2$ and $\hat{c}_1 + \hat{c}_2 = b_1 + b_2 \neq 0$.

Let us now assume $b_1 = b_2$. We have $c_1 \neq c_2$, since otherwise we would have either $a_1 = a_2$ or $a_1 = a_2 + t$. This would imply either $B = A$ or $B = A^{-1}$, contradicting the fact that A, B generate $SL(2, \mathbb{F}_{2^n})$. Let $b := b_1 = b_2$. We have $b \neq 0$ since otherwise the two matrices are lower triangular and they do not generate the whole group. Since $1 = a_i(a_i + t) + bc_i$, this also implies $a_1 + a_2 \neq 0$. The previous system becomes

$$\begin{cases} (c_1 + c_2)wx = (a_1 + a_2) \\ w^2 + x^2 = 0 \\ c_1w^2 + twy + c_2x^2 + txz + b(y^2 + z^2) = 0 \\ wz + xy = 1 \end{cases}.$$

The first two equations are satisfied if $w = x = \sqrt{\frac{a_1+a_2}{c_1+c_2}}$. The last equation is satisfied if $z = y + \sqrt{\frac{c_1+c_2}{a_1+a_2}}$. Finally, a small computation using $a_i^2 + a_it + bc_i = 1$ shows that the third equation is then trivially satisfied. □

From now on, any generator set $\mathcal{S} = \{A, B\}$ of $SL(2, \mathbb{F}_{2^n})$ such that $A = B'$ will be called a *symmetric generator set*. Let $\mathcal{S} := \{A, B\}$ be an arbitrary generator set for G. If the matrices AB and BA generate the whole group, then Lemma 1 and Lemma 2 can be used to reduce the original factorization problem to a new factorization problem with a symmetric set of generators.

Proposition 3 *Let $\mathcal{S} = \{A, B\} \subset SL(2, \mathbb{F}_{2^n})$ such that $\{AB, BA\}$ is a generator set of $SL(2, \mathbb{F}_{2^n})$. Then the factorization problem in $SL(2, \mathbb{F}_{2^n})$ for $\mathcal{S} = \{A, B\}$ is reducible to another factorization problem in $SL(2, \mathbb{F}_{2^n})$ with a symmetric set of generators. The reduction increases the factorization lengths by a factor 2.*

PROOF: Let S be a matrix such that $(S^{-1}ABS)' = S^{-1}BAS$. The proof of Lemma 2 provides a polynomial time construction for S: the most costly part is the computation of square roots in $\mathbb{F}_{2^n}$ which just takes cubic time through a modular exponentiation. Let $\tilde{A} := S^{-1}ABS$ and $\tilde{B} := S^{-1}BAS$. Suppose we have a factorization algorithm for $\tilde{\mathcal{S}} := \{\tilde{A}, \tilde{B}\}$. Given $g \in G$, we apply this algorithm to $S^{-1}gS$ and get a factorization $\tilde{P}$. We then construct a new product P of A and B by replacing any occurrence of $\tilde{A}$ and

$\tilde{B}$ in $\tilde{P}$ by respectively AB and BA. We finally return this factorization P. Its length is exactly twice the length of the factorization returned by the factorization algorithm for $\tilde{A}$ and $\tilde{B}$. $\square$

We remark that the condition "$\{AB, BA\}$ is a generator set of $SL(2, \mathbb{F}_{2^n})$" is required to apply Lemma 2 in the proof of Proposition 3. Indeed, the set $\{AB, BA\}$ does not necessarily generate the whole group even if the set $\{A, B\}$ does.

2.2 Orthogonal matrices

In this section, we study the orthogonal subgroup of G.

Lemma 4 *The set of orthogonal matrices of $SL(2, \mathbb{F}_{2^n})$ is the set*

$$\left\{O(w) := \left(\begin{smallmatrix} w+1 & w \\ w & w+1 \end{smallmatrix}\right) | w \in \mathbb{F}_{2^n}\right\}.$$

Proof: Clearly these matrices are orthogonal. On the other hand, let $w, x, y, z \in \mathbb{F}_{2^n}$ such that $wz+xy = 1$ and

$$\left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right) = \left(\begin{smallmatrix} w & x \\ y & z \end{smallmatrix}\right)\left(\begin{smallmatrix} w & y \\ x & z \end{smallmatrix}\right) = \left(\begin{smallmatrix} w^2+x^2 & wy+xz \\ wy+xz & y^2+z^2 \end{smallmatrix}\right).$$

We deduce $w^2 + x^2 = 1$ hence $w = x + 1$. Similarly, we obtain $y = z + 1$ and finally $1 = wz + xy = xz + z + xz + x = x + z$. $\square$

Orthogonal matrices form an Abelian subgroup of $SL(2, \mathbb{F}_{2^n})$.

Lemma 5 *For any $w_1, w_2 \in \mathbb{F}_{2^n}$, we have $O(w_1)O(w_2) = O(w_1+w_2)$. In particular, for any $w \in \mathbb{F}_{2^n}$ we have $O(w)^2 = I$. For any $w_1, ..., w_k \in \mathbb{F}_{2^n}$ the set of matrices multiplicatively generated by $\{O(w_i)|1 \leq i \leq k\}$ is an Abelian subgroup of G. If $w_1, \ldots, w_k$ are linearly independent over $\mathbb{F}_2$, this group is isomorphic to the vector space $\mathbb{F}_2^k$.*

Conjugations by orthogonal matrices preserve both the trace and the sum of the off-main diagonal elements. In fact, the converse is almost true as well.

Lemma 6 *Let $M_i = \left(\begin{smallmatrix} a_i & b_i \\ c_i & d_i \end{smallmatrix}\right) \in G$, $i = 1, 2$. For $i = 1, 2$, let $t_i := t(M_i) = a_i+d_i$ and $s_i := s(M_i) = b_i+c_i$. If M_1 and M_2 are conjugate by an orthogonal matrix, then $t_1 = t_2$ and $s_1 = s_2$. Conversely,*

1. *If $t_1 = t_2 \neq s_1 = s_2$, then M_1 and M_2 are conjugate by $O(w)$ where $w := \frac{a_1+a_2+b_1+b_2}{s_1+t_1}$.*
2. *If $t_1 = t_2 = s_1 = s_2 \neq 0$ then either $a_1 + a_2 = b_1 + b_2$ or $a_1 + a_2 = b_1 + b_2 + t$.*
   (a) In the first case, then M_1 and M_2 are conjugate by $O(w)$ where $w := \frac{a_1+a_2}{t}$.
   (b) In the second case, M_1 and M_2 are not conjugate by any orthogonal matrix unless $M_1 = M_2$.
3. *If $t_1 = t_2 = s_1 = s_2 = 0$, then M_1 and M_2 are orthogonal matrices. They are not conjugate by any orthogonal matrix unless $M_1 = M_2$.*

Proof: M_1 and M_2 are conjugate by $O(w)$ if and only if $O(w)M_2 = M_1O(w)$, that is

$$\left(\begin{smallmatrix} a_2w+c_2w+a_2 & b_2w+d_2w+b_2 \\ a_2w+c_2w+c_2 & b_2w+d_2w+d_2 \end{smallmatrix}\right) = \left(\begin{smallmatrix} a_1w+b_1w+a_1 & a_1w+b_1w+b_1 \\ c_1w+d_1w+c_1 & c_1w+d_1w+d_1 \end{smallmatrix}\right).$$

Comparing the traces of the left and right-hand sides we obtain

$$(t_2 + s_2)w + t_2 = (t_1 + s_1)w + t_1.$$

Similarly by comparing the sum of the non diagonal terms we get

$$(t_2 + s_2)w + s_2 = (t_1 + s_1)w + s_1.$$

Together, the two equations imply $s_1 = s_2$ and $t_1 = t_2$ which proves the first part of the lemma.

Conversely, suppose that $s := s_1 = s_2$ and $t := t_1 = t_2$. The above matrix equation is equivalent to

$$\begin{cases} (a_1 + a_2 + b_1 + b_2 + t)w = b_1 + b_2, \\ (a_1 + a_2 + b_1 + b_2 + s)w = a_1 + a_2. \end{cases}$$

If $s \neq t$, let $w := \frac{a_1+a_2+b_1+b_2}{s+t}$. Using $\det(M_i) = a_i^2 + b_i^2 + ta_i + sb_i = 1$, , we have

$$\begin{aligned}
&(a_1 + a_2 + b_1 + b_2 + t)w + b_1 + b_2 \\
&\quad = (s+t)^{-1}\left[(a_1 + a_2 + b_1 + b_2 + t)(a_1 + a_2 + b_1 + b_2) + (b_1 + b_2)(s+t)\right] \\
&\quad = (s+t)^{-1}\left[a_1^2 + a_2^2 + b_1^2 + b_2^2 + s(b_1 + b_2) + t(a_1 + a_2)\right] \\
&\quad = (s+t)^{-1}\left[1+1\right] = 0.
\end{aligned}$$

and similarly, $(a_1 + a_2 + b_1 + b_2 + s)w + a_1 + a_2 = 0$.

