Public key cryptosystem MST_3: cryptanalysis and realization

Pavol Svaba
Tran van Trung
Institut für Experimentelle Mathematik
Universität Duisburg-Essen
Ellernstrasse 29
45326 Essen, Germany
{svaba,trung}@iem.uni-due.de

Abstract

A new type of public key cryptosystem, called MST_3, has been recently introduced on the basis of covers and logarithmic signatures for non-abelian finite groups. The class of Suzuki 2-groups has been proposed for a possible realization of the generic scheme. Due to their simple structure, the groups enable us to study the security of the system and also provide an efficient implementation. An earlier relevant result of the cryptanalysis has shown that the transversal logarithmic signatures are unfit for use in this realization. In this paper we present a revised version of MST_3 for the Suzuki 2-groups and show a thorough study of its security. Using heuristic and algebraic methods we establish strong lower bounds for the workload of conceivable direct attacks on the private key of the scheme. We then develop a powerful chosen plaintext attack which allows us to rule out the usage of a certain class of logarithmic signatures. In addition, we show a class of logarithmic signatures withstanding this attack and thus to our knowledge they could be used in the realization of the scheme. Finally, we describe and discuss the implementation issues of the scheme in detail and include data of its performance obtained from an experimental result.

1 Introduction

In recent times, asymmetric cryptography has become essential to many information systems. Many public key cryptosystems have been proposed, but only few of such systems remain unbroken. Most of these are based on the perceived intractibility of certain mathematical problems in very large, finite cyclic groups in certain particular representations. Prominent hard problems are i) the problem of factoring large integers, ii) the Discrete Logarithm Problem (DLP) in particular representations of large cyclic groups, and iii) finding a short basis for a given integral lattice $\mathcal{L}$ of large dimension. Unfortunately, in view of P. Shor's quantum algorithms for integer factoring, and solving the DLP [12], the known public-key systems will be insecure when quantum computers become practical. A recent report edited by P. Nguyen [11] identifies these and other problems facing the field of information security in the future.

Recently, a new type of public key cryptosystem, called MST_3 [6], has been developed on the basis of logarithmic signatures and covers of finite non-abelian groups. For a possible realization of the generic version of this system, the Suzuki 2-groups have been suggested.

Due to their simple structure, these groups make it possible for studying the security of the scheme. As shown in previous results, a lower bound for the work effort required in terms of the size of the underlying groups is obtained [6]. By exploiting the distinguishing feature of the group operation in the Suzuki 2-groups, a further analysis in [10] has shown that the transversal logarithmic signatures are unfit to use in this realization.

In this paper we present an approach to re-designing MST_3 for the Suzuki 2-groups. The method makes use of the characteristics of the group operation as well as the structure of these groups. We present a thorough study of the security of the scheme by using heuristic and algebraic methods. We first determine the complexity for the lower bounds of conceivable direct attacks to recover the private key in terms of the size of the groups. These bounds give a hint of the strength of the system. We further develop a powerful method for a chosen plaintext attack showing that a certain class of transversal logarithmic signatures cannot be used. Moreover, there are classes of logarithmic signatures that withstand this attack when used in MST_3. We examine the usage of one such class in the realization of the scheme for which we are able to determine the complexity of this type of attack.

The paper is organized as follows: In Section 2 we summarize some basic facts above covers and logarithmic signatures for finite groups and their related induced mappings; a description of the Suzuki 2-groups is included. In Section 3 we present a revised version of the cryptosystem MST_3 and show its encryption, decryption. In Section 4 we study various direct attacks on the scheme, namely determining the private key from the public key, and show the lower bounds on the complexity of such attacks. In Section 5 we describe algorithms for generating logarithmic signatures for use in a possible implementation of the scheme. Also methods for factorization with respect to these logarithmic signatures are shown. Section 6 deals with the development of a powerful chosen plaintext attack on the scheme utilizing transversal logarithmic signatures. It is shown that the class of fused transversal logarithmic signatures withstands this type of attack. In Section 7 we present data of performance (including the attack complexity) of the scheme for various parameter sets from an experimental implementation. In addition, a method of reducing the key storage is described. We provide a conclusion in Section 8.

2 Preliminaries

In this section we briefly present notation, definitions and some basic facts about logarithmic signatures, covers for finite groups and their induced mappings. For more details the reader is refered to [8], [9]. The group theoretic notation used is standard and may be found in [4] or in any textbook of group theory.

Let $\mathcal{G}$ be a finite abstract group, we define the *width* of $\mathcal{G}$ to be the positive integer $w = \lceil \log |\mathcal{G}| \rceil$. Denote by $\mathcal{G}^{[\mathbb{Z}]}$ the collection of all finite sequences of elements in $\mathcal{G}$ and view the elements of $\mathcal{G}^{[\mathbb{Z}]}$ as single-row matrices with entries in $\mathcal{G}$. Let $X = [x_1, x_2, \dots, x_r]$ and $Y = [y_1, y_2, \dots, y_s]$ be two elements in $\mathcal{G}^{[\mathbb{Z}]}$. We define

$$X \cdot Y = [x_1y_1, x_1y_2, \dots, x_1y_s, x_2y_1, x_2y_2, \dots, x_2y_s, \dots, x_ry_1, x_ry_2, \dots, x_ry_s]$$

Instead of $X \cdot Y$ we will also write $X \otimes Y$ as ordinary tensor product of matrices, or for short we will write XY. If $X = [x_1, \dots, x_r] \in \mathcal{G}^{[\mathbb{Z}]}$, we denote by $\overline{X}$ the element $\sum_{i=1}^{r} x_i$ in the group ring $\mathbb{Z}\mathcal{G}$.

Suppose that $\alpha = [A_1, A_2, \dots, A_s]$ is a sequence of $A_i \in \mathcal{G}^{[\mathbb{Z}]}$, such that $\sum_{i=1}^{s} |A_i|$ is bounded by a polynomial in $\log |\mathcal{G}|$. Let

$$\overline{A_1} \cdot \overline{A_2} \cdots \overline{A_s} = \sum_{g \in \mathcal{G}} a_g g \ , \qquad a_g \in \mathbb{Z} \tag{2.1}$$

Let $\mathcal{S}$ be a subset of $\mathcal{G}$, then we say that α is

(i) a *cover* for $\mathcal{G}$ (or $\mathcal{S}$), if $a_g > 0$ for all $g \in \mathcal{G}$ $(g \in \mathcal{S})$.

(ii) a *logarithmic signature* for $\mathcal{G}$ (or $\mathcal{S}$), if $a_g = 1$ for every $g \in \mathcal{G}$ $(g \in \mathcal{S})$.

Thus, a *cover* $\alpha = [A_1, \dots, A_s]$ for a subset $\mathcal{S}$ of a finite group $\mathcal{G}$ can be viewed as an ordered collection of subsets A_i of $\mathcal{G}$ with $|A_i| = r_i$ such that each element $h \in \mathcal{S}$ can be expressed in at least one way as a product of the form

$$h = g_1 \cdot g_2 \cdots g_{s-1} \cdot g_s \tag{2.2}$$

for $g_i \in A_i$.

If every $h \in \mathcal{S}$ can be expressed in exactly one way by Equation (2.2), then α is called a *logarithmic signature* for $\mathcal{S}$. Thus, logarithmic signatures are a special class of covers.

The A_i are called the *blocks*, and the vector $(r_1, \dots, r_s)$ with $r_i = |A_i|$ the *type* of α. We say that α is *nontrivial* if $s \geqslant 2$ and $r_i \geqslant 2$ for $1 \leqslant i \leqslant s$; otherwise α is said to be *trivial.* Cover α is called *tame (or factorizable)* if the factorization in Equation (2.2) can be achieved in time polynomial in the width w of $\mathcal{G}$, it is called *wild* if it is not tame. Let $\gamma : 1_{\mathcal{G}} = \mathcal{G}_0 < \mathcal{G}_1 < \cdots < \mathcal{G}_s = \mathcal{G}$ be a chain of subgroups of $\mathcal{G}$, and let A_i be an ordered, complete set of right (or left) coset representatives of $\mathcal{G}_{i-1}$ in $\mathcal{G}_i$. It is clear that $[A_1, \dots, A_s]$ forms a logarithmic signature for $\mathcal{G}$, called *transversal logarithmic signature.* Transversal logarithmic signatures are an important example of tame logarithmic signatures [9].

In general, the problem of finding a factorization in Equation (2.2) with respect to a randomly generated cover is presumedly intractable. There are strong evidences in support of the hardness of the problem. For example, let $\mathcal{G}$ be a cyclic group and g be a generator of $\mathcal{G}$. Let $\alpha = [A_1, A_2, \dots, A_s]$ be any cover for $\mathcal{G}$, for which the elements of A_i are written as powers of g. Then the factorization with respect to α amounts to solving the Discrete Logarithm Problem (DLP) in $\mathcal{G}$.

Remark 2.1 *It is worth noting that the problem of how to generate random covers for finite groups of large order is completely solved in [13]. Probabilistic method shows that generation of random covers for groups of large order can be done with high efficiency and at minimum cost.*

The crucial point that makes covers useful for group based cryptography is that if the above factorization problem is intractable, then the covers essentially induce one-way functions. This can be described as follows. Let $\alpha = [A_1, A_2, \dots, A_s]$ be a cover of type $(r_1, r_2, \dots, r_s)$ for $\mathcal{G}$ with $A_i = [a_{i,1}, a_{i,2}, \dots, a_{i,r_i}]$ and let $m = \prod_{i=1}^{s} r_i$. Let $m_1 = 1$ and $m_i = \prod_{j=1}^{i-1} r_j$ for $i = 2, \dots, s$. Let τ denote the canonical bijection from $Z_{r_1} \oplus \mathbb{Z}_{r_2} \oplus \cdots \oplus \mathbb{Z}_{r_s}$ on $\mathbb{Z}_m$; i.e.

$$\begin{aligned} \tau: \quad & \mathbb{Z}_{r_1} \oplus \mathbb{Z}_{r_2} \oplus \cdots \oplus \mathbb{Z}_{r_s} \to \mathbb{Z}_m \\ \tau(j_1, j_2, \ldots, j_s) &:= \sum_{i=1}^{s} j_i m_i. \end{aligned}$$

Using τ we now define the surjective mapping $\breve{\alpha}$ induced by α.

$$\begin{aligned} \breve{\alpha} \quad &: \quad \mathbb{Z}_m \to \mathcal{G} \\ \breve{\alpha}(x) \quad &:= \quad a_{1,j_1} \cdot a_{2,j_2} \cdots a_{s,j_s}, \end{aligned}$$

where $(j_1, j_2, \ldots, j_s) = \tau^{-1}(x)$. Since τ and τ^{-1} are efficiently computable, the mapping $\breve{\alpha}(x)$ is efficiently computable.

Conversely, given a cover α and an element $y \in \mathcal{G}$, to determine any element $x \in \breve{\alpha}^{-1}(y)$ it is necessary to obtain any one of the possible factorizations of type (2.2) for y and determine indices $j_1, j_2, \ldots, j_s$ such that $y = a_{1,j_1} \cdot a_{2,j_2} \cdots a_{s,j_s}$. This is possible if and only if α is tame. Once a vector $(j_1, j_2, \ldots, j_s)$ has been determined, $\breve{\alpha}^{-1}(y) = \tau(j_1, j_2, \ldots, j_s)$ can be computed efficiently.

There are different types of transformations that can apply to covers. Here, we consider just one type, which is used in the next sections.

Assume that $\alpha = [A_1, A_2, \ldots, A_s]$ is a cover for $\mathcal{G}$. Let $g_0, g_1, \ldots, g_s \in \mathcal{G}$, and consider $\beta = [B_1, B_2, \ldots, B_s]$ with $B_i = g_{i-1}^{-1} A_i g_i$. We say that β is a *two sided transform* of α by $g_0, g_1, \ldots, g_s$; in the special case, where $g_0 = 1$ and $g_s = 1$, β is called *a sandwich* of α. Note that β is a cover for $\mathcal{G}$.

Two covers (logarithmic signatures) α, β are said to be *equivalent* if $\breve{\alpha} = \breve{\beta}$. For example, if β is a sandwich of α, then α and β are obviously equivalent.

We make use of following cryptographic hypothesis that if $\alpha = [A_1, A_2, \ldots, A_s]$ is a random cover for a "large" subset $\mathcal{S}$ of a group $\mathcal{G}$, then finding a factorization in (2.2) is an intractable problem. In other words, the mapping

$$\breve{\alpha} \quad : \quad Z_m \to \mathcal{S}$$

induced by α with $m = \prod_{i=1}^{s} |A_i|$ is a one-way function.

2.1 Suzuki 2-groups

In [6] a generic version of the public-key cryptosystem MST_3 is described for an arbitrarily abstract non-abelian group $\mathcal{G}$. The group $\mathcal{G}$ should only satisfy the following property: $\mathcal{G}$ has a nontrivial center $\mathcal{Z}$ such that $\mathcal{G}$ does not split over $\mathcal{Z}$, i.e. there is no subgroup $\mathcal{H} < \mathcal{G}$ with $\mathcal{H} \cap \mathcal{Z} = 1$ such that $\mathcal{G} = \mathcal{Z} \cdot \mathcal{H}$. Moreover, we assume that the order of $\mathcal{Z}$ is sufficiently large so that exhaustive search problems are computationally infeasible in $\mathcal{Z}$.

The Suzuki 2-groups have been suggested for use in a possible realization of the generic version of MST_3. On one hand, due to their structure, the Suzuki 2-groups allow one to study the security of the system, and on the other hand they possess a simple presentation allowing an efficient implementation of the scheme. Before we present a new version of MST_3 using the Suzuki 2-groups in the next section, we describe for the sake of completeness this special class of 2-groups.

To begin with, we recall some basic facts about finite p-groups, where p denotes a prime number. A finite group $\mathcal{G}$ of order a power of p is called a p-*group*, i.e. $|\mathcal{G}| = p^n$ for a certain positive integer n. The least common multiple of the orders of the elements of $\mathcal{G}$ is called the *exponent* of $\mathcal{G}$. An abelian (commutative) p-group $\mathcal{G}$ of exponent p is called *elementary abelian* p-group. The set $\mathbb{Z}(\mathcal{G}) = \{z \in \mathcal{G} : \ zg = gz, \ \forall g \in \mathcal{G}\}$ is called the *center* of $\mathcal{G}$. It is well-known that $\mathbb{Z}(\mathcal{G})$ is a subgroup of order at least p for any p-group $\mathcal{G}$. The subgroup $\mathcal{G}'$ generated by all the elements of the form $x^{-1}y^{-1}xy$ with $x, y \in \mathcal{G}$ is called the *commutator subgroup* of $\mathcal{G}$. The so-called *Frattini* subgroup of $\mathcal{G}$ denoted $\Phi(\mathcal{G})$ is by definition the intersection of all the maximal subgroups of $\mathcal{G}$. If $\mathcal{G}$ is a p-group, the factor group $\mathcal{G}/\Phi(\mathcal{G})$ is elementary abelian. In particular, if $\mathcal{G}$ is a 2-group, $\Phi(\mathcal{G}) =< g^2 | g \in \mathcal{G} >$. Finally, an element of order 2 in a group is called an *involution*.

Formally a *Suzuki 2-group* is defined as a nonabelian 2-group with more than one involution, having a cyclic group of automorphisms which permutes its involutions transitively. This class of 2-groups was studied and characterized by G. Higman [3]. In particular, in any Suzuki 2-group $\mathcal{G}$ we have $\mathbb{Z}(\mathcal{G}) = \Phi(\mathcal{G}) = \mathcal{G}' = \Omega_1(\mathcal{G})$, where $\Omega_1(\mathcal{G}) =< g \in \mathcal{G} : g^2 = 1 >$ and $|\mathbb{Z}(\mathcal{G})| = q = 2^m, \ m > 1$. It is shown in [3] that the order of $\mathcal{G}$ is either q^2 or q^3. Thus all the involutions of $\mathcal{G}$ are in the center of $\mathcal{G}$, therefore $\mathbb{Z}(\mathcal{G})$ and the factor group $\mathcal{G}/\Phi(\mathcal{G})$ are elementary abelian. Consequently, all elements not in $\mathbb{Z}(\mathcal{G})$ have order 4, i.e. $\mathcal{G}$ is of exponent 4. It is known that $\mathcal{G}$ has an automorphism ξ of order $q-1$ cyclically permuting the involutions of $\mathcal{G}$ [3], (see also [5]).

In our realization of MST_3 we only consider the class of Suzuki 2-groups having order q^2. Using Higman's notation a Suzuki 2-group of order q^2 will be denoted by $A(m, \theta)$. Let $q = 2^m$ with $3 \leqslant m \in \mathbb{N}$ such that the field $\mathbb{F}_q$ has a nontrivial automorphism θ of odd order. This implies that m is not a power of 2. The groups $A(m, \theta)$ can be defined as matrix groups.

