Logarithmic signatures for abelian groups and their factorization

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

Abstract

Factorizable logarithmic signatures for finite groups are the essential component of the cryptosystems MST_1 and MST_3. The problem of finding efficient algorithms for factoring group elements with respect to a given class of logarithmic signatures is therefore of vital importance in the investigation of these cryptosystems. In this paper we are concerned about the factorization algorithms with respect to transversal and fused transversal logarithmic signatures for finite abelian groups. More precisely we present algorithms and their complexity for factoring group elements with respect to these classes of logarithmic signatures. In particular, we show a factoring algorithm with respect to the class of fused transversal logarithmic signatures and also its complexity based on an idea of Blackburn, Cid and Mullan for finite abelian groups.

1 Introduction

Logarithmic signatures and covers for finite groups have found interesting applications in designing cryptographic primitives and pseudo-random number generators [6], [8], [9], [4], [13], [5], [7], [11]. Logarithmic signatures and covers are a kind of factorization of a finite group G through its subsets and they induce surjective mappings from $\mathbb{Z}_{|\mathsf{G}|}$ onto G. An interesting fact is that these mappings can, in general, very efficiently be computed. However, if we take a random cover for a finite group, its induced mapping behaves like a random function, see [11], thus inverting this mapping becomes an intractable problem. There are strong indications supporting this fact. On the other hand, the mapping induced by a logarithmic signature actually is a bijection. As there are various classes of logarithmic signatures which are arised from algebraic structures of the groups, the problem of inverting this bijection needs a careful study. More important is the fact that the logarithmic signatures, whose induced mappings are used as part of private key in a public key cryptosystem, see [9], [4], [13], have to be efficiently invertible. Hence, the question of inverting the induced bijection for a given logarithmic signature is of significance. The problem has not been treated in detail in the literature yet. In [8] Magliveras and Memon have shown that the induced bijections for a specific class of transversal logarithmic signatures derived from a chain of point stabilizer subgroups for permutation groups of degree $\mathfrak{n}$ can be invertible with a complexity of

$O(n^2)$. In [10] it is shown that the induced bijection of a certain specific class of transversal logarithmic signatures for elementary abelian 2-groups can be invertible with a complexity of $O(1)$, see also [15]. In [2] Blackburn, Cid and Mullan state that the bijections induced from the class of fused transversal logarithmic signatures for elementary abelian 2-groups G are *tame*. Unfortunately, the statement is erroneous. The gap in their proof is caused by a lack of an algorithm and therefore an estimate of its complexity. In [13] the problem of inverting induced bijections of fused transversal logarithmic signatures is discussed.

In this paper we study the inverting problem of the bijections induced from transversal and fused transversal logarithmic signatures for abelian groups. We present algorithms and their complexity for solving their inverting problem. In particular, we show an algorithm based on the idea of Blackburn et al. and determine its complexity. We further study the inverting problem by using trapdoor information and show that fused transversal logarithmic signatures for abelian groups are tame with respect to this method.

2 Preliminaries

In this section we briefly present notation, definitions and some basic facts about logarithmic signatures and covers for finite groups and their induced mappings. For more details the reader is referred to [8], [9]. We assume that the reader is familiar with the basis of group theory. The group theoretic notation used is standard and may be found in any textbook of group theory. In this paper we only deal with finite groups.

Let G be a finite group. We define the *width* of G to be the positive integer $w = \lceil \log_2 |G| \rceil$. Suppose that $\alpha = [A_1, A_2, \ldots, A_s]$ is a sequence of subsets $A_i \subset G$, such that $\sum_{i=1}^{s} |A_i|$ is polynomially bounded in the width w of G. Let S be a subset of G. We say that α is a *cover* for S if each element $h \in S$ can be expressed in at least one way as a product of the form

$$h = g_1 \cdot g_2 \cdots g_{s-1} \cdot g_s \tag{2.1}$$

with $g_i \in A_i$.

If every $h \in S$ can be expressed in exactly one way by Equation (2.1), then α is called a *logarithmic signature* for S. If $S = G$, α is called a cover resp. a logarithmic signature for G.

The A_i are called the *blocks*, and the vector $(r_1, \ldots, r_s)$ with $r_i = |A_i|$ the *type* of α. We say that α is *nontrivial* if $s \geqslant 2$ and $r_i \geqslant 2$ for $1 \leqslant i \leqslant s$; otherwise α is said to be *trivial*.

Let $\gamma : G = G_0 > G_1 > \cdots > G_s = 1$ be a chain of subgroups of G, and let A_i be an ordered, complete set of right (or left) coset representatives of G_{i-1} in G_i. Then it is clear that $[A_1, \ldots, A_s]$ forms a logarithmic signature for G, called a *transversal logarithmic signature* (TLS).

