Some results in group-based cryptography

Ciaran Mullan

Technical Report
RHUL–MA–2012–1
10 January 2012

Department of Mathematics
Royal Holloway, University of London
Egham, Surrey TW20 0EX, England

http://www.rhul.ac.uk/mathematics/techreports

Some Results in Group-Based Cryptography

Ciaran Mullan

Thesis submitted to the University of London

for the degree of Doctor of Philosophy

2011

for Noel and Sheelah

Declaration of Authorship

I, Ciaran Mullan, hereby declare that this thesis and the work presented in it is entirely my own. Parts of this thesis are based on the following papers:

- S. R. Blackburn, C. Cid, C. Mullan, Group theory in cryptography, in: C. M. Campbell, M. R. Quick, E. F. Robertson, C. M. Roney-Dougal, G. C. Smith and G. Traustason (editors), *Proceedings of Groups St Andrews 2009 in Bath Volume 1*, Cambridge University Press, 133–149, 2011.
- S. R. Blackburn, C. Cid, C. Mullan, Cryptanalysis of the MST_3 public key cryptosystem, *J. Math. Crypt.* **3** (2009) 321–338.
- C. Mullan, Cryptanalysing variants of Stickel's key agreement protocol, *J. Math. Crypt.* **4** (4) (2011) 365–373.
- S. R. Blackburn, C. Cid, C. Mullan, Cryptanalysis of three matrix-based key establishment protocols, *J. Math. Crypt.* **5** (2) (2011) 159–168.

In the case of joint authorship, all authors contributed equally. Some results of Chapter 3 were discovered jointly with Boaz Tsaban. All computer experiments were performed by the author.

Signed:

Ciaran Mullan

Date: 8 January 2012

Abstract

Group-based cryptography is concerned with the role of nonabelian groups in cryptography. Since its origins in the 1980s, there have been numerous cryptographic proposals based on nonabelian groups, many of which have been broken. This thesis adds to the cryptanalytic literature by demonstrating the insecurity of several group-based cryptosystems. We cryptanalyse several key establishment protocols based on matrix groups, due to (i) Baumslag, Camps, Fine, Rosenberger and Xu, (ii) Habeeb, Kahrobaei, and Shpilrain, (iii) Romanczuk and Ustimenko, and (iv) a modified version of a scheme by Stickel. We also cryptanalyse the MST_3 public key cryptosystem and treat the Tillich–Zémor hash function.

Acknowledgments

My deepest thanks to Carlos and Simon, for shining a light on my path; I have learned so much from you both. Thanks to Boaz, a constant source of inspiration. Thanks to everyone at Royal Holloway, especially Martin, for Sage advice, Gary, for the epigrams, and Ami, for all the coffee and cake.

Contents

List of Tables

Chapter 1

Introduction

In recent times numerous cryptographic protocols have emerged based on group theoretic concepts. Such protocols have not yet led to practical schemes to rival the likes of RSA and Diffie–Hellman, but the ideas are interesting and have led to some rewarding group theory.

With the realisation that quantum computers can efficiently solve both the Integer Factorisation Problem and standard variants of the Discrete Logarithm Problem [79], the search for alternative cryptosystems has become more important. Cryptosystems, including group-based examples, that are not necessarily vulnerable to quantum adversaries have become known as post-quantum cryptosystems. A well-known example is the McEliece cryptosystem based on the difficulty of decoding error-correcting codes. Other examples include lattice-based cryptosystems and cryptosystems based on large systems of multivariate polynomial equations.

This thesis is concerned with group-based cryptography: the design and analysis of cryptographic schemes based on nonabelian groups. More specifically, we explore the role of matrix groups as a platform for group-based cryptosystems. If one is to implement a group-based cryptosystem, one needs an efficient method of representing, storing and manipulating group elements. For these reasons matrix groups are an attractive source of nonabelian groups: matrices are easy to store and represent on a computer, and linear algebra provides an efficient tool to manipulate elements. But linear algebra is also a powerful tool for the cryptanalyst. Indeed, the focus of this thesis is on cryptanalysis. We demonstrate the insecurity of several group-based cryptosystems that employ matrix groups as a platform, rendering them unfit for use in the real world. We treat several key establishment

protocols, a public key cryptosystem, and a group-based hash function. With the exception of the hash function (where we will be concerned with collision resistance) we work in the passive adversary model. Thus we assume that an adversary has the power only of an eavesdropper who knows everything about the system being used except for secret keys and random choices made by individual parties.

An outline of this thesis is as follows. In this introductory chapter we give a short account of the emergence of group-based cryptography. In Chapter 2 we cryptanalyse several matrix-based key establishment protocols [9, 41, 76, 86]. In Chapter 3 we study the Tillich–Zémor hash function [92], and in Chapter 4 we cryptanalyse the public key cryptosystem MST_3 [50]. We close with some remarks and directions for future work in Chapter 5.

We assume the reader has some familiarity with basic concepts from abstract algebra and cryptography. All the necessary prerequisites may be found, for example, in Fraleigh [29] and Stinson [88].

The rest of this chapter is structured as follows. In Section 1.1 we provide some background material and describe two well-known key establishment protocols: the Diffie–Hellman Protocol and Shamir's 'no-key' Protocol; this serves as motivation for the ensuing discussion on group-based protocols. In Section 1.2 we describe the two most widely studied schemes in group-based cryptography, and in Section 1.3 we discuss variants of these schemes. In Section 1.4 we mention some symmetric schemes involving nonabelian groups. We sketch some successful attacks on group-based schemes in Section 1.5, and close in Section 1.6 with a brief discussion.

We remark that this introductory chapter is based on the survey paper [11]. More detailed surveys of group-based cryptography are given by Dehornoy [26], Garber [30], and Myasnikov et al. [67].

1.1 Cryptography using Groups

The first proposal to use nonabelian groups in public key cryptography is due to Wagner and Magyarik [94] in 1985. The cryptosystem is based on the hardness of the Word Problem (or more accurately the Word Choice Problem) for finitely presented groups. However, the scheme is rather theoretical with several unresolved issues: critiques are given by González Vasco and Steinwandt [39] and Levy-dit-Vehel and Perret [52, 53].

The importance of Wagner and Magyarik's scheme lies in its novelty, which commenced an interplay between cryptography and combinatorial group theory. Let G be a group given by a finite presentation. In 1911, Max Dehn [27] posed the following problems:

- The Word Problem: given a word w on the generators of G, decide if $w = 1$ in G.
- The Conjugacy Problem: given words u, v on the generators of G, decide if u and v represent conjugate elements in G.
- The Isomorphism Problem: given two finite presentations, decide if they present isomorphic groups.

The study of these problems has proven remarkably fruitful. For example, a famous result due (independently) to Novikov [69] and Boone [16] states that there exists a finitely presented group with recursively unsolvable word problem. The introduction of group-based cryptography has put a twist on some of these well-studied decision problems. Suppose one is given two elements $g, h \in G$, together with knowledge that they are conjugate in G. The task is to find some $x \in G$ such that $g = x^{-1}hx$. This is not a decision problem, rather it is a search problem, known as the Conjugacy Search Problem. It forms the basis for two high profile group-based key agreement protocols due to Anshel, Anshel and Goldfeld [3] in 1999 and Ko, Lee, Cheon, Han, Kang and Park [49] in 2000. Ko et al. also describe a

public key encryption scheme, the security of which is also based on the Conjugacy Search Problem.

The key agreement scheme of Ko et al. may be seen as a generalisation of the Diffie–Hellman Protocol to nonabelian groups, with the Conjugacy Search Problem acting as a group-theoretic analogue of the Discrete Logarithm Problem. Thus prior to discussing the schemes of Anshel et al. and Ko et al. we recall the Diffie–Hellman Protocol [28] and the Discrete Logarithm Problem. Later, in chapter 2, we will discuss a group-theoretic analogue of Shamir's 'no-key' Protocol [61, Page 500], so we also describe this protocol here.

Diffie–Hellman Key Agreement Protocol

Let $G = \langle g \rangle$ be a cyclic group, with both g and its order d publicly known. To create a shared key, Alice and Bob proceed as follows:

1. Alice selects uniformly at random an integer $a \in [2, d-1]$ and sends g^a to Bob.

2. Bob selects uniformly at random an integer $b \in [2, d-1]$ and sends g^b to Alice.

3. Alice computes $k_a = (g^b)^a \in G$. Bob computes $k_b = (g^a)^b \in G$.

4. The shared key is $k_a = k_b \in G$.

Shamir's 'no-key' Key Transport Protocol

Let $G = \langle g \rangle$ be a cyclic group, with both g and its order d publicly known. To create a shared key, Alice and Bob proceed as follows:

1. Alice selects uniformly at random a key $k = g^r \in G$, an integer $a \in [2, d-1]$, and sends $k^a \in G$ to Bob.

2. Bob selects uniformly at random an integer $b \in [2, d-1]$, and sends $(k^a)^b = k^{ab} \in G$ to Alice.

3. Alice computes $(k^{ab})^{a^{-1}} = k^b \in G$ and sends k^b to Bob.

4. Bob computes $(k^b)^{b^{-1}} = k$.

The security of the Diffie–Hellman Protocol relies on the assumption that knowing $g \in G$ and having observed both g^a and g^b, it is computationally infeasible for an adversary to compute g^{ab}. This is known as the Diffie–Hellman Problem. The Diffie–Hellman Problem is related to the well-known Discrete Logarithm Problem (DLP): given $h \in G = \langle g \rangle$, find an integer t such that $g^t = h$. If one can efficienty solve the DLP then one can efficiently solve the Diffie–Hellman Problem and thus break the Diffie–Hellman Protocol. Similarly Shamir's 'no-key' Protocol relies on the hardness of the DLP. Thus as a minimum requirement one is interested in finding difficult instances of the DLP. The difficulty of the DLP depends heavily on the way the group G is represented, not just on the isomorphism class of G. For example, the DLP is trivial if $G = \mathbb{Z}/d\mathbb{Z}$ is the additive group generated by $g = 1$. However, if G is an appropriately chosen group then solving the DLP is considered computationally infeasible. In practice one often uses $G = \mathbb{F}_{p^l}^*$ for an appropriately selected prime p and exponent l, or the group of points of a properly chosen elliptic curve over a finite field.

1.2 From Exponentiation to Conjugacy

Let G be a nonabelian group. For $g, x \in G$ write g^x for $x^{-1}gx$, the conjugate of g by x. At first sight, the notation suggests that conjugation might be used instead of exponentiation in cryptographic contexts. Assuming that we can find a group where the Conjugacy Search Problem is hard, and assuming the elements of this group are easy to store and manipulate, one can define cryptosystems that are analogues of cryptosystems based on the DLP.

Ko et al. [49] proposed the following analogue of the Diffie–Hellman Protocol.

Ko–Lee–Cheon–Han–Kang–Park Key Agreement Protocol

Let G be a nonabelian group and let g be a publicly known element of G. Let A, B be commuting subgroups of G, that is $ab = ba$ for all $a \in A$, $b \in B$. The groups A, B

and G are public. To create a shared key, Alice and Bob proceed as follows:

1. Alice selects an element $a \in A$ and sends $g^a = a^{-1}ga$ to Bob.
2. Bob selects an element $b \in B$ and sends $g^b = b^{-1}gb$ to Alice.
3. Alice computes $k_a = (g^b)^a \in G$. Bob computes $k_b = (g^a)^b \in G$.
4. Since $ab = ba$, we have $k_a = k_b \in G$.

An important subtlety of the protocol is that although $k_a = k_b$ as group elements, their bitstring representations might be different. But for many groups, a shared key may be derived from k_a and k_b. For example, if G has an efficient algorithm to compute a normal form for a group element, then the shared key can be extracted from the normal form of k_a and k_b. We can summarise the minimum security requirements on the platform group G:

- The group G should have two large commuting subgroups (i.e. of exponential size in some appropriate security parameter, word length with respect to some normal form, for example).
- The group G should have an efficient normal form algorithm.
- The Conjugacy Search Problem in G should be generically hard.

Interest in Ko et al.'s proposal centred on their choice for G and subgroups A and B. The group G is taken to be the braid group B_n on n strings which has presentation

$$B_n = \left\langle \sigma_1, \sigma_2, \ldots, \sigma_{n-1} \,\middle|\, \begin{array}{ll} \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j & \text{for } |i-j| = 1 \\ \sigma_i\sigma_j = \sigma_j\sigma_i & \text{for } |i-j| \geq 2 \end{array} \right\rangle .$$

Let l and r be integers such that $l + r = n$, and define subgroups

$$A = \langle \sigma_1, \sigma_2, \ldots, \sigma_{l-1} \rangle \, , \; B = \langle \sigma_{l+1}, \sigma_{l+2}, \ldots, \sigma_{l+r-1} \rangle \, .$$

The braid group is an attractive choice of platform, since there is an efficient normal form for group elements and group multiplication and inversion can be carried out efficiently.

However, the cryptosystem is not fully specified: besides choosing the values of n, l and r, one must decide how to sample elements $g \in G$, $a \in A$ and $b \in B$. Since the groups G, A and B are infinite, it is not obvious how this should be done.

The following key agreement protocol due to Anshel, Anshel and Goldfeld [3] has an advantage over the Ko et al. protocol: commuting subgroups A and B are not needed.

Anshel–Anshel–Goldfeld Key Agreement Protocol

Let G be a nonabelian group, and let elements $a_1, \ldots, a_l, b_1, \ldots, b_m \in G$ be public. To create a shared key, Alice and Bob proceed as follows:

1. Alice selects a word x in $a_1, \ldots, a_l$ and sends $b_1^x, \ldots, b_m^x$ to Bob.
2. Bob selects a word y in $b_1, \ldots, b_m$ and sends $a_1^y, \ldots, a_l^y$ to Alice.
3. Alice computes x^y. Bob computes y^x.
4. The shared key is the commutator $[x, y] := x^{-1}y^{-1}xy$.

Note that Alice and Bob can both compute the shared commutator: Alice can premultiply x^y by x^{-1} and Bob can premultiply y^x by y^{-1} and then compute the inverse: $[x, y] = (y^{-1}y^x)^{-1}$. Anshel et al. also proposed using braid groups as a platform for the protocol. Note again that the scheme is not fully specified: one needs to state how elements a_i, b_j are chosen and how Alice and Bob generate words x, y.

Picking Elements at Random

The two aforementioned schemes leave open specification details as to how one samples elements from an infinite group such as a braid group. Note that the protocols work equally well for finite groups, in which case one can choose elements uniformly at random. But for infinite groups, the meaning of the word random is unclear, and one needs to specify precisely how group elements are sampled. Let $G = \langle X|R\rangle$ be a finitely presented group. One common method to select elements from G is to choose at random a sequence of integers $j_1, \ldots, j_l$ (from some finite

subset of $\mathbb{Z}$) and for each $1 \leq i \leq l$, select uniformly at random a generator $x_i \in X$. The 'random' element $x_1^{j_1} \cdots x_l^{j_l}$ is then output.

The process of selecting group elements for both the Ko et al. scheme and the Anshel et al. scheme, turns out to be critical for their security. We elaborate on this point in Section 1.5.

1.3 Replacing Conjugation

The Ko et al. scheme uses conjugation in place of exponentiation in the Diffie–Hellman Protocol, but there are many other alternatives. For example, one can define $g^a = \phi(a)ga$ and $g^b = \phi'(b)gb$ for any fixed functions $\phi : A \to A$ and $\phi' : B \to B$ (including the identity maps) and the scheme would work just as well. More generally, we may replace a and $\phi(a)$ by unrelated elements from A.

Since the papers of Ko et al. and Anshel et al. there have been many proposals based on variants of the Conjugacy Search Problem, such as the Decomposition Search Problem and the Twisted Conjugacy Search Problem. Myasnikov et al. [67] provide a discussion on these problems.

1.4 Symmetric Schemes

Group theory has mainly been used in proposals of public key cryptosystems and key establishment schemes, but has also been used in symmetric cryptography. One example is the block cipher PGM based on the symmetric group; we discuss the scheme in Chapter 4. In general, a block cipher (such as DES or AES) may be regarded as a set $\mathcal{S}$ of permutations on the set of all blocks, indexed by the key. The question as to whether $\mathcal{S}$ is in fact a group has an impact on the cipher's security in some situations: if the set is a group, then encrypting a message twice over using the cipher with different keys would be no more secure than a single encryption. However, computing the group generated by a block cipher is often very difficult. For instance, it is known that the group generated by the DES block cipher is a

subgroup of the alternating group $A_{2^{64}}$ [95], with order greater than 2^{56} (and thus $\mathcal{S}$ for DES is not a group [18, 25]); however little more is known about its structure.

Block ciphers are often built as iterated constructions of simpler key-dependent permutations known as round functions, and one can study properties of the permutation groups generated by these round functions. It has been shown, for instance, that the round functions of both DES and AES block ciphers are even permutations; furthermore it can be shown that these generate the alternating groups $A_{2^{64}}$ and $A_{2^{128}}$, respectively [83, 95, 96].

Hash function design is another area of symmetric cryptography where groups have been used in an interesting way. Zémor [98] proposed using walks through Cayley graphs as a basis for hash functions; the most well-known concrete proposal from this idea is a hash function of Tillich and Zémor [92]. We discuss this hash function in Chapter 3.

1.5 Cryptanalysis

In this section we outline techniques developed to demonstrate the insecurity of many group-based schemes.

In 1969, Garside [32] gave the first algorithm to solve the Conjugacy Problem in the braid group B_n. The question of efficiency of Garside's method laid dormant until the late 1980's. Since then there has been a great deal of research, significantly motivated by cryptographic applications, into finding a polynomial time solution to the Conjugacy Problem. Given two braids $x, y \in B_n$, Garside's idea is to construct finite subsets (so called *summit sets*) I_x, I_y of B_n such that x is conjugate to y if and only if $I_x = I_y$. An efficient solution to the Conjugacy Problem via this method would also yield an efficient solution to the Conjugacy Search Problem (and hence render protocols based on the braid Conjugacy Search Problem theoretically insecure). However, for a given braid x, Garside's summit set I_x may be exponentially large. The challenge has thus been to prove a polynomial bound on the size of a suitable invariant set associated with any given conjugacy class. Re-

finements to the summit set method (such as the *super summit set*, *ultra summit set*, and *reduced super summit set* methods) have been made over the years, but a polynomial bound remains elusive. Recent focus has been on an efficient solution to each of the three types of braids: periodic, reducible or pseudo-Anasov, according to the Nielsen–Thurston classification [5, 6, 7].

For the purposes of cryptography however, one need not efficiently solve the Conjugacy Search Problem in order to break a braid-based cryptosystem: one is free to use the specifics of the protocol being employed. An algorithm need work only on a significant proportion of cases, and heuristic algorithms have proven quite successful.

Indeed, Hofheinz and Steinwandt [43] used a heuristic algorithm to solve the Conjugacy Search Problem for braid groups with very high success rates. Their attack is based on the observation that representatives of conjugate braids in the super summit set are likely to be conjugate by a permutation braid, which has a particularly simple structure. Their attack demonstrates inherent weaknesses of both the Ko et al. protocol and the Anshel et al. protocol for random instances, under suggested parameters. Around the same time, several other powerful lines of attack were discovered, which we now mention.

Length Based Attacks. Introduced by Hughes and Tannenbaum [45], length based attacks provide a neat probabilistic way of solving the Conjugacy Search Problem in certain cases. Suppose one is given an instance of the Conjugacy Search Problem in B_n. So one is given braids $x, y^{-1}xy$ and want to find y. Let $l : B_n \to \mathbb{Z}$ be a suitable length function on B_n (for example, the length of the normal form of an element). If one can write $y = y'\sigma_i$ for some i, where y' has a shorter length than y, then $l(\sigma_i y^{-1}xy\sigma_i^{-1})$ should be strictly smaller than $l(\sigma_j y^{-1}xy\sigma_j^{-1})$ for $j \neq i$. So i can be guessed, and the attack repeated for a smaller instance y' of y. The success rate of this probabilistic attack depends on the specific length function employed. For braid groups, there are a number of suitable length functions that open up this line of attack. Garber et al. [31] and Myasnikov and Ushakov [65] provide convincing

attacks on both the Ko et al. and Anshel et al. protocols using this approach.

Linear Algebra Attacks. The idea behind this attack is quite simple: take a linear representation of the braid group and solve the Conjugacy Search Problem using linear algebra in a matrix group. There are two well-known representations of the braid group: the Burau representation (unfaithful for $n \geq 5$) and the faithful Lawrence-Krammer representation. Hughes [44] and Lee and Lee [46] provide convincing attacks on the Anshel et al. protocol using the Burau representation, and Cheon and Jun [19] provide a polynomial time algorithm to break the Ko et al. protocol using the Lawrence-Krammer representation.

1.6 Discussion

Many suggestions have been made to improve the security of the schemes discussed in this chapter. Themes range from changing the underlying problem (and instead investigating problems such as the Decomposition Problem, the Braid Root Problem, the Shifted Conjugacy Problem and more) to changing the platform group (Thompson's group, polycyclic groups and others have been suggested). Furthermore, cryptographers have created other cryptographic primitives based on the Conjugacy Search Problem, such as authentication schemes and signature schemes. However, there are no known cryptographic primitives based on any of these ideas that convincingly survive the aforementioned attacks.

On the bright side, group-based cryptography has motivated some natural questions for the group theorist, in particular the study of generic properties of groups. For example, Myasnikov and Ushakov [66] proved that pure braid groups PB_n satisfy the strong generic free group property: for any generating set of PB_n, when any k elements are chosen 'randomly' (as discussed above) they freely generate a free group of rank k generically.

We will see more examples of this interplay between cryptography and group theory throughout this thesis.

Chapter 2

Matrix-Based Key Establishment Protocols

This chapter is concerned with the security of key establishment protocols using matrix groups. There have been many such proposals made over the years, some of which may be seen as generalisations of protocols based on cyclic groups to a matrix group setting. The first such example is due to Odoni, Varadharajan and Sanders [68], who suggested generalising the Discrete Logarithm Problem to matrix groups and proposed a matrix version of the Diffie–Hellman Protocol. However, Menezes and Wu [63] demonstated a probabilistic polynomial time reduction for the DLP for $\mathrm{GL}_n(\mathbb{F}_q)$ to the DLP over small finite extension fields of $\mathbb{F}_q$. More recent examples include proposals by Alvarez et al. [2] and Climent et al. [22], which were successfully cryptanalysed by González Vasco et al. [34] and Climent et al. [23] respectively. In this chapter we add to the cryptanalytic literature by cryptanalying several matrix-based key establishment protocols. Our exposition is based on the papers [13, 64].

We begin with a matrix-based key transport protocol by Baumslag, Camps, Fine, Rosenberger and Xu from 2006 [9]. In fact, their proposal is more general and several platform groups were suggested, but we consider only their matrix group proposal. We cryptanalyse this scheme in a very strong sense. We show that for practical parameter sizes a passive adversary can feasibly recover the session key after observing just one run of the protocol. We find a better attack if two or more runs of the protocol are observed. Our techniques reduce the problem of breaking the scheme to a sequence of feasible Gröbner basis computations.

Next we cryptanalyse two recently proposed matrix-based key agreement protocols, due to Habeeb, Kahrobaei and Shpilrain [41], and due to Romanczuk and Ustimenko [76]. These schemes both fail due to straightforward linear algebra attacks.

Finally we discuss Stickel's key agreement scheme [86], which was successfully cryptanalysed by V. Shpilrain when $\mathrm{GL}_n(\mathbb{F}_q)$ is used as a platform [80]. Shpilrain suggested the algebra of all (not necessarily invertible) $n \times n$ matrices defined over some finite ring R would make a more secure platform. He also suggested a more general method of generating keys, involving polynomials of matrices over R. When $R = \mathbb{F}_q$, we show that these variants of Stickel's scheme are susceptible to a linear algebra attack. We discuss other natural candidates for R, and conclude that until a suitable ring is proposed, the variant schemes may be considered insecure.