In fact, if we define
$$\mathcal{G} := \{S(a, b) \mid a, b \in \mathbb{F}_q\},$$
where
$$S(a, b) = \begin{pmatrix} 1 & a & b \\ 0 & 1 & a^\theta \\ 0 & 0 & 1 \end{pmatrix}$$
is a 3×3 -matrix over $\mathbb{F}_q$, then it is shown that the group $\mathcal{G}$ is isomorphic to $A(m, \theta)$. Thus $\mathcal{G}$ has order q^2 and we have
$$\mathcal{Z} := \mathbb{Z}(\mathcal{G}) = \Phi(\mathcal{G}) = \mathcal{G}' = \Omega_1(\mathcal{G}) = \{S(0, b) \mid b \in \mathbb{F}_q\}.$$
As the center $\mathbb{Z}(\mathcal{G})$ is elementary abelian of order q, it can be identified with the additive group of the field $\mathbb{F}_q$. Also the factor group $\mathcal{G}/\Phi(\mathcal{G})$ is an elementary abelian group of order q. It is then easily verified that the multiplication of two elements in $\mathcal{G}$ is given by the rule:
$$S(a_1, b_1)S(a_2, b_2) = S(a_1 + a_2 \ , \ b_1 + b_2 + a_1 a_2^\theta). \tag{2.3}$$

In this matrix form representation the Suzuki 2-groups $A(m, \theta)$ can be considered as subgroups of the *general linear* group $GL(3, q)$ over $\mathbb{F}_q$.

It has been shown in [3] that the groups $A(m, \theta)$ and $A(m, \phi)$ are isomorphic if and only if $\phi = \theta^{\pm 1}$.

For any $0 \neq \lambda \in \mathbb{F}_q$ the matrix

$$\Lambda = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda^{\theta+1} \end{pmatrix}$$

induces an automorphism of $A(m, \theta)$. And Λ acts on $A(m, \theta)$ according to the rule

$$\Lambda^{-1} S(a, b) \Lambda = S(a\lambda, b\lambda^{\theta+1}).$$

If $\lambda = \phi$ is a primitive element in $\mathbb{F}_q$, then Λ has order $q - 1$ and permutes cyclically the $q - 1$ involutions in the center of $A(m, \theta)$.

3 Public key cryptosystem MST_3 on Suzuki 2-groups

Notation

- From now on let $\mathcal{G} := A(m, \theta)$ be a Suzuki 2-group defined on $\mathbb{F}_q$ with $q = 2^m$.
- As $\mathcal{Z} = \mathbb{Z}(\mathcal{G})$ is an elementary abelian 2-group of order q, we may view $\mathcal{Z}$ as a vector space of dimension m over $\mathbb{F}_2$. Therefore, the automorphism group of $\mathcal{Z}$ is the general linear group $GL(m, 2)$, (i.e. the group is formed by all $m \times m$ invertible matrices over $\mathbb{F}_2$). Denote $Aut(\mathcal{Z}) := GL(m, 2)$. If $z = S(0, b) \in \mathcal{Z}$ and $\varphi \in Aut(\mathcal{Z})$, then the action of φ on z is defined by $z^{\varphi} := S(0, b^{\varphi})$.
- Let $g = S(x, y) \in \mathcal{G}$. We denote $g_{.a} := x$ and $g_{.b} := y$.

Remark 3.1 *Let* f *be any homomorphism from* $\mathcal{G}$ *to* $\mathcal{Z}$*. Let* $N = Ker(f)$*. Then* N *is normal subgroup of* $\mathcal{G}$ *and* $\mathcal{G}/N \cong f(\mathcal{G}) \subseteq \mathcal{Z}$*. So, the factor group* $\mathcal{G}/N$ *is abelian. As the commutator group* $\mathcal{G}' = \mathcal{Z}$ *we have* $N \geqslant \mathcal{Z}$*. It follows that* $f(z) = 1$ *for every* $z \in \mathcal{Z}$*.*

Key generation

Select a large group $\mathcal{G}$ as described above and generate

1. a factorizable logarithmic signature $\beta = [B_1, \ldots, B_s] := (b_{ij})$ of type $(r_1, \ldots, r_s)$ for $\mathcal{Z}$.
2. a random cover $\alpha = [A_1, A_2, \ldots, A_s] := (a_{ij})$ of the same type as β for a certain subset $\mathcal{J}$ of $\mathcal{G}$ such that $A_1, \ldots, A_s \subseteq \mathcal{G} \setminus \mathcal{Z}$. The elements in each block $A_i = [a_{i,1}, a_{i,2}, \ldots, a_{i,r_i}]$ satisfy the following conditions:
   (i) $a_{(ij_1).a} \neq a_{(ij_2).a}$, for $j_1 \neq j_2$. This is equivalent to say that $a_{(ij_1)}$ and $a_{(ij_2)}$ are not in the same coset of $\mathcal{Z}$.
   (ii) $\sum_{j=1,\ldots,r_i} a_{(ij).a} = 0$. The meaning of this condition will be obvious when we discuss the security of the system in the subsequent section.

Further select

3. $t_0, t_1 \ldots, t_s \in \mathcal{G} \setminus \mathcal{Z}$.
4. a homomorphism $f : \mathcal{G} \longrightarrow \mathcal{Z}$

and compute

5. $\gamma = (h_{ij}), \quad h_{ij} = t_{i-1}^{-1}.a_{ij}.f(a_{ij}).b_{ij}.t_i$

Then $\alpha = (a_{ij})$ and $\gamma = (h_{ij})$ are the public key. The items $\beta = (b_{ij})$, $t_0, \ldots, t_s$, and f are the private key.

Encryption

Input: A message $x \in \mathcal{Z}$ and the public key α and γ.

Output: A ciphertext (y_1, y_2) of the message x.

1. choose a random $R \in \mathbb{Z}_{|\mathcal{Z}|}$ and compute
2. $y_1 = \breve{\alpha}(R).x$,
   $y_2 = \breve{\gamma}(R).x = t_0^{-1}.\breve{\alpha}(R).f(\breve{\alpha}(R)).\breve{\beta}(R).t_s.x$

Decryption

Input: A ciphertext pair (y_1, y_2) and the private key $\beta = (b_{ij}),\ t_0, \ldots, t_s,\ f$.

Output: The message $x \in \mathcal{Z}$ that corresponds to the ciphertext (y_1, y_2).

1. Using the fact that $f(y_1) = f(\breve{\alpha}(R))$ (from Remark 3.1) compute
   $\breve{\beta}(R) = f(\breve{\alpha}(R))^{-1}.y_1^{-1}.t_0.y_2.t_s^{-1} = f(y_1)^{-1}.y_1^{-1}.t_0.y_2.t_s^{-1}$
2. Recover R from $\breve{\beta}(R)$ which is efficiently computable as β is factorizable. By computing $\breve{\alpha}(R)$ we then recover x from y_1.

Specification of the homomorphism f

For the realization of the cryptosystem MST_3 we use the following class of homomorphisms. Let $g = S(g_{.a}, g_{.b}) \in \mathcal{G}$, and let $\sigma \in \mathrm{Aut}(\mathcal{Z}) := \mathrm{GL}(m, 2)$. Define

$$\begin{aligned} &f : \mathcal{G} \longrightarrow \mathcal{Z} \\ &f(g) := S(0, g_{.a}^{\sigma}). \end{aligned}$$

Then f is a homomorphism from $\mathcal{G}$ to $\mathcal{Z}$.

The MST_3 as just described for the Suzuki 2-groups can be generalized, of course, for many other classes of finite groups, for example, the class of special p-groups. An interesting class of p-groups, also dubbed Suzuki p-groups, for odd primes p, see [1], may be viewed as a natural candidate for the underlying groups of MST_3.

The encryption method of MST_3 as described above is a randomized encryption. However, if we consider $\mathbb{Z}_{|\mathcal{Z}|}$ as the message space and encrypt a message $z \in \mathbb{Z}_{|\mathcal{Z}|}$ by computing

$$(y_1, y_2) = (\breve{\alpha}(z), \breve{\gamma}(z))$$

as ciphertext, we obtain a non-randomized encryption. It is worth noting that the non-randomized encryption can be set up within the framework of the randomized encryption method: replace R by z and take $x = 1_{\mathcal{Z}}$.

To make the discussion of the cryptanalysis of the scheme in the subsequent sections simpler, we only consider the non-randomized encryption.

4 Attack on private key

In this section we investigate various types of possible direct attacks on the private key of MST_3. We aim to find lower bounds on the workload with respect to those attacks. It turns out that those bounds have a very large size in terms of the order of the groups used.

4.1 Logarithmic signatures for $\mathcal{Z}$ and their two sided transformations

First we remark that if the adversary attempts to extract information about $\beta = (b_{ij})$, a main part of the private key, it is sufficient for him to obtain a logarithmic signature β' *equivalent* to β, i.e. any β' which is a *sandwich* transform of β. A stronger result in [10] shows that it is even sufficient for the adversary to break the system if he is able to determine a logarithmic signature β^* for $\mathcal{Z}$ such that

$$\breve{\beta}^*(x) = \breve{\beta}(x).c \tag{4.4}$$

for all $x \in \mathbb{Z}_{|\mathcal{Z}|}$, where $c \in \mathcal{Z}$ is a fixed element. For example, if $\beta^* = [B_1^*, \cdots, B_s^*]$ with $B_i^* = z_{i-1}^{-1} B_i z_i$ is a two sided transformation of β with $z_0, z_1, \cdots, z_s \in \mathcal{Z}$, then $\breve{\beta}^*(x) = \breve{\beta}(x).c$, where $c = z_0.z_s$.

The result shows a fact relevant to the way of counting the number of elements t_i used in generating γ. In fact, if we replace t_i by $t_i^* = t_i.z_i$, for $z_i \in \mathcal{Z}$, $i = 0, \ldots, s$, we obtain a β^* such that $\breve{\beta}^*(x) = \breve{\beta}(x). \prod_{i=0}^{s} z_i$. Consequently, the adversary only needs to know the cosets of $\mathcal{Z}$ in $\mathcal{G}$ with coset representatives t_i's. Then (s)he can use any coset representative $t_i^* = t_i.z_i$ in place of t_i. Hence, in the security analysis of the system, it suffices to determine the cosets of t_i with respect to $\mathcal{Z}$ and not the element t_i itself.

We call a logarithmic signature β^* for $\mathcal{Z}$ satisfying (4.4) a *translation* of β.

Definition 4.1 *Let* $\mathcal{K} = [\beta, f, t_0, \ldots, t_s]$ *be a private key for* MST_3. *We say that key* $\mathcal{K}' = [\beta', f, t_0', \ldots, t_s']$ *is an* equivalent *to* $\mathcal{K}$ *if* β' *is a translation of* β *and* $t_i' = t_i.z_i$ *for some* $z_i \in \mathcal{Z}$ *and all* $i \in \{0, \ldots, s\}$.

Our aim is to prove lower bounds on the work effort required for recovering an equivalent private key. The workload is measured in terms of the size of the involved groups and we will apply heuristic and algebraic methods to this analysis.

Now, the adversary attempts to extract information about the private key from the public knowledge of $\alpha = (a_{ij})$ and $\gamma = (h_{ij})$.

By this attack, as adversary, we try to construct a key $\mathcal{K}' = [\beta', f, t'_0, \ldots, t'_s]$ equivalent to the private key $\mathcal{K} = [\beta, f, t_0, \ldots, t_s]$. We first build an equation with unknowns involving information about the private key and then investigate the complexity of solving this equation. For this purpose we particularly exploit the operation (multiplication) in the underlying Suzuki 2-groups.

4.2 Building an equation

For convenience recall that

- $S(a_1, b_1)S(a_2, b_2) = S(a_1 + a_2\ ,\ b_1 + b_2 + a_1 a_2^{\theta})$
- $t_i \in \mathcal{G},\ \ t_i = S(t_{(i).a}\ ,\ t_{(i).b}),\ \ \alpha := (a_{ij}),\ \ a_{ij} = S(a_{(i,j).a}\ ,\ a_{(i,j).b})$.
- $g = S(g_{.a}, g_{.b}) \in \mathcal{G},\ \ f(g) := S(0, g^{\sigma}_{.a})$ where $\sigma \in Aut(\mathcal{Z}) = GL(m, 2)$.

Further recall that $g\mathcal{Z} = h\mathcal{Z}$ in $\mathcal{G}$ with $g = S(x_1, x_2)$ and $h = S(y_1, y_2)$, if and only if $x_1 = y_1$.

We start with

$$\gamma = (h_{ij}) = (t_{i-1}^{-1}\ a_{ij}\ b_{ij}\ f(a_{ij})\ t_i) = (S(h_{(i,j).a}\ ,\ h_{(i,j).b}))$$

and focus on one block of γ. W.l.o.g., let us consider the first block. The elements in this block are $h_{11}, h_{12}, \cdots, h_{1r_1}$. Let $J \subseteq \{1, \ldots, r_1\}$ be a subset such that $|J|$ is even. Then, if we sum up the elements of the first block having indices in J, we obtain the following two expressions corresponding to the ".a part" and ".b part" of the sum.

$$\sum_{j\in J,\ |J| even} h_{(1,j).a} = \sum_{j\in J} a_{(1,j).a} \tag{4.5}$$

$$\begin{aligned}\sum_{j\in J,\ |J| even} h_{(1,j).b} &= \sum_{j\in J} a_{(1,j).b} + \sum_{j\in J} b_{(1,j).b} + \sum_{j\in J} a^{\sigma}_{(1,j).a} \\ &\quad + t_{(0).a}\cdot \sum_{j\in J} a^{\theta}_{(1,j).a} + t^{\theta}_{(1).a}\cdot \sum_{j\in J} a_{(1,j).a}\end{aligned} \tag{4.6}$$

Adding $\sum_{j\in J} a_{(1,j).b}$ to (4.6) results in equation

$$\begin{aligned}\sum_{j\in J} a_{(1,j).b} + \sum_{j\in J} h_{(1,j).b} &= \sum_{j\in J} b_{(1,j).b} + \sum_{j\in J} a^{\sigma}_{(1,j).a} \\ &\quad + t_{(0).a}\cdot \sum_{j\in J} a^{\theta}_{(1,j).a} + t^{\theta}_{(1).a}\cdot \sum_{j\in J} a_{(1,j).a}\end{aligned} \tag{4.7}$$

Note that the left side of Equation (4.7) is known.

From $h_{(1,1).a} = t_{(0).a} + a_{(1,1).a} + t_{(1).a}$ we obtain

$$t_{(0).a} = h_{(1,1).a} + t_{(1).a} + a_{(1,1).a}$$

Replacing $t_{(0).a}$ in (4.7) yields

$$\begin{aligned}\sum_{j\in J,\ |J| even} (a_{(1,j).b} + h_{(1,j).b}) = & \sum_{j\in J} b_{(1,j).b} + \sum_{j\in J} a^{\sigma}_{(1,j).a} \\ & + \Big(a_{(1,1).a} + t_{(1).a} + h_{(1,1).a}\Big). \sum_{j\in J} a^{\theta}_{(1,j).a} \\ & + t^{\theta}_{(1).a}. \sum_{j\in J} a_{(1,j).a}\end{aligned}$$

Considering $t_{(1).a}$ as an unknown we end up with a trinomial of the form

$$At^{\theta}_{(1).a} + Bt_{(1).a} + X \ = \ 0 \tag{4.8}$$

where

$$\begin{aligned}A &= \sum_{j\in J} a_{(1,j).a} \\ B &= \sum_{j\in J} a^{\theta}_{(1,j).a} \\ X &= \sum_{j\in J} a_{(1,j).b} + \sum_{j\in J} h_{(1,j).b} + \sum_{j\in J} b_{(1,j).b} + \sum_{j\in J} a^{\sigma}_{(1,j).a} \\ & \quad + (a_{(1,1).a} + h_{(1,1).a}). \sum_{j\in J} a^{\theta}_{(1,j).a}\end{aligned}$$

We should remark that the term $(t_{(1).a})^{\theta}$ in the trinomial expresses the action of θ on element $t_{(1).a} \in \mathbb{F}_q$. Since θ is an automorphism of $\mathbb{F}_q$ with $q = 2^m$, it can be written as a power of the Frobenius automorphism $\phi :\ a \to a^{\phi} = a^2$ of $\mathbb{F}_q$. Thus the term $(t_{(1).a})^{\theta}$ becomes $(t_{(1).a})^{2^n}$, if $\theta = \phi^n$ for some $1 \leqslant n < m$.

Note that A and B are known, but the term X contains two unknown sums $\sum_{j\in J} b_{(1,j).b}$, and $\sum_{j\in J} a^{\sigma}_{(1,j).a}$.

4.3 Analysis of the equation

The aim of the adversary is to extract information about β. As a matter of the structure of the system (s)he needs to determine $t_{(1).a}$. As in Equation (4.8) the value of X is unknown, the adversary has to guess a value for $t_{(1).a}$. There are $(q - 1)$ possible choices for $t_{(1).a}$.

Having guessed a value for $t_{(1).a}$, the adversary can compute a corresponding value for X from Equation (4.8). In particular, (s)he computes

$$C_J \ := \sum_{j\in J} b_{(1,j).b} + \sum_{j\in J} a^{\sigma}_{(1,j).a}. \tag{4.9}$$

It is important to note that in (4.9) both sums $\sum_{j\in J} b_{(1,j).b}$, $\sum_{j\in J} a^{\sigma}_{(1,j).a}$ remain unknown. For the sake of simplicity define

$$\begin{aligned} b_J &:= \sum_{j\in J} b_{(1,j).b} \\ a^{\sigma}_J &:= \sum_{j\in J} a^{\sigma}_{(1,j).a}. \end{aligned} \tag{4.10}$$

Thus

$$C_J = b_J + a^{\sigma}_J, \tag{4.11}$$

where the values of b_J and a^{σ}_J are not determined. Note that we have to determine σ to recover values b_J and thus gain partial information about β. On the other hand, knowing β would lead to reconstructing σ.

