On Group Based Public Key Cryptography

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von

Anja Nuss

aus Münster

Tübingen
2011

Tag der mündlichen Qualifikation:	27.01.2012
Dekan:	Prof. Dr. Wolfgang Rosentiel
Erster Berichterstatter:	Prof. Dr. Peter Hauck
Zweiter Berichterstatter:	Prof. Dr. Klaus Reinhardt

Zusammenfassung

Diese Arbeit befasst sich mit der Analyse zweier Public Key Kryptosysteme, deren Sicherheit auf Berechnungsproblemen in endlichen Gruppen beruht.

Im ersten Fall werden logarithmische Signaturen untersucht, welches spezielle Faktorisierungen endlicher Gruppen sind. Diese bilden die Basis für die von Lempken, Magliveras, Stinson, van Trung und Wei entwickelten Kryptosysteme MST_1 und MST_3. Neben der Entwicklung einer rigiden Sicherheitsdefintion wurde in zwei verschiedenen Gruppenfamilien untersucht, welche logarithmischen Signaturen existieren und ob darunter logarithmische Signaturen sind, die Kandidaten für sichere MST-Systeme sein könnten. In der ersten Gruppenfamilie wurde eine einheitliche Charakterisierung sämtlicher logarithmischen Signaturen entwickelt mittels der gezeigt werden konnte, dass hier kein sicheres System möglich ist. In der zweiten Gruppenfamilie wurden verschiedene neue Methoden entwickelt um logarithmische Signaturen zu erzeugen und zu untersuchen. Auch hier konnte gezeigt werden, dass mit keiner der aktuell bekannten logarithmischen Signaturen ein sicheres MST-Kryptosystem aufgebaut werden kann.

Im zweiten Fall wird das MOR-System (von Paeng et al.) in einer Erweiterung der Gruppe $GL(n, q)$ untersucht. Die Sicherheit von MOR basiert auf der Schwierigkeit des diskreten Logarithmus Problems in Gruppen von inneren Automorphismen. Es werden zwei Ciphertext-only Angriffe entwickelt, die es ermöglichen die Sicherheit des Systems auf das Diskrete Logarithmus Problem in endlichen Körpern zu reduzieren.

Contents

Chapter 1

Introduction

Cryptographic techniques have become indispensable in our everyday life. Often unnoticed they secure online banking, tickets for public transportation, contactless payment cards, mobile communication, or passports.

Several encryption schemes exist that provide different security properties. One important property of each cryptosystem is whether it is symmetric or asymmetric. Symmetric or private key cryptosystems use the same key for encryption and decryption. Hence this key has to be exchanged in a secure way prior to sending a message. Symmetric techniques are rather old and date back to the Romans (Caesar Cipher), ancient Greeks (Skytale), and Hebrew scholars (Atbash Cipher). In contrast, asymmetric encryption was developed more recently. In 1976, W. Diffie and M. Hellman published the paper *New Directions in Cryptography* ([DH76]) in which they proposed the Diffie-Hellman key exchange and the concept of asymmetric encryption. This is also known as public key cryptography since every user of the system owns two keys: a public key, known to anyone, to encrypt messages to this user, and a private key, that is kept secret, to decrypt messages sent to the user. Computing the private key from the public key should be computationally infeasible: it requires a solution of a computational problem for which there are no efficient algorithms known.

One prominent example for such a problem is the discrete logarithm problem which is defined as follows: Given two elements g and h of a finite cyclic group, find a natural number a such that $g^a = h$. The number of multiplications needed to compute the a-th power of the element g is linear in the bitlength of g and thus efficiently computable. But all known algorithms that compute discrete logarithms are inefficient.

The discrete logarithm problem is closely connected to the security of the Diffie-Hellman key exchange as well as the ElGamal cryptosystem (developed by T. ElGamal in 1984, [ElG84]). In 1978 Rivest, Shamir and Adleman used the problem of finding the factorization of an integer as the basis for another type of cryptosystem, the RSA cryptosystem, [RSA78]. All public key cryptosystems currently in use are either designed via an ElGamal principle (including elliptic curve cryptosystems) or an RSA principle. Such a limited variety makes it an important task to study new approaches to design public key cryptosystems. Furthermore the development of quantum computing will make both today's well established public key cryptosystems insecure, [Sho94].

Several newly proposed cryptosystems use different computational problems that are presumed to be intractable. Among others there are for example the braid group system by Anshel, Anshel and Goldfeld which uses the word problem and the conjugacy problem, [AAFG01], the HFE family introduced by Patarin which uses the difficulty of solving multivariate equations, [Pat96], and the NTRU encryption algorithm which uses the shortest vector problem for lattices, [HPS98]. These systems as well as every new cryptosystem has to be analyzed intensively before it is further tested in implementation and integration into complex software or hardware systems. Although many systems have been proposed several years ago and have been tested and analyzed, none has gained an equal reputation to the RSA and ElGamal family. Some do not offer the same efficiency, some have security flaws, and many are not yet studied sufficiently.

In this work we study basic concepts of two recently proposed public key cryptosystems: MST and MOR. Both systems use a group theoretic idea to design a public key scheme. Introduced in [MSvT02], MST is based on the notion of logarithmic signatures, which are special decompositions of finite groups. As its basic concept MOR [P$^+$01b] uses the difficulty of computing discrete logarithms in groups of inner automorphisms.

In the next chapter we give the basic notations, definitions and theorems fundamental to the mathematical and computational aspects in this work. We introduce logarithmic signatures and the MST cryptosystem and additionally propose a new definition for one-way functions from logarithmic signatures. We conclude this chapter with a description of the MOR system as well as inner automorphisms and semidirect products for linear groups.

There is no known method to derive secure keys for the MST cryptosystems. Our approach in Chapter 3 is to study and characterize all logarithmic signatures in finite cyclic p-groups for a prime p. We give a complete characterization for logarithmic signatures in these groups and develop an efficient factoring

algorithm that can be applied to any cyclic p-group. Furthermore this is the first time that we can give the exact number of logarithmic signatures in a group.

The structure of logarithmic signatures in elementary abelian p-groups is the central point of Chapter 4. Logarithmic signatures of these groups are employed by the cryptosystem MST_3. This system was shown to be insecure if the keys are derived from an exact transversal logarithmic signature via certain basic transformations, see [MSvTZ08] and also [BCM09]. A different way of generating logarithmic signatures of these groups is not known.

Therefore we focus our research on an extensive analysis of logarithmic signatures in elementary abelian p-groups. First we study linear independence between the elements of a logarithmic signature and show that they always contain a basis that is ideally distributed among the buildings blocks of a logarithmic signature. Furthermore we use this result to give a complete characterization of logarithmic signatures in elementary abelian 2-groups with small building blocks. It follows a second characterization of these logarithmic signatures using graphs and a polynomial-time factoring algorithm for a class of logarithmic signatures in these groups. Then we introduce a system of certain linear equations that can be derived from any logarithmic signature in elementary abelian p-groups and may lead to more general results of the hardness of factoring in elementary abelian p-groups. Furthermore we develop a factoring algorithm for logarithmic signatures with a Rédei block in each iteration of the algorithm. We also extend this algorithm to the case in which in a constant number of iterations only the sum of two blocks is a Rédei block. Furthermore we construct a series of logarithmic signatures which are not factorable by the algorithms presented in this thesis.

In Chapter 5 we consider the MOR cryptosystem that was introduced by Paeng et al., see [P$^+$01b, P$^+$01a]. This ElGamal-type public key cryptosystem is based on the intractability of the discrete logarithm problem in the group of inner automorphisms for a non-abelian group. It was analyzed by Tobias in [Tob03, Tob04a] and also in [Kor05, Kor08]. We consider a more general case with a semidirect product of a general linear group and an arbitrary abelian group of automorphisms. First we show how it is possible to compute the components of an automorphism that is a component of the keys, thus justifying the attacks from [Kor05, Kor08] and building the basis for further attacks. For these we assume that computing discrete logarithms in small extensions of finite fields is efficient. In this scenario we demonstrate how to compute a key that allows an adversary to obtain the plaintext. We show that despite its complex structure, this group opens several possibilities for an

attack and does not yield more security than the discrete logarithm problem in the multiplicative group of finite fields.

Many people have influenced this thesis in various ways. I would like to thank my advisor Prof. Peter Hauck who introduced to me the field of cryptology. Thank you for your valuable advice and guidance, your active interest in my work and ongoing support throughout the entire period in Tübingen as well as Bochum. Prof. Klaus Reinhardt who shares the interest in cryptology and readily became my second advisor.

Steffi, Jonas, Claudia, and Michael, I had a great time with you all with cryptographic and non-scientific discussions and riddles, building expertise in sauces and ice cream, and becoming a soccer expert. I would like to thank Christoph and Andreas for extensive discussions on SAT, logarithmic signatures and one-way functions in the theory lounge. Furthermore I would like to thank Rouven Walter with whom I worked on the topic of Chapter 4.2 and Jan deWiljes with whom I could discuss the problem of generating logarithmic signatures. In the final stage several people at the HGI helped: Saqib by proofreading, Eike by simply lending an office, and Enrico and Mazze via valuable discussions on the link between the shortest vector problem and factoring logarithmic signatures.

I would like to thank my family for their constant support from several hundred kilometers away. Julia, it has been a great journey through science and life with you. Marcel, you know and thats why I dedicate this thesis to you.

Finally I would like to thank the Evangelische Studienförderung Villigst e.V. for supporting and enabeling this work with a grant.

Chapter 2

Mathematical and cryptographic background

In this chapter we give the basic notations, definitions and theorems fundamental to the mathematical and computational aspects in this work. We introduce logarithmic signatures and the MST cryptosystem and additionally propose a new definition for one-way functions from logarithmic signatures. We conclude this chapter with a description of the MOR system as well as inner automorphisms and semidirect products for linear groups.

2.1 Notation and mathematical concepts

In our analysis we combine tools from several mathematical areas. We assume that the reader is familiar with the basic concepts of group theory (see [KS98, Rob82]), linear algebra (see [Fis97]), finite fields (see [LN94]), and graph theory (see [BJG08]).

Notation. $\mathbb{N}$ denotes positive integers without zero. For a prime number p and $m \in \mathbb{N}$ we define a finite field of order $q = p^m$ as $\mathbb{F}_q$. For the multiplicative group of a finite field we use the notation $\mathbb{F}_q^*$, where ξ is the generating element. By $|x|_2$ we denote the bitlength of any input x.

Groups. We consider only finite groups. In general we use multiplicative notation, but for abelian groups we use addition to indicate the group operation.

The neutral element is e_G, 0 (additive notation) or 1 (multiplicative notation). For a subset $A = \{a_1, \ldots, a_n\}$ of a group $(G, \cdot)$ we set

$$< A > := \ \{a_1^{i_1} \cdot \ldots \cdot a_n^{i_n} \mid i_1, \ldots, i_n = 0, 1, \ldots, |G|\}.$$

If A only contains one element a we use the notation $< a >$ to describe the set $\{e_G, a^1, \ldots, a^{k-1}\}$ where $k \in \mathbb{N}$ is the order of a. Let

$$Z(G) = \{g \in G | xg = gx \text{ for all } x \in G\}$$

represent the center of a group G and $C_G(x) = \{g \in G | xg = gx\}$ the centralizer of an element x in G. Given a group G and a subgroup H of G, a set X such that $HX = G$ and $|H| \cdot |X| = |G|$ is called a transversal of H in G.

Let p be a prime number and $m, n \in \mathbb{N}$, then

$$\mathbb{Z}_{p^m} = \{0, 1, \ldots, p^m - 1\}$$

stands for the cyclic group of order p^m with componentwise addition modulo p^m as the group operation. Furthermore

$$\mathbb{Z}_p^n = \underbrace{\mathbb{Z}_p \times \ldots \times \mathbb{Z}_p}_{n}$$

is the elementary abelian p-group of order p^n with addition modulo p as the group operation and neutral element $0 = (0, \ldots, 0)$. An element of this group is denoted by a lower case letter (i.e. v) or more specifically by a vector with n positions $(v(1), \ldots, v(n))$. Note that $\mathbb{Z}_p^n$ is also a $\mathbb{Z}_p$-vectorspace. The vectors $e_1, \ldots, e_n$ form the canonical basis, where $e_1 = (1, 0, \ldots, 0)$, $e_2 = (0, 1, 0, \ldots, 0)$ and so on. We use the notation $v(i, j) = (v(i), v(j))$.

Functions. We use the notation $f : M_1 \to M_2$, $x \mapsto f(x)$ to describe a function f that maps each element x of the set M_1 to an element $f(x)$ in M_2. In Chapter 5 we work with MOR and the groups $Aut(G)$ and $Inn(G)$ of automorphisms and inner automorphisms of the group G. Automorphisms are bijective homomorphisms from G to G. For $g \in G$

$$\begin{aligned} I_g : G &\longrightarrow G \\ x &\longmapsto gxg^{-1} \end{aligned}$$

denotes the inner automorphism induced by g. Note that $G/Z(G) \cong Inn(G)$. Two elements $g, h \in G$ induce the same inner automorphism if and only if $g \in h \cdot Z(G)$.

Linear algebra. By $GL(n, q)$ we denote the set of all invertible $n \times n$ matrices with entries from the finite field $\mathbb{F}_q$ for a prime power $q = p^m$ $(n, m \in \mathbb{N})$. The entries of a $n \times n$ matrix are marked with indices (e.g. $M = (m_{ij})_{i,j=1,\ldots,n}$). A matrix is in *row echelon* form if the first nonzero entry from the left of a row is always strictly to the right of the first nonzero entry of the row above it. A matrix is in *reduced row echelon* form if it is in row echolon form, in every row the leading coefficient equals 1, and it is the only nonzero entry in its column.

Cryptography. Convenient references for cryptographic concepts and notions, e.g. one-way function or public key cryptosystem are [Gol01, Gol04, MvOV96].

Efficiency. In order to describe the efficiency or complexity of an algorithm we use the Big O notation; for reference see [Weg03]. In this context a deterministic algorithm is called a *polynomial-time algorithm* if its running time is bounded by a polynomial in the length of its input.

We also consider polynomially bounded *Las-Vegas-algorithms*. These are algorithms that are allowed to make random choices of bits (randomized algorithms) and whose maximal running time (on inputs of length at most n) has an expected value (with regard to the random choices) that is polynomial in n. We call such algorithms also *probabilistic polynomial-time algorithms*.

To be more precise, problems that can be solved by probabilistic polynomial-time algorithms in our sense belong to the complexity class ZPP. ZPP stands for "zero-error probabilistic polynomial time" and refers to the fact that ZPP can also be characterised as consisting of those problems, for which there exists a randomized algorithm of polynomially bounded running time that outputs either a correct solution or a statement of failure, and the latter happens with probability at most 1/2.

There are other complexity classes of randomized polynomial-time algorithms, like RP or BPP, that are of no relevance for our work. We just mention that ZPP is the smallest of these classes containing the class P of problems that can be solved in polynomial time by a non-randomized (deterministic) algorithm. For details we refer to [Weg03], chapter 3.

If there exists a probabilistic polynomial-time algorithm to solve a problem, the problem is *efficiently* or *easy* to solve. Otherwise, the problem is said to be *hard* to solve, or *computationally infeasible*.

We refer to a computational problem P_1 as *polynomial-time reducible* to a problem P_2 (i.e. $P_1 \Rightarrow_P P_2$) if there exists a deterministic polynomial-time

algorithm that solves P_1 and that may use an (unspecified) algorithm for P_2 as a subroutine.

GAP. All programs are written in GAP, a system for computational discrete algebra, especially computational group theory. We used GAP Version 4.4.10, see [GAPrg]. In this work all algorithms are given in GAP-style pseudo code. We only give the code essential for the algorithms and leave out extraneous sections.

2.2 Logarithmic signatures and the MST cryptosystems

The concept of logarithmic signatures was first used for the symmetric cryptosystem PGM by Magliveras in [Mag86]. A public key cryptosystem using logarithmic signatures was proposed by Webb in 1991, see [Web91], and found to be insecure by Qu and Vanstone in [QV94]. Logarithmic signatures are also the basis for the public key cryptosystems MST_1, MST_2 and MST_3. These were designed in 2002 and 2009 by Magliveras, Stinson, van Trung, Lempken and Wei, see [MSvT02, LvTMW09].

2.2.1 Logarithmic signatures

Let G be a finite group. A logarithmic signature α of G is a sequence

$$\alpha = [A_1, \ldots, A_s]$$

of subsets $A_i \subseteq G$ such that for every element g of G there exists exactly one *factorization*

$$(*) \quad g = a_1 \cdot a_2 \cdot \ldots \cdot a_s,$$

where $a_i \in A_i$ for $i = 1, \ldots, s$. The sets A_i are called *blocks*. The size of a block ist denoted by $r_i := |A_i|$. For simplicity, we refer to the elements of $A_1 \cup \ldots \cup A_s$ as the *elements of the logarithmic signature* α. In some cases we consider an ordering on the elements of a block, then for $k_i = 0, \ldots, r_i - 1$ we denote by a_{ik_i} the $(k_i + 1)$-th element of the block A_i. The vector $(r_1, \ldots, r_s)$ is said to be the *type* of α and

$$\ell(\alpha) = \sum_{i=1}^{s} r_i$$

is the *length* of a logarithmic signature. By $\Lambda(G)$ we denote the set of all logarithmic signatures of a group.

From this definition we immediately derive some properties of logarithmic signatures. Because of the existence and the uniqueness of the factorization $(*)$ we have

$$\prod_{i=1}^{s} r_i = |G|$$

and thus r_i divides $|G|$ for all i. This also indicates why these sequences are called logarithmic signatures: The logarithm function turns a product into a sum - a logarithmic signatures provides an abbreviated representation of all group elements that is of length $r_1 + \ldots + r_s$ while the group itself contains $r_1 \cdot \ldots \cdot r_s$ elements.

If $e_G \in A_1 \cap \ldots \cap A_s$, then we say that α is *normalized*. Because of the uniqueness of the factorization we even have $\bigcap_{i=1}^{s} A_i = \{e_G\}$ for normalized logarithmic signatures with more than one block. It is easy to see that in abelian groups we have $|A_i \cap A_j| \leq 1$ for $i \neq j$.

In [GRS03] González Vasco and Steinwandt observed that for $|G| = \prod_{j=1}^{t} p_j^{b_j}$ (p_j prime) there is a lower bound on the length of any logarithmic signature of G given by

$$\ell(\alpha) \geq \sum_{j=1}^{t} b_j \cdot p_j.$$

A logarithmic signature α is called *minimal* if $\ell(\alpha)$ achieves the lower bound, i.e. every block is of prime order or of order 4. Several papers deal with the question of the existence of minimal logarithmic signatures in finite groups, see for example [GRS03, SSM10]. This reasearch suggests that minimal logarithmic signatures exist for all finite groups.

Example 2.1. Let $n \in \mathbb{N}$. For the cyclic group $(\mathbb{Z}_{2^n}, +)$ the sequence

$$\alpha = [\ [0, 2^{n-1}],\ [0, 2^{n-2}],\ \ldots,\ [0, 2],\ [0, 1]\ ]$$

is a normalized minimal logarithmic signature of type $(2, \ldots, 2)$. Computing the factorization of an element is equivalent to computing its binary representation; for example if $n = 4$, then $9 = 1001$ has the factorization $2^3 + 0 + 0 + 2^0$.

2.2.2 Tame and wild logarithmic signatures

Given a logarithmic signature and some group element we study if it is feasible to compute the factorization of the group element. For instance, a brute force search through all possible factorizations given by a logarithmic signature $\alpha = [A_1, \ldots, A_s]$ of a group G takes $|G| \times (s-1)$ group operations in the worst case. Such a search finds the correct factorization for any logarithmic signature, but it is infeasible in general.

Example 2.1 shows that for some logarithmic signatures factorizations are easy to compute. For the MST cryptosystems it is necessary to identify logarithmic signatures for which factorizations are infeasible to compute as well as those with efficient factoring algorithms. Usually, the terms *tame* and *wild* are used to differentiate between logarithmic signatures for which factorizations are easy or hard to compute, see [MSvT02, LvTMW09]. However no precise definition of these terms has been given. Therefore we discuss a rigorous definition related to the concept of one-way functions (as defined in [Gol01], p. 33).

First note that factoring with respect to a single logarithmic signature takes constant time. Therefore when looking at the efficiency of calculations for logarithmic signatures, we consider families $(G_n,\ \alpha_n)$ of logarithmic signatures

$$\alpha_n = [A_1^n, \ldots, A_{s_n}^n]$$

of groups G_n for $n \in \mathbb{N}$. Furthermore we assume that

$$|G_n| \leq |G_{n+1}|$$

and

$$\alpha_n \neq \alpha_m \text{ for } n \neq m.$$

Additionally, let there be a uniform description of elements of G_n such that

$$|g|_2 = \mu_n \text{ for all } g \in G_n.$$

Note that this yields $\mu_n \geq \log\ |G_n|$ for all $n \in \mathbb{N}$.

We make one basic assumption: For a given family $(G_n,\ \alpha_n)$ there exists a (deterministic) polynomial-time algorithm A such that on input $(a_1, \ldots, a_{s_n})$ from $A_1^n \times \ldots \times A_{s_n}^n$ algorithm A outputs the product $a_1 \cdot \ldots \cdot a_{s_n}$. We identify A with the injective function which it computes.

Definition 2.2. For $n \in \mathbb{N}$ let α_n be as above. Then the family $(G_n,\ \alpha_n)_{n\in\mathbb{N}}$ is called *wild* if for every probabilistic polynomial-time algorithm A', every positive polynomial p and all sufficiently large n

$$pr\left(A'(g_n, \alpha_n) = A^{-1}(g_n)\ \right) < \frac{1}{p(\lceil \log |G_n| \rceil)}$$

where g_n denotes a random element uniformly chosen from G_n.

For a wild family of logarithmic signatures $(G_n,\ \alpha_n)_{n\in\mathbb{N}}$ the function A defines a one-way function in the sense of [Gol01]. Among the logarithmic signatures that are not wild we identify those with efficient factorizations for all group elements:

Definition 2.3. For $n \in \mathbb{N}$ let α_n be a logarithmic signature of the group G_n. Then the family $(G_n,\ \alpha_n)_{n\in\mathbb{N}}$ is called *tame* if there exists an algorithm A'' that on input $g_n \in G_n$ and α_n computes the factorization of g_n with respect to α_n in polynomial time.

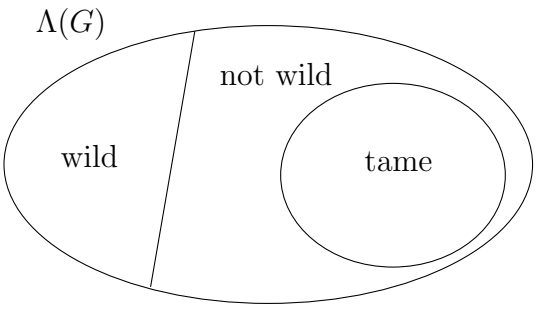


Figure 2.1: wild and tame logarithmic signatures of a group G

There are several remarks to these definitions:

1. The distinction between "not wild" and "tame" is new and unlike prior definitions (in which these two are equal) it allows to give a precise definition. A logarithmic signature that is neither tame nor wild could be one where we can efficiently find the factorization for exactly half the group elements.

2. Any algorithm that given g and α_n randomly guesses a factorization has a success probability of $\frac{1}{|G_n|}$. Therefore the algorithms A' in Definition 2.2 are allowed a slightly greater chance but still less than $\frac{1}{p(\lceil \log |G_n| \rceil)}$ (for any p) to find a factorization.

3. A' could additionally get n as an input to encode the group G_n such that A' knows G_n. But we argue that via the input α_n it is ensured that A' knows the group G_n. We also assume that on input $(a_1, \ldots, a_{s_n})$ algorithm A knows which group operation to use.

4. Note that computing the product of s_n elements, each chosen uniformly at random from a different block of α_n, (i.e. apply A) is the same as uniformly choosing a random element from G_n.

To improve readability, we usually omit the words "family of" and the groups G_n and only use the term wild/tame logarithmic signature (α_n).

Note that it is still an open question whether candidates for wild logarithmic signatures exist. All logarithmic signatures described in literature are shown to be tame (see [MSvT02, LvTMW09, BCM09, SSM10, GRS03]. The MST-cryptosystems (see Section 2.2.4) use wild as well as tame logarithmic signatures to construct public key cryptosystems: While MST_1 is based on the existence of wild logarithmic signatures, the version MST_3 uses tame logarithmic signatures in elementary abelian 2-groups. Therefore we are interested in the structure of tame logarithmic signatures as well as in the question whether there are groups for which wild logarithmic signatures might exist. In Chapters 3 and 4 we study logarithmic signatures and especially the question of the possibility of the existence of wild logarithmic signatures for cyclic and elementary abelian p-groups.

Three parameters are of importance in our analysis: Recall that for $n \in \mathbb{N}$ the logarithmic signature $\alpha_n = [A_1^n, \ldots, A_{s_n}^n]$ is fully described by

$$|\alpha_n| = \sum_{i=1}^{s} |A_i^n|$$

elements from G_n. There might be a shorter description for certain logarithmic signatures, but in general we would need

$$\sum_{i=1}^{s} |A_i^n| \cdot \mu_n$$

bits to describe the logarithmic signature, where an element of G_n is represented by μ_n bits. Let

$$r_n := \max\{|A_i^n|, \ldots, |A_{s_n}^n|\}.$$

The factoring algorithm A' in Defintion 2.2 takes an element of G_n and a logarithmic signature α_n for some $n \in \mathbb{N}$ as an input. The length of that input is at most

$$(s_n \cdot r_n + 1) \cdot \mu_n$$

bits. Therefore we propose to use the three values

$$s_n,\ r_n,\ \text{ and } \mu_n$$

as the parameters to measure the efficiency of a factoring algorithm for α_n. Note that we derive the first two parameters from the structure of α_n, the third parameter is independent of α_n and depends only on the representation of the elements of G_n.

Remark 2.4. For $n \in \mathbb{N}$ let (α_n) be a logarithmic signature in G_n. If for all $g \in G_n$ the factorization with respect to α_n can be retrieved in time polynomial in the three parameters

$$s_n,\ r_n,\ \text{and } \mu_n,$$

then the family $(\alpha_n)_n$ is *tame*.

Example 2.5. Take the family of logarithmic signatures in the group $G_n = \mathbb{Z}_{2^n}$ from Example 2.1. The three efficiency parameters are

$$s_n = n,\ r_n = 2,\ \text{and } \mu_n = n.$$

Given an element $g \in \mathbb{Z}_{2^n}$ by its binary representation $(g_1, \ldots, g_n)$ where $g_i \in \{0, 1\}$, its factorization in terms of α is $(g_1 \cdot 2^{n-1}, \ldots, g_n \cdot 1)$ which we retrieve with at most n multiplications, thus in time linear in n. So $(\alpha_n)_{n\in\mathbb{N}}$ is tame.

We see in Chapters 3 and 4 that these parameters are a good choice for modeling the complexity of algorithms for factoring with respect to logarithmic signatures.

2.2.3 On classifying logarithmic signatures

Now we address the question of how and where to find logarithmic signatures which cannot be candidates for wild logarithmic signatures. We consider certain transformations of logarithmic signatures, such that factoring with respect to the original logarithmic signature is equally efficient as factoring with respect to the transformed logarithmic signature. The idea is that a factoring algorithm for one logarithmic signature in a certain set yields a factoring algorithm for all logarithmic signatures of that set. We also describe the standard approach to classify logarithmic signatures according to the structure of the blocks.

Transformations on logarithmic signatures

We consider five transformations of logarithmic signatures that do not change the property of being tame or wild. We use the standard notation. Let $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signature of a group G. We look at transformations of the blocks $A_1, \ldots, A_s$ of α into blocks $B_1, \ldots, B_s$ such that the resulting sequence $\beta = [B_1, \ldots, B_s]$ is again a logarithmic signature of G.

T1 Let φ be an automorphism of G, and

$$B_i = \varphi(A_i)$$

for $i = 1, \ldots, s$. Then also β is a logarithmic signature. And if $a_1 \cdot a_2 \cdot \ldots \cdot a_s$ is a factorization of an element $g \in G$, then $\varphi(a_1) \cdot \varphi(a_2) \cdot \ldots \cdot \varphi(a_s)$ is the factorization of $\varphi(g)$ in terms of β.

T2 Let $g_0, \ldots, g_s$ be elements of G, and

$$B_i = g_{i-1}^{-1} A_i g_i$$

for $i = 1, \ldots, s$. Then also the sequence β is a logarithmic signature of G, called a *translation* of α. If $g_0 = g_s = e_G$, then β is called a *sandwich* of α. Note that if $a_1 \cdot a_2 \cdot \ldots \cdot a_s$ is a factorization of an element $g \in G$, then $g_0^{-1} a_1 g_1 \cdot g_1^{-1} a_2 g_2 \cdot \ldots \cdot g_{s-1}^{-1} a_s g_s$ is the factorization of $g_0^{-1} g g_s$ in terms of β.

Note that in abelian groups the blocks of a translation β of α are of the form

$$B_i = A_i + h_i$$

for elements $h_1, \ldots, h_s$ of G. And any factorization of an element $g \in G$ in terms of α immediately yields a factorization of $g + \sum_{i=1}^{s} h_i$ in terms of β.

T3 For $i = 1, \ldots, s$ let π_i be a permutation in S_{r_i}, and

$$B_i = [a_{i\pi_i(1)}, \ldots, a_{i\pi_i(r_i)}]$$

for $j = 1, \ldots, r_i$, i.e. the elements of block B_i are a permutation of the elements of block A_i. Then also β is a logarithmic signature. And if

$$a_{1k_1} \cdot a_{2k_2} \cdot \ldots \cdot a_{sk_s}$$

is a factorization of an element $g \in G$, then

$$a_{1\pi_1(k_1)} \cdot a_{2\pi_2(k_2)} \cdot \ldots \cdot a_{s\pi_s(k_s)}$$

is a factorization of g in terms of β.

T4 Now let G be an abelian group, π a permutation in S_s, and

$$B_i = A_{\pi(i)},$$

i.e. $b_{ij} = a_{\pi(i)j}$. Then the sequence β is a logarithmic signature of G, called a *block permutation* of α. Note that if

$$a_{1i_1} + a_{2i_2} + \ldots + a_{si_s}$$

is a factorization of an element $g \in G$ with respect to α, then

$$a_{\pi(1)i_1} + a_{\pi(2)i_2} + \ldots + a_{\pi(s)i_s}$$

is the factorization of g in terms of β. Note that in non abelian groups β might not be a logarithmic signature.

T5 For some $j \in \{1, \ldots, s-1\}$ let

$$B_j = A_j \cdot A_{j+1} = [\ x \cdot y \mid x \in A_j,\ y \in A_{j+1}\ ]$$

and $B_i = A_i$ for $i = 1, \ldots, s-1$ and $i \neq j, j+1$. The sequence $\beta = [B_1, \ldots, B_{s-1}]$ is a logarithmic signature, derived from α by *fusing* two blocks. And if $a_1 \cdot a_2 \cdot \ldots \cdot a_s$ is a factorization of $g \in G$ with respect to α, then $a_1 \cdot a_{j-1} \cdot a \cdot a_{j+2} \ldots \cdot a_s$ with $a = a_j \cdot a_{j+1}$ is the factorization of g in terms of β.

The reverse operation is called *splitting*.

For each of the transformations T1 to T5 we described how a factorization of an element with respect to a logarithmic signature immediately yields a factorization with respect to the transformed logarithmic signature. If we consider these transformations for families of logarithmic signatures (α_n), it is easy to see that switching between factoring algorithms for (α_n) and a transformation of (α_n) is polynomial if the transformation is known or efficiently computable. The same argument applies to the normalization. We propose the

Definition 2.6. Let (α_n) and (β_n) be families of logarithmic signatures for the groups G_n with the parameters $r_n,\ s_n,\ \mu_n$ for (α_n). Then we say that (α_n) *transformes into* (β_n) if (β_n) is computable from (α_n) via (repeated) transformations T1 to T5 in time polynomial in $r_n,\ s_n,\ \mu_n$. Note that in this case the number of transformations between (α_n) and (β_n) is limited.

The *T-set* of (α_n) is defined as

$$T(\alpha_n) = \{(\beta_n) \mid \beta_n \in \Lambda(G_n) \text{ and } \alpha_n \text{ transforms into } \beta_n \text{ for all } n\}.$$

Example 2.7. Take the logarithmic signature

$$\alpha_n = [[0, 2^{n-1}], \ldots, [0, 2], [0, 1]]$$

of the group $\mathbb{Z}_{2^n}$ from Example 2.1. For $n > 4$ let

$$\beta_n = [[1, 5], [0, 1, 2, 3], [0, 8], [0, 16], \ldots, [0, 2^{n-1}]]$$

and $\gamma_n = [[0, 1, 2, 3, \ldots, 2^n - 1]]$. Then $(\beta_n) \in T(\alpha_n)$ and $(\gamma_n) \notin T(\alpha_n)$.

In Chapters 3 and 4 we study logarithmic signatures in cyclic and elementary-abelian p-groups. We identify certain types of tame logarithmic signatures. These results apply not only to the specific family of logarithmic signature that is studied in those chapters but also to all families of logarithmic signatures in the T-sets. One transformation that we use repeatedly is

Definition 2.8. Let $g_0 = 1$ and $g_i = (\prod_{j=1}^{i} a_{j1})^{-1}$ for $i = 1, \ldots, s$. By Translating α with $g_0, \ldots, g_s$ we derive a logarithmic signature β, where

$$b_{i1} = g_{i-1}^{-1} a_{i1} g_i = ((\prod_{j=1}^{i-1} a_{j1})^{-1})^{-1} a_{i1} (\prod_{j=1}^{i} a_{j1})^{-1} = (\prod_{j=1}^{i} a_{j1}) \cdot (\prod_{j=1}^{i} a_{j1})^{-1} = 1,$$

i.e. the first element in each block is the neutral element. We say that β is the *normalization of* α.

Note that in abelian groups we may normalize a logarithmic signature using the translation $B_i = A_i - a_{i1}$. Then the first element of each block is $a_{i1} - a_{i1} = 0$.

Standard classes of logarithmic signatures

There are the following classes that are standard notation: Let G be a finite group. We call a logarithmic signature $\alpha \in \Lambda(G)$ *exact-transversal* if there exists a chain of subgroups

$$e_G = G_0 < G_1 < \ldots < G_{s-1} < G_s = G$$

such that A_i is a transversal of G_{i-1} in G_i, i.e. $G_{i-1}A_i = G_i$ and $|G_{i-1}||A_i| = |G_i|$. Note that the block $A_1 = G_1$ is a subgroup of G. The corresponding class is denoted by $\mathcal{E}$. If α is a sandwich of an exact-transversal logarithmic signature, then α is called *transversal*. The class of transversal logarithmic signatures is $\mathcal{T}$.

