Public key cryptosystem \( \text{MST}_3 \): 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 \( \text{MST}_3 \), 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 \( \text{MST}_3 \) 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 \( L \) 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 \( \text{MST}_3 \) [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 referred to [8], [9]. The group theoretic notation used is standard and may be found in [4] or in any textbook of group theory.

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

$$X \cdot Y = [x_1 y_1, x_1 y_2, \ldots, x_1 y_s, x_2 y_1, x_2 y_2, \ldots, x_2 y_s, \ldots, x_r y_1, x_r y_2, \ldots, x_r y_s]$$

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 $X = [x_1, \ldots, x_r] \in G[\mathbb{Z}]$, we denote by $\overline{X}$ the element $\sum_{i=1}^{r} x_i$ in the group ring $\mathbb{Z}G$. 

2
Suppose that $\alpha = [A_1, A_2, \ldots, A_s]$ is a sequence of $A_1 \in \mathcal{G}^{[Z]}$, such that $\sum_{i=1}^{s} |A_i|$ is bounded by a polynomial in $\log |\mathcal{G}|$. Let

$$A_1 \cdot A_2 \cdot \cdots A_s = \sum_{g \in \mathcal{G}} a_g g, \quad a_g \in \mathbb{Z}$$

(2.1)

Let $\mathcal{S}$ be a subset of $\mathcal{G}$, then we say that $\alpha$ 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 $\alpha = [A_1, \ldots, A_s]$ 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

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

(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 $\alpha$ 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 $(r_1, \ldots, r_s)$ with $r_i = |A_i|$ the type of $\alpha$. We say that $\alpha$ is nontrivial if $s \geq 2$ and $r_i \geq 2$ for $1 \leq i \leq s$; otherwise $\alpha$ is said to be trivial. Cover $\alpha$ 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 $\gamma : \mathcal{G}_0 = \mathcal{G}_0 < \mathcal{G}_1 < \cdots < \mathcal{G}_s = \mathcal{G}$ 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 $[A_1, \ldots, A_s]$ 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 presumably 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 $\alpha = [A_1, A_2, \ldots, A_s]$ 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 $\alpha$ 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 methods show 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 $\alpha = [A_1, A_2, \ldots, A_s]$ be a cover of type $(r_1, r_2, \ldots, r_s)$ for $\mathcal{G}$ with $A_1 = [a_{1,1}, a_{1,2}, \ldots, a_{1,r_1}]$ and let $m = \prod_{i=1}^{s} r_i$. Let $m_1 = 1$ and $m_i = \prod_{j=1}^{i-1} r_j$ for $i = 2, \ldots, s$. Let $\tau$ denote the canonical bijection from $\mathbb{Z}_{r_1} \oplus \mathbb{Z}_{r_2} \oplus \cdots \oplus \mathbb{Z}_{r_s}$ on $\mathbb{Z}_m$; i.e.
\[ \tau : \mathbb{Z}_{r_1} \oplus \mathbb{Z}_{r_2} \oplus \cdots \oplus \mathbb{Z}_{r_s} \rightarrow \mathbb{Z}_m \]
\[ \tau(j_1, j_2, \ldots, j_s) := \sum_{i=1}^{s} j_i m_i. \]

Using \( \tau \) we now define the surjective mapping \( \tilde{\alpha} \) induced by \( \alpha \).

\[ \tilde{\alpha} : \mathbb{Z}_m \rightarrow \mathcal{G} \]
\[ \tilde{\alpha}(x) := a_{1,j_1} \cdot a_{2,j_2} \cdots a_{s,j_s}, \]

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

Conversely, given a cover \( \alpha \) and an element \( y \in \mathcal{G} \), to determine any element \( x \in \tilde{\alpha}^{-1}(y) \) 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 \( \alpha \) is tame. Once a vector \((j_1, j_2, \ldots, j_s)\) has been determined, \( \tilde{\alpha}^{-1}(y) = \tau(j_1, j_2, \ldots, j_s) \) 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 \( B_i = g_i^{-1} A_i g_i \). We say that \( \beta \) is a two sided transform of \( \alpha \) by \( g_0, g_1, \ldots, g_s \); in the special case, where \( g_0 = 1 \) and \( g_s = 1 \), \( \beta \) is called a sandwich of \( \alpha \). Note that \( \beta \) is a cover for \( \mathcal{G} \).

Two covers (logarithmic signatures) \( \alpha, \beta \) are said to be equivalent if \( \tilde{\alpha} = \tilde{\beta} \). For example, if \( \beta \) is a sandwich of \( \alpha \), then \( \alpha \) and \( \beta \) 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 \( S \) of a group \( \mathcal{G} \), then finding a factorization in (2.2) is an intractable problem. In other words, the mapping

\[ \tilde{\alpha} : \mathbb{Z}_m \rightarrow \mathcal{G} \]

induced by \( \alpha \) 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 \( \mathbb{Z} \) such that \( \mathcal{G} \) does not split over \( \mathbb{Z} \), i.e. there is no subgroup \( \mathcal{H} < \mathcal{G} \) with \( \mathcal{H} \cap \mathbb{Z} = 1 \) such that \( \mathcal{G} = \mathbb{Z} \cdot \mathcal{H} \). Moreover, we assume that the order of \( \mathbb{Z} \) is sufficiently large so that exhaustive search problems are computationally infeasible in \( \mathbb{Z} \).

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 $\mathcal{G}'$ is 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 $\xi$, 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 $3 \leq m \in \mathbb{N}$ such that the field $\mathbb{F}_q$ has a nontrivial automorphism $\theta$ 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

$$\mathcal{G} := \{S(a, b) | a, b \in \mathbb{F}_q\},$$

where

$$S(a, b) = \begin{pmatrix}
1 & a & b \\
0 & 1 & a^\theta \\
0 & 0 & 1
\end{pmatrix}$$

is a $3 \times 3$-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

$$\mathbb{Z} := \mathbb{Z}(\mathcal{G}) = \Phi(\mathcal{G}) = \mathcal{G}' = \Omega_1(\mathcal{G}) = \{S(0, b) | b \in \mathbb{F}_q\}.$$ 

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:

$$S(a_1, b_1)S(a_2, b_2) = S(a_1 + a_2, b_1 + b_2 + a_1a_2^\theta). \quad (2.3)$$

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 $\phi = \theta^{\pm1}$. 

5
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 $A(\mathfrak{m}, \theta)$. And $\Lambda$ acts on $A(\mathfrak{m}, \theta)$ 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 $\Lambda$ has order $q - 1$ and permutes cyclically the $q - 1$ involutions in the center of $A(\mathfrak{m}, \theta)$.

3 Public key cryptosystem MST$_3$ on Suzuki 2-groups

Notation

- From now on let $\mathcal{G} := A(\mathfrak{m}, \theta)$ be a Suzuki 2-group defined on $\mathbb{F}_q$ with $q = 2^m$.
- As $\mathcal{Z} = \mathcal{Z}(\mathcal{G})$ 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 $\text{GL}(m, 2)$, (i.e. the group is formed by all $m \times m$ invertible matrices over $\mathbb{F}_2$). Denote $\text{Aut}(\mathcal{Z}) := \text{GL}(m, 2)$. If $z = S(0, b) \in \mathcal{Z}$ and $\varphi \in \text{Aut}(\mathcal{Z})$, then the action of $\varphi$ 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 $N = \text{Ker}(f)$. 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 $N \geq \mathcal{Z}$. 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 $\beta$ for a certain subset $\mathcal{J}$ of $\mathcal{G}$ such that $A_1, \ldots, A_s \subseteq \mathcal{J} \setminus \mathcal{Z}$. The elements in each block $A_i = [a_{i,1}, a_{i,2}, \ldots, a_{i,r_i}]$ satisfy the following conditions:
   (i) $a_{(ij)_1} \neq a_{(ij)_2}$, 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) $\sum_{j=1}^{r_i} a_{(ij)_1} = 0$. The meaning of this condition will be obvious when we discuss the security of the system in the subsequent section.

Further select
3. \( t_0, t_1, \ldots, t_s \in \mathbb{G} \setminus \mathbb{Z} \).

4. A homomorphism \( f: \mathbb{G} \rightarrow \mathbb{Z} \)

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 \( x \in \mathbb{Z} \) and the public key \( \alpha \) and \( \gamma \).

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

1. Choose a random \( R \in \mathbb{Z}_{|\mathbb{G}|} \) and compute

   \[
   y_1 = \tilde{\alpha}(R)x, \quad y_2 = \tilde{\gamma}(R)x = t_0^{-1} \tilde{\alpha}(R)f(\tilde{\alpha}(R)) \tilde{\beta}(R)t_s x
   \]

**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 \( x \in \mathbb{Z} \) that corresponds to the ciphertext \((y_1, y_2)\).

1. Using the fact that \( f(y_1) = f(\tilde{\alpha}(R)) \) (from Remark 3.1) compute

   \[
   \tilde{\beta}(R) = f(\tilde{\alpha}(R))^{-1} y_1^{-1} t_0 y_2 t_s^{-1} = f(y_1)^{-1} y_1^{-1} t_0 y_2 t_s^{-1}
   \]

2. Recover \( R \) from \( \tilde{\beta}(R) \) which is efficiently computable as \( \beta \) is factorizable. By computing \( \tilde{\alpha}(R) \) we then recover \( x \) from \( y_1 \).

**Specification of the homomorphism \( f \)**

For the realization of the cryptosystem \( \text{MST}_3 \) we use the following class of homomorphisms. Let \( g = S(g_a, g_b) \in \mathbb{G} \), and let \( \sigma \in \text{Aut}(\mathbb{Z}) := \text{GL}(m, 2) \). Define

\[
 f: \mathbb{G} \rightarrow \mathbb{Z} \\
 f(g) := S(0, g_a^\sigma).
\]

Then \( f \) is a homomorphism from \( \mathbb{G} \) to \( \mathbb{Z} \).

The \( \text{MST}_3 \) 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 \( \text{MST}_3 \).
The encryption method of MST as described above is a randomized encryption. However, if we consider \( \mathbb{Z}/\mathbb{Z} \) as the message space and encrypt a message \( z \in \mathbb{Z}/\mathbb{Z} \) by computing

\[
(y_1, y_2) = (\tilde{\alpha}(z), \tilde{\gamma}(z))
\]
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 \( x = 1_{\mathbb{Z}} \).

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. 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 \( \mathbb{Z} \) 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 \( \beta' \) equivalent to \( \beta \), i.e. any \( \beta' \) which is a sandwich transform of \( \beta \). 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 \( \beta^* \) for \( \mathbb{Z} \) such that

\[
\tilde{\beta}^*(x) = \tilde{\beta}(x).c
\]
for all \( x \in \mathbb{Z}/\mathbb{Z} \), where \( c \in \mathbb{Z} \) is a fixed element. For example, if \( \beta^* = [B_1^*, \cdots, B_s^*] \) with \( B_i^* = z_{i-1}^{-1}B_i, z_i \in \mathbb{Z} \), then \( \tilde{\beta}^*(x) = \tilde{\beta}(x).c \), 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 \( \gamma \). In fact, if we replace \( t_i \) by \( t_i^* = t_i.z_i \), for \( z_i \in \mathbb{Z}, i = 0, \ldots, s \), we obtain a \( \beta^* \) such that \( \tilde{\beta}^*(x) = \tilde{\beta}(x).\prod_{i=0}^{s} z_i \). Consequently, the adversary only needs to know the cosets of \( \mathbb{Z} \) in \( G \) 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 \( \mathbb{Z} \) and not the element \( t_i \) itself.