4.3.1 Attack on b_J

By this attack the adversary seeks to determine a value for b_J in order to get an equation of the form $a^{\sigma}_J = C_J - b_J$ for σ. Note here that a_J is known. (S)he will try constructing a system of those linearly independent equations and then solving the system to determine σ. Now, as the elements in the first block $B_1 = [b_{11}, \ldots, b_{1r_1}]$ of β are not known, (s)he needs to guess a value for b_J for a given even subset J. As each b_J can take on any value from $\mathbb{F}_q$, where $q = 2^m$, and as the adversary needs at least m equations to reconstruct σ, this approach leads to a complexity of size $\mathcal{O}(q^m)$. Obviously, this type of brute force attack is not feasible as q is large.

4.3.2 Attack on a_J

We describe a more subtle and involved attack using Equation (4.11) on the first block of γ. The attack is described by the following algorithm.

Algorithm 4.2 *(Attack on a_J)*

(i) Determine subsets $J \subseteq \{1, \ldots, r_1\}$ of even size such that $a_J = 0$ and collect equations $b_J = C_J$.

(ii) Try to solve a system of equations from (i) for a set of unknown $D_1 \subseteq B_1$.

(iii) Let $D_1 = \{b_{1j_1}, \ldots, b_{1j_t}\}$. Use the $b_{1j_i} \in D_1$ from step (ii) to build non-trivial equations of the form $a^{\sigma}_{J^*} = d_{J^*}$, where $d_{J^*} := C_{J^*} - b_{J^*}$ is known and $J^* \subseteq \{j_1, \ldots, j_t\}$ is a subset of even size. Then solve the system of these equations to determine σ.

$\llcorner$

We observe that in order to apply this attack the block size of B_1 should satisfy: $r_1 > m$. If this is not the case, we have to fuse block B_1 and $B_2, \ldots, B_\ell$ (i.e. $B_1 \otimes B_2 \otimes \ldots \otimes B_\ell$), to form a larger block satisfying the condition. So, for the rest of the analysis of Algorithm 4.2 we implicitly assume that $r_1 > m$.

Before we go into detailed analysis of Algorithm (4.2), it is worth mentioning that if $a_J = 0$ then $a_J^{\sigma} = 0$. An equation $a_J = 0$ does not give any information about σ, however it does yield an equation for b_J, namely $b_J = C_J$.

We now examine the complexity of the three steps of Algorithm (4.2).

(i) As $a_{(1j).a}$'s are known, the best known efficient way of determining $a_J = 0$ for a certain subset J is to use the birthday attack. More precisely, take two disjoint random subsets J_1 and J_2 of $\{1, \ldots, r_1\}$ such that $|J_1 \cup J_2|$ is even and check if $a_{J_1} = a_{J_2}$. If yes, an $a_J = 0$ is found, where $J = J_1 \cup J_2$. Such a subset J gives rise to an equation $b_J = C_J$. Finding a subset J with $a_J = 0$ by the birthday attack has a complexity of size roughly $\mathcal{O}(q^{1/2})$. Note that in step (i) each even subset J has size at least four, this is because all elements in each block of α belong to distinct cosets modulo $\mathcal{Z}$, i.e $a_{(1,j).a} \neq a_{(1,h).a}$ for $h \neq j$. Of course, the assumption $\sum_{j \in \{1,\ldots,r_1\}} a_{(1,j).a} = 0$ is taken into account. We discuss this condition in the remark below.

(ii) Let denote $\mathcal{P} = \{J_0 = \emptyset, J_1, \ldots, J_w\}$ where $J_i \subseteq \{1, \ldots, r_1\}$ is an even subset with $|J_i| \geqslant 4$ such that $a_{J_i} = 0$ for $i \geqslant 1$. Let $\bigcup_{i=0}^{w} J_i = \{j_1, \ldots, j_t\}$. Each subsum $a_J = 0$ from step (i) corresponds to an equation $b_J = C_J$. The unknowns of these equations are elements $b_{1j_1}, \ldots, b_{1j_t}$ of B_1. Let denote

$$E_{\mathcal{P}} = \{b_J = C_J : J \in \mathcal{P}\}.$$

Since there are t unknowns, we can view the coefficients of each equation in $E_{\mathcal{P}}$ as a vector in $\mathbb{F}_{2^t}$, viewed as a vector space of dimension t over $\mathbb{F}_2$. Each such 0-1 vector has an even Hamming weight of size at least 4. Any linear combination of such two vectors gives rise to a vector corresponding to a subsum $a_J = 0$ with $J \in \mathcal{P}$ and hence to an equation in $E_{\mathcal{P}}$. In other words, the coefficient vectors of the equations in $E_{\mathcal{P}}$ span a linear subspace V of $\mathbb{F}_{2^t}$, where each non-zero vector of V has a weight at least 4. And therefore, the dimension of V is at most $t - 3$. This is equivalent to say that by using elementary row operations the coefficient matrix, say M, of any system of equations from $E_{\mathcal{P}}$ will be transformed into a matrix of row echelon form, for which each row necessary has weight at least 4. Hence, such a system of equations gives rise to at least 3 parameters that can be freely chosen, i.e. the rank of M, denoted $\mathsf{rank}(M)$, is at most $t - 3$. Since each parameter can take on any value from $\mathbb{F}_q$, solving equations for $b_{1j} \in D_1$ in this step requires a complexity of size at least $\mathcal{O}(q^3)$. Having an accurate estimate of $\mathsf{rank}(M)$ appears to be a difficult problem. This is because the rank of M depends on the set $\mathcal{P}$, which in turn depends on the random values of $a_{(1,i_1).a}, \ldots, a_{(1,i_t).a}$.

Note that $t \geqslant 4$. If $t = 4$, we have $\mathsf{rank}(M) = t - 3 = 1$. This fact is easy to see , since $J^* = \{j_1, j_2, j_3, j_4\}$ is the only one non-empty subset with $a_J = 0$. Consequently $b_{1,j_1} + b_{1,j_2} + b_{1,j_3} + b_{1,j_4} = C_J$ is the only possible equation with 4 unknowns we can obtain.

If $t > 4$, we can prove an even stronger bound that $\mathsf{rank}(M) \leqslant t - 4$. As above we denote V the linear subspace of $\mathbb{F}_{2^t}$ spanned by the coefficient vectors of the equations in $E_{\mathcal{P}}$. If any vector of V has weight at least 6, the dimension of V is at most $t-5$. And therefore $\mathsf{rank}(M) \leqslant t-5 < t-4$. So, we assume that V contains a vector v of weight 4. Without loss of generality, we can assume that v is of the form $v = 111100\ldots0$ (just by renaming the unknowns). Consider $w_4 = 111110\ldots0 \in \mathbb{F}_{2^t}$. Let $w_1 = 1000\ldots0$, $w_2 = 0100\ldots0$, $w_3 = 0010\ldots0$. Then $w_1, w_2, w_3, w_4 \notin V$. Let W be the subspace of $\mathbb{F}_{2^t}$ spanned by w_1, w_2, w_3, w_4. Then W has dimension 4. It can be checked that $x + v$ has weight at most 3 for 14 non-zero vectors $x \in W$, i.e. $x \notin V$, except for $x = y = 000110\ldots0 \in W$. But $y \notin V$, as its weight is 2. So we have $W \cap V = \{0\}$. Hence the dimension of V is at most $t - 4$. Consequently, $\mathsf{rank}(M) \leqslant t - 4$. In order to continue the attack we need to guess the values for at least 4 unknowns $b_{1j} \in \mathbb{F}_q$. Therefore the complexity of step (ii) in this case is at least $\mathcal{O}(q^4)$.

(iii) Let $D_1 = \{b_{1j_1}, \ldots, b_{1j_t}\}$ be the subset determined after step (ii). In order to be able to recover $\sigma \in GL(m, 2)$ it is necessary that $t \geqslant m$. Using elements in D_1 the adversary can construct non-zero subsum $a_J^\sigma = C_J - b_J \neq 0$ from Equation (4.9) and tries to solve such a system of equations to recover σ. This can be done in polynomial time.

Note that $t \geqslant m > 4$. We record the result of this attack in the following proposition.

Proposition 4.3 *The complexity required to recover a key equivalent to the private key* $[\beta, f, t_0, \ldots, t_s]$ *by using Algorithm 4.2 amounts to a size at least* $\mathcal{O}(q^5.q^{1/2})$.

This complexity is composed by

- the complexity $\mathcal{O}(q)$ of selecting a correct value for $t_{(1).a}$ in trinomial (4.8),
- the complexity $\mathcal{O}(q^{1/2})$ of the birthday attack in step (i) and the complexity of size at least $\mathcal{O}(q^4)$ of solving equations for $b_{1,j}$'s in step (ii).

It is a challenging open problem to determine a better lower bound on the workload to recover the private key of the system. The task appears to be difficult.

Remark 4.4 *We observe that the upper bound for the rank of the matrix* M *obtained in step (ii) above is far away from its actual value, since, for the reason of simplicity of the discussion, we do not make any restriction on* a_{1j} *'s, i.e. in the argumentation we freely use all possible values for* a_{1j} *'s, when we estimate the dimension of* V. *Evidently, the dimension of* V *depends on the choice of* a_{1j} *'s. One can expect that the dimension of* V *is much smaller and so is the rank of* M. *Therefore the complexity of the attack on* a_J *is much higher than* $\mathcal{O}(q^5.q^{1/2})$. *We conjecture that the values* $t - \mathsf{rank}(M) - 1$ *increase in proportion to the growth of* t *(i.e.* $\mathsf{rank}(M)$ *becomes proportionally smaller, when* t *becomes larger).*

Remark 4.5 *In step (ii) of Algorithm 4.2 the assumption* $\sum_{j \in \{1,\ldots,r_1\}} a_{(1,j).a} = 0$ *is taken into account. If this condition is removed, we shall have* $\sum_{j \in \{1,\ldots,r_1\}} a_{(1,j).a} \neq 0$ *in general. Suppose that we guess a value for* $u = \sum_{j \in \{1,\ldots,r_1\}} b_{(1,j).b}$. *This can be done with the complexity* $\mathcal{O}(q)$. *Consequently, each subsum* $a_J = 0$ *obtained from the birthday attack likely yields* $C_K - (u - b_J) = a_K = \sum_{j \in K} a_{(1,j).a} \neq 0$, *where* $K = \{1, \ldots, r_1\} \setminus J$. *Each* $a_K \neq 0$ *corresponds to a non-trivial linear equation for* σ. *So, if the adversary would collect* m *linearly independent equations, (s)he could reconstruct* σ, *as in step (iii). In this case the complexity of recovering a key equivalent to the private key* $[\beta, f, t_0, \ldots, t_s]$ *would reduce to* $\mathcal{O}(q^2.q^{1/2})$.

4.3.3 Combined Attack on $\mathtt{b}_{\mathtt{J}}$ and $\mathtt{a}_{\mathtt{J}}$

We can envisage a further method of reconstructing σ from equation $a_J^{\sigma} + b_J = C_J$. Two main steps of the following algorithm describe this attack.

Algorithm 4.6 *(Attack on b_J and a_J)*

(a) Construct $2m$ linearly independent vectors of size $2m$ over $\mathbb{F}_2$ to form an $2m \times 2m$ regular binary matrix A. Each row of A is of the form $a_J \| b_J$, ($\|$ denotes the concatenation), where a_J and b_J are considered as vectors of length m over $\mathbb{F}_2$.

(b) Let M denote the $2m \times m$ matrix, whose rows are C_J. Observe that M is known after $t_{(1).a}$ has been chosen. Compute a $2m \times m$ binary matrix X such that $A.X = M$, i.e. $X = A^{-1}.M$.

$\llcorner$

We are making a close look at Algorithm (4.6). We write

$$X := \begin{pmatrix} \sigma^* \\ Y \end{pmatrix}$$

and σ^* and Y are $m \times m$ binary matrices. First observe that any matrix A constructed in step (a) yields a matrix $X = A^{-1}.M$, as M is known. Each row of A takes on a value $a_J \| b_J$ corresponding to an even index subset J. The first part a_J can be computed, because $a_{(1j).a}$'s are known, but we have to guess a value for b_J (an m bit vector) from the unknonws $b_{(1j)}$'s, since they are part of the private key. So, there are q possible choices for each row of A corresponding to q possible values for b_J. If all $2m$ rows of A are correctly selected (i.e. each value of b_J is guessed correctly), the matrix X will have the form

$$X := \begin{pmatrix} \sigma \\ I \end{pmatrix},$$

where I is the $m \times m$ identity matrix. This implies that the complexity of a successful reconstruction of σ (i.e. $\sigma^* = \sigma$), after all $2m$ rows of A are determined is $\mathcal{O}(q^{2m})$. In this case we have $Y = I$.

Remark 4.7 *An pertinent implication of the combined attack is the following fact. If logarithmic signature $\beta = (b_{ij})$ is of the form $\beta = (b_{ij}) = (e_{ij})^{\sigma_1}$, where (e_{ij}) are known and σ_1 is an unknown $m \times m$ regular matrix over $\mathbb{F}_2$, this attack will enable to reconstruct σ and σ_1 as well. The reason can be seen as follows. Equation (4.11) can now be written as $a_J^{\sigma} + b_J = a_J^{\sigma} + e_J^{\sigma_1} = C_J$. If in step (a) we can construct a regular $2m \times 2m$ matrix A with rows of the form $a_J \| e_J$, the matrix $X = A^{-1}.M$ obtained from step (b) will have the form*

$$X = \begin{pmatrix} \sigma \\ \sigma_1 \end{pmatrix},$$

i.e. we are able to recover σ and σ_1. We see that this is only possible because both $a_{(1j).a}$'s, $e_{(1j)}$'s are completely known.

We close the discussion of the security analysis of the direct attacks with a record of the obtained results.

Proposition 4.8 *Comparing the three attacks presented in this section, the strongest one, the Attack on a_J, provides an actual estimate of workload required for recovering a key equivalent to the private key. The workload is lower bounded by $\mathcal{O}(q^5.q^{1/2})$, where $q = \sqrt{|\mathcal{G}|}$.*

Remark 4.9 *Let $\alpha := (S(a_{(i,j).a}, a_{(i,j).b}))$ be a cover used in a set-up of MST_3 such that $a_{(i,j).a} \in H < \mathcal{Z}$, where H is a subgroup of $\mathcal{Z}$ of order $q_0 = 2^\ell$. Then the lower bound given by Proposition 4.8 becomes $\mathcal{O}(q^4.q_0^{3/2})$. The bound is obtained because in the previous analysis the number of possible choices for $t_{(1).a}$ and the workload required for the birthday attack in step (i) of Algorithm 4.2 will be reduced according to the order of H.*

5 Generation of logarithmic signature β and its factorization

In this section we describe a method of generating logarithmic signature β for the realization of MST_3 and show methods of factorization with respect to β. As the center $\mathcal{Z}$ of $\mathcal{G}$ is an abelian group, we will use the following possible transformations in generating logarithmic signature β.

5.1 Transformations of logarithmic signatures

Let $\varepsilon = [E_1, \ldots, E_\nu] := (e_{ij})$ be a logarithmic signature of type $(t_1, \ldots, t_\nu)$ for an abelian group H. We define the following transformations on ε:

T1 transform each element of ε with an automorphism φ of H

T2 fuse j blocks $E_{k_1}, \ldots, E_{k_j}$, i.e. replace blocks $E_{k_1}, \ldots, E_{k_j}$ with a new block of the form $(((E_{k_1}.E_{k_2}).E_{k_3})\ldots E_{k_j})$, where
$E_i \cdot E_j := E_i \otimes E_j = [e_{i1}e_{j1}, \ldots, e_{i1}e_{jt_j}, e_{i2}e_{j1}, \ldots, e_{i2}e_{jt_j}, \ldots, e_{it_i}e_{jt_j}]$

T3 permute the elements within each block E_i with a permutation π_i in S_{t_i}

T4 permute the blocks E_i's with a permutation $\xi \in S_\nu$ (where S_ν is symmetric group on ν symbols)

It is obvious that β obtained from ε by using transformations T_1, T_2, T_3 and T_4 is a logarithmic signature for H. If ε is tame, we can factorize with β using the knowledge of the transformations T_i in polynomial time (as shown by an algorithm presented in a subsequent section).

5.2 Algorithm for generation of β

We will describe an algorithm for generation of a logarithmic signature β for use in MST_3. For the sake of completeness we first include a description of canonical signatures for elementary abelian 2-groups, which are defined in [10]. We will identify the center $\mathcal{Z}$ of $\mathcal{G}$ with a vector space V of dimension m over $\mathbb{F}_2$.

Definition 5.1 *1. Let V be a vector space of dimension m over $\mathbb{F}_2$. Let $\mathcal{P} = K_1 \cup \cdots \cup K_v$, $|K_i| = k_i$, $\sum_{i=1}^{s} k_i = m$, be a random partition of the set $\{1, \ldots, m\}$. A logarithmic signature $\delta = [D_1, \ldots, D_v] := (d_{ij})$ for V is called canonical if for each $i \in \{1, \ldots, v\}$, block D_i has all possible 2^{k_i} vectors with bits set on the positions defined by the subset K_i and zeros elsewhere.*

2. A canonical logarithmic signature is said in standard form, if $K_1 = \{1, \ldots, k_1\}$, $K_2 = \{k_1 + 1, \ldots, k_1 + k_2\}, \ldots,$ $K_v = \{k_1 + \cdots + k_{v-1} + 1, \ldots, m\}$, and for all i and $j_1 < j_2$ it holds $\mathrm{int}(d_{ij_1}) < \mathrm{int}(d_{ij_2})$, where $\mathrm{int}(d_{ij})$ is the integer representation of the vector d_{ij} (i.e. the vectors within D_i are sorted by their integer values).