If $s = t$, then the determinant condition restricts the value of $a + b$ to the solutions of the quadratic equation

$$\det\left(\begin{smallmatrix} a_i & b_i \\ b_i+t & a_i+t \end{smallmatrix}\right) = (a_i + b_i)(a_i + b_i + t) = 1$$

hence either $a_1 + a_2 = b_1 + b_2$ or $a_1 + a_2 = b_1 + b_2 + t$. In the first case, both equations are equivalent to $tw = a_1 + a_2$. In the second case, they give $a_1 + a_2 = b_1 + b_2 = 0$.

Finally, if $s = t = 0$, the determinant condition implies $(a_i + b_i)^2 = 1$, hence $a_i = b_i + 1$. $\square$

A first consequence of this lemma is that any symmetric matrix with non-zero trace can be diagonalized by an orthogonal matrix.

Lemma 7 *Let $M = \left(\begin{smallmatrix} a & b \\ b & d \end{smallmatrix}\right) \in SL(2, \mathbb{F}_{2^n})$ and let $t := a + d \neq 0$. There exist $\lambda, w \in \mathbb{F}_{2^{2n}}$ such that $M = O(w)\Lambda(\lambda)O(w)$. We have $\lambda, w \in \mathbb{F}_{2^n}$ if and only if the roots of $\alpha^2 + t\alpha + 1$ belong to $\mathbb{F}_{2^n}$.*

Proof: By hypothesis, we have $s(M) = s(\Lambda(\lambda)) = 0$ and $t \neq 0$. By Lemma 6, M and $\Lambda(\lambda)$ are conjugate by $O(w)$ if and only if $\lambda + \lambda^{-1} = t$ hence $\lambda^2 + t\lambda + 1 = 0$. Moreover, we have $w = \frac{a+b+\lambda}{t}$. $\square$

The s and t values of a matrix are invariant under transposition. In particular, two matrices that are transpose of each other and verify $t \neq s$ are conjugate of each other by an orthogonal matrix.

Lemma 8 *Let $A := \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right) \in G$ and let $A' := \left(\begin{smallmatrix} a & c \\ b & d \end{smallmatrix}\right)$. Let $t := a + d$ and $s := b + c$. Let $A_1 := A$ and for any $i > 1$, let $A_i := A_{i-1}A_{i-1}A'_{i-1}$. Let $s_i := s(A_i)$ and $t_i := t(A_i)$. We have $s_i = s$ for any $i \geq 1$, $t_1 = t$ and $t_i = t_{i-1}^3 + t_{i-1}s^2 + t_{i-1}$ for any $i > 1$. For any $i > 0$ such that $t_i + s \neq 0$, let $w_i := \frac{s}{t_i+s}$. For these i indices, we have $A_i = O(w_i)A'_iO(w_i)$.*

Proof: For $i = 1$, we have $s(A_1) = s(A'_1) = s$ and $t(A_1) = t(A'_1) = t$. Applying Lemma 6, A_1 and A'_1 are conjugate by an orthogonal matrix with $w_1 = \frac{a+b+a+c}{s+t} = \frac{s}{s+t}$. For $i = 2$, we have

$$A_2 := A_1A_1A'_1 = \begin{pmatrix} a^3+ab^2+abc+b^2d & a^2c+abd+bc^2+bd^2 \\ a^2c+acd+b^2c+bd^2 & ac^2+bcd+c^2d+d^3 \end{pmatrix}.$$

We have $s_2 = (ad + bc)(b + c) = s_1 = s$ and $t_2 = a^3 + d^3 + (a + d)(b^2 + c^2) + bc(a + d) = (a + d)((b^2 + c^2) + a^2 + d^2 + ad + bc) = t_1(t_1^2 + s^2 + 1)$. As in the case $i = 1$, we deduce that A_2 and A'_2 are conjugate by an orthogonal matrix with $w_2 = \frac{s_2}{s_2+t_2} = \frac{s}{(s+t)(t^2+ts+1)}$. The remaining cases follow by induction. $\square$

2.3 Heuristic reductions to simpler generator sets with orthogonal matrices

In this section, we replace an initial symmetric generator set $\mathcal{S} := \{A, A'\}$ by new sets including "simpler" matrices. Let n_1 be a positive integer smaller than n. Let us consider the set

$$\tilde{\mathcal{S}} := \{A_{n_1}, A'_{n_1}\} \cup \{O(w_i) | 1 \leq i \leq n_1\}$$

where the matrices A_{n_1}, $O(w_i)$ are as defined in Lemma 8. In any product

$$P := \prod_{j=1}^{L}\left(\left(\prod_{i=1}^{n_1} O(w_i)^{e_{j,i}}\right) A_{n_1}^{e_{j,a}}(A'_{n_1})^{e_{j,b}}\right) \tag{1}$$

where $e_{j,i}, e_{j,a}, e_{j,b} \in \{0, 1\}$, the matrices A_{n_1} and A'_{n_1} can be substituted by their corresponding products of A and A'. The first step towards removing the orthogonal matrices in the product (1) is the following lemma.

Lemma 9 *Let $\mathcal{S} := \{A, A'\}$ be a generator set of $SL(2, \mathbb{F}_{2^n})$. Let s, t, A_i, s_i, t_i, w_i as in Lemma 8. Let $n_1 \in \mathbb{N}$ such that $t_i \neq s$ for all $i \leq n_1$. Then any product (1) can be reduced by the relations $A_i = O(w_i)A'_iO(w_i)$ and $O(w_i)^2 = I$ into an equivalent product*

$$\tilde{P} := Q\prod_{i=1}^{n_1} O(w_i)^{e_i} \tag{2}$$

where Q is a product of A, A' only and $e_i \in \{0, 1\}$. The product Q has length at most $2{\cdot}3^{n_1-1}{\cdot}L$. Moreover, $\tilde{P}$ can be written with a straight-line program of length at most $6(n_1-1)L+n_1$. The product (2) is unique if $w_1, \ldots, w_{n_1}$ are linearly independent over $\mathbb{F}_{2^n}$.

Proof: For any $I \subset \{1, \ldots, n_1\}$, we have $\left(\prod_{i\in I} O(w_i)\right) A_{n_1} = B_{n_1,I} \left(\prod_{i\in I} O(w_i)\right)$ where

$$B_{1,I} := \begin{cases} A_1 \text{ if } 1 \notin I \\ A'_1 \text{ if } 1 \in I \end{cases}$$

and for $i = 2, ..., n_1$,

$$B_{i,I} := \begin{cases} B_{i-1,I}B_{i-1,I}B'_{i-1,I} \text{ if } i \notin I \\ B_{i-1,I}B'_{i-1,I}B'_{i-1,I} \text{ if } i \in I. \end{cases}$$

Similarly for any $I \subset \{1, \ldots, n_1\}$, we have

$$\left(\prod_{i\in I} O(w_i)\right) A'_{n_1} = \left(\prod_{i\in I} O(w_i)\right) O(w_{n_1})A_{n_1}O(w_{n_1}) = \left(\prod_{i\in I'} O(w_i)\right) A_{n_1}O(w_{n_1})$$
$$= B_{n_1,I'}\left(\prod_{i\in I'} O(w_i)\right) O(w_{n_1}) = B_{n_1,I'}\left(\prod_{i\in I} O(w_i)\right)$$

where I' is the unique subset of $\{1, \ldots, n_1\}$ such that $n_1 \in I \Leftrightarrow n_1 \notin I'$ and $i \in I \Leftrightarrow i \in I'$ for any $i < n_1$.

Using these relations and the identities $O(w_i)^2 = I$, we can succesively move all the orthogonal matrices in (1) from left to right. Recursively, we see that the matrices $B_{i,I}$ are products of $\{A, A'\}$ of length 3^{i-1}. Moreover, both $B_{i,I}$ and $B_{i,I}$ can be constructed from $B_{i-1,I}$ and $B_{i-1,I}$ with a straight-line program of length 3. If the w_i are linearly independent, then the product (2) is unique since every orthogonal matrix corresponds to a unique subset $I \subset \{1, \ldots, n_1\}$. $\square$

We remark that if the values t, s are randomly distributed and if n_1 is small enough compared to n, the probability that $t_i = s$ for some $i \leq n_1$ tends to 0 as n tends to infinity. Indeed from Lemma 8, any t_i can be written as a polynomial function of s and t of degree 3^{i-1} in t. For any given s, the probability that $t_i = s$ is very small as long as n_1 is small enough compared to n, more precisely as long as $(n_1-1)\log_2 3$ is small enough compared to n. Note that in the unlikely event where $t_i = s$ for some $i \leq n_1$, we obtain a matrix $M := A_i$ satisfying $s(M) = t(M)$ and $s(M) = s \neq 0$. The factorization problem can then be solved as in Proposition 11 below.