We call a logarithmic signature \( \beta^* \) for \( \mathbb{Z} \) satisfying (4.4) a translation of \( \beta \).

**Definition 4.1** Let \( K = [\beta, f, t_0, \ldots, t_s] \) be a private key for MST. We say that key \( K' = [\beta', f, t_0', \ldots, t_s'] \) is an equivalent to \( K \) if \( \beta' \) is a translation of \( \beta \) and \( t_i' = t_i.z_i \) for some \( z_i \in \mathbb{Z} \) 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

- $S(a_1, b_1)S(a_2, b_2) = S(a_1 + a_2, b_1 + b_2 + a_1a_2)$
- $t_i \in \mathcal{S}, \quad t_i = S(t_{(i,a)}, t_{(i,b)}), \quad \alpha := (a_{ij}), \quad a_{ij} = S(a_{(i,j),a}, a_{(i,j),b}).$
- $g = S(g_a, g_b) \in \mathcal{S}, \quad f(g) := S(0, g_{(a,b)}, \sigma \in \text{Aut}(\mathcal{Z}) = \text{GL}(m, 2)$

Further recall that $g\mathcal{Z} = h\mathcal{Z}$ in $\mathcal{S}$ 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

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

and focus on one block of $\gamma$. W.l.o.g., let us consider the first block. The elements in this block are $h_{11}, h_{12}, \ldots, h_{1r_1}$. 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.

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

$$\sum_{j \in J, |J| \text{even}} h_{(1,j),b} = \sum_{j \in J} a_{(1,j),b} + \sum_{j \in J} b_{(1,j),b} + \sum_{j \in J} a_{(1,j),a}^\sigma + t_{(0),a} \sum_{j \in J} a_{(1,j),a}^\theta + t_{(1),a} \sum_{j \in J} a_{(1,j),a} \quad (4.6)$$

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

$$\sum_{j \in J} a_{(1,j),b} + \sum_{j \in J} h_{(1,j),b} = \sum_{j \in J} b_{(1,j),b} + \sum_{j \in J} a_{(1,j),a}^\sigma + t_{(0),a} \sum_{j \in J} a_{(1,j),a}^\theta + t_{(1),a} \sum_{j \in J} a_{(1,j),a} + t_{(0),a} \sum_{j \in J} a_{(1,j),a}^\theta + t_{(1),a} \sum_{j \in J} a_{(1,j),a} \quad (4.7)$$

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

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

9
Replacing $t_{(0),a}$ in (4.7) yields

$$
\sum_{j \in J, \, |j| \text{ even}} (a_{(1,j),b} + h_{(1,j),b}) = \sum_{j \in J} b_{(1,j),b} + \sum_{j \in J} a_{(1,j),a}^\sigma \\
+ \left( a_{(1,1),a} + t_{(1),a} + h_{(1,1),a} \right) \sum_{j \in J} a_{(1,j),a}^\theta \\
+ t_{(1),a} \sum_{j \in J} a_{(1,j),a} 
$$

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

$$
At_{(1),a}^\theta + Bt_{(1),a} + X = 0 
$$

where

$$
A = \sum_{j \in J} a_{(1,j),a} \\
B = \sum_{j \in J} a_{(1,j),a}^\theta \\
X = \sum_{j \in J} a_{(1,j),b} + \sum_{j \in J} h_{(1,j),b} + \sum_{j \in J} b_{(1,j),b} + \sum_{j \in J} a_{(1,j),a}^\sigma \\
+ \left( a_{(1,1),a} + h_{(1,1),a} \right) \sum_{j \in J} a_{(1,j),a}^\theta 
$$

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

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

### 4.3 Analysis of the equation

The aim of the adversary is to extract information about $\beta$. 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

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

\[
b_J := \sum_{j \in J} b_{\{1, j\}, b} \tag{4.10}
\]

\[
a_J^\sigma := \sum_{j \in J} a_{\{1, j\}, a}^\sigma.
\]

Thus

\[
C_J = b_J + a_J^\sigma, \tag{4.11}
\]

where the values of \( b_J \) and \( a_J^\sigma \) are not determined. Note that we have to determine \( \sigma \) to recover values \( b_J \) and thus gain partial information about \( \beta \). On the other hand, knowing \( \beta \) would lead to reconstructing \( \sigma \).

### 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 \( a_J^\sigma = C_J - b_J \) for \( \sigma \). Note here that \( a_J \) is known. He will try constructing a system of those linearly independent equations and then solving the system to determine \( \sigma \). Now, as the elements in the first block \( B_1 = [b_1, \ldots, b_{r_1}] \) of \( \beta \) are not known, 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 \( \sigma \), this approach leads to a complexity of size \( O(q^m) \). 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 \( \gamma \). 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 \( b_{1j_t} \in D_1 \) from step (ii) to build non-trivial equations of the form \( a_J^\sigma = d_{J^*} \), 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 \( \sigma \).

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 $a_j^T = 0$. An equation $a_j = 0$ does not give any information about $\sigma$, 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 $a_{\{ij\}}$’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 $\{1, \ldots, r_t\}$ 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 $O(q^{1/2})$. Note that in step (i) each even subset $J$ has size at least four.

(ii) Let denote $\mathcal{P} = \{\emptyset, J_1, \ldots, J_w\}$ where $J_i \subseteq \{1, \ldots, r_t\}$ is an even subset with $|J_i| \geq 4$ such that $a_{J_i} = 0$ for $i \geq 1$. 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 $\text{rank}(M)$, 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 $O(q^3)$. Having an accurate estimate of $\text{rank}(M)$ 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 $a_{\{i,1\}}, a_{\{i,2\}}, \ldots, a_{\{i,4\}}$.

Note that $t \geq 4$. If $t = 4$, we have $\text{rank}(M) = t - 3 = 1$. 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 $b_{1j_1} + b_{1j_2} + b_{1j_3} + b_{1j_4} = C_j$ is the only possible equation with 4 unknowns we can obtain.

If $t > 4$, we can prove an even stronger bound that $\text{rank}(M) \leq t - 4$. 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 $\text{rank}(M) \leq t - 5 < t - 4$. 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 $v = 111100 \ldots 0$ (just by renaming the unknowns). Consider $w_4 = 111110 \ldots 0 \in \mathbb{F}_{2^t}$. Let $w_1 = 1000 \ldots 0$, $w_2 = 0100 \ldots 0$, $w_3 = 0010 \ldots 0$. Then $w_1, w_2, w_3, w_4 \notin V$. Let $W$ be the subspace
of \( \mathbb{F}_2 \) 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 \( x = y = 000110 \ldots 0 \in W \). 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, \( \text{rank}(M) \leq t - 4 \). In order to continue the attack we need to guess the values for at least 4 unknowns \( b_{ij} \in \mathbb{F}_q \). Therefore the complexity of step (ii) in this case is at least \( \Theta(q^4) \).

(iii) Let \( D_1 = \{b_{1j_1}, \ldots, b_{1j_k}\} \) be the subset determined after step (ii). In order to be able to recover \( \sigma \in \text{GL}(m, 2) \) it is necessary that \( t \geq m \). Using elements in \( D_1 \) the adversary can construct non-zero subsum \( q_0^j = C_j - b_j \neq 0 \) from Equation (4.9) and tries to solve such a system of equations to recover \( \sigma \). This can be done in polynomial time.

Note that \( t \geq m > 4 \). 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 \( \Theta(q^5, q^{1/2}) \).

This complexity is composed by

- the complexity \( \Theta(q) \) of selecting a correct value for \( t_{(1),a} \) in trinomial (4.8),
- the complexity \( \Theta(q^{1/2}) \) of the birthday attack in step (i) and the complexity of size at least \( \Theta(q^4) \) of solving equations for \( b_{1j} \)'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 \( a_1 \) is much higher than \( \Theta(q^5, q^{1/2}) \). We conjecture that the values \( t - \text{rank}(M) - 1 \) increase in proportion to the growth of \( t \) (i.e. \( \text{rank}(M) \) 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 \( \Theta(q) \). 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 \( \sigma \). So, if the adversary would collect \( m \) linearly independent equations, (s)he could reconstruct \( \sigma \), 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 \( \Theta(q^2, q^{1/2}) \).
4.3.3 Combined Attack on \(b_J\) and \(a_J\)

We can envisage a further method of reconstructing \(\sigma\) 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 \(AX = M\), i.e. \(X = A^{-1}M\).

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

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

and \(\sigma^*\) 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 \(a_{(1)j}\)'s are known, but we have to guess a value for \(b_J\) (an \(m\) bit vector) from the unknowns \(b_{(1)j}\)'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

\[
X := \begin{pmatrix} \sigma^* \\ 1 \end{pmatrix},
\]

where \(1\) is the \(m \times m\) identity matrix. This implies that the complexity of a successful reconstruction of \(\sigma\) (i.e. \(\sigma^* = \sigma\)), after all \(2m\) rows of \(A\) are determined is \(O(q^{2m})\). In this case we have \(Y = 1\).

**Remark 4.7** An pertinent implication of the combined attack is the following fact. If logarithmic signature \(\beta = (b_{1j})\) is of the form \(\beta = (e_{1j})\sigma_1^{\alpha_1}\), where \((e_{1j})\) are known and \(\sigma_1\) is an unknown \(m \times m\) regular matrix over \(\mathbb{F}_2\), this attack will enable to reconstruct \(\sigma\) and \(\sigma_1\) as well. The reason can be seen as follows. Equation (4.11) can now be written as \(a_J\sigma + b_J = a_J\sigma + e_{1j}\sigma_1 = C_J\). If in step (a) we can construct a regular \(2m \times 2m\) matrix \(A\) with rows of the form \(a_J||e_{1j}\), 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 \(\sigma\) and \(\sigma_1\). We see that this is only possible because both \(a_{(1)j}\)'s, \(e_{(1)j}\)'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 \( O(q^5 \cdot q^{1/2}) \), where \( q = \sqrt{|G|} \).

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 \( a(i,j).a \in H < Z \), where \( H \) is a subgroup of \( Z \) of order \( q_0 = 2^\ell \). Then the lower bound given by Proposition 4.8 becomes \( O(q^4 \cdot q_0^{3/2}) \). 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 \( \beta \) and its factorization

In this section we describe a method of generating logarithmic signature \( \beta \) for the realization of MST\(_3\) and show methods of factorization with respect to \( \beta \). As the center \( Z \) of \( G \) is an abelian group, we will use the following possible transformations in generating logarithmic signature \( \beta \).