A canonical signature δ for V of type $(t_1, t_2, \ldots, t_v)$, $t_i = 2^{k_i}$, can be generated by using the following algorithm.

Algorithm 5.2 *(Generation of a canonical logarithmic signature)*

1. Select a random partition $\mathcal{P} = K_1 \cup \cdots \cup K_v$ of the set $\{1, \ldots, m\}$ with $|K_i| = k_i$.
2. For each $i \in \{1, \ldots, v\}$, construct a block D_i by taking all possible 2^{k_i} vectors in V having bits equal to 0 at positions with indices not in K_i.

The following statement is not difficult to prove, see, for instance, [10].

Proposition 5.3 *Let $\delta := (d_{ij})$ be a canonical logarithmic signature for an elementary abelian 2-group V of order 2^m. Let $\rho \in GL(m, 2)$ be a regular $m \times m$ matrix and define $\delta^* := (d_{ij}^{\rho})$. Then δ^* is a tame logarithmic signature.*

It is clear that the signature δ^* obtained from Proposition 5.3 is a transversal signature for a certain chain of subgroups $1_V = V_0 < V_1 < \cdots < V_s = V$ of V.

Moreover, it is shown in [10] that the factorization with respect to a canonical logarithmic signature will have time complexity $\mathcal{O}(1)$.

We now describe an algorithm for generating logarithmic signature β.

Algorithm 5.4 *(Generation of logarithmic signature β)*

1. Let $\varepsilon = [E_1, E_2, \ldots, E_v] := (e_{i,j})$ be the canonical logarithmic signature in standard form of type $(t_1, t_2, \ldots, t_v)$ for $\mathcal{Z}$ (viewed as an m dimensional vector over $\mathbb{F}_2$) corresponding to the partition $\{K_1, K_2, \ldots, K_v\}$ on the set $\{1, \ldots, m\}$ with $|K_i| = k_i$ and $t_i = 2^{k_i}$ (in Definition 5.1).

   Denote $\varepsilon^* = (e^*_{i,j})$ a logarithmic signature obtained from ε by filling the positions $K_1 \cup \ldots \cup K_{i-1}$ of each block E_i with random bits, $i = 2, \ldots, v$. We call ε^* a *randomized canonical* logarithmic signature.

2. [transformation T_1] Select a random matrix $\rho \in GL(m, 2)$ and compute

$$\delta = [D_1, \ldots, D_v] = (d_{i,j}) := ((e^*_{i,j})^{\rho}).$$

3. [transformation T_2]
   Select a partition $\mathcal{P} = \{P_1, \dots, P_s\}$, $0 < |P_j|$, of a set $\{1, \dots, \nu\}$, such that for each $P_j = \{i_1, \dots, i_u\}$, i.e. $|P_j| = u$, we have $i_h \neq i_\ell + 1$ for $h, \ell \in \{1, \dots, u\}$. Fuse blocks $D_{i_1}, \dots, D_{i_u}$, i.e. construct the product $C_j := (((D_{i_1}.D_{i_2}).D_{i_3} \dots D_{i_u})$. Let $\omega = [C_1, \dots, C_s] := (c_{i,j})$ be the resulting logarithmic signature of type $(r_1, \dots, r_s)$ obtained after this step.

4. [transformation T_3]
   Select random permutations $\pi_i \in S_{r_i}$, $i = 1, \dots, s$, where S_{r_i} is the symmetric group of degree r_i. Define

   $$C_i^* := C_i^{\pi_i} = [c_{i,1^{\pi_i}}, c_{i,2^{\pi_i}}, \dots, c_{i,t_i{}^{\pi_i}}],$$

   i.e. C_i^* is obtained from C_i by permuting the positions of its elements with permutation π_i. Denote $\chi = [C_1^*, \dots, C_s^*]$.

5. [transformation T_4]
   Select a random permutation $\xi \in S_s$ and define

   $$\beta = [B_1, \dots, B_s] := [C^*_{1^\xi}, \dots, C^*_{s^\xi}],$$

   i.e. β is obtained from χ by permuting the positions of its blocks with ξ.

It should be noted that in order to have an efficient factorization with respect to β created using Algorithm 5.4, we keep track of the information about matrix ρ, logarithmic signature ε^*, partition $\mathcal{P}$, and all permutations used in steps (4) and (5).

Definition 5.5 *We call β fused transversal (FT) logarithmic signature, if β is generated by Algorithm 5.4. If step (3) (i.e. fusion of blocks) of the algorithm is not applied, β is called non-fused transversal (NFT) logarithmic signature.*

5.3 Factorization with β

In this section we present algorithms for the factorization with β generated by Algorithm 5.4. We begin by proving the following useful proposition.

Proposition 5.6 *Let $\beta := [B_1, \dots, B_\nu]$ be a transversal logarithmic signature for an abelian group H. Let $\beta' := [B_1', \dots, B_s']$ be a fused logarithmic signature of H obtained by fusion of blocks of β [transformation T2]. Then β' is equivalent to a non-fused logarithmic signature β'' obtained from β by using certain permutation $\mu \in S_\nu$ on blocks B_i [transformation T4]. In other words β' and β'' induce the same function, i.e. $\breve{\beta}' = \breve{\beta}''$.*

Proof. We observe that β' is obtained from β by using the following two operations:

(a) select an appropriate permutation $\mu \in S_\nu$ and compute

$$\beta'' := [B_1'', \dots, B_\nu''] := [B_{1^\mu}, \dots, B_{\nu^\mu}].$$

(b) select a partition $\mathcal{R} = \{R_1, \ldots, R_s\}$ on the set $\{1, \ldots, \nu\}$ with $R_1 = \{1, \ldots, i_1\}, R_2 = \{i_1 + 1, \ldots, i_2\}, \ldots, R_s = \{i_{s-1} + 1, \ldots, i_s\}$ with $|R_j| = u_j$ for $j \in \{1, \ldots, s\}$. Fusing the blocks of β'' according to this partition yields the logarithmic signature $\beta' := [B'_1, \ldots, B'_s]$ of type $(r_1, \ldots, r_s)$ with $B'_j = ((B''_{i_{j-1}+1}.B''_{i_{j-1}+2}) \ldots B''_{i_j})$, where $r_j = |B''_{i_{j-1}+1}|.|B''_{i_{j-1}+2}| \ldots |B''_{i_j}|$ for $j = 1, \ldots, s$ and $i_0 = 0$.
(i.e. each block B'_i is obtained by fusing certain consecutive blocks of β''.)

It is clear that β' is equivalent to β''. □

Remark 5.7 *Let $\mathcal{P} = \{P_1, \ldots, P_s\}$ be a partition on the set $\{1, \ldots, \nu\}$ with $P_1 = \{i_{1,1}, \ldots, i_{1,u_1}\}$, $P_2 = \{i_{2,1}, \ldots, i_{2,u_2}\}, \ldots, P_s = \{i_{s,1}, \ldots, i_{s,u_s}\}$ from the step (3) of Algorithm 5.4. The permutation $\mu \in S_\nu$ from Proposition 5.6 is given by*

$$\begin{pmatrix} 1 & 2 & \ldots & u_1 & u_1 + 1 & \ldots & u_1 + u_2 & \ldots & (u_1 + u_2 + \ldots + u_s) \\ i_{1,1} & i_{1,2} & \ldots & i_{1,u_1} & i_{2,1} & \ldots & i_{2,u_2} & \ldots & i_{s,u_s} \end{pmatrix}$$

and the corresponding partition is

$$\begin{aligned} \mathcal{R} := \Big\{ R_1 = \{1, 2, \ldots, u_1\}, R_2 = \{u_1 + 1, \ldots, u_1 + u_2\}, \ldots, \\ R_s = \{u_1 + \cdots + u_{s-1} + 1, \ldots, u_1 + \cdots + u_s\} \Big\}. \end{aligned}$$

An important consequence of Proposition 5.6 is the construction of the Algorithm 5.8 which allows efficient factorization with respect to the FT logarithmic signature β.

Let ε^* be the randomized canonical signature created after step (1) of Algorithm 5.4. Also let μ be the permutation with corresponding partition $\mathcal{R}$ from Remark 5.7. Then we may efficiently factorize $\breve{\beta}(x)$ using the following algorithm

Algorithm 5.8 *(Factorization with FT signature β)*

Input: y, ε^*, μ, $\mathcal{R} = \{R_1, \ldots, R_s\}$, ξ, $\pi_1, \ldots, \pi_s, \rho$.
Output: $x = x_1 \| x_2 \| \ldots \| x_s$, where $y = \breve{\beta}(x)$.

1. Compute $z = (y^{\rho^{-1}})$ and write $z = z_1 \| z_2 \| \ldots \| z_\nu$. Each z_i is of bit length k_i.
2. Factorize z with respect to ε^* by using Algorithm 5.9. Let denote $j'_1, \ldots, j'_\nu$ the indices obtained by this factorization.
3. Compute $j_\ell = j'_{\ell \mu^{-1}}$ for $\ell = 1, \ldots, \nu$.
4. According to $R_\ell = \{i_1, i_2, \ldots, i_{u_\ell}\}$ set $x'_\ell = j_{i_1} \| j_{i_2} \| \ldots \| j_{i_{u_\ell}}$ for $\ell = 1, \ldots, s$.
5. Compute $x''_\ell = (x'_\ell)^{\pi_\ell^{-1}}$ and finally $x_\ell = x''_{\ell \xi^{-1}}$ for $\ell = 1, \ldots, s$.

In the following we present an algorithm for factorization with respect to an NFT logarithmic signature. To make the description clearer we start with an algorithm for factorization with respect to a randomized canonical logarithmic signature ε^* generated in step (1) of Algorithm 5.4.

Let $x = x_1 \| x_2 \| \ldots \| x_\nu$ be a binary vector of length m, where x_i is of length k_i for $i = 1, \ldots, \nu$ and $t_i = 2^{k_i}$. Let $y = \breve{\varepsilon}^*(x)$. Write $y = y_1 \| y_2 \| \ldots \| y_\nu$, where each y_i is of bit length k_i.

In order to factorize y with respect to ε^* we have to determine indices x_i, for $i = 1, \ldots, \nu$. This can be done with the following algorithm.

Algorithm 5.9 *(Factorization with ε^*)*

Input: $y = y_1 \| y_2 \| \ldots \| y_\nu$, ε^*.
Output: $x = x_1 \| x_2 \| \ldots \| x_\nu$, where $y = \breve{\varepsilon}^*(x)$.

(F) Starting with y_ν we find an element $e^*_{\nu,j}$ in block E^*_ν such that the last k_ν bits of $e^*_{\nu,j}$ are equal to y_ν. Such $e^*_{\nu,j}$ is uniquely determined since the last k_ν bits of elements in E^*_ν form a vector space of dimension k_ν. The index j of $e^*_{\nu,j}$ in block E^*_ν determines the index x_ν.

(R) Compute $y' = y * (e^*_{\nu,j})^{-1}$ and write $y' = y'_1 \| y'_2 \| \ldots \| y'_{\nu-1}$ where each y'_i is of bit length k_i. Repeat step (F) with $y'_{\nu-1}$ for block $E^*_{\nu-1}$ to find $x_{\nu-1}$. Continue this process until x_1 is found.

Now we describe an algorithm for factorization with respect to an NFT logarithmic signature β^*.

Again, let $x = x_1 \| x_2 \| \ldots \| x_\nu$ be a binary vector of length m where x_i is of bit length k_i for $i = 1, \ldots, \nu$ and $t_i = 2^{k_i}$. Let $z = \breve{\beta}^*(x)$. Write $z = z_1 \| z_2 \| \ldots \| z_\nu$ where each z_i is of bit length k_i.

Algorithm 5.10 *(Factorization with NFT signature β^*)*

Input: $z = z_1 \| z_2 \| \ldots \| z_\nu$, β^*, ξ, $\pi_1, \ldots, \pi_\nu$, ρ.
Output: $x = x_1 \| x_2 \| \ldots \| x_\nu$, where $z = \breve{\beta}^*(x)$.

1. Using ξ^{-1}, $\pi_1^{-1}, \ldots, \pi_\nu^{-1}$ and ρ^{-1} construct ε^* from β^*.
2. Compute $y = (z^{\rho^{-1}})$ and write $y = y_1 \| y_2 \| \ldots \| y_\nu$. Each y_i is of bit length k_i.
3. Factorize y with respect to ε^* by using Algorithm 5.9. Let denote $x'_1, \ldots, x'_\nu$ the indices obtained by this factorization.
4. Compute $x''_i = (x'_i)^{\pi_i^{-1}}$ and finally $x_i = x''_{i^{\xi^{-1}}}$ for $i = 1, \ldots, \nu$.

6 Attack on ciphertexts

This section deals with an elaborated chosen plaintext attack on MST_3, when transversal logarithmic signatures are used. This is the case when β is generated by Algorithm 5.4 without applying the fusion step (3). In fact, those logarithmic signatures may essentially be viewed as those from a chain of subgroups of $\mathcal{Z}$. However, the structure of β will be changed if the the fusion step (3) is applied.

The Matrix-permutation attack developed in this section appears to be powerful, as it provides a proof of the fact that the class of non-fused transversal logarithmic signatures cannot be used in a realization of MST_3. The class of fused transversal logarithmic signatures, however, withstands the Matrix-permutation attack, as shown below.

Before we present the Matrix-permutation attack, we would like to mention a simple attack which emerges naturally from the representation of the elements in the Suzuki 2-groups.

6.1 The Basis attack

Based on the description of the scheme, the a.part of α and γ, are merely random covers for the center $\mathcal{Z}$. Note that $\mathcal{Z}$ is a vector space of dimension m over $\mathbb{F}_2$ and we also indentify $\mathcal{Z}$ with $\mathbb{F}_q$. So we call elements of $\mathcal{Z}$ vectors as well. Let denote $\alpha_{.a}$ the cover of $\mathcal{Z}$ whose blocks are formed by the a.part of α. Define $\mathcal{J} := \breve{\alpha}_{.a}(\mathbb{Z}_{|\mathcal{Z}|})$. Thus $\mathcal{J}$ is a subset of $\mathcal{Z}$ and the ratio $\rho := |\mathcal{Z}|/|\mathcal{J}|$ may be viewed as the average number of representations for each element of $\mathcal{J}$ with respect to $\alpha_{.a}$. More precisely, due to the connection between generation of random covers and the occupancy problem, see for instance [13], we can derive an approximation for the ratio ρ given by the following formula

$$\rho \approx \lambda(\frac{e^{\lambda}}{e^{\lambda}-1})$$

where $\lambda = \frac{1}{|\mathcal{Z}_1|}\prod_{i=1}^{s} r_i$, and $(r_1, \ldots, r_s)$ is the type of $\alpha_{.a}$, and $\mathcal{Z}_1$ is the smallest subgroup of $\mathcal{Z}$ containing $\mathcal{J}$. As a matter of linear algebra we may find a maximal subset of linearly independent vectors which come from all the blocks of $\alpha_{.a}$. By using the two sided transformation on $\alpha_{.a}$ we may assume that the first $s-1$ blocks contain the zero vector. The linearly independent vectors together with the zero vectors form a cover which allows an efficient factorization of a certain amount of ciphertexts created by $\breve{\alpha}_{.a}$. This amount is approximately $\frac{1}{\rho}\prod_{i=1}^{s}(k_i+1)$, where $k_i = \lceil \log_2 r_i \rceil$. Therefore the probability that a given ciphertext could be correctly decrypted is given by

$$\approx \frac{1}{\rho}\prod_{i=1}^{s}\frac{(k_i+1)}{r_i}$$

As a result, if ρ or/and r_i are increased, this probability will be decreased. So, if we select the elements of $\alpha_{.a}$ from a subspace $\mathcal{Z}_1$ of $\mathcal{Z}$ such that $\rho = |\mathcal{Z}|/|\mathcal{Z}_1|$ is large, then this simple attack becomes infeasible.

6.2 The Matrix-permutation attack on NFT-MST_3

We now present the Matrix-permutation attack on a realization of MST_3 that uses a non-fused transversal logarithmic signature β (for short we call NFT-MST_3). This strong attack is a chosen plaintext attack type, which attempts to reverse the encryption function of the system. The main idea of the Matrix-permutation attack is to construct a series of matrices and to recover permutations used in generating β that would eventually allow the adversary to decrypt any given ciphertext.