Let $\mathcal{F}$ be a family of groups. The family is assumed to have infinitely many members. Suppose we are given cover α_G for each $G \in \mathcal{F}$. Let $\mathcal{C} = \{\alpha_G \mid G \in \mathcal{F}\}$. If there is a constant k for all $G \in \mathcal{F}$ such that the factorization in Equation (2.1) with respect to each $\alpha_G \in \mathcal{C}$ can be achieved in $O(w^k)$, where w is the width of G, then α_G is called *tame* (i.e. there is an algorithm for factoring with respect to α_G which can be achieved in polynomial time in w). Such an algorithm will also be defined as tame. To state it in a different manner, a class of covers for a family of groups is considered to be tame, if there exists a generic factorization algorithm which is tame for all members of the family. For example, suppose G be a permutation group on the set $X = \{1, \ldots, n\}$. Consider a chain of nested point stabilizers $G = G_0 > G_1 > \cdots > G_s = 1$, where G_i fixes pointwise the symbols $1, 2, \ldots, i$, for any $i \geqslant 1$. It is shown in [8] that a specific constructed class of transversal logarithmic

signatures from this chain of subgroups has a factorization with complexity $O(n^2)$. Hence this class of TLS is tame if n is bounded by a polynomial in the width w of G.

The existence of a constant k in the above definition of the tameness logarithmic signatures (for all members in the family $\mathcal{F}$) is crucial and meaningful. The reason is that if we consider a single group G, then the complexity of the factorization with respect to any logarithmic signature for G can always be expressed as a polynomial in w, because G is a finite group (of course, we implicitly assume that the type of cover is polynomial bounded in w as well). Thus by talking about the tameness of a logarithmic signature for a given group it should be understood in accordance with the above definition.

On the other hand we may encounter classes of factorization algorithms whose computational complexity does not provide the tameness property for the corresponding logarithmic signatures, possessing however a certain degree of efficiency. More precisely, suppose we are given a generic factorization algorithm $A_{\mathcal{F}}$ with respect to a class of LS for all members of $\mathcal{F}$, that is not tame by definition, i.e. there is no constant k, such that the complexity of $A_{\mathcal{F}}$ is $O(w^k)$, for an LS α_G for $G \in \mathcal{F}$, where $w = \lceil \log_2 |G| \rceil$. However, we know that for G the factorization complexity of F can always be expressed by $O(w^\ell)$ for some integer ℓ. Now if $O(w^\ell) << |G|$, then we may say that α_G is *factorizable*. This is to say that the complexity $O(w^\ell)$ is "small" compared with $|G|$. This definition implies that even it is not possible to store all the elements of G, it is still computationally feasible to factor any given element $g \in G$ with respect to α_G, which has a polynomial bounded storage in w.

In general, the problem of finding a factorization in Equation (2.1) with respect to a given cover is presumedly intractable. There are strong evidences in support of the hardness of the problem. For example, let G be a cyclic group and g be a generator of G. Let $\alpha = [A_1, A_2, \ldots, A_s]$ be any cover for G, for which the elements of A_i are written as powers of g. Then the factorization with respect to α amounts to solving the Discrete Logarithm Problem (DLP) in G.

The main point making covers and LS interesting for use in cryptography is that if the above factorization problem is intractable, they 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 G with $A_i = [a_{i,1}, a_{i,2}, \ldots, a_{i,r_i}]$ and let $m = \prod_{i=1}^{s} r_i$. Let $m_1 = 1$ and $m_i = \prod_{j=1}^{i-1} r_j$ for $i = 2, \ldots, s$. Let τ 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.

$$\tau : \quad \mathbb{Z}_{r_1} \oplus \mathbb{Z}_{r_2} \oplus \cdots \oplus \mathbb{Z}_{r_s} \to \mathbb{Z}_m$$

$$\tau(j_1, j_2, \ldots, j_s) := \sum_{i=1}^{s} j_i m_i.$$

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

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

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

Conversely, given a cover α and an element $y \in G$, to determine any element $x \in \breve{\alpha}^{-1}(y)$ it is necessary to obtain any one of the possible factorizations of type (2.1) for y and determine indices $j_1, j_2, \ldots, j_s$ such that $y = a_{1,j_1} \cdot a_{2,j_2} \cdots a_{s,j_s}$. This is possible if and only if α is factorizable. Once a vector $(j_1, j_2, \ldots, j_s)$ has been determined, $\breve{\alpha}^{-1}(y) = \tau(j_1, j_2, \ldots, j_s)$ can be computed efficiently.

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

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

A block A_i of a cover is called *normalized* if A_i contains the identity element of the group, i.e. $id_G \in A_i$. It is obvious that by using a sandwich transformation with $g_i \in A_i$ for $i = 1, \ldots, s-1$ we can transform α to an equivalent β having all $(s-1)$ blocks normalized, the last block B_s is in general not normalized.

Let $\alpha = [A_1, A_2, \ldots, A_s]$ be a LS for a finite group G. Consider $k \geqslant 2$ blocks A_{i_1} and A_{i_k} of α. Define $B := A_{i_1}.A_{i_2}.\ldots.A_{i_k} = \{a_{i_1}.a_{i_2}.\ldots.a_{i_k} \mid a_{i_j} \in A_{i_j}, j = 1, \ldots, k\}$. We call B a fused block of $A_{i_1}, \ldots, A_{i_k}$. If we apply fusion operations to the blocks of α we generally obtain a cover $\beta = [B_1, \ldots, B_t]$ for a subset of G, where $t < s$. However, if G is abelian, then β remains a LS for G. Usually β may not necessarily be equivalent to α, and we call β a *fused logarithmic signature* of α.

