NON-COMMUTATIVE CRYPTOGRAPHY: DIFFIE-HELLMAN AND CCA SECURE CRYPTOSYSTEMS USING MATRICES OVER GROUP RINGS AND DIGITAL SIGNATURES

by

Charalambos M. Koupparis

A dissertation submitted to the Graduate Faculty in Mathematics in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2012

This manuscript has been read and accepted for the Graduate Faculty in Mathematics in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

Delaram Kahrobaei

Date | Chair of Examining Committee

Linda Keen

Date | Executive Officer

Delaram Kahrobaei

Vladimir Shpilrain

Melvyn Nathanson

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

NON-COMMUTATIVE CRYPTOGRAPHY: DIFFIE HELLMAN KEY EXCHANGE AND CCA SECURE CRYPTOSYSTEMS USING MATRICES OVER GROUPS RINGS AND DIGITAL SIGNATURES

By: CHARALAMBOS M. KOUPPARIS

Advisor: Delaram Kahrobaei

As computing speed has been following Moore's law without any inclination of tapering out, the need for ever more secure cryptographic protocols is becoming more and more relevant. During the past one and a half decades the field of non-commutative (or non-abelian) group based cryptography has seen a surge in interest.

Through this work we will present the classical Diffie-Hellman public key exchange protocol (DH PKE) [8] and discuss two important notions related to it, the Computational Diffie-Hellman assumption and the Decision Diffie-Hellman assumption [3]. We then proceed to look at a new platform group based on matrices over group rings and present work done by myself in collaboration with Delaram Kahrobaei and Vladimir Shpilrain, (the work can be found in [7]). We discuss the viability of the new platform group and point out its benefits.

Additionally, I in collaboration with Delaram Kahrobaei and

Vladimir Shpilrain propose to use the new platform group in the Cramer-Shoup cryptosystem. We demonstrate how one can implement the system using our platform and prove that the system is still CCA-2 secure. Our work is presented in [6].

Finally, we discuss the notion of classical digital signatures following the work of Goldwasser and Bellare [14] and Schnorr [30]. We then discuss some non-commutative digital signatures including those proposed by Ko, Choi, Cho and Lee [20], Wang and Hu [35] Anjaneyulu, Reddy and Reddy [1] and Chaum and van Antwerpen [4]. We conclude by presenting work done my myself in conjunction with Delaram Kahrobaei [5] which discusses a new non-commutative digital signature. We propose using groups for which the Conjugacy Search Problem is hard (see [25]), or any group which is secure against length based attacks, such as polycyclic groups (see [12]), as the platform for this signature.

Acknowledgements

I would like to express my sincerest thanks to my thesis advisor, Delaram Kahrobaei, for her continuous support and encouragement. Delaram's enthusiasm and guidance helped keep me motivated and focused along the way. Both Delaram and my second advisor, Vladimir Shpilrain, have provided insightful commentaries, suggestions and discussions throughout my work. I am thankful for their flexibility and grateful to have been afforded the chance to collaborate with them. Vladimir provided valuable guidance and insight through our work together. I would also like to thank our executive officer, Joseph Dodziuk for his advice and support throughout my graduate career. Finally, I would like to thank Melvyn Nathanson for serving on my defense committee.

I acknowledge the support of the Office of Naval Research through a grant by Delaram Kahrobaei and Vladimir Shpilrain, as well as PSC CUNY research grant through Delaram Kahrobaei.

New York C.M.K.
2012

To my wife, Angie Koupparis,

and my parents, Michael and Helen Koupparis,

for their continuous love and unyielding support.

Preface

Non-commutative (or non-abelian) cryptography is a relatively nascent field in mathematics. It however has gained traction since its beginnings some thirteen years ago with the publishing of non-commutative cryptographic protocols by Anshel-Anshel-Goldfeld [2]. Further impetus to research in this field was motivated by research of Ko-Lee et. al. [21] utilizing specific non-abelian groups (braid groups). This field has drawn research from mathematics and computer science, both fields which actively deal with cryptography, and has produced new results for cryptographic systems utilizing non-commutative groups. The primary concern of any cryptographic protocol is its security. This is not only guaranteed by the protocol employed but by the group (platform group) the protocol is executed over. In this work we will investigate various cryptographic protocols over a new (and old) non-commutative platform groups.

We start in **Chapter 1** with a brief introduction to the classical Diffie-Hellman key exchange protocol. We discuss the various

platform groups for which it may be implemented over and give some examples. Lastly we will discuss the various methodologies of attacking the Diffie-Hellman protocol and look at the benefits and drawbacks of each.

In **Chapter 2** we discuss the two important notions related to the Diffie-Hellman protocol, namely the Computational Diffie-Hellman assumption and the Decision Diffie-Hellman assumption. They play a crucial role in the security of the Diffie-Hellman public key exchange.

We discuss group rings in **Chapter 3**, (we refer the reader to [24] and [10] for a more comprehensive list of proposed platform groups.). Group rings will serve as a building block for our platform groups. Furthermore, we briefly talk about units in groups rings and provide some methods of constructing such elements in a given group ring. Units of group rings arise in discussions of the Cramer-Shoup cryptosystem and digital signatures.

We proceed in **Chapter 4** to describe a new platform for the Diffie-Hellman public key exchange protocol. We illustrate how the protocol can be adapted, what parameters must be set and discuss the advantages of our new platform. We present experimental results to verify the some security assumptions and discuss methods of attacking this protocol utilizing our groups. In particular we demonstrate how this setup is more secure than the

standard approach and methodologies employed today.

Next in **Chapter 5** we describe how the previously proposed platform groups can be used in the Cramer Shoup cryptosystem. We show that working with a subset of the original groups that we can still carry out this protocol. We the continue to discuss and prove security of this scheme against adaptive chosen ciphertext attacks. Finally we describe the parameters needed in the setup and a method for generating the needed elements.