Used notation:

Let $\omega := (w_{i,j})$ be a cover of type $(r_1, \ldots, r_s)$ for $\mathcal{G}$, and let $x \in \mathbb{Z}_{|\mathcal{Z}|}$ correspond to $(j_1, \ldots, j_s) \in \mathbb{Z}_{r_1} \oplus \ldots \oplus \mathbb{Z}_{r_s}$ [see Preliminaries]. Let $\phi \in S_s$ and $\nu_\ell := \ell^\phi$ for $\ell \in \{1, \ldots, s\}$. Define

$$\breve{\omega}_{k,\phi}(x) := \prod_{i=1}^{k} w_{\nu_i, j_{\nu_i}} \tag{6.12}$$

We consider an NFT-MST$_3$ scheme. Let $[\alpha, \gamma]$ be the public key with the corresponding private key $[\beta, f, t_0, \ldots, t_s]$. Recall that $\alpha := (a_{i,j_i})$, $\beta := (b_{i,j_i})$ and $\gamma := (h_{i,j_i})$ are of type $(r_1, \ldots, r_s)$ and that ξ is a permutation used in step (5) and $\pi_1, \ldots, \pi_s$ are permutations used in step (4) of Algorithm 5.4.

Proposition 6.1 *Let α, β, γ be the covers of type $(r_1, \ldots, r_s)$ as described above. Let $x \in \mathbb{Z}_{|\mathcal{Z}|}$ correspond to $(j_1, \ldots, j_s) \in \mathbb{Z}_{r_1} \oplus \ldots \oplus \mathbb{Z}_{r_s}$ and $\nu_\ell := \ell^\xi$ for $\ell \in \{1, \ldots, s\}$. Further let $\breve{\alpha}_{\ell,\xi}(x)$, $\breve{\beta}_{\ell,\xi}(x)$, $\breve{\gamma}_{\ell,\xi}(x)$ be the values computed by Equation (6.12). Let $k_\ell := \lceil \log_2 r_\ell \rceil$. Then there exists a binary $(2m+1) \times k_{\nu_\ell}$ matrix M_{ν_ℓ} such that*

$$\left(\breve{\alpha}_{\ell,\xi}(x)_{.a} \parallel \breve{\alpha}_{\ell,\xi}(x)_{.b} + \breve{\gamma}_{\ell,\xi}(x)_{.b} \parallel 1 \right) M_{\nu_\ell} = \pi_{\nu_\ell}(j_{\nu_\ell}). \tag{6.13}$$

where "1" *is the bit set to one.*

Proof. First we show that there exists a binary $(2m+1) \times m$ matrix N_{ν_ℓ} such that

$$\left(\breve{\alpha}_{\ell,\xi}(x)_{.a} \parallel \breve{\alpha}_{\ell,\xi}(x)_{.b} + \breve{\gamma}_{\ell,\xi}(x)_{.b} \parallel 1 \right) N_{\nu_\ell} = \left(\breve{\beta}_{\ell,\xi}(x)_{.b} \right) \tag{6.14}$$

We begin with

$$\begin{aligned} \breve{\alpha}_{\ell,\xi}(x)_{.b} + \breve{\gamma}_{\ell,\xi}(x)_{.b} = {} & \sum_{i=1}^{\ell} b_{(\nu_i, j_{\nu_i}).b} + t_{(0).a} \sum_{i=1}^{\ell} a^{\theta}_{(\nu_i, j_{\nu_i}).a} + t^{\theta}_{(\nu_\ell).a} \sum_{i=1}^{\ell} a_{(\nu_i, j_{\nu_i}).a} \\ & + \sum_{i=1}^{\ell} a^{\sigma}_{(\nu_i, j_{\nu_i}).a} + C_\ell \end{aligned}$$

where $C_\ell = t_{(0).b} + t^{\theta+1}_{(0).a} + t_{(\nu_\ell).b} + t_{(0).a} t^{\theta}_{(\nu_\ell).a}$. As the elements $t_{(0).a}, t_{(\nu_\ell).a} \in \mathbb{F}_q$ are constants, the products $t_{(0).a} \sum a^{\theta}_{(\nu_i, j_{\nu_i}).a}$ and $t^{\theta}_{(\nu_\ell).a} \sum a_{(\nu_i, j_{\nu_i}).a}$ present linear mappings. Therefore there exist binary $m \times m$ matrices T_0 and T_{ν_ℓ} such that

$$\begin{aligned} t_{(0).a} \sum_{i=1}^{\ell} a^{\theta}_{(\nu_i, j_{\nu_i}).a} &= \sum_{i=1}^{\ell} a_{(\nu_i, j_{\nu_i}).a} \, T_0 \\ t^{\theta}_{(\nu_\ell).a} \sum_{i=1}^{\ell} a_{(\nu_i, j_{\nu_i}).a} &= \sum_{i=1}^{\ell} a_{(\nu_i, j_{\nu_i}).a} \, T_{\nu_\ell} \end{aligned}$$

Set

$$N_{\nu_\ell} = \begin{pmatrix} T_0 + T_{\nu_\ell} + \sigma \\ I_m \\ C_\ell \end{pmatrix}$$

where I_m is the $m \times m$ identity matrix. Now it is not difficult to check that the $(2m+1) \times m$ matrix N_{ν_ℓ} satisfies Equation (6.14). Define $\varepsilon' := (e'_{i,j})$ with $e'_{i,j} = b^{\rho^{-1}}_{i,j}$. Clearly

$$\left(\breve{\beta}_{\ell,\xi}(x)_{.b} \right)^{\rho^{-1}} = \left(\breve{\varepsilon}'_{\ell,\xi}(x)_{.b} \right)$$

Consider the linear mapping φ_ℓ defined by

$$\breve{\varepsilon}^*_{\ell,id}(x)_{.b} = \sum_{i=1}^{\ell} e^*_{(i,j_i).b} \xmapsto{\varphi_\ell} j_\ell$$

where id is identity permutation.This mapping is well defined for the class of transversal logarithmic signatures, in particular for ε^* created in Algorithm 5.4 after step (1). Note that j_ℓ is the binary representation of the index for e_{ℓ,j_ℓ} and is identical with the k_ℓ bit vector of e_{ℓ,j_ℓ} at the positions K_ℓ. Let $\varepsilon'' := (e''_{ij})$ be obtained from ε^* by applying step (4) of Algorithm 5.4. Now observe that φ_ℓ acts on $\breve{\varepsilon}''_{\ell,id}(x)_{.b}$ as follows

$$\breve{\varepsilon}''_{\ell,id}(x)_{.b} = \sum_{i=1}^{\ell} e''_{(i,j_i).b} \xmapsto{\varphi_\ell} \pi_\ell(j_\ell)$$

Applying step (5) of Algorithm 5.4 on ε'' we get ε'. Therefore φ_ℓ acts on $\breve{\varepsilon}'_{\ell,\xi}(x)_{.b}$ according to

$$\breve{\varepsilon}'_{\ell,\xi}(x)_{.b} = \sum_{i=1}^{\ell} e'_{(\nu_i,j_{\nu_i}).b} \xmapsto{\varphi_\ell} \pi_{\nu_\ell}(j_{\nu_\ell})$$

Let P_{ν_ℓ} be the $m \times k_{\nu_\ell}$ binary matrix represention of the mapping φ_ℓ. Then we can write

$$\left(\breve{\varepsilon}'_{\ell,\xi}(x)_{.b} \right) P_{\nu_\ell} = \pi_{\nu_\ell}(j_{\nu_\ell})$$

Define the matrix M_{ν_ℓ} as

$$M_{\nu_\ell} := N_{\nu_\ell} \cdot \rho^{-1} \cdot P_{\nu_\ell}$$

Then M_{ν_ℓ} is the binary matrix that satisfies Equation (6.13). □

Let $M_{\ell,p}$ denote the p-th column of the matrix M_ℓ, where $p = 1, \ldots, k_\ell$. We observe that $\pi_\ell(j_\ell)$ is a binary vector of length k_ℓ. Similarly, we denote $\pi_{\ell,p}(j_\ell)$ the p-th bit of $\pi_\ell(j_\ell)$. Using this notation and Proposition 6.1 we obtain the following

Proposition 6.2 *Let* $\nu_\ell := \ell^\xi$ *and* $M_{\nu_\ell,p}$ *be the* p*-th column of* M_{ν_ℓ} *and* $\pi_{\nu_\ell,p}(j_{\nu_\ell})$ *be the* p*-th bit of* $\pi_{\nu_\ell}(j_{\nu_\ell})$ *from Propositon 6.1. Then we have*

$$\big(\ \breve{\alpha}_{\ell,\xi}(x)_{.a} \parallel \breve{\alpha}_{\ell,\xi}(x)_{.b} + \breve{\gamma}_{\ell,\xi}(x)_{.b} \parallel 1\ \big)\ M_{\nu_\ell,p} = \pi_{\nu_\ell,p}(j_{\nu_\ell}). \tag{6.15}$$

Proposition 6.3 *Let* α, β, γ *be the covers of type* $(r_1,\ldots,r_s)$ *as described above. Let* $x \in \mathbb{Z}_{|\mathcal{Z}|}$ *correspond to* $(j_1,\ldots,j_s) \in \mathbb{Z}_{r_1} \oplus \ldots \oplus \mathbb{Z}_{r_s}$*. Further let* $\nu_\ell := \ell^\xi$ *for* $\ell \in \{1,\ldots,s\}$ *and* $k_\ell := \lceil \log_2 r_\ell \rceil$*. Then there exists a binary* $(2m+1) \times k_{\nu_\ell}$ *matrix* L_{ν_ℓ} *such that*

$$\big(\ \breve{\alpha}(x)_{.a} + \mathcal{A}_\ell \,\big\|\, \breve{\alpha}(x)_{.b} + \breve{\gamma}(x)_{.b} + \mathcal{B}_\ell \,\big\|\, 1\ \big)\, L_{\nu_\ell} = \pi_{\nu_\ell}(j_{\nu_\ell}) \tag{6.16}$$

where

$$\begin{aligned}
\mathcal{A}_\ell &:= \sum_{i=\ell+1}^{s} a_{(\nu_i,j_{\nu_i}).a} \\
\mathcal{B}_\ell &:= \sum_{i=\ell+1}^{s} \big(a_{(\nu_i,j_{\nu_i}).b} + h_{(\nu_i,j_{\nu_i}).b}\big) + \sum_{i=\ell+1}^{s} a^\theta_{(\nu_i,j_{\nu_i}).a}\big(t_{(0).a} + t_{(\nu_{i-1}).a}\big) + \\
&\quad \sum_{i=\ell+1}^{s} a_{(\nu_i,j_{\nu_i}).a}\big(t_{(\nu_i).a} + t_{(s).a}\big)^\theta
\end{aligned}$$

for $\ell \in \{1,\ldots,s-1\}$*,* $\mathcal{A}_s = \mathcal{B}_s := (0,\ldots,0)$*, and "1" is the bit set to one.*

Proof. For $\ell = s$ Equation (6.16) is obtained from Proposition 6.1.

So, from now on we assume that $\ell \in \{1,\ldots,s-1\}$.

Note that

$$\begin{aligned}
\breve{\alpha}(x)_{.a} + \sum_{i=\ell+1}^{s} a_{(\nu_i,j_{\nu_i}).a} &= \sum_{i=1}^{\ell} a_{(\nu_i,j_{\nu_i}).a} \\
\breve{\beta}(x)_{.b} + \sum_{i=\ell+1}^{s} b_{(\nu_i,j_{\nu_i}).b} &= \sum_{i=1}^{\ell} b_{(\nu_i,j_{\nu_i}).b}
\end{aligned}$$

First we show that there exists a $(2m+1) \times m$ binary matrix N_{ν_ℓ} such that

$$\left(\sum_{i=1}^{\ell} a_{(\nu_i,j_{\nu_i}).a} \,\Big\|\, \breve{\alpha}(x)_{.b} + \breve{\gamma}(x)_{.b} + \mathcal{B}_\ell \,\Big\|\, 1\right) N_{\nu_\ell} = \sum_{i=1}^{\ell} b_{(\nu_i,j_{\nu_i}).b} \tag{6.17}$$

Here we have

$$\begin{aligned}
&\breve{\alpha}(x)_{.b} + \breve{\gamma}(x)_{.b} + \mathcal{B}_\ell \\
&= \ \breve{\alpha}(x)_{.b} + \breve{\gamma}(x)_{.b} + \sum_{i=\ell+1}^{s} \big(a_{(\nu_i,j_{\nu_i}).b} + h_{(\nu_i,j_{\nu_i}).b}\big) + \\
&\quad \sum_{i=\ell+1}^{s} a^\theta_{(\nu_i,j_{\nu_i}).a}\big(t_{(0).a} + t_{(\nu_{i-1}).a}\big) + \sum_{i=\ell+1}^{s} a_{(\nu_i,j_{\nu_i}).a}\big(t_{(\nu_i).a} + t_{(s).a}\big)^\theta
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=1}^{s} b_{(\nu_i,j_{\nu_i}).b} + \sum_{i=1}^{s} a^{\sigma}_{(\nu_i,j_{\nu_i}).a} + t_{(0).b} + t^{\theta+1}_{(0).a} + t_{(s).b} + t_{(0).a} t^{\theta}_{(s).a} + \\
&\quad t_{(0).a} \sum_{i=1}^{s} a^{\theta}_{(\nu_i,j_{\nu_i}).a} + t^{\theta}_{(s).a} \sum_{i=1}^{s} a_{(\nu_i,j_{\nu_i}).a} + \sum_{i=\ell+1}^{s} b_{(\nu_i,j_{\nu_i}).b} + \sum_{i=\ell+1}^{s} a^{\sigma}_{(\nu_i,j_{\nu_i}).a} + \\
&\quad \sum_{i=\ell+1}^{s} \big(t_{(\nu_{i-1}).a} a^{\theta}_{(\nu_i,j_{\nu_i}).a} + a_{(\nu_i,j_{\nu_i}).a}\ t^{\theta}_{(\nu_i).a} + t_{(\nu_{i-1}).a} t^{\theta}_{(\nu_i).a} + t_{(\nu_{i-1}).b} + \\
&\quad t^{\theta+1}_{(\nu_{i-1}).a} + t_{(\nu_i).b} \big) + t_{(0).a} \sum_{i=\ell+1}^{s} a^{\theta}_{(\nu_i,j_{\nu_i}).a} + \sum_{i=\ell+1}^{s} a^{\theta}_{(\nu_i,j_{\nu_i}).a}\ t_{(\nu_{i-1}).a} + \\
&\quad \sum_{i=\ell+1}^{s} a_{(\nu_i,j_{\nu_i}).a}\ t^{\theta}_{(\nu_i).a} + t^{\theta}_{(s).a} \sum_{i=\ell+1}^{s} a_{(\nu_i,j_{\nu_i}).a} \\
&= \sum_{i=1}^{\ell} b_{(\nu_i,j_{\nu_i}).b} + \sum_{i=1}^{\ell} a^{\sigma}_{(\nu_i,j_{\nu_i}).a} + t_{(0).a} \sum_{i=1}^{\ell} a^{\theta}_{(\nu_i,j_{\nu_i}).a} + t^{\theta}_{(s).a} \sum_{i=1}^{\ell} a_{(\nu_i,j_{\nu_i}).a} + C_\ell
\end{aligned}
$$

where the term

$$
\begin{aligned}
C_\ell = \ & \sum_{i=\ell+1}^{s} \big(t_{(\nu_{i-1}).a} t^{\theta}_{(\nu_i).a} + t_{(\nu_{i-1}).b} + t^{\theta+1}_{(\nu_{i-1}).a} + t_{(\nu_i).b} \big) + \\
& t_{(0).b} + t^{\theta+1}_{(0).a} + t_{s.b} + t_{(0).a} t^{\theta}_{s.a}
\end{aligned}
$$

is viewed as a constant in $\mathbb{F}_{2^m}$. Therefore, Equation (6.17) becomes

$$
\left(\begin{array}{c||c||c}
\sum_{i=1}^{\ell} a_{(\nu_i,j_{\nu_i}).a} &
\begin{array}{l}
\sum_{i=1}^{\ell} b_{(\nu_i,j_{\nu_i}).b} + \sum_{i=1}^{\ell} a^{\sigma}_{(\nu_i,j_{\nu_i}).a} + \\
t_{(0).a} \sum_{i=1}^{\ell} a^{\theta}_{(\nu_i,j_{\nu_i}).a} + \\
t^{\theta}_{s.a} \sum_{i=1}^{\ell} a_{(\nu_i,j_{\nu_i}).a} + C_\ell
\end{array}
& 1
\end{array} \right) N_{\nu_\ell} = \sum_{i=1}^{\ell} b_{(\nu_i,j_{\nu_i}).b} \qquad (6.18)
$$

Because the elements $t_{(0).a}$ and $t_{(s).a}$ are constants, and θ is a linear mapping, there exist $m \times m$ matrices T_0, T_ℓ such that

$$
\begin{aligned}
t_{(0).a} \sum_{i=1}^{\ell} a^{\theta}_{(\nu_i,j_{\nu_i}).a} &= \sum_{i=1}^{\ell} a_{(\nu_i,j_{\nu_i}).a}\ T_0 \\
t^{\theta}_{(s).a} \sum_{i=1}^{\ell} a_{(\nu_i,j_{\nu_i}).a} &= \sum_{i=1}^{\ell} a_{(\nu_i,j_{\nu_i}).a}\ T_\ell
\end{aligned}
$$

Now set

$$
N_{\nu_\ell} = \begin{pmatrix} T_0 + T_\ell + \sigma \\ I_m \\ C_\ell \end{pmatrix}
$$

where I_m is the $m \times m$ identity matrix. Then it is easy to check that the $(2m+1) \times m$ matrix N_{v_ℓ} satisfies Equation (6.18). Similar to the proof of Proposition 6.1, by using

$$\left(\sum_{i=1}^{\ell} b_{(v_i, j_{v_i}).b} \right) \rho^{-1} = \sum_{i=1}^{\ell} e'_{(v_i, j_{v_i}).b} \;:=\; \breve{\varepsilon}'_{\ell,\xi}(x)_{.b}$$

and

$$\left(\; \breve{\varepsilon}'_{\ell,\xi}(x)_{.b} \; \right) P_{v_\ell} = \pi_{v_\ell}(j_{v_\ell})$$

we define

$$L_{v_\ell} := N_{v_\ell} \cdot \rho^{-1} \cdot P_{v_\ell}$$

Then L_{v_ℓ} is the binary matrix that satisfies Equation (6.16). □

We are now in a position to describe an algorithm for recovering permutations $\pi_1, \ldots, \pi_s$ by using Proposition 6.2. The algorithm delivers the permutation ξ as well.