3 Algorithms for factorization with respect to TLS

In this section we present two algorithms for factorization with respect to TLS for abelian groups. One of them shows that TLS are tame.

We first present a generic algorithm for factoring with respect to any TLS α for any group G (abelian or non-abelian).

Algorithm 1 Generic Algorithm

Input: $\mathcal{G}$: a finite group, $\alpha = [A_1, A_2, \ldots, A_s]$ a TLS for G constructed from a chain of subgroups $G = G_0 > G_1 > \cdots > G_s = 1$ of G, $g \in G$.
Output: $a_i \in A_i$ such that $g = a_1 \ldots a_s$.

1: Find a unique element $a_1 \in A_1$ such that $g_1 = a_1.g^{-1} \in G_2$. Find a unique element $a_2 \in A_2$ such that $g_2 = a_2.g_1^{-1} \in G_3$. Continue this process until A_s. Then we have $g = a_1 \ldots a_s$ as a factorization of g with respect to α.

Note that this factorization algorithm requires a complexity of $O(\sum_{i=1}^{s} |A_i|)$ without counting the complexity for checking the membership of g_i in G_{i+1}. If G is a permutation group of degree n, there exist algorithms for solving the membership problem for G in polynomial time with respect to n by using a strong generating set, see [3]. If G is a matrix group over a finite field, the membership problem is still not solved for fields of even characteristic. For odd characteristic, see [1]. Assume that the membership problem would be solved for G with a polynomial time algorithm in $w = \lceil \log_2 |G| \rceil$. With the complexity $O(\sum_{i=1}^{s} |A_i|)$, the generic factorization algorithm can only be considered efficient, i.e. α is factorizable, if $O(\sum_{i=1}^{s} |A_i|) << |G|$.

If G is abelian, we can have a factoring algorithm for TLS with a complexity of $O(w)$, i.e. transversal logarithmic signatures are tame. To be more precise we assume that each operation involving group operations is counted as a unit, for example, multiplying two elements in G, computing in quotient groups of G.

Again let $\alpha = [A_1, A_2, \ldots, A_s]$ be a TLS for G constructed from a chain of subgroups $G = G_0 > G_1 > \cdots > G_s = 1$. Since G is abelian, each G_i is a normal subgroup of G. Therefore we can form the quotient group $\bar{G}^{(i)} := G/G_i$ for $i = 0, \ldots, s$, where $\bar{G}^{(i)} := G/G_i = \{G_i.g \mid g \in G\}$. The elements of $\bar{G}^{(i)}$ are denoted by $\bar{g}^{(i)}$, where $\bar{g}^{(i)} = \phi^{(i)}(g)$ and $\phi^{(i)} : G \longrightarrow \bar{G}^{(i)}$ defined by $\phi^{(i)}(g) = G_i.g$ is the canonical homomorphism.

For each $i = 1, \ldots, s$ define $\bar{\alpha}^{(i)} = [\bar{A}_1^{(i)}, \ldots, \bar{A}_i^{(i)}]$ with $\bar{A}_j^{(i)} = \phi^{(i)}(A_j)$. Note that the blocks $\bar{A}_{i+1}^{(i)}, \ldots, \bar{A}_s^{(i)}$ in the quotient group $\bar{G}^{(i)}$ are viewed as blocks of size 1 with the identity as their unique element. Therefore we ignore them all. For each $i = 1, \ldots, s$ define π_i to be the permutation in S_{r_i} which sorts the elements of A_i according to a certain order, for instance, numerical order. When applying π_i to A_i for all $i = 1, \ldots, s$ we obtain a TLS $\beta = [B_1, B_2, \ldots, B_s]$. The factorization with respect to α can obviously be done via β and π_i. Precisely, if $g = a_{1j_1} \ldots a_{rj_r}$ is a factorization of an element $g \in G$ with respect to β, then $g = a_{1\pi_1^{-1}(j_1)} \cdot \ldots a_{r\pi_r^{-1}(j_r)}$ is a factorization with respect to α, where π_i^{-1} is the inverse of π_i. We now present an algorithm for factoring with respect to a sorted TLS.

Algorithm 2 Factorization with TLS

Input: G: abelian group, $\alpha = [A_1, A_2, \ldots, A_s]$ a sorted TLS for G constructed from a chain of subgroups $G = G_0 > G_1 > \cdots > G_s = 1$ of G, $g \in G$.

Output: $a_i \in A_i$ such that $g = a_1 \ldots . a_s$.