5.1 Transformations of logarithmic signatures

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

T1 transform each element of \( \varepsilon \) with an automorphism \( \varphi \) of \( H \\
T2 fuse j blocks \( E_{k1}, \ldots, E_{kj} \), i.e. replace blocks \( E_{k1}, \ldots, E_{kj} \) with a new block of the form \( (((E_{k1}, E_{k2}), E_{k3}), \ldots, E_{kj}) \), where \( E_{i1} \cdot E_{ij} := E_{i1} \otimes E_{ij} = [e_{ij_{11}}e_{ij_{12}}, \ldots, e_{ij_{1j_1}}e_{ij_{1j_2}}, e_{ij_{2j_1}}, \ldots, e_{ij_{ij_1}}e_{ij_{ij_2}}, \ldots, e_{ij_{ij_1}}e_{ij_{ij_2}}] \) \nT3 permute the elements within each block \( E_i \) with a permutation \( \pi_1 \) in \( S_{t_1} \) \nT4 permute the blocks \( E_i \)'s with a permutation \( \xi \in S_{v} \) (where \( S_{v} \) is symmetric group on \( v \) symbols)

It is obvious that \( \beta \) obtained from \( \varepsilon \) by using transformations \( T_1, T_2, T_3 \) and \( T_4 \) is a logarithmic signature for \( H \). If \( \varepsilon \) is tame, we can factorize with \( \beta \) 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 \( \beta \)

We will describe an algorithm for generation of a logarithmic signature \( \beta \) 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 \( Z \) of \( G \) 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 \( \mathcal{P} = K_1 \cup \cdots \cup K_v \), \(|K_i| = k_i\), \( \sum_{i=1}^v k_i = m \), 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 \( \mathcal{P} = K_1 \cup \cdots \cup K_v \), \(|K_i| = k_i\), \( \sum_{i=1}^v k_i = m \), 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 \( i \in \{1, \ldots, v\} \), it holds \( \text{int}(d_{ij_1}) < \text{int}(d_{ij_2}) \), where \( \text{int}(d_{ij}) \) 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 \( \delta \) 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 \( \rho \in \text{GL}(m, 2) \) be a regular \( m \times m \) matrix and define \( \delta^* := (d_{ij}^\rho) \). Then \( \delta^* \) is a tame logarithmic signature.

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

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

We now describe an algorithm for generating logarithmic signature \( \beta \).

Algorithm 5.4 (Generation of logarithmic signature \( \beta \))

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 \( \mathbb{Z} \) (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 \( \varepsilon^* = (e_{i,j}^\rho) \) a logarithmic signature obtained from \( \varepsilon \) by filling the positions \( K_i \cup \cdots \cup K_{i-1} \) of each block \( E_i \) with random bits, \( i = 2, \ldots, v \). We call \( \varepsilon^* \) a randomized canonical logarithmic signature.

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

\[
\delta = [D_1, \ldots, D_v] = (d_{ij}) := ((e_{i,j}^\rho)) \]
3. [transformation $T_2$] Select a partition $P = \{P_1, \ldots, P_s\}$, $0 < |P_j|$, of a set $\{1, \ldots, v\}$, such that for each $P_j = \{i_1, \ldots, i_u\}$, we have $i_{h+1} \neq i_h + 1$ for $h, \ell \in \{1, \ldots, u\}$. Fuse blocks $D_{i_1}, \ldots, D_{i_u}$, i.e. construct the product $C_j := ((D_{i_1} \ldots D_{i_k}) \ldots D_{i_u})$. Let $\omega = [C_{i_1}, \ldots, C_s] := (c_{i,j})$ be the resulting logarithmic signature of type $(r_1, \ldots, r_s)$ obtained after this step.

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

$$C_i^* := C_i^{\pi_i} = [c_{i,1}^{\tau_{i_1}}, c_{i,2}^{\tau_{i_2}}, \ldots, c_{i,t_i}^{\tau_{i_t}}],$$

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

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

$$\beta := [B_1, \ldots, B_s] := [C_1^*, \ldots, C_s^*],$$

i.e. $\beta$ is obtained from $\chi$ by permuting the positions of its blocks with $\xi$.

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

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

### 5.3 Factorization with $\beta$

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

**Proposition 5.6** Let $\beta := [B_1, \ldots, B_v]$ be a transversal logarithmic signature for an abelian group $H$. Let $\beta' := [B'_1, \ldots, B'_v]$ be a fused logarithmic signature of $H$ obtained by fusion of blocks of $\beta$ [transformation $T2$]. Then $\beta'$ is equivalent to a non-fused logarithmic signature $\beta''$ obtained from $\beta$ by using certain permutation $\mu \in S_v$ on blocks $B_i$ [transformation $T4$]. In other words $\beta'$ and $\beta''$ induce the same function, i.e. $\beta' = \beta''$.

**Proof.** We observe that $\beta'$ is obtained from $\beta$ by using the following two operations:

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

$$\beta'' := [B''_1, \ldots, B''_v] := [B_{1\mu}, \ldots, B_{v\mu}].$$
(b) select a partition $\mathcal{R} = \{R_1, \ldots, R_s\}$ on the set $\{1, \ldots, v\}$ 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 $\beta''$ according to this partition yields the logarithmic signature $\beta' := \{B'_1, \ldots, B'_u\}$ 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 $r_j = |B''_{i_{j-1}+1}| B''_{i_{j-1}+2} |B''_{i_{j-1}+3} |B''_{i_{j-1}+4} |B''_{i_{j-1}+5} \ldots |B''_{i_{j+1}}|$ for $j = 1, \ldots, s$ and $i_0 = 0$.

(i.e. each block $B'_j$ is obtained by fusing certain consecutive blocks of $\beta''$.)

It is clear that $\beta'$ is equivalent to $\beta''$. \hfill $\Box$

**Remark 5.7** Let $\mathcal{P} = \{P_1, \ldots, P_s\}$ be a partition on the set $\{1, \ldots, v\}$ 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 $\mu \in S_v$ 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

\[
\mathcal{R} := \{R_1 = \{1, 2, \ldots, u_1\}, R_2 = \{u_1+1, \ldots, u_1+u_2\}, \ldots, R_s = \{u_1+\cdots+u_{s-1}+1, \ldots, u_1+\cdots+u_s\}\}.
\]

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 $\beta$.

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

**Algorithm 5.8 (Factorization with FT signature $\beta$)**

**Input:** $y$, $\varepsilon^*$, $\mu$, $\mathcal{R} = \{R_1, \ldots, R_s\}$, $\xi$, $\pi_1, \ldots, \pi_s$, $\rho$.

**Output:** $x = x_1 \parallel x_2 \parallel \ldots \parallel x_s$, where $y = \bar{\beta}(x)$.

1. Compute $z = (y^{\rho^{-1}})$ and write $z = z_1 \parallel z_2 \parallel \ldots \parallel z_v$. Each $z_i$ is of bit length $k_i$.
2. Factorize $z$ with respect to $\varepsilon^*$ by using Algorithm 5.9. Let denote $j'_1, \ldots, j'_s$ the indices obtained by this factorization.
3. Compute $j_\ell = j'_\ell \mu^{-1}$ for $\ell = 1, \ldots, v$.
4. According to $\mathcal{R}_\ell = \{i_1, 1, \ldots, i_1, u_1\}$ set $x'_\ell = j_1 \parallel j_2 \parallel \ldots \parallel j_{u_\ell}$ for $\ell = 1, \ldots, s$.
5. Compute $x''_\ell = (x'_\ell)^{\pi_\ell^{-1}}$ and finally $x_\ell = x''_{\ell-1}$ 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 $\varepsilon^*$ generated in step (1) of Algorithm 5.4.
Let \( x = x_1\|x_2\|\ldots\|x_v \) be a binary vector of length \( m \), where \( x_i \) is of length \( k_i \) for \( i = 1, \ldots, v \) and \( t_i = 2^{k_i} \). Let \( y = \tilde{e}^*(x) \). Write \( y = y_1\|y_2\|\ldots\|y_v \), where each \( y_i \) is of bit length \( k_i \).

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

**Algorithm 5.9 (Factorization with \( \varepsilon^* \))**

**Input:** \( y = y_1\|y_2\|\ldots\|y_v, \varepsilon^* \).

**Output:** \( x = x_1\|x_2\|\ldots\|x_v \), where \( y = \tilde{e}^*(x) \).

1. Starting with \( y_v \), we find an element \( e^*_{v,j} \) in block \( E^*_v \) such that the last \( k_v \) bits of \( e^*_{v,j} \) are equal to \( y_v \). Such \( e^*_{v,j} \) is uniquely determined since the last \( k_v \) bits of elements in \( E^*_v \) form a vector space of dimension \( k_v \). The index \( j \) of \( e^*_{v,j} \) in block \( E^*_v \) determines the index \( x_v \).

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

Now we describe an algorithm for factorization with respect to an NFT logarithmic signature \( \beta^* \).

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

**Algorithm 5.10 (Factorization with NFT signature \( \beta^* \))**

**Input:** \( z = z_1\|z_2\|\ldots\|z_v, \beta^*, \xi, \pi_1, \ldots, \pi_v, \rho \).

**Output:** \( x = x_1\|x_2\|\ldots\|x_v \), where \( z = \tilde{\beta}^*(x) \).

1. Using \( \xi^{-1}, \pi_1^{-1}, \ldots, \pi_v^{-1} \) and \( \rho^{-1} \) construct \( \varepsilon^* \) from \( \beta^* \).

2. Compute \( y = (z^\rho)^{-1} \) and write \( y = y_1\|y_2\|\ldots\|y_v \). Each \( y_i \) is of bit length \( k_i \).

3. Factorize \( y \) with respect to \( \varepsilon^* \) by using Algorithm 5.9. Let denote \( x'_1, \ldots, x'_v \) the indices obtained by this factorization.

4. Compute \( x_i'' = (x'_i)^{\pi_i^{-1}} \) and finally \( x_i = x_i'' \) for \( i = 1, \ldots, v \).

**6 Attack on ciphertexts**

This section deals with an elaborated chosen plaintext attack on MST, when transversal logarithmic signatures are used. This is the case when \( \beta \) 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 $\mathbb{Z}$. However, the structure of $\beta$ will be changed if 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 $\alpha$ and $\gamma$, are merely random covers for the center $\mathbb{Z}$. Note that $\mathbb{Z}$ is a vector space of dimension $m$ over $\mathbb{F}_2$ and we also indentify $\mathbb{Z}$ with $\mathbb{F}_{q^r}$. So we call elements of $\mathbb{Z}$ vectors as well. Let denote $\alpha_{a}$ the cover of $\mathbb{Z}$ whose blocks are formed by the a.part of $\alpha$. Define $J := \tilde{\alpha}_{a}(\mathbb{Z} \setminus \mathbb{Z}_{1})$. Thus $J$ is a subset of $\mathbb{Z}$ and the ratio $\rho := |\mathbb{Z}|/|J|$ may be viewed as the average number of representations for each element of $J$ 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 $\rho$ given by the following formula

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

where $\lambda = \frac{1}{|\mathbb{Z}_{1}|} \prod_{i=1}^{s} r_i$, and $(r_1, \ldots, r_s)$ is the type of $\alpha_{a}$, and $\mathbb{Z}_{1}$ is the smallest subgroup of $\mathbb{Z}$ containing $J$. 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 $\tilde{\alpha}_{a}$. 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 $\rho$ or/and $r_i$ are increased, this probability will be decreased. So, if we select the elements of $\alpha_{a}$ from a subspace $\mathbb{Z}_{1}$ of $\mathbb{Z}$ such that $\rho = |\mathbb{Z}|/|\mathbb{Z}_{1}|$ 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 $\beta$ (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 $\beta$ that would eventually allow the adversary to decrypt any given ciphertext.
Used notation:

Let \( \mathbf{w} := (w_{i,j}) \) be a cover of type \( \{r_1, \ldots, r_s\} \) for \( \mathcal{G} \), and let \( x \in \mathbb{Z}_{|G|} \) 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 \( v_\ell := t^{\phi} \) for \( \ell \in \{1, \ldots, s\} \). Define

\[
\hat{w}_{k, \phi}(x) := \prod_{i=1}^{k} w_{v_i, j_{v_i}} \tag{6.12}
\]

We consider an NFT-MST scheme. Let \( [\alpha, \gamma] \) be the public key with the corresponding private key \([\beta, f, t_0, \ldots, t_s]\). Recall that \( \alpha := (a_{i,j}) \), \( \beta := (b_{i,j}) \) and \( \gamma := (h_{i,j}) \) are of type \( \{r_1, \ldots, r_s\} \) and that \( \xi \) 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 \( \alpha, \beta, \gamma \) be the covers of type \( \{r_1, \ldots, r_s\} \) as described above. Let \( x \in \mathbb{Z}_{|G|} \) correspond to \( \{j_1, \ldots, j_s\} \in \mathbb{Z}_{r_1} \oplus \ldots \oplus \mathbb{Z}_{r_s} \) and \( v_\ell := t^{\ell} \) for \( \ell \in \{1, \ldots, s\} \). Further let \( \tilde{\alpha}_\ell, \ell(x), \tilde{\beta}_\ell, \ell(x), \tilde{\gamma}_\ell, \ell(x) \) be the values computed by Equation (6.12). Let \( k_\ell := \lceil \log_2 r_\ell \rceil \). Then there exists a binary \( (2m + 1) \times k_{v_\ell} \) matrix \( M_{v_\ell} \) such that

\[
(\tilde{\alpha}_\ell, \ell(x), a \parallel \tilde{\alpha}_\ell, \ell(x), b + \tilde{\gamma}_\ell, \ell(x), b \parallel 1) \quad M_{v_\ell} = \pi_{v_\ell}(j_{v_\ell}). \tag{6.13}
\]

where “1” is the bit set to one.

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

\[
(\tilde{\alpha}_\ell, \ell(x), a \parallel \tilde{\alpha}_\ell, \ell(x), b + \tilde{\gamma}_\ell, \ell(x), b \parallel 1) \quad N_{v_\ell} = \begin{pmatrix} \tilde{\beta}_\ell, \ell(x), b \end{pmatrix} \tag{6.14}
\]

We begin with

\[
\tilde{\alpha}_\ell, \ell(x), b + \tilde{\gamma}_\ell, \ell(x), b = \sum_{i=1}^{\ell} b_{\{v_i, j_{v_i}\}, b} + t_{(0), a} \sum_{i=1}^{\ell} a_{\{v_i, j_{v_i}\}, a}^{\theta} + t_{(\ell), a} \sum_{i=1}^{\ell} a_{\{v_i, j_{v_i}\}, a}^{\phi} + \sum_{i=1}^{\ell} a_{\{v_i, j_{v_i}\}, a}^{\sigma} + C_\ell
\]

where

\[
C_\ell = t_{(0), b} + t_{(0), a}^{\theta+1} + t_{(\ell), b} + t_{(0), a}^{\phi} t_{(\ell), a}^{\theta}.
\]

As the elements \( t_{(0), a}, t_{(\ell), a} \in \mathbb{F}_q \) are constants, the products \( t_{(0), a} \sum a_{\{v_i, j_{v_i}\}, a}^{\theta} \) and \( t_{(\ell), a} \sum a_{\{v_i, j_{v_i}\}, a}^{\phi} \) present linear mappings. Therefore there exist binary \( m \times m \) matrices \( T_0 \) and \( T_{v_\ell} \) such that

\[
t_{(0), a} \sum_{i=1}^{\ell} a_{\{v_i, j_{v_i}\}, a}^{\theta} = \sum_{i=1}^{\ell} a_{\{v_i, j_{v_i}\}, a} T_0
\]

\[
t_{(\ell), a} \sum_{i=1}^{\ell} a_{\{v_i, j_{v_i}\}, a}^{\phi} = \sum_{i=1}^{\ell} a_{\{v_i, j_{v_i}\}, a} T_{v_\ell}
\]
Set

\[
N_{\ell} = \begin{pmatrix}
T_0 + T_{\ell} + \sigma \\
I_m \\
C_{\ell}
\end{pmatrix}
\]

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

\[
\left( \bar{B}_{\ell,\ell}(x),_b \right)^{\rho-1} = \left( \bar{\epsilon}'_{\ell,\ell}(x),_b \right)
\]

Consider the linear mapping \(\varphi_{\ell}\) defined by

\[
\bar{\epsilon}'_{\ell,\ell}(x),_b = \sum_{i=1}^{\ell} e'_{i,j_{\ell,i}},_b \xrightarrow{\varphi_{\ell}} j_{\ell}
\]

where \(\text{id}\) is identity permutation. This mapping is well defined for the class of transversal logarithmic signatures, in particular for \(\epsilon^*\) created in Algorithm 5.4 after step (1). Note that \(j_{\ell}\) is the binary representation of the index for \(e_{\ell,j_{\ell}}\) and is identical with the \(k_{\ell}\) bit vector of \(e_{\ell,j_{\ell}}\) at the positions \(K_{\ell}\). Let \(\epsilon'' := (e''_{ij})\) be obtained from \(\epsilon^*\) by applying step (4) of Algorithm 5.4. Now observe that \(\varphi_{\ell}\) acts on \(\bar{\epsilon}''_{\ell,\ell}(x),_b\) as follows

\[
\bar{\epsilon}''_{\ell,\ell}(x),_b = \sum_{i=1}^{\ell} e''_{i,j_{\ell,i}},_b \xrightarrow{\varphi_{\ell}} \pi_{\ell}(j_{\ell})
\]

Applying step (5) of Algorithm 5.4 on \(\epsilon''\) we get \(\epsilon'\). Therefore \(\varphi_{\ell}\) acts on \(\bar{\epsilon}'_{\ell,\ell}(x),_b\) according to

\[
\bar{\epsilon}'_{\ell,\ell}(x),_b = \sum_{i=1}^{\ell} e'_{i,j_{\ell,i}},_b \xrightarrow{\varphi_{\ell}} \pi_{\ell}(j_{\ell})
\]

Let \(P_{\ell}\) be the \(m \times k_{\ell}\) binary matrix representation of the mapping \(\varphi_{\ell}\). Then we can write

\[
\left( \bar{\epsilon}'_{\ell,\ell}(x),_b \right) P_{\ell} = \pi_{\ell}(j_{\ell})
\]

Define the matrix \(M_{\ell}\) as

\[
M_{\ell} := N_{\ell} \cdot \rho^{-1} \cdot P_{\ell}
\]

Then \(M_{\ell}\) is the binary matrix that satisfies Equation (6.13).

Let \(M_{\ell,p}\) denote the \(p\)-th column of the matrix \(M_{\ell}\), where \(p = 1, \ldots, k_{\ell}\). We observe that \(\pi_{\ell}(j_{\ell})\) is a binary vector of length \(k_{\ell}\). 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 \( \nu_\ell := \ell^k \) and \( M_{\nu_\ell} \) be the \( \ell \)-th column of \( M_{\nu_\ell} \) and \( \pi_{\nu_\ell} \) be the \( \ell \)-th bit of \( \pi_{\nu_\ell} \) from Proposition 6.1. Then we have

\[
\left( \check{\alpha}_{\ell, \xi}(x), a \parallel \check{\alpha}_{\ell, \xi}(x), b + \check{\gamma}_{\ell, \xi}(x), a \parallel 1 \right) M_{\nu_\ell} = \pi_{\nu_\ell}(j_{\nu_\ell}). \tag{6.15}
\]

Proposition 6.3 Let \( \alpha, \beta, \gamma \) be the covers of type \( (r_1, \ldots, r_s) \) as described above. Let \( x \in \mathbb{Z}_s \) correspond to \( [j_1, \ldots, j_s] \in \mathbb{Z}_{r_1} \oplus \ldots \oplus \mathbb{Z}_{r_s} \). Further let \( \nu_\ell := \ell^k \) for \( \ell \in \{1, \ldots, s\} \) and \( k_\ell := \lfloor \log_2 r_\ell \rfloor \). Then there exists a binary \((2m+1) \times k_\ell\) matrix \( L_{\nu_\ell} \) such that

\[
\left( \check{\alpha}(x), a + A_\ell \parallel \check{\alpha}(x), b + \check{\gamma}(x), a + B_\ell \parallel 1 \right) L_{\nu_\ell} = \pi_{\nu_\ell}(j_{\nu_\ell}) \tag{6.16}
\]

where

\[
A_\ell := \sum_{i=\ell+1}^{s} a_{(i,j_{\nu_\ell})} \cdot a
\]

\[
B_\ell := \sum_{i=\ell+1}^{s} (a_{(i,j_{\nu_\ell})} \cdot b + h_{(i,j_{\nu_\ell})} \cdot b) + \sum_{i=\ell+1}^{s} a_{(i,j_{\nu_\ell})} \cdot a \left( t(0).a + t(j_{\nu_{i-1}}) \cdot a \right) + \sum_{i=\ell+1}^{s} a_{(i,j_{\nu_\ell})} \cdot a \left( t(j_{\nu_i}).a + t(s).a \right) \cdot \theta
\]

for \( \ell \in \{1, \ldots, s-1\} \), \( A_0 = B_0 := (0, \ldots, 0) \), 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

\[
\check{\alpha}(x).a + \sum_{i=\ell+1}^{s} a_{(i,j_{\nu_\ell})} \cdot a = \sum_{i=1}^{\ell} a_{(i,j_{\nu_\ell})} \cdot a
\]

\[
\check{\beta}(x).b + \sum_{i=\ell+1}^{s} b_{(i,j_{\nu_\ell})} \cdot b = \sum_{i=1}^{\ell} b_{(i,j_{\nu_\ell})} \cdot b
\]

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

\[
\left( \sum_{i=1}^{\ell} a_{(i,j_{\nu_\ell})} \cdot a \parallel \check{\alpha}(x), b + \check{\gamma}(x), b + B_\ell \parallel 1 \right) N_{\nu_\ell} = \sum_{i=1}^{\ell} b_{(i,j_{\nu_\ell})} \cdot b \tag{6.17}
\]

Here we have

\[
\check{\alpha}(x).b + \check{\gamma}(x).b + B_\ell
\]

\[
= \check{\alpha}(x).b + \check{\gamma}(x).b + \sum_{i=\ell+1}^{s} \left( a_{(i,j_{\nu_\ell})} \cdot b + h_{(i,j_{\nu_\ell})} \cdot b \right) + \sum_{i=\ell+1}^{s} a_{(i,j_{\nu_\ell})} \cdot a \left( t(0).a + t(j_{\nu_{i-1}}) \cdot a \right) + \sum_{i=\ell+1}^{s} a_{(i,j_{\nu_\ell})} \cdot a \left( t(j_{\nu_i}).a + t(s).a \right) \cdot \theta
\]