Algorithm 6.4 *(Matrix-permutation Attack on NFT-*MST_3*: Permutation recovery)*

Input: Public key $[\alpha, \gamma]$.
Output: Permutations $[\pi_1, \ldots, \pi_s, \xi]$.

```
For $\ell \leftarrow s$ downto 1 do
  (A) Choose random plaintexts $x^{(i)} \mapsto (j_1^{(i)}, \ldots, j_s^{(i)})$, and construct vectors $y^{(i)} := \left( \breve{\alpha}_{\ell,id}(x)^{(i)}_{.a} \parallel \breve{\alpha}_{\ell,id}(x)^{(i)}_{.b} + \breve{\gamma}_{\ell,id}(x)^{(i)}_{.b} \parallel 1 \right)$, as in Proposition 6.2. Define $n_\ell$ to be the maximum number of linearly independent vectors $y^{(i)}$, where $n_\ell = n'_\ell + 1 + \sum_{m=1}^{\ell} k_m$. Here $n'_\ell$ is the maximum number of linearly independent columns of the matrix formed by vectors $\left( \breve{\alpha}_{\ell,id}(x)^{(i)}_{.a} \right)$.
  (B) Set $v \leftarrow 1$.
  (C) For $p \leftarrow 1$ to $k_v$ do
    (C.1) Select a set $J_v$ of $k_v$ randomly chosen vectors in $\mathbb{F}_{2^{k_v}}$.
    (C.2) Choose a random binary vector $w = (w_1, \ldots, w_{k_v}) \in \mathbb{F}_{2^{k_v}}$, and set
          $\pi_{v,p}(j_i) = w_i$ for each $j_i \in J_v$.
    (C.3) Choose random plaintexts $x^{(i)} \mapsto (j_1^{(i)}, \ldots, \mathbf{j}_{\mathbf{v}}^{(\mathbf{i})}, \ldots, j_s^{(i)})$, where $\mathbf{j}_{\mathbf{v}}^{(\mathbf{i})} \in J_v$ and construct vectors $y^{(i)} := \left( \breve{\alpha}_{\ell,id}(x)^{(i)}_{.a} \parallel \breve{\alpha}_{\ell,id}(x)^{(i)}_{.b} + \breve{\gamma}_{\ell,id}(x)^{(i)}_{.b} \parallel 1 \right)$, as in Proposition 6.2.
          Repeat this step for an appropriate number of choices of $x^{(i)}$ and form a matrix $Y_v$ with rows being the linearly independent vectors $y^{(i)}$. If $\mathsf{rank}(Y_v) < n_\ell$ then return to (C.1).
    (C.4) Let $x^{(i)} \mapsto (j_1^{(i)}, \ldots, j_v^{(i)}, \ldots, j_s^{(i)})$, for $(i) = 1, \ldots, n_\ell$, be the plaintext used to construct row $(i)$ of the $n_\ell \times (2m+1)$ binary matrix $Y_v$ in the previous step. Form the $n_\ell \times 1$ matrix $Z_{v,p}$ with value $\pi_{v,p}(j_v^{(i)})$ as entry in row $(i)$.
```

(C.5) Construct a $(2m+1) \times n_\ell$ binary encoding matrix E_ν, such that $\mathrm{rank}(Y_\nu.E_\nu) = n_\ell$

(C.6) Compute matrix $M_{\nu,p} = E_\nu.(Y_\nu.E_\nu)^{-1}.Z_{\nu,p}$

(C.7) For each $j_\nu \in \mathbb{F}_{2^{k_\nu}} \setminus J_\nu$ choose a random plaintext $x \mapsto (j_1, \ldots, j_\nu, \ldots, j_s)$ and compute the value for $\pi_{\nu,p}(j_\nu)$ by
$\pi_{\nu,p}(j_\nu) := \Big(\breve{\alpha}_{\ell,id}(x)_{.a} \parallel \breve{\alpha}_{\ell,id}(x)_{.b} + \breve{\gamma}_{\ell,id}(x)_{.b} \parallel 1 \Big).M_{\nu,p}$

(C.8) Choose a random plaintext $x \mapsto (j_1, \ldots, j_\nu, \ldots, j_s)$ and compute the value
$y = \Big(\breve{\alpha}_{\ell,id}(x)_{.a} \parallel \breve{\alpha}_{\ell,id}(x)_{.b} + \breve{\gamma}_{\ell,id}(x)_{.b} \parallel 1 \Big).M_{\nu,p}$
If $y \neq \pi_{\nu,p}(j_\nu)$ then return to (C.2) and try another choice for $w \in \mathbb{F}_{2^{k_\nu}}$ (this can be done in at most 2^{k_ν} times). If no choice for w in (C.2) is possible, then set $\nu \leftarrow (\nu + 1)$ and return to (C).
If $y = \pi_{\nu,p}(j_\nu)$, repeat (C.8) for an appropriate number of times.

done

(D) Set transposition $\xi_\ell := (\nu, \ell)$. Permute the blocks of α and γ with transposition ξ_ℓ to get α' and γ'. Set $\alpha \leftarrow \alpha'$ and $\gamma \leftarrow \gamma'$.

(E) For each $j_\nu \in \mathbb{F}_{2^{k_\nu}}$, by using $\pi_{\nu,p}(j_\nu)$ for $p = 1, \ldots, k_\nu$, one obtains $\pi_\nu(j_\nu)$, and thus determines permutation π_ν.

done
Return $[\pi_1, \ldots, \pi_s, \xi]$, where $\xi = \xi_s \circ \ldots \circ \xi_1$.

We now clarify the steps of the Algorithm 6.4

(A) To determine the maximum value for the parameter n_ℓ we have to run this step for a sufficient number of random inputs $x^{(i)}$.

(B) This step initializes the parameter ν to start the subsequent steps of the algorithm to determine $\nu = \ell^\xi$.

(C) The inner loop is used to determine each bit $\pi_{\nu,p}(j_\nu)$ of $\pi_\nu(j_\nu)$ for $p = 1, \ldots, k_\nu$, separately, for which $\pi_\nu(j_\nu) := \big(\pi_{\nu,1}(j_\nu) \| \ldots \| \pi_{\nu,k_\nu}(j_\nu)\big)$ for all $j_\nu \in \mathbb{F}_{2^{k_\nu}}$.

(C.1) The choice of the parameter k_ν, i.e. size of the set J_ν, has an effect on the behaviour of the algorithm. If $|J_\nu| < k_\nu$, step (C.3) cannot be finished (i.e. we always get $\mathrm{rank}(Y_\nu) < n_\ell$). If $|J_\nu| > k_\nu$, the workload required in step (C.2) will be increased comparing with the case $|J_\nu| = k_\nu$.

(C.2) In this step we guess the p-th bit $\pi_{\nu,p}(j_\nu)$ of $\pi_\nu(j_\nu)$ for all $j_\nu \in J_\nu$.

(C.3) In this step a plaintext $x^{(i)} \mapsto (j_1^{(i)}, \ldots, \mathbf{j}_\mathbf{v}^{(i)}, \ldots, j_s^{(i)})$ is chosen in such a way that the component $\mathbf{j}_\mathbf{v}^{(i)}$ belongs to J_ν (chosen in step (C.1)). The other components j_u with $u \neq \nu$ are arbitrarily chosen. We repeat this step until we get a matrix Y_ν with $\mathrm{rank}(Y_\nu) = n_\ell$.
If the elements of J_ν, $|J_\nu| = k_\nu$, are chosen in such a way that the set $\{\pi_\nu(j_\nu) \mid j_\nu \in J_\nu\}$ has less than k_ν linearly independent vectors (of size k_ν), the $\mathrm{rank}(Y_\nu)$ will be

smaller than n_ℓ. In this case the algorithm returns to step (C.1), and generates a new set J_ν.

(The other possibility could be to extend the size of set J_ν, i.e. $|J_\nu| > k_\nu$.)

(C.4) We construct the $n_\ell \times 1$ matrix $Z_{\nu,p}$ with values $\pi_{\nu,p}(j_\nu^{(i)})$ using (C.2) and (C.3).

(C.5) In this step we construct a binary $(2m+1) \times n_\ell$ matrix E_ν such that $\mathtt{rank}(Y_\nu.E_\nu) = n_\ell$. This is done in the following way: Let $Q = \{1, \ldots, 2m+1\}$ be the index set of columns of Y_ν. Find a subset $Q_\nu \subseteq Q$ with $|Q_\nu| = n_\ell$, such that the columns with indices in Q_ν are all linearly independent. Consider the identity $(2m+1) \times (2m+1)$ matrix $I_{(2m+1)}$. Remove all columns with indices in $Q \setminus Q_\nu$ from $I_{(2m+1)}$ to form a $(2m+1) \times n_\ell$ matrix E_ν.

(C.6) Using E_ν from step (C.5) we determine the p-th column $M_{\nu,p}$ of the matrix M_ν.

(C.7) This step computes the p-th bit $\pi_{\nu,p}(j_\nu)$ of $\pi_\nu(j_\nu)$ for all remaining $j_\nu \in \mathbb{F}_{2^{k_\nu}}$.

(C.8) This step verifies whether the bit $\pi_{\nu,p}(j_\nu)$ guessed in step (C.2) or computed in step (C.7) for all $j_\nu \in \mathbb{F}_{2^{k_\nu}}$ is correct, and whether the value ν satisfies $\nu = \ell^\xi$. Running this step in an appropriately sufficient number of times allows us to check these requirements.

(D) In this step we use $\nu = \ell^\xi$ determined in the previous loop to construct a transposition ξ_ℓ. We update α and γ, permuting their blocks with ξ_ℓ and continue the main loop with the new value $\ell \leftarrow (\ell - 1)$.

(E) From the p-th bit $\pi_{\nu,p}(j_\nu)$ for all $p = 1, \ldots, k_\nu$ we construct $\pi_\nu(j_\nu)$. By collecting all $\pi_\nu(j_\nu)$, $j_\nu \in \mathbb{F}_{2^{k_\nu}}$, we are able to recover the permutation π_ν.

Proposition 6.5 *Let α, γ be the covers of type $(r_1, \ldots, r_s)$ used as the public key in NFT-MST_3. Let $k_\ell := \lceil \log_2 r_\ell \rceil$. The workload required to recover permutations $[\pi_1, \ldots, \pi_s, \xi]$ using Algorithm 6.4 is bounded by $\mathcal{O}(\sum_{\ell=1}^{s} \ell\, k_\ell\, 2^{k_\ell - 1})$.*

Proof. In step (C.2) of Algorithm 6.4 we have to guess vector w of k_ν bits to set the p-th bit $\pi_{\nu,p}(j_\nu)$ of $\pi_\nu(j_\nu)$ for all $j_\nu \in J_\nu$. The complexity of the algorithm includes the times required to run through all bits $p \in \{1, \ldots, k_\nu\}$ with an average of $\ell/2$ times until step (C.8) successfully terminates by finding $\nu := \ell^\xi$, and also those for the steps in the main loop for $\ell \in \{1, \ldots, s\}$. Summing up these together yields the workload as shown in the bound stated. □

Note that for any $j_m \in \{1, \ldots, r_m\}$

$$\left(t_{(0).a} + t_{(\ell-1).a}\right) = \sum_{m=1}^{\ell-1}\left(a_{(m,j_m).a} + h_{(m,j_m).a}\right) = \sum_{m=1}^{\ell-1}\left(a_{(m,1).a} + h_{(m,1).a}\right)$$

$$\left(t_{(\ell).a} + t_{(s).a}\right)^\theta = \sum_{m=\ell+1}^{s}\left(a_{(m,j_m).a} + h_{(m,j_m).a}\right)^\theta = \sum_{m=\ell+1}^{s}\left(a_{(m,1).a} + h_{(m,1).a}\right)^\theta$$

We use Proposition 6.3 to design the following algorithm.

Algorithm 6.6 *(Matrix-permutation Attack on NFT-*MST_3*: Matrix recovery)*

Input: Public key $[\alpha, \gamma]$, permutations $[\pi_1, \ldots, \pi_s, \xi]$.
Output: Matrices $[L_1, \ldots, L_s]$.

Set $\mathcal{A}_s \leftarrow (0, \ldots, 0)$, an m-bit zero vector.

For $\ell \leftarrow s$ **downto** 1 **do**

(A) Set $\nu \leftarrow \ell^{\xi}$.

(B) Select random plaintexts $x^{(i)} \mapsto (j_1^{(i)}, \ldots, j_s^{(i)})$, and construct vectors $y^{(i)} := \Big(\breve{\alpha}_{\ell,\xi}(x)_{.a}^{(i)} \parallel \breve{\alpha}(x)_{.b}^{(i)} + \breve{\gamma}(x)_{.b}^{(i)} + \mathcal{A}_{\ell}^{(i)} \parallel 1 \Big)$, as in Proposition 6.3. Define n_ν to be the maximum number of linearly independent vectors $y^{(i)}$, where $n_\nu = n'_\nu + 1 + \sum_{m=1}^{\ell} k_{m^{\xi}}$. Here n'_ν is the maximum number of linearly independent columns of the matrix formed by vectors $\Big(\breve{\alpha}_{\ell,\xi}(x)_{.a}^{(i)} \Big)$. Repeat this step for an appropriate number of choices of $x^{(i)}$ and form a matrix Y_ν with n_ν rows being the linearly independent vectors $y^{(i)}$.

(C) Let $x^{(i)} \mapsto (j_1^{(i)}, \ldots, j_s^{(i)})$, for $(i) = 1, \ldots, n_\nu$, be the plaintext used to construct row (i) of the $n_\nu \times (2m+1)$ binary matrix Y_ν in the previous step. Form the $n_\nu \times k_\nu$ matrix Z_ν with value $\pi_\nu(j_\nu)$ as entry in row (i).

(D) Construct a $(2m+1) \times n_\nu$ binary encoding matrix E_ν, such that
$\mathsf{rank}(Y_\nu.E_\nu) = n_\nu$

(E) Compute matrix $L_\nu = E_\nu \,.\, (Y_\nu.E_\nu)^{-1}.\, Z_\nu$
If $\ell = 1$ then return $[L_1, \ldots, L_s]$.

(F) Set

$$\begin{aligned}\mathcal{A}_{\ell-1} \leftarrow\ & \mathcal{A}_{\ell} + a_{(\nu,j_\nu).b} + h_{(\nu,j_\nu).b} + a^{\theta}_{(\nu,j_\nu).a} \sum_{m=1}^{\ell-1} \big(a_{(m^{\xi},1).a} + h_{(m^{\xi},1).a}\big) + \\ & a_{(\nu,j_\nu).a} \sum_{m=\ell+1}^{s} \big(a_{(m^{\xi},1).a} + h_{(m^{\xi},1).a}\big)^{\theta}\end{aligned}$$

done

By making use of the information computed by Algorithms 6.4 and 6.6 we now present an algorithm for the decryption of a given ciphertext $y = (y_1, y_2)$.

Algorithm 6.7 *(Matrix-permutation Attack on NFT-*MST_3*: Factorization)*

Input: $[\pi_1, \ldots, \pi_s, \xi, L_1, \ldots, L_s]$ for the public key $[\alpha,\ \gamma]$, ciphertext $y = (y_1, y_2)$.
Output: Plaintext $x \mapsto (j_1, \ldots, j_s)$, such that $y_1 = \breve{\alpha}(x),\ \ y_2 = \breve{\gamma}(x)$.

Set $\ \mathcal{A}_s \leftarrow (0, \ldots, 0)$.