The chapter is organised as follows. In Section 2.1 we cryptanalyse the scheme of Baumslag et al. In Sections 2.2 and 2.3 we cryptanalyse the schemes of Habeeb, Kahrobaei and Shpilrain, and Romanczuk and Ustimenko. In Section 2.4 we cryptanalyse variants of Stickel's scheme.

2.1 The BCFRX Scheme

We begin by describing a matrix-based key transport protocol by Baumslag, Camps, Fine, Rosenberger and Xu from 2006 [9], which we refer to as the BCFRX scheme. The protocol assumes that Alice and Bob a priori share some secret information, namely their long-term secret key. The goal of the protocol is for Alice and Bob to establish a session key for subsequent cryptographic use. To achieve this, Bob selects the session key and sends it to Alice in three passes, as follows.

Let $\mathcal{G}$ be a finitely presented group. Let $\mathcal{A}$ and $\mathcal{B}$ be two commuting subgroups of $\mathcal{G}$ (so $AB = BA$ for all $A \in \mathcal{A}$ and $B \in \mathcal{B}$). The group $\mathcal{G}$ is made public and the subgroups $\mathcal{A}$ and $\mathcal{B}$ form Alice and Bob's long-term secret key. Then:

1. Bob selects a session key $K \in \mathcal{G}$ and elements $B, B' \in \mathcal{B}$. He sends $C :=$

BKB' to Alice.

2. Alice selects elements $A, A' \in \mathcal{A}$ and sends $D := ACA' = ABKB'A'$ to Bob.

3. Since $\mathcal{A}$ and $\mathcal{B}$ commute, we have that $ABKB'A' = BAKA'B'$. Bob sends $E := B^{-1}DB'^{-1} = AKA'$ to Alice.

4. Alice computes $K = A^{-1}EA'^{-1}$.

One can think of this protocol as Shamir's 'no-key' Protocol (see Chapter 1), with the operation of multiplying on the left and right by a group element replacing the exponentiation operation.

There was no detailed discussion of security of the protocol in [9], but we need to specify a security model and what it means to break the protocol, in order to cryptanalyse it. We will consider the weakest possible notion of security: the passive adversary model. So we will regard the protocol as broken if we can demonstrate the existence of an adversary that can feasibly compute the session key, after eavesdropping on one or more runs of the protocol.

Baumslag et al. suggested several platform groups to serve for $\mathcal{G}$. But we concentrate on their only matrix group proposal: $\mathcal{G} = \mathrm{SL}_4(\mathbb{Z})$, the group of invertible 4×4 matrices of determinant 1 over the integers. It was proposed that the commuting subgroups $\mathcal{A}$ and $\mathcal{B}$ should be constructed as follows. Writing I_2 for the 2×2 identity matrix, define the subgroups $\mathcal{U}$ and $\mathcal{L}$ of $\mathcal{G}$ by

$$\mathcal{U} = \begin{pmatrix} \mathrm{SL}_2(\mathbb{Z}) & 0 \\ 0 & I_2 \end{pmatrix} \text{ and } \mathcal{L} = \begin{pmatrix} I_2 & 0 \\ 0 & \mathrm{SL}_2(\mathbb{Z}) \end{pmatrix}. \tag{2.1.1}$$

Let $M \in \mathrm{SL}_4(\mathbb{Z})$ be a secret matrix known only to Alice and Bob. Then we define

$$\mathcal{A} = M^{-1}\mathcal{U}M \text{ and } \mathcal{B} = M^{-1}\mathcal{L}M. \tag{2.1.2}$$

We may thus view the long-term secret key as the matrix M.

The protocol is not yet fully specified. It remains to specify how the long-term secret key M is chosen, and how Alice and Bob select elements from $\mathcal{A}$ and $\mathcal{B}$. It was stated in [9] that elements are chosen randomly from $\mathcal{A}$ and $\mathcal{B}$, and we presume that

the matrix M is chosen in a similar fashion from $\mathcal{G} = \mathrm{SL}_4(\mathbb{Z})$. But since the group $\mathcal{G}$ and its subgroups $\mathcal{A}, \mathcal{B}$ are infinite, the meaning of the word random is unclear in this context. Any practical cryptanalysis will depend on the details of how these random choices are made; however the cryptanalysis we give below will work for any efficient method for making these random choices that we can think of.

In any fully specified implementation of the protocol, there exists an integer Λ such that the entries of all matrices generated in the protocol lie in the interval $(-\Lambda/2, \Lambda/2)$. Since the standard way to represent a 4×4 integer matrix of this form uses approximately $16 \log_2 \Lambda$ bits, it is natural to think of $\log_2 \Lambda$ as the security parameter of the scheme.

A Cryptanalysis

Our cryptanalysis proceeds in three stages. In Stage 1, we argue that integer computations may be replaced by computations modulo p for various small primes p. In Stage 2 we show that knowledge of a matrix N of a restricted form allows a passive adversary to compute any session key transmitted under the scheme. Finally, in Stage 3, we show that this matrix N may be computed in practice. None of these stages is rigorous (though Stage 2 may be made so), but the stages all work well in practice.

Stage 1: Working Modulo p

Suppose an adversary wishes to discover a session key K. Since the entries of K lie in the interval between $-\Lambda/2$ and $\Lambda/2$, it is enough to find $K \bmod n$ for any $n > \Lambda$. Indeed, this is how we approach our cryptanalysis. We will show that in practice one may efficiently compute $K \bmod p_i$ for small primes p_i of our choice. (We are thinking of p_i as a prime of between 80 and 300 bits in length: in some sense quite large, but in general smaller than Λ.) We run this computation for several different primes p_i until $\prod p_i > \Lambda$. Setting $n = \prod p_i$, we can then appeal to the Chinese remainder theorem to calculate $K \bmod n = K$.

We write this more precisely as follows. Let T be a fully specified version of the BCFRX protocol, with $\mathrm{SL}_4(\mathbb{Z})$ as a platform. For a prime p, let $\mathbb{Z}_p$ be the integers modulo p. Let T_p be the BCFRX protocol under the platform group $\mathcal{G} = \mathrm{SL}_4(\mathbb{Z}_p)$, defined as follows. We identify the subgroups $\mathcal{U}$ and $\mathcal{L}$ defined by (2.1.1) with their images in $\mathrm{SL}_4(\mathbb{Z}_p)$. Let the subgroups $\mathcal{A}$ and $\mathcal{B}$ be chosen to be of the form (2.1.2) for some matrix $M \in \mathcal{G}$ chosen uniformly at random. Let Alice and Bob select all elements from $\mathcal{A}$ and $\mathcal{B}$ uniformly and independently at random. This makes sense since $\mathcal{G}$ is finite. We use T_p to model the protocol T taken modulo p. This model is not quite accurate: for example, it is almost certain that when $M \in \mathrm{SL}_4(\mathbb{Z})$ is chosen according to the method specified in T, the distribution of $M \bmod p$ will not be quite uniform in $\mathrm{SL}_4(\mathbb{Z}_p)$. But for all ways we can think of in which T can be specified, the protocol T_p is a good model for T taken modulo p (in the sense that an adversary that succeeds in practice to recover the session key generated by T_p will also succeed in practice to recover $K \bmod p$ when presented with the matrices from a run of the protocol T). Note that an adversary has great freedom in choosing p, which makes the reduction to T_p difficult to design against. The fact (see below) that the session key for T_p can be feasibly computed in practice shows that T is insecure.

Stage 2: Restricting the Long-Term Key

We consider the protocol T_p over $\mathrm{SL}_4(\mathbb{Z}_p)$ defined above. From now on, let us write an arbitrary 4×4 matrix Z in block form as $Z = \left(\begin{smallmatrix} Z_{11} & Z_{12} \\ Z_{21} & Z_{22} \end{smallmatrix}\right)$, for the obvious 2×2 submatrices Z_{ij} of Z.

The following lemma shows that there are many equivalent long-term keys for the protocol T_p.

Lemma 2.1. *Let $M \in \mathrm{SL}_4(\mathbb{Z}_p)$ be the long-term key shared by Alice and Bob, and define subgroups $\mathcal{A}$ and $\mathcal{B}$ by $\mathcal{A} = M^{-1}\mathcal{U}M$ and $\mathcal{B} = M^{-1}\mathcal{L}M$. Let $N \in \mathrm{GL}_4(\mathbb{Z}_p)$ be any matrix such that $N^{-1}\mathcal{U}N = \mathcal{A}$ and $N^{-1}\mathcal{L}N = \mathcal{B}$. If N is known, then any session key can be efficiently computed by a passive adversary.*

PROOF. An adversary is presented with matrices C, D and E that are transmitted as part of the protocol. We have that $C = BKB'$, $D = ABKB'A'$ and $E = AKA'$ for some unknown matrices $A, A' \in \mathcal{A}$ and $B, B' \in \mathcal{B}$. Suppose that the adversary is also able to obtain a matrix N satisfying the conditions of the lemma. Since $A, A' \in \mathcal{A}$ we may write $A = N^{-1}RN$ and $A' = N^{-1}R'N$ for some unknown matrices $R, R' \in \mathcal{U}$. Similarly we may write $B = N^{-1}SN$ and $B' = N^{-1}S'N$ for some unknown matrices $S, S' \in \mathcal{L}$.

Define a matrix K' by $K' = NKN^{-1}$. Define matrices C', D', E' by

$$\begin{aligned} C' &:= NCN^{-1} = NBKB'N^{-1} = SK'S', \\ D' &:= NDN^{-1} = NABKB'A'N^{-1} = RSK'S'R' \text{ and} \\ E' &:= NEN^{-1} = NAKA'N^{-1} = RK'R'. \end{aligned}$$

Note that the adversary can compute C', D' and E'.

Using the fact that $S, S' \in \mathcal{L}$ and $R, R' \in \mathcal{U}$, we may write

$$\begin{aligned} C' &= \begin{pmatrix} K'_{11} & K'_{12}S'_{22} \\ S_{22}K'_{21} & S_{22}K'_{22}S'_{22} \end{pmatrix}, \\ D' &= \begin{pmatrix} R_{11}K'_{11}R'_{11} & R_{11}K'_{12}S'_{22} \\ S_{22}K'_{21}R'_{11} & S_{22}K'_{22}S'_{22} \end{pmatrix} \text{ and} \\ E' &= \begin{pmatrix} R_{11}K'_{11}R'_{11} & R_{11}K'_{12} \\ K'_{21}R'_{11} & K'_{22} \end{pmatrix}. \end{aligned}$$

Clearly K'_{11} is known to the adversary, since $K'_{11} = C'_{11}$. Moreover, K'_{22} is known since $K'_{22} = E'_{22}$.

To compute K'_{12}, compute a matrix X such that $XD'_{12} = C'_{12}$ (note there may be more than one such X if K'_{12} is noninvertible). This implies $XR_{11}K'_{12} = K'_{12}$, since S'_{22} is invertible. Thus an adversary can compute $XE'_{12} = K'_{12}$. Similarly, to compute K'_{21}, compute a matrix Y such that $D'_{21}Y = C'_{21}$. This implies $K'_{21}R'_{11}Y = K'_{21}$ and an adversary can compute $E'_{21}Y = K'_{21}$.

Once K' is known, the session key K may be recovered since $K = N^{-1}K'N$.

This completes the proof.

Let $\mathrm{Mat}_2(\mathbb{Z}_p)$ be the set of 2×2 matrices over $\mathbb{Z}_p$. Let $\mathcal{I} \subseteq \mathrm{Mat}_2(\mathbb{Z}_p)$ be defined by

$$\mathcal{I} = \left\{ \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \right\}.$$

We say that $N \in \mathrm{GL}_4(\mathbb{Z}_p)$ is of *restricted form* if $N_{11}, N_{22} \in \mathcal{I}$.

Lemma 2.2. *For any long-term key M used in the protocol T_p, there is a matrix N of restricted form satisfying the conditions of Lemma 2.1. Moreover, for an overwhelming proportion of long-term keys M, we may impose the condition that $N_{11} = N_{22} = I_2$, where I_2 is the 2×2 identity matrix.*

PROOF. Let $f : \mathrm{Mat}_2(\mathbb{Z}_p) \to \mathrm{GL}_2(\mathbb{Z}_p)$ be a function such that $f(X)X \in \mathcal{I}$ for all $X \in \mathrm{Mat}_2(\mathbb{Z}_p)$. Such a function f certainly exists: it can be derived from a standard row reduction algorithm.

Define

$$H := \begin{pmatrix} f(M_{11}) & 0 \\ 0 & f(M_{22}) \end{pmatrix} \text{ and } N := HM.$$

The definition of H means that $N_{11}, N_{22} \in \mathcal{I}$, and so N is of restricted form. Also, any matrix

$$H \in \begin{pmatrix} \mathrm{GL}_2(\mathbb{Z}_p) & 0 \\ 0 & \mathrm{GL}_2(\mathbb{Z}_p) \end{pmatrix}$$

has the property that $H^{-1}\mathcal{U}H = \mathcal{U}$ and $H^{-1}\mathcal{L}H = \mathcal{L}$. So

$$N^{-1}\mathcal{U}N = M^{-1}H^{-1}\mathcal{U}HM = M^{-1}\mathcal{U}M = \mathcal{A}$$

and similarly $\mathcal{B} = N^{-1}\mathcal{L}N$. So the main statement of the lemma is proved. To see why the last statement of the lemma holds, note that for an overwhelming proportion of long-term keys M we have that M_{11} and M_{22} are invertible. The function f maps any invertible matrix to its inverse, and so $N_{11} = N_{22} = I_2$ in this case.

Stage 3: Computing the matrix N

We may compute an equivalent long-term key N of restricted form as follows. After eavesdropping on a run of the protocol, we know the matrices C, D, and E. Also,

the matrix N of restricted form must satisfy the equations

$$NDN^{-1} = RNCN^{-1}R', \tag{2.1.3}$$

$$NDN^{-1} = SNEN^{-1}S', \tag{2.1.4}$$

$$NN^{-1} = I_4, \tag{2.1.5}$$

for unknown matrices $R, R' \in \mathcal{U}, S, S' \in \mathcal{L}$. Since N is of restricted form we have $N_{11}, N_{22} \in \mathcal{I}$. There are thus only 9 possibile combinations for N_{11} and N_{22}, so we may perform a trivial exhaustive search to find the right combination. (In practice we would first try $N_{11} = N_{22} = I_2$, since this holds with overwhelming probability.) We assign variables $x_1, \ldots, x_8$ for the remaining unknown entries of N, and $x_9, \ldots, x_{24}$ for the unknown entries of N^{-1}.

Expanding (2.1.3) and (2.1.4), we find

$$(NDN^{-1})_{22} = (NCN^{-1})_{22}, \;\; (NDN^{-1})_{11} = (NEN^{-1})_{11}. \tag{2.1.6}$$

This gives us $4 + 4 = 8$ quadratic equations in the $x_i, i = 1, \ldots, 24$. Adding the 16 quadratic equations from (2.1.5), we have a system of 24 quadratic equations in 24 unknowns and expect a Gröbner basis calculation to reveal N. If we eavesdrop on a second run of the protocol, we learn 8 new equations (from (2.1.3) and (2.1.4)) and expect to compute N more easily.

Experimental results

Over 1,000 trials in Magma on a Intel Core 2 Duo 1.86GHz desktop, it took roughly 12 seconds to compute each Gröbner basis for a random 300-bit prime. In all our experiments, twenty three of the basis elements had the form $x_i + f_i(x_{24})$ for $i = 1, \ldots, 23$, where f_i is a polynomial of degree 5. The final basis element was a degree 6 polynomial in x_{24}. Thus in all our cases we had a maximum of six possibilities for a matrix N of restricted form satisfying the conditions of Lemma 2.1.

If we eavesdrop on a second run of the protocol, we can add 8 new equations to our system. A Gröbner basis calculation then reveals a unique value for N. We

can run the attack for different primes p_i of our choice until $\prod p_i > \Lambda$ and use the Chinese remainder theorem to calculate K.

We conclude that the BCFRX scheme is insecure when $\mathrm{SL}_4(\mathbb{Z})$ is used as a platform. Implementation details of our attack are given in the Appendix.

2.2 The HKS Scheme

Next we turn our attention to a key agreement protocol proposed by Habeeb, Kahrobaei and Shpilrain [41], which we refer to as the HKS scheme.

Let H and Q be two groups and let $\phi : Q \to \mathrm{Aut}(H)$ be a homomorphism, where $\mathrm{Aut}(H)$ denotes the automorphism group of H. The *semidirect product* $H \rtimes_\phi Q$ of H and Q is defined as the group $\{(h, q) : h \in H, q \in Q\}$ with group operation

$$(h, q)(h', q') = (h\phi(q)h', qq').$$

For the HKS scheme we let A be a group and let B be an abelian group. Let $A, B, \mathrm{Aut}(B), a \in A, b \in B, n \in \mathbb{N}$ be public. To create a shared key, Alice and Bob proceed as follows:

1. Alice selects an embedding $\psi : A \to \mathrm{Aut}(B)$ and sends to Bob

$$(x_1, x_2) := (b\psi(a)(b)\psi(a^2)(b) \dots \psi(a^{n-1})(b), a^n) = (b, a)^n \in B \rtimes_\psi A.$$

2. Bob selects an embedding $\phi : A \to \mathrm{Aut}(B)$ and sends to Alice

$$(y_1, y_2) := (b\phi(a)(b)\phi(a^2)(b) \dots \phi(a^{n-1})(b), a^n) = (b, a)^n \in B \rtimes_\phi A.$$

3. Alice computes $b^{-1}y_1 := y$, followed by

$$k_A = \prod_{i=1}^{n-1} \psi(a^i)(y) = \prod_{i=1}^{n-1} \prod_{j=1}^{n-1} \psi(a^i)\phi(a^j)(b).$$

4. Bob computes $b^{-1}x_1 := x$, followed by

$$k_B = \prod_{i=1}^{n-1} \phi(a^i)(x) = \prod_{i=1}^{n-1} \prod_{j=1}^{n-1} \phi(a^i)\psi(a^j)(b).$$

We require that Alice and Bob pick ψ and ϕ so that $\psi(a)\phi(a) = \phi(a)\psi(a)$. If this is done, Alice and Bob have computed a shared key $k = k_A = k_B$.

Before discussing the choice of platform group, we note the following simplifaction to the scheme. Alice does not need to send (x_1, x_2) to Bob, but only x, since Bob uses only x to derive the key. Moreover an adversary can easily compute x from (x_1, x_2), since b is public. Likewise Bob only need send y to Alice. We suppose from now on that this simplification is in place.

The proposal [41] suggests to let A to be a p-group and B to be an elementary abelian p-group of order p^m. Thus B may be viewed as an m-dimensional vector space over $\mathbb{F}_p$, and so $\mathrm{Aut}(B) = \mathrm{GL}_m(\mathbb{F}_p)$. With this choice of platform groups, we can view the protocol as follows.

Define $f(x) = x + x^2 + \cdots + x^{n-1}$. Let b be an m-dimensional column vector over $\mathbb{F}_p$. Alice and Bob choose private $m \times m$ matrices J and K respectively, using some method so that $f(J)$ and $f(K)$ commute. In general, and a little more formally, $J = M_A(r_A)$ and $K = M_B(r_B)$ where M_A and M_B are public algorithms which take as input random sequences of coin tosses r_A and r_B respectively (in addition to the public parameters of the scheme). The algorithms must have the property that the matrices $f(M_A(r_A))$ and $f(M_B(r_b))$ commute for all input sequences r_A and r_B respectively. The paper [41] suggests some candidates for M_A and M_B, but we do not make use of the details of these algorithms in this cryptanalysis.

Alice transmits the column vector $w_A = f(J)b$ to Bob. Bob transmits the column vector $w_B = f(K)b$ to Alice. The common key k is the column vector defined by

$$k = f(J)f(K)v = f(J)w_B = f(K)w_A,$$

the last equality following since $f(J)$ and $f(K)$ commute.

A Cryptanalysis

Suppose an adversary Eve knows the public parameters of the scheme and observes w_A, w_B during transmission.

Let X be any matrix such that $Xb = w_A$, and X commutes with $f(L)$ for all matrices L that can possibly be generated by Bob. Such a matrix exists since $X = f(J)$ satisfies these conditions.

Note that the conditions on the unknown entries of X are linear. This is clear for the condition that $Xb = w_A$. The commutator condition can be expressed as $Xf(L) = f(L)X$, for matrices L output by the algorithm M_B. To compute the commutator condition on X, Eve can run M_B on some random inputs r_E to find suitable matrices $f(L)$ and impose the necessary conditions $Xf(L) = f(L)X$ on X. Since these conditions are linear, the number of random inputs r_E that is required before these necessary conditions become sufficient to imply the commutator condition (at least for an overwhelming proportion of runs of the protocol) is very small.

Since all the conditions on X are linear and easy to find, a suitable matrix X can be computed efficiently.

We claim that $k = Xw_B$. To see this, observe that

$$Xw_B = Xf(K)b = f(K)Xb = f(K)w_A = f(K)f(J)b = f(J)f(K)b = k.$$

This means that the adversary can generate the shared key, and the scheme is broken.

2.3 The RU Scheme

We now cryptanalyse a recent key agreement protocol proposed by Romanczuk and Ustimenko [76], which we refer to as the RU scheme. The protocol works as follows.

Let $\mathrm{GL}_n(\mathbb{F}_q)$ denote the group of invertible $n \times n$ matrices over a finite field $\mathbb{F}_q$ of order q, and let $\mathbb{F}_q[x, y]$ denote the polynomial ring over $\mathbb{F}_q$ in two variables x and y. Let $C, D \in \mathrm{GL}_n(\mathbb{F}_q)$ be two commuting matrices and let $d \in \mathbb{F}_q^n$. The matrices C and D and the vector d are made public.

To agree on a shared key, Alice picks a polynomial $f_A(x, y) \in \mathbb{F}_q[x, y]$ and sends $w_A = f_A(C, D)d$ to Bob. Likewise Bob picks a polynomial $f_B(x, y) \in \mathbb{F}_q[x, y]$ and

sends $w_B = f_B(C, D)d$ to Alice. Alice computes $k_A = f_A(C, D)w_B$, Bob computes $k_B = f_B(C, D)w_A$. Since C and D commute, the same is true for $f_A(C, D)$ and $f_B(C, D)$, and so their shared key is the vector $k := k_A = k_B$.

It was not fully specified how the matrices C, D and the polynomials f_A, f_B are generated. However, the following cryptanalysis applies to any method of generation.

A Cryptanalysis

Suppose a passive adversary Eve observes w_A, w_B during transmission, and knows the public quantities C, D and d. Let X be any matrix such that

$$XC = CX, \quad XD = DX, \quad Xd = w_A.$$

Note that such a matrix exists, since $X = f_A(C, D)$ satisfies these conditions. Since the conditions on X are all linear, such a matrix is easily found. Eve can then compute the key as:

$$Xw_B = Xf_B(C, D)d = f_B(C, D)Xd = f_B(C, D)w_A = k.$$

2.4 Stickel's Scheme

In 2004, E. Stickel [86] proposed a key agreement scheme using matrices in a certain subgroup of $\mathrm{GL}_n(\mathbb{F}_q)$. However, M. Sramka demonstrated weaknesses in the scheme [84], and a full cryptanalysis that works for all of $\mathrm{GL}_n(\mathbb{F}_q)$ was provided by V. Shpilrain [80]. Shpilrain suggested that a more secure platform for the scheme is $\mathrm{Mat}_n(R)$, the algebra of all (not necessarily invertible) $n \times n$ matrices defined over some finite ring R. He also suggested a more general method of generating keys, involving polynomials of matrices in $\mathrm{Mat}_n(R)$.