All other logarithmic signatures are in the class $\mathcal{NT}$ of *non-transversal* logarithmic signatures. We describe two subclasses of $\mathcal{NT}$. If none of the blocks is a coset of a (non-trivial) subgroup of G, then the logarithmic signature is an element of the class $\mathcal{TNT}$ of *totally-non-transversal* logarithmic signatures. The class $\mathcal{TA}$ of *totally-aperiodic* logarithmic signatures contains all logarithmic signatures which do not even have a block that is *periodic*, i.e. the union of cosets of a subgroup of G. Clearly,

$$\mathcal{TA} \subseteq \mathcal{TNT} \subseteq \mathcal{NT} \text{ and } \mathcal{E} \subseteq \mathcal{T}.$$

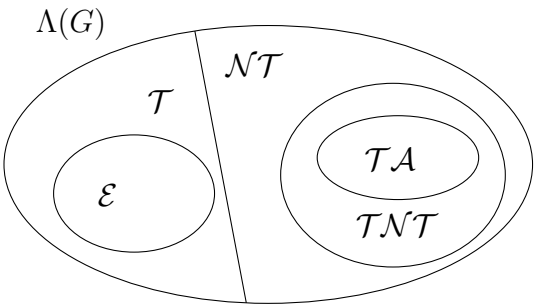


Figure 2.2: Standard classes of logarithmic signatures of a group G

It is easy to see that in certain groups being exact-transversal is a tame inducing property for logarithmic signatures:

Proposition 2.9. *Let (α_n) be a family of exact-transversal logarithmic signatures. Furthermore let subgroup membership testing in G_n be possible in time polynomial in μ_n. Then (α_n) is tame.*

Proof. For readability we omit the index n. Let

$$e_G = G_0 < G_1 < \ldots < G_{s-1} < G_s = G$$

be the subgroup chain corresponding to the logarithmic signature $\alpha = [A_1, \ldots, A_s]$. Let $g \in G$. There exists exactly one factorization for g: This factorization consists of $a_1, \ldots, a_s$ where

$$g \cdot a_s^{-1} \cdot \ldots \cdot a_{k+1}^{-1} = a_1 \cdot \ldots \cdot a_k \in G_k$$

for $k = 1, \ldots, s-1$ and $a_1 \cdot \ldots \cdot a_s = g$. The following naive algorithm finds a factorization for g with respect to α:

GAP: TransversalFactorization

Input: logSig, a logarithmic signature of G, and an element g $\in G$
Output: B, the factorization of g with respect to logSig

```
TransversalFactorization:=function(logSig,g)

B:=[]; h:=g;

for i in [1..Length(logSig)] do

  for a in logSig[s-i+1] do

    if h*Inverse(a) in G_i
       then h:=h*Inverse(a);
            Add(B,a);
    fi;
  od;
od;

return B;

end;
```

The algorithm is deterministic, needs $O(r_n \cdot s_n)$ rounds and in each round one multiplications, one inversion and one subgroup membership test, which is a running time polynomial in μ_n, s_n and r_n. □

Note that it is also immediately clear that if μ_n, s_n and r_n are polynomial in $\lceil \log |G_n| \rceil$ and (α_n) is a tame family of normalized exact-transversal logarithmic signatures, then there exists an efficient subgroup membership test for all subgroups G_i in G_n: $g \in G_i$ only if in the factorization $g = a_1 \cdot \ldots \cdot a_s$ we have $a_{i+1} = \ldots = a_s = e_G$.

With this result in mind it seems likely that non-transversal or even totally-non-transversal logarithmic signatures are good candidates for being wild. But in [BGCS05] Bohli et al. show that it is easy to construct tame families of logarithmic signatures for symmetric and alternating groups that are in the class $\mathcal{TNT}$. In Section 4.5 we give a series of tame logarithmic signatures of

elementary abelian p-groups that are in the class $\mathfrak{TA}$. Thus we observe that the classes do not give a criterion to distinguish between tame and wild logarithmic signatures.

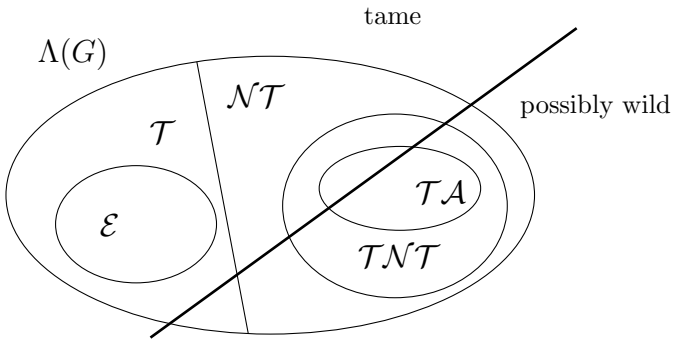


Figure 2.3: Distribution of tame logarithmic signatures among all logarithmic signatures of a generic group G.

In fact if no efficient algorithms for subgroup membership testing exist for the groups G_n, then there might also be wild logarithmic signatures in those groups that are transversal.

We have just seen that for groups with efficient subgroup membership testing algorithms exact-transversal logarithmic signatures are tame. We formalize this with the following defintion.

Definition 2.10. For $n \in \mathbb{N}$ let P_n be a boolean predicate that can be applied to any logarithmic signature of the group G_n. We say that P is a *tame inducing property for* G_n if all families $(\alpha_n)_{n\in\mathbb{N}}$ of logarithmic signatures of (G_n) such that

$$P(\alpha_n) \text{ is true for all } n \in \mathbb{N}$$

are tame.

Example 2.11. (1) Again we look at Example 2.1. For $n \in \mathbb{N}$ let $P_n =$ "every block is of the form $[0, 2^i]$ for $0 \leq i \leq n$". Then we saw in Example 2.5 that P is a tame inducing property for $\mathbb{Z}_{2^n}$.

(2) In groups with efficient subgroup membership testing, being transversal is a tame inducing property.

To improve readability we may omit the parameter n in the following.

2.2.4 The MST cryptosystems

Magliveras, Stinson and van Trung developed two approaches to design public key cryptosystems, called MST_1 and MST_2, see [MSvT02]. While MST_1 uses logarithmic signatures, the second version of MST is based on covers, which differ from logarithmic signatures in that the factorization $(*)$ is not unique. A third cryptosystem, MST_3, was recently presented by Lempken, van Trung, Magliveras and Wei in [LvTMW09]. It uses tame logarithmic signatures as well as covers. Here we give an overview on setup, encryption and decryption for the two systems that involve logarithmic signatures.

To describe the systems a certain notation is used: A logarithmic signature induces an order on the elements of G. This is used to define the following function: Let $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signature of type $(r_1, \ldots, r_s)$ for the group G. If we consider ordered blocks, then we derive a bijection from $\mathbb{Z}_{|G|}$ to G. First we take the canonical bijection that identifies $\mathbb{Z}_{r_1} \times \ldots \times \mathbb{Z}_{r_s}$ with $\mathbb{Z}_{|G|}$:

$$\begin{array}{rccc} \tau & : & \mathbb{Z}_{r_1} \times \ldots \times \mathbb{Z}_{r_s} & \longrightarrow & \mathbb{Z}_{|G|} \\ & & (k_1, \ldots, k_s) & \longmapsto & \displaystyle\sum_{i=1}^{s} (\prod_{j=1}^{i-1} r_j) \cdot k_i \end{array}$$

Then α induces the function

$$\begin{array}{rccc} \alpha & : & \mathbb{Z}_{|G|} & \longrightarrow & G \\ & & x & \longmapsto & \displaystyle\prod_{i=1}^{s} a_{ij_i} \end{array}$$

where $\tau^{-1}(x) = (j_1, j_2, \ldots, j_s)$.

Note that τ and τ^{-1} are efficiently computable and so is the function α. But in order to be able to compute $\alpha^{-1}(g)$ efficiently for any element $g \in G$, the logarithmic signatures α has to be tame.

For some arbitrary but fixed tame logarithmic signature η of G, we define the product for two logarithmic signatures: For $\alpha, \beta \in \Lambda(G)$ and $x \in \mathbb{Z}_{|G|}$ set

$$\alpha \cdot \beta(x) = \alpha(\eta^{-1}(\beta(x)))$$

Public Key Encryption Scheme $\boldsymbol{MST_1}$

In [CV06] Caranti and dalla Volta essentially proof the following: For a non-trivial finite group G, that is not cyclic of order a prime or the square of a prime, every logarithmic signature of G is a product of transversal logarithmic signatures. In [MM92] a special version of this theorem is proven. This fact is used to build a trapdoor in the first potential public key cryptosystem.

Setup Alice chooses a finite group G and generates:

(1) a tame logarithmic signature η of G

(2) k exact-transversal logarithmic signatures $\theta_1, \ldots, \theta_k$, such that $\alpha = \theta_1 \cdot \ldots \cdot \theta_k$ is a wild logarithmic signature

Alice publishes her public key (α, η), keeping $\theta_1, \ldots, \theta_k$ as her private key.

Encryption If Bob wants to send a message $x \in \mathbb{Z}_{|G|}$ to Alice, he

(i) computes $c = \alpha\eta^{-1}(x)$ $(\in \mathbb{Z}_{|G|})$

(ii) sends c to Alice.

Decryption Alice knows the factorization of α and can therefore compute

$$x = \theta_k^{-1} \cdot \ldots \cdot \theta_1^{-1}(c).$$

In [GS02] González Vasco and Steinwandt use a partial inversion attack to show that logarithmic signatures contained in $\mathfrak{TNT}$ do not offer secure instances of MST_1 in general. Furthermore in [BGCS05] they construct tame families of logarithmic signatures for symmetric and alternating groups that are in the class $\mathfrak{TNT}$. In chapters 3 and 4 we study the question if wild logarithmic signatures exist in cyclic and elementary abelian groups.

The cryptosystem MST_3 combines logarithmic signatures and covers:

Public Key Encryption Scheme $\boldsymbol{MST_3}$

Let G be a finite non-abelian group with nontrivial center $\mathcal{Z}$ such that G does not split over $\mathcal{Z}$, i.e. there is no subgroup $H < G$ with $H \cap \mathcal{Z} = e_G$ such that $G = H \cdot \mathcal{Z}$. Assume also that $\mathcal{Z}$ is sufficiently large so that exhaustive search problems are computationally infeasible in $\mathcal{Z}$.

Setup Alice chooses a large group G as described above and generates:

(1) a tame logarithmic signature $\beta = [B_1, \ldots, B_s] := (b_{i,j})$ of type $(r_1, \ldots, r_s)$ for $\mathcal{Z}$

(2) a random cover $\alpha = [A_1, \ldots, A_s] := (a_{i,j})$ of the same type as β for a certain subset $\mathcal{J}$ of G such that $A_1, \ldots, A_s \subseteq G \setminus \mathcal{Z}$.

She then chooses elements $t_0, t_1, \ldots, t_s \in G \setminus \mathcal{Z}$ and computes:

(3) $\tilde{\alpha} = [\tilde{A}_1, \tilde{A}_2, \ldots, \tilde{A}_s]$, where $\tilde{A}_i = t_{i-1}^{-1} A_i t_i$ for $i = 1, \ldots, s$

(4) $\gamma = (h_{ij}) = (b_{ij}\tilde{a}_{ij})$.

Alice publishes her public key (α, γ), keeping $(\beta, (t_0, \ldots, t_s))$ as her private key.

Encryption If Bob wants to send a message $x \in \mathbb{Z}_{|\mathcal{Z}|}$ to Alice, he

(i) computes $y_1 = \tilde{\alpha}(x)$ and $y_2 = \gamma(x)$

(ii) sends (y_1, y_2) to Alice.

Decryption We have
$$\begin{aligned} y_2 &= \gamma(x) \\ &= \breve{\beta}(x) t_0^{-1} \breve{\alpha}(x) t_s \\ &= \breve{\beta}(x) t_0^{-1} y_1 t_s. \end{aligned}$$

With the private key Alice can therefore compute

$$x = \breve{\beta}^{-1}(y_2 t_s^{-1} y_1^{-1} t_0).$$

Lempken et al. propose to use the Suzuki 2-group of order 2^{2m} ($n \in \mathbb{N}$) for MST_3, see [Hig63, BH81]. In MST_3 the first part of the private key is a logarithmic signature of the center of the underlying group. The center of the Suzuki 2-group is an elementary abelian subgroup. In [MSvTZ08] Magliveras et al. show that the system is insecure if β is a logarithmic signatures where each block is a subgroup. Furthermore in [BCM09] Blackburn et al. show that MST_3 is not secure if logarithmic signatures are used that are derived from exact transversal logarithmic signatures via transformations $T2$ to $T5$.

In Chapter 4 we study logarithmic signatures for elementary abelian p-groups for an arbitrary prime p. We give a deeper insight into their structure and show that for $p = 2$ all logarithmic signatures of type $(4, \ldots, 4, 2, \ldots, 2)$ are derived from exact transversal logarithmic signatures via the transformations $T2$ to $T5$. Hence MST_3 is insecure if β does not have blocks at least of size 8.

2.3 The MOR cryptosystem

Earlier we recalled the discrete logarithm problem (DLP) which is the basis for several cryptographic schemes. The DLP is also used in the MOR cryptosystem which was introduced in 2001 by Paeng et al., see [P+01b]. The system's security is based on the hardness of the discrete logarithm problem in the group of inner automorphisms of a non-abelian group:

This group is denoted by G. It is a non-abelian finite group with a set of generators $\{\gamma_1, \ldots, \gamma_l\}$ for $l \in \mathbb{N}$. We assume that the *representation problem* in G is efficiently solvable, i.e. there exists a polynomial time algorithm that computes the representation of an element of G as a product of the generating elements.

For MOR we consider the group $Inn(G)$ of inner automorphisms of G. The elements of this group are of the form

$$\begin{array}{rccc} I_g: & G & \to & G \\ & x & \mapsto & gxg^{-1} \end{array}$$

for $g \in G$. Note that

$$(I_g)^a = I_{g^a} \ (a \in \mathbb{N}),$$

and that

$$I_g = id_G \Leftrightarrow g \in Z(G).$$

In the setting of the MOR cryptosystem the inner automorphism induced by an element $g \in G$ is represented as the set

$$I_g = \{I_g(\gamma_i) : \ 1 \leq i \leq l\},$$

where each $I_g(\gamma_i)$ is again represented as a product of the $\gamma_i{}'s$. With this implicit representation it is possible to hide the element g inside I_g. This is essential for MOR which is described as follows, see also [P$^+$01b, P$^+$01a].

Public Key Encryption Scheme: **MOR**

Setup Alice chooses a group G as described above. She then generates arbitrary elements $g \in G \backslash Z(G)$ and $a \in \mathbb{N}$, $a < ord(I_g)$ and computes:

(1) $I_{g^a} = \{I_{g^a}(\gamma_1), \ldots, I_{g^a}(\gamma_l)\}$

Alice publishes her public key $(I_g,\ I_{g^a})$, keeping a as her private key.

Encryption If Bob wants to send a message $m \in G \backslash Z(G)$ to Alice, he

(i) chooses an arbitrary private encryption exponent $b \in \mathbb{N}$

(ii) computes $(I_{g^a})^b = I_{g^{ab}}$, $\varphi = (I_g)^b$ and $E = I_{g^{ab}}(m)$

(iii) sends $(E,\ \varphi)$ to Alice.

Decryption Alice uses her private key a to compute

$$m = \varphi^{-a}(E) = I_{g^{-ab}}(E) = I_{g^{-ab}}(I_{g^{ab}}(m)).$$

It is clear that MOR is broken if we are able to compute the secret key a. Solving a discrete logarithm problem in $< I_g >$ for the instance $(I_g,\ I_{g^a})$ is one way to achieve this. In Chapter 5 we describe two attacks that use a different approach.

2.3.1 MOR based on the group $GL(n, q) \times_\theta \mathcal{H}$

Consider the following definition:

Definition 2.12. The semidirect product of two groups G and H is the set of tuples from $G \times H$ with a multiplication defined via a homomorphism $\theta : H \to Aut(G)$ as

$$(g_1, h_1) \cdot (g_2, h_2) = (g_1 \theta_{h_1}(g_2), h_1 h_2),$$

where $Aut(G)$ is the group of automorphisms of G. It is denoted by $G \times_\theta H$.

In [P+01b] Paeng et al. propose to use the semidirect product $SL(2, p) \times_\theta \mathbb{Z}_p$ as a group for an implementation of the MOR scheme. On this group MOR has been analyzed and found insecure by Tobias and Paeng et al., see [Tob03, Tob04a, Pae03]. Tobias describes two attacks that enable an attacker to determine the plaintext message up to one unknown variable without compromising the secret key. The techniques presented in their work use two properties:

A_1 The special type of the homomorphism $\theta : \mathbb{Z}_p \to Inn(SL(2, p))$ and the fact that the center of $SL(2, p)$ is (almost) trivial allow an immediate reduction of MOR on $SL(2, p) \times_\theta \mathbb{Z}_p$ to MOR on $SL(2, p)$.

A_2 In a MOR cryptosystem using the group $SL(2, p)$, a ciphertext is a conjugate of the plaintext, and thus both texts have the same eigenvalues. This is enough information to calculate two linear equations in the entries of these matrices. The additional (and efficient) computation of an element of the centralizer of the enciphering matrix allows an adversary to obtain a third linear equation. These equations relate the four unknown entries of the plaintext matrix up to only one unknown variable.

In view of these attacks we proposed to analyze MOR in a more general case for which the attacks described above do not work, see [Kor05, Kor08]. We consider groups of the type

$$GL(n, q) \times_\theta \mathcal{H}$$

where $q = p^m$ is a prime power, $n \in \mathbb{N}_{>1}$, $\mathcal{H}$ is any finite abelian group, and θ is an efficiently computable homomorphism of $\mathcal{H}$ into $Aut(GL(n, q))$. For our purpose, it is sufficient to assume that group operations in the semidirect product can be efficiently computed and the representation problem is efficiently solvable.

Remark 2.13 (Conjugation and exponentiation in semidirect products)**.** Let $G \times_\theta H$ be a semidirect product, and $(g,h),(m,s) \in G \times_\theta H$. Then the inverse of (g,h) is

$$(g,h)^{-1} = \left(\theta_{h^{-1}}(g^{-1}),\ h^{-1}\right).$$

And the conjugate of (m,s) by (g,h) is

$$\begin{aligned}(m,s)^{(g,h)} &= (g,h)(m,s)(g,h)^{-1} \\ &= (g,h)(m,s)(g\,\theta_{h^{-1}}(g^{-1}),\ h^{-1}) \\ &= (g\,\theta_h(m),\ hs)(\theta_{h^{-1}}(g^{-1}),\ h^{-1}) \\ &= (g\,\theta_h(m)\,\theta_{hs}(\theta_{h^{-1}}(g^{-1})),\ hsh^{-1}) \\ &= (g\,\theta_h(m)\,\theta_{hsh^{-1}}(g^{-1})),\ hsh^{-1}).\end{aligned}$$

The a-th power of (g,h) $(a \in \mathbb{N})$ is computed inductively as

$$(g,h)^a = (\prod_{i=0}^{a-1} \theta_{h^i}(g),\ h^a\,).$$

By choosing a group for MOR in this way, we avoid an easy reduction as in A_1 and also the attack A_2. In previous work we described attacks of this new version of MOR in four special cases. For the analysis we simply assumed to know a certain automorphism, $\theta_h \in Aut(GL(n,q))$, which is part of the encryption function, and also that θ_h is a basic automorphism, see Definition 2.14. In Chapter 5 we show that it is possible to extract the components of θ_h efficiently from a public key (I_g, I_{g^a}). Furthermore we define two attacks that allow an adversary to derive a key based on solving discrete logarithms in small extensions of $\mathbb{F}_q$. This key enables the adversary to compute an $\mathbb{F}_q$-multiple of the plaintext for any message even if θ_h is not a basic automorphism.

2.3.2 Automorphisms of $GL(n,q)$

In order to analyze MOR in the group $GL(n,q) \times_\theta \mathcal{H}$, we need to know more about the subgroup $\theta(\mathcal{H})$ of $Aut(GL(n,q))$. In Theorem 4.25 of [Kor05] we give a detailed analysis of this group. The automorphism group of $GL(n,q)$ is a product of four subgroups, i.e.

$$Aut(GL(n,q)) = Inn(GL(n,q)) \cdot Aut_C(GL(n,q)) \cdot < \mathfrak{f} > \cdot < ct > .$$

We give a short overview of these four subgroups:

Inner automorphisms By $Inn(GL(n,q))$ we denote the group of inner automorphisms, which is non-abelian.

Central automorphisms Let ξ be the generator of the multiplicative group $\mathbb{F}_q^*$. Set

$$G_1 = \begin{pmatrix} \xi & & & 0 \\ & 1 & & \\ & & \ddots & \\ 0 & & & 1 \end{pmatrix}.$$

Note that G_1 and a matrix from $SL(n,q)$ generate $GL(n,q)$, see [Tay87].

The subgroup $Aut_c(GL(n,q))$ is the center of $Aut(GL(n,q))$. A central automorphism φ is an automorphism of $GL(n,q)$ such that for all $A \in GL(n,q)$

$$\varphi(A) \in A \cdot Z(GL(n,q)).$$

For each $1 \leq i \leq q-1$, with $in+1$ and $q-1$ coprime, there exists a central automorphism φ_i, such that

$$\varphi_i(G_1) = \xi^i \cdot G_1.$$

This property sufficiently describes all central automorphisms because a central automorphism is trivial on $SL(n,q)$. For $k \in \mathbb{N}$ we have

$$\varphi_i^k = \varphi_{s_i(k)},$$

where $s_i(k) := i \sum_{j=0}^{k-1} (in+1)^j$.

Field automorphisms The group $Aut(\mathbb{F}_q)$ is cyclic of order m and generated by the Frobenius automorphism

$$\begin{aligned} f : \mathbb{F}_q &\longrightarrow \mathbb{F}_q \\ x &\longmapsto x^p. \end{aligned}$$

By applying f to the entries a_{ij} of a matrix A, it induces a *field automorphism* $\mathfrak{f} \in Aut(GL(n,q))$ via

$$\begin{aligned} \mathfrak{f} : GL(n,q) &\longrightarrow GL(n,q) \\ A &\longmapsto (f(a_{ij}))_{i,j=1,\ldots,n}. \end{aligned}$$

The group of all automorphisms of $GL(n,q)$ induced by an automorphism of $\mathbb{F}_q$ is denoted by $< \mathfrak{f} >$ and of order m.

Contragredient automorphism The involution

$$\begin{aligned} ct:\ GL(n,q) &\longrightarrow GL(n,q) \\ A &\longmapsto (A^t)^{-1}, \end{aligned}$$

is called contragredient transformation. It generates a cyclic subgroup $< ct >$ of order 2 in $Aut(GL(n,q))$.

Note that for $n = q = 2$ $ct = I_{\left(\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right)}$ actually is an inner automorphism and for $n = 2$ and $q > 2$ it is in $Inn(GL(2,q)) \times Aut_C(GL(2,q))$.

Definition 2.14. A *basic automorphism* is an automorphism of $GL(n,q)$ which is either an inner, a central, a field or the contragredient automorphism, i.e. it is not a product of different types of automorphisms of $GL(n,q)$.

Commutativity among basic automorphisms For the attacks in Chapter 5 we consider certain combinations of automorphisms. For the analysis we need the following results on commutativity among basic automorphisms. First note that $Aut_c(GL(n,q))$ is the center of $GL(n,q)$ and thus central automorphisms commute with all other automorphisms. Furthermore let $\mathfrak{f}^r \in < \mathfrak{f} >$, then

$$\mathfrak{f}^r(A^{-1}) = \mathfrak{f}^r(A)^{-1} \text{ and } \mathfrak{f}^r(A^t) = \mathfrak{f}^r(A)^t$$

for all $A \in GL(n,q)$. It follows that

$$\mathfrak{f}^r \circ ct = ct \circ \mathfrak{f}^r.$$

The actual analysis of MOR on the group $GL(n,q) \times_\theta \mathcal{H}$ in a generalised setting is given in Chapter 5.

Chapter 3

Logarithmic signatures in cyclic p-groups

In this chapter we study logarithmic signatures of finite cyclic p-groups for a prime p. We give a complete characterization for logarithmic signatures in these groups. From this characterization we develop an efficient generic factoring algorithm. We consider finite cyclic p-groups because these are easy to implement, have a short bit-representation of elements, and are building blocks of several other (non-abelian) groups, e.g. the dihedral groups.

For a prime number p and $n \in \mathbb{N}$ the finite cyclic p-group $\mathbb{Z}_{p^n}$ is the set $\{0, 1, \ldots, p^n - 1\}$ with addition modulo p^n as the group operation. In the following we consider only additive cyclic p-groups, but the results also apply to any multiplicatively represented cyclic p-group:

Let $G_n = < g_n >$ be any cyclic p-group of order p^n ($n \in \mathbb{N}$). Furthermore let

$$\alpha_n = [A_1^n, \ldots, A_{s_n}^n],$$

with $A_i^n = [g_n^0, g_n^{a_{i,2}^n}, \ldots, g_n^{a_{i,r_i}^n}]$ be an arbitrary normalized logarithmic signature for G_n. Then the efficiency parameters from Remark 2.4 fulfill

$$r_n \geq p, \ \mu_n \geq n \log p, \ s_n \leq n.$$

Therefore we consider all computations in time polynomial in p and n to be efficient. Note that all elements in α_n are powers of g_n. The Silver-Pohlig-Hellman algorithm for computing discrete logarithms (see [PH78]) can be used to extract all $\ell(\alpha_n)$ exponents of the elements of α_n and receive

$$\beta_n = [B_1^n, \ldots, B_{s_n}^n],$$

with $B_i^n = [0, a_{i,2}^n, \ldots, a_{i,r_i}^n]$, a logarithmic signature of $\mathbb{Z}_{p^n}$. The running time of the Silver-Pohlig-Hellmann algorithm depends on the smallest prime divisor of the group order, more precisely it is $O(n^2 \log p + \sqrt{p})$ which is efficient. Therefore computing factorizations with respect to α is polynomial-time equivalent to computing factorizations with respect to β. And we can focus our analysis on the additive group $\mathbb{Z}_{p^n}$.

On a standard computer, a brute force algorithm that simply runs through all possible factorizations given by a logarithmic signature of $\mathbb{Z}_{p^n}$ with s blocks needs on average $\frac{1}{2}p^n \cdot (s-1)$ group operations plus $\frac{1}{2}p^n$ equality tests to find the correct factorization of a random element from $\mathbb{Z}_{p^n}$. As group operations and equality tests in $\mathbb{Z}_{p^n}$ can be done in time $O(n)$, the complexity of the brute force algorithm is

$$O(p^n \cdot n \cdot s),$$

which is exponential in p.

In the following we develop a factoring algorithm that runs in polynomial time.

In the first section we describe a reduction sequence that can be uniquely derived from an arbitrary logarithmic signature in any cyclic p-group. This reduction leads to an efficient factoring algorithm which we describe and give test results in the second section. To improve readability we omit the index n for logarithmic signatures and its blocks and elements.

3.1 Complete p-reduction

The group $\mathbb{Z}_{p^n}$ contains exactly $n-1$ nontrivial subgroups. These form the subgroup chain

$$\{0\} \leq < p^{n-1} > \leq < p^{n-2} > \leq \ldots \leq < p^2 > \leq < p > \leq \mathbb{Z}_{p^n}.$$

In particular all nontrivial subgroups of $\mathbb{Z}_{p^n}$ contain the smallest subgroup $< p^{n-1} >$.

Recall that a subset A of a finite group G is called periodic if it is the union of cosets of a subgroup H of G, i.e. $A = H + M$ for some subset M in G and M is called the set of periods. We deduce from Proposition 6.2.1 in [Sza04] that in logarithmic signatures of finite cyclic p-groups at least one block is periodic.

This leads to the following observation: If the logarithmic signature α of $\mathbb{Z}_{p^n}$ is normalized, it is clear that the periodic block contains a non-trivial subgroup. Furthermore none of the other blocks contains a subgroup. Otherwise

two blocks would contain the unique smallest subgroup $< p^{n-1} >$ which is impossible because of the uniqueness of the factorization of a logarithmic signature. Therefore it is easy to identify the periodic block as the only block that contains the element p^{n-1}.

Let A_i be the periodic block of α. Then A_i is of the form

$$A_i = A_i' + < p^{n-1} > .$$

Note that for all $a \in A_i$ and $k \in \{0, \ldots, p-1\}$ we have

$$a + k \cdot p^{n-1} \in A_i.$$

Therefore we may set

$$A_i' = \{a \in A_i : \ a < p^{n-1}\}.$$

By *splitting* A_i we derive a logarithmic signature

$$\alpha' = [A_1, \ldots, A_{i-1}, A_i', \ < p^{n-1} >, A_{i+1}, \ldots, A_s]$$

of the group $\mathbb{Z}_{p^n}$ and of type $(r_1, \ldots, r_{i-1}, r_i/p, p, r_{i+1}, \ldots, r_s)$, where r_i/p may be equal to one (then $A_i' = [0]$).

Note that the block A_i' could also be trivial. In order to keep the notation coherent we allow trivial blocks (in contrast to Definition 2.2.1). Furthermore it could also be the case that we can split off a larger subgroup than $< p^{n-1} >$ from block A_i but for our analysis we only consider the case in which we split off the smallest subgroup.

Proposition 3.1. *The logarithmic signature α' is tame if and only if α is tame.*

Proof. Note that α is only one T5 transformation of α' and we have seen in Chapter 2 that the tameness of one of these two induces the tameness of the other. □

Consider the logarithmic signature

$$\alpha' = [A_1, \ldots, A_{i-1}, A_i', \ < p^{n-1} >, A_{i+1}, \ldots, A_s]$$

of $\mathbb{Z}_{p^n}$. For $k \neq i$ let

$$A_k' = A_k \mod p^{n-1}$$

and consider the sequence

$$\alpha_1 = [A'_1, A'_2, \ldots, A'_{i-1}, A'_i, A'_{i+1}, \ldots, A'_s].$$

All elements in α_1 are smaller than p^{n-1} and we can view these as elements of $\mathbb{Z}_{p^{n-1}}$.

Proposition 3.2. *The sequence α_1 is a (normalized) logarithmic signature of $\mathbb{Z}_{p^{n-1}}$.*

Proof. We know that

$$A_1 + \ldots + A_{i-1} + A'_i + A_{i+1} + \ldots + A_s + \ < p^{n-1} > = \mathbb{Z}_{p^n}.$$

Which implies that $A_1 + \ldots + A_{i-1} + A'_i + A_{i+1} + \ldots + A_s$ is a transversal of $< p^{n-1} >$ in $\mathbb{Z}_{p^n}$. And it follows that

$$A_1 + \ldots + A_{i-1} + A'_i + A_{i+1} + \ldots + A_s \bmod p^{n-1} = \{0, 1, 2, \ldots, p^{n-1} - 1\}.$$

And thus

$$A'_1 + \ldots\ldots + A'_s = \mathbb{Z}_{p^{n-1}}.$$

Since also $\prod_{i=j}^{s} |A'_j| = p^{n-1}$, the sequence α_1 defines a logarithmic signature of $\mathbb{Z}_{p^{n-1}}$. □

Now we have two logarithmic signatures: α for the group $\mathbb{Z}_{p^n}$ and α_1 for $\mathbb{Z}_{p^{n-1}}$, where the first yields the second via efficient computations:

Proposition 3.3. *The logarithmic signature α_1 is tame if and only if α is tame.*

Proof. We already proved that α is tame if and only if α' is tame. Now we consider α' and α_1. W.l.o.g. let $i = 1$, i.e. we consider the two logarithmic signatures

$$\alpha' = [A'_1, \ < p^{n-1} >, A_2, \ldots, A_s]$$

and $\alpha_1 = [A'_1, A'_2, \ldots, A'_s]$. Let $g \in \mathbb{Z}_{p^n}$

A factorization of g in terms of α'

$$g = a_1 + h + a_2 + \ldots + a_s$$

with $a_1 \in A'_1$, $a_j \in A_j$ (for $j = 2, \ldots, s$) and $h \in < p^{n-1} >$ immediately yields the factorization

$$g \bmod p^{n-1} \ = \ (a_1 \bmod p^{n-1}) + (a_2 \bmod p^{n-1}) + \ldots + (a_s \bmod p^{n-1})$$

in terms of α_1. To compute the factorization we need s modulo operations and $s-1$ group operations.

For the reverse problem: If

$$(a_1, \ldots, a_s)$$

is a factorization of g mod p^{n-1} in terms of α_1, then

$$(b, b_1, b_2, \ldots, b_s)$$

is a factorization of g in terms of α', where for $j = 1, \ldots, s$

$$b_j \in A_j \text{ such that } b_j \bmod p^{n-1} = a_j.$$

Finally

$$b = (g - b_1 - \ldots - b_s) \bmod p^n.$$

All computations are efficient: s group operations as well as at most $\ell(\alpha')$ modulo operations and equality tests are needed. □

Definition 3.4. If α_1 is derived from α via the process described above, then we say that α_1 is a *p-reduction* of α and write $\alpha \to_p \alpha_1$.

The *differing elements* of a p-reduction are the elements of α_1 that are not elements of α, i.e. the elements that changed in the modulo reduction.

Definition 3.5. For any logarithmic signature α of a cyclic p-group $\mathbb{Z}_{p^n}$, we can apply the p-reduction and receive a polynomial-time equivalent but shorter logarithmic signature α_1 of the cyclic p-group $\mathbb{Z}_{p^{n-1}}$. Then n iterations of p-reductions yield the unique *complete p-reduction*

$$\alpha \to_p \alpha_1 \to_p \alpha_2 \to_p \ldots \to_p \alpha_n,$$

where

- α_i is some logarithmic signature of $\mathbb{Z}_{p^{n-i}}$ for $i = 1, \ldots, n-1$,
- α_{n-1} is composed of $s-1$ trivial blocks and one block $[0, 1, \ldots, p-1]$.
- α_n is composed of s trivial blocks, i.e. $[[0], \ldots, [0]]$,

Note that in each p-reduction exactly one block is split and all other blocks are reduced modulo the power of p corresponding to the subgroup that is split off. In the first round we split off the subgroup $< p^{n-1} >$ (in $\mathbb{Z}_{p^n}$) from some block that depends on the logarithmic signature. In the second round we split

of the subgroup $< p^{n-2} >$ (in $\mathbb{Z}_{p^{n-1}}$) again from a block that depends on the logarithmic signature, and so on. Since there is always exactly one block per step that is periodic and therefore "splitable", the parameter indicating which subgroup is contained in which block is an invariant of the logarithmic signature.