Likewise, the probability that the w_i are not linearly independent is very close to 1 if n_1 is small enough compared to n. Indeed if the w_i are not linearly independent, then there exist $I_1, I_2 \subset \{1, \ldots, n_1\}$ such that $\prod_{i\in I_1} O(w_i) = \prod_{i\in I_2} O(w_i)$. Following the proof of Lemma 9, we deduce a *collision*

$$B_{n_1,I_1}\left(\prod_{i\in I_1} O(w_i)\right) = B_{n_1,I_2}\left(\prod_{i\in I_1} O(w_i)\right)$$

where $B_{n_1,I_1} \neq B_{n_1,I_2}$. Equivalently, $B_{n_1,I_1} = B_{n_1,I_2}O$ for some $O \neq I$ generated by $\{O(w_i)|i < n_1\}$. Such a collision seems unlikely to occur for a random generator set $\mathcal{S}$ as long as n_1 is small enough compared to n, more precisely as long as $n_1 + 2 + 2n_1\log_2 3$ is small enough compared to $3n$.

We now provide a heuristic algorithm to reduce the factorization problem for the set $\mathcal{S} := \{A, A'\}$ to the factorization problem for the set $\tilde{\mathcal{S}} := \{A_{n_1}, A'_{n_1}\} \cup \{O(w_i)|1 \leq i \leq n_1\}$ where the matrices A_{n_1}, $O(w_i)$ are as defined in Lemma 8. (For the sake of simplicity, we assume that the w_i are linearly independent, but the general case can be treated similarly). Let $\mathbf{A}$ be an algorithm solving the factorization problem for $\tilde{\mathcal{S}} := \{A_{n_1}, A'_{n_1}\} \cup \{O(w_i)|1 \leq i \leq n_1\}$. Let R be a positive integer to be fixed later. Given an element $g \in SL(2, \mathbb{F}_{2^n})$, our algorithm takes the following steps.

1. Create two empty lists $\mathcal{L}_w$ and $\mathcal{L}_{fac}$ to contain elements $w_{(j)} \in \mathbb{F}_{2^n}$ and the factorizations of $O(w_j)$ as products of A and A'. Let $m := 0$ and $j := 1$.
2. Generate a product $r := \prod_{i=1}^{R} (x_i A + (1-x_i)A')$ where $x_i \in \{0,1\}$ are randomly chosen.
3. Apply $\mathbf{A}$ to $h := r^{-1}g$ and use Lemma 9 to write the result in the form $h = QO(w)$ where $w \in < w_1, \dots, w_{n_1} >_+$.
4. Generate a product $r_j := \prod_{i=1}^{R} (x_{ij} A + (1-x_{ij})A')$ where $x_{ij} \in \{0,1\}$ are randomly chosen.
5. Apply $\mathbf{A}$ to $h_j := r_j^{-1}$ and use Lemma 9 to write the result in the form $h_j = Q_j O(w_{(j)})$ where $w_{(j)} \in < w_1, \dots, w_{n_1} >_+$.
6. If $w_{(j)} \notin < w_{(1)}, \dots, w_{(m)} >_+$, add $w_{(j)}$ and the factorization $O(w_{(j)}) = r_j Q_j$ to the lists $\mathcal{L}_w$ and $\mathcal{L}_{fac}$. Increment m and j by 1.
7. Let $V := < w_{(1)}, \dots, w_{(m)} >_+$ be the vector space generated by the elements of $\mathcal{L}_w$.
8. If $w \notin V$, go back to Step 2.
9. If $w \in V$, use linear algebra to write $w = \sum_{j=1}^{m} e_j w_{(j)}$ with $e_j \in \{0,1\}$.
10. Return the factorization $g = Q \prod_{j=1}^{m} (r_j Q_j)^{e_j}$.

Clearly, the algorithm only returns correct answers when it terminates. We now argue that Steps 2 to 8 will only be repeated a bit more than n_1 times on average. If elements r, r_j generated in Steps 2 and 4 were uniformly distributed in $SL(2, \mathbb{F}_{2^n})$, then elements h_j in Step 5 would also be uniformly distributed and the orthogonal matrices produced in Steps 3 and 5 would have the same distribution. This last distribution would depend on $\mathbf{A}$. If it was uniform in $\{O(w) | w \in < w_1, \dots, w_{n_1} >_+\}$, the probability that the condition $w \in < w_{(1)}, \dots, w_{(m)} >_+$ is not satisfied in Step 9 would exponentially decrease when m increases. After $n_1 + 10$ repetitions of Steps 2 to 8, the vector space V would be $< w_1, \dots, w_{n_1} >_+$ with a probability higher than 99%. Moreover if the distribution of the orthogonal matrices was not uniform, the condition in Step 8 would actually only be satisfied sooner.

In the algorithm, matrices r, r_j are not uniformly generated but are generated by *random walks* in Step 2 and 4. It is known that random walks quickly converge to the uniform distribution on undirected expander graphs and it has also been observed that Cayley graphs tend to be good expanders [16]. The analysis of our algorithm relies on the heuristic assumption that random walks quickly converge in directed Cayley graphs of $SL(2, \mathbb{F}_{2^n})$, an assumption supported by the fact that Babai's conjecture has been proved for $SL(2, \mathbb{F}_{2^n})$. In fact, the analysis does not require that elements r, r_i are uniformly distributed, but only that the vector spaces generated by the orthogonal matrices constructed in Steps 3 and 5 intersect with a good probability. We use the following definition to capture an actually stronger requirement.

Definition 1 *Let $\mathcal{S} := \{A, A'\}$ be a generator set of $SL(2, \mathbb{F}_{2^n})$ and let s, t, A_i, s_i, t_i, w_i, n_1 as in Lemma 8. Let $\mathbf{A}$ be an algorithm solving the factorization problem for $\tilde{\mathcal{S}} := \{A_{n_1}, A'_{n_1}\} \cup \{O(w_i) | i \in I\}$ where $I \subset \{1, \dots, n_1\}$ is such that $\{w_i | i \in I\}$ is a basis of $< w_1, \dots, w_{n_1} >_+$. Let $\mathcal{D}_U$ be the uniform distribution on $SL(2, \mathbb{F}_{2^n})$ and let $\mathcal{D}_{g,R}$ be the distribution of the product $g \prod_{i=1}^{R} \left(x_i A^{-1} + (1-x_i) A'^{-1} \right)$ when $x_i \in \{0,1\}$ are randomly chosen. For any $r \in SL(2, \mathbb{F}_{2^n})$, let $w(r) \in < w_1, \dots, w_{n_1} >_+$ be the parameter of the orthogonal matrix obtained by applying the algorithm $\mathbf{A}$ to g, applying Lemma 9 to the result and removing the Q component. For any $g \in SL(2, \mathbb{F}_{2^n})$ and any $m \geq 1$, let $P_{g,R}^{\mathbf{A},m}$ be the probability that $< w(r_1), \dots, w(r_t) >_+ = < w(s_1), \dots, w(s_t) >_+$ when the r_i are generated according to $\mathcal{D}_U$ and the s_i are generated according to $\mathcal{D}_{g,R}$. We say that* $\mathbf{A}$ treats R-short products as random elements *if for any $g \in SL(2, \mathbb{F}_{2^n})$ and any $m > n_1 + 10$, we have $P_{g,R}^{\mathbf{A},m} > 1/2$.*

Since the size of $SL(2, \mathbb{F}_{2^n})$ is roughly 2^{3n} and the diameters of expander graphs tend to be as small as possible, random walks of length $R := 6n$ seem to be sufficient for our algorithm.