Digital signatures are an important development in modern cryptography and are discussed in **Chapter 6** . We look at some classical digital signatures including the more popular RSA and El Gammal digital signatures. After working though commutative digital signatures we discuss some of the newer non-commutative digital signatures.

Lastly, in **Chapter 7**, we describe a new digital signature algorithm that relies on non-commutative groups. We work through the various notions of security for digital signatures and show that this new signature is sound and complete, secure against data forging and existential forgery. We then discuss some of the possible platform groups which can be used for this digital signature.

Table of Contents

List of Figures

List of Tables

Chapter 1

Classical Diffie-Hellman key exchange

1.1 Introduction

Public key cryptography arose out of the deficiencies that surfaced during the early days of cryptography. Namely, when two parties wished to agree upon a secret key which would then be used to securely exchange information they would have to use a secure, but non-cryptographic, method of establishing the secret key; this is known as key exchange. Essentially, key exchange (public or private) is a method in cryptography which allows users to exchange encrypted messages by fixing a secret key and following a simple protocol which ensures that only the legitimate parties involved in the protocol exchange information.

The initial methods of key exchange involved sending the key

via a trusted courier for example or even setting the key via a face-to-face meeting. Clearly this presents a number of significant practical limitations and difficulties to securely exchanging secret information. Public key cryptography arose as a natural answer to the then established methods of secure communication. These new methods allowed parties to exchange information securely, while communicating over a public channel, without actually having to agree on upon a secret key a priori.

1.2 Diffie-Hellman Protocol

The Diffie-Hellman public key exchange protocol is an example of one of the earliest public key exchange protocols implemented in the field of cryptography. This protocol allows information to be exchanged securely and secretly between two parties who do not necessarily have knowledge of each other, in such a way so that no one else can easily obtain access to the transferred "secrets."

The Diffie-Hellman key exchange was introduced in 1976 by Whitfield Diffie and Martin Hellman [8]. This protocol allowed users to establish secure channels on which to exchange information (or secret keys), regardless of whether or not an adversary is listening on the same communication channel. We note however

that the DH protocol as presented hadn't addressed the problem of verifying the actual identity of the other person involved in the key exchange.

The protocol is based on number theoretic properties and was initially implemented using the multiplicative group of integers modulo p ($\mathbb{Z}_p$), where p is a prime number. This is one of the simplest implementations and for it one also needs an element $g \in \mathbb{Z}_p$, which is a primitive root modulo p. If two parties, Alice and Bob, wish to use the DH protocol to establish a secret key they perform the following algorithm.

1. Alice first determines the group $\mathbb{Z}_p$ and chooses $g \in \mathbb{Z}_p$, a primitive root mod p.

2. Alice then chooses an integer $a \in \mathbb{Z}$ and computes $A := g^a \mod p$. She then makes publicly available the tuple, (g, p, A).

3. Bob upon receiving the public information from Alice chooses $b \in \mathbb{Z}$ and computes and publishes $B := g^b \mod p$.

4. Alice computes $K_A := B^a \mod p$.

5. Bob computes $K_B := A^b \mod p$.

6. Now since $(g^a)^b = (g^b)^a$ the shared, but secret key, K, which

Alice an Bob can use to encrypt messages is known to both parties as $K := K_A = K_B$.

Example 1.2.1. *Suppose Alice chooses the prime 647 and 37 as a primitive root mod 647.*

- *Alice then chooses* $a = 21$*, and computes* $37^{21} \mod 647 = 247$*.*
- *Alice publishes* $(p, g, A) = (647, 37, 247)$*.*
- *Bob picks* $b = 53$ *and computes* $37^{53} \mod 647 = 222$*. Bob publishes* $(B) = (222)$*.*
- *Alice computes* $222^{21} \mod 647 = 193$*.*
- *Bob computes* $247^{53} \mod 647 = 193$*.*

Alice and Bob are now both in possession of the same secret key, $K = 193$*.*

1.3 Security of Diffie-Hellman protocol

Provided that the group G and the element g are chosen appropriately the protocol is considered secure, see e.g. [22] for details. One method that an eavesdropper Eve may use to figure out the secret key is to solve the *Discrete Logarithm Problem.*

Definition 1.3.1. Given a multiplicative group G of order p, the *Discrete Logarithm Problem* (DLP) for G is defined as follows. Given $g, y \in G$ determine an integer x such that $g^x = y \mod p$.

Of course in the example previously given the DLP is easily solved since the numbers we used were relatively small. In practice the prime p used for the DH protocol is 1000 bits in size, that is, of the order of $2^{1000} \sim 10^{300}$. This then allows choosing much larger integers a, b as well. Hence a brute force attack, simply cycling through all powers of g to determine a or b, is not a simple task.

One may think that Alice and Bob face the same problem of having to compute a large number of multiplications, thus facing the same problem as the eavesdropper. However, as Alice and Bob are each in possession of their secret keys a and b respectively, they can simply use a "square and multiply" algorithm that requires $O(\log_2 n)$ multiplications to compute g^n. For example $g^27 = (((g^2)^2)^2)^2 \cdot ((g^2)^2)^2 \cdot g^2 \cdot g$.

Currently there is no known efficient algorithm of quickly solving the DLP. However, while it may seem that Alice and Bob's secret key is safe assuming Eve cannot solve the DLP, this is not entirely correct. Eve in fact knows the values of g^a and g^b, and

her problem is to determine g^{ab}. Hence, while solving DLP is infeasible, this is not the precise problem Eve needs to solve. The security of the DH protocol relies on the difficulty of the following, potentially easier, problem.