1: Using the chain of quotient groups $\bar{G}^{(s-1)}, \ldots, \bar{G}^{(1)}$, the chain of TLS $\bar{\alpha}^{(s-1)}, \ldots, \bar{\alpha}^{(1)}$, and the chain of elements $\bar{g}^{(s-1)}, \ldots, \bar{g}^{(1)}$, we carry out the factorization of g as follows. First, find a unique element $\bar{a}_1^{(1)} \in \bar{\alpha}^{(1)} = [\bar{A}_1^{(1)}]$ such that $\bar{g}^{(1)} = \bar{a}_1^{(1)}$ (note that $\bar{A}_1^{(1)}$ is identical to the quotient group $G/G_1 := \bar{G}^{(1)}$).
In the quotient group $\bar{G}^{(2)}$ we have $\bar{\alpha}^{(2)} = [\bar{A}_1^{(2)}, \bar{A}_2^{(2)}]$ and the element $\bar{g}^{(2)}$ has a factorization $\bar{g}^{(2)} = \bar{a}_1^{(2)}.\bar{a}_2^{(2)}$ with respect to $\bar{\alpha}^{(2)}$, where $\bar{a}_1^{(2)}$ corresponds to $\bar{a}_1^{(1)}$ in $\bar{G}^{(1)}$, which is already known. So we can compute $\bar{a}_2^{(2)} = (\bar{a}_1^{(2)})^{-1}.\bar{g}^{(2)}$. From the known factorization of $\bar{g}^{(2)} = \bar{a}_1^{(2)}.\bar{a}_2^{(2)}$ with respect to $\bar{\alpha}^{(2)}$ we obtain a factorization of $\bar{g}^{(3)} = \bar{a}_1^{(3)}.\bar{a}_2^{(3)}.\bar{a}_3^{(3)}$ with respect to $\bar{\alpha}^{(3)}$, where $\bar{a}_3^{(3)} = (\bar{a}_2^{(3)})^{-1}.(\bar{a}_1^{(3)})^{-1}.\bar{g}^{(3)}$ and $\bar{a}_1^{(3)}$, $\bar{a}_2^{(3)}$ are elements in $\bar{G}^{(3)}$ having their images under the canonical homomorphism as $\bar{a}_1^{(2)}$ and $\bar{a}_2^{(2)}$ in $\bar{G}^{(2)}$ respectively. Continuing this process in $(s-1)$ steps we obtain a factorization of $\bar{g}^{(s-1)} = \bar{a}_1^{(s-1)} \ldots \bar{a}_{s-1}^{(s-1)}$ with respect to $\bar{\alpha}^{(s-1)}$ in the quotient group $\bar{G}^{(s-1)}$. Finally we obtain a factorization of $g = a_1 \ldots a_{s-1}.a_s$ with respect to α, where $a_s = a_{s-1}^{-1} \ldots a_1^{-1}.g$, and $a_1, \ldots, a_{s-1}$ are the elements in $A_1, \ldots, A_{s-1}$ (respectively) giving the corresponding elements $\bar{a}_1^{(s-1)}, \ldots, \bar{a}_{s-1}^{(s-1)}$ in $\bar{G}^{(s-1)}$.

The main complexity of the factorization in step i depends on the search of element $\bar{a}_i^{(i)}$ in $\bar{A}_i^{(i)}$. This can be done in time of $O(\log_2 |A_i|)$, since the elements of A_i are sorted. Hence $O(\sum_{i=1}^{s} \log_2 |A_i|) = O(w)$ is the complexity of Algorithm 2. The only extra operation for factoring with respect to an unsorted TLS is the application of the inverse permutations π_i^{-1} to the result obtained from a sorted TLS, as discussed above. Moreover, computing with each π_i can be carried out in unit time. Hence, we obtain the following theorem as a consequence of Algorithm 2.

Theorem 3.1 *Any transversal logarithmic signature for a finite abelian group is tame.*

Remark 3.2 Algorithm 2 can be applied to a TLS for a non-abelian group if each subgroup of the chain is normal in the underlying group. In particular, for a Hamiltonian group (a non-abelian group in which any subgroup is normal) any TLS is tame.

4 Algorithms for factorization with respect to FTLS

In this section we present algorithms for factoring group elements with respect to a fused transversal logarithmic signature (FTLS) for abelian groups. Let $\alpha = [A_1, A_2, \ldots, A_s]$ be a transversal logarithmic signature of type $(r_1, \ldots, r_s)$ for an abelian G. We define the following transformations on α.

(i) permute the blocks A_i's,

(ii) permute the elements within blocks A_i,

(iii) replace a block A_i with $A_i g$ for some $g \in G$, (as G is abelian, this replacement is in fact an application of a two side transformation on A_i, namely $h_{i-1}^{-1} A_i h_i = A_i g$, where $g = h_{i-1}^{-1}.h_i$),

(iv) replace two blocks A_i and A_j with a single block $A_i.A_j = \{xy \mid x \in A_i, y \in A_j\}$ (we call this operation the *fusion* of A_i and A_j).

A logarithmic signature obtained from a transversal logarithmic signature by applying a finite number of the transformations (i), (ii), (iii) and (iv) is called a *fused transversal* logarithmic signature (FTLS).

Definition 4.1 *A subset A of a finite abelian group G is called **periodic** if there exists an element $g \in G \setminus \{1\}$ with $gA = A$. We call such an element g the **period** of A.*

We refer the reader to [14] for details concerning periodicity properties for blocks of logarithmic signatures.