Assumption 1 *Let $\mathcal{S}$, s, t, A_i, s_i, t_i, w_i, n_1 and an algorithm $\mathbf{A}$ as in Definition 1. Then $\mathbf{A}$ treats $6n$-short products as random elements.*

Proposition 10 *Let $\mathcal{S} := \{A, A'\}$ be a generator set of $SL(2, \mathbb{F}_{2^n})$ and let s, t, A_i, s_i, t_i, w_i, n_1 as in Lemma 8. Under Assumption 1, the factorization problem for the set $\mathcal{S}$ can be reduced to the factorization problem for the set $\tilde{\mathcal{S}} := \{A_{n_1}, A'_{n_1}\} \cup \{O(w_i) | 1 \leq i \leq n_1\}$. Let $\mathbf{A}$ be an algorithm for $\tilde{\mathcal{S}}$ and let $\hat{L}$ be the maximal value L appearing in all the products* (1) *returned by $\mathbf{A}$. The reduction algorithm returns factorizations of lengths at most $2(n_1+1)3^{n_1-1}\hat{L} + 6(n_1+1)n$. Moreover, these factorizations can be written as straight-line programs of lengths at most $(n_1+1)(6n_1\hat{L} - 6\hat{L} + 6n + n_1 + 1)$.*

Proof: Under Assumption 1, taking $R := 6n$ ensures that the reduction algorithm succeeds with high probability. The factorization returned is composed of a Q component and at most n_1 orthogonal components. Following Lemma 9, each of them has length at most $2 \cdot 3^{n_1-1}\hat{L} + 6n$ and each component can be written as a straight-line program of length at most $6(n_1-1)\hat{L} + n_1 + 6n$. $\square$

To conclude with the reduction tools introduced in this section, let us consider Proposition 10 with $n_1 = 1$. Let us define $M := AA'$ and suppose that the set $\mathcal{S}_2 := \{M, O(w_1)\}$ generates $SL(2, \mathbb{F}_{2^n})$. With a probability about one half, the symmetric matrix M can be replaced by a diagonal matrix $\Lambda(\lambda)$ through conjugation by the orthogonal matrix given in Lemma 7. Moreover with a very large probability, λ then has algebraic degree n. The field $\mathbb{F}_{2^n}$ can then be seen as $\mathbb{F}_2[\lambda]/(p(\lambda))$ where p is the minimal polynomial of λ over $\mathbb{F}_2$. If one of the two conditions is not satisfied, the matrix M can be replaced by another short symmetric product of A and A', until both conditions are satisfied (heuristically, we expect that very short products will suffice). To solve the factorization problem for any set in practice, our heuristic analysis shows that it is sufficient to focus on generator sets

$$\mathcal{S}_w := \{\Lambda(X), O(w)\}$$

where $w \in \mathbb{F}_{2^n}$ depends on the initial parameters but X is a constant primitive element in the field.

The heuristic reduction tools presented in this section therefore restrict the generator sets of interest from a family of roughly 2^{6n} elements (the group $SL(2, \mathbb{F}_{2^n})$ contains about 2^{3n} matrices and most pairs of them generate it) to some "one-dimensional" families of size 2^n.

3 Changing the target: trapdoor matrices

Starting from some of the generator sets constructed in Section 2, we now reduce the factorization of any element in $SL(2, \mathbb{F}_{2^n})$ to the factorization of particular elements that we call *trapdoor matrices.*

3.1 Matrices with eigenvector $(1,1)'$

For the parameters of the Tillich-Zémor hash function, Petit and Quisquater [35] showed how to factor any matrix of $SL(2, \mathbb{F}_{2^n})$ from the factorization of one particular matrix with a certain property. The following proposition is inspired by this approach.

Proposition 11 *Let $\mathcal{S} := \{A, A'\}$ be a generator set of $G := SL(2, \mathbb{F}_{2^n})$. Let $M \in SL(2, \mathbb{F}_{2^n})$ with eigenvector equal to $\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$ such that the corresponding eigenvalue has algebraic degree n. Then any matrix of $SL(2, \mathbb{F}_{2^n})$ can be written in polynomial time as a product of at most $8n^2$ matrices M and M' and 4 matrices A. Given a straight-line program of length L' factoring M as a product of A and A', any element of $SL(2, \mathbb{F}_{2^n})$ can be represented as a straight-line program of length $7(n+1) + 2L'$.*

Proof: We first remark that $M = \left(\begin{smallmatrix}a & b\\ c & d\end{smallmatrix}\right) \in G$ has $\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$ as eigenvector if and only if $a+b+c+d = 0$. Indeed, we have $M\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right) = \left(\begin{smallmatrix}a+b\\ c+d\end{smallmatrix}\right)$ so $\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$ is eigenvector if and only if $a+b = c+d$. The corresponding eigenvalue is $\lambda := a+b$. Therefore, we can write M as $M = \left(\begin{smallmatrix}a & b\\ b+t & a+t\end{smallmatrix}\right)$ for some a, b, t satisfying $a^2 + at + b^2 + bt = 1$. Note that any power of M, say M^k, also has eigenvector $\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$, this time with corresponding eigenvalue λ^k.

By our assumption on $\mathcal{S}$, the knowledge of a factorization of M also provides a factorization of M' and MM'. So we have a factorization of

$$MM' = \begin{pmatrix}a & b\\ b+t & a+t\end{pmatrix}\begin{pmatrix}a & b+t\\ b & a+t\end{pmatrix} = \begin{pmatrix}a^2+b^2 & (a+b)t\\ (a+b)t & a^2+b^2\end{pmatrix} = \begin{pmatrix}a^2+b^2 & a^2+b^2+1\\ a^2+b^2+1 & a^2+b^2\end{pmatrix} = O(\lambda^2+1)$$

and for any $k \in \mathbb{Z}$, a factorization of

$$M^k M'^k = O(\lambda^{2k}+1).$$

We now argue that the matrices of the set $\{O(\lambda^{2k}+1), k = 1, ..., n\}$ generate the whole orthogonal subgroup of $SL(2, \mathbb{F}_{2^n})$. Let p be the minimal polynomial of λ. It is also the minimal polynomial of λ^2.

By assumption, p has degree n. Let $p_i \in \mathbb{F}_2, i = 0, \ldots, n$ such that that $p(X) = \sum_{i=0}^{n} p_i X^i$. Since p is irreducible we have $p_0 = 1$ (otherwise X would divide p) and $\sum_{i=0}^{n} p_i = 1 \bmod 2$ (otherwise $X+1$ would divide p). Since p is minimal, the set $\{1, \lambda^2, \ldots, \lambda^{2(n-1)}\}$ is a basis of $\mathbb{F}_{2^n}$ as a vector space over $\mathbb{F}_2$. By elementary operations on the elements of this basis, we obtain another basis $\{1 + \sum_{i=1}^{n-1} p_i(\lambda^{2i}+1), \lambda^2 + 1, \ldots, \lambda^{2(n-1)} + 1\}$. Finally, we observe that $1 + \sum_{i=1}^{n-1} p_i(\lambda^{2i}+1) = \lambda^{2n} + p(\lambda^2) + \sum_{i=1}^{n-1} p_i = \lambda^{2n} + 1$, hence the result.

It remains to show how to factor any non-orthogonal matrix g. Let $A := \left(\begin{smallmatrix} a' & b' \\ c' & d' \end{smallmatrix}\right)$ with $a'd' + b'c' = 1$ and let $g := \left(\begin{smallmatrix} \alpha & \beta \\ \gamma & \delta \end{smallmatrix}\right)$ with $\alpha\delta + \beta\gamma = 1$. Consider the equation

$$g = O(w_1)AO(w_2)AO(w_3)AO(w_4) \tag{3}$$

in the four unknowns $w_1, w_2, w_3, w_4 \in \mathbb{F}_{2^n}$. Equivalently, we have

$$AO(w_2)A0(w_3) = O(w_1)gO(w_4)A^{-1}. \tag{4}$$

This equation can be translated into a system of four polynomial equations over $\mathbb{F}_{2^n}$. By elementary transformations on these equations, we can instead consider the system

$$\begin{cases} \left(\begin{smallmatrix} 1 & 1 \end{smallmatrix}\right) AO(w_2)AO(w_3) \left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 1 \end{smallmatrix}\right) O(w_1)gO(w_4)A^{-1} \left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right) \\ \left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right) AO(w_2)AO(w_3) \left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right) O(w_1)gO(w_4)A^{-1} \left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right) \\ \left(\begin{smallmatrix} 1 & 1 \end{smallmatrix}\right) AO(w_2)AO(w_3) \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 1 \end{smallmatrix}\right) O(w_1)gO(w_4)A^{-1} \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right) \\ \left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right) AO(w_2)AO(w_3) \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right) = \left(\begin{smallmatrix} 1 & 0 \end{smallmatrix}\right) O(w_1)gO(w_4)A^{-1} \left(\begin{smallmatrix} 1 \\ 0 \end{smallmatrix}\right) \end{cases}$$