Definition 1.3.2. Given a prime number p and an integer g. The *Diffie-Hellman Problem* (DHP) is the task of computing the value of $g^{ab} \mod p$, given the known values of $g^a \mod p$ and $g^b \mod p$.

Clearly if one is able to solve the DLP one can solve the DHP. We point out however, that no one has yet been able to determine if solving the DHP is equivalent to solving the DLP.

1.4 Platform Groups

The original Diffie-Hellman protocol was proposed using the multiplicative group of integers modulo p, $\mathbb{Z}_p^*$, where p is a prime. Since then various other groups have been proposed as a platform for the DH protocol. Key elements of a platform group include its size, security, and the amount of resources expended to store elements of and compute with the group.

Since groups derived from integers inherit many number theoretic properties and are easily understood and investigated, these were some of the original (and current) platforms used for the DH

protocol. A more recent development in public key cryptography is to use elliptic curves as platform groups.

An elliptic curve over a finite field $\mathbb{F}_q$, $E(\mathbb{F}_q)$, is defined in terms of the solutions to an equation over $\mathbb{F}_q$. Different forms of equations may be used to define elliptic curves over $\mathbb{F}_q$ depending on the choice of q.

Example 1.4.1. *Let $\mathbb{F}_p$ be a prime finite field, where p is an odd prime. Let $a, b \in \mathbb{F}_p$ satisfy $4a^3 + 27b^2 \neq 0 \pmod p$. Then an elliptic curve $E(\mathbb{F}_p)$ over $\mathbb{F}_p$ defined by the parameters $a, b \in \mathbb{F}_p$ consists of the set of solutions (points) $P = (x, y)$, where $x, y \in \mathbb{F}_p$ to the equation:*

$$y^2 = x^3 + ax + b \pmod p$$

along with the extra point O called the point at infinity.

One can define an elliptic curve Diffie-Hellman public key exchange protocol using elliptic curves over finite fields along with their inherited multiplication.

1.5 Methods of Attacking the Diffie-Hellman protocol

The various choices of platform groups for the DH protocol make the DLP and DHP problems hard to efficiently solve. However, while the DLP and DHP problems are intractable by themselves, there do exist algorithms that take advantage of the underlying group structure and allow attacks to be carried out on the DH scheme, by attacking the DLP problem.

1.5.1 Baby-step Giant-step Algorithm

One method of attacking the classical discrete logarithm problem which is due to Shanks [31], is the baby-step giant-step algorithm. The algorithm computes discrete logarithms in a group of order q in $O\left(\sqrt{q}\ \text{polylog}(q)\right)$ time, where polylog(q) is $O((\log(q))^c)$ for some constant c. The algorithm proceeds as follows:

This algorithm basically relies on a space-time tradeoff. The order of the algorithm and the space complexity is roughly $O(\sqrt{n})$ (versus $O(n)$ for a brute force attack), while the algorithm requires $O(s)$ memory. Hence one can choose s as needed to optimize the

Baby-step giant-step algorithm

Input: g, $y \in G$, $n = |G|$
Output: $x \in \mathbb{Z}, \ni g^x = y \mod n$
Set $s := \lceil \sqrt{n} \rceil$
Set $t := \lceil n/s \rceil$
Set $h = y$
for $j = 0$ to s
    Compute $g_j = g^j$ and store (j, g_j)
for $i = 0$ to t
    Check if h is the second component g_j of a pair, (j, g_j), previously stored
    If so, return i$s + j$
    Else set $h = hg^{-s}$

Table 1.1: Baby-step giant-step algorithm

algorithm for either speed or space requirements. Baby-step giant-step is mostly used for prime order groups, but works for every finite cyclic group. Finally, it is not necessary to explicitly know the order of the group, an upper bound to the order is also sufficient.

1.5.2 Pohlig-Hellman Algorithm

When the order of a group is not prime, the algorithm introduced by Pohlig and Helman [28] is more efficient than the baby-step giant-step algorithm. This algorithm leverages the composite order of the group by breaking down the original DLP into smaller subproblems by use of the generalized Chinese remainder problem.

Specifically, let the order of an element $g \in G$ be q. For the Diffie-Hellman scheme we would like to find an x such that $g^x = y$. Suppose additionally we know a factorization

$$q = \prod_{i=1}^{n} q_i,$$

where the q_i are relatively prime. Then we have that

$$\left(g^{q/q_i}\right)^x = (g^x)^{q/q_i} = y^{q/q_i}, \text{ for } i = 1, ..., n.$$

Now by the Chinese Remainder theorem we can write

$$\mathbb{Z}_q \cong \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_n}$$

and we are left to solve the n instances of the discrete logarithm problem in the smaller groups, i.e. if we define $g_i = g^{q/q_i}$, we are looking to find the solutions $\{x_i\}_{i=1}^n$ for which $g_i^{x_i} = y^{q/q_i} = g^x$.

The complexity of this algorithm in the worst case, i.e. when n is prime, is $O(\sqrt{n})$, however, the algorithm is more efficient when the order is smooth. In particular, if $n = \prod_i p_i^{e_i}$ is the prime factorization of n, then the complexity can be stated as $O(\sum_i e_i(\log n + \sqrt{p_i}))$ ([22]).

1.5.3 Pollard's Rho Algorithm

A final approach which should be discussed for attacking the discrete logarithm problem is Pollard's rho algorithm, [29]. The inputs to this algorithm are the group elements g and y, and the output is an integer x such that $g^x = y$. The algorithm first looks for an orbit, which has the general form $g^a y^b = g^c y^d$, for a, b, c and $d \in \mathbb{N}$. This is usually achieved by using Floyd's cycle-finding algorithm. So long as $b \neq d$ one can take the logarithm with base g to determine n:

$$\begin{aligned} g^a y^b &= g^c y^d \\ \Rightarrow a + b \log_g y &= c + d \log_g y \\ \Rightarrow \frac{a-c}{d-b} &= \log_g y \\ \Rightarrow g^{\frac{a-c}{d-b}} &= y \end{aligned}$$

The running time of this algorithm is $O(\sqrt{(}p)$, where p is the largest factor of n, the order of the group.