Lemma 4.2 *Let $\beta = [B_1, B_2, \ldots, B_t]$ a fused transversal logarithmic signature for an abelian group G. Then the following holds*

(i) At least one block B_i of β is periodic.

(ii) Let $x \in B_i$ the period of B_i and let $\bar{G} = G/ < x >$ be the quotient group of G modulo the cyclic group $< x >$. Then the logarithmic signature $\bar{\beta} = [\bar{B}_1, \bar{B}_2, \ldots, \bar{B}_t]$ induced from β is a FTLS for $\bar{G}$.

Proof. (i) Let $\alpha = [A_1, A_2, \ldots, A_s]$ be a transversal logarithmic signature for G, which is used to create β. Here we may assume that all the blocks of both α and β are normalized. Thus the block A_1, which is a normal subgroup of G, is contained in some block B_i of β. It is a simple observation that each element $x \in A_1 \setminus \{1\}$ is a period of B_i.

The second statement (ii) is obvious. □

Lemma 4.2 can be found in [2]. It is used by Blackburn, Cid and Mullan to prove that FTLS for elementary abelian 2-groups are tame. The authors have given a group argumentation for the proof without showing details. We now show an algorithm for the

factorization with respect to an FTLS for any abelian groups based on the Blackburn-Cid-Mullan idea and we determine its complexity.

Again let $\alpha = [A_1, A_2, \ldots, A_s]$ be a transversal logarithmic signature for an abelian G. Let $\beta = [B_1, B_2, \ldots, B_t]$ be a fused transversal logarithmic signature obtained by applying a finite number of the transformations (i), (ii), (iii) and (iv) to α. Let g be an element of G which we want to factorize by using β. Here we assume that all the blocks B_i's of β are normalized. The main idea of factoring with respect to an FTLS for elementary abelian 2-groups as described in [2] is as follows: Find a period x for a certain block of β and transform β to $\bar{\beta}$ in the quotient group $\bar{G} = G/ < x >$. Again $\bar{\beta}$ is an FTLS for $\bar{G}$ by Lemma 4.2, so the process is repeated with $\bar{\beta}$ and $\bar{G}$ until we reach the trivial quotient group, and the resulting FTLS becomes a trivial logarithmic signature. In this process we also keep track of the induced elements of g in the quotient groups.

Based on the idea of Blackburn, Cid and Mullan we show the following factoring algorithm with respect to an FTLS for abelian groups.

Algorithm 3 Factorization with FTLS

Input: G: abelian group, $\alpha = [A_1, A_2, \ldots, A_s]$ a normalized TLS for G constructed from a chain of subgroups $G = G_1 > G_2 > \cdots > G_{s+1} = 1$ of G, $\beta = [B_1, B_2, \ldots, B_t]$ a FTLS of type $(r_1, \ldots, r_t)$ obtained from α, $g \in G$.

Output: $b_i \in B_i$ such that $g = b_1 \cdots b_t$.

1: (a) Find a period x_1 for a periodic block B_i.

(b) Consider $\bar{\beta}^{(1)} = [\bar{B}_1^{(1)}, \bar{B}_2^{(1)}, \ldots, \bar{B}_t^{(1)}]$ induced by β in the quotient group $\bar{G}^{(1)} = G/ < x_1 >$. (Then $\bar{\beta}$ is an FTLS for $\bar{G}$ by Lemma 4.2. $\bar{\beta}$ is of type $(r_1, \ldots, r_{i-1}, r_i/\delta_1, r_{i+1}, \ldots, r_t)$, where δ_1 is the order of x_1.)

(c) Define $\bar{g}^{(1)}$ to be the induced element of g in the quotient group $\bar{G}^{(1)}$.

Repeat (a), (b) and (c) for $\bar{\beta}^{(1)}$, $\bar{G}^{(1)}$ and $\bar{g}^{(1)}$ to obtain $\bar{\beta}^{(2)}$, $\bar{G}^{(2)}$ and $\bar{g}^{(2)}$, where $\bar{G}^{(2)} = \bar{G}^{(1)}/ < \bar{x}_2 >$ and $\bar{x}_2$ is a period of some block $\bar{B}_j^{(1)}$. Continuing this process we eventually obtain a trivial LS $\bar{\beta}^{(u)}$ for the trivial group $\bar{G}^{(u)}$ after a finite number of steps, say u. Also, the induced element $\bar{g}^{(u)} \in \bar{G}^{(u)}$ becomes the identity element.