Since the determinant of both sides in (4) is 1 by construction, the last equation of this system is automatically verified if the first three equations are satisfied. The first equation leads to

$$(a'^2 + b'^2 + c'^2 + d'^2)w_2 + (a' + b' + c' + d')(\alpha + \beta + \gamma + \delta)w_4 + (a' + d')(a' + b' + c' + d') \\ + (a' + c')(\beta + \delta) + (b' + d')(\alpha + \gamma) = 0$$

We have $a' + b' + c' + d' \neq 0$, otherwise A and A' have eigenvector $(1, 1)$ and they do not generate $SL(2, \mathbb{F}_{2^n})$. Therefore, we have

$$w_2 = \frac{(\alpha + \beta + \gamma + \delta)w_4 + (a' + d')}{a' + b' + c' + d'} + \frac{(a' + c')(\beta + \delta) + (b' + d')(\alpha + \gamma)}{a'^2 + b'^2 + c'^2 + d'^2}. \tag{5}$$

The second equation provides

$$(a' + b' + c' + d')(\alpha + \beta + \gamma + \delta)w_4 w_1 + [(a' + c')(\beta + \delta) + (b' + d')(\alpha + \gamma)]w_1 + p_1(w_2, w_4) = 0$$

where p_1 is a linear function of w_2 and w_4. As long as

$$(a' + b' + c' + d')(\alpha + \beta + \gamma + \delta)w_4 \neq (a' + c')(\beta + \delta) + (b' + d')(\alpha + \gamma), \tag{6}$$

the second equation implies

$$w_1 = \frac{p_1(w_2, w_4)}{(a' + b' + c' + d')(\alpha + \beta + \gamma + \delta)w_4 + (a' + c')(\beta + \delta) + (b' + d')(\alpha + \gamma)}. \tag{7}$$

Finally, the third equation gives

$$(a' + b' + c' + d')^2 w_2 w_3 + (a' + d')(a' + b' + c' + d')w_3 + p_2(w_2, w_4) = 0$$

where p_2 is an afine function of w_2 and w_4. By Equation (5), Condition (6) is equivalent to $(a' + b' + c' + d')w_2 \neq a' + d'$. As long as this condition holds, we have

$$w_3 = \frac{p_2(w_2, w_4)}{(a' + b' + c' + d')[w_2(a' + b' + c' + d') + (a' + d')]}. \tag{8}$$

Summarizing, if there exists w_4 satisfying (6), then Equations (5), (7) and (8) provide corresponding w_1, w_2, w_3 to factor g as in Equation (3). Looking at the condition (6) more closely, this will be the case as long as $\alpha + \beta + \gamma + \delta \neq 0$, in other words as long as g does not have eigenvector $\left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right)$. On the other hand, if g has eigenvector $\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$, then $A^{-1}g$ does not have eigenvector $\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$, otherwise A and A' will also have eigenvector $\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$, and they will not generate the whole group. In that case, we can apply the previous reasoning to $A^{-1}g$ and deduce a decomposition of g as

$$g = AO(w_1)AO(w_2)AO(w_3)AO(w_4).$$

From the proof, we see that any matrix g can be written as a product of at most 4 orthogonal matrices and 4 occurrences of A. Each orthogonal matrix can be written as a product of at most n matrices $O(\lambda^{2k}+1)$ and each of these matrices can be written as a product of at most n matrices M and n matrices M'. If M is given as a straight-line program of length L', then M' can also be given as a straight-line program of length L'. All matrices $O(\lambda^{2k}+1)$ for $k < n$ can be constructed with a straight-line program of length $2L' + 3n$, any four orthogonal matrices need straight-line programs of lengths at most $2L' + 7n$, and any element g needs a straight-line program of length at most $2L' + 7(n + 1)$. □

Remarks. The condition on the eigenvalue is not very restrictive. It can be formulated equivalently as "the eigenvalue does not belong to any subfield of $\mathbb{F}_{2^n}$". If M is chosen randomly among all the matrices with eigenvector $\left(\begin{smallmatrix}1\\1\end{smallmatrix}\right)$, the probability that the condition is satisfied is approximately $1 - 2^{n'-n}$ where n' is the largest non trivial divisor of n. In particular if n is prime, the condition can be formulated as "the eigenvalue is neither 0 nor 1" or "M is not an orthogonal matrix".

In cryptographic protocols, the factorization of the special matrix required in Proposition 11 could be used as a *trapdoor* since its knowledge allows performing some computational task that remains challenging today (despite the progress achieved in Section 4.2). However, we point out that this trapdoor can easily be recovered from any factorization returned by the algorithm of Proposition 11. This unusual property (not present in the RSA trapdoor one-way function [42]) would limit the practical use of the trapdoor, although specific cryptographic protocols may precisely require it [7].

3.2 Towards a reduction to the Tillich-Zémor parameters

The factorization problem in $SL(2, \mathbb{F}_{2^n})$ was solved in [35] for the Tillich-Zémor parameters. The following proposition partly fills the gap between these parameters and the reductions of Section 2.

Proposition 12 *Let $G := SL(2, \mathbb{F}_{2^n})$ and let $\mathcal{S} := \{M, O(w)\}$ be a set of generators of G such that M is a symmetric matrix. Let $t := t(M)$ be the trace of M. Suppose that $wt = 1$ and that t has algebraic degree n. Then the factorization problem for the generators $\mathcal{S}$ reduces to the factorization problem for the Tillich-Zémor generators.*

Proof: By Lemma 6, there exists $\tilde{w} \in \mathbb{F}_{2^n}$ such that

$$T := \left(\begin{smallmatrix}t & 1\\1 & 0\end{smallmatrix}\right) = O(\tilde{w})MO(\tilde{w}).$$

Let $S_1 := \left(\begin{smallmatrix}1 & 1\\1 & 1+t\end{smallmatrix}\right)$. We have

$$S_1^{-1}TS_1 = T$$

and

$$S_1^{-1}O(w)S_1 = \left(\begin{smallmatrix}1 & wt\\0 & 1\end{smallmatrix}\right).$$

Let $S := O(\tilde{w})S_1$. We have

$$S^{-1}MS = S_1^{-1}O(\tilde{w})MO(\tilde{w})S_1 = S_1^{-1}TS_1 = T$$

and

$$\begin{aligned} T_1 := S^{-1}O(w)MS &= S^{-1}O(w)S \cdot S^{-1}MS = S_1^{-1}O(\tilde{w})O(w)O(\tilde{w})S_1 \cdot T \\ &= S_1^{-1}O(w)S_1 \cdot T = \left(\begin{smallmatrix}1 & wt\\0 & 1\end{smallmatrix}\right)\left(\begin{smallmatrix}t & 1\\1 & 0\end{smallmatrix}\right) = \left(\begin{smallmatrix}t(w+1) & 1\\1 & 0\end{smallmatrix}\right) = \left(\begin{smallmatrix}t+1 & 1\\1 & 0\end{smallmatrix}\right). \end{aligned}$$

Finally, let

$$Z := TT_1T^{-1} = \left(\begin{smallmatrix}t & t+1\\1 & 1\end{smallmatrix}\right).$$

Let p be the minimal polynomial of t. By hypothesis, p has degree n. Representing the field $\mathbb{F}_{2^n}$ as

$$\mathbb{F}_{2^n} \approx \mathbb{F}_2[t]/(p(t)),$$

the matrices T and Z are exactly the Tillich-Zémor generators [45]. □

Remark. Following [35], the factorization problem can be solved for the Tillich-Zémor parameters in time $\mathcal{O}(n^3)$, with factorizations of lengths bounded by $12n^3 + 6n^2 + 3n + 5$. If n is prime, the algorithm is deterministic and does not rely on any heuristic assumption [35].

4 New heuristic algorithms for factoring in $SL(2, \mathbb{F}_{2^n})$

In this section, we give two new heuristic algorithms for solving the factorization problem in $G := SL(2, \mathbb{F}_{2^n})$. Interestingly, our algorithms are not restricted to a particular generator set like in [35].

4.1 A heuristic exponential time algorithm for polynomial length factorizations

Propositions 3 and 11 suggest the following algorithm. Given a generator set $\mathcal{S}$ for G and a matrix $g \in G$,

1. Find two short products $\tilde{A}$ and $\tilde{B}$ of A and B such that $\tilde{A}\tilde{B}$ and $\tilde{B}\tilde{A}$ generate $SL(2, \mathbb{F}_{2^n})$. Replace $\mathcal{S}$ by $\{\tilde{A}, \tilde{B}\}$.
2. Replace $\{\tilde{A}, \tilde{B}\}$ by $\tilde{\mathcal{S}} := \{A, A'\}$ and g by $\tilde{g}$ using Proposition 3.
3. Find a product M of A and A' that satisfies the conditions of Proposition 11.
4. Apply Proposition 11 to find a factorization of $\tilde{g}$ as a product of A and A'.
5. Deduce a factorization of g for the generator set $\mathcal{S}$.