Chapter 2

Computational Diffie-Hellman and Decision Diffie-Hellman

2.1 Introduction

As most cryptographic protocols are probabilistic in nature, either in generation of the underlying groups, parameters or secret (and hence shared) keys, we will further discuss what notions of security there are for such protocols. In this chapter we will discuss 2 concepts following Boneh's exposition [4], namely the *Computational Diffie- Hellman* (CDH) and the *Decision Diffie-Hellman* assumptions. These assumptions are what help guarantee the security of probabilistic cryptographic protocols that rely on the Diffie-Hellman algorithm.

Loosely speaking, satisfying the CDH assumption implies that there does not exist an efficient algorithm for breaking the DH

problem. I.e. if we define the Diffie-Hellman function as

$$DH_g(g^a, g^b) = g^{ab},$$

then the claim is that there is no efficient algorithm which computes the function $DH_g(x, y)$ in a group G. While the DDH assumption implies that the information once can extract from (g^a, g^b) is minimal. To be precise, the DDH assumption implies that there is no efficient algorithm which can distinguish between the two distributions generated by the two triples (g^a, g^b, g^{ab}) and (g^a, g^b, g^c), the first being a valid DH triple, the latter being a random triple.

2.2 Definitions

We will proceed to define the Decision Diffie Hellman and Computational Diffie Hellman algorithms in this section. We should note that DDH assumption is more stringent than CDH assumption. It can be shown that even if a group satisfies the CDH assumption that while the eavesdropper, Eve, may not be able to recover the shared secret key, it is still feasible for her to predict 80% of the bits of g^{ab} with non-negligible probability. It has been noted that while in some groups the CDH assumption is believed to be trivially true, while the DDH assumption is trivially false. Hence

most cryptographic systems rely on the DDH assumption and not the CDH assumption.

Definition 2.2.1. A *group family* $\mathbb{G}$ is a set of groups $\mathbb{G} = \{G_{\mathfrak{p}}\}$ (Boneh restricts to only cyclic groups), where $\mathfrak{p}$ ranges over an infinite index set. We denote by $|\mathfrak{p}|$ the size of the binary representation of $\mathfrak{p}$. We assume there is a polynomial time algorithm (in $\mathfrak{p}$ which given $\mathfrak{p}$ and two elements of $G_{\mathfrak{p}}$ outputs their sum.

Definition 2.2.2. An *Instance Generator*, $\mathcal{IG}$, for $\mathbb{G}$ is a randomized algorithm that given an integer n, runs in polynomial time in n and outputs some random index $\mathfrak{p}$ and a generator g of $G_{\mathfrak{p}}$. Note that for each n, the Instance Generator induces a distribution on the set of indices $\mathfrak{p}$.

We provide some simple examples of group families:

1. Let $p = 2p_1 + 1$ where both p and p_1 are prime. The Q_p be the subgroup of quadratic residues in $\mathbb{Z}_p^*$. This is a cyclic group of prime order. This family of groups is parameterized by p.

2. Given a prime p and $E_{a,b}/\mathbb{F}_p$ be an elliptic curve where $|E_{a,b}|$ is prime. The group is parameterized by (a, b, p).

Definition 2.2.3. A *CDH algorithm* for $\mathbb{G}$ is a probabilistic polynomial time (in $|\mathfrak{p}|$) algorithm satisfying, for some fixed $\alpha > 0$ and

sufficiently large n:

$$\Pr[\mathcal{A}(\mathfrak{p}, g, g^a, g^b) = g^{ab}] > \frac{1}{n^\alpha}$$

where g is a generator of $G_\mathfrak{p}$. The probability is over the random choices of $(\mathfrak{p}, g)$ according to the distribution induced by $\mathcal{IG}(n)$, the random choice of a, b in the range $[1, |G_\mathfrak{p}|]$ and all the random bits used by $\mathcal{A}$. The group family $\mathbb{G}$ satisfies the CDH assumption is there is no CDH algorithm for $\mathbb{G}$.

Definition 2.2.4. A *DDH algorithm* for $\mathbb{G}$ is a probabilistic polynomial time (in $|\mathfrak{p}|$) algorithm satisfying, for some fixed $\alpha > 0$ and sufficiently large n:

$$\left|\Pr[\mathcal{A}(\mathfrak{p}, g, g^a, g^b, g^{ab}) = \text{“}true\text{”}] - \Pr[\mathcal{A}(\mathfrak{p}, g, g^a, g^b, g^c) = \text{“}false\text{”}]\right| > \frac{1}{n^\alpha}$$

where g is a generator of $G_\mathfrak{p}$. The probability is over the random choices of $(\mathfrak{p}, g)$ according to the distribution induced by $\mathcal{IG}(n)$, the random choice of a, b in the range $[1, |G_\mathfrak{p}|]$ and all the random bits used by $\mathcal{A}$. The group family $\mathbb{G}$ satisfies the DDH assumption is there is no DDH algorithm for $\mathbb{G}$.

Note we call the difference between the two probabilities in the definition of DDH the *advantage* of algorithm $\mathcal{A}$. We point out that the definition of DDH aims to capture the difference between the distributions of $(\mathfrak{p}, g, g^a, g^b, g^{ab})$ and $(\mathfrak{p}, g, g^a, g^b, g^c)$, i.e. these distributions are computationally indistinguishable.