Definition 3.6. Let $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signatures of $\mathbb{Z}_{p^n}$. The *subtype* of α is the n-tuple

$$st(\alpha) := (k_1, \ldots, k_n) \in \{1, \ldots, s\}^n$$

such that $p^{n-i} \in A_{k_i} \bmod p^{n-i+1}$. In other words, the p-reduction $\alpha_{i-1} \to_p \alpha_i$ splits block k_i. Note that each index $1, \ldots, s$ of a block occurs exactly $\log_p |A_i|$ times in $st(\alpha)$.

We define the *plain logarithmic signature* $\alpha_{plain} = [B_1, \ldots, B_s]$ *corresponding to the subtype* $(k_1, \ldots, k_n)$ via the blocks

$$B_i := \sum_{k_j = i} [0, p^{n-j}, \ldots, (p-1)p^{n-j}]$$

for $i = 1, \ldots, s$.

The difference between the plain logarithmic signature α_{plain} and any other logarithmic signature α corresponding to a fixed subtype is the following: If we apply the p-reduction to α_{plain}, then we split one block and the other blocks do not change in the modulo reduction. For any other logarithmic signature α there is always at least one element that changes in some modulo reduction step.

To clarify the situation we give an extensive example.

Example 3.7. Our toy example is a normalized logarithmic signature $\alpha = [A_1, A_2, A_3]$ of the group $\mathbb{Z}_{2^9}$ with the subtype $(1, 2, 2, 1, 3, 1, 3, 2, 3)$ where

$$\begin{aligned} A_1 &= [0, 72, 96, 232, 256, 328, 352, 488], \\ A_2 &= [0, 30, 64, 158, 192, 222, 350, 384], \\ A_3 &= [0, 19, 67, 356, 404, 423, 464, 471]. \end{aligned}$$

We look at the complete p-reduction for α and α_{plain} as well as the factorization of 71 with respect to both logarithmic signatures. In α exactly one block, A_1, contains the element $2^{9-1} = 256$ and also the subgroup $[0, 256]$ of $\mathbb{Z}_{2^9}$.

It is periodic of size 2^3. If we take the 2^{3-1} smallest elements, i.e. $A'_1 = [0, 72, 96, 232]$, then we have

$$A_1 = [0, 256] + A'_1 = [0, 256] + [0, 72, 96, 232].$$

Now we reduce the other blocks A_2 and A_3 modulo $2^8 = 256$ and derive three blocks

$$\begin{aligned} A'_1 &= [0, 72, 96, 232], \\ A'_2 &= [0, 30, 64, 158, 192, 222, 94, 128], \\ A'_3 &= [0, 19, 67, 100, 148, 167, 208, 215], \end{aligned}$$

which form the logarithmic signature $\alpha_1 = [A'_1, A'_2, A'_3]$ of $\mathbb{Z}_{2^8}$ by Proposition 3.3. The differing elements are $94, 128$ in the second block and 100, 148, 167, 208, 215 in the third block. Table 3.1 on page 36 illustrates the 9 iterations of the complete p-reduction.

The plain logarithmic signature $\alpha_{plain} = [B_1, B_2, B_3]$ corresponding to the subtype $(1, 2, 2, 1, 3, 1, 3, 2, 3)$ is

$$\begin{aligned} B_1 &= [0, 2^8] + [0, 2^5] + [0, 2^3] = [0, 8, 32, 40, 256, 264, 288, 296], \\ B_2 &= [0, 2^7] + [0, 2^6] + [0, 2^1] = [0, 2, 64, 66, 128, 130, 192, 194], \\ B_3 &= [0, 2^4] + [0, 2^2] + [0, 2^0] = [0, 1, 4, 5, 16, 17, 20, 21]. \end{aligned}$$

The structure of the Table 3.2 on page 37 illustrating the p-reduction for the plain logarithmic signature is simple. There are no differing elements which is characteristic for plain logarithmic signatures.

A factorization for any $x = \sum_{i=0}^{8} c_i 2^i \in \mathbb{Z}_{2^9}$ is simply computed as

$$\begin{aligned} &[c_8 \cdot 2^8 + c_5 \cdot 2^5 + c_3 \cdot 2^3, \\ &\ c_7 \cdot 2^7 + c_6 \cdot 2^6 + c_1 \cdot 2^1, \\ &\ c_4 \cdot 2^4 + c_2 \cdot 2^2 + c_0 \cdot 2^0\ ] \end{aligned}$$

For example $71 = 64 + 4 + 2 + 1 = 0 + 66 + 5$.

$st(\alpha)$	A_1	A_2	A_3
	[0, 72, 96, 232, 256, 328, 352, 488]	[0, 30, 64, 158, 192, 222, 350, 384]	[0, 19, 67, 356, 404, 423, 464, 471]
1	[0, 256] + [0, 72, 96, 232]	[0, 30, 64, 158, 192, 222, 94, 128]	[0, 19, 67, 100, 148, 167, 208, 215]
2	[0, 72, 96, 104]	[0, 128] + [0, 30, 64, 94]	[0, 19, 67, 100, 20, 39, 80, 87]
2	[0, 8, 32, 40]	[0, 64] + [0, 30]	[0, 19, 3, 36, 20, 39, 16, 23]
1	[0, 32] + [0, 8]	[0, 30]	[0, 19, 3, 4, 20, 7, 16, 23]
3	[0, 8]	[0, 14]	[0, 16] + [0, 3, 4, 7]
1	[0, 8] + [0]	[0, 6]	[0, 3, 4, 7]
3	[0]	[0, 2]	[0, 4] + [0, 3]
2	[0]	[0, 2] + [0]	[0, 1]
3	[0]	[0]	[0, 1] + [0]
	[0]	[0]	[0]

Table 3.1: This table illustrates the complete p-reduction for our toy example.
From row i to row $i+1$ the block indicated at the left is split (for $i = 1, \ldots, 9$). The other two blocks are reduced modulo 2^{10-i}. We underlined the differing elements.

Example factorization: The elements involved in the factorization of $71 = 96 + 64 + 423 \bmod 512$ are highlighted.

$st(\alpha)$	A_1	A_2	A_3
	[0 , 8, 32, 40, 256, 264, 288, 296]	[0, 2, 64, 66 , 128, 130, 192, 194]	[0, 1, 4, 5 , 16, 17, 20, 21]
1	[0 , 256] + [0, 8, 32, 40]	[0, 2, 64, 66 , 128, 130, 192, 194]	[0, 1, 4, 5 , 16, 17, 20, 21]
2	[0 , 8, 32, 40]	[0, 128] + [0, 2, 64, 66]	[0, 1, 4, 5 , 16, 17, 20, 21]
2	[0 , 8, 32, 40]	[0, 64] + [0, 2]	[0, 1, 4, 5 , 16, 17, 20, 21]
1	[0 , 32] + [0 , 8]	[0, 2]	[0, 1, 4, 5 , 16, 17, 20, 21]
3	[0 , 8]	[0, 2]	[0, 16] + [0, 1, 4, 5]
1	[0 , 8]	[0, 2]	[0, 1, 4, 5]
3	[0 , 8] + [0]	[0, 2]	[0, 4] + [0, 1]
2	[0]	[0 , 2] + [0]	[0, 1]
3	[0]	[0]	[0 , 1] + [0]
	[0]	[0]	[0]

Table 3.2: This table illustrates the complete p-reduction for the plain logarithmic signature corresponding to our toy example. From row i to row $i+1$ the block indicated at the left is split (for $i = 1, \ldots, 9$). Note that there are no differing elements.

Example factorization: The elements involved in the factorization of $71 = 0 + 66 + 5 \bmod 512$ are highlighted.

3.2 Efficient factoring algorithm

Factoring with respect to a plain logarithmic signature is simple: Given only the subtype $(k_1, \ldots, k_n)$ a factorization for any

$$x = \sum_{i=0}^{n-1} c_i p^i \in \mathbb{Z}_{p^n}$$

is computed as

$$\sum_{i=1}^{s} \prod_{k_j=i} c_{n-j} p^{n-j}.$$

Here the subtype yields all the information necessary to compute factorizations.

For an arbitrary logarithmic signature α the situation is slightly different as we need to consider the differing elements. The logarithmic signature α yields a unique complete p-reduction from which we compute a factorization as follows:

Any block of a logarithmic signature in cyclic p-groups is determined by the order of splits and modulo reductions in the n steps of the complete p-reduction. The idea for factoring runs backwards through this p-reduction. We start with the *partial factorization* $[f_1, \ldots, f_s] = [0, \ldots, 0]$ corresponding to α_n. To factorize $g \in \mathbb{Z}_{p^n}$ we set $f_{k_n} = g \bmod p$. Note that $g \bmod p$ is an element of block k_n of α_{n-1}. We lift the partial factorization to $\mathbb{Z}_{p^2}$ using the inversion of the p-reduction $\alpha_{n-2} \to_p \alpha_{n-1}$:

(1) Only if f_{k_n} is a difference element we change it into the corresponding element $f_{k_n} + k \cdot p$, where $k \in \{1, \ldots, p-1\}$ is determined by the p-reduction.

(2) Then $f_{k_{n-1}} = (g - f_{k_n}) \bmod p^2$.

The partial factorization is lifted to $\mathbb{Z}_{p^3}$ and further until we derive a solution in $\mathbb{Z}_{p^n}$. In each following round i all elements of the current partial factorization that are differing elements are lifted from α_{n-i} to α_{n-i+1} and then we set

$$f_{k_i} = f_{k_i} + (g - (f_1 + \ldots + f_s)) \bmod p^i.$$

Note that for this process it is essential that we have access to the complete p-reduction of α. Table 3.3 illustrates the process of factoring 71 for our toy example (see highlighted elements in Table 3.1).

	$st(\alpha)$	modulo computations and differing elements	$\Rightarrow$	partial factorization
α_9		initialize $[f_1, f_2, f_3]$	$\Rightarrow$	$[0, 0, 0]$
α_8	3	$71 \bmod 2^1 = 1$	$\Rightarrow$	$[0, 0, 1]$
α_7		$\underline{1} \to 3$	$\Rightarrow$	$[0, 0, 3]$
	2	$(71 - 3) \bmod 2^2 = 0$	$\Rightarrow$	$[0, 0, 3]$
α_6			$\Rightarrow$	$[0, 0, 3]$
	3	$(71 - 3) \bmod 2^3 = 4$	$\Rightarrow$	$[0, 0, 7]$
α_5			$\Rightarrow$	$[0, 0, 7]$
	1	$(71 - 7) \bmod 2^4 = 0$	$\Rightarrow$	$[0, 0, 7]$
α_4			$\Rightarrow$	$[0, 0, 7]$
	3	$(71 - 7) \bmod 2^5 = 0$	$\Rightarrow$	$[0, 0, 7]$
α_3		$\underline{7} \to 39$	$\Rightarrow$	$[0, 0, 39]$
	1	$(71 - 39) \bmod 2^6 = 32$	$\Rightarrow$	$[32, 0, 39]$
α_2		$\underline{32} \to 96$	$\Rightarrow$	$[96, 0, 39]$
	2	$(71 - 96 - 39) \bmod 2^7 = 64$	$\Rightarrow$	$[96, 64, 39]$
α_1		$\underline{39} \to 167$	$\Rightarrow$	$[96, 64, 167]$
	2	$(71 - 96 - 64 - 167) \bmod 2^8 = 0$	$\Rightarrow$	$[96, 64, 167]$
α		$\underline{167} \to 423$	$\Rightarrow$	$[96, 64, 423]$
	1	$(71 - 96 - 64 - 423) \bmod 2^9 = 0$	$\Rightarrow$	$[96, 64, 423]$

Table 3.3: Computing the factorization of 71 for Example 3.7 from the complete p-reduction in Table 3.1.

A factoring algorithm for logarithmic signatures in $\mathbb{Z}_{p^n}$ that uses this idea has two stages. The first stage (*preprocessing*) computes the complete p-reduction including the subtype. The second stage computes the actual factorization for any element in $\mathbb{Z}_{p^n}$. Besides the subtype the only information from the p-reduction that is necessary are those about differing elements, i.e. 7 changed to 39 by 2^5. All other information in the complete p-reduction is redundant. We developed two different approaches to give a compact description of the complete p-reduction:

(v1) Consider one p-reduction from α_i to α_{i+1}. Note that for $p = 2$ the differing elements change by 2^{n-i}. If $p > 2$, then the differing elements change by p^{n-i}, $2 \cdot p^{n-i}, \ldots,$ or $(p-1)p^{n-i}$. We keep count on this information by

storing the differing elements in one of $p-1$ lists per round, i.e. if $a = a \bmod p^{n-i} + k \cdot p^{n-i}$ then a is stored in list k.

In every round of a factoring algorithm we simply check whether the elements of the current partial factorization $[f_1, \ldots, f_s]$ are in one of the $p-1$ lists. If f_j is in list k in round i, then we lift f_j to $f_j + k \cdot p^{n-i}$ and finally

$$f_{k_{n-i}} = f_{k_{n-i}} + (g - (f_1 + \ldots + f_s)) \bmod p^i.$$

In this case we have to store $n(p-1)$ lists which overall contain less elements than the $n \cdot s$ lists of the complete p-reduction. The Table 3.4 shows the lists of the compact p-reduction (version v1) deduced from Table 3.1. As $p = 2$ there is only one list per round.

$st(\alpha)$	difference elements
1	$[0, 72, 96, 232, 256, 328, 352, 488]$, $[0, 30, 64, 158, 192, 222, 350, 384]$, $[0, 19, 67, 356, 404, 423, 464, 471]$
2	$[94, 128, 100, 148, \boxed{167}, 208, 215]$
2	$[20, \boxed{39}, 80, 87, 104]$
1	$[3, 8, 16, 23, \boxed{32}, 36, 40]$
3	$[4, \boxed{7}]$
1	$[14]$
3	$[6]$
2	$[2]$
3	$[\boxed{1}]$

Table 3.4: v1-description of the compact p-reduction for Example 3.7. Each row only contains the differing elements. The elements involved in the factorization of $71 = 96 + 64 + 423 \bmod 512$ are highlighted.

(v2) Another compact way to describe a complete p-reduction consists of $p-1$ lists for each block per round plus one permutation p_i per round.

The elements of the lists are now the positions of the differing elements in their blocks, i.e. if $a_j = a_j \bmod p^{n-i} + k \cdot p^{n-i}$, then j is stored in list k for block this block.

Again we start at the end of the p-reduction with the trivial partial factorization indicating positions $[1, \ldots, 1]$. Additionally we set a counter $c := g$, where g is to be factored.

In every round of a factoring algorithm we simply check whether the positions in the current partial factorization $[f_1, \ldots, f_s]$ are in one of the $p-1$ lists. If f_j is in list k of block j in round i then we subtract $k \cdot p^{n-i}$ from c ($j \neq k_{n-i}$). In a second step we set

$$f_{k_{n-i}} = p_i(f_{k_{n-i}} \cdot ((c \bmod p^{n-i+1})/p^{n-i} + 1)).$$

In this case we have to store $n \cdot s \cdot (p-1)$ lists which overall contain less elements than the $n \cdot s$ lists of the complete p-reduction.

We illustrate the v2-description with an example and more detail. Consider round 2 of the complete p-reduction of Example 3.7. The first block changes from

$$[\ 0, 72, 96, 232\ ]$$

to

$$[\ 0, 72, 96, 104\ ]$$

with one differing element at position 4. The third block changes from

$$[\ 0, 19, 67, 100, 148, 167, 208, 215\ ]$$

to

$$[\ 0, 19, 67, 100, 20, 39, 80, 87\ ]$$

where the differing elements are at positions 5, 6, 7 and 8. The second block $[0, 30, 64, 158, 192, 222, 94, 128]$ splits into

$$[0, 128] + [0, 30, 64, 94] = [0, 30, 64, 94, 128, 158, 192, 222].$$

The permutation $(4, 6, 8, 5, 7)$ describes the difference between those two blocks. We have the following parameters: The counter is $c := -64$ and the current partial factorization is $[f_1, f_2, f_3] = [3, 3, 6]$ and $k_{9-2} = 2$. First consider f_1 and f_3. $f_1 = 3$ is not a position of a difference element, but $f_3 = 6$ is the position of a difference element for block 3. There we set $c := -64 - 2^7 = -196$. Since now $c \bmod 2^8 = 64$ we set $f_2 = (4, 6, 8, 5, 7)(3 \cdot (0 + 1)) = 3$. Table 3.5 shows the lists of the compact p-reduction (version v2) deduced from Table 3.1. As $p = 2$ there is only one sublist per block and round.

$st(\alpha)$	positions of difference elements	permutations
	$[0, 72, 96, 232, 256, 328, 352, 488]$, $[0, 30, 64, 158, 192, 222, 350, 384]$, $[0, 19, 67, 356, 404, 423, 464, 471]$	
1	$[[]],\ [[7, 8]],\ [[4, 5, \boxed{6}, 7, 8]]$	$()$
2	$[[4]],\ [[]],\ [[5, \boxed{6}, 7, 8]]$	$(4, 6, 8, 5, 7)$
2	$[[2, \boxed{3}, 4]],\ [[]],\ [[3, 4, 7, 8]]$	$()$
1	$[[]],\ [[]],\ [[4, \boxed{6}]]$	$()$
3	$[[]],\ [[2]],\ [[]]$	$(2, 6, 4, 3)(5, 7)$
1	$[[]],\ [[2]],\ [[]]$	$()$
3	$[[]],\ [[2]],\ [[]]$	$()$
2	$[[]],\ [[]],\ [[\boxed{2}]]$	$()$
3	$[[]],\ [[]],\ [[]]$	$()$

Table 3.5: v2-description of the compact p-reduction for Example 3.7. Each row only contains the positions of the differing elements plus one permutation for the block that splits in that round. The elements involved in the factorization of $71 = 96 + 64 + 423 \bmod 512$ are highlighted.

We implemented preprocessing and factoring algorithms for $\mathbb{Z}_{p^n}$ in GAP. There are four versions, which can be found in the Appendix 6.1. The first version v1.0 is only for $p = 2$ and uses the v1-compact description of the complete p-reduction. The second version v1.1 is implemented for arbitrary primes and also uses the v1-compact description. Version v2.0 uses the v2-compact description of the complete p-reduction, where version v2.1 uses an improved v2-compact description that combines all permutations corresponding to one block.

Proposition 3.8. *In $\mathbb{Z}_{p^n}$ the running time of the factoring algorithms using descriptions v1 and v2 is polynomial in p, r_n and n.*

Proof. From the GAP-Code in the Appendix 6.1 we derive the following estimates for the worst case running time of the different algorithms.

algorithm	version	worst case running time
preprocessing	v1.0	$O(n \cdot (l + l^2 + n \cdot r_n))$
	v1.1	$O(n \cdot (l + l^2 + n \cdot r_n^2))$
	v2.0	$O(n \cdot (l + l^2 + n \cdot r_n^2))$
	v2.1	$O(n \cdot (l + l^2 + n \cdot r_n^2) + n(p + n \cdot p))$
factor	v1.0	$O(n^3 \cdot r_n)$
	v1.1	$O(n^3 \cdot p)$
	v2.0	$O(n^2 \cdot p + n \cdot r_n))$
	v2.1	$O(n^2 \cdot p + n)$

Table 3.6: Running times of the preprocessing and factoring algorithms for logarithmic signatures in $\mathbb{Z}_{p^n}$

Here l is the number of elements contained in the logarithmic signature, thus $l \leq n \cdot r_n$. With the result on Page 30 we see that independent of the representation all running times are polynomial in p, r_n and n. □

It follows

Theorem 3.9. *All logarithmic signatures in cyclic p-groups are tame.*

The algorithms were tested on an AMD Athlon(tm) 64 X2 Dual Core Processor 3800+, 2 x 1 GHz with 2 x 512 KB RAM. Tables 3.8, 3.9, and 3.10 give an overview on the running times of the algorithms and also compare the different implementations for $p = 2, 3$ and several group sizes. For these tests we computed pseudo-random logarithmic signatures via pseudo-randomly choosing a subtype and differing elements, see Section 6.1.

To compare to the running time of the proposed algorithms: Table 3.7 shows the average running time for a complete search for a factorization through a random logarithmic signature of type $(4, \ldots, 4)$ of a cyclic 2-group of order between 2^{20} and 2^{32} on the same computer. This brute force search for a factorization was only feasible for group sizes far below 2^{100}. Table 3.9 shows that the new algorithms need only 10 milliseconds for a factorization in groups of size 2^{100} and even in groups of size 2^{1000} a factorization can be found in less than a second in one case of v2-algorithm.

The different running times of the v1- and v2-algorithms listed in Table 3.6 also appear in the following tables. We see that the v1-algorithms are up to 20-times faster in the preprocessing phase compared to the v2-algorithms, but in the factoring phase the v2-algorithms are much faster by a factor of up to 90.

group size	time for complete search
2^{20}	3 sec
2^{22}	12.6 sec
2^{24}	52.2 sec
2^{26}	214 sec
2^{28}	897 sec
2^{30}	61 min
2^{32}	4,17 h
2^{40}	52.6 days*
2^{60}	0.22 m years*
2^{80}	331 bn years*
2^{100}	501 tn years*

Table 3.7: Average running times for a complete search of a logarithmic signature with blocks of size smaller than 4 for a factorization in the group $\mathbb{Z}_{2^{100}}$. 2 to 100 samples were generated to compute each value. The values marked with * are estimates and extrapolated from the previous.

group size	Algorithm v1.0	size of larges block				
		2^2	2^4	2^6	2^8	2^{10}
2^{100}	preprocessing	0,04	0,05	0,03	0,03	0,14
	factor	0,01	0,01	0,01	0,01	0,02
2^{200}	preprocessing	0,13	0,19	0,25	0,34	1
	factor	0,04	0,06	0,08	0,09	0,2
2^{400}	preprocessing	0,65	0,87	1	2,6	4,8
	factor	0,41	0,51	0,61	1,1	2,8
2^{600}	preprocessing	1,9	2	2,5	5,9	15,4
	factor	1,7	1,7	2,3	5,1	17,5
2^{800}	preprocessing	3,6	4,9	6,2	10,4	29,4
	factor	4,1	4,5	5,6	13,4	41,3
2^{1000}	preprocessing	7	8,4	10,7	21,9	39,5
	factor	8,4	8,9	13,5	35,9	87,5

group size	Algorithm v1.0	2^{12}	2^{14}	2^{16}	2^{18}	2^{20}	2^{22}
2^{100}	preprocessing	0,57	4,2	4,8	44,4	108,8	11 min
	factor	0,07	0,33	0,37	1	2,5	10,2

Table 3.8: Preprocessing and factoring for $p = 2$ with v1.0. For each combination of group and largest block size there are two values. The top indicates the average time (in seconds) needed for the preprocessing algorithm, the second value indicates the average time (in seconds) needed to factor a random element from the corresponding group. With the algorithm in Section 6.1.1 at least 10 and up to 10,000 samples were generated to compute each value in the table.

group size	Algorithm	v1.0	v1.1	v2.0	v2.1
2^{100}	preprocessing	0,04	0,02	0,02	0,02
	factor	0,01	0,01	0,01	0,01
2^{200}	preprocessing	0,13	0,06	0,1	0,12
	factor	0,04	0,11	0,05	0,04
2^{400}	preprocessing	0,65	0,45	0,69	0,86
	factor	0,41	1,2	0,26	0,26
2^{600}	preprocessing	1,9	1,1	1,8	2,2
	factor	1,7	4,1	0,61	0,37
2^{800}	preprocessing	3,6	2,5	3,9	4,5
	factor	4,1	9,4	1,1	0,62
2^{1000}	preprocessing	7	4,6	7,3	8,7
	factor	8,4	18,8	1,8	0,96

Table 3.9: Comparison of versions v1.0 to v2.1 of the preprocessing and factoring algorithms for $p = 2$ and largest block of size 4. For each combination of group and algorithm are two values. The top indicates the average time (in seconds) needed for the preprocessing algorithm, the second value indicates the average time (in seconds) needed to factor a random element from the corresponding group. With the algorithm in Section 6.1.1 at least 10 and up to 10,000 samples were generated to compute each value in the table.

	v1.1	v2.0	v2.1	v1.1	v2.0	v2.1	v1.1	v2.0	v2.1	v1.1	v2.0	v2.1
	3^2			3^4			3^6			3^8		
3^{100}	0,03	0,04	0,05	0,14	0,18	0,19	0,56	1,1	1,3	4,1	55,1	95,5
	0,03	0,01	0,01	0,07	0,02	0,02	0,16	0,012	0,009	0,95	0,012	0,012
3^{200}	0,22	0,3	0,38	0,7	0,88	9,8	3,8	5,9	5,4	17,3	308,3	304,8
	0,31	0,07	0,06	0,5	0,06	0,06	1,4	0,05	0,049	8,5	0,052	0,048
3^{300}	0,74	0,99	1,2	2,3	2	2,4	9,4	16,1	15,4	38,6	761	568,2
	1,1	0,2	0,2	1,6	1,3	1,3	8,8	0,102	0,106	35	0,11	0,14
3^{400}	1,6	1,9	2,2	4,2	5	4,8	17	30,9	34,8	70,4	1286,4	957,6
	2,6	0,37	0,36	3,9	0,23	0,24	20,7	0,19	0,19	93,1	0,19	0,18

Table 3.10: Comparison of versions v.1.1, v2.0 and v2.1 of the preprocessing and factoring algorithms for $p = 3$. The first column indicates the size of the group, the first row indicates which version of the algorithms is used, the second row indicates the size of the largest block. For each combination of group, largest block size and algorithm there are two values. The top indicates the average time (in seconds) needed for the preprocessing algorithm, the second value indicates the average time (in seconds) needed to factor a random element from the corresponding group. With the algorithm in Section 6.1.1 at least 10 and up to 10,000 samples were generated to compute each value in the table.

3.3 On the number of logarithmic signatures in cyclic p-groups

The characterization of logarithmic signatures in cyclic p-groups via the complete p-reduction yields a proposition on the exact number of logarithmic signatures for these groups. In one p-reduction there are two different steps. First one block is split and then every element (except 0) in the other blocks might be reduced by a multiple of a fixed power of p. Given the subtype it is clear which block is split in which round of the complete p-reduction. Now we count the possible modulo reductions.

Proposition 3.10. *For the group $\mathbb{Z}_{p^n}$ and the subtype $(k_1, \ldots, k_n)$ there exist exactly*

$$\prod_{i=1}^{s} \prod_{j=1}^{l_i} (p^{p^j-1})^{f(i,j)}$$

normalized logarithmic signatures, where

- *$s = \max\{k_i\}$ is the number of blocks*
- *$l_i = \sum_{k_j=i} 1$ is the p-logarithm of the block size*
- *$f(i,j)$ is the number of elements between the j-th and the $(j+1)$-th occurrence of i in the sequence $(k_n, \ldots, k_1)$ (reversed subtype). $f(i, l_i)$ is the number of elements after the last occurrence of i.*

Proof. Let α be a logarithmic signature of $\mathbb{Z}_{p^n}$ and subtype $(k_1, \ldots, k_n)$. First note that the modulo reductions in one block are independent of the structure of the other blocks. Therefore we consider block A_i for any $i \in \{1, \ldots, s\}$, count the number ν_i of possible structures for this block induced by the subtype.

Assume that the block A_i splits in the rounds m_1 to m_{r_i} of the complete p-reduction, i.e. $k_{m_1} = \ldots = k_{m_{r_i}} = i$.

We consider the complete p-reduction from bottom up. First A_i is the trivial block $[0]$. In round m_{r_i} it changes to $< p^{n-r_i} >$ mod $p^{m_{r_i}}$ and contains p elements.

Up to round $r_i - 1$ block A_i contains $p-1$ non-zero elements. In each round j of the reverse p-reduction ($j = 1, \ldots, n$) each element changes by one of the p elements $0, p^{n-j}, 2p^{n-j}, \ldots, (p-1)p^{n-j}$. Thus between round r_i and $r_i - 1$ there are $(p^{p-1})^{f(i,1)}$ possible changes.

In round $r_i - 1$ the size of the block increases to $p^2 - 1$ non-zero elements, thus there are $(p^{p-1})^{f(i,2)}$ possible changes.

This leads to

$$\nu_i = \prod_{j=1}^{l_i} (p^{p^j-1})^{f(i,j)}$$

possible blocks A_i. For the whole logarithmic signature we have

$$\prod_{i=1}^{s} \nu_i = \prod_{i=1}^{s} \prod_{j=1}^{l_i} (p^{p^j-1})^{f(i,j)}.$$

□

For Example 3.7 we have the subtype $(1, 2, 2, 1, 3, 1, 3, 2, 3)$ and the number of logarithmic signatures of the same type in $\mathbb{Z}_{2^9}$ is

$$\begin{aligned} \nu_1 \cdot \nu_2 \cdot \nu_3 &= (2^{2^1-1})^{f(1,1)} (2^{2^2-1})^{f(1,2)} (2^{2^3-1})^{f(1,3)} \\ &\quad \cdot (2^{2^1-1})^{f(2,1)} (2^{2^2-1})^{f(2,2)} (2^{2^3-1})^{f(2,3)} \\ &\quad \cdot (2^{2^1-1})^{f(3,1)} (2^{2^2-1})^{f(3,2)} (2^{2^3-1})^{f(3,3)} \\ &= (2^1 \cdot 2^6 \cdot 2^0) \cdot (2^4 \cdot 2^0 \cdot 2^7) \cdot (2^1 \cdot 2^3 \cdot 2^{28}) \\ &= 2^{50} \end{aligned}$$

In this chapter we gave a full characterization of logarithmic signatures in cyclic p-groups. By developing an efficient factoring algorithm we showed that all logarithmic signatures in cyclic p-groups are tame. Furthermore this is the first time that we can give the exact number of logarithmic signatures in a group.

Further research could focus on the MST_3-cryptosystem which uses tame logarithmic signatures in the center of the underlying group.

Chapter 4

Logarithmic signatures in elementary abelian p-groups

In this chapter we study the structure of logarithmic signatures in elementary abelian p-groups. Logarithmic signatures of these groups are employed by the cryptosystem MST_3. This system was shown to be insecure if the keys are derived from an exact transversal logarithmic signature via the transformations $T1$ to $T5$, see [MSvTZ08] and also [BCM09]. A different way of generating logarithmic signatures of these groups was not known.

Therefore we focus our research for this chapter on an extensive analysis of logarithmic signatures in elementary abelian p-groups. In the first section we study linear independence between the elements of a logarithmic signature. It follows a partial classification of logarithmic signatures in elementary abelian 2-groups and a polynomial-time factoring algorithm for a class of logarithmic signatures in these groups. In the third section we introduce a system of certain linear equations that can be derived from any logarithmic signature in elementary abelian p-groups. Furthermore we consider a class of logarithmic signatures in which we give a polynomial factoring algorithm based on this system of linear equations.

4.1 Ideally distributed basis

In [QV94] Qu and Vanstone proved the following theorem to analyze a cryptosystem using minimal logarithmic signatures proposed by Webb, see [Web91]: Let $\alpha = [A_1, \ldots, A_n]$ be a logarithmic signature of $\mathbb{Z}_p^n$ of type $(p, \ldots, p)$ and

$\mathcal{B} := \{b_1, \ldots, b_n\}$ a collection of n vectors such that one non-zero vector is selected from every block, i.e. $b_i \in A_i$ and $b_i \neq 0$ for $i \in \{1, \ldots, n\}$. Then $\mathcal{B}$ is a basis of $\mathbb{Z}_p^n$.

The proof relies on the fact that every block is of prime order, i.e. $|A_1| = \ldots = |A_n| = p$. In order to be able to analyze logarithmic signatures with blocks of prime power order, we develop a new approach. For the logarithmic signature α let the blocks be of order $|A_i| = p^{t_i}$. The number of blocks, s, is smaller than the number n of vectors in a basis of $\mathbb{Z}_p^n$. In order to assemble a basis from α, we would have to choose more than one vector from some blocks - according to the block sizes: Recall that $\sum_{i=1}^{s} t_i = n$. So, if a block is of order p^{t_i} then we take t_i vectors from this block and obtain a set of n vectors in total. The question, whether this approach is sufficient to obtain a basis, is immediately answered in the negative:

Example 4.1. Consider the logarithmic signature $\alpha = [A_1, A_2]$ from $\mathbb{Z}_3^3$ with two blocks

$$\begin{aligned} A_1 &= [\ (0,0,0),\ (1,0,0),\ (0,1,0)\ ] \\ A_2 &= [\ (0,0,0),\ (1,1,0),\ (2,2,0),\ (0,0,1), \\ &\quad\ (0,0,2),\ (1,1,1),\ (1,1,2),\ (2,2,1),\ (2,2,2)\ ] \end{aligned}$$

The set $\{(1,0,0),\ (1,1,0),\ (2,2,0)\}$ is chosen from α with a distribution 1:2 according to the block sizes, no vector is trivial, but the vectors are linearly dependent.

4.1.1 Existence of a basis in a logarithmic signature

In this example we see that a generalization of the theorem of Qu and Vanstone is not straightforward, and we need to determine if it is in general possible to find a basis of $\mathbb{Z}_p^n$ in a logarithmic signature and how to select the vectors for this basis. It is not immediately clear whether there always exists a basis that is distributed according to the block sizes, but a short proof shows that there always exists some basis of $\mathbb{Z}_p^n$ in α with some distribution.

Lemma 4.2. *Let p be a prime, $n \in \mathbb{N}$, and $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signature of the elementary abelian group $\mathbb{Z}_p^n$. Then there exists a basis $\mathcal{B}$ of $\mathbb{Z}_p^n$ such that*

$$\mathcal{B} \subseteq A_1 \cup \ldots \cup A_s.$$

We say that α contains the basis $\mathcal{B}$.

Proof. Verify that

$$\mathbb{Z}_p^n = A_1 + \ldots + A_s \subseteq < A_1 \cup \ldots \cup A_s > \subseteq \mathbb{Z}_p^n.$$

Thus $< A_1 \cup \ldots \cup A_s > = \mathbb{Z}_p^n$ and it follows that $A_1 \cup \ldots \cup A_s$ contains a basis of $\mathbb{Z}_p^n$ □

An algorithm that computes a basis in a logarithmic signature is polynomial in the length $\ell(\alpha)$ of the signature - as can be seen from the following algorithm that simply runs through all elements of α and adds an element to the set B if it is linearly independent of all elements that already are in B.