Testing whether $\{\tilde{A}\tilde{B}, \tilde{B}\tilde{A}\}$ generate $SL(2, \mathbb{F}_{2^n})$ can be done in (Monte-Carlo) probabilistic polynomial time [8]. For most generator sets $\{A, B\}$, the set $\{AB, BA\}$ generates $SL(2, \mathbb{F}_{2^n})$ and we can take $\tilde{A} = A$ and $\tilde{B} = B$ in Step 1. Otherwise, very short products will suffice. From the results of Sections 2 and 3, all steps but the third one can be carried out in probabilistic polynomial time. For the third step, we suggest to combine a birthday search with distinguished points as in [37]. More precisely, we generate about $2^{n/2}$ random products of A and A' with a polynomial length L to be fixed later. For each product M_i, we compute $\left(\begin{smallmatrix} a_{i1} \\ b_{i1} \end{smallmatrix}\right) := M_i \left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} a_{i2} \\ b_{i2} \end{smallmatrix}\right) := M_i^{-1} \left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right)$. We store the factorizations of M_i together with the *projective points* $p_{i1} := [a_{i1} : b_{i1}]$ and $p_{i2} := [a_{i2} : b_{i2}]$. If $p_{i1} = p_{i'2}$ and $i \neq i'$, we have

$$\left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right) = \frac{a_{i'2}}{a_{i1}} M_{i'} M_i \left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right)$$

hence $\left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right)$ is an eigenvector of $M_{i'}M_i$. Moreover, $I \neq M_{i'}M_i$ and the condition on the eigenvalue of $M_{i'}M_i$ hold with very large probability, larger than $1 - 2^{n'-n}$ where n' is the largest non trivial divisor of n. Since a proportion of about 2^{1-n} matrices of G has $\left(\begin{smallmatrix} 1 \\ 1 \end{smallmatrix}\right)$ as eigenvector, random products of size $L = n/2$ should be sufficient. The algorithm has an exponential time complexity $2^{n/2}$ times a "small" polynomial in n. The storage needs can be reduced to a negligible quantity using *distinguished points* techniques [39] as in [37].

We point out that a trivial birthday search in the group would have a complexity about $2^{3n/2}$. Moreover, our algorithm has two main advantages over the preimage algorithms of [37]. First, the use of trapdoor matrices avoids solving a multiplicative knapsack problem. Second, the non-polynomial time part can be achieved once and for all during a *precomputing phase* when many factorizations have to be found for the same generator set.

Unfortunately, the complexity of the precomputing phase remains exponential in parameter n.

4.2 Trading time for space and factorization lengths

The previous algorithm factored one of the trapdoor matrices of Proposition 11. Instead, the algorithm of this section reduces the factorization problem for an arbitrary generator set to the factorization problem for the Tillich-Zémor generators. Proposition 12 suggests starting with a generator set containing a symmetric matrix and at least one orthogonal matrix, then progressively modifying the trace of the symmetric matrix to satisfy the condition $wt = 1$. In order to modify the trace in some controllable way, we recursively use the following observation.

Lemma 13 *Let $M \in G$ be a symmetric matrix with trace $t := t(M)$, let $w \in \mathbb{F}_{2^n}$ and let $O := O(w)$. Then the matrix MOM is a symmetric matrix with trace $t^2(w + 1)$.*

PROOF: Let $M := \left(\begin{smallmatrix} a & b \\ b & d \end{smallmatrix}\right)$. We compute

$$MOM = \left(\begin{smallmatrix} (a^2+b^2)(w+1) & (a+b)(b+d)w+bt \\ (a+b)(b+d)w+bt & (b^2+d^2)(w+1) \end{smallmatrix}\right)$$

and $t(MOM) = (a^2 + d^2)(w + 1) = t^2(w + 1)$. $\square$

Let $\mathcal{S} := \{A, B\}$ be a generator set for $G := SL(2, \mathbb{F}_{2^n})$ and let $g \in G$. The factorization problem for $\mathcal{S}$ and g can be solved as follows.

1. Find two short products $\tilde{A}$ and $\tilde{B}$ of A and B such that
$$\tilde{\mathcal{S}} := \{M\} \cup \{O(w_i) | w_i \in \mathbb{F}_{2^n}, i = 1, ..., n_1\}$$
generate $SL(2, \mathbb{F}_{2^n})$, where $M := A_{n_1} A'_{n_1}$ and A_i, $O(w_i)$ are constructed by applying Proposition 3 and Lemma 8 to $\{\tilde{A}, \tilde{B}\}$. Let $\tilde{g}$ be as given by the proof of Proposition 3. (The parameter n_1 will be fixed below.)
2. Let $t := t(M)$. Let n_2 be a fixed integer. Find $w_{(i)} \in < w_1, ..., w_{n_1} >_+$ such that
$$w_{(0)} t^{2^{n_2}} \prod_{i=1}^{n_2} (w_{(i)} + 1)^{2^{n_2 - i}} = 1 \tag{9}$$
(Parameter n_2 will be fixed below.)
3. Applying Lemma 13 n_2 times, deduce a *trapdoor matrix* that is a product $\tilde{M}$ of length $2^{n_2+1} - 1$ in the elements of $\tilde{\mathcal{S}}$, such that $w_{(0)} t(\tilde{M}) = 1$. If the minimal polynomial of $t(\tilde{M})$ does not have degree n, search for a new solution of Equation (9), if necessary after increasing n_2.
4. Use Proposition 12 and the preimage algorithm for the Tillich-Zémor hash function [35] to compute a factorization of $\tilde{g}$ as a product of the elements of $\tilde{\mathcal{S}}$.
5. Deduce a factorization of g as a product of the elements of $\mathcal{S}$.

The only "hard" part of this algorithm is to solve Equation (9), a particular form of a multiplicative knapsack problem in $\mathbb{F}^*_{2^n}$. When various group elements have to be factored with respect to the same generator set, a solution of this equation can be precomputed to reduce the online time of our algorithm.

4.3 Solution to the multiplicative knapsack problem

We give a naive algorithm solving Equation (9), already sufficient to reduce the time complexity of the factorization problem from more than $2^{n/2}$ to $2^{n/\beta \log n}$ for some constant $\beta < 1$. Our algorithm uses Coppersmith's version of the function field sieve algorithm [11,2] to solve discrete logarithms in the field $\mathbb{F}^*_{2^n}$ and to reduce the multiplicative knapsack to a particular additive knapsack. For appropriately chosen parameters n_1 and n_2, the additive knapsack is solved using Wagner's generalized birthday approach [47]. We obtain the following algorithm:

1. Pick a random $w_{(0)} \in < w_1, ..., w_{n_1} >_+$. Fix $n_2 > 2$ and $n_1 := \left\lceil \frac{n}{1+\log n_2} \right\rceil$. Pick a random generator g of $\mathbb{F}^*_{2^n}$.
2. For every $w \in < w_1, ..., w_{n_1} >_+$, use Coppersmith's algorithm [11] to find e_w such that $w + 1 = g^{e_w}$. Also compute T such that $g^T = w_{(0)} t^{2^{n_2}}$. Let $E := \{e_w | w \in < w_1, ..., w_{n_1} >_+\}$.
3. Solve
$$\sum_{i=1}^{n_2} 2^{n_2 - i} e_i = -T \bmod (2^n - 1), \qquad e_i \in E \tag{10}$$
using Wagner's k-lists algorithm [47] with the k_2 lists $E_1 := E, E_2 := \{2e | e \in E\}, E_3 := \{4e | e \in E\}, ..., E_{n_2} := \{2^{n_2 - 1} e | e \in E\}$.

We have $n_1 n_2 = \frac{n n_2}{1+\log n_2} > n$ so Equation (9) is expected to have at least one solution (and many solutions for large values of n_2). We now argue that the above algorithm likely computes a solution of Equation (9), using memory $n_2 2^{n_1}$ and time $2^{n_1 + c n^{1/3} (\log n)^{2/3}}$, where c is a constant that can be approximated by 1.58.

The relation between n_1 and n_2 ensures that the lists in Step 3 are big enough to apply Wagner's k-lists algorithm [47]. This algorithm progressively solves Equation (10) up to $\log n_2$ blocks of $n/\log n_2$ bits by recursively merging couples of lists. The algorithm works if all the elements of all lists are independently and uniformly distributed, an assumption taken by Wagner in his analysis. In practice, the algorithm can often be used if all the lists are equal. It can even tolerate some dependencies between the elements as long as the different blocks still appear uniformly distributed.