23
\[
\begin{align*}
&= \sum_{i=1}^{s} b_{(v_i,j_{v_i})}.b + \sum_{i=1}^{s} a_{(v_i,j_{v_i})}.a + t(0).b + t^{\theta+1}(0).a + t(s).b + t(0).a t^\theta(s).a + \\
t(0).a \sum_{i=1}^{s} a_{(v_i,j_{v_i})}.a + t^\theta(s).a \sum_{i=1}^{s} a_{(v_i,j_{v_i})}.a + \sum_{i=1}^{s} b_{(v_i,j_{v_i})}.b + \sum_{i=1}^{s} a_{(v_i,j_{v_i})}.a + \\
&= \sum_{i=1}^{s} (t_{(v_i-1)}.a a_{(v_i,j_{v_i})}.a + a_{(v_i,j_{v_i})}.a t^\theta_{(v_i)}.a + t_{(v_i-1)}.a a_{(v_i,j_{v_i})}.a + t_{(v_i-1)}.b + \\
t^{\theta+1}_{(v_i-1)}.a + t_{(v_i)}.b) + t(0).a \sum_{i=1}^{s} a_{(v_i,j_{v_i})}.a + \sum_{i=1}^{s} a_{(v_i,j_{v_i})}.a t_{(v_i-1)}.a + \\
&= \sum_{i=1}^{\ell} b_{(v_i,j_{v_i})}.b + \sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a + t(0).a \sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a + t^\theta(s).a \sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a + C_\ell
\end{align*}
\]

where the term

\[
C_\ell = \sum_{i=1}^{s} (t_{(v_i-1)}.a a_{(v_i,j_{v_i})}.a + a_{(v_i,j_{v_i})}.a t^\theta_{(v_i)}.a + t_{(v_i-1)}.a a_{(v_i,j_{v_i})}.a + t_{(v_i-1)}.b + \\
t^{\theta+1}_{(v_i-1)}.a + t_{(v_i)}.b) + t(0).a + t(s).b + t(0).a t^\theta(s).a
\]
is viewed as a constant in \( F_{2^m} \). Therefore, Equation (6.17) becomes

\[
\begin{pmatrix}
\sum_{i=1}^{\ell} b_{(v_i,j_{v_i})}.b + \sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a \\
\sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a \\
\sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a
\end{pmatrix}
\begin{pmatrix}
\sum_{i=1}^{\ell} b_{(v_i,j_{v_i})}.b + \sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a \\
\sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a \\
\sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a + C_\ell
\end{pmatrix}
\begin{pmatrix}
n_{v_\ell} \\
1
\end{pmatrix}
\]

(6.18)

Because the elements \( t(0).a \) and \( t(s).a \) are constants, and \( \theta \) is a linear mapping, there exist \( m \times m \) matrices \( T_0, T_\ell \) such that

\[
\begin{align*}
t(0).a \sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a &= \sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a T_0 \\
t^\theta(s).a \sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a &= \sum_{i=1}^{\ell} a_{(v_i,j_{v_i})}.a T_\ell
\end{align*}
\]

Now set

\[
N_{v_\ell} = \begin{pmatrix}
T_0 + T_\ell + \sigma \\
I_m \\
C_\ell
\end{pmatrix}
\]

24
where $I_m$ is the $m \times m$ identity matrix. Then it is easy to check that the $(2m + 1) \times m$ matrix $N_{\nu_t}$ satisfies Equation (6.18). Similar to the proof of Proposition 6.1, by using

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

and

$$
(\tilde{\varepsilon}'_{\ell,\xi}(x) \cdot b) \cdot P_{\nu_t} = \pi_{\nu_t}(j_{\nu_t})
$$

we define

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

Then $L_{\nu_t}$ 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 $\xi$, as well.

**Algorithm 6.4 (Matrix-permutation Attack on NFT-MST₃: 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 $x^{(i)} \mapsto (j^{(i)}_1, \ldots, j^{(i)}_s)$, and construct vectors $y^{(i)} := \left( \tilde{\alpha}_{\ell,\text{id}}(x^{(i)})_a \parallel \tilde{\alpha}_{\ell,\text{id}}(x^{(i)})_b + \gamma_{\ell,\text{id}}(x^{(i)})_b \parallel 1 \right)$, 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 $\left( \tilde{\alpha}_{\ell,\text{id}}(x^{(i)})_a \right)$.

(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 $\mathbb{F}_{2k_v}$.

(C.2) Choose a random binary vector $w = (w_1, \ldots, w_{k_v}) \in \mathbb{F}_{2k_v}$, and set $\pi_{v,p}(j_i) = w_i$ for each $j_i \in J_v$.

(C.3) Choose random plaintexts $x^{(i)} \mapsto (j^{(i)}_1, \ldots, j^{(i)}_v, \ldots, j^{(i)}_s)$, where $j^{(i)}_v \in J_v$ and construct vectors $y^{(i)} := \left( \tilde{\alpha}_{\ell,\text{id}}(x^{(i)})_a \parallel \tilde{\alpha}_{\ell,\text{id}}(x^{(i)})_b + \gamma_{\ell,\text{id}}(x^{(i)})_b \parallel 1 \right)$, 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 $\text{rank}(Y_v) < n_{\ell}$ then return to (C.1).

(C.4) Let $x^{(i)} \mapsto (j^{(i)}_1, \ldots, j^{(i)}_v, \ldots, j^{(i)}_s)$, 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 $\pi_{v,p}(j^{(i)}_v)$ as entry in row $(i)$. 

25
Construct a $(2m + 1) \times n_\ell$ binary encoding matrix $E_v$, such that $\text{rank}(Y_v, E_v) = n_\ell$.

Compute matrix $M_{v,p} = E_v(Y_v, E_v)^{-1}Z_{v,p}$.

For each $j_v \in F_{2^{k_v}} \setminus J_v$ choose a random plaintext $x \mapsto (j_1, \ldots, j_v, \ldots, j_s)$ and compute the value for $\pi_{v,p}(j_v)$ by $\pi_{v,p}(j_v) := (\tilde{a}_{\ell,1}(x).a \| \tilde{a}_{\ell,1}(x).b + \tilde{y}_{\ell,1}(x).b \| 1).M_{v,p}$.

Choose a random plaintext $x \mapsto (j_1, \ldots, j_v, \ldots, j_s)$ and compute the value $y = (\tilde{a}_{\ell,1}(x).a \| \tilde{a}_{\ell,1}(x).b + \tilde{y}_{\ell,1}(x).b \| 1).M_{v,p}$.

If $y \neq \pi_{v,p}(j_v)$ then return to (C.2) and try another choice for $w \in F_{2^{k_v}}$ (this can be done in at most $2^{k_v}$ times). If no choice for $w$ in (C.2) is possible, then set $v \leftarrow (v + 1)$ and return to (C).

If $y = \pi_{v,p}(j_v)$, repeat (C.8) for an appropriate number of times.

**done**

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

For each $j_v \in F_{2^{k_v}}$, by using $\pi_{v,p}(j_v)$ for $p = 1, \ldots, k_v$, one obtains $\pi_v(j_v)$, and thus determines permutation $\pi_v$.

**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 $n_\ell$ we have to run this step for a sufficient number of random inputs $x^{(i)}$.

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

(C) The inner loop is used to determine each bit $\pi_{v,p}(j_v)$ of $\pi_v(j_v)$ for $p = 1, \ldots, k_v$, separately, for which $\pi_v(j_v) := (\pi_{v,1}(j_v)) \cdots (\pi_{v,k_v}(j_v))$ for all $j_v \in F_{2^{k_v}}$.

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

(C.2) In this step we guess the $p$-th bit $\pi_{v,p}(j_v)$ of $\pi_v(j_v)$ for all $j_v \in J_v$.

(C.3) In this step a plaintext $x^{(i)} \mapsto (j_1^{(i)}, \ldots, j_v^{(i)}, \ldots, j_s^{(i)})$ is chosen in such a way that the component $j_v^{(i)}$ belongs to $J_v$ (chosen in step (C.1)). The other components $j_u$ with $u \neq v$ are arbitrarily chosen. We repeat this step until we get a matrix $Y_v$ with $\text{rank}(Y_v) = n_\ell$.

If the elements of $J_v$, $|J_v| = k_v$, are chosen in such a way that the set $\{\pi_v(j_v) \mid j_v \in J_v\}$ has less than $k_v$ linearly independent vectors (of size $k_v$), the $\text{rank}(Y_v)$ will be
smaller than \( n_\ell \). In this case the algorithm returns to step (C.1), and generates a new set \( J_v \).

(The other possibility could be to extend the size of set \( J_v \), i.e. \(|J_v| > k_v\).)

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

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

(C.6) Using \( E_v \) from step (C.5) we determine the \( p\)-th column \( M_{v,p} \) of the matrix \( M_v \).

(C.7) This step computes the \( p\)-th bit \( \pi_{v,p}(j_v) \) of \( \pi_v(j_v) \) for all remaining \( j_v \in F_{2^{k_v}} \).

(C.8) This step verifies whether the bit \( \pi_{v,p}(j_v) \) guessed in step (C.2) or computed in step (C.7) for all \( j_v \in F_{2^{k_v}} \) is correct, and whether the value \( v \) satisfies \( v = \ell^x \). Running this step in an appropriately sufficient number of times allows us to check these requirements.

(D) In this step we use \( v = \ell^x \) determined in the previous loop to construct a transposition \( \xi_\ell \). We update \( \alpha \) and \( \gamma \), permuting their blocks with \( \xi_\ell \) and continue the main loop with the new value \( \ell \leftarrow (\ell - 1) \).

(E) From the \( p\)-th bit \( \pi_{v,p}(j_v) \) for all \( p = 1, \ldots, k_v \) we construct \( \pi_v(j_v) \). By collecting all \( \pi_v(j_v) \), \( j_v \in F_{2^{k_v}} \), we are able to recover the permutation \( \pi_v \).

**Proposition 6.5** Let \( \alpha, \gamma \) 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 \([\pi_1, \ldots, \pi_s, \ell] \) using Algorithm 6.4 is bounded by \( \Theta(\sum_{\ell=1}^s \ell k_\ell 2^{k_\ell - 1}) \).

**Proof.** In step (C.2) of Algorithm 6.4 we have to guess vector \( w \) of \( k_v \) bits to set the \( p\)-th bit \( \pi_{v,p}(j_v) \) of \( \pi_v(j_v) \) for all \( j_v \in J_v \). The complexity of the algorithm includes the times required to run through all bits \( p \in \{1, \ldots, k_v\} \) with an average of \( \ell/2 \) times until step (C.8) successfully terminates by finding \( v := \ell^x \), 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. \( \square \)

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

\[
(t_{(0)}, a) + t_{(\ell - 1)}, a) = \sum_{m=1}^{\ell - 1} (a_{(m, j_m)}, a + h_{(m, j_m)}, a) = \sum_{m=1}^{\ell - 1} (a_{(m, 1)}, a + h_{(m, 1)}, a)
\]

\[
(t_{(\ell)}, a) + t_{(s)}, a) = \sum_{m=\ell + 1}^{s} (a_{(m, j_m)}, a + h_{(m, j_m)}, a) = \sum_{m=\ell + 1}^{s} (a_{(m, 1)}, a + h_{(m, 1)}, a)
\]

We use Proposition 3.3 to design the following algorithm.
Algorithm 6.6 (Matrix-permutation Attack on NFT\textsubscript{3}: Matrix recovery)

**Input:** Public key \([\alpha, \gamma]\), permutations \([\pi_1, \ldots, \pi_s, \xi]\).

**Output:** Matrices \([L_1, \ldots, L_s]\).

Set \(A_s \leftarrow (0, \ldots, 0)\), an \(m\)-bit zero vector.