GAP: FindBasisLogSig

Input: logSig, a logarithmic signature of $\mathbb{Z}_p^n$, where p is prime and $n \in \mathbb{N}$
Output: B, a basis of $\mathbb{Z}_p^n$ contained in logSig

```
FindBasisLogSig:=function(logSig)

local i, a, B;
B:= [];

for i in [1..Length(logSig)] do
  for a in logSig[i] do

  Add(B, a);

    if Rank(B) < Length(B)
    then Remove(B,a);
    fi;

  od;
od;

return B;

end;
```

4.1.2 Distribution of a basis

Because of Lemma 4.2 this algorithm outputs a basis of $\mathbb{Z}_p^n$ contained in a logarithmic signature α, but with a distribution that does not necessarily reflect the block sizes. For the logarithmic signature in Example 4.1 the algorithm returns the set $\{(1,0,0),\ (0,1,0),\ (0,0,1)\}$. With two vectors from the first block and only one vector from the second block, it is not distributed according to the block sizes, in a ratio 1:2. In fact, the distribution of the basis given by the algorithm FindBasisLogSig is such that the basis contains $Rank(< A_1 >)$ vectors from the first block, $Rank(< A_1, A_2 >) - Rank(< A_1 >)$ vectors from the second block, and so on. If for example

$$Rank(< A_1, \ldots, A_{s-1} >) = n,$$

the basis returned by this algorithm does not contain a vector from the last block of α.

An example shows that it is not sufficient to change the algorithm such that once $\log(|A_i|)$ vectors are collected from a block A_i it skips running through the remaining vectors of that block: Take the logarithmic signature $\alpha = [A_1, A_2]$ from $\mathbb{Z}_3^4$ with the blocks

$$\begin{aligned} A_1 &= [\ (0,0,0,0),\ (1,0,0,0),\ (0,1,0,0),\ (0,0,0,1),\\ &\quad\ (1,0,0,1),\ (0,1,0,1),\ (0,0,0,2),\ (1,0,0,2),\ (0,1,0,2)\ ]\\ A_2 &= [\ (0,0,0,0),\ (1,1,0,0),\ (2,2,0,0),\ (0,0,1,0),\\ &\quad\ (0,0,2,0),\ (1,1,1,0),\ (1,1,2,0),\ (2,2,1,0),\ (2,2,2,0)\ ]. \end{aligned}$$

The modified algorithm would return the set

$$\{(1,0,0,0),\ (0,1,0,0),\ (0,0,1,0)\}$$

which is not a basis.

Regarding the distribution of a basis in a logarithmic signature we now study the questions

(1) Does there always exist a basis in a logarithmic signature that is distributed according to the block sizes?

(2) How do we find or compute such a basis?

In order to study this topic, we give definitions that describe how a basis is distributed in a logarithmic signature.

Definition 4.3. Let α be a logarithmic signature of $\mathbb{Z}_p^n$ and let $\mathcal{B}$ be a basis of $\mathbb{Z}_p^n$ contained in α. The *distribution* of $\mathcal{B}$ with respect to α is the vector

$$d_\alpha(\mathcal{B}) = (\ |A_1 \cap \mathcal{B}|,\ \ldots,\ |A_s \cap \mathcal{B}|\ ).$$

If

$$|A_i \cap \mathcal{B}| = \log_p(|A_i|)$$

for $i = 1, \ldots, s$, then we call $\mathcal{B}$ a basis of *ideal distribution* for α.

For the Example 4.1 we consider the basis $\mathcal{B}_1 = \{(1,0,0),\ (0,1,0),\ (0,0,1)\}$ and $\mathcal{B}_2 = \{(1,0,0),\ (1,1,0),\ (0,0,1)\}$ with the distribution

$$d_\alpha(\mathcal{B}_1) = (2,1),\ \text{and}\ d_\alpha(\mathcal{B}_2) = (1,2).$$

Then only $\mathcal{B}_2$ is a basis of ideal distribution for α.

Given any basis in a logarithmic signature α, we define a measure how close this basis is to a basis of ideal distribution for α.

Definition 4.4. Let α be a logarithmic signature of $\mathbb{Z}_p^n$, let $\mathcal{B}$ be a basis of $\mathbb{Z}_p^n$ contained in α, and set $t_i := \log_p(|A_i|)$. The *quality* of $\mathcal{B}$ with respect to α is the natural number that indicates the difference between the two vectors $d_\alpha(\mathcal{B})$ and $(t_1, \ldots, t_s)$, i.e.

$$q_\alpha(\mathcal{B}) = \sum_{i=1}^{s} |t_i - |A_i \cap \mathcal{B}||.$$

We say that a basis $\mathcal{B}$ *is better than* a basis $\mathcal{B}'$ with respect to α if

$$q_\alpha(\mathcal{B}) < q_\alpha(\mathcal{B}').$$

For the Example 4.1 we have

$$q_\alpha(\mathcal{B}_1) = |1-2| + |2-1| = 2$$

and

$$q_\alpha(\mathcal{B}_2) = |1-1| + |2-2| = 0.$$

Thus $\mathcal{B}_2$ is better than $\mathcal{B}_1$ with respect to α.

We immediately derive some properties of the quality function q_α:

Proposition 4.5. *Let α be a logarithmic signature and $\mathcal{B}$ be a basis of $\mathbb{Z}_p^n$ contained in α. Then*

1. *$q_\alpha(\mathcal{B})$ is even.*
2. *$0 \leq q_\alpha(\mathcal{B}) < 2n$*
3. *$q_\alpha(\mathcal{B}) = 0$ if and only if $\mathcal{B}$ is a basis of ideal distribution.*

Proof. 1. Let $t'_i = |A_i \cap \mathcal{B}|$. Recall that $t_i = \log_p(|A_i|)$. Since $\mathcal{B} \subseteq A_1 \cup \ldots \cup A_s$, we have

$$\sum_{i=1}^{s} t_i = \sum_{i=1}^{s} t'_i \ (= n).$$

And therefore

$$t_s - t'_s = -\sum_{i=1}^{s-1} t_i - t'_i.$$

It follows that

$$\begin{aligned} q_\alpha(\mathcal{B}) &= \sum_{i=1}^{s-1} |t_i - t'_i| + |t_s - t'_s| \\ &= \sum_{i=1}^{s-1} |t_i - t'_i| + |\sum_{i=1}^{s-1} t_i - t'_i|. \end{aligned}$$

For integers $x_i \in \mathbb{Z}$ and any $m \in \mathbb{N}$ the sum $\sum_{i=1}^{m} |x_i|$ is even if and only if $|\sum_{i=1}^{m} x_i|$ is even. And thus it follows that $q_\alpha(\mathcal{B})$ is always even.

2. $0 \leq q_\alpha(\mathcal{B})$ is immediately clear and from the triangle inequality it follows that $q_\alpha(\mathcal{B}) \leq 2n$. Now assume that $q_\alpha(\mathcal{B}) = 2n$. Then $|t_i - t'_i| = t_i + t'_i$ for $i = 1, \ldots, s$. Since $t'_i \geq 0$ and $t_i > 0$, it follows that $t'_1 = \ldots = t'_s = 0$ which is a contradiction to $\mathcal{B} \subseteq A_1 \cup \ldots \cup A_s$. Therefore $q_\alpha(\mathcal{B}) < 2n$

3. $q_\alpha(\mathcal{B}) = 0$ is equivalent to $t_i = |A_i \cap \mathcal{B}|$ for $i = 1, \ldots, n$, which equals the definition of a basis of ideal distribution. □

Remark 4.6. One transformation that we will use at several points in this chapter is the T1-transformation, i.e. changing the elements of a logarithmic signature by applying an automorphism.. For a logarithmic signature $\alpha = [A_1, \ldots, A_s]$ (of the elementary abelian group $\mathbb{Z}_p^n$) and an automorphism φ (of $\mathbb{Z}_p^n$) we defined

$$\varphi(\alpha) = [\varphi(A_1), \ldots, \varphi(A_s)].$$

Recall that if φ and φ^{-1} are polynomial-time computable algorithms, then the factorization problem for α is polynomial time equivalent to the factorization problem for $\varphi(\alpha)$, see also Page 14.

It is clear that linear independence of elements in α implies linear independence of elements in $\varphi(\alpha)$ and vice versa. And given a basis $\mathcal{B} = \{b_1, \ldots, b_n\}$ of elements in α, there always exists the automorphism φ' that maps this basis to the canonical basis, i.e. $\varphi'(b_i) = e_i$. Then the logarithmic signature $\varphi'(\alpha)$ contains the canonical basis $\{e_1, \ldots, e_n\}$. Thus regarding the efficiency of factoring we may always assume that a logarithmic signature contains the canonical basis in some distribution. Or, given a basis of a certain distribution for α we may assume this basis is the canonical basis.

4.1.3 Existence of a basis with ideal distribution

Now we have several definitions at hand to describe the distribution of a basis in a logarithmic signature. In order to motivate our approach we look at

Example 4.7. Let α be a logarithmic signature of type $(4, 4, 8)$ from $\mathbb{Z}_2^7$. We assume that we are given a basis with distribution $(2, 1, 4)$ and used an automorphism to set this basis to the canonical basis

$$\mathcal{B} = \{e_1,\ e_2,\ e_3,\ e_4,\ e_5,\ e_6,\ e_7\}.$$

In this example α has the three blocks

$$\begin{aligned} A_1 &= [\ 0,\ e_1,\ e_2,\ (1,1,0,0,1,0,0)\ ] \\ A_2 &= [\ 0,\ e_3,\ (0,0,0,1,1,0,1),\ (1,0,1,1,1,1,1)\ ] \\ A_3 &= [\ 0,\ e_4,\ e_5,\ e_6,\ e_7, \\ &\qquad (0,0,0,0,1,1,0),\ (1,0,0,0,0,1,1),\ (1,0,0,1,0,1,0)\ ] \end{aligned}$$

The canonical basis $\mathcal{B}$ is not of ideal distribution for α, as $q_\alpha(\mathcal{B}) = 0+1+1 = 2$. A basis with ideal distribution for α has the distribution $(2, 2, 3)$. So $\mathcal{B}$ contains sufficient vectors from the first block, not enough from the second block and too many vectors from the third block. Starting with $\mathcal{B}$ we would like to exchange one of the canonical vectors from A_3 for any vector from A_2 such that the resulting set $\mathcal{B}'$ still is a basis of $\mathbb{Z}_2^7$ but with distribution $(2, 2, 3)$. In this case there are several possibilities: e_4, e_5, and e_7 could be exchanged for either $(0, 0, 0, 1, 1, 0, 1)$ or $(1, 0, 1, 1, 1, 1, 1)$, whereas e_7 can only be exchanged for $(1, 0, 1, 1, 1, 1, 1)$.

From the example we observe the general strategy for exchanging vectors in a basis to improve its quality: Remove vectors from a block A that contains too many vectors in the basis and add a vector from a block B that does not have sufficient vectors in the basis via: $e_i \in A$ can be exchanged for any other vector $v \in B$ such that $v(i) \neq 0$. This idea is used in the proof of the following Theorem 4.9. First we give a lemma essential for the proof of the theorem.

Lemma 4.8. *Let $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signature, $i_1, \ldots, i_k \in \{1, \ldots, s\}$, $k \leq s$, and $j \in \{1, \ldots, n\}$. If there exists*

$$v \in A_{i_1} + \ldots + A_{i_k} \text{ such that } v(j) \neq 0,$$

then there exists an element

$$v' \in A_{i_1} \cup \ldots \cup A_{i_k} \text{ such that } v'(j) \neq 0.$$

Proof. The equation $v'(j) = 0$ for all $v' \in A_{i_1} \cup \ldots \cup A_{i_k}$ directly implies $v(j) = 0$ for all $v \in A_{i_1} + \ldots + A_{i_k}$. □

Theorem 4.9. *Let p be a prime number, $n \in \mathbb{N}$, and $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signature of the elementary abelian group $\mathbb{Z}_p^n$. Then there exists a basis $\mathcal{B}$ of $\mathbb{Z}_p^n$ with ideal distribution for α.*

Proof. By Lemma 4.2 α contains a basis $\mathcal{B}$ of $\mathbb{Z}_p^n$. Let

$$t_i = \log_p |A_i|,$$

and

$$t_i' = |A_i \cap \mathcal{B}|$$

for $i = 1, \ldots, s$. Consider the quality of the basis $\mathcal{B}$

$$q_\alpha(\mathcal{B}) = \sum_{i=1}^{s} |t_i - t_i'| \geq 0.$$

If $q_\alpha(\mathcal{B}) = 0$, then $\mathcal{B}$ is a basis with ideal distribution $d_\alpha(\mathcal{B}) = (t_1, \ldots, t_s)$ and the theorem holds.

If $q_\alpha(\mathcal{B}) > 0$ we show that α contains a basis $\mathcal{B}'$ that is better than $\mathcal{B}$, such that

$$q_\alpha(\mathcal{B}') = q_\alpha(\mathcal{B}) - 2.$$

Without loss of generality we renumber the blocks of α such that for some $k, l \in \{0, \ldots, s\}$ the first k blocks contain the right amount of basis vectors

from $\mathcal{B}$, the blocks $k+1$ through l contain more basis vectors and the blocks $l+1$ to s contain less basis vectors than necessary such that $\mathcal{B}$ is a basis of ideal distribution. We have

$$\begin{aligned} t_i = t'_i \quad &\text{for} \quad i = 1, \ldots, k \\ t_i < t'_i \quad &\text{for} \quad i = k+1, \ldots, l \\ t_i > t'_i \quad &\text{for} \quad i = l+1, \ldots, s \end{aligned}$$

Furthermore due to Remark 4.6 we may assume that $\mathcal{B}$ is the canonical basis

$$\mathcal{B} = \{e_1, \ldots, e_n\}$$

and the canonical vectors are distributed on the blocks of α in ascending order:

$$\begin{aligned} e_1, \ldots, e_{k'} &\in A_1 \cup \ldots \cup A_k \\ e_{k'+1}, \ldots, e_{l'} &\in A_{k+1} \cup \ldots \cup A_l \\ e_{l'+1}, \ldots, e_n &\in A_{l+1} \cup \ldots \cup A_s \end{aligned}$$

where

$$k' = \sum_{i=1}^{k} t_i = \sum_{i=1}^{k} t'_i$$

and $l' = \sum_{i=1}^{l} t'_i$.

We consider two cases:

(I) $t'_i \neq t_i$ for $i = 1, \ldots, n$.

We have $k = 0$. In this case the blocks of α contain either too many or not enough vectors of $\mathcal{B}$ such that $\mathcal{B}$ is of ideal distribution for α. Combine those blocks that contain to many vectors in

$$B_1 = A_1 + \ldots + A_l,$$

and those that do not contain enough vectors in

$$B_2 = A_{l+1} + \ldots + A_s.$$

Consider the logarithmic signature

$$\beta = [B_1, B_2].$$

A basis of ideal distribution for α has the distribution

$$(t_1, \ldots, t_s).$$

Therefore a basis of ideal distribution for β would have the distribution

$$(\sum_{i=1}^{l} t_i, \sum_{i=l+1}^{s} t_i).$$

The distribution of $\mathcal{B}$ with respect to β is

$$d_\beta(\mathcal{B}) = (l', n - l')$$

because the first block of β contains the canonical vectors $e_1, \ldots, e_{l'}$ and the second block contains $e_{l'+1}, \ldots, e_n$. And since $t'_i < t_i$ for $i = 1, \ldots, l$

$$l' < \sum_{i=1}^{l} t_i$$

and $n - l' > \sum_{i=l+1}^{s} t_i$.

Now let $M := < e_1, \ldots, e_{l'} > \cap B_1$. Then

$$|M| \leq | < e_1, \ldots, e_{l'} > | = p^{l'} < p^{\sum_{i=1}^{l} t_i} = |B_1|.$$

This implies that the set $B_1 \backslash M$ is not empty. Let $x \in B_1 \backslash M$, then one of the entries $x(l'+1), \ldots, x(n)$ is not zero. W.l.o.g. we may assume that

$$x(l'+1) \neq 0.$$

Note that x is not necessarily an element of one of the blocks of α; but by Lemma 4.8 there exists an element $x' \in A_1 \cup \ldots \cup A_l$ such that

$$x'(l'+1) \neq 0.$$

W.l.o.g we may assume that $x' \in A_l$ and also $e_{l'+1} \in A_{l+1}$. (The arguments for $x' \in A_1, \ldots, x \in A_{l-1}$ and $e_{l'+1} \in A_{l+2}, \ldots, e_{l'+1} \in A_s$ are the same).

Set

$$\mathcal{B}' = \{e_1, \ldots, e_{l'}, x', e_{l'+2}, \ldots, e_n\}.$$

Since $x'(l'+1) \neq 0$, the set $\mathcal{B}'$ is a basis of $\mathbb{Z}_p^n$ and also $\mathcal{B}' \subseteq A_1 \cup \ldots \cup A_s$. Because $t_l < t'_l$ and $t_{l+1} > t'_{l+1}$ we get

$$q_\alpha(\mathcal{B}') = |t_1 - t'_1| + \ldots + |t_{l-1} - t'_{l-1}| + |t_l - (t'_l + 1)| + |t_{l+1} - (t'_{l+1} - 1)| + \ldots + |t_s - t'_s|$$

$$= q_\alpha(\mathcal{B}) - 2,$$

i.e. $\mathcal{B}'$ is better than $\mathcal{B}$ with respect to α.

(II) $t'_i = t_i$ for some $i \in \{1, \ldots, n\}$.

In this case $k > 0$ and $l < n$. From α we derive a new logarithmic signature

$$\beta = [B_1, B_2, B_3].$$

with the three blocks

$$\begin{aligned} B_1 &= A_1 + \ldots + A_k, \\ B_2 &= A_{k+1} + \ldots + A_l, \\ B_3 &= A_{l+1} + \ldots + A_s. \end{aligned}$$

A basis of ideal distribution for α has the distribution $(t_1, \ldots, t_s)$. Therefore a basis of ideal distribution for β would have the distribution

$$(\sum_{i=1}^{k} t_i, \sum_{i=k+1}^{l} t_i, \sum_{i=l+1}^{s} t_i).$$

Since

$$\begin{aligned} e_1, \ldots, e_{k'} &\in B_1 \\ e_{k'+1}, \ldots, e_{l'} &\in B_2 \\ e_{l'+1}, \ldots, e_n &\in B_3 \end{aligned}$$

the distribution of $\mathcal{B}$ with respect to β is

$$d_\beta(\mathcal{B}) = (k', l' - k', n - l').$$

And since $t'_i < t_i$ for $i = k+1, \ldots, l$

$$l' - k' = \sum_{i=k+1}^{l} t'_i < \sum_{i=k+1}^{l} t_i$$

and also

$$n - l' = \sum_{i=l+1}^{s} t'_i > \sum_{i=l+1}^{s} t_i.$$

We consider two cases

1. There exists a vector $x \in B_2$ such that

$$x(i) \neq 0 \text{ for some } i \in \{l'+1, \ldots, n\}.$$

W.l.o.g. we may assume that

$$x(l'+1) \neq 0.$$

Note that $x \in B_2$ is not necessarily an element of one of the blocks of α; but by Lemma 4.8 there exists an element

$$x' \in A_{k+1} \cup \ldots \cup A_l$$

such that

$$x'(l'+1) \neq 0.$$

W.l.o.g we may assume that

$$x' \in A_l$$

and also

$$e_{l'+1} \in A_{l+1}.$$

(The arguments for $x' \in A_1, \ldots, x \in A_{l-1}$ and $e_{l'+1} \in A_{l+2}, \ldots, e_{l'+1} \in A_s$ are the same).

Set

$$\mathcal{B}' = \{e_1, \ldots, e_{l'}, x', e_{l'+2}, \ldots, e_n\}.$$

Since $x'(l'+1) \neq 0$, the set $\mathcal{B}'$ is a basis of $\mathbb{Z}_p^n$ and also $\mathcal{B}' \subseteq A_1 \cup \ldots \cup A_s$. Because

$$t_l > t'_l \text{ and } t_{l+1} < t'_{l+1}$$

we get

$$\begin{aligned} q_\alpha(\mathcal{B}') &= |t_1 - t'_1| + \ldots + |t_{l-1} - t'_{l-1}| + |t_l - (t'_l + 1)| + \\ &\quad |t_{l+1} - (t'_{l+1} - 1)| + |t_{l+2} - t'_{l+2}| + \ldots + |t_s - t'_s| \\ &= q_\alpha(\mathcal{B}) - 2, \end{aligned}$$

i.e. $\mathcal{B}'$ is better than $\mathcal{B}$ with respect to α.

2. $x(i) = 0$ for all $x \in B_2$ and all $i \in \{l'+1, \ldots, n\}$.

In this case we have

$$B_2 \subseteq < e_1, \ldots, e_{l'} > .$$

Since B_3 only contains the canonical vectors $e_{l'+1}, \ldots, e_n$, it is not possible - as in the case before - to find a vector in B_2 that might be exchanged for a canonical vector in B_3 to yield a basis with better distribution. Instead of one exchange of vectors, two exchanges of vectors are necessary. Since

$$| < e_{k'+1}, \ldots, e_{l'} > | = p^{l'-k'} < p^{\sum_{i=k+1}^{l} t_i} = |B_2|,$$

there exists $v \in B_2$ such that $v(j) \neq 0$ for some $j \in \{1, \ldots, k'\}$.

By Lemma 4.8 there exists an element $v' \in A_{k+1} \cup \ldots \cup A_l$ such that

$$v'(j) \neq 0.$$

W.l.o.g we may assume that

$$v' \in A_{k+1}$$

and also

$$j = k'.$$

(The arguments for $v' \in A_k + 1, \ldots, v' \in A_{l-1}$ and $j = 1, \ldots, k' - 1$ are the same). Set

$$\mathcal{B}'' = \{\underbrace{e_1, \ldots, e_{k'-1}}_{B_1}, \underbrace{v', e_{k'+1}, \ldots, e_{l'}}_{B_2}, \underbrace{e_{l'+1}, \ldots, e_n}_{B_3}\}.$$

Then the distribution of $\mathcal{B}''$ with respect to β is

$$(k' - 1, l' - k' + 1, n - l')$$

and differs from the distribution of $\mathcal{B}$, but

$$\begin{aligned} q_\alpha(\mathcal{B}'') &= |\sum_{i=1}^{k} t_i - (k' - 1)| + |\sum_{i=k+1}^{l} t_i - (l' - k' + 1)| \\ &\quad + |\sum_{i=l+1}^{s} t_i - (n - l')'| \\ &= |\sum_{i=1}^{k} t_i - k'| - 1 + |\sum_{i=k+1}^{l} t_i - (l' - k')| + 1 \\ &\quad + |\sum_{i=l+1}^{s} t_i - (n - l')'| \\ &= q_\alpha(\mathcal{B}), \end{aligned}$$

i.e. the quality of $\mathcal{B}''$ has not improved compared to the quality of $\mathcal{B}$.

Now we look at the elements of B_1: Suppose that $w(k) = 0$ for all $w \in B_1$ and $k \in \{l'+1, \ldots, n\}$. Then

$$B_1 \subseteq < e_1, \ldots, e_{l'} >$$

and even

$$B_1 + B_2 \subseteq < e_1, \ldots, e_{l'} > .$$

This yields the contradiction:

$$|B_1 + B_2| \leq | < e_1, \ldots, e_{l'} > | = p^{l'} < p^{\sum_{i=1}^{l} t_i} = |B_1 + B_2|.$$

Therefore there exists a vector $w \in B_1$ such that

$$w(h) \neq 0 \text{ for some } h \in \{l'+1, \ldots, n\},$$

and by Lemma 4.8 there exists a vector $w' \in A_1 \cup \ldots \cup A_k$ such that

$$w'(h) \neq 0.$$

W.l.o.g we may assume that

$$w' \in A_k$$

and also

$$h = l' + 1.$$

(The arguments for $w' \in A_1, \ldots, w' \in A_{k-1}$ and $h = l'+2, \ldots, n$ are similar).

Set

$$\mathcal{B}' = \{\underbrace{e_1, \ldots, e_{k'-1'}, w'}_{B_1}, \underbrace{v', e_{k'+1}, \ldots, e_{l'}}_{B_2}, \underbrace{e_{l'+2}, \ldots, e_n}_{B_3}\}.$$

Since $v'(k') \neq 0$ and $w'(l'+1) \neq 0$, the set $\mathcal{B}'$ is a basis of $\mathbb{Z}_p^n$ and also $\mathcal{B}' \subseteq A_1 \cup \ldots \cup A_s$. Because

$$t_{k+1} > t'_{k+1} \text{ and } t_{l+1} < t'_{l+1}$$

we get

$$\begin{aligned} q_\alpha(\mathcal{B}') &= |t_1 - t'_1| + \ldots + |t_{k-1} - t'_{k-1}| + |t_k - (t'_k - 1 + 1)| + \\ &\quad |t_{k+1} - (t'_{k+1} + 1)| + |t_{k+2} - t'_{k+2}| + \ldots + |t_l - t'_l| + \\ &\quad |t_{l+1} - (t'_{l+1} - 1)| + |t_{l+2} - t'_{l+2}| + \ldots + |t_s - t'_s| \\ &= q_\alpha(\mathcal{B}) - 2, \end{aligned}$$

i.e. $\mathcal{B}'$ improves $\mathcal{B}$ with respect to α.

We have shown that given a basis of $\mathbb{Z}_p^n$ in a logarithmic signature α this is either a basis of ideal distribution or we can find a basis that improves the quality of the first basis by 2. The quality function is always even, if its input is a basis contained by the considered logarithmic signature, see Proposition 4.5. Repeating the improvement process at most n times yields a basis of ideal distribution. □

With this theorem we see that there is a basic structure common to all logarithmic signatures in elementary abelian p-groups. However, the theorem does not lead immediately to a full characterization, but it will help in our analysis of logarithmic signatures in the following sections.

Theorem 4.9 does not only answer the first question posed on page 53 regarding the existence of a basis of ideal distribution for all logarithmic signatures of elementary abelian groups. The proof of Theorem 4.9 gives the outline of an algorithm to compute such a basis in polynomial time and this answers the second question.

4.1.4 Normal form of a logarithmic signature

Due to Theorem 4.9 and Remark 4.6 we showed that for any logarithmic signature α in $\mathbb{Z}_p^n$ there exists an efficiently computable automorphism φ such that $\varphi(\alpha)$ contains the canonical basis with ideal distribution. It is also clear that computing factorizations for α and $\varphi(\alpha)$ are polynomial-time equivalent.

Therefore we can restrict our further analysis to logarithmic signatures that contain the the canonical basis with ideal distribution. We can also make the following constraint on the order of the basis vectors in the logarithmic signature.

Definition 4.10. A logarithmic signature that contains the canonical basis with ideal distribution in ascending order (i.e. $e_i \in A_k$, $e_j \in A_{k+1}$, then $i < j$) is called a logarithmic signature in *normal form*.

Note that there might be several different automorphisms that transform α such that it contains the canonical basis with ideal distribution - depending on the number of different bases with ideal distribution for α. For example, the logarithmic signature α in $\mathbb{Z}_3^3$ from Example 4.1 with the two blocks

$$\begin{aligned} A_1 &= [\ (0,0,0),\ (1,0,0),\ (0,1,0)\ ] \\ A_2 &= [\ (0,0,0),\ (1,1,0),\ (2,2,0),\ (0,0,1), \\ &\quad\ (0,0,2),\ (1,1,1),\ (1,1,2),\ (2,2,1),\ (2,2,2)\ ]. \end{aligned}$$

includes the bases $(1,0,0),\ (1,1,0),\ (0,0,1)$ and $(0,1,0),\ (1,1,1),\ (1,1,2)$ with ideal distribution. It also contains the canonical basis, but not with ideal distribution. The automorphism defined by

$$\varphi_1 :\ \ e_1 \mapsto e_1,\ e_2 \mapsto (2,1,0),\ e_3 \mapsto e_3$$

changes α into the normal form

$$\begin{aligned} \varphi_1(A_1) &= [\ (0,0,0),\ (1,0,0),\ (2,1,0)\ ] \\ \varphi_1(A_2) &= [\ (0,0,0),\ (0,1,0),\ (0,2,0),\ (0,0,1), \\ &\quad\ (0,0,2),\ (0,1,1),\ (0,1,2),\ (0,2,1),\ (0,2,2)\ ]. \end{aligned}$$

The automorphism defined by

$$\varphi_2 :\ \ e_1 \mapsto e_1,\ e_2 \mapsto (2,2,1),\ e_3 \mapsto e_2$$

changes α into the normal form

$$\begin{aligned} \varphi_2(A_1) &= [\ (0,0,0),\ (1,0,0),\ (2,2,1)\ ] \\ \varphi_2(A_2) &= [\ (0,0,0),\ (0,2,1),\ (0,1,2),\ (0,1,0), \\ &\quad\ (0,2,0),\ (0,0,1),\ (0,1,1),\ (0,2,2),\ (0,0,2)\ ]. \end{aligned}$$

The logarithmic signatures $\varphi_1(\alpha)$ and $\varphi_2(\alpha)$ both contain the canonical basis with ideal distribution, but differ in other elements. Thus the normal form is not unique.

Given any logarithmic signature α in $\mathbb{Z}_2^n$, where $|A_i| \leq 4$, the computation of the normal form is efficient.

GAP: NormalFormLogSig

Input: logSig, a normalized logarithmic signature of type $(4,\ldots,4,2,\ldots,2)$ of $\mathbb{Z}_2^n$, where $n \in \mathbb{N}$
Output: the normal form of logSig

```
NormalFormLogSig:=function(logSig)

local i,j,Automorphism; Automorphism:=[];

for i in [1..Length(logSig)] do

  if Size(logSig[i])=2) then
    Append(Automorphism, logSig[i][2] );
    else Append(Automorphism, logSig[i]{[2..4]} );
  fi;
od;

return logSig*Inverse(Automorphism);

end;
```

The running time is dominated by the final matrix multiplication and inversion and is therefore approximately $O(n^2 \cdot s \cdot r_\alpha)$.

4.2 Logarithmic signatures in $\mathbb{Z}_2^n$

In this section we will study logarithmic signatures of elementary abelian 2-groups. First in a general case where we give a characterization of logarithmic signatures via sets of linearly independent vectors. In the case where blocks are of size less or equal to 4, we will show that it is possible to characterize these logarithmic signatures via acyclic digraphs (with labeled vertices). It follows that in such logarithmic signatures there is always one block that is a subgroup. Furthermore we derive an efficient factoring algorithm. This is joint work with Rouven Walter.

4.2.1 A characterization via linearly independent sets

In Theorem 4.9 we saw that there are always certain sets of linearly independent vectors in a logarithmic signature of an elementary abelian p-group. For elementary abelian 2-groups $\mathbb{Z}_2^n$ we can even give a characterization of logarithmic signatures via the existence of certain sets of linearly independent vectors. The following theorem will also be used in the proof of Theorem 4.15.

Theorem 4.11. *Let $s, n, l \in \mathbb{N}$ and for $i = \{1, \ldots, s\}$ take subsets $B_i \subseteq \mathbb{Z}_2^n$ with $0 \in B_i$, and $\prod_{i=1}^{s} |B_i| = 2^n$. Then $[B_1, \ldots, B_s]$ is a logarithmic signature if and only if all sets*

$$B = \{b_1, \ldots, b_l\} \subset B_1 \cup \ldots \cup B_s$$

such that

1. $0 \notin B$,
2. $b_i \neq b_j$ *for* $i \neq j$,
3. *for all* $i = \{1, \ldots, s\}$*:* $|B \cap B_i| \in \{0, 1, 2\}$

are linearly independent.

Proof. Let $[B_1, \ldots, B_s]$ be a logarithmic signature. Let $B = \{b_1, \ldots, b_t\} \subset B_1 \cup \ldots \cup B_s$ as given in the theorem. W.l.o.g we may assume that $t = 2s$. Let $b_i, b_{s+i} \in B_i$ for $i = 1, \ldots, s$.

Suppose that the vectors of B are linearly dependent, i.e. there exist bits $l_1, \ldots, l_{2s} \in \mathbb{Z}_2$ (at least one of these equal to 1) such that

$$\sum_{i=1}^{2s} l_i b_i = 0.$$

As all non-trivial vectors of $\mathbb{Z}_2^n$ are their own inverses, it follows that

$$\sum_{i=1}^{s} l_i b_i = \sum_{i=s+1}^{2s} l_i b_i.$$

Let

$$c_i = \begin{cases} b_i, & l_i = 1 \\ 0, & l_i = 0 \end{cases}$$

Then $c_i, c_{i+s} \in B_i$. Because of the properties *1.* to *3.* the sets $\{c_1, \ldots, c_s\}$ and $\{c_{s+1}, \ldots, c_{2s}\}$ are two different factorizations of the same element, which is a contradiction to the hypothesis that $[B_1, \ldots, B_s]$ is a logarithmic signature. Therefore the three properties on the elements of B are necessary properties of a logarithmic in an elementary abelian 2-group.

Now, let all collections of vectors with the properties 1. to 3. be linearly independent.

Suppose that $[B_1, \ldots, B_s]$ does not define a logarithmic signature of $\mathbb{Z}_2^n$. Since $\prod_{i=1}^{s} |B_i| = 2^n$, there exists an element $x \in \mathbb{Z}_2^n$ with at least two different factorizations with respect to $[B_1, \ldots, B_s]$ (otherwise it would be a logarithmic signature), i.e.

$$x = \sum_{i=1}^{s} b_i = \sum_{i=1}^{s} b'_i,$$

where $b_i, b'_i \in B_i$. Let $B = \{b_1, \ldots, b_s, b'_1, \ldots, b'_s\}$. Consider the set

$$C = \{b \in B | b \neq 0, b \neq b'\}.$$

This set is not empty, fulfills properties 1. to 3. and since

$$\sum_{c \in C} c = \sum_{i=1}^{s} b_i + b'_i = 0,$$

the vectors of C are linearly dependent. This is a contradiction to the assumption that all collections of vectors with the properties 1. to 3. are linearly independent. Thus $[B_1, \ldots, B_s]$ is a logarithmic signature. □

Comparison to Theorem 4.9: If we consider a logarithmic signature in $\mathbb{Z}_2^n$ with blocks of length less or equal to 4, then the theorem above gives a generalization: Theorem 4.9 shows the existence of a basis of ideal distribution, while Theorem 4.11 shows that every maximal collection of vectors with properties *1.* to *3.* is a basis with ideal distribution.