For $\ell \leftarrow s$ **downto** 1 **do**

```
(1) Set $\nu \leftarrow \ell^{\xi}$.
(2) Construct a vector
    $w = \Big( y_{1.a} \parallel y_{1.b} \oplus y_{2.b} \oplus \mathcal{A}_\ell \parallel 1 \Big)$
(3) Compute $\pi_\nu(j_\nu) = w \cdot L_\nu$
(4) Recover $j_\nu$ using $\pi_\nu(j_\nu)$ and permutation $\pi_\nu$.
    If $\ell = 1$ then return $(j_1, \ldots, j_s)$.
(5) Set  $y_{1.a} \leftarrow y_{1.a} \oplus a_{(\nu,j_\nu).a}$
    $\mathcal{A}_{\ell-1} \leftarrow \mathcal{A}_\ell + a_{(\nu,j_\nu).b} + h_{(\nu,j_\nu).b} + a^{\theta}_{(\nu,j_\nu).a} \sum_{m=1}^{\ell-1} \big(a_{(m^{\xi},1).a} + h_{(m^{\xi},1).a}\big) +$
        $a_{(\nu,j_\nu).a} \sum_{m=\ell+1}^{s} \big(a_{(m^{\xi},1).a} + h_{(m^{\xi},1).a}\big)^{\theta}$
done
```

As presented above, the Matrix-permutation attack on NFT-MST_3 makes use of Algorithm 6.4 to recover permutations $[\pi_1, \ldots, \pi_s, \xi]$ and then Algorithm 6.6 to construct matrices $[L_1, \ldots, L_s]$. The knowledge of $[L_1, \ldots, L_s]$ and $[\pi_1, \ldots, \pi_s, \xi]$ allows the adversary to decrypt any ciphertext by using Algorithm 6.7. The usage of non-fused transversal signatures permits the construction of such matrix L_i for any block $i = \{1, \ldots, s\}$ and to compute the image $\pi_i(j_i)$ of j_i under permutation π_i as shown in Proposition 6.3. This fact is used in step (3) of Algorithm 6.7. As π_i is a bijection, the preimage j_i can be recovered if $\pi_i(j_i)$ is known, as shown in step (4) of the same algorithm.

Remark 6.8 *The determination of permutations* $[\pi_1, \ldots, \pi_s, \xi]$ *and the construction of matrices* $[L_1, \ldots, L_s]$ *could be designed in a single algorithm. However, such an algorithm would become very involved. Therefore, for the sake of clarity regarding the description of the Matrix-permutation attack we have presented two separated algorithms, namely Algorithm 6.4 and Algorithm 6.6.*

As the workload required for Algorithm 6.6 is negligible, the complexity of the Matrix-permutation attack is reduced to the complexity of the determination of permutations $[\pi_1, \ldots, \pi_s, \xi]$ by Algorithm 6.4. Thus we have the following proposition.

Proposition 6.9 *By using the same notation as in Proposition 6.5, the workload required to recover the cleartext for a given ciphertext by using the Matrix-permutation attack on NFT-*MST_3 *scheme is roughly of the same amount as required to recover permutations* $[\pi_1, \ldots, \pi_s, \xi]$*, and is bounded by* $\mathcal{O}(\sum_{\ell=1}^{s} \ell\, k_\ell\, 2^{k_\ell - 1})$.

The complexity as given in Proposition 6.9 shows in particular that for relatively small values k_i, which are usually used in a real version of the MST_3 scheme, say $k_i \leqslant 15$, the non-fused transversal logarithmic signatures cannot be used for a secure realization of MST_3.

6.3 The Matrix-permutation attack on FT-MST_3

In this section we attempt to determine the complexity of the Matrix-permutation attack on FT-MST_3.

As shown in the previous section, the Matrix-permutation attack exploits fully the way of how to factorize with respect to a non-fused transversal logarithmic signature β (Algorithm 5.10), even though the adversary does not know β. Thus, the knowledge provided by a factorization with respect to β by Algorithm 5.10 will be the crucial information for the estimation of the complexity of recovering the cleartext when the Matrix-permutation attack is applied.

To simplify the description of the Matrix-permutation attack on FT-MST_3 we confine ourselves to use step (1) and step (3) of Algorithm 5.4 only to create logarithmic signature β.

Let $\{K_1, K_2, \dots, K_\nu\}$ be a partition on the set $\{1, \dots, m\}$ with $|K_i| = k_i$ and $t_i = 2^{k_i}$ as described in Algorithm 5.4. Let $\varepsilon^* := (e^*_{i,j})$ be a signature of type $(t_1, \dots, t_\nu)$ created after step (1) of the Algorithm 5.4.

W.l.o.g. we consider $\beta := (b_{i,j})$ to be a logarithmic signature created by fusion of blocks $(\ell, \ell + 2)$ and $(\ell + 1, \ell + 3)$ of ε^*. (Note that no consecutive blocks are fused.) Then $\beta = [B_1, \dots, B_s]$ is of type $(r_1, \dots, r_s)$, where $r_\ell = t_\ell \cdot t_{\ell+2}$ and $r_{\ell+1} = t_{\ell+1} \cdot t_{\ell+3}$. We now consider one fused block, say $B_{\ell+1}$, of β.

Let $u^{[n]}_{i,j_i}$ (resp. $\mathfrak{e}^{[n]}_{i,j_i}$) be a vector of length k_n, consisting of the bits of b_{i,j_i} on the positions corresponding to K_n.

Let $x \mapsto (j_1, \dots, j_\nu)$, also let $x' \mapsto (j'_1, \dots, j'_s)$, where $j'_\ell = j_\ell \| j_{\ell+1}$ and $j'_{\ell+1} = j_{\ell+2} \| j_{\ell+3}$.

Then

$$\begin{array}{rlcccccccccl}
e^*_{1,\,j_1} & = (\,\dots & \| & \bar{0} & \| & \bar{0} & \| & \bar{0} & \| & \bar{0} & \| & \dots\,) \\
\vdots & & & & & & & & & & & \\
e^*_{\ell-1,\,j_{\ell-1}} & = (\,\dots & \| & \bar{0} & \| & \bar{0} & \| & \bar{0} & \| & \bar{0} & \| & \dots\,) \\
e^*_{\ell,\,j_\ell} & = (\,\dots & \| & \mathfrak{e}^{[\ell]}_{\ell,\,j_\ell} & \| & \bar{0} & \| & \bar{0} & \| & \bar{0} & \| & \dots\,) \\
e^*_{\ell+1,\,j_{\ell+2}} & = (\,\dots & \| & u^{[\ell]}_{\ell+1,\,j_{\ell+2}} & \| & \mathfrak{e}^{[\ell+1]}_{\ell+1,\,j_{\ell+2}} & \| & \bar{0} & \| & \bar{0} & \| & \dots\,) \\
e^*_{\ell+2,\,j_{\ell+1}} & = (\,\dots & \| & u^{[\ell]}_{\ell+2,\,j_{\ell+1}} & \| & u^{[\ell+1]}_{\ell+2,\,j_{\ell+1}} & \| & \mathfrak{e}^{[\ell+2]}_{\ell+2,\,j_{\ell+1}} & \| & \bar{0} & \| & \dots\,) \\
e^*_{\ell+3,\,j_{\ell+3}} & = (\,\dots & \| & \underbrace{u^{[\ell]}_{\ell+3,\,j_{\ell+3}}}_{K_\ell} & \| & \underbrace{u^{[\ell+1]}_{\ell+3,\,j_{\ell+3}}}_{K_{\ell+1}} & \| & \underbrace{u^{[\ell+2]}_{\ell+3,\,j_{\ell+3}}}_{K_{\ell+2}} & \| & \underbrace{\mathfrak{e}^{[\ell+3]}_{\ell+3,\,j_{\ell+3}}}_{K_{\ell+3}} & \| & \dots\,)
\end{array}$$

Then

$\breve{\beta}(x') = b_{1,j'_1} \oplus \dots \oplus b_{\ell,j'_\ell} \oplus b_{\ell+1,j'_{\ell+1}} \oplus \dots \oplus b_{s,j'_s}$

where

$$\begin{array}{lllllll}
b_{\ell,j'_\ell} & = (\ldots \; \| & \begin{array}{l}\mathfrak{e}^{[\ell]}_{\ell,\, j_\ell} \oplus \\ u^{[\ell]}_{\ell+2,\, j_{\ell+1}}\end{array} & \| \; u^{[\ell+1]}_{\ell+2,\, j_{\ell+1}} & \| \; \mathfrak{e}^{[\ell+2]}_{\ell+2,\, j_{\ell+1}} & \| \quad \bar{0} & \| \; \ldots)
\end{array}$$

$$\begin{array}{lllllll}
b_{\ell+1,\, j'_{\ell+1}} & = (\ldots \; \| & \begin{array}{l}u^{[\ell]}_{\ell+1,\, j_{\ell+2}} \oplus \\ u^{[\ell]}_{\ell+3,\, j_{\ell+3}}\end{array} & \| \; \begin{array}{l}\mathfrak{e}^{[\ell+1]}_{\ell+1,\, j_{\ell+2}} \oplus \\ u^{[\ell+1]}_{\ell+3,\, j_{\ell+3}}\end{array} & \| \; u^{[\ell+2]}_{\ell+3,\, j_{\ell+3}} & \| \; \mathfrak{e}^{[\ell+3]}_{\ell+3,\, j_{\ell+3}} & \| \; \ldots)
\end{array}$$

and therefore

$$\sum_{i=1}^{\ell+1} b_{i,j'_i} = (\ldots \; \| \; \overbrace{\begin{array}{l}\mathfrak{e}^{[\ell]}_{\ell,\, j_\ell} \oplus \\ u^{[\ell]}_{\ell+2,\, j_{\ell+1}} \oplus \\ u^{[\ell]}_{\ell+1,\, j_{\ell+2}} \oplus \\ u^{[\ell]}_{\ell+3,\, j_{\ell+3}}\end{array}}^{K_\ell} \; \| \; \overbrace{\begin{array}{l}u^{[\ell+1]}_{\ell+2,\, j_{\ell+1}} \oplus \\ \mathfrak{e}^{[\ell+1]}_{\ell+1,\, j_{\ell+2}} \oplus \\ u^{[\ell+1]}_{\ell+3,\, j_{\ell+3}}\end{array}}^{K_{\ell+1}} \; \| \; \overbrace{\begin{array}{l}\mathfrak{e}^{[\ell+2]}_{\ell+2,\, j_{\ell+1}} \oplus \\ u^{[\ell+2]}_{\ell+3,\, j_{\ell+3}}\end{array}}^{K_{\ell+2}} \; \| \; \overbrace{\mathfrak{e}^{[\ell+3]}_{\ell+3,\, j_{\ell+3}}}^{K_{\ell+3}} \; \| \; \ldots)$$

Assume we use the factorization scheme as given by Algorithm 5.10. As the bits of $u^{[m]}_{i,j_i}$, are randomly chosen, only the bits of $\mathfrak{e}^{[m]}_{i,j_i}$ can be used for factoring with respect to β. Therefore, to factorize $\sum_{i=1}^{\ell+1} b_{i,j'_i}$, i.e. to recover the index $j'_{\ell+1}$ for block $B_{\ell+1}$, we may only use bits of $\mathfrak{e}^{[\ell+3]}_{\ell+3,\, j_{\ell+3}}$, i.e. the bits on positions $K_{\ell+3}$. However, as $B_{\ell+1}$ has length $r_{\ell+1} = 2^{k_{\ell+1}+k_{\ell+3}}$, there are $2^{k_{\ell+1}}$ elements of $B_{\ell+1}$ having the same value $\mathfrak{e}^{[\ell+3]}_{\ell+3,\, j_{\ell+3}}$ on positions $K_{\ell+3}$. In other words, only $k_{\ell+3}$ bits from index $j'_{\ell+1}$ can be determined.

Having obtained this information, we now return to the Matrix-permutation attack when a fused signature β is used in an FT-MST_3. Similar to Proposition 6.3 we may show that there exists a matrix $L_{\ell+1}$ such that

$$\big(\; \breve{\alpha}(x)_{.a} + \mathcal{A}_{\ell+1} \; \| \; \breve{\alpha}(x)_{.b} + \breve{\gamma}(x)_{.b} + \mathcal{B}_{\ell+1} \; \| \; 1 \; \big) \, L_{\ell+1} = \mathfrak{e}^{[\ell+3]}_{\ell+3,\, j_{\ell+3}}$$

Using such a matrix we can recover only $k_{\ell+3}$ from $(k_{\ell+1} + k_{\ell+3})$ bits of $j'_{\ell+1}$ for $B_{\ell+1}$.

This shows that the Matrix-permutation attack applied to FT-MST_3 can recover only a portion of bits of the index in each fused block of β. Thus we have the following proposition.

Proposition 6.10 *Let* B_ℓ *be a block of a fused transversal logarithmic signature* β *used in FT-*MST_3*. Let* $B_\ell = ((D_{i_1}.D_{i_2}) \ldots D_{i_{u_\ell}})$ *as defined Algorithm 5.4, where* $i_1 < i_2 < \cdots < i_{u_\ell}$*. Let* $k_i = \lceil \log_2 D_i \rceil$*. By using the Matrix-permutation attack on FT-*MST_3 *one can determine* $k_{i_{u_\ell}}$ *from* $\sum_{j=1}^{u_\ell} k_{i_j}$ *bits for the index in block* B_ℓ*.*

The complexity of factoring a ciphertext by using the Matrix-permutation attack on FT-MST_3 is thus given as the product of the complexities for factoring with respect to each block B_ℓ, $\ell = 1, \ldots, s$. Moreover, as the factorization has to be proceeded implicitly according to the permutation ξ of Algorithm 5.4, it turns out that the last attacked block can be carried out by a table search, and therefore has a negligible complexity. To summarize we record the complexity of the Matrix-permutation attack on FT-MST_3 in the following proposition.

Proposition 6.11 *Let m be an input length of an FT-MST_3 scheme with a fused transversal logarithmic signature β created by Algorithm 5.4. Let $\mathcal{P} = \{P_1, \dots, P_s\}$ be a partition used in step (3) of this algorithm where $P_\ell = \{i_{\ell,1}, \dots, i_{\ell,u_\ell}\}$ for $\ell = 1, \dots, s$. Let $k_\ell = \lceil \log_2 D_\ell \rceil$, where D_ℓ is defined by the same algorithm. Then the workload still needed after the Matrix-permutation attack to recover the plaintext for a given ciphertext is of $\mathcal{O}(2^c)$ where*

$$c = (m - \sum_{\ell=2}^{s} k_{i_{\ell,u_\ell}} - \sum_{j=1}^{u_1} k_{i_{1,j}})$$

Remark 6.12 *We can envisage a further method of using the Matrix-permutation attack on the FT-MST_3 scheme. Suppose that the adversary attempts to keep fusing the blocks of α and γ to eventually obtain a new $\tilde{\alpha}$ and $\tilde{\gamma}$, in which the corresponding logarithmic signature $\tilde{\beta}$ (inside $\tilde{\gamma}$) has a block $\tilde{B}_i$ which forms a subspace of dimension m_i in $\mathcal{Z}$. Note that the adversary actually does not know $\tilde{B}_i$ and therefore cannot verify whether $\tilde{B}_i$ is a subspace or not. Assuming that $\tilde{B}_i$ is a subspace (s)he may attempt to apply the Matrix-permutation attack to $\tilde{\alpha}$ and $\tilde{\gamma}$ to compute the index in $\tilde{B}_i$ for the plaintext from a given ciphertext. It is fairly easy to prevent this type of attack by selecting a partition P in step (3) of Algorithm 5.4 in a way that such a block $\tilde{B}_i$ necessarily has a large dimension m_i. This makes the Matrix-permutation attack impossible because of its complexity, as given in Proposition 6.9.*

7 Implementation aspects of MST_3

In this section we consider practical implementation issues of FT-MST_3 with the underlying Suzuki 2-groups. The Algorithm 5.4 as decribed in Section 5 will be used for generating logarithmic signatures β. As observed, if the information at each step of Algorithm 5.4 is kept track, in particular, the knowledge of partition $\mathcal{P} = \{P_1, \dots, P_s\}$ used in step (3), we have a highly efficient factorization method with respect to β as shown in Algorithm 5.8. By taking the discussion of Subsection 6.1 into account we may, if necessary, select the elements of $\alpha_{.a}$ in a subspace $\mathcal{Z}_1$ such that $\rho = |\mathcal{Z}|/|\mathcal{Z}_1|$ is sufficiently large.

7.1 Computing with the Suzuki 2-groups

Let $q = 2^m$, where $m \geqslant 3$ is not a power of 2 and let θ be a non trivial odd-order automorphism of $\mathbb{F}_q$. Let $\mathcal{G} = A(m, \theta)$ be the Suzuki 2-group of order q^2 described in Section 2. Recall that the multiplication of two elements in $\mathcal{G}$ is given by the rule:

$$S(a_1, b_1)S(a_2, b_2) = S(a_1 + a_2\ ,\ b_1 + b_2 + a_1 a_2^\theta). \tag{7.19}$$

We could store the group elements $S(a, b)$ as pairs (a, b), but this would require that we compute some a^θ each time we compute a product of group elements. In turn, each computation a^θ requires at most $2\lceil \log_2 m \rceil$ multiplications in $\mathbb{F}_q$. It is therefore more time efficient to store the group elements as triples (a, b, a^θ). Thus, the product $S(a_1, b_1) \cdot S(a_2, b_2)$ is identified with the triple

$$(a_1 + a_2\ ,\ b_1 + b_2 + a_1 a_2^\theta\ ,\ a_1^\theta + a_2^\theta)$$

and computation of the product requires just a single multiplication and four additions in $\mathbb{F}_q$.

7.2 Public key size and cipher expansion

Let $\alpha = \big((a_{(ij).a}, a_{(ij).b})\big)$ and $\gamma = \big((h_{(ij).a}, h_{(ij).b})\big)$. For a given i we have $h_{(ij).a} = a_{(ij).a} + t_{(i-1).a} + t_{(i).a}$ for all $j = 1, \ldots, r_i$. This means that for each i, if $a_{(ij).a}$'s and the sum $t_{(i-1).a} + t_{(i).a}$ are known, $h_{(ij).a}$'s can be derived. Therefore, for the public key we need to store $[\alpha, (h_{(ij).b})]$ (i.e. the "b.part" of γ) and the s values $t_{(i-1).a} + t_{(i).a}$, for $i = 1, \ldots, s$.