2: Working backward from $\bar{\beta}^{(u)}, \bar{\beta}^{(u-1)}, \ldots$ to $\bar{\beta}^{(1)}$ we can factorize g with respect to β as follows. Here, we describe one step of the factorization process. First note that $\bar{\beta}^{(i)}$ and $\bar{\beta}^{(i-1)}$ have all blocks of the same type except one block $\bar{\beta}^{(i-1)}$ containing the period $\bar{x}^{(i-1)}$ which is used to define $\bar{\beta}^{(i)}$ from $\bar{\beta}^{(i-1)}$. W.l.o.g. we may assume that this periodic block is the first block $\bar{B}_1^{(i-1)}$ of $\bar{\beta}^{(i-1)} = [\bar{B}_1^{(i-1)}, \bar{B}_2^{(i-1)}, \ldots, \bar{B}_t^{(i-1)}]$. Let $\bar{\beta}^{(i)} = [\bar{B}_1^{(i)}, \bar{B}_2^{(i)}, \ldots, \bar{B}_t^{(i)}]$. Assume by induction that $\bar{g}^{(i)} = \bar{b}_{1j_1}^{(i)} . \bar{b}_{2j_2}^{(i)} \ldots \bar{b}_{tj_t}^{(i)}$ is a known factorization of $\bar{g}^{(i)}$ with respect to $\bar{\beta}^{(i)}$ (i.e. $\bar{b}_j^{(i)} \in \bar{B}_j^{(i)}$, $j = 1, \ldots, t$). Now $\bar{g}^{(i-1)}$ is known as $\bar{g}^{(i)}$ is known by the induction assumption. Let $\bar{g}^{(i-1)} = \bar{b}_{1k_1}^{(i-1)} . \bar{b}_{2k_2}^{(i-1)} \ldots \bar{b}_{tk_t}^{(i-1)}$ be a factorization of $\bar{g}^{(i-1)}$ with respect to $\bar{\beta}^{(i-1)}$. Then we have $k_m = j_m$ for $m = 2, \ldots, t$. Hence the element $\bar{b}_{1k_1}^{(i-1)} \in \bar{B}_1^{(i-1)}$ is uniquely determined by

$$\bar{b}_{1k_1}^{(i-1)} = \bar{g}^{(i-1)} . (\bar{b}_{tj_t}^{(i-1)})^{-1} \ldots (\bar{b}_{2j_2}^{(i-1)})^{-1}.$$

In the following we attempt to determine the complexity of Algorithm 3 for elementary abelian p-groups. To simplify the formula involved we asumme further that $r_i = r$ for

$i = 1, \ldots, t$ and $|A_i| = z$ for $i = 1, \ldots, s$.

Let G be an elementary abelian p-group. Let $\alpha = [A_1, A_2, \ldots, A_s]$ be a TLS constructed from a chain of subgroups $G = G_1 > G_2 > \cdots > G_{s+1} = 1$ of G of type $(z, \ldots, z)$ (i.e. $|A_i| = z$ for all $i = 1, \ldots, t$). We also assume that $r_i = r$ for all $i = 1, \ldots, t$. So, we have $r_i = p^e$ for $i = 1, \ldots, t$.

One main part of the complexity of the algorithm is the finding of periodic elements in the process of constructing induced FTLS for the quotient group $\bar{G}^{(j)}$ for each $j = 1, \ldots, u$, where u is the smallest number such that the quotient group $\bar{G}^{(u)}$ becomes the identity group.

To start with we have to find a period in a certain block of β. There are t possible choices for such a block, say B_i. For an $x \in B_i$, verifying whether x is a period, i.e. $xB_i = B_i$, requires a complexity of $O(|B_i| \log_2 |B_i|)$. This complexity is composed of computing $|B_i|$ times multiplications $x.b_{i1}, \ldots, x.b_{ir}$ and of checking if $x.b_{ij} \in B_i$. The checking has a complexity $O(\log_2 |B_i|)$, if block B_i is sorted (otherwise it would be of complexity $O(|B_i|)$). Therefore, we will assume that each block B_i is sorted once. Sorting of B_i has a complexity of $O(|B_i| \log_2 |B_i|)$. For each step of moving to the quotient group the unique block of $\bar{\beta}^{(k)}$ whose size is decreased needs also to be sorted (more precisely, if x is a period in $\bar{B}_i^{(k-1)}$, the block $\bar{B}_i^{(k)}$ of $\bar{\beta}^{(k)}$ in the quotient group $\bar{G}^{(k)} = \bar{G}^{(k-1)}/ < x >$ is of size $|\bar{B}_i^{(k)}| = |\bar{B}_i^{(k-1)}|/p$ and we have to sort $\bar{B}_i^{(k)}$).

As the computation of pointer elements b_i's in the factorization of g in step 2 is deterministic, we may regard the time spent for this step as being constant and therefore its complexity will be neglected.

The total number of operations in step 1 comprises the number of operations for finding periods, denoted by A, and the number of operations for block sorting, denoted by B. Here we have

$$\begin{aligned} A &= t\big((r/p^0)^2 \log_2(r/p^0) + (r/p)^2 \log_2(r/p) + \cdots + (r/p^{e-1})^2 \log_2(r/p^{e-1})\big) \\ &= t \sum_{i=0}^{e-1} (r/p^i)^2 \log_2(r/p^i) \\ &= \frac{t}{\log_p 2} \sum_{i=1}^{e} (p^i)^2 \log_p(p^i) \\ &= \frac{t}{\log_p 2} \sum_{i=1}^{e} (p^2)^i i, \end{aligned}$$

and

$$\begin{aligned} B &= t\big((r/p^0) \log_2(r/p^0) + (r/p) \log_2(r/p) + \cdots + (r/p^{e-1}) \log_2(r/p^{e-1})\big) \\ &= \frac{t}{\log_p 2} \sum_{i=1}^{e} p^i i. \end{aligned}$$