In this section we show how, when $R = \mathbb{F}_q$, Shpilrain's modifications of Stickel's scheme are susceptible to a linear algebra attack. We begin in Subsection 2.4.1 by describing Stickel's scheme and Shpilrain's original attack on the scheme. We

then discuss Shpilrain's modifications of the scheme in Subsection 2.4.2. In Subsection 2.4.3 we offer a cryptanalysis of these modifications when $R = \mathbb{F}_q$. We do this by modifying Shpilrain's original linear algebra attack on the scheme, and run some computer experiments to demonstrate the feasibility of this attack. Our experiments demonstrate that a passive adversary can feasibly compute the shared key for practical parameter sizes. We close in Subsection 2.4.4 with a discussion on working over other finite rings $R \neq \mathbb{F}_q$, and conclude that until a suitable ring R is proposed, the scheme may be considered insecure.

2.4.1 Stickel's Key Agreement Scheme

Stickel's key agreement scheme is as follows. Let $\mathcal{Z}$ denote the centre of $\mathrm{GL}_n(\mathbb{F}_q)$, and let $A, B, W \in \mathrm{GL}_n(\mathbb{F}_q)$ be fixed public matrices such that $AB \neq BA$. To agree on a shared key, Alice and Bob perform the following:

1. Alice randomly picks $l, m \in \mathbb{N}$, $C_1 \in \mathcal{Z}$ and sends $U = C_1 A^l W B^m$ to Bob.
2. Bob randomly picks $r, s \in \mathbb{N}$, $C_2 \in \mathcal{Z}$ and sends $V = C_2 A^r W B^s$ to Alice.
3. Alice computes $K_A = C_1 A^l V B^m$. Bob computes $K_B = C_2 A^r U B^s$.

Alice and Bob have agreed on a common key K, for

$$K_A = K_B = C_1 C_2 A^{l+r} W B^{m+s}.$$

A simpler key agreement scheme with $C_1 = C_2 = 1$ was also proposed by Stickel, but we will consider the more general version. As a secure parameter for the scheme, Stickel proposed $n = 31$. The parameter q was left unspecified, but Shpilrain inferred in his paper that $q = 2^k$ for some $k \in [2, 31]$ (we will also consider $k = 1$). In fact, Stickel proposed that elements A, B, W lie in a particular subgroup of $\mathrm{GL}_n(\mathbb{F}_q)$. But as we will see, Shpilrain's attack applies to all matrices in $\mathrm{GL}_n(\mathbb{F}_q)$. We remark that Sramka's attack [84] on the scheme concentrated on recovering a private exponent l, m, r or s, whereas Shpilrain's more efficient attack works to compute the key without knowledge of any private exponents.

Shpilrain's Attack

Shpilrain noted that for an adversary to compute the key, it suffices upon intercepting the transmitted messages U and V to find invertible matrices X and Y such that

$$XA = AX, \;\; YB = BY, \;\; U = XWY. \tag{2.4.1}$$

One can then compute the key as

$$XVY = XC_2A^rWB^sY = C_2A^rXWYB^s = C_2A^rUB^s = K_B.$$

The matrix equations (2.4.1) define a *nonlinear* system of $3n^2$ equations in $2n^2$ unknowns. But since X is invertible, and $XA = AX$ if and only if $X^{-1}A = AX^{-1}$, one can instead set $X_1 = X^{-1}$ and solve the following overdetermined system of *linear* equations for X_1 and Y:

$$\begin{aligned} X_1A &= AX_1 \\ YB &= BY \\ WY &= X_1U. \end{aligned} \tag{2.4.2}$$

The key can then be computed as $XVY = K_B$. A solution to (2.4.2) is guaranteed, since the equations are always satisfied by $X_1 = (C_1A^l)^{-1}$ and $Y = B^m$, and may be efficiently found via Gaussian elimination.

2.4.2 Shpilrain's Modifications

Shpilrain suggested a couple of modifications to Stickel's scheme to avoid his linear algebra attack. We describe these modifications now, and name them the *power version* and the *polynomial version* of Stickel's scheme.

Power Version

Shpilrain first suggested that the public elements A, B, W be noninvertible matrices in $\mathrm{Mat}(n, R)$, for some finite ring R. The ring R was not specified, but we will consider the case $R = \mathbb{F}_q$. The rest of the protocol remains unchanged. As a valid

attack on the scheme, one can still attempt to find matrices $X, Y \in \mathrm{Mat}_n(\mathbb{F}_q)$ satisfying (2.4.1) by solving a system of nonlinear equations. However, Shpilrain's trick of transforming this into a system of linear equations no longer works in general. That is, if one restricts X to be invertible with inverse $X^{-1} = X_1$ and tries to solve (2.4.2) (note that in this case Y need not be invertible), one typically finds no solutions.

Polynomial Version

Stickel's scheme uses the fact that all powers A^l of a matrix A commute. The same is true of all polynomials $\sum c_i A^i$ in A, for $c_i \in R$, where we now assume R is a finite commutative ring. Again, we consider the case $R = \mathbb{F}_q$. This suggests a generalisation of the power version of Stickel's scheme, whereby Alice and Bob choose polynomials (instead of powers) of the public noninvertible matrices A and B. Shpilrain suggested that the polynomials chosen by Alice and Bob have zero constant term, to ensure the resulting matrices are noninvertible (and hence his linear algebra attack fails).

Note that for any $k \geq n$ one can express A^k as $A^k = \sum_{i=0}^{n-1} c_i A^i$, since by the Cayley–Hamilton theorem, A satisfies its own characteristic equation (see e.g. [48], Page 86). Thus it suffices to consider polynomials of degree at most $n-1$. To fully specify the scheme we therefore assume polynomials are picked uniformly at random from the set of all polynomials of degree at most $n-1$ with zero constant term. In detail, we have the following scheme.

Let $\mathcal{Z}$ denote the centre of $\mathrm{Mat}_n(\mathbb{F}_q)$, and let $A, B, W \in \mathrm{Mat}_n(\mathbb{F}_q)$ be fixed public noninvertible matrices such that $AB \neq BA$. To agree on a shared key, Alice and Bob perform the following:

1. Alice randomly picks a matrix $C_1 \in \mathcal{Z}$ and polynomials $l(x), m(x) \in \mathbb{F}_q[x]$ of degree at most $n-1$ with zero constant term. She sends $U = C_1 l(A) W m(B)$ to Bob.

2. Bob randomly picks a matrix $C_2 \in \mathcal{Z}$ and polynomials $r(x), s(x) \in \mathbb{F}_q[x]$ of

degree at most $n-1$ with zero constant term. He sends $V = C_2 r(A) W s(B)$ to Alice.

3. Alice computes $K_A = C_1 l(A) V m(B)$. Bob computes $K_B = C_2 r(A) U s(B)$.

Alice and Bob have agreed on a common key K, for

$$K_A = K_B = C_1 C_2 l(A) r(A) W m(B) s(B).$$

2.4.3 A Cryptanalysis

We now offer a cryptanalysis of the power variant and polynomial variant of Stickel's scheme. Note that the power version is a special case of the polynomial version with $l(x) = x^l, m(x) = x^m, r(x) = x^r, s(x) = x^s$. We thus concentrate on cryptanalysing the polynomial version.

There are two main stages of our attack. The first stage involves guessing a polynomial $q(x)$. We then use this polynomial to construct a system of linear equations similar to (2.4.2), from which the key may be derived.

Let $m_A(x)$ denote the minimum polynomial of A. The polynomial $q(x)$ of interest to an adversary is:

$$q(x) := \gcd(l(x), m_A(x)).$$

We will see later how $q(x)$ is typically easy to guess for an adversary. We first show how knowledge of $q(x)$ is sufficient to perform a linear algebra attack to recover the key.

Once $q(x)$ is Found

Suppose an adversary has knowledge of $q(x)$. We begin with a lemma.

Lemma 2.3. *Let $A \in \mathrm{Mat}_n(\mathbb{F}_q)$ and let $p(x) \in \mathbb{F}_q[x]$. If $\gcd(p(x), m_A(x)) = 1$ then $p(A)$ is invertible.*

PROOF. Since $gcd(p(x), m_A(x)) = 1$, there exist polynomials $r(x), s(x) \in \mathbb{F}_q[x]$ such that $p(x)r(x) + m_A(x)s(x) = 1$. Since $m_A(A) = 0$, we have $p(A)r(A) = I$. Thus $p(A)$ is invertible. This completes the proof.

Suppose an adversary has correctly guessed $q(x) := \gcd(l(x), m_A(x))$. Note that we can write $l(x)$ as $l(x) = p(x)q(x)$, where $\gcd(p(x), m_A(x)) = 1$, for some polynomial $p(x) \in \mathbb{F}_q[x]$. By the lemma, $p(A)$ is invertible. Now it suffices for an adversary to find matrices $X, Y \in \mathrm{Mat}_n(\mathbb{F}_q)$ such that:

$$XA = AX \tag{2.4.3}$$

$$YB = BY \tag{2.4.4}$$

$$q(A)WY = XU \tag{2.4.5}$$

$$X \text{ is invertible.} \tag{2.4.6}$$

A solution to (2.4.3)–(2.4.6) is now guaranteed, since the equations are always satisfied by $X = (C_1 p(A))^{-1}$ and $Y = m(B)$. Note that an adversary does not need to know $l(x)$ or $p(x)$ to perform this attack. The key may be computed as:

$$\begin{aligned} X^{-1}q(A)VY &= X^{-1}q(A)C_2 r(A)Ws(B)Y \\ &= C_2 r(A)X^{-1}q(A)WYs(B) \\ &= C_2 r(A)Us(B) \\ &= K. \end{aligned}$$

One way to find a solution to (2.4.3)–(2.4.6) is to first solve the system of linear equations defined by (2.4.3)–(2.4.5). Performing Gaussian elimination will yield a number of free variables, from which one can search, by running over the free variables, for an invertible matrix X.

To investigate this attack in more depth, we performed computer experiments for some reasonable parameters $(n, q) = (80, 2), (40, 4), (20, 8)$. For $(n, q) = (80, 2)$ we used Sage [78], version 4.3, and for the remaining parameters we used (for efficiency reasons) Magma [59], version 2.16. For completeness, we ran our experiments on both variants of the scheme; experiments were performed on a 2.6 GHz dual core Opteron 885 Sun server. Over 100 trials, we recorded the average/maximum number of free variables found by solving the linear system of equations defined by (2.4.3)–(2.4.5). We also recorded the average/maximum time taken to find

an invertible matrix X, by naive exhaustive search over the free variables. (This is clearly not optimal, but sufficient for our purposes.) The results are displayed in Table 2.1.

			#free var.		time (mins)	
version	n	q	avg	max	avg	max
power	80	2	3	13	407	455
	40	4	2	5	9.97	10.18
	20	8	2	4	0.41	0.43
poly	80	2	4	13	403	434
	40	4	2	7	9.84	10.24
	20	8	2	6	0.41	0.44

Table 2.1: Finding an Invertible Matrix X.

It is clear from the results in Table 2.1 that once an adversary has correctly guessed $q(x)$, it will not take long (for any reasonable parameters) to derive the key.

Finding $q(x)$

An adversary needs to know $q(x)$ to perform the above linear algebra attack. We now show that $q(x)$ is typically easy to guess.

First suppose the power version of the scheme is used. Then $q(x) = x^i$ for some i, $1 \leq i \leq n$. There are thus only n choices for $q(x)$. (If $q(x) = x^n$ then the matrix A is nilpotent: $A^k = 0$ for all $k \geq n$, and Alice and Bob will agree on the key $K = 0$.) A good cryptanalytic strategy is as follows. First guess that $q(x)$ equals x and attempt to solve (2.4.3)–(2.4.6). If we find an invertible matrix X, we are done. Else, we have made an incorrect guess for $q(x)$; instead, keep guessing $q(x) = x^i$, for $i = 2, 3, \ldots$ and attempt to solve (2.4.3)–(2.4.6), until we find an invertible matrix. In experiments, $q(x)$ has low degree (typically 1 or 2), and over 10,000 trials, the maximum degree of $q(x)$ found was 13 for $(n, q) = (80, 2)$, 8 for $(n, q) = (40, 4)$, and 5 for $(n, q) = (20, 8)$. Thus after only a small number of guesses, an adversary can discover $q(x)$ and compute the key.

Now suppose the polynomial version of the scheme is used. Since x divides $l(x)$, and A is noninvertible, an adversary knows that x divides $q(x)$. Over 10,000 trials, we recorded the most frequent values for $q(x)$, for the same parameters as before. The results are displayed in Table 2.2 (a is a generator of $\mathbb{F}_q^*$).

$(n,q)=(80,2)$		$(n,q)=(40,4)$		$(n,q)=(20,8)$	
polynomial	freq.	polynomial	freq.	polynomial	freq.
x	5548	x	7619	x	8748
$x(x+1)$	2280	$x(x+1)$	622	$x(x+a)$	171
$x(x^2+1)$	598	$x(x+a)$	600	$x(x+a^5)$	166
$x(x^2+x+1)$	479	$x(x+a^2)$	599	$x(x+a^3)$	157
$x(x^3+1)$	179	$x(x^2+x+1)$	63	$x(x+a^4)$	157
$x(x^3+x^2+x+1)$	161	$x(x^2+ax+a^2)$	49	$x(x+1)$	153
$x(x^3+x+1)$	114	$x(x^2+a^2x+a)$	43	$x(x+a^2)$	143
$x(x^3+x^2+1)$	100	$x(x^2+a^2)$	39	$x(x+a^6)$	141
Total:	9459	Total:	9634	Total:	9836

Table 2.2: Statistics of $q(x)$.

Experiments indicate that after 8 guesses an adversary has a good chance of discovering $q(x)$ (94.59% for $n = 80$, the worst case scenario in our experiments). In fact, the situation is better than this. Besides $q(x)$, there are 3 other polynomials of interest to an adversary, namely $\gcd(m(x), m_B(x))$, $\gcd(r(x), m_A(x))$, and $\gcd(s(x), m_B(x))$. Since $m(x), r(x), s(x)$ are picked independently, they are guessed just as $l(x)$ is guessed. From knowledge of any one of these polynomials, one can set up the obvious system of linear equations to derive the key. Moreover, in our experiments the maximum degree of $q(x)$ was 15 for $(n,q) = (80,2)$, 7 for $(n,q) = (40,4)$, and 5 for $(n,q) = (20,8)$. Since the matrix A is public, an adversary can compute its factorised characteristic polynomial, and run over combinations of its irreducible factors to find $q(x)$.

2.4.4 Discussion

The key to avoiding Shpilrain's attack is to ensure that $\gcd(l(x), m_A(x))$ is nontrivial. Thus in the polynomial version, instead of picking polynomials divisible by x at random, one could pick polynomials at random until one has a nontrivial factor

in common with $m_A(x)$. However, typically (as one might expect) this nontrivial factor will be small and hence feasibly guessed.

Instead of working over a finite field, Shpilrain suggested working over some finite ring R. Two obvious candidates for R are $R_1 = \mathbb{Z}_n$ where $n = pq$ is a product of two primes, and $R_2 = \mathbb{Z}_{2^n}$. For R_1, assuming the factorisation of n is known, we may compute $K \bmod p$ and $K \bmod q$ and use the Chinese remainder theorem to compute K. Likewise the Chinese remainder theorem applies if one considers the polynomial ring $\mathbb{F}_q[x]/(f(x))$. For R_2 we may compute $K \bmod 2$, followed by $K \bmod 2^2$, and so on all the way up to $K = K \bmod 2^n$, and derive, at the very least, partial information about the key.

We conclude that until a suitable ring R is proposed, the variant schemes may be considered insecure.

Chapter 3

Tillich–Zémor Hash Function

A *hash function* $h : A^* \to G$ is a function from the set of all strings over some finite alphabet A to a finite set G. Hash function design is an important and active area of research; applications include digital signatures, message authentication codes, and efficient password storage. There are three standard notions of security for a hash function, namely *preimage resistance:* given $g \in G$, find a string s such that $h(s) = g$, *second preimage resistance:* given a string s, find a string t such that $h(s) = h(t)$, and *collision resistance:* find two strings s and t such that $h(s) = h(t)$.

In 1994, Tillich and Zémor [92] designed the following hash function. Let G be the group $\mathrm{SL}_2(\mathbb{F}_{2^n})$ of 2×2 matrices of determinant 1 with entries in the finite field $\mathbb{F}_{2^n}$. Let α be an element of $\mathbb{F}_{2^n}$ not contained in any proper subfield, and define $S = \{s_0, s_1\}$, where

$$s_0 = \begin{pmatrix} \alpha & 1 \\ 1 & 0 \end{pmatrix}, \quad s_1 = \begin{pmatrix} \alpha & \alpha+1 \\ 1 & 1 \end{pmatrix}.$$

A binary string $b_1 b_2 \dots b_m$ is hashed to the matrix $s_{b_1} s_{b_2} \cdots s_{b_m} \in G$.

Note that it is easy to find a collision for this hash function. For example, the empty string and the string consisting of $|\mathrm{SL}_2(\mathbb{F}_{2^n})|$ ones both hash to the identity element. However, the latter string is impractically long (it cannot be written out in polynomial time), so one is interested in shorter collisions.

The Tillich–Zémor hash function is an example of a more general class of hash functions based on concepts from group theory, originating in work by Tillich and Zémor [91] and Zémor [97, 98]. Let A be an alphabet and let $G = \langle S \rangle$ be a finite group, with $|A| = |S|$. Let $\phi : A \to S$ be a set bijection and define a function $h : A^* \to G$ by sending a string $a_1 \dots a_m$ to the group element $\phi(a_1) \cdots \phi(a_m)$. For

applications, one is interested in the binary alphabet, so we assume $S = \{s_0, s_1\}$.

Since its design, several partial cryptanalytic results for the Tillich–Zémor hash function have been discovered. This culminated in the recent work of Grassl, Ilić, Magliveras and Steinwandt [40], who devised an efficient method to find short collisions of length $2n + 2$ by constructing certain palindromic bitstings. Based on this attack, an efficient preimage-finding algorithm was then constructed by Petit and Quisquater [70].

The cryptanalysis of Grassl et al. crucially depends on the characteristic of the underlying field and the specific generators s_0, s_1. It is natural to explore the following questions. Can we produce a secure hash function, that in particular avoids the attack of Grassl et al. by replacing the underlying finite field $\mathbb{F}_{2^n}$ by another finite field $\mathbb{F}_q$? We say any such hash function is of *TZ-type*. Can we produce a secure hash function by changing our generators s_0 and s_1, with or without changing the underlying field? We say any such hash function is of SL_2*-type*. Can we produce a secure hash function by changing the group G? We say that such hash functions are *group-based*. To answer these questions, we need to understand how far the Grassl et al. attack generalises to these settings. We also need to understand *generic collision-finding algorithms*: attacks that can be applied to all group-based hash functions.

This chapter contains the following results. We first present generic collision-finding algorithms for SL_2-type hash functions. The algorithms work by successively hashing into smaller and smaller subgroups, until one lands in an abelian subgroup (which could be the trivial group). Once inside an abelian group, collisions are easily constructed. The algorithms provide collisions of length roughly $2(\log_2 q)^2$ in time and space complexity $\mathcal{O}(\sqrt{q})$. Although (to the author's knowledge) the algorithms have not appeared in print before, they were clearly known to the designers of the hash function, since the choice of platform group renders the approach to finding collisions infeasible. Moreover, the algorithms are somewhat naive, employing a general meet-in-the-middle approach. Having said this, we use commutator calculus to find a simplification of the algorithms when the underlying

field has characteristic 2. This method finds collisions of length roughly $2n(n+1)$, but still runs in exponential time.

Next we demonstrate that the attack by Grassl et al. does not extend in a straightforward manner to TZ-type hash functions over fields of odd characteristic. We then provide a method for finding short collisions of length roughly $4\log_2 q$ for TZ-type hash functions that is independent of the characteristic of the underlying field. However, the algorithm has the same complexity as generic attacks: $\mathcal{O}(\sqrt{q})$.

The chapter is organised as follows. In Subsection 3.1 we discuss previous work on the Tillich–Zémor hash function. In Subsection 3.2 we outline our collision-finding algorithms. In Subsection 3.3 we discuss Grassl et al.'s attack and provide a method for finding short collisions for hash functions of TZ-type. We end in Subsection 3.4 with a summary of collision-finding algorithms.

3.1 Previous Work

There has been a significant amount of research on group-based hash functions. We provide a brief survey of the literature.

Group-based hash functions date back to work by Tillich and Zémor [91] and Zémor [97, 98]. There is good motivation for choosing $G = \mathrm{SL}_2(\mathbb{F}_{2^n})$ as a platform group. In general, one may view the process of computing the hash function as taking a walk in the Cayley graph $\mathcal{C}_S(G)$ of G with respect to the generating set $S = \{s_0, s_1\}$. Hence security notions take on a graph theoretical interpretation. For example, finding a collision in the hash function is equivalent to finding a cycle in $\mathcal{C}_S(G)$. Thus the security of group-based hash functions depends on (but is not obviously equivalent to) the hardness of the *representation problem:* given a finite group G with generating set S, find a short word w on S such that $w = 1$ in G.

The Cayley graph $\mathcal{C}_S(G)$ displays some nice security properties. For example, one can show that the distribution of hashes of length l tends to uniformity as l tends to infinity. Also, the (directed) girth of $\mathcal{C}_S(G)$ is larger than n, which protects against small modifications of a bitstring. Another reason for choosing

$G = \mathrm{SL}_2(\mathbb{F}_{2^n})$ is efficiency: computing a hash amounts to fast arithmetic in $\mathbb{F}_{2^n}$. A detailed discussion is given in [92].

Prior to the choice of $G = \mathrm{SL}_2(\mathbb{F}_{2^n})$, Tillich and Zémor considered the platform group $SL_2(\mathbb{Z}_p)$ with generators

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

However, the hash function was shown to be insecure due to a so called *density attack.* The attack works by efficiently finding a matrix $U \in SL_2(\mathbb{Z})$ that reduces to the identity matrix modulo p, and can also be expressed as a short word on s_0, s_1; details are given in [98]. The choice of platform group $G = \mathrm{SL}_2(\mathbb{F}_{2^n})$ avoids this attack. Charles and Pieprzyk [21] derived conditions on the polynomial defining $\mathbb{F}_{2^n}$ which lead to short collisions, but Abdukhalinov and Kim [1] showed that the conditions hold with very low probability if a randomly chosen irreducible polynomial is selected to define the field. Abdukhalinov and Kim also introduced a generalisation to arbitrary finite fields (what we are calling TZ-type hash functions) by letting $G = \mathrm{SL}_2(\mathbb{F}_q)$ and using the generators

$$t_0 = \begin{pmatrix} \alpha & -1 \\ 1 & 0 \end{pmatrix}, \quad t_1 = \begin{pmatrix} \alpha + 1 & -1 \\ 1 & 0 \end{pmatrix}.$$

Moreover, they demonstrated that this hash function displays similar security properties to the original Tillich–Zémor construction. Geiselmann [33] considered a discrete logarithm approach to finding collisions, but his method produces impractically long collisions. Steinwandt et al. [85] considered a particular subgroup attack, but this may be avoided by a suitable parameter choice. Petit et al. [73] constructed a generic collision-finding algorithm (and also considered finding preimages). However, their algorithm runs in exponential time. Thus prior to the attack of Grassl et al. (which we discuss later) and Petit and Quisquater's follow up work to find preimages [70], the hash function remained essentially unbroken.