It is not possible to get better or different results by choosing three or more vectors from one block for the set B. In Example 4.7 with

$$\begin{aligned} A_1 &= [\, 0,\ e_1,\ e_2,\ (1,1,0,0,1,0,0)\,] \\ A_2 &= [\, 0,\ e_3,\ (0,0,0,1,1,0,1),\ (1,0,1,1,1,1,1)\,] \\ A_3 &= [\, 0,\ e_4,\ e_5,\ e_6,\ e_7, \\ &\qquad (0,0,0,0,1,1,0),\ (1,0,0,0,0,1,1),\ (1,0,0,1,0,1,0)\,] \end{aligned}$$

we see that for example the vectors $e_1,\ e_2,\ (1,1,0,0,1,0,0)$ and e_5 are linearly dependent, where the first three vectors are chosen from the same block A_1.

It is also not possible to extend this characterization of logarithmic signatures in elementary abelian 2-groups to elementary abelian p-groups for an odd prime p: Take the logarithmic signature $\alpha = [A_1, A_2]$ from $\mathbb{Z}_3^4$ with the blocks

$$\begin{aligned} A_1 &= [\ (0,0,0,0),\ (1,0,0,0),\ (0,1,0,0),\ (0,0,0,1),\\ &\quad\ (1,0,0,1),\ (0,1,0,1),\ (0,0,0,2),\ (1,0,0,2),\ (0,1,0,2)\ ]\\ A_2 &= [\ (0,0,0,0),\ (1,1,0,0),\ (2,2,0,0),\ (0,0,1,0),\\ &\quad\ (0,0,2,0),\ (1,1,1,0),\ (1,1,2,0),\ (2,2,1,0),\ (2,2,2,0)\ ]. \end{aligned}$$

We immediately see that there are elements, both in one block, that are linearly dependent, for example $(1,1,0,0)$ and $(2,2,0,0)$ in A_2. Even if we choose only linearly independent vectors from each block for the set B, the theorem still does not hold: the set $\{(1,0,0,0),\ (0,1,0,0),\ (1,1,0,0),\ (2,2,1,0)\}$ is chosen as in Theorem 4.11 with the additional property that vectors from the same block are linearly independent, but the set itself is linearly dependent. So $p = 2$ is a necessary condition for the Theorem to hold. In fact, the property that elements of $\mathbb{Z}_2^n$ are their own inverses is essential to the proof of Theorem 4.11

4.2.2 Relating logarithmic signatures to graphs

In this section we relate logarithmic signatures of $\mathbb{Z}_2^n$ with blocks of size less or equal to 4 to acyclic digraphs with labeled vertices. We show that this connection leads to a characterization and a factoring algorithm for these logarithmic signatures. The definition of acyclic digraphs and all graph theoretic notations are according to [BJG08].

Definition 4.12. A *digraph* or directed graph is a graph $G = (V, E)$ with a finite set V of vertices and a set of arcs $E \subseteq V \times V$. An arc is an ordered pair of distinct vertices $(u, v) \in E$ that defines a directed edge from vertex u to vertex v, where u is called *tail* and v is called the *head* of (u, v).

If G does not contain parallel arcs, loops or directed cycles it is called *acyclic digraph.*

For a vertex $v \in V$ the *in-degree* is the number of arcs with head v, the *out-degree* is the number of arcs with tail v.

Proposition 4.13. *In an acyclic digraph there exists a vertex of in-degree zero as well as a vertex of out-degree zero.*

Proof. see [BJG08], Proposition 1.4.2. □

Let $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signature in $\mathbb{Z}_2^n$, where $|A_i| \leq 4$ and $|A_i| \geq |A_j|$ for $i \leq j$ in normal form, i.e. for some $k \leq n$ the blocks of α are

$$\begin{aligned} A_1 &= [\ 0,\ e_1,\ e_2,\ v_1\ ], \\ A_2 &= [\ 0,\ e_3,\ e_4,\ v_2\ ], \\ &\vdots \\ A_k &= [\ 0,\ e_{2k-1},\ e_{2k},\ v_k\ ], \\ A_{k+1} &= [\ 0,\ e_{2k+1}\ ], \\ &\vdots \\ A_s &= [\ 0,\ e_n\ ], \end{aligned}$$

where the vectors $v_1, \ldots, v_k \in \mathbb{Z}_p^n$ are not further specified. We focus on the indeterminates $v_1, \ldots, v_k$, which suffice to characterize a logarithmic signature of the type $(4, \ldots, 4, 2, \ldots, 2)$. Therefore we may ignore the blocks of size 2 and consider only logarithmic signatures of type $(4, \ldots, 4)$.

Now we take a closer look at the indeterminates $v_1, \ldots, v_k$. We observe that two entries in each v_i are connected to the two canonical vectors in the same block A_i.

Proposition 4.14. *Let α be a logarithmic signature of $\mathbb{Z}_2^n$ with type $(4, \ldots, 4)$ in normal form and $i \in \{1, \ldots, n\}$. If v_i denotes the vector of block A_i that is no canonical vector, then*

$$v_i(2i-1) = v_i(2i) = 1.$$

Proof. Let α be a logarithmic signature of $\mathbb{Z}_2^n$ with type $(4, \ldots, 4)$ in normal form and $i \in \{1, \ldots, n\}$. We consider the entries $v_i(2i-1)$ and $v_i(2i)$ of the vector $v_i \in A_i$.

Suppose that $v_i(2i-1) = 0$ for some $i \in \{1, \ldots, n\}$. W.l.o.g we may assume that $i = 1$, i.e. $v_1(1) = 0$. Consider the collection

$$C = \{\underbrace{v_1, e_2}_{\in A_1}, \underbrace{e_3, e_4}_{\in A_2}, \ldots, \underbrace{e_{n-1}, e_n}_{\in A_s}\}.$$

Since $v_1(1) = 0$ we have

$$v_1 = \sum_{j=1}^{n} v_1(j) e_j = 0 + \sum_{j=2}^{n} v_1(j) e_j,$$

i.e. the vectors of C are linearly dependent. This leads to a contradiction, because by Theorem 4.11 the set C is linearly independent. Thus $v_i(2i-1) = 1$.

With the same argument we see that $v_i(2i) = 1$. □

Propositions 4.13 and 4.14 help to give a complete characterization of the normal form of logarithmic signatures of type $(2, \ldots, 2, 4, \ldots, 4)$ in elementary abelian 2-groups via acyclic digraphs. We state the theorem for the case where each block is of order 4, but it is easily adapted for logarithmic signatures with blocks of order 2 and 4.

Theorem 4.15. *Let $n \in \mathbb{N}$ and $\alpha = [A_1, \ldots, A_s]$ be a sequence of ordered subsets of $\mathbb{Z}_2^n$ of the form*

$$\begin{aligned} A_1 &= [\ 0,\ e_1,\ e_2,\ v_1\ ], \\ A_2 &= [\ 0,\ e_3,\ e_4,\ v_2\ ], \\ &\vdots \\ A_i &= [\ 0,\ e_{2i-1},\ e_{2i},\ v_i\ ], \\ &\vdots \\ A_s &= [\ 0,\ e_{n-1},\ e_n,\ v_s\ ], \end{aligned}$$

for $s = \frac{n}{2}$. The sets

$$V = \{v_1, \ldots, v_s\} \subseteq \mathbb{Z}_2^n.$$

and

$$E = \{(v_i, v_j) \in V \times V \mid v_i \neq v_j,\ v_i(2j-1, 2j) \neq (0,0)\ \}.$$

define a digraph

$$G(\alpha) := (V, E).$$

Then

$$\alpha \text{ is a logarithmic signature of } \mathbb{Z}_2^n$$

$$\Longleftrightarrow$$

$$G(\alpha) \text{ is an acyclic digraph}$$

Proof. 1. Let α be a logarithmic signature.

Suppose that $G(\alpha)$ is not acyclic. Then there exists a *shortest* (directed) cycle

$$C = v_{j_1}, \ldots, v_{j_r}, v_{j_1}$$

of length r in $G(\alpha)$, with $j_1, \ldots, j_r \in \{1, \ldots, s\}$, and

$$v_{j_i} \neq v_{j_k}$$

for $1 \leq i < k \leq r$.

Note that permuting the blocks of α does not change the structural properties of $G(\alpha)$, especially the existence of cycles. For easier notation we may therefore assume that the vertices in C are

$$C = v_1, \ldots, v_r, v_1.$$

It follows that

$$(v_i, v_{i+1}) \in E$$

for $i = 1, \ldots, r-1$ and also $(v_r, v_1) \in E$.

We look at the first r entries of the vectors $v_1, \ldots, v_r$. Set

$$y_i \;=\; v_i(2i+1,\ 2i+2)$$

for $i = 1, \ldots, r-1$, and

$$y_r = v_r(1,\ 2).$$

Note that for $i = 1, \ldots, r$

$$y_i \neq (0, 0).$$

By Proposition 4.14 we have

$$v_i(2i-1, 2i) = (1, 1)$$

for $i = 1, \ldots, r$. Thus several entries of the vectors corresponding to the vertices of C are known:

$$\begin{array}{lccccccccc}
v_1 & = (& 1\ 1 & y_1(1)\ y_1(2) & * * & \ldots & * * & * * & \ldots &) \\
v_2 & = (& * * & 1\ 1 & y_2(1)\ y_2(2) & \ldots & * * & * * & \ldots &) \\
v_3 & = (& * * & * * & 1\ 1 & \ldots & * * & * * & \ldots &) \\
 & \vdots & & & & & & & & \\
v_{r-1} & = (& * * & * * & * * & \ldots & 1\ 1 & y_{r-1}(1)\ y_{r-1}(2) & \ldots &) \\
v_r & = (& y_r(1)\ y_r(2) & * * & * * & \ldots & * * & 1\ 1 & \ldots &)
\end{array}$$

If one of the unknown entries denoted by $*$ is not zero, then there exists a cycle with length less than r. This would be a contradiction to the assumption that C is the shortest cycle in $G(\alpha)$. Therefore all entries $*$ are zero and the first $2r$ entries of the vectors $v_1, \ldots, v_r$ are fixed. We now define a second set of vectors $w_1 \in A_2, w_2 \in A_3, \ldots, w_{r-1} \in A_r, w_r \in A_1$ via

$$w_i = \begin{cases} 0, & \text{if } y_i = (1,1) \\ e_{2i-1}, & \text{if } y_i = (0,1) \\ e_{2i}, & \text{if } y_i = (1,0) \end{cases}$$

for $i = 1, \ldots, r$, i.e.

$$\begin{array}{lllllllllll}
w_1 & = (& 0 & 0 & \overline{y_1(1)} & \overline{y_1(2)} & 0 & 0 & \ldots & & 0) \\
w_2 & = (& 0 & 0 & 0 & 0 & \overline{y_2(1)} & \overline{y_2(2)} & 0 & \ldots & 0) \\
 & \vdots & & & & & & & & & \\
w_{r-1} & = (& 0 & 0 & & & \ldots & & 0 & \overline{y_{r-1}(1)}\ \overline{y_{r-1}(2)} & 0\ \ldots\ 0) \\
w_r & = (& \overline{y_r(1)} & \overline{y_r(2)} & 0 & 0 & & & & \ldots & 0)
\end{array}$$

where we use the notation

$$\overline{y} = y + 1$$

for $y \in \mathbb{Z}_2$. Now consider the vector

$$x = \sum_{i=1}^{r} v_i + \sum_{i=1}^{r} w_i,$$

where $v_1, \ldots, v_r, w_1, \ldots, w_r \in A_1 \cup \ldots \cup A_r$. Note that

$$x(1,2) = (1,1) + (y_r(1),\ y_r(2)) + (\overline{y_r(1)}, \overline{y_r(2)}) = (0,0)$$

and

$$x(2i-1, 2i) = (y_{i-1}(1),\ y_{i-1}(2)) + (1,1) + (\overline{y_{i-1}(1)}, \overline{y_{i-1}(2)}) = (0,0)$$

for $i = 2, \ldots, r$. Thus

$$x \in < e_{2r+1}, \ldots, e_n >$$

and there exists a linear combination

$$x = \sum_{i=2r+1}^{n} c_i e_i$$

for some $c_{2r+1}, \ldots, c_n \in \mathbb{Z}_2$. The set of vectors

$$\begin{aligned} M = \quad & \{v_1, \ldots, v_r\} \\ & \cup \{w_i \mid w \neq 0,\ i \in \{1, \ldots, r\}\} \\ & \cup \{e_i \mid c_i \neq 0,\ i \in \{2r+1, \ldots, n\}\} \end{aligned}$$

fulfills

1. $0 \notin M$
2. All listed elements of M are pairwise distinct.
3. for all $i = \{1, \ldots, s\}$: $|M \cap A_i| \in \{0, 1, 2\}$

Due to Theorem 4.11 M is a set of linearly independent vectors. This contradicts

$$\begin{aligned} \textstyle\sum_{m \in M} m &= \textstyle\sum_{i=1}^{r} v_i + \sum_{i=1, w_i \neq 0}^{r} w_i + \sum_{i=2r+1, c_i \neq 0}^{n} e_i \\ &= \textstyle\sum_{i=1}^{r} v_i + \sum_{i=1}^{r} w_i + \sum_{i=2r+1}^{n} c_i e_i \\ &= x + x \\ &= 0. \end{aligned}$$

Therefore the assumption that $G(\alpha)$ contains a cycle is false and therefore $G(\alpha)$ is acyclic.

2. Let α be a sequence as given in the premise of this theorem and let the corresponding graph $G(\alpha)$ be acyclic.

We use Theorem 4.11 to show that α is a logarithmic signature. First note that for $i = 1, \ldots, s$ we have $A_i \subseteq \mathbb{Z}_2^n$ with $0 \in A_i$, and $\prod_{i=1}^{s} |A_i| = \prod_{i=1}^{s} 2^2 = 2^{2s} = 2^n$. Now we show that all sets

$$B = \{a_1, \ldots, a_n\} \subset A_1 \cup \ldots \cup A_s$$

such that

1. $0 \notin B$,
2. $a_i \neq a_j$ for $i \neq j$,
3. for all $i = \{1, \ldots, s\}$: $|B \cap A_i| \in \{0, 1, 2\}$

are linearly independent. It suffices to consider only sets B of maximal size n, because linear independence of a set B of maximal size implies linear independence for collections of less than n vectors. Thus we assume that $|B \cap A_i| = 2$.

Let

$$B = \{a_1, \ldots, a_s, a'_1, \ldots, a'_s\}$$

such that for $i = 1, \ldots, s$ we have $a_i, a'_i \in A_i$, $a_i, a'_i \neq 0$ and $a'_i \neq v_i$.

Case 1. If $a_i \neq v_i$ for $i = 1, \ldots, n$, then $B = \{e_1, \ldots, e_n\}$ is linearly independent.

Case 2. There exists $r \in \{1, \ldots, s\}$ such that

$$a_{j_1} = v_{j_1}, \ldots, a_{j_r} = v_{j_r}$$

for some $j_1, \ldots, j_r \in \{1, \ldots, s\}$ and $a_i \neq v_i$ for all $i \in \{1, \ldots, s\} \setminus \{j_1, \ldots, j_r\}$.

Note that permuting the blocks of α does not change the linear dependence or independence of B. For easier notation we may therefore assume that

$$B = \{v_1, \ldots, v_r, a_{r+1}, \ldots, a_s, a'_1, \ldots, a'_s\}$$

For $c_1, \ldots, c_s, c'_1, \ldots, c'_s \in \mathbb{Z}_2$ consider the equation

$$\sum_{i=1}^{s} c_i a_i + \sum_{i=1}^{s} c'_i a'_i = 0. \quad (*)$$

We consider two cases

1. $c_i = 1$ for some $i \in \{1, \ldots, r\}$.
2. $c_i = 0$ for $i = 1, \ldots, r$

1. By reordering only the first r blocks of α we may assume that

$$c_1 = \ldots = c_k = 1$$

and

$$c_{k+1} = \ldots = c_r = 0$$

for some $k \in \{1, \ldots, r\}$.

Consider the subgraph G' of $G(\alpha)$ corresponding to the first k blocks of α

$$G' = (\{v_1, \ldots, v_k\}, \quad \{(v_i, v_j) \in E |\ i, j \leq k\}).$$

The graph G' is acyclic because $G(\alpha)$ is acyclic. By Lemma 4.13 there exists a vertex v_l $(1 \leq l \leq k)$ with in-degree zero. By a permutation of the blocks A_1 and A_l we may assume that the vertex v_1 has in-degree zero. We have

$$v_2(1,2) = \ldots = v_k(1,2) = (0,0).$$

Also

$$a_{r+1}(1,2) = \ldots = a_s(1,2) = (0,0),$$

because $a_{r+1}, \ldots, a_s \in \{e_{2r+1}, \ldots, e_n\}$. With Proposition 4.14 we have

$$v_1(1,2) = (1,1),$$

and it follows that

$$\sum_{i=1}^{s} c_i a_i(1,2) = (1,1).$$

Now consider the vectors $a'_1, \ldots, a'_s$. We have

$$a'_2(1,2) = \ldots = a'_s(1,2) = (0,0)$$

because $a'_2, \ldots, a'_s \in \{e_3, \ldots, e_n\}$. The first two entries in a'_1 are either $(1,0)$ or $(0,1)$. Therefore we have

$$(\sum_{i=1}^{s} c_i a_i + \sum_{i=1}^{s} c'_i a'_i)(1,2) \in \{(0,1),\ (0,1)\},$$

i.e. the equation $(*)$ can not be fulfilled with $c_i = 1$ for some $i \in \{1, \ldots, r\}$.

2. If $c_1 = \ldots = c_r = 0$, then $\sum_{i=r+1}^{s} c_i a_i + \sum_{i=1}^{s} c'_i a'_i = 0$. Since the vectors $a_{r+1}, \ldots, a_s, a'_1, \ldots, a'_s$ are distinct canonical vectors und thus linearly independent, we find that

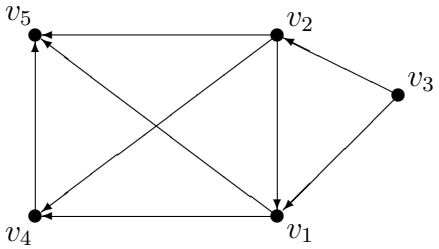
$$c_{r+1} = \ldots = c_s = c'_1 = \ldots, c'_s = 0.$$

It follows that the equation $(*)$ can only be fulfilled for $c_1 = \ldots = c_s = c'_1 = \ldots, c'_s = 0$ and therefore B is linearly independent. With Theorem 4.11 it follows that α is a logarithmic signature. □

Example 4.16. In the group $\mathbb{Z}_2^{10}$ we consider a logarithmic signature $\alpha = [A_1, A_2, A_3, A_4, A_5]$ of type $(4,4,4,4,4)$ in normal form with the blocks

$$\begin{aligned}
A_1 &= [\ 0,\ e_1,\ e_2,\ (1,1,0,0,0,0,1,0,1,1)\ ],\\
A_2 &= [\ 0,\ e_3,\ e_4,\ (0,1,1,1,0,0,1,1,1,0)\ ],\\
A_3 &= [\ 0,\ e_5,\ e_6,\ (1,0,1,0,1,1,0,0,0,0)\ ],\\
A_4 &= [\ 0,\ e_7,\ e_8,\ (0,0,0,0,0,0,1,1,0,1)\ ],\\
A_5 &= [\ 0,\ e_9,\ e_{10},\ (0,0,0,0,0,0,0,0,1,1)\ ]
\end{aligned}$$

The acyclic digraph $G(\alpha)$ associated to this logarithmic signature has five vertices $v_1, \ldots, v_5$ and there exists an arc from v_i to v_j if the entry $2j-1$ or $2j$ in the vector v_i is not zero for $i \neq j$, $i, j = 1, \ldots, 5$.



We immediately see that there is a vertex, v_3, with in-degree zero and a vertex, v_5, with out-degree zero. Note that the block A_5 corresponding to the vertex v_5 with out-degree zero is a subgroup. This cannot be observed just in this example but is a general property:

Corollary 4.17. *In a (normalized) logarithmic signature α of $\mathbb{Z}_2^n$ with blocks of size less or equal to 4 one block is a subgroup of $\mathbb{Z}_2^n$, more exactly all vertices with out-degree zero in $G(\alpha)$ correspond to a block in α that is a subgroup of $\mathbb{Z}_2^n$.*

Proof. Any block of size two is a subgroup. Therefore we need to consider the case, where all blocks are of size 4. Let $G(\alpha)$ be the digraph corresponding to α. By Theorem 4.15 we see that $G(\alpha)$ is acyclic and therefore one vertex v_i has out-degree zero for some $i \in \{1, \ldots, \frac{n}{2}\}$ by Proposition 4.13. With the definition of $G(\alpha)$ it follows that $v_i(2j-1, 2j) = (0, 0)$ for all $j \in \{1, \ldots, \frac{n}{2}\}$, $j \neq i$. By Proposition 4.14 it is $v_i(2i-1, 2i) = (1, 1)$ and thus $v_i = e_{2i-1} + e_{2i}$. Therefore $A_i = [0, e_{2i-1}, e_{2i}, v_i]$ is a subgroup of $\mathbb{Z}_p^n$. □

With different proof methods this corollary might also be found in [Sza04].

4.2.3 Factoring algorithm for block size 2 and 4

Here we use the results from the previous subsection to develop an efficient factoring algorithm for logarithmic signatures of elementary abelian 2-groups with blocks of size 2 and 4. In Theorem 4.15 we saw that every such logarithmic signature is fully described by the vectors v_i from the blocks of order 4. We combine these in a matrix that is similar but not equivalent to an adjacency matrix:

Definition 4.18. For a logarithmic signature $\alpha = [A_1, \ldots, A_s]$ in normal form and of type $(4, \ldots, 4)$ we define the *structural matrix*

$$M(\alpha) := (\ v_i(2j-1) + 2 \cdot v_i(2j)\ )_{i,j=1,\ldots,s}\,.$$

By Proposition 4.14 we see that all diagonal elements of $M(\alpha)$ are equal to 3. Corollary 4.17 shows that there exists $i \in \{1, \ldots, s\}$ such that row i is equal to $3 \cdot e_i$. Moreover we have

Corollary 4.19. *Let $n \in \mathbb{N}$ and $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signature of type $(4, \ldots, 4)$ in normal form. Then there exists a permutation matrix P such that $P \cdot M(\alpha) \cdot P^{-1}$ is an upper triangular matrix with diagonal elements equal to 3.*

Proof. Let $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signature of type $(4, \ldots, 4)$ in normal form with the corresponding structural matrix $M(\alpha)$. We use an inductive argument on s. First if $M(\alpha)$ is a 1×1 matrix there is nothing to prove. Now let the corollary be true for some $s \in \mathbb{N}$. And let $M(\alpha)$ be an $s+1 \times s+1$ matrix. Then by Corollary 4.17 one block of α is a subgroup and the corresponding row i in $M(\alpha)$ contains only zeros except for the diagonal entry which is 3. The matrix M' derived from $M(\alpha)$ by deleting row and column i is an $s \times s$ matrix and also the structural matrix of a logarithmic signature α' with s blocks. Let P_s be the permutation matrix such that by induction $P_s \cdot M' \cdot P_s$ is an upper triangular matrix with diagonal elements equal to 3. Let P be the $(s+1) \times (s+1)$ permutation matrix that permutes i and s. Then

$$P_{s+1} = \begin{pmatrix} & & 0 \\ & P_s & \vdots \\ & & 0 \\ 0 \ldots 0 & & 1 \end{pmatrix} \cdot P$$

is the permutation matrix that transforms $M(\alpha)$ into an upper triangular matrix with diagonal elements equal to 3. □

Example 4.20. We consider Example 4.16 with

$$M(\alpha) := \begin{pmatrix} 3 & 0 & 0 & 1 & 3 \\ 2 & 3 & 0 & 3 & 1 \\ 1 & 1 & 3 & 0 & 0 \\ 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 0 & 3 \end{pmatrix}.$$

With the permutation matrix

$$P := \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

we get

$$P \cdot M(\alpha) \cdot P^{-1} = \begin{pmatrix} 3 & 1 & 1 & 0 & 0 \\ 0 & 3 & 2 & 3 & 1 \\ 0 & 0 & 3 & 1 & 3 \\ 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 0 & 3 \end{pmatrix}.$$

Note that this matrix is the structural matrix of the normal form of α with blocks 1 and 3 exchanged, denoted by $\beta = [A_3, A_2, A_1, A_4, A_5]$. And the upper triangular form of the matrix leads to a simple factoring approach: To factor an element $x \in Z_2^{10}$ consider the first two entries of x. The first block of β is the only block that contains vectors with non-zero entries in the first and second position. The correct factor of x from the first block is the one vector that coincides with x in the first two positions. Let this vector be b_1. Then the correct factor from the second block is the one vector that coincides with $x - b_1$ in the third and forth position and so on.

It follows an implementation of this idea:

GAP: FactorByLogSig4

Input: logSig, a normalized logarithmic signature of type $(4, \ldots, 4)$ of $\mathbb{Z}_p^n$, where $n \in \mathbb{N}$, a vector $x \in \mathbb{Z}_p^n$
Output: Fact, the factorization vectors of x with respect to logSig

```
FactorByLogSig4:=function(logSig, x)

LS:=NormalFormLogSig(logSig);

S:= []; M:= []; Fact:= [];
y:= StructuralCopy(x);
```

```
# (1) Build the structural matrix M
for i in [1..n/2] do
    Add(M,[]);
    for j in [1..n/2] do
      if logSig[i][4]{[2*j-1,2*j]} = Z(2)*[0,0] then
        M[i][j]:=0;
        else M[i][j]:=1;
      fi;
    od;
od;

# (2) Find the permutation (matrix)
for i in [1..n/2] do
    k:=PositionProperty(M,x -> Sum(x)=1);
    Add(S,k);
    for j in [1..n/2] do
       M[j][k]:=0;
    od;
od;

# (3) Compute the factors
for i in Reversed(S) do
    Fact[i]:=logSig[i][IntFFE(y[2*i-1]) + 2*IntFFE(y[2*i])+1];
    y:= y - Fact[i];
od;

return Fact;
end;
```

The running time of this algorithm is dominated by the running time $O(n^2 \cdot s \cdot r_\alpha) = O(n^3)$ of NormalFormLogSig. The other parts each need time $O(n^2)$ or less. This running stays the same if the algorithm is extended to take logarithmic signatures with blocks of size 2 and 4. Alltogether we have

Theorem 4.21. *In elemtary abelian p-groups logarithmic signatures with blocks of size less than or equal to 4 are tame.*

4.3 Factoring logarithmic signatures in $\mathbb{Z}_p^n$ using linear equations

In the previous section we showed results on the structure and factorability of certain logarithmic signatures in elementary abelian 2-groups. Here we introduce a factoring approach using linear equations that applies to all logarithmic signatures of elementary abelian p-groups. Consider the following matrix notation.

Definition 4.22. Let α be a logarithmic signature of $\mathbb{Z}_p^n$. The notation $\alpha = (A_1|\ldots|A_s)$ refers to the augmented matrix with its columns equal to the vectors from α, i.e.

$$\alpha = (a_{11}^t, a_{12}^t, \ldots, a_{1r_1}^t|\ a_{21}^t, \ldots, a_{sr_s}^t),$$

where $A_i = [a_{i1}, \ldots, a_{ir_i}]$. This is an $n \times \ell(\alpha)$ matrix with elements from $\mathbb{Z}_p^n$ and rank n. Note that always $n < \ell(\alpha)$.

Example 4.23. Consider the example on page 68 of a logarithmic signature α of type $(9, 9)$ from $\mathbb{Z}_3^4$. The matrix of α is

$$\alpha = \left(\begin{array}{ccccccccc|ccccccccc} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 2 & 0 & 0 & 1 & 1 & 2 & 2 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 2 & 0 & 0 & 1 & 1 & 2 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 2 & 1 & 2 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right).$$

Every such matrix defines a system of n linear equations with $\ell(\alpha)$ unknowns that is under-determined. By Lemma 4.2 the rank of this matrix is always n. Therefore the set of solutions is of size $p^{\ell(\alpha)-n}$. For every $g \in \mathbb{Z}_p^n$ exactly one of the solutions of the equation system

$$\alpha \cdot x = g$$

describes the factorization of g with respect to α. Denote this solution by x_g. Unfortunately $p^{\ell(\alpha)-n} \geq p^{p \cdot n - n}$, thus a search through the set of solutions takes at least as long as a complete search through all p^n possible factorizations of α.

Since α is a logarithmic signature, the solution vector x_g is of a certain form and there is only one such vector in the set of solutions. This vector contains exactly s entries equal to 1 and all other entries are zero. Furthermore the

nonzero entries are distributed such that among the entries in x_g corresponding to one block exactly one is equal to 1.

In Example 4.23 we consider the set of solutions for $\alpha \cdot x = (1,1,0,0)^t$. The solution corresponding to the factorization of $(1,1,0,0)$ is

$$x_g = (1,0,0,0,0,0,0,0,0 \mid 0,1,0,0,0,0,0,0,0).$$

Other solutions are for example

$$x_1 = (0,0,0,0,0,0,0,0,0 \mid 0,1,0,0,0,0,0,0,0)$$
$$x_2 = (1,0,0,0,0,0,0,0,0 \mid 0,0,2,0,0,0,0,0,0)$$
$$x_3 = (1,0,0,0,0,0,0,0,0 \mid 1,0,1,1,0,0,0,1,0)$$
$$x_4 = (0,1,1,1,1,1,1,1,1 \mid 0,1,0,1,1,0,0,0,0)$$

Based on this observation we modify the equation $\alpha \cdot x = g$ in order to be able to extract x_g from the solutions: For each block A_i of α we add a row that consists of zeros except for the entries corresponding to the block, which we set equal to 1:

Let $m_0 = 0$ and $m_k = \sum_{i=1}^{k} r_i$ for $k = 1, \ldots, s$. Then for $i = 1, \ldots, s$ add the row

$$\sum_{j=m_{i-1}+1}^{m_i} e_i.$$

Furthermore we extend g to length $n+s$ by adding ones. We derive the extended matrix equation

$$\begin{pmatrix} & & & & & \\ A_1 \mid & A_2 \mid & & \ldots & & \mid A_s \\ & & & & & \\ 1\ldots1 & 0 & & \ldots & & 0 \\ 0\ldots0 & 1\ldots1 & 0 & \ldots & & 0 \\ & & \ldots & & & \\ 0 & & \ldots & & 0 & 1\ldots1 \end{pmatrix} \cdot x = \begin{pmatrix} g(1) \\ \vdots \\ g(n) \\ 1 \\ \\ \vdots \\ 1 \end{pmatrix}$$

Note that by adding these equations we reduce the solution vectors to such vectors where the sum of the entries corresponding to one block always equals 1 modulo p, i.e.

$$\sum_{j=m_{i-1}+1}^{m_i} x(j) \mod p = 1.$$

For Example 4.23 we have the extended matrix equation

$$\left(\begin{array}{ccccccccc|ccccccccc} 0&1&0&0&1&0&0&1&0&0&1&2&0&0&1&1&2&2\\ 0&0&1&0&0&1&0&0&1&0&1&2&0&0&1&1&2&2\\ 0&0&0&0&0&0&0&0&0&0&0&0&1&2&1&2&1&2\\ 0&0&0&1&1&1&2&2&2&0&0&0&0&0&0&0&0&0\\ 1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1 \end{array}\right) \cdot x_g = \left(\begin{array}{c} 1\\1\\0\\0\\1\\1 \end{array}\right)$$

Note that now the vectors x_1, x_2 and x_4 are no longer in the set of solutions.

As the new equations are linearly independent to the equations directly defined by α, the size of the set of solutions reduces to $p^{\ell(\alpha)-(n+s)}$. This is not significantly less than before, but x_g now has a special property by which it can be identified among all solutions.

Proposition 4.24. *Let α be a logarithmic signature of $\mathbb{Z}_p^n$ and $g \in \mathbb{Z}_p^n$. Consider the set M of solutions of the extended system of equations derived from $\alpha \cdot x = g$. Let x_g be the unique vector that corresponds to the factorization of g with respect to α.*

Then x_g is of Hamming weight s and it is the unique vector in M with minimal Hamming weight.

Proof. 1. As x_g has one nonzero entry per block it is of Hamming weight s.

2. We know that $\sum_{j=m_{i-1}+1}^{m_i} x(j) \mod p = 1$. It follows that at least one of the entries in x corresponding to each block is nonzero. Thus every solution vector has Hamming weight at least s.

3. Assume that $y \in M$ with Hamming weight s. Thus for each block of α there is exactly one nonzero entry in y. These can be any nonzero element in $\mathbb{Z}_p$. But since $\sum_{j=m_{i-1}+1}^{m_i} x(j) \mod p = 1$, it follows that every nonzero entry is equal to 1. Now y defines a factorization of g with respect to α. As these are unique, we have $y = x_g$.

Therefore x_g is the unique minimal vector in M with minimal Hamming weight s. □

Our idea is to find a polynomial link between this description of a logarithmic signature and some computational problem that is known to be either difficult or efficient to solve. Our reduction from 3SAT or the shortest vector problem to solving linear systems derived from logarithmic signatures was not yet successful and could be focus of further research.

4.4 Factoring for Rédei groups

In this section we develop a reduction algorithm for a class of logarithmic signatures in $\mathbb{Z}_p^n$ that leads to a polynomial-time factoring algorithm.

In the previous section we introduced the matrix notation for logarithmic signatures in elementary abelian p-group. Note that elementary row operations on the matrix α describe an automorphism applied to α. So we can modify α via Gaussian elimination until we get a matrix α^r in reduced row echelon form with the result that the factoring problem for the logarithmic signatures α and α^r have the same complexity, see page 55.

It is immediately clear that the reduced row echelon form of a logarithmic signature is unique but several different logarithmic signatures may have the same reduced row echelon form.