**For** \(\ell \leftarrow s\) **downto** 1 **do**

(A) Set \(v \leftarrow \ell^s\).

(B) Select random plaintexts \(x^{(i)} \mapsto (j_1^{(i)}, \ldots, j_s^{(i)})\), and construct vectors
\[
y^{(i)} := \left( \tilde{\alpha}_{\ell, \xi}(x)^{(i)}_a \parallel \tilde{\gamma}(x)^{(i)}_b + \tilde{\gamma}(x)^{(i)}_b \parallel A^{(i)}_\ell \right) \parallel 1\]

as in Proposition 6.3. Define \(n_v\) to be the maximum number of linearly independent vectors \(y^{(i)}\), where \(n_v = n'_v + 1 + \sum_{m=1}^\ell k_{m, \ell}\). Here \(n'_v\) is the maximum number of linearly independent columns of the matrix formed by vectors \(\tilde{\alpha}_{\ell, \xi}(x)^{(i)}_a\). Repeat this step for an appropriate number of choices of \(x^{(i)}\) and form a matrix \(Y_v\) with \(n_v\) rows being the linearly independent vectors \(y^{(i)}\).

(C) Let \(x^{(i)} \mapsto (j_1^{(i)}_1, \ldots, j_s^{(i)}_s)\), for \(i = 1, \ldots, n_v\), be the plaintext used to construct row \(i\) of the \(n_v \times (2m + 1)\) binary matrix \(Y_v\) in the previous step. Form the \(n_v \times k_v\) matrix \(Z_v\) with value \(\pi_v(j_v)\) as entry in row \(i\).

(D) Construct a \((2m + 1) \times n_v\) binary encoding matrix \(E_v\), such that \(\text{rank}(Y_v, E_v) = n_v\).

(E) Compute matrix \(L_v = E_v \cdot (Y_v, E_v)^{-1} \cdot Z_v\)

   If \(\ell = 1\) then return \([L_1, \ldots, L_s]\).

(F) Set
\[
A_{\ell-1} \leftarrow A_\ell + a^{(v, j_v)}_a + h^{(v, j_v)}_a + a^{(v, j_v)}_a \sum_{m=1}^{\ell-1} (a^{(m, 1)}_a + h^{(m, 1)}_a) + \sum_{m=\ell+1}^s (a^{(m, 1)}_a + h^{(m, 1)}_a)^\theta
\]

---

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\textsubscript{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 \(y_1 = \tilde{\alpha}(x)\), \(y_2 = \tilde{\gamma}(x)\).

Set \(A_s \leftarrow (0, \ldots, 0)\).

**For** \(\ell \leftarrow s\) **downto** 1 **do**

---

28
Set \( v \leftarrow \ell \xi \).

(2) Construct a vector
\[
\mathbf{w} = \left( \mathbf{y}_1.a \parallel \mathbf{y}_1.b \parallel \mathbf{y}_2.b \parallel \mathbf{A}_\ell \parallel 1 \right)
\]

(3) Compute \( \pi_v(j_v) = \mathbf{w} \cdot \mathbf{L}_v \)

(4) Recover \( j_v \) using \( \pi_v(j_v) \) and permutation \( \pi_v \).

If \( \ell = 1 \) then return \( \{j_1, \ldots, j_s\} \).

(5) Set \( \mathbf{y}_1.a \leftarrow \mathbf{y}_1.a \oplus \mathbf{a}_{(v,j_v),a} \)

\[
\mathbf{A}_{\ell-1} \leftarrow \mathbf{A}_\ell + \mathbf{a}_{(v,j_v),b} + \mathbf{h}_{(v,j_v),b} + \sum_{m=1}^{\ell-1} \left( \mathbf{a}_{(m,\ell,1),a} + h_{(m,\ell,1),a} \right) + \sum_{m=\ell+1}^{s} \left( \mathbf{a}_{(m,\ell,1),a} + h_{(m,\ell,1),a} \right)^\theta
\]

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 \( [\mathbf{L}_1, \ldots, \mathbf{L}_s] \). The knowledge of \( [\mathbf{L}_1, \ldots, \mathbf{L}_s] \) 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 \( \mathbf{L}_i \) for any block \( i = \{1, \ldots, s\} \) and to compute the image \( \pi_i(j_i) \) of \( j_i \) under permutation \( \pi_i \) as shown in Proposition 6.3. This fact is used in step (3) of Algorithm 6.7. As \( \pi_i \) 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 \( [\mathbf{L}_1, \ldots, \mathbf{L}_s] \) 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 \( O(\sum_{\ell=1}^{s} \ell k_\ell 2^{k_\ell-1}) \).

The complexity as given in Proposition 6.9 shows in particular that for relatively small values \( k_\ell \), which are usually used in a real version of the MST_3 scheme, say \( k_1 \leq 15 \), the non-fused transversal logarithmic signatures cannot be used for a secure realization of MST_3.
6.3 The Matrix-permutation attack on FT-MST₃

In this section we attempt to determine the complexity of the Matrix-permutation attack on FT-MST₃.

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 \( \beta \) (Algorithm 5.10), even though the adversary does not know \( \beta \). Thus, the knowledge provided by a factorization with respect to \( \beta \) 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-MST₃ we confine ourselves to use step (1) and step (3) of Algorithm 5.4 only to create logarithmic signature \( \beta \).

Let \( \{K₁, K₂, \ldots, Kᵥ\} \) be a partition on the set \{1, \ldots, m\} with \(|Kᵢ| = kᵢ\) and \(tᵢ = 2ᵏᵢ\) as described in Algorithm 5.4. Let \( \varepsilon^* := (ε^*_{₁,j}) \) be a signature of type \((t₁, \ldots, tᵥ)\) created after step (1) of the Algorithm 5.4.

W.l.o.g. we consider \( \beta := (b₁,j₁) \) to be a logarithmic signature created by fusion of blocks \((ℓ, ℓ + 2)\) and \((ℓ + 1, ℓ + 3)\) of \( ε^* \). (Note that no consecutive blocks are fused.) Then \( \beta = [B₁, \ldots, Bₙ] \) is of type \((r₁, \ldots, rₙ)\), where \( rᵢ = tᵢ \cdot tᵢ₊₂ \) and \( rᵢ₊₁ = tᵢ₊₁ \cdot tᵢ₊₃ \). We now consider one fused block, say \( Bₙ₊₁ \), of \( \beta \).

Let \( u[n]_{₁,j₁} \) (resp. \( e[n]_{₁,j₁} \)) be a vector of length \( kₙ \), consisting of the bits of \( b₁,j₁ \) on the positions corresponding to \( Kₙ \).

Let \( x := (j₁, \ldots, jᵥ) \), also let \( x' := (j'_₁, \ldots, j'_ᵥ) \), where \( j'_ℓ = jℓ || jₖ₊₁ \) and \( j'_ℓ₊₁ = jℓ₊₂ || jₖ₊₃ \).

Then

\[
\begin{align*}
ε^*_{₁,j₁} &= (\ldots || \bar{0} || \bar{0} || \bar{0} || \bar{0} || \bar{0} || \ldots) \\
ε^*_{ℓ₋₁,jₖ₋₁} &= (\ldots || \bar{0} || \bar{0} || \bar{0} || \bar{0} || \bar{0} || \ldots) \\
ε^*_{ₖ,jₖ} &= (\ldots || ε^*_{ₖ,jₖ} || \bar{0} || \bar{0} || \bar{0} || \bar{0} || \ldots) \\
ε^*_{ₖ₊₁,jₖ₊₂} &= (\ldots || u[u[,]_{ₖ₊₁,jₖ₊₂} || ε^*_{ₖ₊₁,jₖ₊₂} || \bar{0} || \bar{0} || \bar{0} || \ldots) \\
ε^*_{ₖ₊₂,jₖ₊₃} &= (\ldots || u[u[,]_{ₖ₊₂,jₖ₊₃} || u[u[,]_{ₖ₊₂,jₖ₊₃} || ε^*_{ₖ₊₂,jₖ₊₃} || \bar{0} || \bar{0} || \ldots) \\
ε^*_{ₖ₊₃,jₖ₊₄} &= (\ldots || u[u[,]_{ₖ₊₃,jₖ₊₄} || u[u[,]_{ₖ₊₃,jₖ₊₄} || u[u[,]_{ₖ₊₃,jₖ₊₄} || u[u[,]_{ₖ₊₃,jₖ₊₄} || ε^*_{ₖ₊₃,jₖ₊₄} || \ldots)
\end{align*}
\]

Then

\[
\tilde{β}(x') = b₁,j₁ + \ldots + bₖ,jₖ + bₖ₊₁,jₖ₊₁ + \ldots + bᵥ,jᵥ
\]
where
\[
\begin{align*}
\mathbf{b}_{\ell,j}^{[\ell]} &= (\ldots || \mathbf{c}_{\ell}^{[\ell]} || \mathbf{u}_{\ell+2,j_{\ell+1}} || \mathbf{c}_{\ell+2,j_{\ell+1}} || \mathbf{c}_{\ell+2,j_{\ell+1}} || 0 || \ldots ) \\
\mathbf{b}_{\ell+1,j_{\ell+1}}^{[\ell+1]} &= (\ldots || \mathbf{u}_{\ell+1,j_{\ell+1}} || \mathbf{c}_{\ell+1,j_{\ell+1}} || \mathbf{u}_{\ell+2,j_{\ell+1}} || \mathbf{c}_{\ell+2,j_{\ell+1}} || \mathbf{c}_{\ell+3,j_{\ell+1}} || \ldots )
\end{align*}
\]

and therefore
\[
\sum_{i=1}^{\ell+1} \mathbf{b}_{i,j_{i}}^{[\ell]} = (\ldots || \mathbf{K}_{\ell} || \mathbf{K}_{\ell+1} || \mathbf{K}_{\ell+2} || \mathbf{K}_{\ell+3} || \ldots )
\]

Assume we use the factorization scheme as given by Algorithm 5.10. As the bits of \( \mathbf{u}_{i,j_{i}}^{[m]} \) are randomly chosen, only the bits of \( \mathbf{c}_{i,j_{i}}^{[m]} \) can be used for factoring with respect to \( \beta \). Therefore, to factorize \( \sum_{i=1}^{\ell+1} \mathbf{b}_{i,j_{i}}^{[\ell]} \), i.e. to recover the index \( j_{\ell+1}^{[\ell]} \) for block \( \mathbf{B}_{\ell+1} \), we may only use bits of \( \mathbf{c}_{\ell+3,j_{\ell+3}}^{[\ell+3]} \) i.e. the bits on positions \( \mathbf{K}_{\ell+3} \). However, as \( \mathbf{B}_{\ell+1} \) has length \( r_{\ell+1} = 2^{k_{\ell+1}+k_{\ell+3}} \), there are \( 2^{k_{\ell+1}} \) elements of \( \mathbf{B}_{\ell+1} \) having the same value \( \mathbf{c}_{\ell+3,j_{\ell+3}}^{[\ell+3]} \) on positions \( \mathbf{K}_{\ell+3} \). In other words, only \( k_{\ell+3} \) bits from index \( j_{\ell+1}^{[\ell]} \) can be determined.