2.3 Results concerning DDH

Since we know that CDH is weaker than DDH and for a secure cryptographic protocol one wants to ensure that the DDH assumption holds, we ask what is the weakest assumption that implies DDH? Presently there are no known simple assumptions that imply DDH except for slightly weaker assumption of *perfect*-DDH.

Definition 2.3.1. Let $\mathbb{G}$ be a family of finite cyclic groups. A *perfect-DDH* algorithm $\mathcal{A}$ for $\mathbb{G}$ correctly decides with overwhelming probability whether a given triplet $(x, y, z) \in G_{\mathfrak{p}}^3$ is a proper Diffie-Hellman triplet. To be precise, for fixed $\alpha > 0$ and enough n we have

$$\Pr[\mathcal{A}(\mathfrak{p}, g, g^a, g^b, g^c) = \text{“}true\text{”} | a = bc] > 1 - \frac{1}{n^\alpha}$$
$$\Pr[\mathcal{A}(\mathfrak{p}, g, g^a, g^b, g^c) = \text{“}true\text{”} | a \neq bc] > \frac{1}{n^\alpha}$$

where the probability is over the random choices of $(\mathfrak{p}, g)$ according to the distribution induced by $\mathcal{IG}(n)$, the random choice of a, b, c in the range $[1, |G_{\mathfrak{p}}|]$ and all the random bits used by $\mathcal{A}$. We say that $\mathbb{G}$ satisfies the perfect-DDH assumption is there is no polynomial time algorithm which decides whether $DH_g(x, y) = z$ for most triplets.

It was shown independently by Stadler [33] and then Naor and Reingold [27] that the two assumptions, DDH and perfect-DDH, are equivalent. Boneh provides a slightly stronger result by applying it to groups with only an upper bound to their size.

Theorem 2.3.1. *(Boneh) Let $\mathbb{G}$ be a family of finite cyclic groups of prime order. Let $s(\mathfrak{p})$ be an efficiently computable function such that $|G_{\mathfrak{p}}| \leq s(\mathfrak{p})$ for all $\mathfrak{p}$. The $\mathbb{G}$ satisfies the perfect DDH assumption if and only if it satisfies the perfect-DDH assumption.*

Proof Sketch[Boneh] The fact that the DDH assumption implies perfect-DDH is trivial. We prove the converse. Let $\mathcal{O}$ be a DDH oracle. That is, there exists an $\alpha > 0$ such that for large enough n

$$\left|\Pr[\mathcal{A}(\mathfrak{p}, g, g^a, g^b, g^{ab}) = \text{“}true\text{”}] - \Pr[\mathcal{A}(\mathfrak{p}, g, g^a, g^b, g^c) = \text{“}true\text{”}]\right| > \frac{1}{n^\alpha}$$

The probability is over the random choice of a, b, c in $[1, G_{\mathfrak{p}}|]$, and the random choice of $(\mathfrak{p}, g)$ according to the distribution induced by $\mathcal{IG}(n)$. We construct a probabilistic polynomial time (in $s(\mathfrak{p})$ and $|\mathfrak{p}|$) perfect-DDH algorithm, $\mathcal{A}$, which makes use of the oracle $\mathcal{O}$. Given $\mathfrak{p}, g$ and $x, y, z \in G_{\mathfrak{p}}$, algorithm $\mathcal{A}$ must determine with overwhelming probability whether it is a valid Diffie- Hellman triplet or not. Consider the following statistical experiment: pick random integers u_1, u_2, v in the range $[1, s(\mathfrak{p}^2]$ and construct the

triplet

$$(x', y', z') = (x^v g^{u_1}, y g^{u_2}, z^v y^{u_1} x^{v u_2} g^{u_1 u_2})$$

Case 1. Suppose (x, y, z) is a valid triplet, then $x = g^a, y = g^b, z = g^{ab}$. For some a, b. It follows that (x', y', z') is also a valid triplet. Furthermore, one can show that (x', y', z') is chosen from a distribution which is statistically indistinguishable from the uniform distribution on proper Diffie-Hellman triplets in $G_{\mathfrak{p}}$.

Case 2. Suppose (x', y', z') is not a valid triplet. Then $x = g^a, y = g^b, z = g^{ab+c}$.for some $c \neq 0$. In this case, $x' = g^{a'}, y' = g^{b'}, z' = g^{a'b'} g^{cv}$. Note that since $c \neq 0$ we know that g^c is a generator of $G_{\mathfrak{p}}$. Consequently, the distribution of g^{cv} is statistically indistinguishable from uniform. It is not difficult to show that the distribution on (x', y', z') is statistically indistinguishable from the uniform distribution on $G_{\mathfrak{p}}^3$. □

We note that during subsequent results we will restrict the CDH and DDH assumptions to a group family of only one member as later we will be focusing on single groups, and general results for the group families yet to be investigated.

Chapter 3

Group Rings

3.1 Introduction

The security of a cryptographic protocol lies in both the actual details of the protocol, but also in the choice of platform. Depending on which algebraic structure a protocol is defined over, this offer various benefits or drawbacks to the cryptographic scheme. For our particular protocols we will be interested in matrices defined over group rings. This results in an algebraic structure that is a semigroup and which inherits a lot of the important properties of groups, rings and matrices. We will briefly discuss group rings, provide some examples and some key results concerning group rings.

3.2 Group Rings

Definition 3.2.1. Let G be a group written multiplicatively and let R be any commutative ring with nonzero unity. The *group ring* $R[G]$ is defined to be the set of all formal sums

$$\sum_{g_i \in G} r_i g_i$$

where $r_i \in R$, and all but a finite number of r_i are zero. If F is a field, then $F[G]$ is the *group algebra* of G over F.