We mention briefly a couple of notable variants of the Tillich–Zémor hash function. They work similarly to the Tillich–Zémor hash function, and are based on

taking walks in certain expander graphs. Charles, Goren and Lauter [20] considered basing a hash function on LPS expander graphs, and Petit, Lauter and Quisquater [71] investigated Morgenstern expander graphs as a suitable platform. However, the underlying representation problem turned out not be as difficult as first thought [72, 93].

3.2 Collision-Finding Algorithms

We now describe our generic collision-finding algorithms for SL_2-type and group-based hash functions.

3.2.1 Collisions for SL_2-Type Hash Functions

Let $G = \mathrm{SL}_2(\mathbb{F}_q)$, and let h be the associated hash function (using any generators). We present two algorithms, Algorithm A and Algorithm B, for finding collisions for SL_2-type hash functions that work for any parameter choice. They do not depend on the generators used or the characteristic of the underlying field. In Algorithm A we fix a particular subgroup chain of G and use a meet in the middle approach to hash into successive subgroups. Collisions may be formed once we hash into an abelian subgroup of G. This approach may be seen as a straightforward extension of Petit et al.'s generic attack [73], which we first describe. In Algorithm B we do not fix a subgroup chain ahead of time.

Petit et al. attack. The most successful completely generic attack to date is that of Petit et al. [73]: a preimage to the identity is found as follows.

1. Find N strings m_i such that $(1\ 0)h(m_i) = \lambda_i(1\ 0)$.
2. Find $\{e_i\}$ such that $\prod \lambda_i^{e_i} = 1$ and $\sum |e_i|$ is not too large.
3. Let m_i' be the concatenation of m_i with itself e_i times. Construct

$$d = m_1' || \cdots || m_N'.$$

4. Let $d' = d||d$. Then $h(d') = 1$.

We note that although Petit et al. were considering the case $p = 2$, one trivially extends this attack to arbitrary characteristic p by concatenating d with itself p times in step 4 above (and hence their attack is completely generic). How good is this approach? Step 1 is achieved via a meet in the middle approach with time complexity $\mathcal{O}(\sqrt{q})$. We can approximate the length of the N strings to be roughly $\log_2 q$. A solution to step 2 is provided by fixing a generator g for $\mathbb{F}_q^*$ and solving N discrete logarithms $\lambda_i = g^{\alpha_i}$. One then finds a suitable set $\{e_i\}$ by solving $\sum \alpha_i e_i \equiv 0 \bmod q-1$, using the LLL algorithm. Taking $N = n$, one expects to find collisions of length roughly $2np(\log_2 q)$.

Algorithm A. Consider the subgroup chain $1 < K < H < G$, where

$$H = \left\{ \begin{pmatrix} a & b \\ 0 & a^{-1} \end{pmatrix} : a, b \in \mathbb{F}_q, a \neq 0 \right\}, \quad K = \left\{ \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix} : b \in \mathbb{F}_q \right\}.$$

The first step of our algorithm finds two strings whose hash values lie in the subgroup H. Note that the index of H in G is $q + 1$. We use a meet in the middle approach as follows:

- Generate and store approximately $\sqrt{q}$ strings v and distinct cosets $h(v)^{-1}H$.
- Generate strings u until $h(u)H$ equals some stored value $h(v)^{-1}H$.
- Return vu.

We repeat this process to obtain two different strings x and y whose hash value lies in H.

The next step is to hash into the subgroup K. Note that K has index $q - 1$ in H. We use $h(x)$ and $h(y)$ as generators of H, and repeat the above process to find two strings s and t that hash into K. (Note that if $h(x)$ and $h(y)$ do not generate H, then they generate a proper subgroup H' of H. One can then apply the same procedure to H'.) A collision is given by $h(st) = h(ts)$, since K is an abelian subgroup of G.

Heuristically, one expects x and y to have length approximately $\log_2 q$. Hence one expects to find collisions of length roughly $2(\log_2 q)^2$. The time complexity of this algorithm is $\mathcal{O}(\sqrt{q})$. In terms of memory, the algorithm has space complexity $\mathcal{O}(\sqrt{q})$ provided one has an efficient encoding of the cosets. An efficient encoding of the cosets of H in G is given by the bijection $G : H \to \mathbb{F}_q \cup \{\infty\}$, where

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} H \mapsto \begin{cases} ac^{-1} \text{ if } c \neq 0 \\ \infty \quad \text{if } c = 0. \end{cases}$$

Similarly, the cosets of K in H may be encoded via the bijection

$$\begin{array}{ccc} H : K & \to & \mathbb{F}_q^* \\ \begin{pmatrix} a & b \\ 0 & a^{-1} \end{pmatrix} K & \mapsto & a. \end{array}$$

Algorithm B. Instead of fixing a subgroup chain for G as in Algorithm A, we aim to hash into any conjugate subgroup of H followed by any conjugate subgroup of K (note that any conjugate subgroup of K is abelian in G). There are many conjugate subgroups of H and K in G, so one might expect this approach to find shorter collisions than Algorithm A.

The first step is to find two elements in a conjugate subgroup of H. An element $g \in G$ lies in a conjugate subgroup of H if and only if it has an eigenvalue in $\mathbb{F}_q$. It is straightforward to check this condition (e.g. one could check for reducibility of the characteristic polynomial of g). A birthday attack works as follows:

- Generate and store approximately $\sqrt{q}$ strings v subject to the condition that $h(v)$ has an eigenvalue $\lambda \in \mathbb{F}_q$. Store the eigenvector associated with λ (as a projective point).
- Generate strings u until an eigenvector of $h(u)$ equals an eigenvector of $h(v)$ (up to scalar multiples), for some stored v.
- Return u, v.

Thus $h(u)$ and $h(v)$ lie in the same conjugate subgroup of H.

The next step is to find two strings whose hash value lies in a conjugate subgroup of K. Suppose $h(u)$ and $h(v)$ have an eigenvalue λ_1 and λ_2 respectively (we assume, as is likely, that $h(u)$ and $h(v)$ do not have the same eigenvalues). To hash into a conjugate subgroup of K, we seek a short word w on $\{\lambda_1, \lambda_2\}$ such that $w = 1$ in $\mathbb{F}_q^*$. We again employ a meet in the middle attack to solve this:

- Generate approximately $\sqrt{q}$ words x on $\{\lambda_1, \lambda_2\}$. Store x^{-1}.
- Generate words y on $\{\lambda_1, \lambda_2\}$ until $y = x^{-1}$ for some stored x^{-1}.
- Return xy.

Now we have an element in a conjugate subgroup of K. It is easy to generate more elements in the same subgroup since we are concerned only with the combination of λ_i, and not their arrangement in the product of x and y above. One may then form collisions as in Algorithm A, since we are inside an abelian subgroup.

We note that Algorithm B has the same complexity as Algorithm A.

Remark. We remark that the m_i satisfying the condition in step 1 of Petit et al.'s attack are precisely the lower triangular matrices, and if one considers right eigenvectors instead, one gets upper triangular matrices, i.e. H. Also note that $h(d)$ in step 3 lies in an abelian subgroup K' of G. Since the order of the concatenation of the m_i' is irrelevant, one easily finds a different string $d' \neq d$ such that $h(d') \in K'$. A collision is then given by $h(d)h(d') = h(d')h(d)$.

3.2.2 A Simplification in Characteristic 2

In characteristic 2, there is a simplification to our algorithms to produce collisions. We provide an exposition for Algorithm A, but it also applies to Algorithm B and Petit et al.'s algorithm.

Suppose we have found 2 strings (via the meet in the middle approach) whose hash values u and v lie in H. We can construct a collision based on certain combinations of uv and vu as follows. Let $k = u^{-1}v^{-1}uv$ be the commutator of u and

v. Then $k \in K$. Treating K as the vector space $\mathbb{F}_2^n$, define a linear map ϕ on K by $\phi(k') = (vu)^{-1}k'(vu)$. Then a collision is determined by the minimum polynomial $m_\phi(x)$ of ϕ. For example, suppose $m_\phi(x) = x^2 + x + 1$. Then a collision is given by

$$\begin{aligned} uvuvuv &= vukvukvuk \\ &= vuvu\phi(k)vu\phi(k)k \\ &= vuvuvu\phi^2(k)\phi(k)k \\ &= vuvuvu. \end{aligned}$$

In general, since $m_\phi(x)$ has degree at most n, we expect to find collisions of length $2(n+1)\log_2 q = 2n(n+1)$. This method only seems to work in characteristic 2, since there is no obvious way to deal with non-binary coefficients of $m_\phi(x)$.

3.2.3 Collisions for Group-Based Hash Functions

Algorithm A extends in a straigtforward manner to group-based hash functions.

Let $1 = G_0 < G_1 < \ldots < G_d = G$ be a chain of subgroups of G, and let k_i be the index of G_{i-1} in G_i. The goal is to find a preimage of the identity element of G, from which collisions may be formed. The meet in the middle approach used in Algorithm A is performed iteratively on the subgroup chain down to the identity subgroup (provided one has an efficient encoding of the cosets). One finds a bitstring that hashes to the identity, with expected length approximately

$$\prod_{i=1}^{d} \log_2 k_i.$$

The complexity of this approach is governed by the largest index k_i, so the algorithm has complexity $\mathcal{O}(\sqrt{max\{k_i\}})$.

3.3 Short Collisions for TZ-Type Hash Functions

In this section we outline Grassl et al.'s attack on the Tillich–Zémor hash function and show that it does not extend to TZ-type hash functions in a straightforward manner. We then provide a method for finding short collisions for TZ-type hash functions.

3.3.1 Grassl et al.'s Attack

Before describing the attack we need a couple of definitions. Let V denote the set of finite bitstrings and let $h : V \to G$ be the Tillich–Zémor hash function. The *length* of a string $v = b_1 \dots b_m$ is $|v| := m$. The *reversal* of a string $v = b_1 \dots b_m$ is defined as $v^r := b_m \dots b_1$. A string v is a *palindrome* if $v = v^r$.

The attack works as follows. Firstly, the generators s_0 and s_1 are conjugated by s_0. So we have new generators $c_0 = s_0$ and $c_1 = s_0^{-1} s_1 s_0$, where

$$c_0 = \begin{pmatrix} \alpha & 1 \\ 1 & 0 \end{pmatrix}, \quad c_1 = \begin{pmatrix} \alpha + 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Since conjugation preserves collisions (that is, any collision found based on the generators c_0, c_1 is also a collision based on the generators s_0, s_1), it suffices to find collisions using the symmetric generators c_0 and c_1. The next step to their attack is to consider the lifting map $H : V \to \mathrm{SL}_2(\mathbb{F}_2[x])$, which is defined similarly to h except that the matrix images lie over the polynomial ring $\mathbb{F}_2[x]$ instead of $\mathbb{F}_q$. They consider the map

$$\begin{aligned} \rho : \quad V &\to \mathbb{F}_2[x]^{2\times 2} \\ v &\mapsto \mathrm{H(0v0)} + \mathrm{H(1v1)}, \end{aligned}$$

and show that if v is an even length palindrome then $\rho(v) = \begin{pmatrix} a^2 & a^2 \\ a^2 & 0 \end{pmatrix}$ for some polynomial a. The task of finding collisions is now reduced to finding a such that $a^2 \equiv 0$ modulo the polynomial defining $\mathbb{F}_q$, for then $h(0v0) = h(1v1)$ is a collision. A very elegant solution to this problem is found by considering maximal length chains in the Euclidean algorithm, from which short collisions of length $2n + 2$ are constructed.

We want to know to what extent Grassl et al.'s attack generalises. The following lemma shows that their attack does not extend to TZ-type hash functions in a straightforward manner.

For the remainder of this section, we let $G = \mathrm{SL}_2(\mathbb{F}_q)$ and work with the natural generators t_0 and t_1 defined in Section 2 (note that in characteristic 2 these

are precisely the generators used by Grassl et al.). We let h be the associated hash function.

Lemma 3.1. *Let $v \in V$ be a palindrome and let $h(v) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Then*

(i) $h(v) = \begin{pmatrix} a & b \\ -b & d \end{pmatrix}$.

(ii) If $\mathbb{F}_q$ does not have characteristic 2 then $h(0v0) \neq h(1v1)$.

PROOF. For part (i), we proceed by induction on the length of $v \in V$. If $|v| = 1$ then $h(v)$ is t_0 or t_1, and by inspection both t_0 and t_1 have the desired form. Assume the result holds for odd length bitstrings, so $h(v) = \begin{pmatrix} a & b \\ -b & d \end{pmatrix}$ and $|v|$ is odd. Then direct calculation gives

$$h(\beta v \beta) = \begin{pmatrix} a(\alpha+\beta)^2 + 2b(\alpha+\beta) - d & -(a\alpha + a\beta + b) \\ a\alpha + a\beta + b & -a \end{pmatrix},$$

which has the desired form. The case for even length bitstrings is identical. Hence by induction part (i) of the lemma holds.

For part (ii), since v is a palindrome we have that $h(v) = \begin{pmatrix} a & b \\ -b & d \end{pmatrix}$, by part (i). Note that

$$h(0v0) - h(1v1) = \begin{pmatrix} -2a\alpha - a - 2b & a \\ -a & 0 \end{pmatrix}.$$

Hence for $h(0v0) = h(1v1)$ to hold in odd characteristic, we require $a = 0$. This in turn implies $b = 0$. But this contradicts the fact that $h(v) \in \mathrm{SL}_2(\mathbb{F}_q)$.

3.3.2 Short Collisions for TZ-Type Hash Functions

We now illustrate a way to find short collisions for TZ-type hash functions. We begin with a lemma.

Lemma 3.2. *Let $v \in V$ and let $h(v) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Then*

(i) $h(v^r) = \begin{pmatrix} a & -c \\ -b & d \end{pmatrix}$.

(ii) $h(v) - h(v^r) = \begin{pmatrix} 0 & b+c \\ b+c & 0 \end{pmatrix}$.

(iii) $h(0vv^r1) - h(1vv^r0) = \begin{pmatrix} 0 & a^2-c^2 \\ a^2-c^2 & 0 \end{pmatrix}$.

(iv) $h(0v^rv1) - h(1v^rv0) = \begin{pmatrix} 0 & a^2-b^2 \\ a^2-b^2 & 0 \end{pmatrix}$.

(v) $h(vv) - h(v^rv^r) = \begin{pmatrix} 0 & (a+d)(b+c) \\ (a+d)(b+c) & 0 \end{pmatrix}$.

PROOF. For part (i), we proceed by induction. If $|v| = 1$ then $h(v)$ is t_0 or t_1, and by inspection both t_0 and t_1 have the desired form. Assume the result holds for $|v| = k$. Then a bitstring of length $k+1$ may be written as $v\beta$. By the inductive step, v has the desired form, i.e. $h(v) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and $h(v^r) = \begin{pmatrix} a & -c \\ -b & d \end{pmatrix}$. Direct calculation gives

$$h(v\beta) = \begin{pmatrix} a(\alpha+\beta)+b & -a \\ c(\alpha+\beta)+d & -c \end{pmatrix}$$

and

$$h(\beta v^r) = \begin{pmatrix} a(\alpha+\beta)+b & -c(\alpha+\beta)-d \\ a & -c \end{pmatrix},$$

and we see that the matrices have the desired form. Hence by induction part (i) of the lemma holds. Parts (ii) to (vi) of the lemma may be seen by simple matrix calculations.

The lemma provides good heuristic reasons to expect to find collisions of length of the order $\log_2 q$. By parts (ii) and (v) of Lemma 3.2, to find a collision it suffices to find a non-palindrome v such that either $b+c = 0$ or $a+d = 0$ in $h(v)$. By parts (iii)

and (iv), to find a collision it suffices to find any bitstring $v \in V$ such that $a = \pm b$ or $a = \pm c$ in $h(v) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$.

The following proposition and corollary show that we can find collisions of length roughly $4 \log_2 q$.

Proposition 3.3. *Let $u = b_1 \dots b_r$ be a bitstring that hashes into H, i.e. $h(u) \in H$. Let $v = b_1 b_r b_{r-1} \dots b_2$. Then $h(v) \in H$. Moreover, bitstrings uv and vu hash into K, i.e. $h(uv), h(vu) \in K$.*

PROOF. Let $h(b_1) = \begin{pmatrix} x+\beta & -1 \\ 1 & 0 \end{pmatrix}$ and $h(b_2 \dots b_r) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, where $\beta = b_1$. Then

$$h(u) = h(b_1)h(b_2 \dots b_r) = \begin{pmatrix} (x+\beta)a - c & (x+\beta)b - d \\ a & b \end{pmatrix}.$$

Since $h(u) \in H$ we have that $a = 0$ and hence $-c = b^{-1}$. Thus

$$h(u) = \begin{pmatrix} b^{-1} & (x+\beta)b - d \\ 0 & b \end{pmatrix}.$$

Now,

$$h(v) = h(b_1)h(b_r \dots b_2) = \begin{pmatrix} x+\beta & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} a & -c \\ -b & d \end{pmatrix} \text{ (by lemma 2.3(i)),}$$

$$= \begin{pmatrix} b & (x+\beta)b^{-1} - d \\ 0 & b^{-1} \end{pmatrix} \text{ (since } a = 0 \text{ and } -c = b^{-1}\text{).}$$

Thus $h(v) \in H$. Moreover, one sees that

$$h(uv) = \begin{pmatrix} 1 & (x+\beta)(1+b^{-2}) - 2b^{-1}d) \\ 0 & 1 \end{pmatrix}.$$

$$h(vu) = \begin{pmatrix} 1 & (x+\beta)(1+b^{2}) - 2b^{-1}d) \\ 0 & 1 \end{pmatrix}.$$

Thus $h(uv), h(vu) \in K$.

Corollary 3.4. *Let $u = b_1 \dots b_r$ be such that $h(u) \in H$. Let $v = b_1 b_r \dots b_2$. Then bitstrings $uvvu$ and $vuuv$ hash to the same value.*

PROOF. By Proposition 3.1, $h(uv)$ and $h(vu)$ lie in K. But K is an abelian subgroup of G. Hence $h(uvvu) = h(uv)h(vu) = h(vu)h(uv) = h(vuuv)$.

3.4 Summary

We close with a summary of the various collision-finding algorithms. The algorithms described in Subsection 3.2.1 are completely generic; they work for any generators and any finite field. Although they find collisions of reasonable length, the algorithms have exponential running time. Fixing the natural generators t_0 and t_1, we can find shorter collisions, but again the algorithm (call this Algorithm C) is inefficient, since we cannot efficiently hash into the subgroup H.

We summarise these algorithms in Table 3.1 indicating which algoriths are specific to characteristic 2 or specific to the natural generators. For completeness include the trivial collision-finding algorithm of writing down $|\mathrm{SL}_2(\mathbb{F}_q)|$ ones. Grassl

Algorithm	Exp. Coll. Length	Cxty (t)	gen. spc?	char. spc?
Trivial	$max\{q+1, 2p\}$	$\mathcal{O}(1)$	no	no
Grassl et al.	$2\log_2 q$	$\mathcal{O}(n^3)$	yes	yes
Petit et al.	$2np(\log_2 q)$	$\mathcal{O}(\sqrt{q})$	no	no
A	$2(\log_2 q)^2$	$\mathcal{O}(\sqrt{q})$	no	no
B	$2(\log_2 q)^2$	$\mathcal{O}(\sqrt{q})$	no	no
C	$4\log_2 q$	$\mathcal{O}(\sqrt{q})$	yes	no

Table 3.1: Collision-Finding Algorithms

et al.'s attack is specific to both the natural generators and the characteristic of the underlying field. We do not have much confidence that changing the generators (but staying in characteristic 2) would lead to an increase in security. Certainly such a scheme would be less natural (and probably less efficient) than the original proposal. Moving to odd characteristic naturally avoids Grassl et al.'s attack, as the results in this chapter indicate. But the loss in efficiency may be too much to overcome if the design it to rival more traditional hash functions.

Chapter 4

MST_3 *Public Key Cryptosystem*

Logarithmic signatures (or *group factorisations* as they are also known) find applications in numerous areas of mathematics and computer science, for example coding theory, combinatorics and cryptography [90]. They first appear in the cryptographic literature in connection with Permutation Group Mappings (PGM), a symmetric cipher invented by Magliveras [54]. The ideas behind PGM have inspired several public key cryptosystems based on logarithmic signatures, most notably the family of MST cryptosystems [50, 57].

In this chapter we explore the role of logarithmic signatures in cryptography. Our main result is a cryptanalysis of a public key cryptosystem MST_3, proposed by Lempken, Magliveras, van Trung and Wei [50].

We begin in Section 4.1 with the definition of a logarithmic signature, a general method of construction, and a small example. In Section 4.2 we provide an overview of the use of logarithmic signatures in cryptography, before moving on to discuss constructions for elementary abelian 2-groups in Section 4.3: this is motivated as they occur as part of the private key for MST_3. We provide a cryptanalysis of MST_3 in Section 4.4; our exposition follows [12]. After the publication of [12], Svaba and van Trung [89] published a paper with modifications to MST_3, resulting in a revised version of the cryptosystem. We close the chapter with some comments on this paper in Section 4.5.

4.1 Definitions, Constructions, Examples

Let G be a finite group, $S \subseteq G$ a subset of G and s a positive integer. For all $1 \leq i \leq s$, let $A_i = [\alpha_{i1}, \ldots, \alpha_{ir_i}]$ be a finite sequence of elements of G of length $r_i > 1$, and

let $\alpha = [A_1, \ldots, A_s]$ be the ordered sequence of A_i. We say that α is a *cover* for S if any $g \in S$ can be written as a product $g = g_1 \cdots g_s$, where $g_i = \alpha_{ik_i} \in A_i$. If such a decomposition is unique for every $g \in S$, then α is said to be a *logarithmic signature* for S. The *type* of a cover α is the vector $(r_1, \ldots, r_s)$, the sets A_i are called *blocks*, and the *length* of α is $l(\alpha) := \sum_{i=1}^{s} r_i$.

Given an element $g \in S$ and a cover α of S, obtaining a factorisation $g = \alpha_{1k_1} \cdots \alpha_{sk_s}$ associated with α seems to be a hard problem in general. Indeed, in some situations the problem is a discrete logarithm problem. For example, let G be generated by an element g of large order, and define $A_{i+1} = [1, g^{2^i}]$. Let $S = \{g^a \mid 0 \leq a \leq 2^s\}$. Then the ith bit of the discrete logarithm of $h \in S$ is equal to 1 if and only if $k_i = 2$ in the factorisation $h = \alpha_{1k_1} \cdots \alpha_{sk_s}$.

If a factorisation can be efficiently computed for every $g \in S$, we say that α is a *tame* cover for S; otherwise α is a *wild* cover for S.