Example 4.25. Consider Example 4.7 of a logarithmic signature α of type $(4, 4, 8)$ from $\mathbb{Z}_2^7$. In the notation of this section we have

$$\alpha = \left(\begin{array}{cccc|cccc|cccccccc} 0&1&0&1&0&0&0&1&0&0&0&0&0&0&1&1\\ 0&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&1&0&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&0&0&1&1&0&1&0&0&0&0&0&1\\ 0&0&0&1&0&0&1&1&0&0&1&0&0&1&0&0\\ 0&0&0&0&0&0&0&1&0&0&0&1&0&1&1&1\\ 0&0&0&0&0&0&1&1&0&0&0&0&1&0&1&0 \end{array}\right)$$

leading to the reduced row echelon form

$$\alpha^r = \left(\begin{array}{cccc|cccc|cccccccc} 0&1&0&0&0&0&0&0&0&0&1&1&1&0&1&0\\ 0&0&1&0&0&0&0&0&0&0&1&0&1&1&1&0\\ 0&0&0&1&0&0&0&0&0&0&1&0&1&1&1&0\\ 0&0&0&0&0&1&0&0&0&0&0&1&0&1&1&1\\ 0&0&0&0&0&0&1&0&0&0&0&1&1&1&0&1\\ 0&0&0&0&0&0&0&1&0&0&0&1&0&1&1&1\\ 0&0&0&0&0&0&0&0&0&1&0&0&1&0&1&1 \end{array}\right)$$

We make an observation: The rank of the matrix $(A_1|A_2)$ is 6 and therefore the last row of $(A_1, A_2)^r$ contains only zeros. If we want to factor an element

$(*,*,*,*,*,*,1) \in \mathbb{Z}_2^7$ with respect to α^r we immediately know that only one half of the elements of block A_3 are possible factors: the vectors with last position equal to 1.

This observation leads to the following definition which extends the definition of the Rédei property in Chapter 9.2 of [Sza04].

Definition 4.26. Let $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signature and $i \in \{1, \ldots, s\}$. A block A_i of α is called a *Rédei block* if $\bigcup_{j \neq i} A_j$ does not contain a basis of $\mathbb{Z}_p^n$, i.e.

$$\mathrm{Rank}(A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_s) < n.$$

Definition 4.27. Let $m > 1$ be a natural number. A finite elementary abelian group $\mathbb{Z}_p^n$ is said to have the *Rédei m-property*, if every logarithmic signature with m blocks has a Rédei block.

If $\mathbb{Z}_p^n$ has the Rédei m-property for all $1 < m < n$, then we say that $\mathbb{Z}_p^n$ has the *Rédei property*.

If only the logarithmic signatures of a certain class $\mathcal{A}$ of logarithmic signatures have a Rédei block, then we say that $\mathbb{Z}_p^n$ has the *Rédei property for $\mathcal{A}$*.

Note that in [Sza04] the Rédei 2-property was studied, but without the focus on factoring. Moreover here we are interested in logarithmic signatures with more than two blocks such that the length $\ell(\alpha)$ of the logarithmic signatures is polynomial in n.

Finding a Rédei block is efficient as the following pseudo-code shows.

GAP: FindRedeiBlockLogSig

Input: logSig, the $n \times \ell(\alpha)$ matrix of a normalized logarithmic
signature α of $\mathbb{Z}_p^n$
type, the type of logSig
Output: i, the index of a Rédei block of logSig

```
FindRedeiBlockLogSig:=function(logSig, type)

local i,Block;

for i in [1..Length(type)] do
```

```
  Block:=RemoveBlock(logSig, type, i);

  if RankMat(logSig) = n then

    AddBlock(logSig, Block, type, i);

    else return i;

  fi;
od;

end;
```

The running time of this algorithm is dominated by RankMat which uses Gaussian elimination as a subroutine and needs $O(n^2 \cdot \ell(\alpha))$ operations. With s iterations we get a rough estimation of the polynomial running time $O(n^2 \cdot s^2 \cdot r_\alpha)$.

Lemma 4.28. *Let $\alpha = (A_1 \mid \ldots \mid A_s)$ be a logarithmic signature of $\mathbb{Z}_p^n$ with a Rédei block A_i of size $|A_i| = p^{t_i}$. Consider the logarithmic signature*

$$\alpha' = (A_1 \mid \ldots \mid A_{i-1} \mid A_{i+1} \mid \ldots \mid A_s \mid A_i).$$

Then - with a permutation of the elements in A_i - the last row of α'^r is of the form

$$(\underbrace{0\ 0\ 0\ \ldots\ 0\ 0\ 0}_{\ell(\alpha)-p^{t_i}}\ \underbrace{0\ \ldots\ 0}_{p^{t_i}-1}\ \underbrace{1\ \ldots\ 1}_{p^{t_i}-1}\ldots\ \underbrace{p-1\ \ldots\ p-1}_{p^{t_i}-1}).$$

Proof. Let $\alpha = (A_1 \mid \ldots \mid A_s)$ be a logarithmic signature of $\mathbb{Z}_p^n$ with a Rédei block A_i of size $|A_i| = p^{t_i}$. W.l.o.g let $i = s$, i.e. $\alpha = \alpha'$. Consider the *matrix*

$$A := (A_1 \mid \ldots \mid A_{s-1}).$$

Since A_s is a Rédei block, we have

$$\mathrm{Rank}(A) < n,$$

and therefore the last row of the reduced row echelon form of A contains only zeros. Therefore the last row of α^r starts with $\ell(\alpha) - p^{t_s}$ zeros.

Now consider the last p^{t_s} positions of the last row of

$$\alpha^r := (A_1^r \mid \ldots \mid A_s^r).$$

For $x \in \mathbb{Z}_p$ let k_x be the number of positions equal to x. Suppose that $k_x < p^{t_s-1}$ for some $x \in \mathbb{Z}_p$. Then the set $A_1^r + \ldots + A_s^r$ contains

$$p^{t_1} \cdot \ldots \cdot p^{t_{s-1}} \cdot k_x < p^{n-1}$$

vectors with the last position equal to x. This is a contradiction to a^r being a logarithmic signature which implies that $A_1^r + \ldots + A_s^r$ contains exactly p^{n-1} vectors with the last position equal to x.

It follows that $k_x \geq p^{t_s-1}$ for all $x \in \mathbb{Z}_p$. We have

$$p^{t_s} = |A_s^r| = \sum_{x \in \mathbb{Z}_p} k_x \geq |\mathbb{Z}_p| \cdot p^{t_s-1} = p^{t_s}$$

and therefore $k_x = p^{t_s-1}$ for all x, i.e. every element of $\mathbb{Z}_p$ occurs exactly p^{t_s-1} times among the last p^{t_s} elements of the last row of α^r. □

Proposition 4.29. *Let $\alpha = (A_1|\ldots|A_s)$ be a logarithmic signature of $\mathbb{Z}_p^n$ with a Rédei block. Then the factoring problem for α is polynomial-time reducible to the factoring problem for a logarithmic signature of $\mathbb{Z}_p^{n-1}$ with s or $s-1$ blocks.*

Proof. Let $\alpha = (A_1|\ldots|A_s)$ be a logarithmic signature of $\mathbb{Z}_p^n$. W.l.o.g let α be in reduced row echelon form with Rédei block A_s. We consider the problem of finding the factorization for some vector $x \in \mathbb{Z}_p^n$ with respect to α under the assumption that the factoring problem for logarithmic signatures of $\mathbb{Z}_p^{n-1}$ with s or $s-1$ blocks is efficiently solvable.

Let

$$A_0 := [\ y \in A_s \mid y(n) = x(n)\ ],$$

i.e. A_0 is constructed from A_s by taking all vectors with last position equal to $x(n)$. By Lemma 4.28 we have $|A_0| = p^{t_s-1}$.

Note that by Lemma 4.28 the last position in all vectors from A_1 to A_{s-1} is zero. Then the set

$$A_1 + \ldots + A_{s-1} + A_0$$

is the subset of $A_1 + \ldots + A_s = \mathbb{Z}_p^n$ that contains all p^{n-1} elements of $\mathbb{Z}_p^n$ with the last position equal to $x(n)$. Furthermore one of the vectors of A_0 is the factor from A_s in the factorization of x with respect to α.

For $i = 0, 1, \ldots, s-1$ let

$$A'_i := [\ a(1, \ldots, n-1) \mid a \in A_i\ ],$$

i.e. A'_i is constructed from A_i by projection on the first $n-1$ positions of each vector.

If $|A'_0| > 1$, then

$$A'_1 + \ldots + A'_{s-1} + A'_0$$

contains all p^{n-1} elements of $\mathbb{Z}_p^{n-1}$ and

$$|A'_1| \cdot \ldots \cdot |A'_{s-1}| \cdot |A'_0| = p^{n-1}.$$

Therefore $\alpha'' = (\ A'_1 \mid \ldots \mid A'_{s-1} \mid A'_0\ )$ is a logarithmic signature of $\mathbb{Z}_p^{n-1}$ with s blocks. By assumption factoring with respect to α'' is efficient:

Let

$$x(1, \ldots, n-1) = a'_1 + \ldots + a'_{s-1} + a'_0$$

be the factorization of $x(1, \ldots, n-1)$ with respect to α'', then

$$x = a_1 + \ldots + a_s$$

is the factorization for x with respect to α, where

$$a_i := (\ a_i'^t \mid 0\ ) \in A_i$$

for $1 \leq i \leq s-1$ and

$$a_s = (\ a_0'^t \mid x(n)\ ) \in A_s.$$

If $|A'_0| = 1$, then

$$A'_1 + \ldots + A'_{s-1}$$

contains all p^{n-1} elements of $\mathbb{Z}_p^{n-1}$ and

$$|A'_1| \cdot \ldots \cdot |A'_{s-1}| = p^{n-1}.$$

Therefore $(\ A'_1 \mid \ldots \mid A'_{s-1}\ )$ is a logarithmic signature of $\mathbb{Z}_p^{n-1}$ with $s-1$ blocks. By assumption factoring with respect to α'' is efficient:

Let $A'_0 = \{a\}$ and let

$$x(1, \ldots, n-1) - a = a'_1 + \ldots + a'_{s-1}$$

be the factorization of $x(1, \ldots, n-1) - a$ with respect to α'', then

$$x = a_1 + \ldots + a_s$$

is the factorization for x with respect to α, where

$$a_i := (\ a_i'^t \mid 0 \ ) \in A_i$$

for $1 \leq i \leq s-1$ and

$$a_s = (\ a^t \mid x(n) \ ) \in A_s.$$

□

The proposition shows that we can efficiently reduce the problem of factoring from logarithmic signatures in $\mathbb{Z}_p^n$ to logarithmic signatures in $\mathbb{Z}_p^{n-1}$ if we start with a logarithmic signature that has a Rédei block. In case we obtain a logarithmic signature with a Rédei block in this process, we can reduce further. If we iterate this process and obtain a logarithmic signature with a Rédei block in every step, then after n reduction steps we obtain the desired factorization.

It follows a pseudo-code implementation of this idea:

GAP: FactorByRedeiLogSig

Input: logSigx, the concatenation of the matrix of a partial normalized logarithmic signature of $\mathbb{Z}_p^n$ and a vector x
type, the type of logSig
index, an iteration counter

Output: Fact, the factorization of x w.r.t. logSig or an error message if FindRedeiBlockLogSig returns an error in some iteration

```
FactorByRedeiLogSig:=function(logSigx, type, index)

local i,j,k,rb,LS,Fact,y;

# Build matrix
LS:=StructuralCopy(logSigx);
y:= StructuralCopy(x);
s:=Length(type);

# Find the Redei block
rb:=FindRedeiBlockLogSig(LogSig);

# RotateBlocks
PermutedLS(LS, (rb, s));
```

```
Permuted(type, (rb, s));

# Compute the row echelon form
LS:=TriangulizeMat(LS);

# Delete vectors
j:=sum(type); #=Length(LS)-1;
k:=j-type[s];
DeleteColumns(LS{[j..k]}, i ->  i[n] = LS[index][j+1]);
type[s]=type[s]/p;

# Test whether factorization already found
if index = 1 then
  return LS;
  else FactorByRedeiLogSig(LS, type, index-1);

end;
```

Note that the running time this algorithm is dominated by the Gaussian elimination used by FindRedeiBlockLogSig and also TriangulizeMat. Therefore we can approximate the running time with $O(n^2 \cdot s^2 \cdot r_\alpha)$. An example illustrates this algorithm.

Example 4.30. Consider Example 4.25 of a logarithmic signature α of type $(4, 4, 8)$ from $\mathbb{Z}_2^7$ and the problem of finding a factorization for the vector

$$x = (0, 1, 0, 1, 1, 1, 1).$$

We take the matrix $(A_1 \mid A_2 \mid A_3 \mid x^t\ ) =$

$$\left(\begin{array}{cccc|cccc|cccccccc|c}
0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
\hline
1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 &
\end{array}\right),$$

where we label every column according to its occurrence in each block of α. The rank of $(A_1 \mid A_2)$ is 6, thus the third block of α is a Rédei block. We convert this matrix to the reduced row echelon form

$$(A_1 \mid A_2 \mid A_3 \mid x^t)^r = \left(\begin{array}{cccc|cccc|cccccccc|c} 0&1&0&0&0&0&0&0&0&0&1&1&1&0&1&0&1\\ 0&0&1&0&0&0&0&0&0&0&1&0&1&1&1&0&1\\ 0&0&0&1&0&0&0&0&0&0&1&0&1&1&1&0&0\\ 0&0&0&0&0&1&0&0&0&0&0&1&0&1&1&1&1\\ 0&0&0&0&0&0&1&0&0&0&0&1&1&1&0&1&0\\ 0&0&0&0&0&0&0&1&0&0&0&1&0&1&1&1&1\\ 0&0&0&0&0&0&0&0&\underline{0}&1&\underline{0}&\underline{0}&1&\underline{0}&1&1&\underline{0}\\ \hline 1&2&3&4&1&2&3&4&1&2&3&4&5&6&7&8& \end{array}\right)$$

Since the last entry in x^r is 0, only the vectors $1, 3, 4$ and 5 are possible factors from the third block. Thus we can discard all columns with last entry equal to 1 from this block. We also delete the last row and obtain a logarithmic signature of $\mathbb{Z}_p^6$:

$$\left(\begin{array}{cccc|cccc|cccc|c} 0&1&0&0&0&0&0&0&0&1&1&0&1\\ 0&0&1&0&0&0&0&0&0&1&0&1&1\\ 0&0&0&1&0&0&0&0&0&1&0&1&0\\ 0&0&0&0&0&1&0&0&0&0&1&1&1\\ 0&0&0&0&0&0&1&0&0&0&1&1&0\\ 0&0&0&0&0&0&0&1&0&0&1&1&1\\ \hline 1&2&3&4&1&2&3&4&1&3&4&6& \end{array}\right)$$

Now the first block is a Rédei block. And we obtain the reduced row echelon form (keeping the order of the blocks)

$$\left(\begin{array}{cccc|cccc|cccc|c} 0&1&1&0&0&1&0&0&0&0&0&0&1\\ 0&1&1&0&0&0&1&0&0&0&0&0&0\\ 0&1&1&0&0&0&0&1&0&0&0&0&1\\ 0&0&1&0&0&0&0&0&0&1&0&1&1\\ 0&1&1&0&0&0&0&0&0&0&1&1&0\\ 0&0&\underline{1}&\underline{1}&0&0&0&0&0&0&0&0&\underline{1}\\ \hline 1&2&3&4&1&2&3&4&1&3&4&6& \end{array}\right)$$

Since the last entry in the last column is 1, we can discard all columns with last entry equal to 0 from the first block.

$$\left(\begin{array}{cc|cccc|cccc|c} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ \hline 3 & 4 & 1 & 2 & 3 & 4 & 1 & 3 & 4 & 6 & \end{array}\right)$$

Now the third block is a Rédei block. And we obtain the reduced row echelon form

$$\left(\begin{array}{cc|cccc|cccc|c} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \underline{1} & \underline{1} & 0 & \underline{1} \\ \hline 3 & 4 & 1 & 2 & 3 & 4 & 1 & 3 & 4 & 6 & \end{array}\right) \longrightarrow \left(\begin{array}{cc|cccc|cc|c} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ \hline 3 & 4 & 1 & 2 & 3 & 4 & 3 & 4 & \end{array}\right)$$

Now the second block is a Rédei block. And we obtain the reduced row echelon form

$$\left(\begin{array}{cc|cccc|cc|c} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \underline{1} & \underline{1} & 0 & 0 & 0 & \underline{1} \\ 0 & 0 & \underline{0} & 1 & \underline{0} & 1 & 0 & 0 & \underline{0} \\ \hline 3 & 4 & 1 & 2 & 3 & 4 & 3 & 4 & \end{array}\right) \longrightarrow \left(\begin{array}{cc|c|cc|c} \underline{1} & 0 & 0 & 0 & 0 & \underline{1} \\ 0 & 0 & 0 & 1 & \underline{0} & \underline{0} \\ \hline 3 & 4 & 3 & 3 & 4 & \end{array}\right)$$

Since the last column is $(1,0)^t$, we can discard vector 4 from the first block and vector 3 from the third block. The process terminates with

$$\left(\begin{array}{c|c|c|c} 3 & 3 & 4 & \end{array}\right)$$

Thus the factorization of x with respect to the original logarithmic signature α is $A_1[3] + A_2[3] + A_3[4]$, i.e.

$$x = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}.$$

Limit and extension of the factoring algorithm This algorithm computes a correct factorization only if every step produces a logarithmic signature with a Rédei block. Fortunately, we can extend the algorithm such that it outputs the correct factorization for a larger class of logarithmic signatures.

Look at the following case:

Let $\alpha = (A_1, \ldots, A_s)$ be a logarithmic signature of $\mathbb{Z}_p^n$ without a Rédei block - but with the property that $\text{Rank}(A_1, \ldots, A_{s-2}) < n$, i.e. in the last row of the row echelon form of α the entry 1 only occurs in the entries belonging to the last *two* blocks. In this case we can split the problem of finding a factorization into only two subproblems, of which only one has a solution.

Suppose that we want to find the factorization for $(z_1, \ldots, z_{n-1}, 0)$ with respect to α. And let $(\alpha|z)$ be in row echelon form

$$\left(\begin{array}{c|c|c|c|c|c} & & & & & z_1 \\ * & \ldots & * & * & * & \vdots \\ & & & & & z_{n-1} \\ 0 \ldots 0 & & 0 \ldots 0 & 0 \ldots 0\ 1 \ldots 1 & 0 \ldots 0\ 1 \ldots 1 & 0 \end{array}\right).$$

We divide the last two blocks of α according to the last row of this matrix as follows

$$\begin{aligned} A_{s-1}^0 &:= \{x \in A_{s-1} \mid x[n] = 0\} \\ A_{s-1}^1 &:= \{x \in A_{s-1} \mid x[n] = 1\} \\ A_s^0 &:= \{x \in A_s \mid x[n] = 0\} \\ A_s^1 &:= \{x \in A_s \mid x[n] = 1\} \end{aligned}$$

Since α is a logarithmic signature, there exists exactly one factorization for z. And this factorization can be found in one and only one of the *partial* logarithmic signatures

$$\alpha_1 = [A_1, \ldots, A_{s-2}, A_{s-1}^0, A_s^0]$$

and

$$\alpha_2 = [A_1, \ldots, A_{s-2}, A_{s-1}^1, A_s^1].$$

It is not clear which of the two partial logarithmic signatures contains the correct factorization. So we have to work with both. In one case we will obtain the desired factorization; in the other case we will get no solution. Note that the last row of the matrices $(\alpha_1|z)$ and $(\alpha_2|z)$ does not contain any further information that helps to find a factorization for z. Therefore we make a projection on the first $n-1$ entries in each column:

$$\alpha_1' = [A_1', \ldots, A_{s-2}', A_{s-1}^{0'}, A_s^{0'}]$$

and

$$\alpha_2' = [A_1', \ldots, A_{s-2}', A_{s-1}^{1'}, A_s^{1'}],$$

where $A' := \{a(1, \ldots, n-1) \mid a \in A\}$ for $A \subseteq \mathbb{Z}_p^n$.

Now we reduced the problem of finding the factorization for z in $\mathbb{Z}_p^n$ to the problem of finding a factorization for $z' = z(1, \ldots, n-1)$ in $\mathbb{Z}_p^{n-1}$ via two partial logarithmic signatures. This works very similar to factoring with respect to logarithmic signatures, but with the special case that in one of the two cases we get an error message.

We simply adapt the notion of a Rédei block for the reduced partial logarithmic signatures. And we take both α_1' and α_2' as input for the algorithm `FactorByRedeiLogSig` to further reduce the problem of finding the factorization for z. We call this *splitting* process.

With this method we may extend the `FactorByRedeiLogSig` algorithm such that it includes the splitting process. If the input logarithmic signature is in an adequate form (with a Rédei block in every iteration), the algorithm takes n iterations to compute a factorization. If there is a splitting in every iteration, the algorithm takes 2^n iterations, which would make the running time exponential in the input parameter n. But for logarithmic signatures for which the splitting occurs only in a constant number of steps, say $k \in \mathbb{N}$, the running time would increase with respect to the optimal case by the constant factor 2^k.

4.5 Constructing non-Rédei logarithmic signatures

The algorithm FactorByRedeiLogSig factors a large class of logarithmic signatures in elementary abelian groups. Here we introduce a new method for manipulating logarithmic signatures called selective shifts. With selective shifts we construct a series of logarithmic signatures that are not efficiently factorable by this algorithm. Furthermore these logarithmic signatures are neither transversal, nor periodic, i.e. they do not belong to a class of logarithmic signatures for which other factoring algorithms are known.

For the construction we need a special operation.

Definition 4.31. Let $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signature of a finite abelian group. For some $i_1, \ldots, i_k \in \{1, \ldots, s\}$ $(k < n)$ let

$$A_{i_1} + \ldots + A_{i_k}$$

be periodic with the set of periods M (see Page 30). Take $m \in M$ and add m to some element of block A_l, where $l \neq i_1, \ldots, i_k$.

This operation is called a *selective shift on* α.

Lemma 4.32. *A selective shift on a logarithmic signature yields a logarithmic signature.*

Proof. Let $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signature of a finite abelian group G. For $i_1, \ldots, i_k \in \{1, \ldots, s\}$ $(k < n)$ let $A_{i_1} + \ldots + A_{i_k}$ be periodic with the set of periods M. Take $m \in M$ and add m to some element of block A_l $(l \neq i_1, \ldots, i_k)$. W.l.o.g let $l = s$ and add m to the last element of $A_s = [a_1, \ldots, a_r]$. Let

$$A'_s := [a_1, \ldots, a_{r-1}, a_r + m].$$

Note that m is also a period of the set $A_1 + \ldots + A_{s-1}$. Then

$$\begin{aligned}
& A_1 + \ldots + A_{s-1} + A'_s \\
= \; & (A_1 + \ldots + A_{s-1}) + [a_1, \ldots, a_{r-1}] \;\cup\; (A_1 + \ldots + A_{s-1}) + a_r + m \\
= \; & (A_1 + \ldots + A_{s-1}) + [a_1, \ldots, a_{r-1}] \;\cup\; (A_1 + \ldots + A_{s-1}) + a_r \\
= \; & A_1 + \ldots + A_{s-1} + A_s \\
= \; & G
\end{aligned}$$

Thus $[A_1, \ldots, A_{s-1}, A'_s]$ is a logarithmic signature of G. □

We start with a logarithmic signature $\alpha = [A, B]$ of type $(9, 6)$ in $\mathbb{Z}_2^{15}$ from p. 246 in [Sza04], where $< A > = < B > = \mathbb{Z}_2^{15}$. This is the shortest known example of a logarithmic signature without a Rédei block. Also none of the blocks is periodic. The explicit construction can be found in the appendix on page 132.

In this example, A contains the canonical basis $e_1, \ldots, e_{15}$ and also the elements

$$\begin{aligned} e_2, &\quad e_1 + e_2, \\ e_3, &\quad e_1 + e_3, \\ e_6, &\quad e_1 + e_6, \\ e_2 + e_6, &\quad e_1 + e_2 + e_6. \end{aligned}$$

The block B contains a different basis of $\mathbb{Z}_2^{15}$, which we do not consider. Now we expand α in three steps using specific selective shifts:

1. Lift all elements of A and B to $\mathbb{Z}_2^{19}$ by adding four zeros to each vector. Take a third block $C =< e_{16}, \ldots, e_{19} >$ (subgroup of $\mathbb{Z}_2^{19}$). Then $[A, B, C]$ is a logarithmic signature of $\mathbb{Z}_p^{19}$.

2. Block C is a subgroup and therefore it is periodic in $\mathbb{Z}_2^{19}$. We do eight selective shifts on the block A

$$\begin{aligned} e_2 &\rightarrow e_2 + e_{16} \\ e_1 + e_2 &\rightarrow e_1 + e_2 + e_{16} \\ e_3 &\rightarrow e_3 + e_{17} \\ e_1 + e_3 &\rightarrow e_1 + e_3 + e_{17} \\ e_6 &\rightarrow e_6 + e_{18} \\ e_1 + e_6 &\rightarrow e_1 + e_6 + e_{18} \\ e_2 + e_6 &\rightarrow e_2 + e_6 + e_{19} \\ e_1 + e_2 + e_6 &\rightarrow e_1 + e_2 + e_6 + e_{19} \end{aligned}$$

Denote the resulting block by A'. By Lemma 4.32 the sequence $[A', B, C]$ is a logarithmic signature of $\mathbb{Z}_2^{19}$. Also $< A' + B > = \mathbb{Z}_p^{19}$ and $A' + B$ is periodic with period e_1.

3. We make selective shifts on the block C by e_1:

$$\begin{aligned} e_{16} + e_{17} + e_{18} + e_{19} &\rightarrow e_{16} + e_{17} + e_{18} + e_{19} + e_1 \\ e_{17} + e_{18} + e_{19} &\rightarrow e_{17} + e_{18} + e_{19} + e_1 \end{aligned}$$

Denote the resulting block by C'. Again by Lemma 4.32 the sequence $[A', B, C']$ is a logarithmic signature of $\mathbb{Z}_2^{19}$. None of the blocks is periodic, none is a Rédei block, neither is the logarithmic signature transversal.

Starting with the logarithmic signature $\alpha_1 = [A', B, C']$ we use an inductive argument to create a series of logarithmic signatures of type $(2^9, 2^6, 2^2, \ldots, 2^2)$ in $\mathbb{Z}_2^d$ for $d = 15 + 4 \cdot k$ for $k \in \mathbb{N}$. Given the logarithmic signature $\alpha_k = [A', B, C_1'', \ldots, C_{k-1}'', C_k']$ of $\mathbb{Z}_2^d$ in this series ($|C_i| = 4$), we extend this example inductively and as follows

1. Lift the vectors in α_k to $\mathbb{Z}_p^{d+4}$ by adding four zeros to each vector. Add a block $C_{k+1} = < e_{d+1}, \ldots, e_{d+4} >$.

2. As C_{k+1} is a subgroup it is periodic. In block C_k' do the following selective shifts using elements from block C_{k+1}

$$\begin{aligned}
e_d &\rightarrow e_d + e_{d+4} \\
e_d + e_{d-3} &\rightarrow e_d + e_{d-3} + e_{d+4} \\
e_{d-1} &\rightarrow e_{d-1} + e_{d+2} \\
e_{d-1} + e_{d-3} &\rightarrow e_{d-1} + e_{d-3} + e_{d+2} \\
e_{d-2} &\rightarrow e_{d-2} + e_{d+1} \\
e_{d-2} + e_{d-3} &\rightarrow e_{d-2} + e_{d-3} + e_{d+1} \\
e_{d-2} + e_{d-1} &\rightarrow e_{d-2} + e_{d-1} + e_{d+3} \\
e_{d-3} + e_{d-2} + e_{d-1} &\rightarrow e_{d-3} + e_{d-2} + e_{d-1} + e_{d+3}
\end{aligned}$$

Denote the resulting block by C_k''. By Lemma 4.32 the sequence

$$[A', B, C_1'', \ldots, C_{k-1}'', C_k'', C_{k+1}]$$

is a logarithmic signature of $\mathbb{Z}_2^{d+4}$. The vector e_{d-3} is a period of $A' + B + C_1'' + \ldots + C_k''$.

3. We do a selective shift in C_{k+1} by e_{d-3}:

$$\begin{aligned}
e_{d+1} + e_{d+2} + e_{d+3} + e_{19} &\rightarrow e_{16} + e_{17} + e_{18} + e_{19} + e_1 \\
e_{17} + e_{18} + e_{19} &\rightarrow e_{17} + e_{18} + e_{19} + e_1
\end{aligned}$$

Denote the resulting block by C_{k+1}'. By Lemma 4.32 the sequence

$$[A', B, C_1'', \ldots, C_k'', C_{k+1}']$$

is a logarithmic signature of $\mathbb{Z}_2^{d+4}$.

In the appendix on page 132 we give an example of this logarithmic signature with 5 blocks. This example shows the inductive structure of the logarithmic signature. Neither is one of the blocks periodic, nor a Rédei block, nor is the logarithmic signature transversal. So this logarithmic signature is not factorable with known factoring algorithms.

This shows the limits of the factoring algorithm form the previous section. As each logarithmic signature in this series contains a lot of structure, we do not claim that factoring with respect to these logarithmic signatures is hard.

Outlook In this chapter we studied logarithmic signatures in elementary abelian p-groups. We showed that they always contain a basis with ideal distribution. Furthermore we used this result to give a complete characterization of logarithmic signatures in elementary abelian 2-groups with blocks of size less than 4. Another characterization of these logarithmic signatures using graph theory was given. In a further section we developed a factoring algorithm for logarithmic signatures with a Rédei block in each iteration of the algorithm. We also extended this algorithm to the case in which in a constant number of iterations only the sum of two blocks is a Rédei block. Furthermore we gave an example of logarithmic signatures which are not factorable by known algorithms.

It is still an open question whether candidates for wild logarithmic signatures exist. These can only exist if $P \neq NP$. One way to search for candidates for wild logarithmic signatures is to find a computational problem that is known to be NP-hard and also polynomial time equivalent to the factoring problem for logarithmic signatures. A start for further research could be the extended matrix representation presented in this chapter.

Chapter 5

Cryptanalysis of the MOR cryptosystem

Paeng et al. introduced the MOR cryptosystem in 2001, see [P+01b, P+01a]. This ElGamal-type public key cryptosystem is based on the intractability of the discrete logarithm problem in the inner automorphism group $Inn(G)$ of a non-abelian group G.

Paeng et al. propose to use the semidirect product $SL(2,p) \times_\theta \mathbb{Z}_p$ as a group for the MOR cryptosystem. But MOR on this group is insecure, as shown by Tobias in [Tob03, Tob04b, Tob04a]. The presented attacks enable an adversary to derive significant parts of the plaintext or even the secret encryption exponent. These attacks use special properties of $SL(2,p)$ and do not work for semidirect products of $GL(n,q)$ by an arbitrary abelian group. However, in [Kor05, Kor08] we presented ciphertext-only attacks for certain special cases of such semidirect products. We showed that the security of MOR on such groups solely depends on the difficulty of the discrete logarithm problem in small extension fields of $\mathbb{F}_q$. However the analysis lacks generality and we assumed to know the automorphism θ_h, which is only implicitly given with the public keys.

In this chapter we consider the general case. First we show how it is possible to compute the components of the automorphism θ_h. Thus justifying the attacks from [Kor05, Kor08] and building the basis for further attacks. We assume that computing discrete logarithms in small extensions of $\mathbb{F}_q$ is efficient. In this scenario we demonstrate how to compute a key that allows an adversary to obtain (an $\mathbb{F}_q$-multiple of) the plaintext. The notation in this chapter is introduced in Section 2.3.

5.1 Related problems and the setting of MOR

When working with MOR on the group $GL(n,q) \times_\theta \mathcal{H}$ we assume that computations in $GL(n,q) \times_\theta \mathcal{H}$ are efficient. A matrix in $GL(n,q)$ is given by n elements of $\mathbb{F}_q$. Therefore we consider any calculation in time polynomial in n and $\log q$ as efficient. The efficiency of the following three problems is closely connected to the efficiency of our attacks. The first two problems are efficiently solvable in our setting.

Inner automorphism problem The inner automorphism problem (IAP) is defined as follows: Given a group G and an inner automorphism $I_g \in Inn(G)$ via $I_g = \{I_g(\gamma_i) : 1 \leq i \leq l\}$. Find $h \in G$ such that $I_h = I_g$.

Note that a solution of the IAP is not necessarily unique. In fact we have:

$$I_h = I_g \Leftrightarrow h \in g \cdot Z(G).$$

[Kor08] describes how the IAP in $GL(n,q)$ is efficiently solvable by solving n^2 linear equations over $\mathbb{F}_q$ even if the inner automorphism is given only implicitly as a black box function.

Special DLP We define the special discrete logarithm problem (sDLP) in $\mathbb{F}_q$ as follows: Given an element $\nu \in \mathbb{F}_q$ and the generator ξ of $\mathbb{F}_q^*$ such that $\nu = \xi^{p^r}$ or $\nu = \xi^{-p^r}$, where $1 \leq r \leq m$. Find r and the sign of the exponent.

There are $2m \in O(\log q)$ possible values for ν. An algorithm that performs a search over all $2m$ values, efficiently finds the right exponent (i.e. in time $O(\log q)$):

```
x:= xi;

for i= 0..m-1 do

   if x = nu then
      return (j,+);
      else if x = nu^(-1) then
           return (j,-);
           fi;
   x := x^p;
   fi;
od;
```

Note that there is a single solution to this problem: Suppose that there exist $0 < t \leq s < m$ such that $\xi^{p^t} = \xi^{-p^s}$. Then

$$\xi^{p^t+p^s} = 1$$

and $q-1 = p^m - 1 \mid p^t + p^s$. Since $p^t + p^s < 2(q-1)$, it follows $p^m - 1 = p^t + p^s$. Thus $p | p^t + p^s = p^m - 1$ which is a contradiction. □

The discrete logarithm problem with inner automorphisms Building on the argumentation in [Kor05, Kor08] we show that the discrete logarithm problem in $Inn(GL(n,q))$ is efficiently reducible to the DLP in $\mathbb{F}_{q^i}$ for certain $i \in \{1, \ldots, n\}$. The reduction has two steps. First the problem is reduced to a centralized DLP by solving two instances of an IAP in $GL(n,q)$. Here the centralized discrete logarithm problem (cDLP) is defined as follows: Given a group G, $g \in G$ and $h \in < g > \cdot Z(G)$. Find $a \in \mathbb{N}$ and $z \in Z(G)$ such that $g^a = zh$. In a second step we reduce the cDLP in $GL(n,q)$ to the DLP in small extension fields of $\mathbb{F}_q$. For a detailed proof we refer to [Kor05, Kor08].

It follows that if we assume that discrete logarithms in small extension fields of $\mathbb{F}_q$ are efficiently computable, the third problem above is also efficiently solvable.