Loosely speaking a group ring can be thought of a free module over the group G with scalars from R endowed with the multiplication from the ring. One can naturally define the augmentation map $\epsilon : R[G] \to R$, which sends $g_i \mapsto e$, i.e.

$$\epsilon \left(\sum_{g_i \in G} r_i g_i \right) = \sum_i r_i$$

The sum of two elements in $R[G]$ is defined by

$$\left(\sum_{g_i \in G} a_i g_i \right) + \left(\sum_{g_i \in G} b_i g_i \right) = \sum_{g_i \in G} (a_i + b_i) g_i.$$

Note that $(a_i + b_i) = 0$ for all but a finite number of i, hence the above sum is in $R[G]$. Thus $(R[G], +)$ is an abelian group.

Multiplication of two elements of $R[G]$ is defined via the distributive law and by the use of the multiplications in G and R as

follows:

$$\left(\sum_{g_i \in G} a_i g_i\right)\left(\sum_{g_i \in G} b_i g_i\right) = \sum_{g_i \in G}\left(\sum_{g_j g_k = g_i} a_j b_k\right) g_i.$$

As an example of a group ring, we consider the symmetric group S_5 and the ring $\mathbb{Z}_7$ and form the group ring $\mathbb{Z}_7[S_5]$. We will write the identity element of S_m as e. Sample elements and operations are

$$\begin{aligned}
a &= 5(123) + 2(15)(24) + (153) \\
b &= 3(123) + 4(1453) \\
a + b &= (123) + 2(15)(24) + (153) + 4(1453) \\
ab &= (5(123) + 2(15)(24) + (153))(3(123) + 4(1453)) \\
&= 15(132) + 20(145)(23) + 6(14235) \\
&\quad + 8(124)(35) + 3(12)(35) + 4(1435) \\
&= (132) + 6(145)(23) + 6(14235) + (124)(35) \\
&\quad + 3(12)(35) + 4(1435) \\
ba &= (3(123) + 4(1453))(5(123) + 2(15)(24) + (153)) \\
&= 15(132) + 6(15243) + 3(15)(23) + 20(12)(345) \\
&\quad + 8(13)(254) + 4(1345) \\
&= (132) + 6(15243) + 3(15)(23) + 6(12)(345) + (13)(254) \\
&\quad + 4(1345)
\end{aligned}$$

3.2.1 Units of Group Rings

We will later be discussing units in group rings and hence a brief description of units and their construction is in order. If R is a ring we denote by $\mathcal{U}(R)$ the multiplicative group of units of R,

$$\mathcal{U}(R) = \{r \in R \mid \exists s \in R \ni rs = sr = 1\}$$

Similarly for a group ring, $R[G]$ we denote its group of units as $\mathcal{U}(R[G])$. We denote by $\mathcal{U}_1(R[G])$ the subgroup of units of augmentation 1 (where 1 is the identity element of R) in $\mathcal{U}(R[G])$, namely

$$\mathcal{U}_1(R[G]) = \{\mathcal{U}(R[G]) \mid \epsilon(u) = 1\}$$

For example if our group ring is $\mathbb{Z}[G]$ for some group G then $\epsilon(u) = \pm 1$ and hence

$$\mathcal{U}(\mathbb{Z}[G]) = \pm\mathcal{U}_1(\mathbb{Z}[G])$$

In the same manner for any arbitrary ring R we will have

$$\mathcal{U}(R[G]) = \mathcal{U}(R) \times \mathcal{U}_1(R[G])$$

Some units can be easily constructed as they can be built up from the units found in R and elements of G with finite order. For example if $r \in R$ is an idempotent element, $r^2 = 0$, then $(1-r)(1+r) = 1$ and hence $1 \pm r$ are units in R and by extension

$R[G]$. In a similar fashion any idempotent element of R ($r^k = 0$) is a unit or R and $R[G]$.

If $g \in G$ is an element with finite order, p^n ($g^{p^n} = 0$), where the characteristic of the ring R is p, then $(1-g)^{p^n} = 0$ in $R[G]$, therefore $\alpha = 1-g$ is nilpotent. Hence the elements $1 \pm \alpha$ are units, $1 - \alpha = g$ and is trivial, but $1 + \alpha = 2 - g$ is always nontrivial provided $\mathrm{char}(G) \neq 2$. Note that $g - g^2 = g(1-g)$ is also nilpotent hence $1 \pm (g - g^2)$ are nontrivial units unless $g^2 = 1$, the pattern can be repeated to obtain further units. For more advanced methods of constructing units, or classification of units we refer the reader to [23] (chap. 8).

Chapter 4

A New Platform For The Diffie-Hellman PKE

Although the Diffie-Hellman public key exchange protocol has been around for some time, there is still ongoing research as to how one may improve the protocol. Currently, various veins of research focus on improving and coming up with different platform groups to use in this protocol. In general the choice of the "correct" platform group affords any protocol with an additional layer of security and in many cases may be the deciding factor in the protocol's security and use.

Depending on which group a scheme is implemented and carried out over there are often multiple security parameters which need to be specified and investigated as to their ability to provide more (or less) security. In addition to the parameters that define the platform group, the methods which are used to attack a protocol

often depend on the structure of the group. In this chapter we proceed to define a new platform group (semigroups in our case) for the Diffie-Hellman public key exchange. We then demonstrate its security and discuss its effectiveness as a choice for a platform group, we follow our work in [18].

4.1 Diffie-Hellman public key exchange protocol

Let's recall the original implementation of the Diffie-Hellman public exchange protocol which uses the multiplicative group of integers modulo a prime p as their platform, $\mathbb{Z}_p$. Alice and Bob are in possession of private keys a and b respectively and they can both compute their shared private key g^{ab} from the public information (g, G, g^a, g^b, p). If it not feasible for Eve to compute g^{ab} without solving the Discrete Logarithm Problem 1.3.1 or the Diffie-Hellman Problem 1.3.2.