Let α be a cover for S of type $(r_1, \ldots, r_s)$, let k_i be integers, $1 \leq k_i \leq r_i$, and let $q = \prod_{i=1}^{s} r_i$. Consider the maps λ_α and θ_α defined by

$$\begin{array}{rccl} \lambda_\alpha : & \mathbb{Z}_{r_1} \times \ldots \times \mathbb{Z}_{r_s} & \longrightarrow & \mathbb{Z}_q \\ & (k_1, \ldots, k_s) & \longmapsto & \sum_{i=1}^{s} (k_i \prod_{j=1}^{i-1} r_j), \end{array}$$

and

$$\begin{array}{rccl} \theta_\alpha : & \mathbb{Z}_{r_1} \times \ldots \times \mathbb{Z}_{r_s} & \longrightarrow & G \\ & (k_1, \ldots, k_s) & \longmapsto & \alpha_{1k_1} \cdots \alpha_{sk_s}. \end{array}$$

Note that λ_α is a bijection, and both λ_α and λ_α^{-1} are efficiently computable. Define the map

$$\begin{array}{rccl} \breve{\alpha} : & \mathbb{Z}_q & \longrightarrow & G \\ & k & \longmapsto & \theta_\alpha(\lambda_\alpha^{-1}(k)). \end{array}$$

Given $g \in S$, computing an s-tuple $(k_1, \ldots, k_s)$ such that $g = \alpha_{1k_1} \cdots \alpha_{sk_s}$ is equivalent to computing an element of $\theta_\alpha^{-1}(g)$. It follows that α is tame if and only if an element of $\breve{\alpha}^{-1}(g)$ can be efficiently computed for every $g \in S$.

Constructing Logarithmic Signatures

A natural way to construct a logarithmic signature for a finite group G is to choose a subgroup chain $1 = G_0 < G_1 < \cdots < G_s = G$ and let A_i be a complete set of (left or right) coset representatives for G_{i-1} in G_i. Clearly $\alpha = [A_1, \ldots, A_s]$ is a logarithmic signature for G. Call such a logarithmic signature a *transversal logarithmic signature*. We can construct new logarithmic signatures for G by applying any sequence of the following operations to α:

- Permute elements within a block.
- Replace a block A_i by a translate $g \cdot A_i$ or $A_i \cdot g$ for some $g \in G$.
- Amalgamate blocks, i.e. replace two blocks A_i and A_{i+1} by the single block $A_i \cdot A_{i+1} := \{gh \mid g \in A_i, h \in A_{i+1}\}$.

Let $B_i = t_{i-1}^{-1} A_i t_i$ for some $t_i \in G$, and let $\beta = [B_1, \ldots, B_s]$. Then β is a logarithmic signature for G. Note that by taking

$$(t_0, t_1, t_2, \ldots, t_s) = (1, \alpha_{11}^{-1}, (\alpha_{11}\alpha_{21})^{-1}, \ldots, (\alpha_{11}\alpha_{21} \cdots \alpha_{s1})^{-1}),$$

we have that the first element in each block of β is the identity (we use this fact to simplify our cryptanalysis of MST_3 in Subsection 4.4.4).

An Example

Let us consider a small example and construct some logarithmic signatures for $G = \mathbb{F}_2^3$, the elementary abelian 2-group of order 8. We may view the group operation as XOR, and represent elements as binary vectors of length 3, so

$$G = \{000, 001, 010, 011, 100, 101, 110, 111\}.$$

Consider the subgroup chain

$$\{000\} < G_1 < G_2 < G,$$

where $G_1 = \{000, 001\}$ and $G_2 = \{000, 001, 010, 011\}$. We construct a transversal logarithmic signature α for G as follows. First consider A_1, a complete set of coset

representatives for G_0 in G_1. Clearly $A_1 = G_1$ (and in general A_1 will always be a group for such a construction). We have some freedom in choosing A_2 and A_3, so let us pick $A_2 = [010, 001], A_3 = [011, 100]$. Then $\alpha = [A_1, A_2, A_3]$ is a transversal logarithmic signature for G of type $(2, 2, 2)$. Suppose we replace A_1 by the translate $A_1' = A_1 \cdot (111) = [111, 110]$. Then $\beta = [A_1', A_2, A_3]$ is a logarithmic signature for G of type $(2, 2, 2)$, but it is not transversal. Suppose we amalgamate blocks A_2 and A_3 of β to a single block $A_2' = [001, 110, 010, 101]$. Then $\gamma = [A_1', A_2']$ is a logarithmic signature for G of type $(2, 4)$.

4.2 Logarithmic Signatures in Cryptography

Logarithmic signatures were first used in cryptography by Magliveras [54] to construct a symmetric cipher known as Permutation Group Mappings (PGM). The idea of the cipher is simple. Let G be a permutation group of degree n and let α, β be transversal logarithmic signatures for G of length polynomial in n. Such logarithmic signatures are tame [56], hence $\breve{\alpha}^{-1}$ and $\breve{\beta}^{-1}$ are efficiently computable. Encryption of a message $m \in \mathbb{Z}_q$ is given by $c = \breve{\beta}^{-1}\breve{\alpha}(m)$ and decryption is given by $\breve{\alpha}^{-1}\breve{\beta}(c) = m$.

Although PGM satisfies some nice algebraic and statistical properties (such as robustness, scalability and a large key space), fast implementation becomes an issue, making it a rather inefficient cipher compared with more traditional block ciphers. An attempt was made to improve PGM by letting the platform group be a 2-group, but again speed remains an issue [17].

However, PGM has inspired much research into designing public key cryptosystems based on logarithmic signatures. Notable proposals include MST_1 and MST_2, invented by Magliveras, Stinson and van Trung [57].

The MST_1 cryptosystem is theoretical in nature, and relies on efficient generation of wild logarithmic signatures α for a permutation group G. The main problem with MST_1 is constructing wild logarithmic signatures that are typically hard to invert. Magliveras et al. [57] had the idea of restricting α to be *totally non-transversal*,

that is no block of α forms a coset of a non-trivial subgroup of G. However, Bohli et al. [15] constructed instances of totally non-transversal logarithmic signatures that are in fact tame. Further weaknesses were pointed out by González Vasco and Steinwandt [38], and it remains unclear how to generate secure instances of MST_1.

Key generation is also a problem for MST_2 [57]. This cryptosystem employs certain covers called *meshes*. A cover $\alpha = [A_1, \ldots, A_s]$ for S is an $[s, r]$-*mesh* if $|A_i| = r$, and, if there are a_g different factorisations for $g \in S$ with respect to α, then the probability distribution $\{a_g/r^s : g \in S\}$ is approximately uniform.

Let G and H be finite groups and let $f : G \to H$ be a surjective homorphism. Let α be an $[s, r]$-mesh for G and let $\beta = f(\alpha)$ be an $[s, r]$-mesh for H. The public key for MST_2 is (G, H, α, β) and the private key is the homomorphism f. Encryption of a message $m \in H$ is given by selecting a random $R \in \mathbb{Z}_{r^s}$ and computing $y_1 = \breve{\alpha}(R), y_2 = \breve{\beta}(R), y_3 = my_2$. The ciphertext is $c = (y_1, y_3)$. Decryption is given by computing $y_2 = f(y_1)$ followed by $y_3 y_2^{-1} = m$.

MST_2 is also theoretical in nature, and suffers from a lack of specification: the groups G and H were not specified, and secure instances of this scheme are not known. A critique of MST_2 is given in [38].

Related public key proposals based on logarithmic signatures have met similar fates. For example, schemes by Birget, Magliveras and Wei [10], and Qu and Vanstone [75] have been cryptanalysed by González Vasco et al. [35] and Blackburn et al. [14] respectively.

Recently, Lempken, Magliveras, van Trung and Wei [50] designed a new public key cryptosystem called MST_3. A practical instance of the cryptosystem was proposed based on Suzuki 2-groups. Partial cryptanalytic results were discovered by Magliveras et al. [58] and González Vasco et al. [36]. We will discuss their work later in Subsection 4.4.3, but we state now that crucial to the security of the scheme is how one generates logarithmic signatures for elementary abelian 2-groups. Thus before discussing MST_3, we review known constructions of logarithmic signatures for elementary abelian 2-groups. This constitutes the next section.

4.3 Elementary Abelian 2-Groups

Let $\mathcal{Z}$ be an elementary abelian 2-group of order 2^m. We identify $\mathcal{Z}$ with $\mathbb{F}_2^m$ and view elements as binary vectors of length m, with XOR acting as group operation. In this section we describe known methods of constucting logarithmic signatures for $\mathcal{Z}$. We aim to be consistent with notation: $\alpha = [A_1, \ldots, A_s]$ denotes a logarithmic signature or cover for a general group G, and $\beta = [B_1, \ldots, B_s]$ denotes a logarithmic signature for an elementary abelian 2-group.

Canonical Logarithmic Signatures

The following construction is by Magliveras, Svaba, van Trung and Zajac [58]. Randomly partition $X = \{1, 2, \ldots, m\}$ into disjoint sets $C_1, C_2, \ldots, C_s$. Define integers d_i by $d_i = |C_i|$, and let $r_i = 2^{d_i}$. Let $B_1, B_2, \ldots, B_s$ be subgroups of $\mathcal{Z}$ where B_i consists of the 2^{d_i} vectors that are all zero except possibly the positions indexed by integers in C_i. Clearly $\beta = [B_1, B_2, \ldots, B_s]$ is a logarithmic signature of type $(r_1, r_2, \ldots, r_s)$. Call such a construction a *canonical logarithmic signature*. Note these logarithmic signatures form a very special class: all blocks B_i are subgroups of $\mathcal{Z}$. Further note that canonical logarithmic signatures are attractive in terms of storage on a computer, as one only need store a minimal generating set for each subgroup. We have the following.

Lemma 4.1. *Let $\beta = [B_1, \ldots, B_s]$ be a canonical logarithmic signatures. Then β is tame.*

PROOF. Let $z = b_{1k_1} b_{2k_2} \ldots b_{sk_s}$ be any element in $\mathcal{Z}$. We give an algorithm to find $k_1, \ldots, k_s$. First scan $B_1, \ldots, B_s$ to find the partition sets $C_1, \ldots, C_s$. To determine k_i set $\overline{z}_i$ to be equal to z but with all bit positions $X \setminus C_i$ of z set to zero. Then $\overline{z}_i \in B_i$ and k_i is given by the position of $\overline{z}_i \in B_i$.

Amalgamated Transversal Logarithmic Signatures

A more general method of construction is as follows. Let

$$1 = \mathcal{Z}_0 < \mathcal{Z}_1 < \cdots < \mathcal{Z}_s = \mathcal{Z}$$

be a subgroup chain of $\mathcal{Z}$. Let B_i be a complete set of coset representatives for $\mathcal{Z}_{i-1}$ in $\mathcal{Z}_i$. Then $\beta = [B_1, \ldots, B_s]$ is a transversal logarithmic signature for $\mathcal{Z}$.

We can now derive new logarithmic signatures from β by performing the following *elementary operations*:

1. Permute elements within a block.
2. Permute the blocks.
3. Replace B_i by a *translate* $B_i \cdot g$ for some $g \in G$.
4. *Amalgamate* blocks, i.e. replace two blocks B_i and B_j by the single block $B_i \cdot B_j := \{gh \mid g \in B_i, h \in B_j\}$.

Clearly applying any sequence of these operations results in a new logarithmic signature for $\mathcal{Z}$. Call such a construction an *Amalgamated Transversal Logarithmic Signature* (ATLS). Note that one obtains a canonical logarithmic signature from an ATLS by taking $B_i = \mathcal{Z}_i/\mathcal{Z}_{i-1}$.

In general, a *periodic point* of a logarithmic signature $\alpha = [A_1, \ldots, A_s]$ is an element $a_{ij} \in A_i$ such that $A_i a_{ij} = A_i$. Clearly if the identity element lies in A_i then it is a periodic point. A logarithmic signature is *nonperiodic* if there are no nonidentity periodic points; otherwise it is *periodic*. We note the following important properties that hold for any ATLS.

Lemma 4.2. *Let $\beta = [B_1, \ldots, B_s]$ be an ATLS. Then β is periodic.*

PROOF. The subgroup $\mathcal{Z}_1$ has been amalgamated into one of the subsets B_i. It is not difficult to see that $\mathcal{Z}_1 \subseteq B_i$ and $B_i\mathcal{Z}_1 = B_i$, and every (nonidentity) element of $\mathcal{Z}_1$ is a periodic point.

Lemma 4.3. *Let $\beta = [B_1, \ldots, B_s]$ be an ATLS. Then β is tame.*

PROOF. We prove the lemma by induction on the order of $\mathcal{Z}$. The lemma is trivial when $\mathcal{Z}$ has order 1. Assume the lemma holds for all elementary abelian 2-groups of order smaller than $|\mathcal{Z}|$.

Without loss of generality, we may assume that $1 \in B_i$ for all i. Let ϵ be a transversal logarithmic signature, and let $\epsilon = \epsilon_0, \epsilon_1, \epsilon_2, \ldots, \epsilon_\ell = \beta$ be the sequence of logarithmic signatures generated when constructing β (so each ϵ_i is obtained from ϵ_{i-1} by applying an elementry operation).

There exists $z = b_{ij} \in \mathcal{Z}$ such that $B_i b_{ij} = B_i$, by Lemma 4.2. Such an element may be found efficiently by checking at most $\sum_i^s |B_i|$ elements. Let $N = \{1, z\}$ be the subgroup generated by z. Define a logarithmic signature $\overline{\beta}$ for $\mathcal{Z}/N$ by setting $\overline{B}_k = B_k N/N$ (where we remove the redundant elements from B_i so that $\overline{B}_i$ has half the size of B_i). Then $\overline{\beta}$ is an ATLS; this is shown by the sequence $\overline{\epsilon} = \overline{\epsilon}_0, \overline{\epsilon}_1, \overline{\epsilon}_2, \ldots, \overline{\epsilon}_\ell = \overline{\beta}$ of logarithmic signatures of $\mathcal{Z}/N$ defined using the same sequence of amalgamation and translation operations as in the construction of β.

Since $\overline{\beta}$ is an ATLS, it is tame by our inductive hypothesis and so we may efficiently determine $\breve{\overline{\beta}}^{-1}(xN)$ for any $x \in \mathcal{Z}$. We may efficiently invert $\breve{\beta}$ as there are only two possibilities for $\breve{\beta}^{-1}(x)$ once $\breve{\overline{\beta}}^{-1}(xN)$ is known. So β is tame, as required.

Nonperiodic Constructions

A *tiling* of $\mathcal{Z} = \mathbb{F}_2^m$ is a logarithmic signature of the form $\beta = [B_1, B_2]$. Nonperiodic tilings of $\mathcal{Z}$ were shown to exist for all $m \geq 6$ by Cohen, Litsyn, Vardy and Zémor [24]. Furthermore, they demonstrate a unique (up to coordinate transformations) nonperiodic tiling of $\mathbb{F}_2^6$, and explicitly construct nonperiodic tilings for all $m \geq 7$. For odd m their construction is as follows.

Let $\nu = 2^n - 1, n \geq 3$, and let $h_1, \ldots, h_\nu$ be a sorted list of nonzero elements of $\mathbb{F}_2^n$. Let $\pi = (1,4,2)(3,6)(5,7)(8,9)(10,11)\cdots(\nu-1,\nu)$ be the permutation on the set $\{1, \ldots, \nu\}$. Consider $A_0, A_1, B_2 \subset \mathbb{F}_2^{2n+1}$ given by

$$A_0 = \left\{ \begin{array}{cccc} \mathbf{0} & h_1 & \cdots & h_\nu \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ 0 & 0 & \cdots & 0 \end{array} \right\}, \quad A_1 = \left\{ \begin{array}{cccc} \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & h_1 & \cdots & h_\nu \\ 1 & 1 & \cdots & 1 \end{array} \right\},$$

$$B_2 = \left\{ \begin{array}{cccc} \mathbf{0} & h_1 & \cdots & h_\nu \\ \mathbf{0} & h_{\pi(1)} & \cdots & h_{\pi(\nu)} \\ 0 & 0 & \cdots & 0 \end{array} \right\},$$

where elements of A_0, A_1, B_2 are represented as column vectors and $\mathbf{0}$ denotes the zero vector of length n. Let $B_1 = A_0 \cup A_1$. Then $\beta = [B_1, B_2]$ is a nonperiodic tiling of $\mathbb{F}_2^{2n+1}$. To see that β is a tiling note that a vector $z \in \mathbb{F}_2^{2n+1}$ can be written uniquely as $z = b_1 b_2$ for some $b_1 \in B_1, b_2 \in B_2$ (if the last bit of z is zero then $b_1 \in A_0$, else $b_1 \in A_1$). To see that B_1 is nonperiodic, observe that any nonzero vector in A_0 summed with any vector in A_1 yields a vector not in B_1. The nonperiodicity of B_2 may be seen by inspection of π. The case for even m is dealt with similarly. Clearly such logarithmic signatures are tame.

4.4 The MST_3 Cryptosystem

We now come to the main result of this chapter: a cryptanalysis of the MST_3 public key cryptosystem. In Subsections 4.4.1 to 4.4.3 we describe MST_3 and make some initial observations on its security. We also discuss the cryptanalytic results of Magliveras et al. [58] and of González Vasco et al. [36]. In Subsection 4.4.4 we describe a simplification of the cryptosystem. Finally we present our attacks against MST_3 in Subsection 4.4.5.

4.4.1 Description

Let G be a finite non-abelian group with non-trivial centre $\mathcal{Z}$, with the property that G does not split over $\mathcal{Z}$ (so G cannot be written as a direct product $G = \mathcal{Z} \times H$ for some subgroup H). The MST_3 cryptosystem works as follows.

Key Generation:

- Generate a tame logarithmic signature $\beta = [B_1, \ldots, B_s] := (\beta_{ij})$ of type $(r_1, \ldots, r_s)$ for $\mathcal{Z}$.

- Generate a random cover $\alpha = [A_1, \ldots, A_s] := (\alpha_{ij})$ of the same type as β for a certain (large) subset $J \subseteq G$.
- Select random elements $t_0, \ldots, t_s \in G \backslash \mathcal{Z}$ and compute $\overline{\alpha} = [\overline{A}_1, \ldots, \overline{A}_s] := (\overline{\alpha}_{ij})$, where $\overline{A}_k = t_{k-1}^{-1} A_k t_k$ for $k = 1, \ldots, s$.
- Compute $\gamma := (\gamma_{ij}) = (\beta_{ij}\overline{\alpha}_{ij})$.

The pair (α, γ) is the public key, while $(\beta, (t_0, \ldots, t_s))$ is the corresponding private key.

Encryption:

A message $p \in \mathbb{Z}_{|\mathcal{Z}|}$ is encrypted as the pair $(\breve{\alpha}(p), \breve{\gamma}(p)) := (y_1, y_2)$ (recall that given covers α, γ, one can efficiently compute the mappings $\breve{\alpha}$ and $\breve{\gamma}$).

Decryption:

The plaintext p can be obtained from the ciphertext (y_1, y_2) as follows:

- Since $y_2 = \breve{\gamma}(p) = \beta_{1j_1}\overline{\alpha}_{1j_1} \cdot \beta_{2j_2}\overline{\alpha}_{2j_2} \cdots \beta_{sj_s}\overline{\alpha}_{sj_s}$, and the elements β_{ij} are in the centre of G, we have

$$\begin{aligned} y_2 &= (\beta_{1j_1}\beta_{2j_2} \cdots \beta_{sj_s}) t_0^{-1} (\alpha_{1j_1}\alpha_{2j_2} \cdots \alpha_{sj_s}) t_s \\ &= \breve{\beta}(p) t_0^{-1} \breve{\alpha}(p) t_s \\ &= \breve{\beta}(p) t_0^{-1} y_1 t_s. \end{aligned}$$

  As a result one can compute $\breve{\beta}(p) = y_2 t_s^{-1} y_1^{-1} t_0$.

- Now one can recover $p = \breve{\beta}^{-1}(y_2 t_s^{-1} y_1^{-1} t_0)$, since β is tame.

We note that Lempken et al [50] require the random cover α to have the property that $A_k \subseteq G \backslash \mathcal{Z}$ for $k = 1, \ldots, s$. We have dropped this requirement, since: the property is not needed for the encryption and decryption algorithms to work correctly; the property holds with high probability if the elements α_{ij} are chosen uniformly and independently at random; we wish to allow all covers α as valid private keys for the purposes of our cryptanalysis.

It seems reasonable to assume that the elements of α and the elements t_i are chosen uniformly and independently at random. But the cryptosystem is not yet

completely specified: a suitable platform group G needs to be defined, and we need to specify how to generate the logarithmic signature β. We discuss these issues next.

4.4.2 A Realisation of MST_3

In [50], the authors propose a practical realisation for MST_3 using Suzuki 2-groups. (See Higman [42] for a description of these groups.) Let $m \geq 3$ be a natural number, not a power of 2. Let θ be a non-trivial automorphism of odd order of the finite field $\mathbb{F}_q$, where $q = 2^m$. The Suzuki 2-group G of order q^2 can be realised as the subgroup of $GL_3(q)$ consisting of the matrices

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

Thus $G = \{S(a,b) : a, b \in \mathbb{F}_q\}$ with centre $\mathcal{Z} = \{S(0,b) : b \in \mathbb{F}_q\}$. Multiplication and inversion in G are given by

$$S(a,b) \cdot S(x,y) = S(a+x, b+y+a^\theta x),$$

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

It follows that all elements in the centre have order 2, while elements not in the centre have order 4.

Lempken et al [50] impose an extra condition on α when Suzuki 2-groups are used as a platform for MST_3, namely that no two elements of a set A_i should lie in the same coset of $\mathcal{Z}$. Since this condition holds for an overwhelming proportion of keys for interesting parameters (and since the condition is not relevant to our attacks) we ignore it for the sake of simplicity.

The issue of how to generate β is not discussed in depth in [50], but we will suppose β is constructed as an ATLS. This is most general yet practical method we know for generating tame logarithmic signatures for $\mathcal{Z}$. Nonperiodic tilings of type (r_1, r_2) are too costly in terms of storage. Moreover, it is not clear how a

general strategy would work to decompose a nonperiodic tiling into a logarithmic signature of type $(r_1, \ldots, r_s)$ for $s > 2$.

Again for storage reasons, the number of amalgamation operations in the ATLS construction has to be kept small: an amalgamation increases the number of elements we have to store by $|E_i||E_j| - (|E_i| + |E_j|)$, and so an indiscriminate use of amalgamation could lead to an exponential storage requirement. From the perspective of efficiency, generating an ATLS of type $(2, 2, \ldots, 2)$ is very attractive (though this would mean that we are unable to use amalgamation to construct them).

4.4.3 Previous Work

In this subsection, we briefly review previous work addressing the security of MST_3, and make some elementary observations on the system's security.

In [50], the authors of MST_3 provide a brief discussion on the security of the scheme, and give an attack on the cryptosystem in the passive adversary model with complexity approximately q^2 when Suzuki 2-groups are used, where $q = |\mathcal{Z}| = |G/\mathcal{Z}|$. (Note that q is exponential in the security parameter, so attacks that are polynomial in q in fact have exponential complexity.)

Magliveras et al [58] provide a better attack with complexity $\mathcal{O}(q)$. They only claim that their attack applies when the Suzuki 2-groups are used as the platform, but in fact their attack works for any platform group. We provide a similar generic attack in Section 4.4.4 below, as the first step in our cryptanalysis. Magliveras et al go on to show that MST_3 is insecure whenever β is a canonical logarithmic signature. (In fact their attack does not work in the interesting special case when $d_i = 1$ for all i, as they need that the sum of the vectors in a subspace is zero; our cryptanalysis will cover this special case.) Note that it is easy to avoid the attack in [58]: either choose $d_i = 1$ for all i, or generate an ATLS as described in Subsection 4.3 (which is very unlikely to be canonical).

The authors of MST_3 assume [50, Section 1] that a randomly chosen cover α in a finite group will (with overwhelming probability) induce a one-way function $\breve{\alpha}$. This is a reasonable assumption, but the authors claim (in Section 4.4 of their paper)

that this assumption is not actually needed to establish the security of MST_3 (in a passive model). Gonzalez Vasco, Perez del Pozo and Taborda Duarte [36] provide strong evidence that this last claim is false, by showing that when α does not induce a one-way function, MST_3 is insecure unless the quotient $|\mathcal{Z}|/|J|$ is large. They then provide experimental evidence that $|\mathcal{Z}|/|J|$ is usually rather small. Gonzalez Vasco et al also show that a randomised version of MST_3 is insecure in the sense of indistinguishability, even for passive adversaries.