Having obtained this information, we now return to the Matrix-permutation attack when a fused signature \( \beta \) is used in an \( \text{FT-MST}_3 \). Similar to Proposition 6.3 we may show that there exists a matrix \( \mathbf{L}_{\ell+1} \) such that
\[
(\tilde{\mathbf{x}}(x)_{a} + A_{\ell+1} \parallel \tilde{\mathbf{x}}(x)_{b} + \mathbf{B}_{\ell+1} \parallel 1 ) L_{\ell+1} = \mathbf{c}_{\ell+3,j_{\ell+3}}^{[\ell+3]}
\]

Using such a matrix we can recover only \( k_{\ell+3} \) bits from \( (k_{\ell+1} + k_{\ell+3}) \) bits of \( j_{\ell+1}^{[\ell]} \) for \( \mathbf{B}_{\ell+1} \).

This shows that the Matrix-permutation attack applied to \( \text{FT-MST}_3 \) can recover only a portion of bits of the index in each fused block of \( \beta \). Thus we have the following proposition.

**Proposition 6.10** Let \( \mathbf{B}_{\ell} \) be a block of a fused transversal logarithmic signature \( \beta \) used in \( \text{FT-MST}_3 \). Let \( \mathbf{B}_{\ell} = ((\mathbf{D}_{i_{1}}, \mathbf{D}_{i_{2}}, \ldots \mathbf{D}_{i_{u_i}}) \ldots \mathbf{D}_{i_{u_{s}}}) \) as defined Algorithm 5.4, where \( i_{1} < i_{2} < \cdots < i_{u_i} \). Let \( k_{i} = \lfloor \log_2 D_{i} \rfloor \). By using the Matrix-permutation attack on \( \text{FT-MST}_3 \) one can determine \( k_{u_{i}} \) from \( \sum_{i=1}^{u_{i}} k_{i} \) bits for the index in block \( \mathbf{B}_{\ell} \).

The complexity of factoring a ciphertext by using the Matrix-permutation attack on \( \text{FT-MST}_3 \) is thus given as the product of the complexities for factoring with respect to each block \( \mathbf{B}_{\ell} \), \( \ell = 1, \ldots, s \). Moreover, as the factorization has to be proceeded implicitly according to the permutation \( \xi \) 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 \( \text{FT-MST}_3 \) in the following proposition.
Proposition 6.11 Let \( \mathfrak{m} \) be an input length of an FT-MST\(_3\) scheme with a fused transversal logarithmic signature \( \beta \) created by Algorithm 5.4. Let \( \mathcal{P} = \{P_1, \ldots, P_s\} \) be a partition used in step (3) of this algorithm where \( P_\ell = \{\ell_1, \ldots, \ell_{u_\ell}\} \) for \( \ell = 1, \ldots, s \). Let \( k_\ell = \lfloor \log_2 D_\ell \rfloor \), where \( D_\ell \) 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 \( O(2^c) \) where

\[
c = \left( \mathfrak{m} - \sum_{\ell=2}^s k_{\ell u_\ell} - \sum_{j=1}^{u_1} k_{1 j} \right)
\]

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 \( \alpha \) and \( \gamma \) 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 \( \mathfrak{m}_i \) in \( \mathcal{Z} \). Note that the adversary actually does not know \( B_i \) and therefore cannot verify whether \( B_i \) 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 \( \mathcal{P} \) in step (3) of Algorithm 5.4 in a way that such a block \( B_i \) necessarily has a large dimension \( \mathfrak{m}_i \). 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 described in Section 5 will be used for generating logarithmic signatures \( \beta \). As observed, if the information at each step of Algorithm 5.4 is kept track, in particular, the knowledge of partition \( \mathcal{P} = \{P_1, \ldots, P_s\} \) used in step (3), we have a highly efficient factorization method with respect to \( \beta \) as shown in Algorithm 5.8. By taking the discussion of Subsection 6.1 into account we may, if necessary, select the elements of \( \alpha, \gamma \) 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 \( m \geq 3 \) is not a power of 2 and \( \theta \) be a non trivial odd-order automorphism of \( \mathbb{F}_q \). Let \( \mathcal{G} = A(\mathfrak{m}, \theta) \) 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:

\[
S(a_1, b_1)S(a_2, b_2) = S(a_1 + a_2, b_1 + b_2 + a_1 a_2^\theta).
\]  

(7.19)

We could store the group elements \( S(a, b) \) as pairs \( (a, b) \), but this would require that we compute some \( a^\theta \) each time we compute a product of group elements. In turn, each computation \( a^\theta \) requires at most \( 2\lfloor \log_2 \mathfrak{m} \rfloor \) multiplications in \( \mathbb{F}_q \). It is therefore more time efficient to store the group elements as triples \( (a, b, a^\theta) \). 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 $\alpha = ( ( a_i), a_i )$ and $\gamma = ( ( h_i), h_i )$. For a given $i$ we have $h_i = a_i + t_i$ for all $j = 1, \ldots, r_i$. This means that for each $i$, if $a_i$'s and the sum $t_i$ are known, $h_i$'s can be derived. Therefore, for the public key we need to store $[\alpha, \gamma]$ (i.e. the "b.part" of $\gamma$) and the $s$ values $t_i$, for $i = 1, \ldots, s$.

However, for a practical implementation of MST3 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 $A$, which generates random cover $\alpha$ satisfying conditions in Section 3. Essentially, the Algorithm $A$ utilizes a pseudo-random number generator $R$. To simplify the description we assume that a logarithmic signature $\beta$ has been generated by Algorithm 5.4 separately.

Algorithm 7.1 (Reduced key storage)

**External:** Algorithm $A$, a pseudo-random number generator $R$

**Input:** [\beta, f, t_0, \ldots, t_s], a seed $S$ for $R$

**Output:** $[\alpha, \gamma]$

(a) Using $A$, $R$ and $S$ to generate $\alpha = ( ( a_i), a_i )$

(b) Create $\gamma = ( ( h_i), h_i )$ from $\alpha$ as described in Section 3.

From Algorithm 7.1 it is clear that for the public key one has to publish $(h_i,b_i)$ together with $t_i = a_i + t_i$ for $i = 1, \ldots, s$, and $S$ only. To obtain the complete public key, i.e. $[\alpha, \gamma]$, one first generates $\alpha$ from step (a) using $R$ and seed $S$, then computes $(h_i,a_i)$ from $a_i$ and $t_i = a_i + t_i$. This approach will reduce the key size of the system roughly to only one third of $[\alpha, \gamma]$ (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 MST3.

The cipher expansion of MST3 is of a factor three. For, suppose $(y_1,y_2)$ is a ciphertext pair with $y_1 = (y_1,b_1)$ and $y_2 = (y_2,b_2)$, then, it suffices to send $y_1$ and $y_2$ as the ciphertext. This is because $y_2$ can be obtained from the equation $y_2 = y_1 + t_i$ by using the private key $t_0$ and $t_s$.

7.3 Examples for generation of $\beta$

We need to introduce some notation. We say a logarithmic signature (cover) $\beta$ is of type $(v_1^{u_1},v_2^{u_2},\ldots,v_{u_1})$ if $\beta$ has the first $u_1$ blocks of size $v_1$, the next $u_2$ blocks of size $v_2$, 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 $\varepsilon' = \varepsilon_0 < \varepsilon_1 < \cdots < \varepsilon_s = \varepsilon_s$ such that each block $E_i$ consists of a complete set of coset representatives of $\varepsilon_{i-1}$ in $\varepsilon_i$.

We also write $[r_1, r_2, \cdots, r_{u_1}]$ to denote the fusion of $\ell$ blocks $E_1, E_2, \ldots, E_\ell$, where $|E_1| = r_1$ and $u_1 < u_2 < \cdots < u_\ell$. Specially, $[r_1]$ denotes a non-fused block $E_1$ with $|E_1| = r_1$. We say a block $B_1$ of $\beta$ is of fusion type $[r_1, r_2, \cdots, r_{u_1}]$ if
Let \( \beta \) be a fused logarithmic signature of fusion type, say, \([v_1]^{e_1} \ldots [v_{l-1}]^{e_{l-1}} [v_l \times u_i]^{e_1} \ldots [v_{j-1} \times u_{j-1}]^{e_{i-1}} [v_j \times u_j \times w_j]^{e_j} \ldots [v_\ell \times u_\ell \times w_\ell]^{e_\ell} \), generated by Algorithm 5.4. If \( \beta \) is used for a set-up of FT-MST, the workload of the Matrix-permutation attack required to recover the plaintext is roughly bounded by \( \mathcal{O}(v_1^{e_1} \ldots v_{l-1}^{e_{l-1}} (v_j, u_j)^{e_j} \ldots (v_\ell, u_\ell)^{e_\ell}) \) (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 \( \beta \) for \( \mathbb{Z} \) starting from \( \varepsilon \) of type \((128^2.32^{30}.4^{30})\), i.e. \( \nu = 62 \). 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 \( \beta \) is of type \((128^{32})\), and of fusion type \([128^2.32^4]^30\).

2. Using \( \beta \) 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 \( \beta \) for \( \mathbb{Z} \) starting from \( \varepsilon \) of type \((256.32^{25}.8^{24}.4^{25})\), i.e. \( \nu = 75 \). 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 \( \beta \) is of type \((256.128.1024^{24})\), and of fusion type \([256].[32 \times 4].[32 \times 8 \times 4]^{24}\).

2. Using \( \beta \) 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 \( \theta \) (EXP(\( \theta \))), the generation of \( m \)-bit random R (PRNG), and the factorization of \( \tilde{\varepsilon}(R) \in \mathbb{Z} \) with respect to a transversal logarithmic signature \( \varepsilon \) using the Algorithm 5.9 (FACTOR).
Table 1: Number of basic operations for one encryption/decryption of FT-MST₃.

<table>
<thead>
<tr>
<th></th>
<th>$F_{2^m}$ ADD</th>
<th>$F_{2^m}$ MULT</th>
<th>$F_{2^m}$ EXP($θ$)</th>
<th>$F_{2^m}$ PRNG</th>
<th>FACTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>encryption</td>
<td>7s − 7</td>
<td>2s − 2</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>decryption</td>
<td>$m + 4s + 8$</td>
<td>$s + 3$</td>
<td>2</td>
<td>-</td>
<td>1</td>
</tr>
</tbody>
</table>

We note that an intrinsic property of MST₃ is that there is a trade-off between the key storage and the speed of the scheme. For example, if $F_{2^{20}}$ is the underlying field for the Suzuki 2-group $G$, then the corresponding FT-MST₃ has an input of 320 bit length; if $α$ and $γ$ have type $(4.6453)$, 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 NTL¹ and measured on a 64-bit machine of 1.8 GHz.