To obtain our asymptotic estimates, we take the heuristic assumption that Wagner's algorithm behaves with the lists of Step 3 as it would behave on random lists. As the e_i are the discrete logarithms of the elements of some vector subspace, the assumption seems reasonable since the additive and multiplicative structures of finite fields are normally not correlated.

The complexity of our algorithm is therefore evaluated as follows. Wagner's algorithm requires space and time $n_2 2^{n_1}$. Coppersmith's algorithm has time complexity

$$exp((c' + o(1))n^{1/3}(\log_e n)^{2/3})$$

where c' is a constant and $o(1)$ is an asymptotically vanishing function of n [11]. In our estimates, we approximate $c' \approx 1.4$ [44] and we ignore the $o(1)$ term. The memory requirements of Coppersmith's algorithm are negligible compared to the size of the lists in Wagner's algorithm, whereas the total time complexity of our algorithm is dominated by the computation of 2^{n_1} discrete logarithms. The above estimates follow.

Remark. At first sight, solving Equation (9) might not appear much harder than solving a discrete logarithm problem in the unit group of the field $\mathbb{F}_{2^n}$, a problem that can certainly be solved today when we take $n \approx 160$ as was suggested by Tillich and Zémor in their paper [45]. (The current record in 2010 is a discrete logarithm computation in $\mathbb{F}^*_{2^{613}}$ [17].) However, it turns out that the hardness of Equation (9) comes less from its discrete logarithm component than from the special requirements of its (additive) knapsack component.

Although knapsack problems are NP-hard in general, they have often been solved in practice using the LLL algorithm [23,30,18]. Interestingly, some relaxations of Equation (10) can be solved this way. For example, the problems

$$\sum_{i=1}^{n_2} a_i e_i = -T \bmod (2^n - 1), \qquad e_i \in E, a_i \in \{0, 1, 2, 3, ..., 2^{n_2} - 1\}$$

or

$$\sum_{i=1}^{n_3} a_i e_i = -T \bmod (2^n - 1), \qquad e_i \in E, a_i \in \{1, 2, 4, ..., 2^{n_2-1}\}$$

for some $n_3 >> n_2$ can be solved by applying LLL on appropriately defined lattices, in a time approximately 2^{n^α} for some $0 < \alpha < 1$ (much better than the complexity obtained above using Wagner's algorithm). However, the solutions of these relaxations only provide solutions to Equation (10) if they additionally satisfy non-linear constraints between the a_i. The efficient insertion of these non-linear constraints into a lattice problem might not be possible at all. We therefore used Wagner's algorithm instead of LLL.

We leave a more efficient algorithm for solving Equation (9) to further work and we come back to our original problem.

4.4 Heuristic complexity bounds for factoring in $SL(2, \mathbb{F}_{2^n})$

We are now ready to give complexity estimates for the algorithm of Section 4.2.

As recalled in the remark after Proposition 12, the factorization algorithm of [35] returns factorizations of length about $12n^3$ for the Tillich-Zémor generators. From the proof of Proposition 12 and the construction of the trapdoor matrix in our algorithm, each Tillich-Zémor matrix leads to a product containing exactly 2^{n_2} matrices M as defined in Step 1 of our algorithm. The value L in Proposition 10 can therefore be approximated by $12n^3 \cdot 2^{n_2}$ and the algorithm of this section returns factorizations of length

$$2^{n_2+3}3^{n_1}(n_1+1)n^3 + 6(n_1+1)n \approx 2^{n_2+3}3^{n_1}n_1n^3.$$

These factorizations can be returned in the compressed form of straight-line programs of length bounded by

$$(n_1+1)(6n_1L - 6L + 6n + n_1 + 1) \approx 3^2 2^{n_2+3}n_1^2n^3.$$

The time complexity has two main components. The knapsack resolution requires time $2^{n_1+cn^{1/3}(\log n)^{2/3}}$. The reductions essentially take a time equal to the length of the straight-line programs needed to return the factorizations. We can therefore bound the time complexity by

$$2^{n_1+cn^{1/3}(\log n)^{2/3}} + 3^2 2^{n_2+3}n_1^2n^3 \leq 2\max(2^{n_1+cn^{1/3}(\log n)^{2/3}}, 3^2 2^{n_2+3}n_1^2n^3).$$

The memory complexity comes from Wagner's k-lists algorithm. It can be approximated by

$$n_2 2^{n_1}.$$

Taking $n_2 = n^\alpha$ for some constant $0 < \alpha < 1$ and $n_1 = \left\lceil \frac{n}{1+\log n_2} \right\rceil$ as in Section 4.3, we obtain subexponential time and memory algorithms producing factorizations of subexponential lengths for any generator set of $SL(2, \mathbb{F}_{2^n})$. We point out that despite the common "subexponential" qualification, the complexity of our algorithm is worse than the complexity of state-of-the-art discrete logarithm or factoring algorithms [2,11]. The "subexponential" complexity here essentially means $2^{\frac{n}{\beta \log n}}$ for some $0 < \beta < 1$ related to α.

To evaluate the hardness of the factorization problem in practice, we computed these bounds for $n \in \{160, 256, 512, 1024\}$ and various values of the parameter n_2.

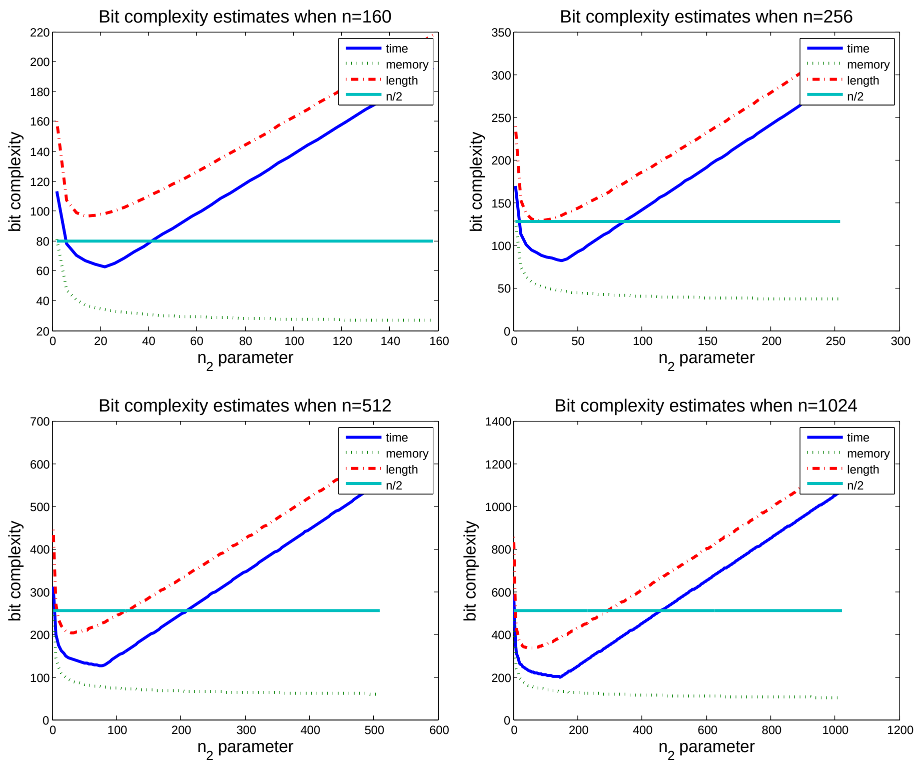

Fig. 1 Complexity bounds for factoring in $SL(2, \mathbb{F}_{2^n})$

We stress that the numbers obtained must be taken with some caution due to the various approximations involved in our complexity estimates. Nevertheless, they do give some insight on the practical hardness of the problem. The results are shown in Figure 4.4 together with the $n/2$ bit complexity obtained for the attack of Section 4.1. As expected, the time complexity advantage of our second algorithm increases with n. For $n = 160$ (the parameter size suggested by Tillich and Zémor in their paper [45]) and $n_2 = 22$, our second algorithm finds factorizations of length 2^{99} in time 2^{63} and memory 2^{34}. For $n = 1024$ and $n_2 = 146$, it finds factorizations of length 2^{385} in time 2^{200} and memory 2^{133}. In cryptographic applications, the time complexity advantage of our algorithm will have to be balanced with the factorization lengths.

5 Conclusion

In this paper, we studied the hardness of the factorization problem in the group $G = SL(2, \mathbb{F}_{2^n})$ for *generic* generator sets. According to Babai's conjecture, polylogarithmic length factorizations exist for any element of G. Although the conjecture has now been proved for this group, all previous algorithms for factoring in G either focused on particular generator sets, ran in time exponential or produced factorizations with exponential length in $\log |G|$. In contrast, we argued that the heuristic algorithm of Section 4.2 requires subexponential time and memory and it produces subexponential lengths factorizations.