However, for a practical implementation of MST_3 we describe a more efficient method dealing with the key storage. The idea is that we generate the key by using a publicly known Algorithm $\mathcal{A}$, which generates random cover α satisfying conditions in Section 3. Essentially, the Algorithm $\mathcal{A}$ utilizes a pseudo-random number generator $\mathcal{R}$. To simplify the description we assume that a logarithmic signature β has been generated by Algorithm 5.4 separately.

Algorithm 7.1 *(Reduced key storage)*

External: Algorithm $\mathcal{A}$, a pseudo-random number generator $\mathcal{R}$
Input: $[\beta,\ f,\ t_0,\ \ldots, t_s]$, a seed S for $\mathcal{R}$
Output: $[\alpha,\ \gamma]$

(a) Using $\mathcal{A}$, $\mathcal{R}$ and S to generate $\alpha = \big((a_{(ij).a}, a_{(ij).b})\big)$

(b) Create $\gamma = \big((h_{(ij).a}, h_{(ij).b})\big)$ from α as decribed in Section 3.

From Algorithm 7.1 it is clear that for the public key one has to publish $(h_{(ij).b})$ together with $t_{(i-1).a} + t_{(i).a}$ for $i = 1, \ldots, s$, and S only. To obtain the complete public key, i.e. $[\alpha, \gamma]$, one first generates α from step (a) using $\mathcal{R}$ and seed S, then computes $(h_{(ij).a})$ from $a_{(ij).a}$ and $t_{(i-1).a} + t_{(i).a}$. This approach will reduce the key size of the system roughly to only *one third* of $[\alpha, (h_{(ij).b})]$ (i.e. one fourth of the public-key $[\alpha, \gamma]$). In fact, this key size appears to be the minimum key storage that can be realized for MST_3.

The cipher expansion of MST_3 is of a factor three. For, suppose (y_1, y_2) is a ciphertext pair with $y_1 = (y_{(1).a}, y_{(1).b})$ and $y_2 = (y_{(2).a}, y_{(2).b})$, then, it suffices to send y_1 and $y_{(2).b}$ as the ciphertext. This is because $y_{(2).a}$ can be obtained from the equation $y_{(2).a} = y_{(1).a} + t_{(0).a} + t_{(s).a}$ by using the private key t_0 and t_s.

7.3 Examples for generation of β

We need to introduce some notation. We say a logarithmic signature (cover) β is of type $(\nu_1^{u_1}.\nu_2^{u_2}.\ldots.\nu_t^{u_t})$ if β has the first u_1 blocks of size ν_1, the next u_2 blocks of size ν_2, etc.

Let $\varepsilon = [E_1, E_2, \ldots, E_s]$ be a transversal logarithmic signature created by Algorithm 5.4 by steps (1) and (2). Then, there exists a chain of subgroups $1_{\mathcal{G}} = \mathcal{G}_0 < \mathcal{G}_1 < \cdots < \mathcal{G}_s = \mathcal{G}$, such that each block E_i consists of a complete set of coset representatives of $\mathcal{G}_{i-1}$ in $\mathcal{G}_i$.

We also write $[r_{u_1} \times r_{u_2} \times \cdots \times r_{u_\ell}]$ to denote the fusion of ℓ blocks $E_{u_1}, E_{u_2}, \ldots, E_{u_\ell}$, where $|E_{u_i}| = r_{u_i}$ and $u_1 < u_2 < \cdots < u_\ell$. Specially, $[r_u]$ denotes a non-fused block E_u with $|E_u| = r_u$. We say a block B_i of β is of fusion type $[r_{u_1} \times r_{u_2} \times \cdots \times r_{u_\ell}]$ if

$\mathsf{B}_i = \mathsf{E}_{u_1} \otimes \mathsf{E}_{u_2} \otimes \cdots \otimes \mathsf{E}_{u_\ell}$. We write

$$\mathsf{F}_1^{e_1}.\mathsf{F}_2^{e_2} \ldots \mathsf{F}_\ell^{e_\ell}$$

to denote the *fusion type of* β for which the first e_1 blocks are of fusion type F_1, the next e_2 blocks of fusion type F_2, etc. For example, the notation $[256]^2.[32\times 4]^5.[32\times 8\times 4]^{10}$ denotes a set-up for β with the first two non-fused blocks of length 256, the next five created by the fusion of two blocks of size 32 and 4, and the remaining ten blocks obtained by fusing three blocks of size 32, 8 and 4 respectively.

Let β be a fused logarithmic signature of fusion type, say, $[v_1]^{e_1} \ldots [v_{i-1}]^{e_{i-1}}.[v_i \times u_i]^{e_i} \ldots [v_{j-1} \times u_{j-1}]^{e_{j-1}}.[v_j \times u_j \times w_j]^{e_j} \ldots [v_\ell \times u_\ell \times w_\ell]^{e_\ell}$, generated by Algorithm 5.4. If β is used for a set-up of FT-MST_3, the workload of the Matrix-permutation attack required to recover the plaintext is roughly bounded by $\mathcal{O}(v_i^{e_i} \ldots v_{j-1}^{e_{j-1}}.(v_j.u_j)^{e_j} \ldots (v_\ell.u_\ell)^{e_\ell})$ (see Proposition 6.11).

Example 1

Here we show an example of the set-up for FT-MST_3 as given in the Table 2 below. Let $m = 224$ and $s = 32$. The following steps are required to successfully set-up the scheme.

Set-up:

(1) Using Algorithm 5.4 generate a logarithmic signature β for $\mathcal{Z}$ starting from ε of type $(128^2.32^{30}.4^{30})$, i.e. $v = 62$. Particularly, for step (3) of the algorithm take a partition $\mathcal{P} = \{P_1, \ldots, P_{32}\}$ with $P_1 = \{1\}, P_2 = \{2\}$, and $P_i = \{i, v - i + 3\}$ for $i = 3, \ldots, 30$, and $P_{31} = \{31, 33\}$, $P_{32} = \{32, 34\}$. After finishing Algorithm 5.4 the logarithmic signature β is of type (128^{32}), and of fusion type $[128]^2.[32 \times 4]^{30}$.

(2) Using β and Algorithm 7.1 create public key $[\alpha, \gamma]$.

Example 2

Let $m = 255$ and $s = 26$.

Set-up:

(1) Using Algorithm 5.4 generate β for $\mathcal{Z}$ starting from ε of type $(256.32^{25}.8^{24}.4^{25})$, i.e. $v = 75$. For the step (3) of this algorithm take a partition $\mathcal{P} = \{P_1, \ldots, P_{32}\}$ with $P_1 = \{1\}, P_2 = \{2, 75\}$, and $P_i = \{i,\ i + 24,\ i + 48\}$ for $i = 3, \ldots, 26$. Therefore β is of type $(256.128.1024^{24})$, and of fusion type $[256].[32 \times 4].[32 \times 8 \times 4]^{24}$.

(2) Using β and Algorithm 7.1 create public key $[\alpha, \gamma]$.

7.4 Performance of the system

In this section we show the data of the performance of MST_3 acquired from a concrete implementation of the scheme.

The Table 1 shows the number of operations required for one encryption or decryption of the FT-MTS_3. Namely, the addition (ADD), the multiplication (MULT), the exponentiation with θ ($\mathsf{EXP}(\theta)$), the generation of m-bit random R (PRNG), and the factorization of $\breve{\varepsilon}(\mathsf{R}) \in \mathcal{Z}$ with respect to a transversal logarithmic signature ε using the Algorithm 5.9 (FACTOR).

Table 1: Number of basic operations for one encryption/decryption of FT-MST_3.

	$\mathbb{F}_{2^m}$ ADD	$\mathbb{F}_{2^m}$ MULT	$\mathbb{F}_{2^m}$ EXP(θ)	$\mathbb{F}_{2^m}$PRNG	FACTOR
encryption	$7s - 7$	$2s - 2$	-	1	-
decryption	$m + 4s + 8$	$s + 3$	2	-	1

We note that an intrinsic property of MST_3 is that there is a trade-off between the key storage and the speed of the scheme. For example, if $\mathbb{F}_{2^{320}}$ is the underlying field for the Suzuki 2-group $\mathcal{G}$, then the corresponding FT-MST_3 has an input of 320 bit length; if α and γ have type (4.64^{53}), the public key size is of 135 kBytes, whereas if α and γ have the type (256^{40}), we have a public key of 402 kBytes. This implementation shows that for the first case we have an encryption/decryption speed of 287/471 kB/s, whereas for the second case 377/581 kB/s.

The Table 2 presents data related to the public key size, type and fusion type for β, the speed of the encryption and decryption together with the workload (W) of the Matrix-permutation attack required to recover the plaintext. The performance tests were implemented by using the library NTL[1] and measured on a 64-bit machine of 1.8 GHz.

8 Conclusions

We have presented a revised version of the MST_3 public-key cryptosystem on the basis of the Suzuki 2-groups. An elaborate investigation of recovering the private key by using heuristic and algebraic arguments has produced strong lower bounds for the workload required. We have developed a powerful chosen plaintext attack on the scheme, called Matrix-permutation attack, which shows specially that the class of non-fused transversal logarithmic signatures for the center of the underlying groups are unfit for use in the realization of MST_3. The class of fused transversal logarithmic signatures, however, withstands the Matrix-permutation attack. We have determined the complexity of this attack on the scheme using fused transversal logarithmic signatures. This result enables us to choose the right parameters for the scheme, which we have discussed in the last section. Data of key storage and of speed performance of a concrete implementation of the scheme have been included. A further challenging problem regarding the realization of the scheme is the question of how to use the class of non-transversal logarithmic signatures or random covers for β. We will deal with this problem in a future work.

[1] NTL C++ Library, written by Victor Shoup, http://www.shoup.net/ntl

Table 2: Various data for parameters, performance and security of FT-MST_3.

m	s	*type of* β	pk [kB]	*fusion type of* β	W	E [kB/s]	D [kB/s]
160	26	$(256^2 \cdot 64^{24})$	43	$[256].[16 \times 4 \times 4].[16 \times 4]^{24}$	2^{102}	607	859
160	23	$(64 \cdot 128^{22})$	57	$[64].[128].[32 \times 4]^{21}$	2^{105}	604	852
160	20	(256^{20})	100	$[256].[16 \times 4 \times 4]^{19}$	2^{114}	671	895
160	18	$(256^2 \cdot 512^{16})$	170	$[256].[16 \times 4 \times 4].[32 \times 4 \times 4]^{16}$	2^{118}	689	904
160	16	(1024^{16})	320	$[1024].[32 \times 8 \times 4]^{15}$	2^{120}	758	941
192	32	(64^{32})	49	$[64]^3.[16 \times 4]^{29}$	2^{116}	571	854
192	28	$(8 \cdot 128^{27})$	82	$[8].[128]^2.[32 \times 4]^{25}$	2^{125}	529	783
192	24	(256^{24})	145	$[256].[16 \times 4 \times 4]^{23}$	2^{138}	609	851
192	22	$(8 \cdot 512^{21})$	253	$[8].[512].[32 \times 4 \times 4]^{20}$	2^{140}	679	914
192	20	$(4 \cdot 1024^{19})$	457	$[4].[1024].[32 \times 8 \times 4]^{18}$	2^{144}	720	924
224	38	$(4 \cdot 64^{37})$	66	$[4].[64]^4.[16 \times 4]^{33}$	2^{132}	511	772
224	32	(128^{32})	113	$[128]^2.[32 \times 4]^{30}$	2^{150}	565	827
224	28	(256^{28})	197	$[256].[16 \times 4 \times 4]^{27}$	2^{162}	595	845
224	25	$(256 \cdot 512^{24})$	344	$[256].[32 \times 4 \times 4]^{24}$	2^{168}	637	875
224	23	$(256 \cdot 64 \cdot 1024^{21})$	597	$[256].[16 \times 4].[32 \times 8 \times 4]^{21}$	2^{172}	678	894
255	43	$(8 \cdot 64^{42})$	85	$[8].[64]^4.[16 \times 4]^{38}$	2^{152}	532	808
255	37	$(8 \cdot 128^{36})$	145	$[8].[128]^2.[32 \times 4]^{34}$	2^{170}	576	852
255	32	$(256^{31} \cdot 128)$	252	$[256].[16 \times 4 \times 4]^{30}.[32 \times 4]$	2^{185}	602	865
255	29	$(8 \cdot 512^{28})$	447	$[8].[512].[32 \times 4 \times 4]^{27}$	2^{189}	637	887
255	26	$(256 \cdot 128 \cdot 1024^{24})$	778	$[256].[32 \times 4].[32 \times 8 \times 4]^{24}$	2^{197}	708	932
288	48	(64^{48})	110	$[64]^5.[16 \times 4]^{43}$	2^{172}	306	502
288	41	$(256 \cdot 128^{40})$	190	$[256].[128].[32 \times 4]^{39}$	2^{195}	325	523
288	36	(256^{36})	325	$[256]^2.[16 \times 4 \times 4]^{34}$	2^{204}	381	593
288	32	(512^{32})	577	$[512].[32 \times 4 \times 4]^{31}$	2^{217}	407	595
288	29	$(512^2 \cdot 1024^{27})$	1009	$[512].[32 \times 4 \times 4].[32 \times 8 \times 4]^{27}$	2^{223}	457	668
320	54	$(4 \cdot 64^{53})$	135	$[4].[64]^5.[16 \times 4]^{48}$	2^{192}	287	471
320	46	$(8 \cdot 512 \cdot 128^{44})$	242	$[8].[512].[32 \times 4]^{44}$	2^{220}	305	490
320	40	(256^{40})	402	$[256]^2.[16 \times 4 \times 4]^{38}$	2^{228}	377	581
320	36	$(32 \cdot 512^{35})$	703	$[32].[512].[32 \times 4 \times 4]^{34}$	2^{238}	403	604
320	32	(1024^{32})	1281	$[1024].[32 \times 8 \times 4]^{31}$	2^{248}	450	650
352	59	$(16 \cdot 64^{58})$	163	$[16].[64]^6.[16 \times 4]^{52}$	2^{208}	246	408
352	51	$(4 \cdot 128^{50})$	277	$[4].[128]^3.[32 \times 4]^{47}$	2^{235}	292	475
352	44	(256^{44})	486	$[256]^2.[16 \times 4 \times 4]^{42}$	2^{252}	304	481
352	40	$(2 \cdot 512^{39})$	860	$[2].[512].[32 \times 4 \times 4]^{38}$	2^{266}	352	537
352	36	$(8 \cdot 512 \cdot 1024^{34})$	1431	$[8].[512].[32 \times 8 \times 4]^{34}$	2^{272}	378	566
384	64	(64^{64})	195	$[64]^6.[16 \times 4]^{58}$	2^{232}	252	421
384	55	$(64 \cdot 128^{54})$	330	$[64].[128]^3.[32 \times 4]^{51}$	2^{255}	287	466
384	48	(256^{48})	578	$[256]^2.[16 \times 4 \times 4]^{46}$	2^{276}	303	485
384	43	$(64 \cdot 512^{42})$	1013	$[64].[512].[32 \times 4 \times 4]^{41}$	2^{287}	352	535
384	39	$(16 \cdot 1024^{38})$	1827	$[16].[1024].[32 \times 8 \times 4]^{37}$	2^{296}	364	554

References

[1] Y. Berkovich, Z. Janko Groups of Prime Power Order, Volume 2 Walter de Gruyter, Berlin, New York 2008.

[2] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory,* **31**(1985), 469–472.

[3] G. Higman, Suzuki 2-groups, *Illinois J. Math.*, **7** (1963), 79-96.

[4] n B. Huppert Endliche Gruppen I Springer-Verlag Berlin Heidelberg New York 1967.

[5] B. Huppert and N. Blackburn, Finite Groups II Springer-Verlag Berlin Heidelberg New York 1982.

[6] W. Lempken, S.S. Magliveras, Tran van Trung, W. Wei, A public key cryptosystem based on non-abelian finite groups, *J. Cryptology* **22** (2009), 62–74.

[7] S. S. Magliveras, A cryptosystem from logarithmic signatures of finite groups, In *Proceedings of the 29'th Midwest Symposium on Circuits and Systems*, Elsevier Publishing Company, (1986), 972–975.

[8] S. S. Magliveras and N. D. Memon, The Algebraic Properties of Cryptosystem PGM, *J. of Cryptology*, **5** (1992), 167-183.

[9] S. S. Magliveras, D. R. Stinson and Tran van Trung, New approaches to designing public key cryptosystems using one-way functions and trapdoors in finite groups, *J. Cryptology,* **15** (2002), 285–297.

[10] S. S. Magliveras, P. Svaba, Tran van Trung and P. Zajac, On the security of a realization of cryptosystem MST_3, *Tatra Mt. Math. Publ.* **41** (2008), 1-13.

[11] P. Nguyen, Editor, *New Trends in Cryptology,* European project "STORK – Strategic Roadmap for Crypto" – IST-2002-38273. http://www.di.ens.fr/ pnguyen/pub.html#Ng03

[12] Peter Shor, Polynomial time algorithms for prime factorization and discrete logarithms on quantum computers. *SIAM Journal on Computing*, 26(5) (1997), 1484-1509.

[13] P. Svaba and Tran van Trung On generation of random covers for finite groups *Tatra Mt. Math. Publ.* **37** (2007), 105–112.