The papers above still leave open the question of whether MST_3 is secure in practice if canonical transversal logarithmic signatures are avoided in the generation of the private key. The aim of the next section is to provide a practical cryptanalysis of the MST_3 cryptosystem when private keys are generated using the ATLS method.

We close this section with two elementary remarks on the security of the scheme:

1. Note that although the private key consists of the tame logarithmic signature β and the $s+1$ randomly generated elements $\{t_0, \ldots, t_s\}$, the $s-1$ elements $t_1, \ldots, t_{s-1}$ are not actually needed: only β and t_0, t_s are used in the decryption procedure.

2. Note that any triplet of the form $(\beta, g \cdot t_0, g \cdot t_s)$, where g is in the centralizer of J (in particular, if $g \in \mathcal{Z}$), can be used to decrypt the ciphertext. Thus there are many equivalent private keys.

4.4.4 A Simplification

The aim of this section is to simplify the problem of cryptanalysing MST_3: we will show that it is sufficient to consider a much smaller class of public and private keys than in the original definition. This simplification works for all suitable platform groups, not just the Suzuki 2-groups considered above.

Let (α, γ) be a public key for MST_3, with $(\beta, (t_0, t_1, \ldots, t_s))$ the corresponding private key. Recall that $\alpha = [A_1, A_2, \ldots, A_s]$ and $\beta = [B_1, B_2, \ldots, B_s]$, and define subsets H_i by $\gamma = [H_1, H_2, \ldots, H_s]$. Note that the algorithm for deriving γ from the

private key implies that

$$\gamma_{ij} = \beta_{ij} t_{i-1}^{-1} \alpha_{ij} t_i. \tag{4.4.1}$$

Define elements p_i, q_i and z_i by setting $p_0 = q_0 = z_0 = 1$ and for $i \in \{1, 2, \ldots, s\}$ defining

$$p_i = \prod_{k=1}^{i} \alpha_{k1},\ q_i = \prod_{k=1}^{i} \gamma_{k1} \text{ and } z_i = \prod_{k=1}^{i} \beta_{k1}.$$

Note that (4.4.1) and the fact that the elements β_{ij} are central together imply that

$$q_i = \prod_{k=1}^{i} (\beta_{k1} t_{k-1}^{-1} \alpha_{k1} t_k) = z_i t_0^{-1} p_i t_i. \tag{4.4.2}$$

Define $\alpha' = [A'_1, A'_2, \ldots, A'_s]$, $\gamma' = [H'_1, H'_2, \ldots, H'_s]$ and $\beta' = [B'_1, B'_2, \ldots, B'_s]$ by

$$A'_i = p_{i-1} A_i p_i^{-1},$$
$$H'_i = q_{i-1} H_i q_i^{-1},$$
$$B'_i = z_{i-1} B_i z_i^{-1}.$$

The following lemma is easy to prove.

Lemma 4.4. *We use the notation defined above. For all $i \in \{1, 2, \ldots, s\}$, the first elements α'_{i1}, γ'_{i1}, β'_{i1} of the sets A'_i, H'_i, B'_i are all equal to the identity. Moreover,*

$$\breve{\alpha'}(x) = \breve{\alpha}(x) p_s^{-1}, \breve{\gamma'}(x) = \breve{\gamma}(x) q_s^{-1} \text{ and } \breve{\beta'}(x) = \breve{\beta}(x) z_s^{-1}.$$

In particular, β' is a logarithmic signature for $\mathcal{Z}$, and α' is a cover for some subset $\mathcal{J}'$ of G.

Lemma 4.5. *Let (α, γ) be a public key for* MST$_3$*, with $(\beta, (t_0, t_1, \ldots, t_s))$ the corresponding private key. Define α', γ' and β' as above, and let $t'_0 = t'_1 = \cdots = t'_s = t_0$. Then (α', γ') is a public key for* MST$_3$*, with corresponding private key $(\beta', (t'_0, t'_1, \ldots, t'_s))$.*

PROOF. Suppose we use α', β' and $t'_0, t'_1, \ldots, t'_s$ to generate a public key (α', δ), where $\delta = [D_1, D_2, \ldots, D_s]$, so $\delta_{ij} = \beta'_{ij} (t'_{i-1})^{-1} \alpha'_{ij} t'_i$. It suffices to show that $\delta = \gamma'$.

But

$$\begin{aligned}\delta_{ij} &= \beta'_{ij} t_0^{-1} \alpha'_{ij} t_0 \\ &= z_{i-1} \beta_{ij} z_i^{-1} t_0^{-1} \alpha'_{ij} t_0 \\ &= z_{i-1} \beta_{ij} z_i^{-1} t_0^{-1} p_{i-1} \alpha_{ij} p_i^{-1} t_0 \\ &= \beta_{ij} z_{i-1} z_i^{-1} t_0^{-1} p_{i-1} \alpha_{ij} p_i^{-1} t_0 \\ &= \beta_{ij} \beta_{i1}^{-1} t_0^{-1} p_{i-1} \alpha_{ij} p_i^{-1} t_0.\end{aligned}$$

Equation (4.4.2) implies that $t_0^{-1} p_{i-1} = z_{i-1}^{-1} q_{i-1} t_{i-1}^{-1}$ and $p_i^{-1} t_0 = t_i q_i^{-1} z_i$. So

$$\begin{aligned}\delta_{ij} &= \beta_{ij} \beta_{i1}^{-1} z_{i-1}^{-1} q_{i-1} t_{i-1}^{-1} \alpha_{ij} t_i q_i^{-1} z_i \\ &= \beta_{ij} q_{i-1} t_{i-1}^{-1} \alpha_{ij} t_i q_i^{-1}\end{aligned}$$

by the definition of z_i, and since z_i is central. But

$$\gamma'_{ij} = q_{i-1} \gamma_{ij} q_i^{-1} = q_{i-1} \beta_{ij} t_{i-1}^{-1} \alpha_{ij} t_i q_i^{-1}$$

by (4.4.1). Since β_{ij} is central, we have that $\gamma'_{ij} = \delta_{ij}$, as required.

We define the **Restricted OWE problem** for MST_3 as follows. The input is a public key (α, γ) for MST_3 and a challenge ciphertext (y_1, y_2). The public key must have the extra property that $\alpha_{i1} = \gamma_{i1} = 1$ for $1 \leq i \leq s$; the corresponding private key must have the property that $t_0 = t_1 = \cdots = t_s$ and also that $\beta_{i1} = 1$ for $1 \leq i \leq s$. The output is the plaintext p corresponding to the ciphertext (y_1, y_2).

Theorem 4.6. *There is a polynomial time reduction from the OWE problem for* MST_3 (*for general keys*) *to the Restricted OWE problem for* MST_3. (*Indeed, only one call to the Restricted OWE oracle is needed.*)

PROOF. Let $\mathcal{O}(\alpha, \gamma, y_1, y_2)$ be an oracle for the restricted OWE problem for MST_3. We show that this oracle can be used to solve the OWE problem for MST_3 for general keys.

Suppose (α, γ) is an (unrestricted) public key, with corresponding private key $(\beta, (t_0, t_1, \ldots, t_s))$. Let (y_1, y_2) be a challenge ciphertext with corresponding message p.

Suppose we are given (α, γ) and (y_1, y_2). Define (α', γ') as above. Note that α' and γ' can be efficiently constructed from α and γ using public information only. By Lemmas 4.4 and 4.5, (α', γ') is a public key with corresponding private key $(\beta', (t_0, t_0, \ldots, t_0))$, and these keys satisfy our restrictions. Define $y_1' = y_1 p_s^{-1}$ and $y_2' = y_2 q_s^{-1}$. Again, we note that p_s and q_s are defined using public information, so y_1' and y_2' can be efficiently computed from the information we are given.

We call the oracle $\mathcal{O}$ on $(\alpha', \gamma', y_1', y_2')$, and receive a message p such that

$$(\alpha'(p), \gamma'(p)) = (y_1', y_2').$$

Then p is the message we require, since

$$\breve{\alpha}(p) = \breve{\alpha'}(p) p_s = y_1' p_s = y_1 p_s^{-1} p_s = y_1 \text{ and}$$
$$\breve{\gamma}(p) = \breve{\gamma'}(p) q_s = y_2' q_s = y_2 q_s^{-1} q_s = y_2.$$

4.4.5 A Cryptanalysis

This subsection is concerned with the cryptanalysis of MST_3. We first provide an (exponential) attack that is independent of the underlying platform group G. Next we outline an approach that works for platform groups G such that $G/\mathcal{Z}$ is abelian. Finally, we report on our experiments with implementing these attacks in the case when G is a Suzuki 2-group.

A Generic Attack

From now on, we assume our public key (α, γ) and corresponding private key $(\beta, (t_0, t_1, \ldots, t_s))$ are such that

$$\alpha_{i1} = \beta_{i1} = \gamma_{i1} = 1$$

for $1 \leq i \leq s$ and there exists $t \in G$ such that

$$t_0 = t_1 = \cdots = t_s = t.$$

Theorem 4.6 shows that we may do this without loss of generality.

The secret logarithmic signature β can be obtained from the public key once t is known, since

$$\beta_{ij} = \gamma_{ij} t^{-1} \alpha_{ij}^{-1} t. \tag{4.4.3}$$

So we may think of the private key of the cipher as being the single group element t.

Define $\bar{t} \in G/\mathcal{Z}$ by $\bar{t} = t\mathcal{Z}$. Let $z \in \mathcal{Z}$. Replacing t by tz does not change the value of the right hand side of (4.4.3), and does not change the output of the decryption algorithm. So once $\bar{t}$ is known, the cryptosystem is broken as an equivalent private key can be derived efficiently. A search over all $|G/\mathcal{Z}|$ possibilities for $\bar{t}$ will therefore break the cipher. This cryptanalysis can be regarded as a generalisation of the 'attack on t_0' presented by Magliveras et al [58, Subsection 4.1] in the case of Suzuki 2-groups. Note that this attack will in general be exponential in the security parameter, but is clearly much more efficient than a naive exhaustive search over possible private keys.

A More Efficient Approach

We would like to break the cipher much more efficiently than the attack in the previous subsection. We are most interested in the case when G is a Suzuki 2-group. However, in this subsection we consider a more general situation that includes these groups: the case when $G/\mathcal{Z}$ is abelian.

Let $t' \in G$ be a guess for the value of t. (Of course, it is only the coset $t'\mathcal{Z}$ that matters.) Define

$$\mathfrak{b}_{ij} = \gamma_{ij} (t')^{-1} \alpha_{ij}^{-1} t'$$

for all i and j. (Note that $\mathfrak{b}_{ij}$ can be computed without knowledge of the private key.) Define a cover $\mathfrak{b} = [\mathfrak{B}_1, \mathfrak{B}_2, \ldots, \mathfrak{B}_s]$ for some subset $\mathfrak{J}$ of G by

$$\mathfrak{B}_i = [\mathfrak{b}_{i1}, \mathfrak{b}_{i2}, \ldots, \mathfrak{b}_{ir_i}].$$

Let $q = |\mathcal{Z}|$, and define the map $\omega : \mathbb{Z}_q \to G$ by

$$\omega(x) = \breve{\gamma}(x) t'^{-1} \breve{\alpha}(x)^{-1} t'$$

for all $x \in \mathbb{Z}_q$. (Note that ω can also be computed without knowledge of the private key.)

When $t \equiv t' \bmod \mathcal{Z}$ (so our guess for t' is correct) we have that $\mathfrak{b} = \beta$ and $\omega = \breve{\beta} = \breve{\mathfrak{b}}$. In particular, when we have guessed correctly:

1. $\mathfrak{b}$ is a tame logarithmic signature for $\mathcal{Z}$, and
2. $\omega = \breve{\mathfrak{b}}$.

Lemma 4.7. *If the above two conditions are satisfied for a particular guess t', then $(\mathfrak{b}, (t', t', \ldots, t'))$ is an equivalent private key for the cipher.*

PROOF. Since $\mathfrak{b}$ is a tame logarithmic signature for $\mathcal{Z}$, the pair $(\mathfrak{b}, (t', t', \ldots, t'))$ is a valid private key. Let (y_1, y_2) be the ciphertext obtained as encryption of the plaintext p under the public key corresponding to the private key $(\beta, (t, t, \ldots, t))$. So $y_1 = \breve{\alpha}(p)$ and $y_2 = \breve{\gamma}(p)$. Decryption using the key $(\mathfrak{b}, (t', t', \ldots, t'))$ gives us

$$\breve{\mathfrak{b}}^{-1}(y_2 t'^{-1} y_1^{-1} t') = \omega^{-1}(y_2 t'^{-1} y_1^{-1} t') = \omega^{-1}(\breve{\gamma}(p) t'^{-1} \breve{\alpha}(p)^{-1} t') = \omega^{-1}(\omega(p)) = p,$$

as required.

Lemma 4.8. *Suppose that $G/\mathcal{Z}$ is abelian. For any choice of t' we have that $\mathfrak{b}$ is a cover of a subset of $\mathcal{Z}$.*

PROOF. For any i and j we have that

$$\begin{aligned}\mathfrak{b}_{ij}\mathcal{Z} &= \gamma_{ij}(t')^{-1}\alpha_{ij}^{-1}t'\mathcal{Z} = \gamma_{ij}\alpha_{ij}^{-1}t'^{-1}t'\mathcal{Z} \\ &= \gamma_{ij}\alpha_{ij}^{-1}t^{-1}t\mathcal{Z} = \gamma_{ij}t^{-1}\alpha_{ij}^{-1}t\mathcal{Z} = \beta_{ij}\mathcal{Z} = \mathcal{Z}.\end{aligned}$$

So the elements of the cover $\mathfrak{b}$ all lie in $\mathcal{Z}$, as required.

Lemma 4.9. *Suppose that $G/\mathcal{Z}$ is abelian. For any choice of t' we have that $\omega = \breve{\mathfrak{b}}$.*

PROOF. By an abuse of notation we will identify the sets $\mathbb{Z}_q$ and $\mathbb{Z}_{r_1} \times \ldots \times \mathbb{Z}_{r_s}$ via the map λ_α defined in Subsection 4.1. So we think of the domain of the functions $\breve{\mathfrak{b}}$ and ω as being $\mathbb{Z}_{r_1} \times \ldots \times \mathbb{Z}_{r_s}$ rather than $\mathbb{Z}_q$.

We first note that

$$\mathfrak{b}_{i1} = \gamma_{i1}(t')^{-1}\alpha_{i1}^{-1}t' = (t')^{-1}t' = 1$$

for all i. In particular,

$$\breve{\mathfrak{b}}(x_1, x_2, \ldots, x_k, 1, 1, \ldots, 1) = \prod_{i=1}^{s} \mathfrak{b}_{ix_i} = \prod_{i=1}^{k} \mathfrak{b}_{ix_i}.$$

Moreover, writing $x = (x_1, x_2, \ldots, x_k, 1, 1, \ldots, 1)$, we find that

$$\breve{\alpha}(x) = \prod_{i=1}^{k} \alpha_{ix_i} \text{ and } \breve{\gamma}(x) = \prod_{i=1}^{k} \gamma_{ix_i},$$

since $\alpha_{i1} = \gamma_{i1} = 1$.

We will prove the lemma by induction. Let $P(k)$ be the following statement:

$$\omega(x_1, x_2, \ldots, x_s) = \breve{\mathfrak{b}}(x_1, x_2, \ldots, x_s) \text{ whenever } x_{k+1} = x_{k+2} = \cdots = x_s = 1.$$

The first paragraph of the proof shows that $\breve{\mathfrak{b}}(1, 1, \ldots, 1) = 1$. Moreover, since $\breve{\gamma}(1, 1, \ldots, 1) = \breve{\alpha}(1, 1, \ldots, 1)$ we find that $\omega(1, 1, \ldots, 1) = 1$. Hence $P(0)$ holds.

Assume, as an inductive hypothesis that $P(k-1)$ holds. Our assumption that $G/\mathcal{Z}$ is abelian implies that $\mathfrak{b}_{ij} \in \mathcal{Z}$ for all i and j. Let $x_1, x_2, \ldots, x_k$ be fixed. Define x and x' by

$$x = (x_1, x_2, \ldots, x_k, 1, \ldots, 1) \text{ and } x' = (x_1, x_2, \ldots, x_{k-1}, 1, \ldots, 1).$$

Then

$$\begin{aligned}
\breve{\mathfrak{b}}(x) &= \prod_{i=1}^{k} \mathfrak{b}_{ix_i} = \breve{\mathfrak{b}}(x')\mathfrak{b}_{kx_k} = \omega(x')\mathfrak{b}_{kx_k} \text{ (by our inductive hypothesis)} \\
&= \breve{\gamma}(x')t'^{-1}\breve{\alpha}(x')^{-1}t'\mathfrak{b}_{kx_k} \\
&= \breve{\gamma}(x')\mathfrak{b}_{kx_k}t'^{-1}\breve{\alpha}(x')^{-1}t' \text{ (since } \mathfrak{b}_{kx_k} \in \mathcal{Z}) \\
&= \left(\prod_{i=1}^{k-1} \gamma_{ix_i}\right) \gamma_{kx_k} t'^{-1} \alpha_{kx_k}^{-1} t' t'^{-1} \left(\prod_{i=1}^{k-1} \alpha_{ix_i}\right)^{-1} t' \\
&= \breve{\gamma}(x)t'^{-1}\breve{\alpha}(x)t' \\
&= \omega(x).
\end{aligned}$$

So $P(k)$ is true whenever $P(k-1)$ is true. By induction, $P(s)$ holds and so the lemma follows.

Theorem 4.10. *Let G be such that $G/\mathcal{Z}$ is abelian. Then $(\mathfrak{b}, (t', t', \ldots, t'))$ is an equivalent private key for MST_3 if and only if $\breve{\mathfrak{b}} : \mathbb{Z}_{|\mathcal{Z}|} \to \mathcal{Z}$ is a bijection whose inverse is efficiently computable.*

PROOF. It is clear that a private key $(\mathfrak{b}, (t', t', \ldots, t'))$ for MST_3 must have the property that $\breve{\mathfrak{b}} : \mathbb{Z}_{|\mathcal{Z}|} \to \mathcal{Z}$ is a bijection whose inverse is efficiently computable.

To prove the converse, first note that $\mathfrak{b}$ is a cover for a subset of $\mathcal{Z}$, by Lemma 4.8. We are assuming that $\breve{\mathfrak{b}}$ is a bijection whose inverse is efficiently computable, so in fact $\mathfrak{b}$ is a tame logarithmic signature for $\mathcal{Z}$. By Lemma 4.9, we have $\omega = \breve{\mathfrak{b}}$. Hence, by Lemma 4.7, we find that $(\mathfrak{b}, (t', t', \ldots, t'))$ is an equivalent private key for the cipher, as required.

So a general approach to finding a private key for MST_3 may be described as follows. We use the fact that $\breve{\mathfrak{b}}$ must be a bijection to derive some conditions on t'. If applying these conditions leads to a small number of possibilities for t', we perform an exhaustive search to find a private key that works. If there are still many possibilities for t', we choose one at random and hope that $\breve{\mathfrak{b}}^{-1}$ is efficiently computable: the probability that this will be successful will depend on the way that the logarithmic signature β has been generated. In the next subsection we will analyse the performance of this attack for Suzuki 2-groups.

Recovering the Key in Practice

We now describe our computer experiments to verify that the method outlined above works well in practice. All computer experiments were performed using the mathematics software SAGE [77].

Let $m = 81$. Our platform group is the Suzuki 2-group over the field $\mathbb{F}_q$, where $q = 2^m$. The generic attack described in Subsection 4.4.5 requires a search of size q to succeed: we fix $m = 81$ so that this generic attack is not feasible. Note that the public key is already rather long when $m = 81$: in the most efficient case we consider (Case 1 below), we need over $19\,000$ bits to store the non-identity elements

in the logarithmic signatures α and γ. Our techniques do not seem to depend significantly on the automorphism θ in the definition of the Suzuki 2-group, so we fix θ to be the squaring automorphism in all our experiments.

We construct our logarithmic signature β using the ATLS method discussed in Subsection 4.3. We wish to generate logarithmic signatures of type $(r_1, r_2, \ldots, r_s)$, where $\prod_{i=1}^{s} r_i = 2^m$. Note that the integers r_i must be fairly small, as otherwise the logarithmic signatures we produce cannot be stored efficiently. The precise method we use to generate β depends on its type: we give explicit details below. By Theorem 4.6, it is enough to consider logarithmic signatures that have an extra property: the elements β_{i1} are all equal to the identity. Our methods for generating β always produce logarithmic signatures with this property (and no generality is lost by generating logarithmic signatures in this way).

We follow the approach in Subsection 4.4.5 in our cryptanalysis. In the notation of that subsection, we begin by deriving conditions that t must satisfy as a consequence of the fact that β is bijective. We then choose t' at random subject to these conditions; our attack is successful if we obtain a valid private key after trying a small number of guesses t'. In our experiments, our attack was always successful.

Recall the notation $S(a, b)$ for an element in the Suzuki 2-group defined in Subsection 4.4.2. Our remark at the end of Subsection 4.4.3 shows that we may assume that $t = S(x, 0)$ where $x \in \mathbb{F}_q$ is unknown, and so we restrict our guess t' to be of the form $S(y, 0)$ for some $y \in \mathbb{F}_q$. The conditions on t that we derive are $\mathbb{F}_2$-linear conditions, so it is easy to choose t' satisfying these conditions at random. The precise conditions on t we derive will depend on the number of components r_i of the type of β that are equal to 2: when there are many of such components, the conditions we derive are weaker. For this reason, we provide three cases to illustrate our methods. In Case 1, $r_i = 2$ for all i. In this case we find no conditions on t, but simply randomly choosing a small number of values for t' leads to a successful attack. In Case 2, $r_i \neq 2$ for all i. In this case, we find that every condition we derive restricts t' to such a small number of possibilities that a negligible exhaustive search can be carried out. Case 3, with approximately half of the components of the type

of β being equal to 2, illustrates an intermediate case. Here, each condition limits the number of possibilities for t' significantly (to approximately 2^{40} possibilities). Very few guesses t' can satisfy two of these conditions simultaneously, so combining two conditions allows us to derive an equivalent private key by a negligible exhaustive search.

Case 1: β has type $(2, 2, \ldots, 2)$

In this case, we assume β consists of 81 blocks of size 2. Such logarithmic signatures are very attractive from the perspective of efficiency: we only need to store the 81 non-trivial elements in the sets B_i; moreover these elements form a basis of $\mathcal{Z}$ when $\mathcal{Z}$ is considered as a 81-dimensional vector space over $\mathbb{F}_2$, and computations with β can be carried out using straightforward linear algebra. (We note that β is an example of *canonical logarithmic signature* as defined in [58]; however the attack described in that paper does not work in this particular case.)