Depending on the choice of platform group however, these problems vary in difficulty. Hence, one wishes to work with a group which is secure against attacks to these these particular problems. Various groups have been proposed for this algorithm and in the next section we will investigate and porpose a new platform group.

4.2 Platform Group Parameters, Size and Advantages

Although the protocol is generally considered adequate, there is a slight disadvantage to working with $\mathbb{Z}_p$ where p, a and b are chosen to be fairly large. The main drawback to this is that computation with 300-digit numbers (1000-bit binary numbers) is not particularly efficient, both in terms of time and space. Additionally, reducing the result mod p is further computationally intensive. This is one of the main reasons why the Diffie-Hellman key exchange protocol isn't suitable for devices that have limited computational resources. Hence there is a need and ongoing research for other platform groups where the Diffie-Hellman protocol can be carried out more efficiently, in particular one seeks public/private keys of smaller size.

The platform which we propose to use is a semigroup of matrices (of a small size) over a group ring, with the usual matrix multiplication operation. More specifically, we will be working with matrices over the group ring $\mathbb{Z}_n[S_m]$, where $\mathbb{Z}_n$ is the ring of integers modulo n and S_m is the symmetric group of degree m. In order to assert that this semigroup is a plausible choice for

a platform group one needs to address the *Computational Diffie-Hellman* and *Decision Diffie-Hellman* problems, (chapter 2). In addition one must also address certain questions about the structure of the chosen semigroups.

The parameters we suggest (2×2 or 3×3 matrices over $\mathbb{Z}_7[S_5]$) provide for a fairly large keyspace as compared to the original Diffie-Hellman groups ($7^{480} \sim 10^{406}$ for 2×2 matrices and $7^{1080} \sim 10^{913}$ for 3×3 matrices). When using these groups storing a single 2×2 matrix over $\mathbb{Z}_7[S_5]$ requires roughly 1440 bits, and a single 3×3 matrix requires about 3240 bits. Hence, the key size is roughly similar to what is needed in the "classical" Diffie-Hellman protocol where storing an integer of the order of 10^{300} requires around 1000 bits. The matrix storage requirements can be further reduced by about $\frac{1}{7}^{th}$ if we use a storage container that requires no space for elements in the polynomials which are zero.

An additional advantage to this choice of platform group is the multiplication of matrices over $\mathbb{Z}_7[S_5]$. In this setup multiplying elements is relatively easier and potentially faster than multiplying numbers for large p. There is an additional speed advantage since one can precompute the group multiplication table for S_5 (of order 120). Hence, in order to multiply two elements of $\mathbb{Z}_7[S_5]$

there isn't any actual multiplication involved in S_5, there is just rearrangement of a bit string and multiplication in $\mathbb{Z}_{p^\not\phi}$.

Finally, these platforms groups offer an additional advantage from a security standpoint over $\mathbb{Z}_p$. In particular we discuss how "standard " attacks (baby-step giant-step, Pohlig-Hellman, Pollard's rho) of the Diffie-Hellman protocol will not work in our setup.

4.3 Diffie-Hellman Key Exchange Protocol Using Matrices Over $\mathbb{Z}_n[S_m]$

While the symmetric group S_m is relatively small for small m, when we form the group ring $\mathbb{Z}_n[S_m]$, the size of this new structure grows reasonably fast, even for small values of n and m. This is the primary reason for choosing this structure as the underlying platform for the Diffie-Hellman protocol. Furthermore, once we start considering matrices over these group rings the platform's size is even larger. For example for 3×3 matrices over $\mathbb{Z}_7[S_5]$, $M_3\mathbb{Z}_7[S_5]$, the order of this ring is $7^{5!^9} \sim 10^{913}$. Since we're only using matrix multiplication these rings are really semigroups in our case and these serve as the platform for our protocol. The protocol to be carried out by Alice and Bob is essentially the same.

Alice chooses a matrix $M \in M_3\mathbb{Z}_7[S_5]$ and a large positive integer a, computes M^a and publishes (M, M^a) keeping a secret. Bob chooses another large integer b and computes then publishes M^b. Both Alice and Bob can now compute the same shared secret key, namely $(M^a)^b = (M^b)^a$.

As we have previously mentioned multiplication of matrices in this semigroup is very efficient. Of course, in this semigroup, as in any other semigroup, one can use the "square and multiply" algorithm for quickly computing powers of the elements involved.

In order to assess the security of this platform, we need to address the two Diffie-Hellman assumptions about this semigroup. Namely, does our semigroup satisfy the DDH and CDH definitions? We will investigate the stronger DDH assumption in a later section.

Finally, one must also be concerned with the algebraic structure of $M_3(\mathbb{Z}_7[S_5])$. To be precise, we need to make sure that powers of matrices in this semigroup do not fall into an orbit of low order. Essentially what we wish to investigate and avoid is the concern that if Alice chooses a random integer a and a random matrix M, when does $M^n = M^k$? For security purposes this relationship should not hold for $n < k << a$ (similarly for b chosen by Bob).

If this were to happen, then an eavesdropper Eve could first determine n and k, then she would easily be able to find the values c and d, where $1 < c, d < k$, such that $M^a = M^c$ and $M^b = M^d$. Hence Eve can easily compute the shared secret key as

$$M^{ab} = (M^a)^b = (M^c)^b = (M^b)^c = (M^d)^c = M^{cd}.$$

4.4 Experimental Results

While the CDH assumption may only be answered theoretically, we can still investigate the DDH assumption experimentally. We will also look at the low orbit question in this section. To construct the matrix semigroups we implemented the necessary group ring and matrix procedures in C++. We allowed for the choice of which symmetric group to use and which ring $\mathbb{Z}_n$ to use as well. For symmetric group multiplication we imported their Cayley tables. To generate random group ring elements and hence random matrices, we used a standard implementation of a uniform distribution on the coefficients of the polynomials.