More importantly in our opinion, this paper introduces a series of new techniques that could ultimately lead to a polynomial time algorithm producing polynomial length factorizations for any generator set of $SL(2, \mathbb{F}_{2^n})$. We suggested the following 3-step approach to tackle the problem:

1. Replace the generator set by another simpler one.
2. Identify a class of *trapdoor matrices*, such that factoring only one of these matrices would allow factoring any matrix in the group.
3. Factor a trapdoor matrix.

We made important progress on the first two steps of this program in Sections 2 and 3. In particular, we reduced the generator sets of interest to some classes containing 2^n pairs of generators with a "nice" special structure and we identified large classes of trapdoor matrices. Some of our reductions do not work in all generality but only with a probability very close to 1 for large values of n. We leave the rigorous treatment of all special cases to further work. Some reductions also rely on heuristic arguments. In particular, the reductions of Section 2.3 rely on the ability to generate random elements of $SL(2, \mathbb{F}_{2^n})$ from short random products of a symmetric pair of generators.

The third and last step of our program is partially solved in Section 4. We argued that subexponential length factorizations can asymptotically be computed in subexponential time using the algorithm of Section 4.2. We believe that this result can be significantly improved, either by better solving Equation (9) or by bridging the results of Steps 1 and 2 in a different way.

From a practical and cryptographic point of view, the algorithms of Section 4 do not really invalidate the parameter size $n \approx 160$ suggested by Tillich and Zémor [45]. However, they warn that the complexity of the problem is lower than birthday searches when n increases. Moreover, we expect extensions of our work to lead to even better and more practical attacks on Cayley hash functions. In particular, we believe that a probabilistic polynomial time algorithm for Problem 1 exists and can be built on the results of Sections 2 and 3.

Acknowledgements The author would like to thank Sylvie Baudine and the anonymous referees for their help in improving 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. L. M. Adleman. The function field sieve. In L. M. Adleman and M.-D. A. Huang, editors, *ANTS*, volume 877 of *Lecture Notes in Computer Science*, pages 108–121. Springer, 1994.
3. L. Babai. On the diameter of Eulerian orientations of graphs. In *SODA*, pages 822–831. ACM Press, 2006.
4. L. Babai and Ákos Seress. On the diameter of permutation groups. *European J. Combin.*, 13(4):231–243, 1992.

5. L. Babai, G. Hetyei, W. M. Kantor, A. Lubotzky, and Á. Seress. On the diameter of finite groups. In *FOCS*, volume II, pages 857–865. IEEE, 1990.
6. E. Breuillard, B. Green, and T. Tao. Approximate subgroups of linear groups. *Geom. Funct. Anal.*, 21(4):774–819, 2011.
7. J. Cathalo and C. Petit. One-time trapdoor one-way functions. In M. Burmester, G. Tsudik, S. S. Magliveras, and I. Ilic, editors, *ISC*, volume 6531 of *Lecture Notes in Computer Science*, pages 283–298. Springer, 2010.
8. F. Celler and C. Leedham-Green. A non-constructive recognition algorithm for the special linear and other classical groups. In *Groups and Computation II*, pages 61–67. American Mathematical Society, 1997.
9. D. X. Charles, K. E. Lauter, and E. Z. Goren. Cryptographic hash functions from expander graphs. *J. Cryptology*, 22(1):93–113, 2009.
10. C. Charnes and J. Pieprzyk. Attacking the SL_2 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.
11. D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. 30(4):587–593, 1984.
12. 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.
13. M. Grassl, I. Ilic, S. S. Magliveras, and R. Steinwandt. Cryptanalysis of the Tillich-Zémor hash function. *J. Cryptology*, 24(1):148–156, 2011.
14. H. A. Helfgott. Growth and generation in $sl_2(z/pz)$. *Ann. of Math. (2)*, 167 (2):601–623, 2008.
15. H. A. Helfgott. Growth and generation in $SL_3(Z/pZ)$. *Journal of the European Mathematical Society (JEMS)*, 13 (3):761–851, 2011.
16. S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43:439–561, 2006.
17. A. Joux and R. Lercier. Discrete logarithms in $GF(2^{607})$ and $GF(2^{613})$. Email on the NMBRTHRY mailing list, September 2005.
18. A. Joux and J. Stern. Lattice reduction: A toolbox for the cryptanalyst. *J. Cryptology*, 11(3):161–185, 1998.
19. W. M. Kantor. Some large trivalent graphs having small diameters. *Discrete Appl. Math.*, 37/38:353–357, 1992.
20. M. Kassabov and T. R. Riley. Diameters of Cayley graphs of Chevalley groups. *Eur. J. Comb.*, 28(3):791–800, 2007.
21. M. Larsen. Navigating the Cayley graph of $SL_2(\mathbb{F}_p)$. *International Mathematics Research Notices. IMRN*, 27:1465–1471, 2003.
22. A. Lauder. Continued fractions of Laurent series with partial quotients from a given set. *Acta Arithmetica XC*, 3:252–271, 1999.
23. A. K. Lenstra, J. Lenstra, H. W., and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(5):515–534, 1982.
24. M. W. Liebeck and A. Shalev. The probability of generating a finite simple group. *Geom. Dedicata*, 56:103–113, 1995.
25. A. Lubotzky. *Discrete groups, expanding graphs and invariant measures.* Birkhaüser Verlag, 1994.
26. N. A. Lynch. Straight-line program length as a parameter for complexity analysis. *J. Comput. System Sc.*, 21(3):251–280, 1980.
27. R. J. McEliece. A public-key cryptosystem based on algebraic coding theory. The Deep Space Network Progress Report, DSN PR 42-44, January and February 1978. 114-116.
28. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Inc., Boca Raton, FL, USA, 1996.
29. J. P. Mesirov and M. M. Sweet. Continued fraction expansions of rational expressions with irreducible denominators in characteristic 2. *J. Number Theory*, 27:144–148, 1987.
30. A. M. Odlyzko. The rise and fall of knapsack cryptosystems. In *In Cryptology and Computational Number Theory*, pages 75–88. A.M.S, 1990.
31. J. Patarin. Hidden fields equations (hfe) and isomorphisms of polynomials (ip): Two new families of asymmetric algorithms. In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 1996.
32. C. Petit. *On graph-based cryptographic hash functions.* PhD thesis, Université catholique de Louvain, 2009. `http://perso.uclouvain.be/christophe.petit/files/thesis.pdf`.
33. 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.
34. C. Petit, K. E. Lauter, and J.-J. Quisquater. Cayley hashes: A class of efficient graph-based hash functions. Available at `http://perso.uclouvain.be/christophe.petit/index.html`, 2007.
35. C. Petit and J.-J. Quisquater. Preimages for the Tillich-Zémor hash function. In A. Biryukov, G. Gong, and D. R. Stinson, editors, *Selected Areas in Cryptography*, volume 6544 of *Lecture Notes in Computer Science*, pages 282–301. Springer, 2010.
36. C. Petit and J.-J. Quisquater. Rubik's for cryptographers. Available at `http://perso.uclouvain.be/christophe.petit/index.html`, 2010.
37. 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.
38. L. Pyber and E. Szabó. Growth in finite simple groups of Lie type. arXiv:1001.4556v1, Jan 2010.
39. J.-J. Quisquater and J.-P. Delescaille. How easy is collision search? application to DES (extended summary). In *EUROCRYPT*, pages 429–434, 1989.
40. O. Regev. Lattice-based cryptography. In C. Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 131–141. Springer, 2006.
41. T. R. Riley. Navigating in the Cayley graphs of $SL_N(\mathbb{Z})$ and $SL_N(\mathbb{F}_p)$. *Geom. Dedicata*, 113/1:215–229, 2005.
42. R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signatures and public-key cryptosystems (reprint). *Commun. ACM*, 26(1):96–99, 1983.

43. R. Steinwandt, M. Grassl, W. Geiselmann, and T. Beth. Weaknesses in the $SL_2(\mathbb{F}_{2^n})$ hashing scheme. In M. Bellare, editor, *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 287–299. Springer, 2000.
44. E. Thomé. Computation of discrete logarithms in $\mathbb{F}_{2^{607}}$. In C. Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 107–124. Springer, 2001.
45. 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.
46. 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.
47. D. Wagner. A generalized birthday problem. In M. Yung, editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 288–303. Springer, 2002.
48. 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.
49. G. Zémor. Hash functions and Cayley graphs. *Des. Codes Cryptog.*, 4(4):381–394, 1994.