We derive public and private keys for the MST_3 cryptosystem as follows. We randomly choose a generating set $\{z_1, \ldots, z_{81}\}$ for $\mathcal{Z}$. Define elements $d_{i2} \in \mathbb{F}_q$ by $z_i = S(0, d_{i2})$, so the elements d_{i2} form an $\mathbb{F}_2$-basis for $\mathbb{F}_q$. Set

$$\beta = [B_1, \ldots, B_{81}], \text{ where } B_i = \{1, S(0, d_{i2})\}.$$

We then generate elements $e_{i2}, f_{i2} \in \mathbb{F}_q$ at random, and define

$$\alpha = [A_1, \ldots, A_{81}], \text{ where } A_i = \{1, S(e_{i2}, f_{i2})\}.$$

Let $t = S(x, 0)$ where $x \in \mathbb{F}_q$ is chosen at random. We construct γ as specified in the definition of MST_3. So we define

$$\begin{aligned}
\gamma_{i2} &= \beta_{i2} t^{-1} \alpha_{i2} t \\
&= S(0, d_{i2}) S(x, x^\theta x) S(e_{i2}, f_{i2}) S(x, 0) \\
&= S(e_{i2}, d_{i2} + f_{i2} + e_{i2} x^\theta + e_{i2}^\theta x) =: S(e_{i2}, g_{i2}),
\end{aligned}$$

and set $\gamma = [C_1, \ldots, C_{81}]$, where $C_i = \{1, \gamma_{i2}\}$.

no. guesses t'	1	2	3	4	5	6	7	8	9
frequency	2829	2111	1429	1048	799	490	374	279	181
no. guesses t'	10	11	12	13	14	15	16	17	18
frequency	133	98	66	47	31	26	19	11	5
no. guesses t'	19	20	21	22	23	24	25	26	27
frequency	3	7	7	4	2	1	0	0	0

Table 4.1: Experimental Results for Case 1

Our attack works as follows. Let $t' = S(y, 0)$ be a random guess for t. We form $\mathfrak{b} = [\mathfrak{B}_1, \ldots, \mathfrak{B}_{81}]$, where $\mathfrak{B}_i = \{1, \mathfrak{b}_{i2}\}$ and $\mathfrak{b}_{i2}$ is given by

$$\begin{aligned}\mathfrak{b}_{i2} &= \gamma_{i2} {t'}^{-1} \alpha_{i2}^{-1} t' \\ &= S(e_{i2}, g_{i2}) S(y, y^{\theta} y) S(e_{i2}, {e_{i2}}^{\theta} e_{i2} + f_{i2}) S(y, 0) \\ &= S(0, g_{i2} + f_{i2} + e_{i2} y^{\theta} + {e_{i2}}^{\theta} y).\end{aligned}$$

If the set $\{\mathfrak{b}_{i2}\}_{i=1}^{81}$ is linearly independent, then $\breve{\mathfrak{b}}$ is a bijection and it follows from Theorem 4.10 that we have an equivalent private key. If the set is linearly dependent, we repeat this process with another guess t'.

We have implemented this attack for 10 000 random instances of MST_3. The results of this experiment, which took a few minutes to carry out on a standard PC, are given in Table 4.1. The average number of guesses for t' before finding an equivalent private key, was approximately 3.47. Thus the scheme is insecure in this case.

Case 2: β has type $(8, 64, 64, \ldots, 64)$

We now consider the case when our logarithmic signatures consist of one block of size 8 and thirteen blocks of size 64.

We construct β as follows. We generate a random basis $\{z_1, \ldots, z_{81}\}$ for $\mathcal{Z}$. We consider the subgroup chain

$$1 = \mathcal{Z}_0 < \mathcal{Z}_1 < \cdots < \mathcal{Z}_{27} = \mathcal{Z},$$

where $Z_i = \langle z_1, \ldots, z_{3i} \rangle$ for $1 \leq i \leq 27$. We form a transversal logarithmic signature of type $(8, 8, \ldots, 8)$ (with 27 blocks in total), whose ith block is a transversal for Z_{i-1}

in Z_i containing the identity as its first element. We then randomly amalgamate 26 blocks of size 8 in pairs to form 13 blocks of size 64. Reordering the blocks we have constructed an ATLS $\beta = [B_1, B_2, \ldots, B_{14}]$ of type $(8, 64, 64, \ldots, 64)$ for $\mathcal{Z}$. Define elements $d_{ij} \in \mathbb{F}_q$ by $\beta_{ij} = S(0, d_{ij})$.

We generate the element $t = S(x, 0)$, the elements $\alpha_{ij} = S(e_{ij}, f_{ij})$, the elements $\gamma_{ij} = S(e_{ij}, g_{ij})$, and the covers α and γ as in Case 1. In particular, the equation

$$g_{ij} = d_{ij} + f_{ij} + e_{ij}^{\theta} x + e_{ij} x^{\theta}$$

holds.

Our attack recovers a private key directly by a small exhaustive search, rather than guessing an equivalent private key. Lemma 4.2 implies that there exists i and j such that $j \geq 2$ and $B_i \cdot b_{ij} = B_i$. There is only a small number of possibilities for i and j so (using a negligible exhaustive search) we may assume that a valid choice for i and j are known. We know that

$$d_{ij} = g_{ij} + f_{ij} + e_{ij}^{\theta} x + e_{ij} x^{\theta}. \tag{4.4.4}$$

Moreover, when $B_i \cdot b_{ij} = B_i$, the equation

$$d_{ij} + d_{ik} = d_{il} \tag{4.4.5}$$

holds for at least $|B_i| - 2$ pairs of indices k, l where $2 \leq k, l \leq |B_i|$ and where j, k and l are distinct. Writing u_{ijkl} for $u_{ij} + u_{ik} + u_{il}$, equations (4.4.4) and (4.4.5) combine to give

$$g_{ijkl} + f_{ijkl} = {e_{ijkl}}^{\theta} x + e_{ijkl} x^{\theta}. \tag{4.4.6}$$

Note that the elements e_{ijkl}, f_{ijkl} and g_{ijkl} are all known (forming part of the public key (α, γ)), but x is unknown. For a fixed $e \in \mathbb{F}_q$, the map

$$\phi_e : \mathbb{F}_q \to \mathbb{F}_q \text{ given by } x \mapsto e^{\theta} x + e x^{\theta}$$

is an $\mathbb{F}_2$-linear map. Moreover, when $e \neq 0$, we have that ϕ_e has a kernel of size 2. Assuming (as is very likely) that e_{ijkl} is non-zero, we find that each equation of the form (4.4.6) is satisfied by at most two possibilities for x (and these choices

are easily computed using elementary linear algebra). There are fewer than 2^{18} choices for i, j, k and l. Once these choices are fixed, there are at most 2 values for x that satisfy equation (4.4.6). So we can recover x by an exhaustive search though 2^{20} possibilities. (For each possibility for x, we can construct $\mathfrak{b}$ and check to see whether $\breve{\mathfrak{b}}$ is a bijection: this check can be carried out efficiently for an ATLS.)

Note that this attack makes use of the fact that $|B_i| > 2$ in an essential way: if $|B_i| = 2$ then there are no valid choices for j and k. Note also that when we have a correct value for i and j the same element x will occur at least $|B_i| - 2$ times as a solution to (4.4.6) as j and k vary over all possible values: this observation can be used to recover x more efficiently. Finally, we note that when i is guessed correctly the set B_i has the property that the product of its elements must be the identity (as the same is true for any coset of a subgroup of $\mathcal{Z}$ of order 4 or more); this property can be used to find x without the need to guess j, k or l.

We implemented the attack using SAGE on a standard PC, and in each run the randomly chosen secret value x was returned correctly within 30 minutes. Thus the MST_3 cryptosystem is also insecure in this case.

Case 3: β has type $(2, 2, \ldots 2, 16, 16, \ldots, 16)$

Finally, we consider the case when β consists of 41 sets of size 2 and 10 sets of size 16. In this situation, the analogue of equation (4.4.6) does not restrict the number of possibilities for x sufficiently, and so we combine two equations to recover x.

We construct β by starting with the subgroup chain

$$1 = \mathcal{Z}_0 < \mathcal{Z}_1 < \cdots < \mathcal{Z}_{61} = \mathcal{Z},$$

where each $\mathcal{Z}_i$ has index 2 in $\mathcal{Z}_{i+1}$ for $0 \leq i \leq 40$ and index 4 for $41 \leq i \leq 60$. We form a random transversal logarithmic signature for this chain (including the identity as the first element in each transversal): this logarithmic signature will consist of 41 sets of size 2 and 20 sets of size 4. We then amalgamate the 20 sets of size 4 in pairs to form 10 sets of size 16, where the pairing of these sets is chosen

No. possibilities for x	0	1	2	4	8	16
Freq. (correct indices)	0	579	386	33	2	0
Freq. (incorrect indices)	276	543	170	10	1	0

Table 4.2: Experimental Results for Case 3

at random. The result is an ATLS $\beta = [B_1, B_2, \ldots, B_{41}, B_{42}, \ldots, B_{51}]$ of the type we are seeking. We then choose t and α, and construct γ, just as before.

Our attack in this case is as follows. Define the subgroup $H = \langle B_1, \ldots, B_{41} \rangle$. Write $\beta_{ij} = S(0, d_{ij})$, and define $V = \langle d_{i2} : 1 \leq i \leq 41 \rangle$. Note that V has dimension 41 and $H = \{S(0, v) : v \in V\}$. Clearly the image of $[B_{42}, \ldots, B_{51}]$ in $\mathcal{Z}/H$ is an ATLS for $\mathcal{Z}/H$ with no blocks of size 2. So we may proceed in the same way as in Case 2, this time working in the quotient $\mathcal{Z}/H$ to derive equations that x must satisfy modulo V. Using the notation from Case 2, we obtain equations of the form

$$g_{ijkl} + f_{ijkl} + V = \phi_{e_{ijkl}}(x), \tag{4.4.7}$$

where $\phi_{e_{ijkl}}$ is an $\mathbb{F}_2$-linear map. On the assumption that e_{ijkl} is non-zero, an equation of this form restricts x to lie in an affine subspace of dimension at most 42, and so we have reduced the size of an exhaustive search for x to 2^{42} possibilities. But a correct guess for i means that x satisfies at least $|B_i| - 2 \geq 2$ such equations as j, k and l vary. If we correctly guess two such combinations of j, k and l, we know that x lies in the intersection of two affine subspaces of dimension at most 42 (namely the solution sets corresponding to the two equations), and this reduces the number of possibilities for x to a negligible number. The validity of each possibility for x can be determined by checking the bijectivity of $\breve{\mathfrak{b}}$ as in Case 2.

Implementing these ideas, we generated 1000 random ATLSs for $\mathcal{Z}/H$. For each ATLS we picked a random pair of equations (4.4.7) where the indices i, j, k and l have been guessed correctly, and computed the size of the intersection of the two solution sets. We did the same when the indices have been guessed incorrectly, to check that the number of possibilities for x is not too large in this case. We record the results in Table 4.2.

As Table 4.2 indicates, in either case the number of possibilities for x is small. There are less than 2^{20} pairs of equations to check, and so we typically expect (guided by Table 4.2) an exhaustive search for x to be of size 2^{24} at most. (Furthermore, within this search we expect x to occur with a relatively high frequency, since it appears for *every* correct pair of equations (4.4.7).) Thus we conclude that the MST_3 cryptosystem is also insecure in this case.

We conclude that until a method for generating secure tame logarithmic signatures is invented, the MST_3 cryptosystem is insecure.

4.5 A Revised Version of MST_3

After the publication of [12], Svaba and van Trung [89] presented a revised version of MST_3. The platform group (Suzuki 2-groups) and method of generating tame logarithmic signatures for $\mathcal{Z}$ (ATLS) remain the same, but the encryption and decryption processes are changed slightly, and there is extra key material. Svaba and van Trung discuss lower bounds on the workload of known direct atttacks on the private key, all of which run in exponential time. Methods are described for efficiently inverting $\breve{\beta}$ for an ATLS β for $\mathcal{Z}$, which amount to keeping track of the elementary operations applied to a canonical logarithmic signature, and reversing the process.

A chosen plaintext attack ('Matrix Permutation attack') is then presented which, although runs in exponential time, for practical parameter sizes rules out the use of logarithmic signatures for $\mathcal{Z}$ which apply only the elementary operations (1)–(3), i.e. with no amalgamation. Evidence is then provided that this line of attack is rather inefficient for general ATLS constructions involving amalgamation. Finally implementation issues are discussed.

It is natural to ask whether the cryptanalysis described in Section 4.4 of the original MST_3 cryptosystem can be applied to the revised version of MST_3, which we call MST_3^r. In this subsection we describe MST_3^r and give some preliminary comments as to what extent the foregoing cryptanalysis of MST_3 applies to MST_3^r.

The revised version MST_3^r works as follows. We indicate with an asterisk the steps that differ from the MST_3 key generation process.

Key Generation:

- Generate a tame logarithmic signature $\beta = [B_1, \ldots, B_s] := (\beta_{ij})$ of type $(r_1, \ldots, r_s)$ for $\mathcal{Z}$.

* Generate a homomorphism $f : G \to \mathcal{Z}$.

- Generate a random cover $\alpha = [A_1, \ldots, A_s] := (\alpha_{ij})$ of the same type as β for a certain (large) subset $J \subseteq G$.

* Select random elements $t_0, \ldots, t_s \in G \backslash \mathcal{Z}$ and compute $\overline{\alpha} = [\overline{A}_1, \ldots, \overline{A}_s] := (\overline{\alpha}_{ij})$, where $\overline{A}_k = t_{k-1}^{-1} A_k f(A_k) t_k$ for $k = 1, \ldots, s$.

- Compute $\gamma := (\gamma_{ij}) = (\beta_{ij}\overline{\alpha}_{ij})$.

The pair (α, γ) is the public key, while $(\beta, (t_0, \ldots, t_s), f)$ is the corresponding private key.

Encryption:

A message $p \in \mathbb{Z}_{|\mathcal{Z}|}$ is encrypted as the pair $(\breve{\alpha}(p), \breve{\gamma}(p)) := (y_1, y_2)$ (recall that given covers α, γ, one can efficiently compute the mappings $\breve{\alpha}$ and $\breve{\gamma}$).

Decryption:

The plaintext p can be obtained from the ciphertext (y_1, y_2) as follows:

- Since $y_2 = \breve{\gamma}(p) = \beta_{1j_1}\overline{\alpha}_{1j_1} \cdot \beta_{2j_2}\overline{\alpha}_{2j_2} \cdots \beta_{sj_s}\overline{\alpha}_{sj_s}$, and the elements β_{ij} are in the centre of G, we have

$$\begin{aligned} y_2 &= (\beta_{1j_1}\beta_{2j_2} \cdots \beta_{sj_s}) t_0^{-1} (\alpha_{1j_1}\alpha_{2j_2} \cdots \alpha_{sj_s}) f(\alpha_{1j_1}\alpha_{2j_2} \cdots \alpha_{sj_s}) t_s \\ &= \breve{\beta}(p) t_0^{-1} \breve{\alpha}(p) f(\breve{\alpha}(p)) t_s \\ &= \breve{\beta}(p) t_0^{-1} y_1 f(\breve{\alpha}(p)) t_s. \end{aligned}$$

  As a result one can compute $\breve{\beta}(p) = y_2 t_s^{-1} f(\breve{\alpha}(p))^{-1} y_1^{-1} t_0$.

- Now one can recover p since β is tame.

The homomorphism f is specified as $f(S(a,b)) = S(0, a^\sigma)$, for some (presumably random) $\sigma \in \mathrm{Aut}(\mathcal{Z}) = \mathrm{GL}_m(2)$. Note that if f is the trivial homomorphism then we have a description of the original MST_3 cryptosystem.

Following Subsection 4.4.4 it is straightforward to show a reduction for the OWE problem for MST_3^r for general keys to the OWE problem for MST_3^r with key pairs $(\alpha, \gamma), (\beta, t_i, f)$ satisfying $\alpha_{i1} = \gamma_{i1} = \beta_{i1}$ and $t_i = t_0 = t$, for $1 \leq i \leq s$.

In this restricted setting f induces a cover $C' = (c'_{ij}) = [C'_1, \ldots, C'_s]$ of type $(r_1, \ldots, r_s)$ for some subset of $\mathcal{Z}$, with $c'_{i1} = 1$ for $1 \leq i \leq s$.

Note the workload of the generic attack of Subsection 4.4.5 has increased, as one has to guess more key material, namely both t and f. However, just as before we can guess values t', f' for t, f, define

$$\mathfrak{b}_{ij} = \gamma_{ij}(t')^{-1}\alpha_{ij}^{-1} f'(\alpha_{ij})^{-1} t',$$

and arrive at the following.

Theorem 4.11. *Let G be such that $G/\mathcal{Z}$ is abelian. Then $(\mathfrak{b}, (t', t', \ldots, t'), f')$ is an equivalent private key for MST_3^r if and only if $\breve{\mathfrak{b}} : \mathbb{Z}_{|\mathcal{Z}|} \to \mathcal{Z}$ is a bijection whose inverse is efficiently computable.*

This immediately indicates the insecurity of using logarithmic signatures of type $[2, \ldots, 2]$, as picking random values for t', f' will quickly yield a bijection. Furthermore, suppose $\alpha_{ij} = S(e_{ij}, f_{ij})$, where e_{ij} lie in a subspace $\mathcal{Z}_1$ of $\mathcal{Z}$. It seems relevant that to avoid some attacks (see Subsections 6.1 and 6.2 of [89]) one requires $|\mathcal{Z}|/|\mathcal{Z}_1|$ to be 'large'. Thus one can immediately reduce the search for f by considering only homomorphisms $f' : G \to \langle e_{ij} \rangle$. However, the size of $\mathcal{Z}_1$ is unspecified in [89]. From this preliminary analysis, it is not clear what exact security guarantees MST_3^r provides. Moreover the revised scheme is somewhat less natural than the original MST_3 cryptosystem.

Chapter 5

Closing Remarks

Despite over ten years of strong interest in group-based cryptography, a well studied candidate for a secure, fully-specified and efficient cryptosystem is yet to emerge. Can such a platform group be found? We need a candidate group whose elements can be manipulated and stored efficiently, and an associated problem that is hard in the overwhelming majority of instances. There has been a great deal of attention on infinite groups given by a finite presentation (such as braid groups), but it seems that 'random' or 'generic' instances of these protocols lead to particularly simplified atttacks. The case for finite groups is also delicate. Groups with small linear representations are often problematic, as linear algebra can be used as a line of attack; groups with many normal subgroups (such as p-groups) are vulnerable to attacks based on reducing a problem to smaller quotients; groups with permutation representations of low degree are vulnerable to attacks based on the well developed theory of computational permutation group theory. So great care must be taken in the choice of group, and the choice of supposedly hard problem.

More generally, we can move beyond the Ko et al. and Anshel at al. schemes, and ask: is there a secure and efficient key establishment protocol based on group theoretic ideas? There are regular proposals, but the field is still waiting for a proposal that stands up to long-term scrutiny.

Reflecting on the schemes of Chapter 2, it seems unwise to employ matrix groups in such key establishment protocols. However, it remains open to explore other classes of groups for these and similar protocols, for example surface braid groups as suggested by Baumslag et al. for the BCFRX Scheme.

Concerning the Tillich–Zémor hash function, there are several interesting ques-

tions. Can one find secure generators for the scheme in characteristic 2? Can one find an efficient attack in odd characteristic? Can one find a more suitable platform group? FInally we mention some interesting problems that have arisen from analysing cryptosystems based on logarithmic signatures.

Let G be a finite group of order $\prod_{j=1}^{t} p_j^{a_j}$, with p_j distinct primes. Let $\alpha = [A_1, \ldots, A_s]$ be a logarithmic signature for G of type $(r_1, \ldots, r_s)$. The length $l(\alpha) = \sum_{i=1}^{s} r_i$ of α is an efficiency measure: it is the number of elements that must be stored in order to specify a typical logarithmic signature. Naturally one is interested in minimising $l(\alpha)$ to reduce the key size of cryptosystems involving logarithmic signatures. Since $|G| = \prod_{i=1}^{s} r_i$, we must have that $l(\alpha) \geq \sum_{j=1}^{t} a_j p_j$ (this was first observed in [38]). A logarithmic signature for G achieving this bound is called a *minimal logarithmic signature* (MLS).

A natural question to ask is: does every finite group admit a minimal logarithmic signature? If G has a normal subgroup N with $G/N \cong H$ and H and N both have minimal logarithmic signatures then G has a minimal logarithmic signature. Moreover, by considering composition series, it is clear that soluble groups admit MLSs. Hence to answer the question in the affirmative it suffices to consider non-abelian simple groups.

Magliveras [55] used an inductive argument to show the alternating groups have MLSs. González Vasco et al. [37] derived the existence of MLSs for all groups of order less than 175,560, and Holmes [47] proved the sporadic groups J_1, J_2, HS, $\mathrm{M^cL}$, Co_3 and He admit MLSs. Lempken and van Trung [51] used a double coset decomposition method to demonstrate the existence of MLSs for all but 8 simple groups of order at most 10^{10}. Furthermore, they show that their construction cannot be applied to the 8 exceptions found. Several families of classical groups have been shown to admit MLSs by Singhi et al. and Singhi and Singhi [81, 82], notably the finite projective special linear groups and finite projective symplectic groups. The question of whether or not all finite (simple) groups admit a minimal logarithmic signature remains open.

All of the logarithmic signature constructions of an elementary abelian 2-group

described in Chapter 4 are tame. Is this true for all constructions? Are there any qualitatively different methods of constructing logarithmic signatures of an elementary abelian 2-group? One can ask the same questions for elementary abelian p-groups, and abelian groups in general. Finally, what nonabelian groups admit periodic/nonperiodic tilings?

Appendix A

Computer Code

We provide illustrative implementation details of our attack on the BCFRX scheme from Chapter 2. Computations were performed in Magma [59] on an Intel Core 2 Duo 1.66GHz laptop (and hence timings are somewhat slower than those quoted in Chapter 2).