Setting of MOR Now we take a closer look at the structure of a ciphertext for MOR on $GL(n,q) \times_\theta \mathcal{H}$. For $C \in GL(n,q)$ and $h \in \mathcal{H}$ let

$$I_{(C,h)} \in Inn(GL(n,q) \times_\theta \mathcal{H}).$$

If a denotes the secret key of Alice, then the pair of inner automorphisms

$$(I_{(C,h)},\ I_{(C,h)^a})$$

is the corresponding public key. Recall that all the inner automorphisms are given as sets of values on the generators of $GL(n,q) \times_\theta H$. An intercepted ciphertext of Bob is of the form

$$(I_{(C,h)^b},\ I_{(C,h)^{ab}}(M,s))$$

where b is Bobs secret key. We use the following notation for a ciphertext of a plaintext message $(M,s) \in GL(n,q) \times_\theta \mathcal{H}$ by

$$(M',s) = I_{(C,h)^{ab}}(M,s).$$

Since $\mathcal{H}$ is abelian, the second component s of the plaintext is sent in clear and the actual ciphertext is the matrix M'. Due to Remark 2.13 we see that this matrix is of the form

$$M' = p_{ab} \cdot \theta_{h^{ab}}(M) \cdot \theta_s(p_{ab}^{-1}) \in GL(n, q),$$

where for $x \in \mathbb{N}$:

$$p_x := \prod_{i=0}^{x-1} \theta_{h^i}(C).$$

With the public key of Alice we can extract useful information that is similar to a ciphertext matrix.

Definition 5.1. Let $A \in GL(n, q)$. Given the public key of Alice, and without knowing the secret key a, we are always able to compute the following, where $x \in \{1, a\}$:

Since $I_{(C,h)^x}(A,\ 1) = (I_{p_x}(\theta_{h^x}(A)),\ 1)$, we define

$$\Psi(x, A) := I_{p_x}(\theta_{h^x}(A)).$$

Another definition will be used throughout the following analysis:

Definition 5.2. For $X, Y \in GL(n, q)$ we define

$$X \cong Y \Leftrightarrow X \in Y \cdot Z(GL(n, q)),$$

and $\overline{X}$ denotes any matrix such that $\overline{X} \cong X$.

5.2 Extracting the partial secret key

The public keys are inner automorphisms which are not given by the defining elements (C, h), $(C, h)^a$, and $(C, h)^b$, but as sets of values on the generators of $GL(n, q) \times_\theta \mathcal{H}$. Thus, neither h nor the automorphism θ_h is explicitly given with the public keys.

From Section 2.3.2 we know that θ_h is of the form

$$\theta_h = I_H \circ \varphi_i \circ \mathfrak{f}^r \circ ct^*,$$

where

- $H \in GL(n, q)$ defines an inner automorphism
- $1 \leq i \leq q-1$ $(gcd(in+1, q-1) = 1)$ defines a central automorphism
- $1 \leq r \leq m$ defines a field automorphism
- $* \in \{0, 1\}$ indicates the use of the central automorphism.

Depending on the parameters of θ_h there are different methods to break the system. For example in [Kor05], it is shown how to attack MOR if θ_h is a basic automorphism. There is no method known, that computes the components of θ_h. Therefore, we would like to extract the parameters H, i, r, and $*$ efficiently. In the following we show how to compute this information using Ψ and the partial public key $I_{(C,h)}$.

There are two steps. The first and major step finds all parameters apart from the matrix H. In the second step we extract a matrix $\overline{CH}$ which is actually sufficient for attacking the system.

5.2.1 Computing the parameters i, r and ct

Recall the structure of the matrix G_1 from page 27 and compute the value

$$\Psi(1, G_1) = CHct^*(\xi^{ip^r} G_1^{p^r}) H^{-1} C^{-1}$$

$$= CH \cdot \begin{pmatrix} \alpha & & & 0 \\ & \beta & & \\ & & \ddots & \\ 0 & & & \beta \end{pmatrix} \cdot (CH)^{-1},$$

where

$$\alpha = \xi^{\pm(i+1)p^r} \text{ and } \beta = \xi^{\pm ip^r}$$

$(\pm = +$ if $* = 0$ and $\pm = -$ if $* = 1)$. Note that α and β contain the three parameters that indicate which central, field and contragredient automorphism are the components of θ_h. The conjugation with the matrix CH hides α and β. The following idea is used to extract both values: We consider the coefficients of the characteristic polynomial χ of this matrix which are efficiently computable. As these do not change under conjugation, we have

$$\chi = \sum_{i=0}^{n} s_{n-i} x^i,$$

where

$$\begin{aligned} s_1 &= \alpha + (n-1)\beta \\ s_2 &= (n-1)\alpha\beta + \binom{n-1}{2}\beta^2, \end{aligned}$$

and (if $n \geq 3$)

$$s_3 = \binom{n-1}{2}\alpha\beta^2 + \binom{n-1}{3}\beta^3.$$

At first we consider the case that we have these three equations and all of the binomial coefficients are well defined, i.e. $n \geq 3$ and $p \geq 5$. We use the first equation to eliminate α from the second and third equation:

$$\begin{aligned} s_2 &= (n-1)\alpha\beta + \binom{n-1}{2}\beta^2 \\ &= (n-1)(\alpha + (n-1)\beta)\beta - \binom{n}{2}\beta^2 \\ &= (n-1)s_1\beta - \binom{n}{2}\beta^2 \end{aligned}$$

$$\begin{aligned} s_3 &= \binom{n-1}{2}\alpha\beta^2 + \binom{n-1}{3}\beta^3 \\ &= \binom{n-1}{2}(\alpha + (n-1)\beta)\beta^2 - 2\binom{n}{3}\beta^3 \\ &= \binom{n-1}{2}s_1\beta^2 - 2\binom{n}{3}\beta^3 \end{aligned}$$

Now we rewrite the second equation to

$$\begin{aligned} \beta^2 &= \frac{2}{n}s_1\beta - \frac{2}{n(n-1)}s_2 \\ &=: x\beta - y \end{aligned}$$

and use it to linearize the third equation:

$$\begin{aligned}
2\binom{n}{3}\beta^3 &= \binom{n-1}{2}s_1\beta^2 - s_3 \\
2\binom{n}{3}\beta(x\beta - y) &= \binom{n-1}{2}s_1(x\beta - y) - s_3 \\
2\binom{n}{3}x\beta^2 - 2\binom{n}{3}y\beta &= \binom{n-1}{2}xs_1\beta - y\binom{n-1}{2}s_1 - s_3 \\
2\binom{n}{3}x^2\beta - 2\binom{n}{3}xy - 2\binom{n}{3}y\beta &= \binom{n-1}{2}xs_1\beta - y\binom{n-1}{2}s_1 - s_3 \\
\left(2\binom{n}{3}x^2 - \binom{n-1}{2}xs_1 - 2\binom{n}{3}y\right)\beta &= 2\binom{n}{3}xy - y\binom{n-1}{2}s_1 - s_3.
\end{aligned}$$

The expression in the parentheses on the left side of the equation evaluates to

$$\begin{aligned}
& \frac{2n(n-1)(n-2)4s_1^2}{3\cdot 2n^2} - \frac{(n-1)(n-2)2s_1^2}{2n} - \frac{2n(n-1)(n-2)2s_2}{3\cdot 2n(n-1)} \\
= \; & \frac{4(n-1)(n-2)s_1^2 - 3(n-1)(n-2)s_1^2 - 2n(n-2)s_2}{3n} \\
= \; & \frac{(n-2)((n-1)s_1^2 - 2ns_2)}{3n}.
\end{aligned}$$

The right side of the above equation yields

$$\begin{aligned}
& \frac{2n(n-1)(n-2)2s_1 2s_2}{3\cdot 2nn(n-1)} - \frac{(n-1)(n-2)s_1 2s_2}{2n(n-1)} - s_3 \\
= \; & \frac{4(n-2)s_1s_2 - 3(n-2)s_1s_2 - 3ns_3}{3n} \\
= \; & \frac{(n-2)s_1s_2 - 3ns_3}{3n}.
\end{aligned}$$

It follows that

$$\frac{(n-2)((n-1)s_1^2 - 2ns_2)}{3n}\beta = \frac{(n-2)s_1s_2 - 3ns_3}{3n}.$$

We derive

$$\beta = \frac{(n-2)s_1s_2 - 3ns_3}{(n-2)((n-1)s_1^2 - 2ns_2)},$$

and

$$\alpha = s_1 - (n-1)b = s_1 - \frac{(n-1)((n-2)s_1s_2 - 3ns_3)}{(n-2)((n-1)s_1^2 - 2ns_2)},$$

which are efficiently computable.

Now we look at s_1 and s_2 in the cases in which the above operations and transformations are critical. These six cases consider the relation between p and n:

	p, n	s_1	s_2	see
1)	$n = 2$	$\alpha + \beta$	$\alpha\beta$	1.
2)	$p = 2$	$\alpha + \beta$ if $n \equiv 0$ (2)	$\alpha\beta + \beta^2$ if $n \equiv 0$ (4)	2.
			$\alpha\beta$ if $n \equiv 2$ (4)	6.
		α if $n \equiv 1$ (2)	β^2 if $n \equiv 1$ (4)	3.
			0 if $n \equiv 3$ (4)	3.
3)	$p = 3$	$\alpha - \beta$ if $n \equiv 0$ (3)	$-\alpha\beta + \beta^2$	2.
		α if $n \equiv 1$ (3)	0	4.
		$\alpha + \beta$ if $n \equiv 2$ (3)	ab	6.
4)	$p \mid n$	$\alpha - \beta$	$-\alpha\beta + \beta^2$	2.
5)	$p \mid n - 1$	α	0	5.
6)	$p \mid n - 2$	$\alpha + \beta$	$\alpha\beta$	6.

Note that in case 2 to 6 we assume that $n > 2$ and in case 4 to 6 we assume that $p \geq 5$. We combine the cases for the analysis as seen in the last column. Additionally, we look at the cases in which the values found for s_1 and s_2 are not sufficient for computing α and/or β. We use a slightly modified approach: Instead of using $\Psi(1, G_1)$, we compute the value

$$\Psi(1, \begin{pmatrix} \xi & & & \\ & \xi & 0 & \\ & & 1 & \\ & 0 & & \ddots \\ & & & & 1 \end{pmatrix} = CH \cdot \begin{pmatrix} \alpha\beta & & & \\ & \alpha\beta & 0 & \\ & & \beta^2 & \\ & 0 & & \ddots \\ & & & & \beta^2 \end{pmatrix} \cdot (CH)^{-1}$$

with $\alpha = \xi^{\pm(i+1)p^r}$ and $\beta = \xi^{\pm ip^r}$ ($\pm = +$ if $* = 0$ and $\pm = -$ if $* = 1$) as above.

The coefficients of the characteristic polynomial $\tilde{\chi} = \sum_{i=0}^{n} \tilde{s_{n-i}} x^i$ of this matrix are

$$\begin{aligned} \tilde{s_1} &= 2\alpha\beta + (n-2)\beta^2 \\ \tilde{s_2} &= \alpha^2\beta^2 + 2(n-2)\alpha\beta^3 + \binom{n-2}{2}\beta^4. \end{aligned}$$

Whenever necessary we will use these equations instead of those for s_1 and s_2. Now we look at the eleven cases in the table and summarize them as indicated in the last column:

1. If $n = 2$, then $ct \in Inn \cdot Aut_c \cdot < f >$ and w.l.o.g. we can assume that $* = 0$. Now we want to derive the solutions α and β of the quadratic equation $x^2 - s_1 x + s_2 = (x - \alpha)(x - \beta) = 0$. Since we are able to solve the DLP, we compute the solutions $x_1,\ x_2$ of this equation efficiently. Finally, we have to decide which of the solutions equals α and which equals β. Note that

$$x_1 x_2^{-1} = \begin{cases} \xi^{p^r} & \text{if} \quad x_1 = \alpha,\ x_2 = \beta, \\ \xi^{-p^r} & \text{if} \quad x_1 = \beta,\ x_2 = \alpha. \end{cases}$$

   As we are able to solve the special DLP, we can determine α and β.

2. We have $s_1 = \alpha \pm \beta$ and insert this into

$$s_2 = \pm\alpha\beta + \beta^2 = -\beta \cdot (\alpha \pm \beta) = -\beta s_1.$$

   And thus $\beta = -\frac{s_2}{s_1}$ (Note that $s_1 \neq 0$). It follows that $\alpha = s_1 \mp \beta = s_1 \pm \frac{s_2}{s_1}$.

3. In this case $s_1 = \alpha$ and $s_2 = \beta^2$ (or $\tilde{s_2} = \beta^2$). There exists $1 \leq k \leq \frac{q-1}{2}$ such that $s_2 = \xi^2 k$. Solving a DLP yields $2k$. And thus $\beta = \pm\xi^k$. As α is known, it is easy to find out, which of the two solutions equals β.

4. As s_2 is 0, we look at $\tilde{s_1}$ and $\tilde{s_2}$. These are

$$\tilde{s_1} = -\alpha\beta - \beta^2 = -\beta(\alpha + \beta)$$

   and

$$\tilde{s_2} = (\alpha\beta)^2 + \alpha\beta^3 = \alpha\beta^2(\alpha + \beta) = -\alpha\beta\tilde{s_1}.$$

   Since

$$\alpha = s_1,$$

   it follows that

$$\beta = -\frac{\tilde{s_2}}{\alpha\tilde{s_1}} = -\frac{\tilde{s_2}}{s_1\tilde{s_1}}.$$

5. Again s_2 is 0, and we look at $\tilde{s_1}$ and $\tilde{s_2}$. These are

$$\tilde{s_1} = 2\alpha\beta - \beta^2$$

and

$$\tilde{s_2} = (\alpha\beta)^2 - 2\alpha\beta^3.$$

We use the first equation to linearize the second:

$$\begin{aligned}
\tilde{s_2} &= \alpha^2(2\alpha\beta - \tilde{s_1}) - 2\alpha\beta(2\alpha\beta - \tilde{s_1}) \\
&= 2\alpha^3\beta - \alpha^2\tilde{s_1} - 4\alpha^2\beta^2 - 2\alpha\beta\tilde{s_1} \\
&= 2\alpha^3\beta - \alpha^2\tilde{s_1} - 4\alpha^2(2\alpha\beta - \tilde{s_1}) + 2\alpha\beta\tilde{s_1} \\
&= 2\alpha^3\beta - \alpha^2\tilde{s_1} - 8\alpha^3\beta + 4\alpha^2\tilde{s_1} + 2\alpha\beta\tilde{s_1} \\
&= -6\alpha^3\beta + 3\alpha^2\tilde{s_1} + 2\alpha\beta\tilde{s_1}
\end{aligned}$$

It follows that

$$\beta = \frac{\tilde{s_2} - 3\alpha^2\tilde{s_1}}{2\alpha\tilde{s_1} - 6\alpha^3}$$

and

$$\alpha = s_1.$$

6. In these cases we want to derive the solutions α and β of the quadratic equation $x^2 - s_1x + s_2 = (x-\alpha)(x-\beta) = 0$. Since we are able to solve the DLP, we compute the solutions $x_1,\ x_2$ of this equation efficiently. Finally, we have to decide which of these equals α and which equals β. We use the determinant $s_n = \alpha\beta^{n-1}$ and compute $x_1x_2^{n-1}$. By comparison we decide if $x_1 = \alpha$ and $x_2 = \beta$ or vice versa.

Now we computed $\alpha = \xi^{\pm(i+1)p^r}$ and $\beta = \xi^{\pm ip^r}$ and from that we derive

$$\alpha\beta^{-1} = \xi^{\pm p^r}.$$

The sDLP algorithm extracts the parameters r and $*$ efficiently. Finally, we compute $\pm ip^r$ as the discrete logarithm of β. Since we know $\pm p^r$, we also know i.

Thus, we obtain three of the four parameters of the automorphism θ_h.

5.2.2 Computing the parameter CH

Currently there exists no method to extract the matrix H. Here we show how it is possible to derive a matrix $\overline{CH}$, that also will suffice for our analysis. We use the notation $\theta_h = I_H \circ g$, where g is the automorphism which we identified with the methods of the previous section. For any matrix $A \in GL(n,q)$ we have

$$\Psi(1, A) = I_{CH} \circ g(A).$$

As we know g, we can also compute all values of the inner automorphism I_{CH} through

$$\Psi(1, g^{-1}(A)) = I_{CH}(A).$$

With the IAP algorithm we compute $\overline{CH}$ efficiently.

5.3 Two attacks on the secret key

In this section we present two ciphertext-only attacks on MOR with the group $GL(n,q) \times_\theta \mathcal{H}$ in the case when the secret key automorphism θ_h is either a product of an inner and a central automorphism or none of its components is an inner automorphism. In each case we obtain a key $\mathring{a}$ that allows an adversary to compute an $\mathbb{F}_q$-multiple of the plaintext.

5.3.1 Attack I: without inner automorphism

We consider the case that none of the components of θ_h is an inner automorphism, i.e. we have

$$\theta_h = \varphi_i \circ \mathfrak{f}^r \circ ct^*,$$

where we assume $1 \leq i \leq q-1$, $1 \leq r \leq m$, and $* \in \{0,1\}$ to be known with the methods of Section 5.2. Note that we also know

$$v := ord(\mathfrak{f}^r \circ ct^*),$$

with

$$v \in O(\log q).$$

We assume w.l.o.g. that $ord(\mathfrak{f}^r) = m$ (otherwise use $\frac{m}{gcd(r,m)}$).

Definition 5.3. For $x \in \mathbb{N}$ we denote

$$\bar{x} = x \mod v$$

and

$$\hat{x} = \frac{x - \bar{x}}{v},$$

i.e. $x = \bar{x} + \hat{x}v$, where $\bar{x} \leq v$.

Now we are able to compute

$$X := \prod_{k=0}^{v-1} \mathfrak{f}^{rk} \circ ct^k(C).$$

Furthermore we define

$$Y(x) := \prod_{k=0}^{x-1} \mathfrak{f}^{rk} \circ ct^k(C).$$

We consider a special representation of the value p_x: On page 28 we showed that central automorphisms commute with all other automorphism of $GL(n, q)$ and also that

$$\mathfrak{f} \circ ct = ct \circ \mathfrak{f}.$$

Note that for any $x \in \mathbb{N}$

$$x - 1 = \hat{x} + \hat{x} \cdot v - 1.$$

With this we have

$$\begin{aligned}
p_x &= \prod_{k=0}^{x-1} \theta_{h^k}(C) \\
&= \prod_{k=0}^{x-1} (\varphi_i \circ \mathfrak{f}^r \circ ct^*)^k(C) \\
&= \prod_{k=0}^{x-1} \varphi_i^k \circ \mathfrak{f}^{rk} \circ ct^k(C) \\
&\cong \prod_{k=0}^{x-1} \mathfrak{f}^{rk} \circ ct^k(C) \\
&= \left(\prod_{k=0}^{\bar{x}-1} \mathfrak{f}^{kr} \circ ct^k(C) \right) \cdot \left(\prod_{k=0}^{v-1} \mathfrak{f}^{rk} \circ ct^k(C) \right)^{\widehat{x}} \\
&= Y(\bar{x}) \cdot X^{\widehat{x}}
\end{aligned}$$

In the following two steps we compute two values $\hat{a}$ and $\tilde{a}$ that will define $\mathring{a}$.

Step 1 - Compute $\bar{a}$

Evaluate

$$\begin{aligned}
\Psi(a, G_1) &= p_a \cdot \theta_h^a(G_1) \cdot p_a^{-1} \\
&= p_a \cdot \varphi_i^a \circ \mathfrak{f}^{ar} \circ ct^{*a}(G_1) \cdot p_a^{-1} \\
&= p_a \cdot \varphi_i^a \circ \mathfrak{f}^{ar}(G_1)^{(-1)^{*a}} \cdot p_a^{-1} \\
&= p_a \cdot \varphi_i^a(G_1)^{(-1^* p^r)^a} \cdot p_a^{-1} \\
&= p_a \cdot (\xi^{s_i(a)} G_1)^{(-1^* p^r)^a} \cdot p_a^{-1} \\
&= p_a \cdot \begin{pmatrix} \gamma & & & \\ & \mu & & \\ & & \ddots & \\ & & & \mu \end{pmatrix} \cdot p_a^{-1},
\end{aligned}$$

where

$$\gamma = \xi^{(s_i(a)+1)(-1^* p^r)^a}$$

and

$$\mu = \xi^{s_i(a)(-1^* p^r)^a}.$$

In the same way as in Section 5.2 we are able to extract both γ and μ and can compute

$$\gamma \mu^{-1} = \xi^{(-1^* p^r)^a}.$$

Using the sDLP algorithm we extract $\bar{a}$ as follows:

If $* = 1$, the algorithm outputs $ra \bmod m$ - and also the information whether a is even or odd. Since we know r, and r is invertible mod m ($\gcd(r, m) = 1$), we can compute $a \bmod m$ and also

$$a \bmod 2m = a \bmod v = \bar{a}.$$

If $* = 0$, the algorithm outputs

$$ra \bmod m = ra \bmod v.$$

Since we can compute $r^{-1} \bmod m$, we derive $a \bmod v = \bar{a}$.

Step 2 - Compute $\hat{a}$

Note that for every $W \in GL(n, q)$

$$\begin{aligned}
\Psi(a, W) &= I_{p_a}(\theta_{h^a}(W)) \\
&= I_{X^{\hat{a}} \cdot Y(\bar{a})}(\varphi_i^a \circ (\mathfrak{f}^r \circ ct^*)^{\bar{a}}(W)).
\end{aligned}$$

Since we know $\bar{a}$ and thus $Y(\bar{a})$ and $(\mathfrak{f}^r \circ ct^*)^{\bar{a}}$, we can compute

$$\gamma(W) := I_{Y(\bar{a})^{-1}}\left(\Psi(a, (\mathfrak{f}^r \circ ct^*)^{-\bar{a}}(W))\right) = I_{X^{\hat{a}}} \circ \varphi_i^a(W).$$

Using the invariance of the trace under conjugation, we can compute the inner automorphism $I_{X^{\hat{a}}}$ as follows: For each $W \in GL(n,q)$ there exists $\alpha_W \in \mathbb{F}_q$ such that

$$\gamma(W) = \alpha_W \cdot I_{X^{\hat{a}}}(W),$$

and

$$\begin{aligned} trace(\gamma(W)) &= trace(\alpha_W \cdot I_{X^{\hat{a}}}(W)) \\ &= \alpha_W \cdot trace(I_{X^{\hat{a}}}(W)) = \alpha_W \cdot trace(W). \end{aligned}$$

We know both W and $\gamma(W)$ and can therefore compute α_W and also

$$I_{X^{\hat{a}}}(W) = \alpha_W^{-1} \cdot \gamma(W).$$

Now we know $I_{X^{\hat{a}}}$. We also know I_X. By solving a DLP in $Inn(GL(n,q))$, we obtain $\tilde{a} \in \mathbb{N}$ such that $X^{\tilde{a}} \cong X^{\hat{a}}$.

With

$$\mathring{a} := \bar{a} + \tilde{a} \cdot v$$

we computed a value that may be equal to a. It is at least equivalent to a in the following sense.

Proposition 5.4. *Let $(I_{(C,h)^b},\ (M', s) = I_{(C,h)^{ab}}(M, s))$ be a ciphertext message that was send to Alice, where $M' = p_{ab} \cdot \theta_{h^{ab}}(M) \cdot \theta_s(p_{ab}^{-1})$.*

Then we have

$$p_{\mathring{a}b} \cong p_{ab}$$

and for all $W \in GL(n,q)$

$$\theta_{h^{ab}}(W) \cong \theta_{h^{\mathring{a}b}}(W) \text{ or } \theta_{h^{ab}}(W) \cong \theta_{h^{\mathring{a}b}}(ct(W)).$$

Proof. We have $\bar{\mathring{a}} = \overline{\bar{a} + \tilde{a} \cdot v} = \bar{\bar{a}} = \bar{a}$. It follows that $\overline{\mathring{a}b} = \overline{\bar{\mathring{a}}\bar{b}} = \overline{\bar{a}\bar{b}} = \overline{ab}$. And thus

$$Y(\overline{\mathring{a}b}) = Y(\overline{ab}).$$

Furthermore we have

$$\begin{aligned}\mathring{a}b &= (\bar{a}+\tilde{a}\cdot v)(\bar{b}+\hat{b}\cdot v)\\ &\quad \overline{ab}+(\underbrace{\widehat{\bar{a}\bar{b}}+\bar{a}\hat{b}+\tilde{a}\bar{b}+\tilde{a}\hat{b}v}_{\widehat{a\bar{b}}})\cdot v\end{aligned}$$

and

$$\begin{aligned}ab &= (\bar{a}+\hat{a}\cdot v)(\bar{b}+\tilde{b}\cdot v)\\ &\quad \overline{ab}+(\underbrace{\widehat{\bar{a}\bar{b}}+\bar{a}\hat{b}+\hat{a}\bar{b}+\hat{a}\hat{b}v}_{\widehat{\bar{a}\bar{b}}})\cdot v\end{aligned}$$

Since $X^{\tilde{a}} \cong X^{\hat{a}}$ it follows that

$$\begin{aligned}X^{\mathring{a}b} &= X^{\overline{ab}+(\widehat{\bar{a}\bar{b}}+\bar{a}\hat{b}+\tilde{a}\bar{b}+\tilde{a}\hat{b}v)v}\\ &\cong X^{\overline{ab}+(\widehat{\bar{a}\bar{b}}+\bar{a}\hat{b}+\hat{a}\bar{b}+\hat{a}\hat{b}v)v}\\ &= X^{ab}.\end{aligned}$$

We combine these results to

$$p_{\mathring{a}b} = Y(\overline{\mathring{a}b})\cdot X^{\mathring{a}b} \cong Y(\overline{ab})\cdot X^{ab} = p_{ab}$$

For the automorphism θ_h we look at the case $* = 1$ and $\bar{ab} \neq \overline{\mathring{a}b}$ separately. We have for all $W \in GL(n,q)$

$$\begin{aligned}\theta_{h^{ab}}(W) &= \varphi_i^{ab}\circ \mathfrak{f}^{rab}\circ ct^{ab}(W)\\ &\cong \mathfrak{f}^{r\overline{ab}}\circ ct^{\bar{ab}}(W)\\ &= \begin{cases}\mathfrak{f}^{r\overline{\mathring{a}b}}\circ ct^{\overline{\mathring{a}b}}(W) & \text{if } * = 0 \text{ or } \overline{\mathring{a}b} = \overline{ab} \bmod 2\\ \mathfrak{f}^{r\overline{\mathring{a}b}}\circ ct^{\overline{\mathring{a}b}}\circ ct(W) & \text{else}\end{cases}\\ &\cong \begin{cases}\theta_{h^{\mathring{a}b}}(W) & \text{if } * = 0 \text{ or } \overline{\mathring{a}b} = \overline{ab} \bmod 2\\ \theta_{h^{\mathring{a}b}}\circ ct(W) & \text{else}\end{cases}\end{aligned}$$

□

And it immediately follows:

Lemma 5.5. *For the key $\mathring{a}$ let*

$$(I_{(C,h)^b})^{-\mathring{a}}(M',s) = (M'',s).$$

Then either M'' or $ct(M'')$ is an $\mathbb{F}_q$-multiple of the plaintext M.

Proof. With Proposition 5.4 we have

$$\begin{aligned} M'' &= \theta_{h^{-\mathring{a}b}}(p_{\mathring{a}b}^{-1} \cdot M' \cdot \theta_s(p_{\mathring{a}b})) \\ &= \theta_{h^{-\mathring{a}b}}(p_{\mathring{a}b}^{-1} \cdot p_{ab} \cdot \theta_{h^{ab}}(M) \cdot \theta_s(p_{ab}^{-1}) \cdot \theta_s(p_{\mathring{a}b})) \\ &\cong \theta_{h^{-\mathring{a}b}}(\theta_{h^{ab}}(M)) \\ &\cong \begin{cases} M & * = 0 \text{ or } \overline{\mathring{a}b} \bmod 2 = \overline{ab} \bmod 2 \\ ct(M) & else. \end{cases} \end{aligned}$$

□

Lemma 5.5 and Proposition 5.4 show that the key $\mathring{a}$ is sufficient for computing an $\mathbb{F}_q^*$-multiple of the plaintext of any ciphertext message in most cases. In the case that $* = 1$ and also $\overline{ab} \neq \overline{\mathring{a}b} \bmod 2$, we would get the contragredient of the plaintext $ct(M)$ and could simply compute $ct(ct(M)) = M$.

5.3.2 Attack II: $\theta(\mathcal{H}) \subseteq Inn(GL(n,q)) \cdot Aut_c(GL(n,q))$

We will now analyze MOR for

$$\theta_h = I_H \circ \varphi_i,$$

where we assume a matrix $\overline{CH}$ and $1 \leq i \leq q-1$ to be known by Section 5.2.

On page 28 we showed that φ_i and I_H commute. In this case we have for any $x \in \mathbb{N}$

$$\begin{aligned} p_x &= \prod_{k=0}^{x-1} \theta_{h^k}(C) \\ &= \prod_{k=0}^{x-1} (I_H \circ \varphi_i)^k(C) \\ &= \prod_{k=0}^{x-1} I_{H^k} \circ \varphi_i^k(C) \\ &\cong \prod_{k=0}^{x-1} I_{H^k}(C) \\ &= \prod_{k=0}^{x-1} H^k C H^{-k} \\ &= C \cdot HCH^{-1} \cdot H^2CH^{-2} \cdot \ldots \cdot H^{x-1}CH^{-x+1} \\ &= (CH)^x H^{-x} \end{aligned}$$

and for $W \in GL(n,q)$

$$\begin{aligned}\theta_{h^x}(W) &= I_{H^x} \circ \varphi_i^x(W) \\ &\cong I_{H^x}(W).\end{aligned}$$

Since we know $\overline{CH}$ we also know the inner automorphism I_{CH}. For any matrix $W \in GL(n,q)$ we have

$$\begin{aligned}\Psi(a,W) &= I_{(CH)^a H^{-a}} \circ \theta_h^a(W) \\ &= I_{(CH)^a} \circ \varphi_i^a(W)\end{aligned}$$

In the same way as on page 112, we use Ψ to compute the inner automorphism $I_{(CH)^a}$.The DLP in $Inn(GL(n,q))$ is efficiently reducible to the DLP in finite fields (see page 101). Since we assume that we are able to solve the latter, we can assume to know a value $\mathring{a}$ such that $I_{(CH)^{\mathring{a}}} = I_{CH^a}$ and also

$$(CH)^{\mathring{a}} \cong (CH)^a.$$

Proposition 5.6. *Let $(I_{(C,h)^b},\ (M',s) = I_{(C,h)^{ab}}(M,s))$ be a ciphertext message that was send to Alice, where $M' = p_{ab} \cdot \theta_{h^{ab}}(M) \cdot \theta_s(p_{ab}^{-1})$.*

For the key $\mathring{a}$ let

$$(I_{(C,h)^b})^{-\mathring{a}}(M',s) = (M'',s).$$

Then M'' is an $\mathbb{F}_q$-multiple of the plaintext M.

Proof. The representation of p_x and θ_{h^x} yield

$$M' \cong p_{ab}\ \theta_{h^{ab}}(M)\ \theta_s(p_{ab}^{-1}) \cong (CH)^{ab} \cdot M \cdot \theta_s(CH^{-ab}).$$

and

$$M'' \cong (CH)^{-\mathring{a}b} \cdot M' \cdot \theta_s(\overline{CH}^{\mathring{a}b}).$$

And since $(CH)^{\mathring{a}} \cong (CH)^a$ it follows that

$$\begin{aligned}M'' &\cong (CH)^{-\mathring{a}b} \cdot M' \cdot \theta_s(CH^{\mathring{a}b}) \\ &\cong (CH)^{-\mathring{a}b}(CH)^{ab} \cdot M \cdot \theta_s(CH^{-ab})\theta_s(CH^{\mathring{a}b}) \\ &\cong M.\end{aligned}$$

□

We showed how to compute a value $\mathring{a}$ that allows an attacker, who is able to compute discrete logarithms in certain finite fields, to derive an $\mathbb{F}_q^*$-multiple of the plaintext.

The complex structure of the group $GL(n,q) \times_\theta \mathcal{H}$ opens several possibilities for an attack. And as we have seen, it does not yield more security than the discrete logarithm problem in the multiplicative group of finite fields.

Our analysis shows that there is no advantage in using MOR on $GL(n,q) \times_\theta \mathcal{H}$. Further research might consider MOR on different platform groups.

Chapter 6

Appendix

6.1 Algorithms for cyclic p-groups

Here we give the GAP-Code for the four different versions of the preprocessing and factoring algorithms for logarithmic signatures of finite cyclic p-groups.