8 Conclusions

We have presented a revised version of the MST₃ 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 MST₃. 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.

¹NTL C++ Library, written by Victor Shoup, http://www.shoup.net/ntl
Table 2: Various data for parameters, performance and security of FT-MST₃.

<table>
<thead>
<tr>
<th>m</th>
<th>s</th>
<th>type of β</th>
<th>pₓk [kB]</th>
<th>fusion type of β</th>
<th>W [kB/s]</th>
<th>E [kB/s]</th>
<th>D [kB/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>160</td>
<td>26</td>
<td>(256² - 64)²</td>
<td>43</td>
<td>[256], [16 x 4 x 4], [16 x 4]¹⁴</td>
<td>2¹³⁴⁵</td>
<td>607</td>
<td>859</td>
</tr>
<tr>
<td>160</td>
<td>23</td>
<td>(64 - 128)²²</td>
<td>57</td>
<td>[64], [128], [32 x 4]²¹</td>
<td>2¹³⁵</td>
<td>604</td>
<td>852</td>
</tr>
<tr>
<td>160</td>
<td>20</td>
<td>(256²⁰)²⁰</td>
<td>100</td>
<td>[256], [16 x 4 x 4]¹⁹</td>
<td>2¹³⁴</td>
<td>671</td>
<td>895</td>
</tr>
<tr>
<td>160</td>
<td>18</td>
<td>(256² - 512)¹⁶</td>
<td>170</td>
<td>[256], [16 x 4 x 4], [32 x 4 x 4]¹⁶</td>
<td>2¹³⁸</td>
<td>689</td>
<td>904</td>
</tr>
<tr>
<td>160</td>
<td>16</td>
<td>(1024⁰)⁰</td>
<td>320</td>
<td>[1024], [32 x 8 x 4]¹⁵</td>
<td>2¹³⁹</td>
<td>758</td>
<td>941</td>
</tr>
<tr>
<td>192</td>
<td>32</td>
<td>(64)⁴</td>
<td>49</td>
<td>[64], [16 x 4]²⁰</td>
<td>2¹³⁴</td>
<td>571</td>
<td>854</td>
</tr>
<tr>
<td>192</td>
<td>28</td>
<td>(8 - 128)²⁴</td>
<td>82</td>
<td>[8], [128], [32 x 4]²⁵</td>
<td>2¹³⁵</td>
<td>529</td>
<td>783</td>
</tr>
<tr>
<td>192</td>
<td>24</td>
<td>(256)²⁴</td>
<td>145</td>
<td>[256], [16 x 4 x 4]²³</td>
<td>2¹³⁴</td>
<td>609</td>
<td>851</td>
</tr>
<tr>
<td>192</td>
<td>22</td>
<td>(8 - 512)²¹</td>
<td>253</td>
<td>[8], [512], [32 x 4 x 4]²⁰</td>
<td>2¹³⁴</td>
<td>679</td>
<td>914</td>
</tr>
<tr>
<td>192</td>
<td>20</td>
<td>(4 - 1024)¹⁹</td>
<td>457</td>
<td>[4], [1024], [32 x 8 x 4]¹⁸</td>
<td>2¹³⁴</td>
<td>720</td>
<td>924</td>
</tr>
<tr>
<td>224</td>
<td>38</td>
<td>(4 - 64)⁴⁴</td>
<td>66</td>
<td>[4], [64], [16 x 4]³⁴²</td>
<td>2¹³⁵</td>
<td>511</td>
<td>772</td>
</tr>
<tr>
<td>224</td>
<td>32</td>
<td>(128)³²</td>
<td>113</td>
<td>[128], [32 x 4]³⁰</td>
<td>2¹³⁵</td>
<td>565</td>
<td>827</td>
</tr>
<tr>
<td>224</td>
<td>28</td>
<td>(256²⁸)²⁸</td>
<td>197</td>
<td>[256], [16 x 4 x 4]²⁷</td>
<td>2¹³⁵</td>
<td>595</td>
<td>845</td>
</tr>
<tr>
<td>224</td>
<td>25</td>
<td>(256 - 512²⁴</td>
<td>344</td>
<td>[256], [32 x 4 x 4]²⁴</td>
<td>2¹³⁵</td>
<td>637</td>
<td>875</td>
</tr>
<tr>
<td>224</td>
<td>23</td>
<td>(256 - 64·1024²¹</td>
<td>507</td>
<td>[256], [16 x 4], [32 x 8 x 4]²¹²</td>
<td>2¹³⁴</td>
<td>678</td>
<td>989</td>
</tr>
<tr>
<td>255</td>
<td>43</td>
<td>(8 - 64)⁴⁴</td>
<td>85</td>
<td>[8], [64], [16 x 4]³⁸²</td>
<td>2¹³⁵</td>
<td>532</td>
<td>808</td>
</tr>
<tr>
<td>255</td>
<td>37</td>
<td>(8 - 128)³⁰</td>
<td>145</td>
<td>[8], [128], [32 x 4]³⁴</td>
<td>2¹³⁵</td>
<td>576</td>
<td>852</td>
</tr>
<tr>
<td>255</td>
<td>32</td>
<td>(256³¹ - 128)²⁵</td>
<td>252</td>
<td>[256], [16 x 4 x 4], [32 x 4]³⁰</td>
<td>2¹³⁵</td>
<td>602</td>
<td>865</td>
</tr>
<tr>
<td>255</td>
<td>29</td>
<td>(8 - 512)²⁸</td>
<td>447</td>
<td>[8], [512], [32 x 4 x 4]²⁷</td>
<td>2¹³⁴</td>
<td>637</td>
<td>875</td>
</tr>
<tr>
<td>255</td>
<td>26</td>
<td>(256 - 128·1024²⁴</td>
<td>778</td>
<td>[256], [32 x 4], [32 x 8 x 4]²⁴²</td>
<td>2¹³⁴</td>
<td>678</td>
<td>989</td>
</tr>
<tr>
<td>288</td>
<td>48</td>
<td>(64)⁸⁸</td>
<td>110</td>
<td>[64], [16 x 4]³⁴²</td>
<td>2¹³⁴</td>
<td>306</td>
<td>502</td>
</tr>
<tr>
<td>288</td>
<td>41</td>
<td>(256 - 128)⁰</td>
<td>190</td>
<td>[256], [128], [32 x 4]³⁵³</td>
<td>2¹³⁴</td>
<td>325</td>
<td>523</td>
</tr>
<tr>
<td>288</td>
<td>36</td>
<td>(256³⁶)³⁶</td>
<td>325</td>
<td>[256], [16 x 4 x 4]³⁴</td>
<td>2¹³⁴</td>
<td>381</td>
<td>593</td>
</tr>
<tr>
<td>288</td>
<td>32</td>
<td>(512³²)³²</td>
<td>577</td>
<td>[512], [32 x 4 x 4]³¹</td>
<td>2¹³⁴</td>
<td>407</td>
<td>595</td>
</tr>
<tr>
<td>288</td>
<td>29</td>
<td>(512²·1024²⁷</td>
<td>1009</td>
<td>[512], [32 x 4 x 4], [32 x 8 x 4]²⁷</td>
<td>2¹³⁴</td>
<td>457</td>
<td>668</td>
</tr>
<tr>
<td>320</td>
<td>54</td>
<td>(4 - 64)⁴⁴</td>
<td>135</td>
<td>[4], [64], [16 x 4]³⁸²</td>
<td>2¹³⁴</td>
<td>287</td>
<td>471</td>
</tr>
<tr>
<td>320</td>
<td>46</td>
<td>(8 - 512 - 128)⁴⁴</td>
<td>242</td>
<td>[8], [512], [32 x 4]³⁴</td>
<td>2¹³⁴</td>
<td>305</td>
<td>490</td>
</tr>
<tr>
<td>320</td>
<td>40</td>
<td>(256⁰)⁰</td>
<td>402</td>
<td>[256], [16 x 4 x 4]³⁸</td>
<td>2¹³⁴</td>
<td>377</td>
<td>581</td>
</tr>
<tr>
<td>320</td>
<td>36</td>
<td>(32 - 512³⁵</td>
<td>703</td>
<td>[32], [512], [32 x 4 x 4]³⁴</td>
<td>2¹³⁴</td>
<td>403</td>
<td>604</td>
</tr>
<tr>
<td>320</td>
<td>32</td>
<td>(1024³²)³²</td>
<td>1281</td>
<td>[1024], [32 x 8 x 4]³¹</td>
<td>2¹³⁴</td>
<td>450</td>
<td>650</td>
</tr>
<tr>
<td>352</td>
<td>59</td>
<td>(16 - 64)⁸⁸</td>
<td>163</td>
<td>[16], [64], [16 x 4]³⁸²</td>
<td>2¹³⁴</td>
<td>246</td>
<td>408</td>
</tr>
<tr>
<td>352</td>
<td>51</td>
<td>(4 - 128)⁰</td>
<td>277</td>
<td>[4], [128], [32 x 4]³⁴</td>
<td>2¹³⁴</td>
<td>292</td>
<td>475</td>
</tr>
<tr>
<td>352</td>
<td>44</td>
<td>(256⁴⁴)⁴⁴</td>
<td>486</td>
<td>[256], [16 x 4 x 4]³⁴²</td>
<td>2¹³⁴</td>
<td>304</td>
<td>481</td>
</tr>
<tr>
<td>352</td>
<td>40</td>
<td>(2 - 512³⁰</td>
<td>800</td>
<td>[2], [512], [32 x 4 x 4]³⁶</td>
<td>2¹³⁴</td>
<td>352</td>
<td>537</td>
</tr>
<tr>
<td>352</td>
<td>36</td>
<td>(8 - 512 - 1024²⁴</td>
<td>1431</td>
<td>[8], [512], [32 x 8 x 4]³⁴</td>
<td>2¹³⁴</td>
<td>378</td>
<td>566</td>
</tr>
<tr>
<td>384</td>
<td>64</td>
<td>(64⁴⁴)⁴⁴</td>
<td>195</td>
<td>[64], [16 x 4]³⁸²</td>
<td>2¹³⁴</td>
<td>252</td>
<td>421</td>
</tr>
<tr>
<td>384</td>
<td>55</td>
<td>(64 - 128)⁴⁴</td>
<td>330</td>
<td>[64], [128], [32 x 4]³⁵</td>
<td>2¹³⁴</td>
<td>287</td>
<td>466</td>
</tr>
<tr>
<td>384</td>
<td>48</td>
<td>(256⁴⁴)⁴⁴</td>
<td>578</td>
<td>[256], [16 x 4 x 4]³⁴</td>
<td>2¹³⁴</td>
<td>303</td>
<td>485</td>
</tr>
<tr>
<td>384</td>
<td>43</td>
<td>(64 - 512³²</td>
<td>1013</td>
<td>[64], [512], [32 x 4 x 4]³³</td>
<td>2¹³⁴</td>
<td>352</td>
<td>535</td>
</tr>
<tr>
<td>384</td>
<td>39</td>
<td>(16 - 1024³⁸</td>
<td>1827</td>
<td>[16], [1024], [32 x 8 x 4]³⁷</td>
<td>2¹³⁴</td>
<td>364</td>
<td>554</td>
</tr>
</tbody>
</table>
References