A.1 BCFRX Scheme

Here is the code for our attack on the BCFRX scheme. Recall we are looking to compute the matrix N from Chapter 2, Section 2.1.

```
BCFRX := function(p)
  G := SL(2,p);
  Key := Random(SL(4,p));
  M := Random(SL(4,p));
  P<x1,x2,x3,x4,x5,x6,x7,x8,y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12,y13,
      y14,y15,y16> := PolynomialRing(GF(p),24);
  N := Matrix(P,4,4,[1,0,x1,x2,0,1,x3,x4,x5,x6,1,0,x7,x8,0,1]);
  Ninv := Matrix(P,4,4,[y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12,y13,y14,
      y15,y16]);

  // Let us store the sixteen equations N*Ninv=I_4 in a list S:

  T1 := N*Ninv - ScalarMatrix(4,1);
  S := [];
  for i in [1..4] do
    for j in [1..4] do
      Append(~S,T1[i,j]);
    end for;
```

```
  end for;

  // For l runs of the protocol, store the 8l equations~(2.1.6) in S. For
       a single run of the protocol we have:

  l := 1;
  for k in [1..l] do
    a1 := M^-1 * DiagonalJoin(Matrix(Random(G)),Identity(G)) * M;
    a2 := M^-1 * DiagonalJoin(Matrix(Random(G)),Identity(G)) * M;
    b1 := M^-1 * DiagonalJoin(Identity(G),Matrix(Random(G))) * M;
    b2 := M^-1 * DiagonalJoin(Identity(G),Matrix(Random(G))) * M;
    C := b1 * Key * b2;
    D := a1 * C * a2;
    E := a1 * Key * a2;
    T2 := N * D * Ninv - N * C * Ninv;
    T3 := N * D * Ninv - N * E * Ninv;

    for i in [1..2] do
      for j in [1..2] do
        Append(~S,T2[i,j]);
        Append(~S,T3[i+2,j+2]);
      end for;
    end for;
  end for;

  // Now compute the ideal I generated by S, followed by the (lex ordered
      ) Groebner basis of I:

  I := ideal<P|S>;
  GB := GroebnerBasis(I);
  return GB;
end function;
```

Let's take a look at the Groebner basis for a small prime, say $p = 97$:

```
BCFRX(97);
[
    x1 + 42*y16^5 + 49*y16^4 + 21*y16^3 + 55*y16^2 + 63*y16 + 18,
```

```
    x2 + 71*y16^5 + 89*y16^4 + 87*y16^3 + 9*y16^2 + 15*y16 + 52,
    x3 + 24*y16^5 + 61*y16^4 + 75*y16^3 + 81*y16^2 + 28*y16 + 93,
    x4 + 13*y16^5 + 70*y16^4 + 29*y16^3 + 85*y16^2 + 76*y16 + 50,
    x5 + 26*y16^5 + 50*y16^4 + 40*y16^3 + 32*y16^2 + 69*y16 + 86,
    x6 + 93*y16^5 + 66*y16^4 + 37*y16^3 + 84*y16^2 + 27*y16 + 92,
    x7 + 76*y16^5 + 65*y16^4 + 85*y16^3 + y16^2 + 62*y16 + 3,
    x8 + 78*y16^5 + 12*y16^4 + 79*y16^3 + 31*y16^2 + 92*y16 + 30,
    y1 + 23*y16^5 + 65*y16^4 + 29*y16^3 + 39*y16^2 + 8*y16 + 72,
    y2 + 58*y16^5 + 39*y16^4 + 28*y16^3 + 38*y16^2 + 46*y16 + 7,
    y3 + 70*y16^5 + y16^4 + 25*y16^3 + 64*y16^2 + 64*y16 + 69,
    y4 + 91*y16^5 + 2*y16^4 + 49*y16^3 + 56*y16^2 + 35*y16 + 80,
    y5 + 60*y16^5 + 80*y16^4 + 67*y16^3 + 93*y16^2 + 44*y16 + 33,
    y6 + 92*y16^5 + 84*y16^4 + 89*y16^3 + 25*y16^2 + 95*y16 + 17,
    y7 + 39*y16^5 + 49*y16^4 + 37*y16^3 + 10*y16^2 + 3*y16 + 80,
    y8 + 44*y16^5 + 79*y16^4 + 68*y16^3 + 11*y16^2 + 34*y16 + 16,
    y9 + 79*y16^5 + 75*y16^4 + 95*y16^3 + 52*y16^2 + 93*y16 + 85,
    y10 + 80*y16^5 + 36*y16^4 + 82*y16^3 + 12*y16^2 + 41*y16 + 2,
    y11 + 18*y16^5 + 52*y16^4 + 21*y16^3 + 64*y16^2 + 7*y16 + 89,
    y12 + 33*y16^5 + 84*y16^4 + 7*y16^3 + 86*y16^2 + 39*y16 + 5,
    y13 + 84*y16^5 + 20*y16^4 + 49*y16^3 + 2*y16^2 + 14*y16 + 95,
    y14 + 65*y16^5 + 13*y16^4 + 49*y16^3 + 8*y16^2 + 88*y16 + 66,
    y15 + 20*y16^5 + 87*y16^4 + 81*y16^3 + 25*y16^2 + 12*y16 + 86,
    y16^6 + 17*y16^5 + 11*y16^4 + 48*y16^3 + 71*y16^2 + 79*y16 + 84
]
```

Note the particular form of these equations: once `y16` is known the system is determined. We see that the final basis element is a degree six polynomial in `y16`. Thus computing the zeros of this polynomial yields at most six values for N:

```
Factorisation(GB[24]);
[
    <y16 + 8, 1>,
    <y16 + 15, 1>,
    <y16 + 18, 1>,
    <y16 + 20, 1>,
    <y16 + 71, 1>,
    <y16 + 79, 1>
```

```
]
```

Observing another run of the protocol (executing the `for k in [1..l] do` loop again to add 8 new equations to our system) reveals a unique value for N:

```
GB;
[
    x1 + 25, x2 + 37, x3 + 87, x4 + 53, x5 + 21, x6 + 23, x7 + 80, x8 +
        29, y1 + 76, y2 + 6, y3 + 94, y4 + 26, y5 + 70, y6 + 49, y7 + 96,
        y8 + 46, y9 + 5, y10 + 89, y11 + 10, y12 + 52, y13 + 59, y14 + 58,
         y15 + 22, y16 + 18
]
```

To explore the feasibility of this attack, we return to the $l = 1$ case and run some trials for larger values of p. To test that the Groebner basis elements typically take the form of the above example, we create a function which on input a prime p runs the above attack and verifies that (i) the first 23 basis elements have the form $z_i + f_i(y_{16})$, where $z_i \in \{x_i, y_i\}$ and (ii) the final basis element is a degree six polynomial in y_{16}:

```
BCFRX := function(p)
  ..  // as above
  t := Cputime();
  GB := GroebnerBasis(I);
  t := Cputime(t)
  for i in [1..#GB-1] do
    if Degree(Monomials(GB[i])[1]) ne 1 or Monomials(GB[i])[2] lt y16
        then
      print "The knights who say";
    end if;
  end for;
  if Degree(GB[#GB]) ne 6 or IsUnivariate(GB[#GB],y16) ne true then
    print "Ni";
  end if;
  return t;
end function;
```

For example, running 1000 trials for random 100 bit primes we have:

```
n := 1000;
c := 0;
for m in [1..n] do
  c := c + BCFRX(RandomPrime(100));
end for;
print c/n;
16.735
```

Similarly, for random 200 and 300 bit primes it takes roughly 23 seconds and 33 seconds respectively.

Bibliography

[1] K. S. Abdukhalikov, C. Kim, On the security of the hashing scheme based on SL_2, *Fast Software Encryption - FSE '98* (S. Vaudenay, ed.), Lecture Notes in Computer Science **1372** (Springer-Verlag, 1998) 93–102.

[2] R. Álvarez, L. Tortosa, J. F. Vicent, A. Zamora, Analysis and design of a secure key exchange scheme, *Information Sciences* **179** (12) (2009) 2014–2021.

[3] I. Anshel, M. Anshel, D. Goldfeld, An algebraic method for public-key cryptography, *Math. Res. Lett.* **6** (1999) 287-291.

[4] E. Artin, The theory of braids, *Annals of Math.* **48** (1947) 101–126.

[5] J. S. Birman, V. Gebhardt, J. González-Meneses, Conjugacy in Garside groups I: cycling, powers and rigidity, *Groups Geom. Dynamics* **1** (2007) 221–279.

[6] J. S. Birman, V. Gebhardt, J. González-Meneses, Conjugacy in Garside groups II: structure of the ultra-summit set, *Groups Geom. Dynamics* **2** (2008) 13–61.

[7] J. S. Birman, V. Gebhardt, J. González-Meneses, Conjugacy in Garside groups III: periodic braids, *J. Algebra* **316** (2007) 746–776.

[8] J. S. Birman, K. H. Ko, S. J. Lee, A new approach to the word and conjugacy problems in the braid groups, *Adv. Math.* **139** (1998) 322–353.

[9] G. Baumslag, T. Camps, B. Fine, G. Rosenberger, X. Xu, Designing key transport protocols using combinatorial group theory, Algebraic methods in cryptography: AMS/DMV Joint International Meeting, June 16-19, 2005, Mainz, Germany:

International Workshop on Algebraic Methods in Cryptography, November 17-18, 2005, Bochum, Germany, *Contemporary mathematics* **418** (2006) 35–43.

[10] J.-C. Birget, S. S. Magliveras, W. Wei, Trap doors from subgroup chains and recombinant bilateral transversals, *Proceedings of RECSI VII* (2002) 31–48.

[11] S. R. Blackburn, C. Cid, C. Mullan, Group theory in cryptography, in: C. M. Campbell, M. R. Quick, E. F. Robertson, C. M. Roney-Dougal, G. C. Smith and G. Traustason (editors), *Proceedings of Groups St Andrews 2009 in Bath Volume 1*, Cambridge University Press, 133–149, 2011.

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

[13] S. R. Blackburn, C. Cid, C. Mullan, Cryptanalysis of three matrix-based key establishment protocols. *To appear in J. Math. Crypt*.

[14] S. R. Blackburn, S. Murphy, J. Stern, The cryptanalysis of a public key implementation of Finite Group Mappings, *J. Cryptology* **8** (1995) 157–166.

[15] J.-M. Bohli, M. I. González Vasco, C. Martínez, R. Steinwandt, Weak keys in MST_1, *Des. Cod. Crypt.* **37** (2005) 509–524.

[16] W. W. Boone, Certain simple unsolvable problems in group theory, I–VI, Nederl. Akad. Wetensch Proc. Ser. **A57**,231–237, 492–497 (1954), **58**, 252–256, 571–577 (1955), **60**, 22–27, 227–232 (1957).

[17] V. Canda, T. van Trung, S. S. Magliveras, T. Horvath, Symmetric block ciphers based on group bases, *Selected Areas in Cryptography, SAC 2000* (D.R. Stinson and S.E. Tavares, eds.), Lecture Notes in Computer Science **2012** (Springer-Verlag, Berlin, 2001) 89–105.

[18] K. W. Campbell, M. J. Wiener, DES is not a group, *Advances in Cryptology – CRYPTO '92* (E.F. Brickell, ed.), Lecture Notes in Computer Science **740** (Springer–Verlag, Berlin, 1993) 512–520.

[19] J. H. Cheon, B. Jun, A polynomial time algorithm for the braid Diffie-Hellman conjugacy problem, *Advances in Cryptology – CRYPTO 2003* (D. Boneh, ed.), Lecture Notes in Computer Science **2729** (Springer, Berlin, 2003) 212–225.

[20] D. Charles, E. Goren, K. Lauter, Cryptographic hash functions from expander graphs, *J. Cryptology* **22** (1) (2009) 93–113.

[21] C. Charles, J. Pieprzyk, Attacking the SL_2 hashing scheme, *Advances in Cryptology - ASIACRYPT '94*(J. Pieprzyk, R. Safavi-Naini, eds.), Lecture Notes in Computer Science **917** (Springer-Verlag, 1995) 322–330.

[22] J. J. Climent, F. Ferrández, J. F. Vicent, A. Zamora, A nonlinear elliptic curve cryptosystem based on matrices, *Applied mathematics and computation* **174** (1) (2006) 150–164.

[23] J. J. Climent, E. Gorla, J. Rosenthal, Cryptanalysis of the CFVZ cryptosystem, *Advances in Mathematics of Communications* **1** (1) (2007) 1–11.

[24] G. Cohen, S. Litsyn, A. Vardy, G. Zémor, Tilings of binary spaces, *SIAM Journal on Discrete Mathematics* **9** (1996) 393–412.

[25] D. Coppersmith, The Data Encryption Standard (DES) and its strength against attacks, IBM Research Report RC 18613 (IBM, 1992).

[26] P. Dehornoy, Braid-based cryptography, *Contemp. Math*, **360**, 5–33 (2004).

[27] M. Dehn, Über unendliche diskontinuerlich gruppen, *Math. Ann.* **69**(1911) 116–144.

[28] W. Diffie and M. E. Hellman, New directions in cryptography, *IEEE Trans. Information Theory* **22** (1976) 644–654.

[29] J. B. Fraleigh, A first course in abstract algebra, Pearson Education India, 1971.

[30] D. Garber, Braid group cryptography, available at `http://arxiv.org/abs/0711.3941`.

[31] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, U. Vishne, Probabilistic solutions of equations in the braid group, *Adv. Appl. Math.* **35** (2005) 323–334.

[32] F.A. Garside, The braid group and other groups, *Quart. J. Math. Oxford* **20** (1969) 235–254.

[33] W. Geiselmann, A note on the hash function of Tillich and Zémor, *Cryptography and Coding* (C. Boyd, ed.), Lecture Notes in Computer Science **1025** (Springer-Verlag, 1995) 257–263.

[34] M. I. González Vasco, A. L. P. del Pozo, P.T. Duarte, Cryptanalysis of a key exchange scheme based on block matrices, available at `http://eprint.iacr.org/2009/5532009`.

[35] M. I. González Vasco, D. Hofheinz, C. Martínez, R. Steinwandt, On the security of two public key cryptosystems using non-abelian groups, *Des. Codes Cryptography* **32** (2004) 207–216.

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

[37] M. I. González Vasco, M. Rötteler, R. Steinwandt, On minimal length factorizations of finite groups, *J. Exp. Math.* **12** (2003) 1–12.

[38] M. I. González Vasco, R. Steinwandt, Obstacles in two public-key cryptosystems based on group factorizations, *Tatra Mountains Math. Pub.* **25** (2002) 23–37.

[39] M. I. González Vasco, R. Steinwandt, A reaction attack on a public key cryptosystem based on the word problem, *Applicable Algebra in Engineering, Communication and Computing* **14** (2004) 335–340.

[40] M. Grassl, I. Ilić, S.S. Magliveras, R. Steinwandt, Cryptanalysis of the Tillich-Zémor hash function, *J. Cryptology* (2011) 148–156.

[41] M. Habeeb, D. Kahrobaei, V. Shpilrain, A public key exchange using semidirect products of groups, *Proceedings of SCC 2010* (2010) 137–141, available at `http://scc2010.rhul.ac.uk/program.php`.

[42] G. Higman, Suzuki 2-groups, *Ill. J. Math* **7** (1963) 79–96.

[43] D. Hofheinz, R. Steinwandt, A practical attack on some braid group based cryptographic primitives, *Public Key Cryptography – PKC 2003* (Y.G. Desmedt, ed.), Lecture Notes in Computer Science **2384** (Springer, Berlin, 2002), 176–189.

[44] J. Hughes, A linear algebraic attack on the AAFG1 braid group cryptosystem, Lecture Notes in Computer Science **2384** (2002) 176–189.

[45] J. Hughes, A. Tannenbaum, Length-based attacks for certain group based encryption rewriting systems, available at `http://arxiv.org/PS_cache/cs/pdf/0306/0306032v1.pdf`.

[46] Sang Jin Lee and Eonkyung Lee, Potential weaknesses of the commutator key agreement protocol based on braid groups, *Advances in Cryptology – EUROCRYPT 2002* (L. Knudsen, ed.), Lecture Notes in Computer Science **2332** (Springer, Berlin, 2002) 14–28.

[47] P. E. Holmes, On minimal factorisations of sporadic groups, *J. Exp. Math.* **13** (2004) 435–440.

[48] R. A. Horn, C. R. Johnson, Matrix Analysis, Cambridge University Press, 1990.

[49] K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, J. Kang, C. Park, New public-key cryptosystem using braid group, *Advances in Cryptology - CRYPTO 2000* (M. Bellare, ed.), Lecture Notes in Computer Science **1880** (Springer, Berlin, 2000) 166–183.

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

[51] W. Lempken, T. van Trung, On minimal logarithmic signatures of finite groups, *J. Exp. Math.* **14** (2005) 257–269.

[52] F. Levy-dit-Vehel, L. Perret, On the Wagner–Magyarik cryptosystem, *Coding and Cryptography* (O. Ytrehus, ed.), (Springer, Berlin, 2006) 316–329.

[53] F. Levy-dit-Vehel, L. Perret, Security analysis of word problem-based cryptosystems, *Des. Codes Cryptography* **54** (1) (2010) 29-41.

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

[55] S. S. Magliveras, Secret and public-key cryptosystems from group factorizations, *Tatra Mt. Math. Publ.* **25** (2002) 1–12.

[56] S. S. Magliveras, N. D. Memon, The algebraic properties of cryptosystem PGM, *J. Cryptology* **5** (1992) 167–183.

[57] S. S. Magliveras, D. R. Stinson, T. van Trung, New approaches to designing public key cryptosystems using one-way functions and trap-doors in finite groups, *J. Cryptology* **15** (2002) 167–183.

[58] S.S. Magliveras, P. Svaba, T. van Trung, P. Zajac, On the security of a realization of cryptosystem MST_3, *Tatra Mt. Math. Publ.* **41** (2008) 65-78.

[59] Wieb Bosma, John Cannon, and Catherine Playoust, "The Magma Algebra System I: The User Language", *J. Symbolic Comput.* **24** (1997), 235–265.

[60] R.J. McEliece, A public key cryptosystem based on algebraic coding theory, *DSN Progress Report* **42 - 44** (Jet Propulsion Lab, Pasadena, 1978) 114–116.

[61] A.J. Menezes, P.C. van Oorschot, S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, 1997.

[62] A. J. Menezes, S. A. Vanstone, A note on cyclic groups, finite fields and the discrete logarithm problem, *Applicable Algebra in Engineering, Communication and Computing*, **3** (1992) 67–74.

[63] A.J. Menezes and Y.H. Wu, The discrete logarithm problem in $\mathrm{GL}(n, q)$, *Ars Combinatoria* **47** (1997) 23–32.

[64] C. Mullan, Cryptanalysing variants of Stickel's key agreement protocol, *J. Math. Crypt.* **4** (4) (2011) 365–373.

[65] A. D. Myasnikov, A. Ushakov, Length based attack and braid groups: cryptanalysis of Anshel-Anshel-Goldfeld key exchange protocol, *Public Key Cryptography – PKC 2007* (T. Okamoto, X. Wang, eds.), Lecture Notes in Computer Science **4450** (Springer, Berlin, 2007) 76–88.

[66] A.G. Myasnikov and A. Ushakov, Random subgroups and analysis of the length-based and quotient attacks, *J. Math. Crypt.* **2** (2008) 29–61.

[67] A. Myasnikov, V. Shpilrain, A. Ushakov, *Group-based Cryptography*, Advanced Courses in Mathematics CRM Barcelona (Birkhäuser, Basel, 2008).

[68] R. W. K. Odoni, V. Varadharajan, P. W. Sanders, Public key distribution in matrix rings, *Electronic Letters* **20** (9) (1984) 386–387.

[69] P. S. Novikov, On the algorithmic unsolvability of the word problem in group theory, Trudy Mat. Inst. Steklov **44** (1955) 1–143.

[70] C. Petit, J.-J. Quisquater, Preimages for the Tillich-Zémor hash function, *Selected areas in cryptography – SAC 2010* (A. Biryukov, G. Gong, D. R. Stinson), Lecture Notes in Computer Science **6544** (Springer, 2011) 282–301.

[71] C. Petit, K. Lauter, J.-J. Quisquater, Cayley hashes: a class of efficient graph-based hash functions, available at `http:/www.dice.ucl.ac.be/~petit/files/Cayley.pdf`.

[72] C. Petit, K. Lauter, J. J. Quisquater, Full cryptanalysis of LPS and Morgenstern hash functions, *SCN* (R. Ostrovsky, R. D. Prisco, I. Visconti, eds.), *Lecture Notes in Computer Science* **5229** (Springer, 2008) 263–277.

[73] C. Petit, J. J. Quisquater, J. P. Tillich, G. Zémor, Hard and easy components of collision search in the Zémor-Tillich hash function: new attacks and reduced variants with equivalent security, *Topics in Cryptology - CT-RSA 2009* (M. Fischlin, ed.), Lecture Notes in Computer Science **5473** (Springer-Verlag, 2009) 182–194.

[74] C. Petit, J.-J. Quisquater, N. Veyrat-Charvillon, Efficiency and pseudo-randomness of a variant of Zémor-Tillich hash function, *ICECS2008 - IEEE International Conference on Electronics, Circuits, and Systems* (J. Micallef, ed.), IEEE (2008) 906–909.

[75] M. Qu, S. A. Vanstone, New public-key cryptosystems based on factorizations of finite groups, *AUSCRYPT '92 Preproceedings*.

[76] U. Romanczuk and V. Ustimenko, On the $\mathrm{PSL}_2(q)$, Ramanujan graphs and key exchange protocols, available at `http://aca2010.info/index.php/aca2010/aca2010/paper/viewFile/80/3`.

[77] William A. Stein et al., The Sage Development Team, `http://www.sagemath.org`. Sage Mathematics Software (Version 3.4.1).

[78] William A. Stein et al., The Sage Development Team, `http://www.sagemath.org`. Sage Mathematics Software (Version 4.4).

[79] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, *SIAM J. Computing* **26** (1997) 1484–1509.

[80] V. Shpilrain, Cryptanalysis of Stickel's key exchange scheme, *Proceedings of Computer Science in Russia* **5010** (2008) 283–288.

[81] N. Singhi, N. Singhi, Minimal logarithmic signatures for classical groups, *Des. Codes Cryptography*, Online first, available at `http://www.springerlink.com/content/6wn387r508646463/`.

[82] N. Singhi, N. Singhi, S. S. Magliveras, Minimal logarithmic signatures for finite groups of Lie type, *Des. Codes Cryptography* **55** (2010) 243–260.

[83] R. Sparr and R. Wernsdorf, Group theoretic properties of RIJNDAEL-like ciphers, *Discrete Appl. Math.* **156** (2008) 3139–3149.

[84] M. Sramka, On the security of Stickels key exchange scheme, available at `http://crises-deim.urv.cat/~msramka/pubs/sramka-stickelkesecurity.pdf`.

[85] T. Beth, W. Geiselmann, M. Grassl, R. Steinwandt, Weaknesses in the $\mathrm{SL}_2(\mathbb{F}_{2^n})$ hashing scheme, *Advances in Cryptology - CRYPTO 2000* (M. Bellare, ed.), Lecture Notes in Computer Science **1880** (Springer-Verlag, 2000) 287–299.

[86] E. Stickel, A new public-key cryptosystem in non abelian groups, *Proceedings of the Thirteenth International Conference on Information Technology and Applications* (2005) 426–430.

[87] P. Svaba, T. van Trung, On generation of random covers for finite groups, available at `http://www.exp-math.uni-essen.de/preprints/RanCoversTran.pdf`.

[88] D. R. Stinson, Cryptography: Theory and Practice, Third Edition (Chapman & Hall, Boca Raton, 2005).

[89] P. Svaba, T. van Trung, Public key cryptosystem MST_3: cryptanalysis and realization, *J. Math. Crypt.* **4** (2010) 271–315.

[90] U. Tamm, Group factorizations and information theory, *Information Theory and Applications Workshop* (2007) 384–387.

[91] J.P. Tillich, G. Zémor, Group-theoretic hash functions, *Algebraic Coding, First French-Israeli Workshop* (G. D. Cohen, S. Litsyn, A. Lobstein, G. Zémor, eds.), Lecture Notes in Computer Science **781** (Springer-Verlag, 1994) 90-110.

[92] J.P. Tillich, G. Zémor, Hashing with SL_2. *Advances in Cryptology - CRYPTO '94* (Y. Desmedt, ed.), Lecture Notes in Computer Science **839** (Springer-Verlag, 1991) 508–511.

[93] J.P. Tillich, G. Zémor, Collisions for the LPS expander graph hash function, *Advances in cryptology – EUROCRYPT 2008* (N. P. Smart, ed.), Lecture Notes in Computer Science **4965** (Springer, 2008) 254–269.

[94] N. R. Wagner, M. R. Magyarik, A public key cryptosystem based on the word problem, in *Advances in Cryptology – CRYPTO '84* (G.R. Blakley, D. Chaum, eds.), Lecture Notes in Computer Science **196** (Springer, Berlin, 1985) 19–36.

[95] R. Wernsdorf, The one-round functions of the DES generate the alternating group, *Advances in Cryptology – EUROCRYPT 1992* (R.A. Rueppel, ed.), Lecture Notes in Computer Science **658** (Springer–Verlag, Berlin, 1993) 99–112.

[96] R. Wernsdorf, The round functions of RIJNDAEL generate the alternating group *FSE* (2002) 143–148.

[97] G. Zémor, Hash functions and graphs with large girths, *Advances in Cryptology – EUROCRYPT '91* (D. W. Davies, ed.), *Lecture Notes in Computer Science* **547** (Springer-Verlag, 1991) 508–511.

[98] G. Zémor, Hash functions and Cayley graphs, *Des. Codes Cryptography*, **4**(4) (1994) 381–394.