6.1.1 Sample generation

The algorithm `LogSig` computes a pseudorandom logarithmic signature of $\mathbb{Z}_{p^n}$ with the type $r = (r_1, ..., r_s)$.

```
LogSig:= function(r,p,n)

################ variables:

local i,j,k,pn,a,c,s,A,logSig,subtype;
a:=0;           # counter
c:=1            # counter
s:=Length(r);   # number of blocks
pn:=p^n;        # order of the group
subtype:=[];    # subtype of output logarithmic signature
A:=[];          # plain logarithmic signature corresponding to r
logSig:=[];     # output

################# initialize LogSig
```

```
for i in [1..n] do
         A[i]:= [];
od;

for i in [1..Length(r)] do
         logSig[i]:= [0];
od;

for i in [1..n] do
    a:=c;
    for j in [1..p-1] do
        Add(A[i],a);
        a:= a + c mod pn;
    od;

    A[i]:=Permuted(A[i],Random(SymmetricGroup(p-1)));
    Add(A[i],0,1);
    c:=c*p; n
od;

################## pseudorandomly choose a subtype

for i in [1..Length(r)] do
    for j in [1..r[i]] do
        Add(subtype, i);
    od;
od;

subtype:= Permuted(subtype,Random(SymmetricGroup(n)));

#################
# Fuse two blocks and add random elements to all other blocks
c:=1;

for i in [1..n] do
    c:=c*p;

    logSig[subtype[i]]:= Fuse([ logSig[subtype[i]], A[i] ] );

    for k in [1..Length(r)] do
```

```
        if not k = subtype[i] then
            for j in [2..Length(logSig[k])] do

    logSig[k][j]:= (logSig[k][j] + Random([0..(p-1)])*c) mod pn;

            od;
        fi;
    od;
od;

return logSig;
end;
```

6.1.2 Version v1.0

GAP-Code for version v1.0 of the prefactoring and factoring algorithms which is optimized for $p = 2$. `PreprocessingLS-v1.0` returns the compact complete p-reduction and the subtype for a logarithmic signature of $\mathbb{Z}_{2^n}$.

```
PreprocessingLS-v1.0 := function(logSig,n)

################## variables:

local a,i,j,k,l,m, subtype, pReduction,LS;

l:=[];
subtype := [];
pReduction := [];
LS:=StructuralCopy(logSig);
a:=2^n;

#################
# initialize lists for the compact complete p-reduction

for i in [1..n] do
    Add(pReduction,[]);
od;

################# 0) Monitor n p-reductions
```

```
for i in [1..n] do
    a:=a/2;

################## 1) Find periodic block

k := PositionProperty(LS, j -> a in j);
Add(subtype, k, 1);

################## 2) Reduce periodic block k

Sort(LS[k]);
j:= 1/2 * Length(LS[k]);
LS[k] := LS[k]{[1..j]};

################## 3) Reduce other blocks
# and store difference elements in "pReduction"!

for j in [1..Length(LS)] do
    l := Difference( LS[j] mod a, LS[j] );
    Append(pReduction[n-i+1],l);
od;

LS:= LS mod a;

od;

return [subtype, pReduction];

end;
```

The algorithm `FactorLS-v1.0` returns the factorization of any element from $\mathbb{Z}_{2^n}$.

```
FactorLS-v1.0 := function(prefactoring,element)

################## initialize variables

local a,b,c,d,i,j,Fact,subtype,pReduction,Counter;
subtype:= prefactoring[1];
pReduction:=prefactoring[2];
```

```
Fact:= [];
b:=Maximum(subtype);
Counter :=element;
c:=2^n;
d:=2;

for i in [1..b] do Add(Fact,0); od;

################## round 1 - Fusion:

a := Counter mod 2;

Fact[subtype[1]] := (Fact[subtype[1]] + a)  mod c;

Counter := (Counter - a) mod c;

for i in [2..n] do

################## 1. Change difference elements

for j in [1..b] do
    if (not i = j) and Fact[j] in pReduction[i]
        then Fact[j] := (Fact[j] + d) mod c;
        Counter := (Counter - d) mod c;
    fi;
od;

d:=2*d;

################## 2. Fuse block k_{n-i}

a := Counter mod d;

Fact[subtype[i]] := (Fact[subtype[i]] + a) mod c;

Counter := (Counter - a) mod c;

od;

return Fact;
end;
```

6.1.3 Version v1.1

GAP-Code for version v1.1 of the preprocessing and factoring algorithms. `PreprocessingLS-v1.1` returns the compact complete p-reduction and the subtype

```
PreprocessingLS-v1.1 := function(LogSig)

################## variables:

local a,i,j,jj,k,l,m, subtype, pReduction,LS;

a:=p^n;
l:=[];
subtype := [];
pReduction := [];
LS:=StructuralCopy(LogSig);

#################
# initialize lists for the compact complete p-reduction

for i in [1..n] do
    pReduction[i]:=[];
    for j in [1..p-1] do
        Add(pReduction[i],[]); od;od;

################# 0) Monitor n p-reductions

for i in [1..n] do
a:=a/p;

################# 1) Find periodic block

k := PositionProperty(LS, j -> a in j);
Add(subtype, k, 1);

################# 2) Reduce periodic block k

Sort(LS[k]);
j:= 1/p * Length(LS[k]);
LS[k] := LS[k]{[1..j]};
```

```
################## 3) Reduce other blocks
# and store difference elements in sublists of "pReduction"

for j in [1..Length(LS)] do
    if not j = k then
        for m in [1..Length(LS[j])] do
        jj:= LS[j][m] - (LS[j][m] mod a);

        if jj > 0 then

Append(pReduction[n-i+1][jj/a], [LS[j][m] mod a] );

LS[j][m]:= LS[j][m] mod a;

fi;od;fi;od;

od;

return [subtype, pReduction];

end;
```

FactorLS-v1.1 returns the factorization of an element in $\mathbb{Z}_{p^n}$.

```
FaktorLS-v1.1 := function(preprocessing,element)

################## initialize variables

local a,c,d,i,j,k,s,Fact,subtype,pReduction,Counter;

subtype:= preprocessing[1];
pReduction:=preprocessing[2];
Fact:= [];
s:=Maximum(subtype);
Counter :=element;
c:=p^n;
d:=p;

for i in [1..s] do Add(Fact,0); od;
```

```
################## round 1 - Fusion:

a := Counter mod p;

Fact[subtype[1]] :=  a ;

Counter := (Counter - a) mod c;

for i in [2..n] do

################## 1. Change difference elements

for j in [1..s] do

    k:= PositionProperty(pReduction[i], m -> Fact[j] in m); #NEW

    if (not i = j) and (not k = fail) then

        Fact[j] := (Fact[j] + k*d) mod c;

        Counter := (Counter - k*d) mod c;

    fi;
od;

d:= d*p;

################## 2. Fuse block k_{n-i}

a := Counter mod d;

Fact[subtype[i]] := (Fact[subtype[i]] + a) mod c;

Counter := (Counter - a) mod c;

od;

return Fact;

end;
```

6.1.4 Version v2.0

GAP-Code for version v2.0 of the preprocessing and factoring algorithms. `PreprocessingLS-v2.0` returns the compact complete p-reduction for positions, a list of n permutations, and the subtype of a logarithmic signature in $\mathbb{Z}_{p^n}$.

```
PreprocessingLS_v2_0 := function(LogSig)
local a,i,j,jj,k,l,m, subtype, pReduction,LS,Permu;
a:=p^n;
l:=[];
subtype := [];
pReduction := [];
Permu:=[];
LS:=StructuralCopy(LogSig);

##################
# initialize lists for the compact complete p-reduction

for i in [1..n] do
    pReduction[i]:=[];
    for j in [1..Length(LS)] do
        pReduction[i][j]:=[];
        for k in [1..p-1] do
            Add(pReduction[i][j],[]);
od;od; od;

################## 0) Monitor n p-reductions

for i in [1..n] do
a:=a/p;

################## 1) Find periodic block

k := PositionProperty(LS, j -> a in j);
Add(subtype, k, 1);

################## 2) Reduce periodic block k

Add(Permu,Sortex(LS[k])); #NEW
j:= 1/p * Length(LS[k]);
```

```
LS[k] := LS[k]{[1..j]};


################## 3) Reduce other blocks
# and store positions of difference elements in "pReduction"!

for j in [1..Length(LS)] do
    if not j = k then
        for m in [1..Length(LS[j])] do
            jj:= LS[j][m] - (LS[j][m] mod a);

            if jj > 0 then
#NEW
Add(pReduction[n-i+1][j][jj/(p^(n-i))],
                                  Position(LS[j],LS[j][m]) );

LS[j][m]:= LS[j][m] mod a;

fi;od;fi;od;od;

return [subtype, pReduction, Permu];

end;
```

FactorLS-v2.0 returns the factorization of element in $\mathbb{Z}_{p^n}$.

```
FactorLS_v2_0 := function(preprocessing,zahl)

################## initialize variables

local a,c,pn,i,j,k,s,Fact,subtype,pReduction,
      Counter,Permu,FuseCount;

subtype:= preprocessing[1];
pReduction:=preprocessing[2];
Permu:=preprocessing[3];
Fact:= [];
FuseCount:=[];
s:=Maximum(subtype);
Counter :=zahl;
```

```
pn:=p^n;

for i in [1..s] do Add(Fact,1); od;
for i in [1..s] do Add(FuseCount,0); od;

################# round 1 - Fusion:

a := Counter mod p;
Fact[subtype[1]] := (a + 1)/Permu[n] mod pn;

Counter := (Counter - a) mod pn;

FuseCount[subtype[1]]:=FuseCount[subtype[1]] + 1;

c:=p;

for i in [2..n] do

################# 1. Change positions of difference elements

for j in [1..s] do

   if not j = subtype[i] then

     k:= PositionProperty(pReduction[i][j], m -> Fact[j] in m);

     if not k = fail then

         Counter := (Counter - k*c) mod pn;
      fi;
   fi;
od;

################# 2. Fuse block k_{n-i}

a := ((Counter mod (p*c))/c)  mod (p*c) + 1;

Fact[subtype[i]] := (((a-1)*p^FuseCount[subtype[i]]
                    + Fact[subtype[i]]) mod pn)/Permu[n+1-i];

FuseCount[subtype[i]]:=FuseCount[subtype[i]] + 1;
```

```
Counter := (Counter - (Counter mod (p*c))) mod pn;

c:=c*p;

od;

return Fact;

end;
```

6.1.5 Version v2.1

GAP-Code for version v2.1 of the preprocessing and factoring algorithms. `PreFactorLS-v2.1` returns the compact complete p-reduction for positions, a list of s permutations, and the subtype of a logarithmic signature in $\mathbb{Z}_{p^n}$.

```
PreFactorLS_v2_1 := function(LogSig)

################## variables:

local a,i,j,jj,k,m,s,u,v,subtype,pReduction,
      LS,PermCollect,Permut,FuseCount;

a:=p^n;
s:=Length(LogSig);
subtype := [];
pReduction := [];        #positions of difference elements
PermCollect:=[];         #collect all permutations
Permut:=[];              #one permutation per block
FuseCount:=[];           #counts the p-reduction rounds
LS:=StructuralCopy(LogSig);

#################
# initialize lists for the compact complete p-reduction


for i in [1..n] do
    pReduction[i]:=[];
    for j in [1..s] do
```

```
        pReduction[i][j]:=[];
            for k in [1..p-1] do
            Add(pReduction[i][j],[]);
od;od;od;

for j in [1..s] do
    Add(FuseCount, 0);
    Add(Permut, ());
od;

################## 0) Monitor n p-reductions

for i in [1..n] do
a:=a/p;


################## 1) Find periodic block k

k := PositionProperty(LS, j -> a in j);
Add(subtype, k);

################## 2) Reduce periodic block k

Add(PermCollect,Sortex(LS[k]));
j:= 1/p * Length(LS[k]);
LS[k] := LS[k]{[1..j]};


################## 3) Reduce other blocks
# and store positions of difference elements in "pReduction"!

for j in [1..s] do
    if not j = k then
        for m in [1..Length(LS[j])] do
            jj:= LS[j][m] - (LS[j][m] mod a);

                if jj > 0 then

Add(pReduction[n-i+1][j][jj/(p^(n-i))],
                                  Position(LS[j],LS[j][m]));
```

```
LS[j][m]:= LS[j][m] mod a;

fi;od;fi;od;

od;

subtype:=Reversed(subtype);

################# NEW 4) Combine permutations per block
# and permute all positions in "pReduction" accordingly

for i in [1..n] do

    v:=ListPerm(Permut[subtype[i]]);
    while Length(v)<p^FuseCount[subtype[i]] do
        Add(v, Length(v)+1);
    od;

    u:=StructuralCopy(v);

    for j in [1..p-1] do

        Append(v, u + j*p^FuseCount[subtype[i]]);
    od;

    FuseCount[subtype[i]]:=FuseCount[subtype[i]] + 1;

    Permut[subtype[i]]:= PermCollect[n+1-i]*PermList(v);

    for j in [1..s] do
        for k in [1..p-1] do
            Apply(pReduction[i][j][k], x -> x^Permut[j]);
        od;
    od;
od;

return [subtype, pReduction, Permut];

end;
```

FactorLS-v2.1 returns the factorization of an element in $\mathbb{Z}_{p^n}$.

```
FactorLS-v2.1 := function(preprocessing,zahl)

################## initialize variables

local a,c,pn,i,j,k,s,Fact,subtype,pReduction,
      Counter,Permu,FuseCount;

subtype:= preprocessing[1];
pReduction:=preprocessing[2];
Permu:=preprocessing[3];
Fact:= [];
FuseCount:=[];            # counter for blocksize
s:=Maximum(subtype);      # number of blocks
Counter :=zahl;
pn:=p^n;

for i in [1..s] do
    Add(Fact,1);
    Add(FuseCount,0);
od;

################## round 1 - Fusion:

a := Counter mod p;
Fact[subtype[1]] := (a + 1) mod pn; #NEW

Counter := (Counter - a) mod pn;

FuseCount[subtype[1]]:=FuseCount[subtype[1]] + 1;

c:=p;

for i in [2..n] do

################## 1. Change positions of difference elements

for j in [1..s] do

   if not j = subtype[i] then
```

```
     k:= PositionProperty(pReduction[i][j], m -> Fact[j] in m);

     if not k = fail then

         Counter := (Counter - k*c) mod pn;
     fi;
   fi;
od;

################# 2. Fuse block k_{n-i}

a := ((Counter mod (p*c))/c)  mod (p*c) +1 ;

Fact[subtype[i]] := (((a-1)*p^FuseCount[subtype[i]]
                                    + Fact[subtype[i]]) mod pn);

FuseCount[subtype[i]]:=FuseCount[subtype[i]] + 1;

Counter := (Counter - (Counter mod (p*c))) mod pn;

c:=c*p;

od;

#NEW only one permutation per block

for i in [1..s] do Fact[i]:=Fact[i]/Permu[i]; od;

return Fact;

end;
```

6.2 Construction of a non-Rédei example

Here we give the GAP-Code for constructing a series of logarithmic signatures, that are neither transversal, do not have a Rédei block, nor are these factorable via the algorithm proposed in Section 4.4.

We construct the logarithmic signature for $\mathbb{Z}_2^{15}$ from p. 246 of [Sza04], add blocks and modify blocks using selective shifts.

GAP: Construction of a series of logarithmic signatures
Input: s, the number of blocks, $s \geq 4$
Output: ls, a logarithmic signature with s blocks

```
### Start with the canonical basis
V:= GF(2)^19;
Bs:=Basis(V);
s:=   ;

### Construction of block A

A1:=[Zero(V),Bs[1],Bs[2],Bs[3], Bs[1]+Bs[2], Bs[1]+Bs[3],
     Bs[2]+Bs[3]+Bs[4], Bs[1]+Bs[2]+Bs[3]+Bs[5]];;

A2:=[Zero(V),Bs[6],Bs[7],Bs[8], Bs[6]+Bs[7], Bs[6]+Bs[8],
     Bs[7]+Bs[8]+Bs[9], Bs[6]+Bs[7]+Bs[8]+Bs[10]];;

A3:=[Zero(V),Bs[11],Bs[12],Bs[13], Bs[11]+Bs[12], Bs[11]+Bs[13],
     Bs[12]+Bs[13]+Bs[14], Bs[11]+Bs[12]+Bs[13]+Bs[15]];;

A:=Fuse( [ Fuse([A1,A2]) , A3]);;

### Selective Shifts in block A

A[3]:=A[3]+Bs[16];
A[5]:=A[5]+Bs[16];

A[4]:=A[4]+Bs[17];
A[6]:=A[6]+Bs[17];

A[9]:=A[9]+Bs[18];
A[10]:=A[10]+Bs[18];

A[11]:=A[11]+Bs[19];
A[13]:=A[13]+Bs[19];
```

```
### Construction of block B

H1:=[ Zero(V), Bs[4], Bs[5], Bs[4] + Bs[5]];;
H2:=[ Zero(V), Bs[9], Bs[10], Bs[9] + Bs[10]];;
H3:=[ Zero(V), Bs[14], Bs[15], Bs[14] + Bs[15]];;

B:= Fuse([Fuse([ H1, H2  ]), H3  ]);;

### Selective Shifts in block B

B[4]:=B[4] + Bs[6];;
B[8]:=B[8] + Bs[6];;
B[12]:=B[12] + Bs[6];;
B[13]:=B[13] + Bs[11];;
B[14]:=B[14] + Bs[12];;
B[15]:=B[15] + Bs[13];;
B[16]:=B[16] + Bs[6];;

B[20]:=B[20] + Bs[7];;
B[24]:=B[24] + Bs[7];;
B[28]:=B[28] + Bs[7];;
B[29]:=B[29] + Bs[11];;
B[30]:=B[30] + Bs[12];;
B[31]:=B[31] + Bs[13];;
B[32]:=B[32] + Bs[7];;

B[36]:=B[36] + Bs[8];;
B[40]:=B[40] + Bs[8];;
B[44]:=B[44] + Bs[8];;
B[45]:=B[45] + Bs[11];;
B[46]:=B[46] + Bs[12];;
B[47]:=B[47] + Bs[13];;
B[48]:=B[48] + Bs[8];;

B[49]:=B[49] + Bs[1];;
B[50]:=B[50] + Bs[1];;
B[51]:=B[51] + Bs[1];;
B[52]:=B[52] + Bs[1];;

B[53]:=B[53] + Bs[2];;
```

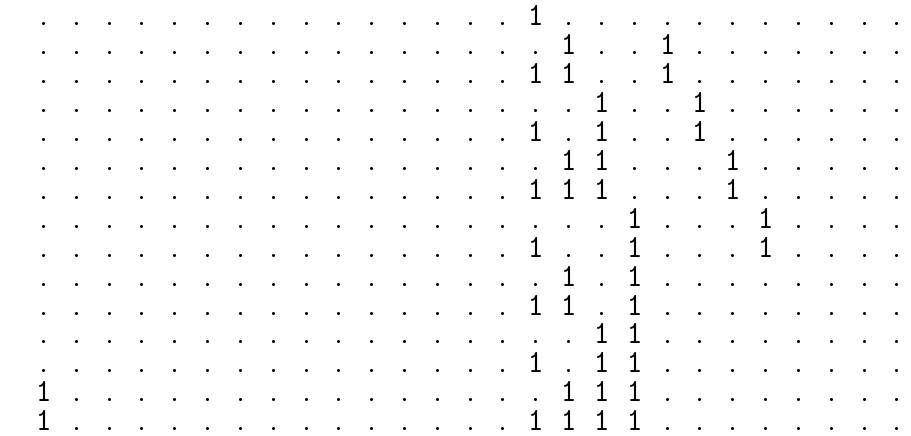
```
B[54]:=B[54] + Bs[2];;
B[55]:=B[55] + Bs[2];;
B[56]:=B[56] + Bs[2];;

B[57]:=B[57] + Bs[3];;
B[58]:=B[58] + Bs[3];;
B[59]:=B[59] + Bs[3];;
B[60]:=B[60] + Bs[3];;

B[61]:=B[61] + Bs[11];;
B[62]:=B[62] + Bs[12];;
B[63]:=B[63] + Bs[13];;

### Construction of block C

C1:=[Zero(V), Bs[16], Bs[17], Bs[16]+Bs[17]];;
C2:=[Zero(V), Bs[18], Bs[19], Bs[18]+Bs[19]];;
C:=Fuse([C1, C2]);

### Selective Shifts in block C

C[16]:=C[16]+Bs[1];
C[15]:=C[15]+Bs[1];

### Add s-3 new blocks to [A,B,C]

ls:=[A,B,C];;
d:=19;

for i in [4..s] do

Bs:=Basis(GF(2)^(d+4));

### Lift the vectors to Z_2^d+4

for x in ls do
Lift(x,d+4);
od;

### Add new block C_{k+1}
```

```
D1:=[Zero(GF(2)^(d+4)), Bs[d+1], Bs[d+2], Bs[d+1]+Bs[d+2]];;
D2:=[Zero(GF(2)^(d+4)), Bs[d+3], Bs[d+4], Bs[d+3]+Bs[d+4]];;
ls[i]:=Fuse([D1, D2]);;

### Selective shifts on block C_k (= ls[i-1])

ls[i-1][3]:=ls[i-1][3]+Bs[d+1];;
ls[i-1][4]:=ls[i-1][4]+Bs[d+1];;

ls[i-1][5]:=ls[i-1][5]+Bs[d+2];;
ls[i-1][6]:=ls[i-1][6]+Bs[d+2];;

ls[i-1][7]:=ls[i-1][7]+Bs[d+3];;
ls[i-1][8]:=ls[i-1][8]+Bs[d+3];;

ls[i-1][9]:=ls[i-1][9]+Bs[d+4];;
ls[i-1][10]:=ls[i-1][10]+Bs[d+4];;

### Selective shifts on block C_{k+1} (= ls[i])

ls[i][16]:=ls[i][16]+Bs[d-3];;
ls[i][15]:=ls[i][15]+Bs[d-3];;

d:=d+4;
od;

### auxiliary functions

Lift:= function(set, zahl)
local L,i;
L:=set;
for i in [1..Length(set)] do
L[i]:=Concatenation(L[i], Zero(GF(2)^(zahl-Length(L[i]))));
od;
return L;
end;

Fuse:=function(list)
local i,j,wert;
wert:=[];
for i in [1..Length(list[2])] do
```

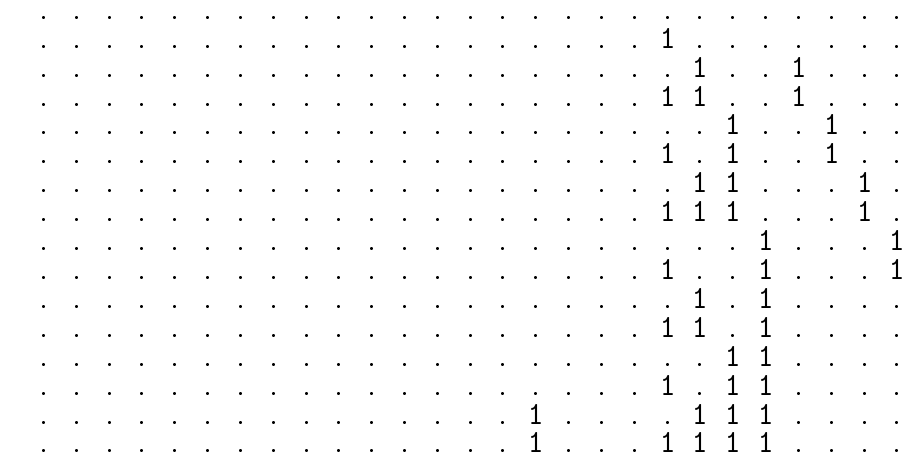
```
for j in [1..Length(list[1])] do
Add(wert, (list[1][j] + list[2][i]));
od;od;
return wert;
end;
```

As an example we give the logarithmic signature `ls` with $s = 5$ blocks that is constructed using the above code. Each line represents a vector from GF(2)^27. For better readability each zero is replaced by a dot.

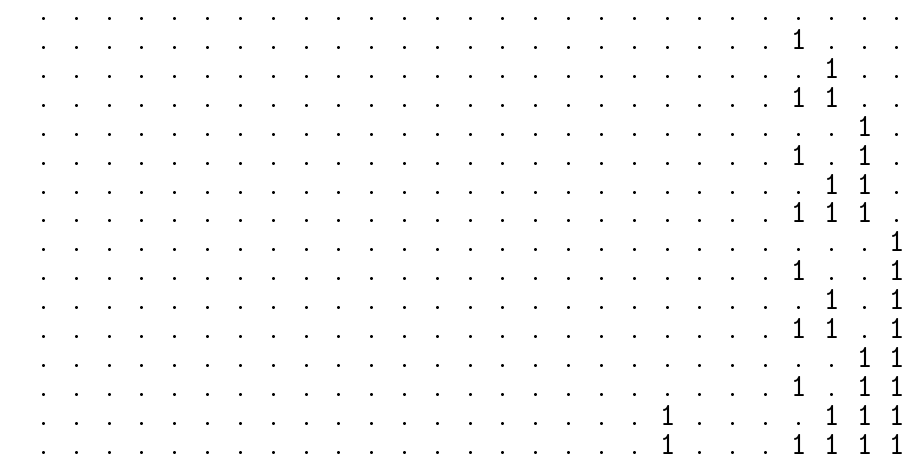
```
ls[1][1, 2, ..., 13, 2^9-2, 2^9-1, 2^9] =
 . . . . . . . . . . . . . . . . . . . . . . . . . . .
 1 . . . . . . . . . . . . . . . . . . . . . . . . . .
 . 1 . . . . . . . . . . . . . . . 1 . . . . . . . . . .
 . . 1 . . . . . . . . . . . . . . . 1 . . . . . . . . .
 1 1 . . . . . . . . . . . . . . . 1 . . . . . . . . . .
 1 . 1 . . . . . . . . . . . . . . . 1 . . . . . . . . .
 . 1 1 1 . . . . . . . . . . . . . . . . . . . . . . . .
 1 1 1 . 1 . . . . . . . . . . . . . . . . . . . . . . .
 . . . . . 1 . . . . . . . . . . . . . 1 . . . . . . . .
 1 . . . . 1 . . . . . . . . . . . . . 1 . . . . . . . .
 . 1 . . . 1 . . . . . . . . . . . . . . 1 . . . . . . .
 . . 1 . . 1 . . . . . . . . . . . . . . . . . . . . . .
 1 1 . . . 1 . . . . . . . . . . . . . . 1 . . . . . . .
 1 . 1 . . 1 1 1 . 1 1 1 1 . 1 . . . . . . . . . . . . .
 . 1 1 1 . 1 1 1 . 1 1 1 1 . 1 . . . . . . . . . . . . .
 1 1 1 . 1 1 1 1 . 1 1 1 1 . 1 . . . . . . . . . . . . .

ls[2][1, 2, ..., 13, 2^6-12, 2^6-8, 2^6-4, 2^6] =
 . . . . . . . . . . . . . . . . . . . . . . . . . . . .
 . . . 1 . . . . . . . . . . . . . . . . . . . . . . . .
 . . . . 1 . . . . . . . . . . . . . . . . . . . . . . .
 . . . 1 1 1 . . . . . . . . . . . . . . . . . . . . . .
 . . . . . . . . 1 . . . . . . . . . . . . . . . . . . .
 . . . 1 . . . . 1 . . . . . . . . . . . . . . . . . . .
 . . . . 1 . . . 1 . . . . . . . . . . . . . . . . . . .
 . . . 1 1 1 . . 1 . . . . . . . . . . . . . . . . . . .
 . . . . . . . . 1 . . . . . . . . . . . . . . . . . . .
 . . . 1 . . . . 1 . . . . . . . . . . . . . . . . . . .
 . . . . 1 . . . 1 . . . . . . . . . . . . . . . . . . .
 . . . 1 1 1 . . 1 . . . . . . . . . . . . . . . . . . .
 . . . . . . . . 1 1 1 . . . . . . . . . . . . . . . . .
 1 . . 1 1 . . . . . . . 1 1 . . . . . . . . . . . . . .
 . 1 . 1 1 . . . 1 . . . . 1 1 . . . . . . . . . . . . .
 . . 1 1 1 . . . . 1 . . . 1 1 . . . . . . . . . . . . .
 . . . 1 1 . . . 1 1 . . . 1 1 . . . . . . . . . . . . .

ls[3] =
 . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```



ls[4] =



ls[5] =



Bibliography

[AAFG01] Iris Anshel, Michael Anshel, Benji Fisher, and Dorian Goldfeld. New key agreement protocols in braid group cryptography. In *CT-RSA*, pages 13–27, 2001.

[Art55] E. Artin. The orders of linear groups. *Communications on Pure and Applied Mathematics*, 8:355 – 366, 1955.

[BCM09] S.R. Blackburn, C. Cid, and C. Mullan. Cryptanalysis of the MST 3 public key cryptosystem. *Journal of Mathematical Cryptology*, 3(4):321–338, 2009.

[BGCS05] J. Bohli, M.I. Gonzáles Vasco, Martínez C.J.M., and R. Steinwandt. Weak keys in MST 1. *Designs, Codes and Cryptography*, 37(3):509–524, 2005.

[BH81] N. Blackburn and B. Huppert. *Finite Groups II*. Springer, 1981.

[BJG08] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer Verlag, 2008.

[BO81] M. Ben-Or. Probabilistic algorithms in finite fields. *IEEE Symposium on Foundations of Computer Science*, 22:394 – 398, 1981.

[Boz08] V. Bozovic. *Factorization of finite groups*. Verlag Dr. Müller, 2008.

[BP08] V. Bozovic and N. Pace. On Group Factorizations using Free Mappings. *Journal of Algebra and Its Applications*, 7(5):647–662, 2008.

[Coh93] H. Cohen. *A Course in Computational Number Theory*. Springer, 1993.

[Cus00] C.A. Cusack. Group factorizations in cryptography. *ETD collection for University of Nebraska-Lincoln*, 2000.

[CV06] A. Caranti and F.D. Volta. The round functions of cryptosystem PGM generate the symmetric group. *Designs, Codes and Cryptography*, 38(1):147–155, 2006.

[DH76] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22:644 – 654, 1976.

[Die51] J. Dieudonné. On the automorphisms of classical groups. *Memoirs of the American Mathematical Society*, 2, 1951.

[ElG84] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *Crypto '84*, pages 10 – 18, 1984.

[Fis97] G. Fischer. *Lineare Algebra*. Vieweg, 1997.

[GAPrg] The GAP Group. *GAP – Groups, Algorithms, and Programming, Version 4.4.10*, http://www.gap-system.org.

[GM84] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270 – 299, 1984.

[Gol01] O. Goldreich. *Foundations of Cryptography: Basic Tools*. Cambridge University Press, 2001.

[Gol04] O. Goldreich. *Foundations of cryptography: Basic Applications*. Cambridge University Press, 2004.

[GRS03] M.I. Gonzáles Vasco, M. Rotteler, and R. Steinwandt. On minimal length factorizations of finite groups. *Experimental Mathematics*, 12(1):1–12, 2003.

[GS02] M.I. Gonzáles Vasco and R. Steinwandt. Obstacles in two public key cryptosystems based on group factorizations. *Tatra Mountains*, 25:23–37, 2002.

[Hig63] G. Higman. Suzuki 2-groups. *Illinois journal of mathematics*, 7(1):79–96, 1963.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In *ANTS*, pages 267–288, 1998.

[KL08] J. Katz and Y. Lindell. *Introduction to Modern Cryptography.* Chapman & Hall/CRC, 2008.

[Kor05] A. Korsten. The discrete logarithm problem in semidirect products and the reduction of the mor cryptosystem. *Diplomarbeit*, 2005.

[Kor08] A. Korsten. Cryptanalysis of mor and discrete logarithms in inner automorphism groups. *Proceedings of WEWORC 2007*, LNCS 2139:470 – 485, 2008.

[KS98] H. Kurzweil and B. Stellmacher. *Theorie der endlichen Gruppen.* Springer, 1998.

[L+04] I.-S. Lee et al. On the security of MOR public key cryptosystem. *ASIACRYPT 2004*, LNCS 3329:387 – 400, 2004.

[LN94] R. Lidl and H. Niederreiter. *Introduction to Finite Fields and their Application.* Cambridge University Press, 1994.

[LvTMW09] W. Lempken, T. van Trung, S.S. Magliveras, and W. Wei. A public key cryptosystem based on non-abelian finite groups. *Journal of Cryptology*, 22(1):62–74, 2009.

[Mag86] S. Magliveras. A cryptosystem from logarithmic signatures of finite groups. *Proceedings of the 29th Midwest Symposium on Circuits and Systems*, pages 972–975, 1986.

[Mao03] W. Mao. *Modern Cryptography: Theory and Practice.* Prentice Hall PTR, 2003.

[Men93] A. J. Menezes, editor. *Applications of Finite Fields.* Kluwer Academic Publishers, 1993.

[MM90] S. Magliveras and N. Memon. Properties of cryptosystem PGM. In *Advances in Cryptology - CRYPTO'89 Proceedings*, pages 447–460. Springer, 1990.

[MM92] S. Magliveras and N.D. Memon. Algebraic Properties of Cryptosystem PGM. *Journal of Cryptology*, 5(3):167–183, 1992.

[MSvT02] S. Magliveras, D. Stinson, and T. van Trung. New approaches to designing public key cryptosystems using one-way functions and trap-doors in finite groups. *Journal of Cryptology*, 15:285–297, 2002.

[MSvTZ08] S. Magliveras, P. Svaba, T. van Trung, and P. Zajac. On the security of a realization of cryptosystem MST3. *Tatra Mountains*, 2008.

[MvOV96] A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[MW98] A. J. Menezes and Y. Wu. The discrete logarithm problem in GL(n,p). *Ars Combinatoria*, 47:23 – 32, 1998.

[O'M74] O. T. O'Meara. Lectures on linear groups. Regional Conference Series in Mathematics, 22, AMS, 1974.

[P+01a] S. H. Paeng et al. Improved public key cryptosystem using finite non abelian groups. *IACR EPrint-Server, Report 2001/066*, 2001. http://eprint.iacr.org/2001/066.

[P+01b] S. H. Paeng et al. New public key cryptosystem using finite non abelian groups. *Advances in Cryptology - Crypto 2001*, LNCS 2139:470 – 485, 2001.

[Pae03] S. H. Paeng. On the security of cryptosystem using automorphism groups. *Information Processing Letters*, 88:293 – 298, 2003.

[Pat96] Jacques Patarin. Hidden fields equations (hfe) and isomorphisms of polynomials(ip): Two new families of asymmetric algorithms. In *EUROCRYPT*, pages 33–48, 1996.

[PH78] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24(1):106–110, 1978.

[QV94] M. Qu and S.A. Vanstone. Factorizations in the elementary abelian p-group and their cryptographic significance. *Journal of Cryptology*, 7(4):201–212, 1994.

[Rob82] D. J. S. Robinson. *A Course in the Theory of Groups*. Graduate Texts in Mathematics. Springer, 1982.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *FOCS*, pages 124–134, 1994.

[SSM10] N. Singhi, N. Singhi, and S. Magliveras. Minimal logarithmic signatures for finite groups of Lie type. *Designs, Codes and Cryptography*, 55(2):243–260, 2010.

[Ste60] R. Steinberg. Automorphisms of finite linear groups. *Canadian Journal of Mathematics*, 22:606 – 615, 1960.

[Sti06] D.R. Stinson. *Cryptography: Theory and Practice*. CRC press, 2006.

[SvdW28] O. Schreier and B. L. van der Waerden. Die Automorphismen der projektiven Gruppen. *Abh. math. Seminar Hamburg*, 6:303 – 322, 1928.

[Sza04] S. Szabó. *Topics in Factorization of Abelian Groups*. Birkhäuser, 2004.

[Tay87] D. Taylor. Pairs of generators for matrix groups, I. *The Cayley Bulletin*, 3:76 – 85, 1987.

[Tob03] C. Tobias. Security analysis of the MOR cryptosystem. *Proceedings of the PKC 2003*, LNCS 2567:175 – 186, 2003.

[Tob04a] C. Tobias. *Design und Analyse kryptographischer Bausteine auf nicht-abelschen Gruppen*. PhD thesis, Justus-Liebig-Universität Gießen, 2004.

[Tob04b] C. Tobias. Security analysis of MOR using $GL(2, R) \times \mathbb{Z}\_p$. *WOSIS*, 2:170 – 179, 2004.

[vdW35] B. L. van der Waerden. *Gruppen von linearen Transformationen*. Springer, 1935.

[vzGG99] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, 1999.

[Web91] W. Webb. Cryptography using complementing subsets of polynomials over finite fields. *Finite Fields, Coding Theory and Advances in Communications and Computing*, pages 7–10, 1991.

[Weg03] I. Wegener. *Komplexitätstheorie*. Springer, 2003.