By using the formula

$$\sum_{i=1}^{n} i x^i = \frac{n x^{n+2} - (n+1) x^{n+1} + x}{(x-1)^2},$$

where $x \neq 1$, the total number of operations in step 1 amounts to

$$A + B = \frac{t}{\log_p 2}\left(\frac{e(p^2)^{e+2} - (e+1)(p^2)^{e+1} + p^2}{(p^2-1)^2} + \frac{ep^{e+2} - (e+1)p^{e+1} + p}{(p-1)^2}\right)$$

Thus $A + B$ can be approximated by

$$\begin{aligned} A + B &\approx O\left(\frac{t}{\log_p 2}(p^2)^e \log_p p^e\right) \\ &\approx O\left(tr^2 \log_2 r\right). \end{aligned}$$

We record the result of the above analysis in the following theorem.

Theorem 4.3 *Let G be a finite abelian p-group and let β be an FTLS of type $(r_1, r_2, \ldots, r_t) = (r, r, \ldots, r)$ for G obtained from a TLS of type $(z_1, \ldots, z_s) = (z, z, \ldots, z)$. Then the factorization of an element $g \in G$ with respect to β using Algorithm 3 has a complexity of $O\left(tr^2 \log_2 r\right)$.*

The complexity as given in Theorem 4.3 shows that if the sizes for r are small, Algorithm 3 could still be considered as "efficient", but if r is getting large, Algorithm 3 will no longer be efficient. And because of the term r^2 involving in the complexity estimate, Algorithm 3 cannot be used to prove the tameness of FTLS for abelian groups.

In the next section we show that if the information of the transformations used for generating an FTLS β from a TLS is known, then we can construct a factoring algorithm proving the tameness of β.

4.1 Factorization with respect to FTLS by using trapdoor information

Assume that an FTLS β for an abelian group $\mathcal{G}$ is constructed from a TLS α using the four transformations (i), (ii), (iii) and (iv) as described at the beginning of the section. To be more precise, let the TLS $\alpha = [A_1, A_2, \ldots, A_s]$ of type $(z_1, \ldots, z_s)$ be derived from a chain of subgroups $G = G_0 > G_1 > \cdots G_s = 1$ of G.

In general, there is no particular order of using the transformations (i), (ii), (iii) and (iv), but for the sake of clarity we will generate an FTLS according to the following steps.

(T1) (Fusion) Perform a fusion of the blocks of α. The fusion transformation (iv) will be done as follows.

- Select a permutation $\varphi \in S_s$ and compute a logarithmic signature α' from α by

$$\alpha' = [A'_1, \ldots, A'_s] = [A_{\varphi(1)}, \ldots, A_{\varphi(s)}].$$

- Select a partition $P = \{P_1, \ldots, P_t\}$ on the set $\{1, \ldots, s\}$ with $P_1 = \{1, \ldots, i_1\}$, $P_2 = \{i_1 + 1, \ldots, i_2\}, \ldots, P_t = \{i_{s-1} + 1, \ldots, i_s\}$ with $|P_j| = u_j$, for $j = 1, \ldots s$. Fusing the blocks of α' according to this partition yields a logarithmic signature $\beta' := [B'_1, \ldots, B'_t]$ of type $(r_1, \ldots, r_t)$ with

$$B'_j = A'_{i_{j-1}+1}.A'_{i_{j-1}+2} \ldots A'_{i_j},$$

and $r_j = |A'_{i_{j-1}+1}|.|A'_{i_{j-1}+2}| \ldots |A'_{i_j}|$ for $j = 1, \ldots, t$ and $i_0 = 0$.
(i.e. each block B'_i is obtained by fusing certain consecutive blocks of α'.)

(T2) Select random permutations $\pi_j \in S_{r_j}$, $j = 1, \ldots, t$. Permute the positions of the elements of each block B'_j with permutation π_j. Let $\beta'' = [B''_1, \ldots, B''_t]$ denote the resulting logarithmic signature obtained from β' after this step.

(T3) Select random elements $g_j \in G$ and replace each block B''_j of β'' with $B''' := B''_j.g_j$. The resulting object is a logarithmic signature $\beta''' = [B'''_1, \ldots, B'''_t]$.

(T4) Select a random permutation $\xi \in S_t$ and permute the blocks of β''' by using ξ. The result obtained from this last step is our constructed FTLS $\beta = [B_1, \ldots, B_t]$.

We call the information about the transformations T1, T2, T3 and T4, which are used to generate an FTLS β from a TLS α, the *trapdoor information.*

Proposition 4.4 *Let $\alpha := [A_1, \ldots, A_s]$ be a transversal logarithmic signature for an abelian group G. Let $\beta' := [B'_1, \ldots, B'_t]$ be a fused transversal logarithmic signature for G obtained from α by using (only) the fusion transformation T1 . Then β' is equivalent to a logarithmic signature α' obtained from α by permuting its blocks with the permutation used by T1.*

Proof. Now suppose that β' is given. Let $\alpha' = [A'_1, \ldots, A'_s]$ be the logarithmic signature obtained from α by using the permutation $\varphi \in S_s$ for transformation T1, i.e.

$$\alpha' = [A'_1, \ldots, A'_s] = [A_{\varphi(1)}, \ldots, A_{\varphi(s)}].$$

Then it is clear that β' is equivalent to α'. □

As a consequence of Proposition 4.4 we see that instead of factoring with respect to an FTLS β we can factorize with respect to α by using the knowledge of transformations T1, T2, T3 and T4. This is presented in the following algorithm.

Algorithm 4 Factorization with FTLS by using trapdoor information

Input: α, $\varphi \in S_s$, $P = \{P_1, \ldots, P_t\}$, $\pi_i \in S_{r_i}$, $g_i \in G$, $i = 1, \ldots, t$, $\xi \in S_t$, and $y \in G$.
Output: $x = x_1 \| x_2 \| \ldots \| x_t$, such that $y = \beta(x)$.

1: Compute $y' = y.\prod_{i=1}^{t} g_i$ (here, $g_1, \ldots, g_t$ are elements in $\mathcal{G}$ which are used for transformation T3). Write $y' = y'_1 \| y'_2 \| \ldots \| y'_s$. Each y'_i is of $\lceil \log_2(r_i) \rceil$ bit length.

2: Factorize y' with respect to α by using Algorithm 2. Let denote $j'_1, \ldots, j'_s$ the indices obtained by this factorization.

3: Compute $j_\ell = j'_{\varphi^{-1}(\ell)}$ for $\ell = 1, \ldots, s$.

4: According to $P_\ell = \{i_1, i_2, \ldots, i_{u_\ell}\}$ set $x'_\ell = j_{i_1} \| j_{i_2} \| \ldots \| j_{i_{u_\ell}}$ for $\ell = 1, \ldots, t$.

5: Compute $x''_\ell = \pi_\ell^{-1}(x'_\ell)$ and finally compute $x_\ell = x''_{\xi^{-1}(\ell)}$ for $\ell = 1, \ldots, t$.

In Algorithm 4 we may assume that performing steps 1, 3, 4, 5 will take a constant time. Thus the complexity for factoring y with respect to β is reduced to the complexity of factoring y' with respect to the TLS α in step 2, which is $O(w)$ by Theorem 3.1, where $w = \lceil \log_2 |G| \rceil$. Thus we have the following theorem.

Theorem 4.5 *Let $\beta := [B_1, \ldots, B_t]$ be an FTLS constructed from a TLS $\alpha := [A_1, \ldots, A_s]$ for an abelian group G by using the transformations T1, T2, T3 and T4. Then β is tame if the trapdoor information about these transformations is known.*

5 Conclusion

We have presented factorization algorithms and their computational complexities for the classes of tranversal and fused transversal logarithmic signatures for finite abelian groups. The results have shown that transversal logarithmic signatures are tame, however, fused transversal logarithmic signatures are tame when trapdoor information is used. We have also presented a factorization algorithm for fused transversal logarithmic signatures based on the idea of Blackburn, Cid and Mullan; the complexity of this algorithm does not prove the tameness of the fused transversal logarithmic signatures. It is an interesting open problem to decide whether or not fused transversal logarithmic signatures for abelian groups are tame without using the trapdoor information.

References

[1] L. Babai, R. Beals and A. Seress, Polynomial-time Theory of Matrix Groups, *STOC'09, May 31-June 2, 2009*, 55–64.

[2] S. R. Blackburn, C. Cid, C. Mullan, Cryptanalysis of the MST_3 Public Key Cryptosystem, *J. Math. Crypt.* **3** (2009), 321–338.

[3] M. L. Furst, J. Hopcroft, and E. M. Luks, Polynomial-time algorithms for permutation groups, *Proc. 21st FOCS*, IEEE C.S., (1980), 36–41.

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

[5] S. S. Magliveras, B. A. Oberg and A. J. Surkan, A New Random Number Generator from Permutation Groups, In *Rend. del Sem. Matemat. e Fis. di Milano, **LIV** (1984)*, 203–223.

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

[7] S. S. Magliveras and N. D. Memon, Random Permutations from Logarithmic Signatures, *Computing in the 90's, First Great Lakes Comp. Sc. Conf., Lecture Notes in Computer Science,* Springer-Verlag, **507** (1989), 91–97.

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

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

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

[11] P. Marquardt, P. Svaba and Tran van Trung, Pseudorandom number generators based on random covers for finite groups, *Des. Codes Cryptogr.*, 10.1007/s10623-011-9485-1, Online 24 February 2011.

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

[13] P. Svaba and Tran van Trung, Public key cryptosystem MST3: cryptanalysis and realization, *J. Math. Crypto.* **4** (2010), 271–315.

[14] Sándor Szabó, Topics in Factorization of Abelian Groups, Birkhäuser Verlag, Basel - Boston - Berlin 2004.

[15] M. I. G. Vasco, A. I. P. del Pozo, P. T. Duarte A note on the security of MST_3 *Des. Codes Cryptogr.* **55** (2010), 189–200.