We carried out various experiments in $M_k(\mathbb{Z}_n[S_m])$. While the symmetric group can be varied, we propose using S_5. This symmetric group is well researched and is small so it makes computations and analysis tractable. Furthermore, one also retains the benefit

of S_5 in that it only has one normal subgroup, the alternating subgroup A_5 of index 2. Thus any group theoretic attacks, such as using homomorphisms to tease out information about the protocol are restricted on S_5 to the *sign homomorphism* from S_5 to $\mathbb{Z}_2$.

We also naturally implemented a "square and multiply" routine to speed up computations for exponentiation. With this procedure we can easily and quickly compute high powers of random matrices from our matrix semigroups. Some of the results are summarized in table 4.1.

We should point out that the computations were carried out on an Intel Core2 Duo 2.26GHz machine, utilizing only one core, with 4GB of memory. The average time was computed after randomly drawing 250 matrices and measuring the time required for each exponentiation. From a design standpoint no real optimizations were in effect, and we were only using one processor. Thus, we believe that with additional insight and thinking the computational time may be reduced significantly. In particular one may wish to set up the program to utilize all available cores on a machine. There may be further optimizations based on how one stores the matrices and how one implements the various algorithms and procedures involved in the DH protocol.

Table 4.1: Speed of Computation

Matrix Size	$\mathbb{Z}_n$	Exponent	Avg. Time (s)
2×2	2	10^{10}	0.06
2×2	3	10^{10}	0.06
2×2	5	10^{10}	0.06
2×2	7	10^{10}	0.06
2×2	2	10^{100}	0.58
2×2	3	10^{100}	0.58
2×2	5	10^{100}	0.58
2×2	7	10^{100}	0.59
2×2	2	10^{1000}	5.97
2×2	3	10^{1000}	6.11
2×2	5	10^{1000}	5.98
2×2	7	10^{1000}	6.66
3×3	2	10^{10}	0.19
3×3	3	10^{10}	0.20
3×3	5	10^{10}	0.20
3×3	7	10^{10}	0.20
3×3	2	10^{100}	1.95
3×3	3	10^{100}	1.95
3×3	5	10^{100}	1.94
3×3	7	10^{100}	1.94
3×3	2	10^{1000}	20.17
3×3	3	10^{1000}	20.15
3×3	5	10^{1000}	19.72
3×3	7	10^{1000}	19.74

As a comparison for our computational times we refer the reader to the results of ([19]). In the paper, E. Kasper presented an implementation of the DH key exchange protocol over the elliptic curve P-224. Without any optimizations they can carry out 1800 operations per second for the DH protocol. We note that in using P-224 you require approximately 340 operations for a single "exponentiation". Based on this, they would require about 0.2 seconds per DH exponentiation, versus our 0.6 seconds in $M_2(\mathbb{Z}_7[S_5])$.

We note than while working through the experiments involving computing average speeds of exponentiations we noticed that the time it took per exponentiation was independent of the number of non-zero terms in the polynomials we chose as entries for our matrices, it didn't matter if we started with polynomials have many non-zero terms or only a few. An simple intuitive explanation for this results relies on the fact that any symmetric group can be generated by a set of two particular group elements. If we're working with 2×2 matrices there is a good chance that, in choosing the 4 polynomial entries of the matrix, when multiplied together they will invariably generate all elements of the symmetric group. The argument is even stronger for matrices of larger size. Hence with only a few multiplications it is conceivable that we get random

group ring elements of random length mixing through the matrix entries.

A generic random group ring element from $\mathbb{Z}_2[S_5]$ is of the form $\sum_{\sigma_i \in S_5, \alpha_i \in \mathbb{Z}_2} \alpha_i \sigma_i$. Using a simple binomial distribution calculation shows that with a probability around 93% a random element of this group ring will have between 50 and 70 non-zero terms.

4.4.1 Results concerning the Decision Diffie Hellman assumption

While in this section results will only be presented for 2×2 matrices, it is reasonable to assume that these results would also apply and hold for larger sized matrices. In order to test whether or not the DDH assumption holds for these platform groups we need to look at two distributions, one generated by M^{ab} and another generated by M^c. If the DDH assumption holds there these two distributions should be indistinguishable. To setup the experiments we randomly chose a and b from the interval $[10^{22}, 10^{28})$, and a random c in the interval $[10^{44}, 10^{56})$. This setup generates ab that have about the same size as c.

In order to investigate the distributions generated by these parameters we needed to look at each polynomial entry in the final

matrices M^{ab} and M^c. For 500 random choices of M, a, b and c we looked at the frequency distribution of the elements of S_5 in each position of the final matrices.

From these 500 runs we created the Q-Q plots of M^{ab} versus M^c, which tested the similarity of the frequency distributions of the elements of S_5 per entry of the matrices. In the following graphs we used the notation $M = \left(\begin{smallmatrix} a_1 & a_2 \\ a_3 & a_4 \end{smallmatrix}\right)$. Q-Q plots, also known as quantile plots, are a graphical tool used in statistics for comparing quantiles of the cumulative distribution function (cdf) of a distribution F to the corresponding quantiles of the cdf of a distribution G. Q-Q plots are plots of the pairs $(F^{-1}(p), G^{-1}(p))$, where $p \in [0, 1]$. If the Q-Q plot is a straight line, then the two distributions are of the same type, however they may still have different means and standard deviations as described by their slopes and intercepts, see [5] for more details.

We can see the Q-Q plots in figure 4.1 of the distribution of M^{ab} versus M^c. From the figure we can see that the two distributions appear to be identical. Thus there is no information leaked by Alice or Bob about M^{ab} by their respective choices of a and b. This experimentally confirms the validity of the DDH assumption in our situation.

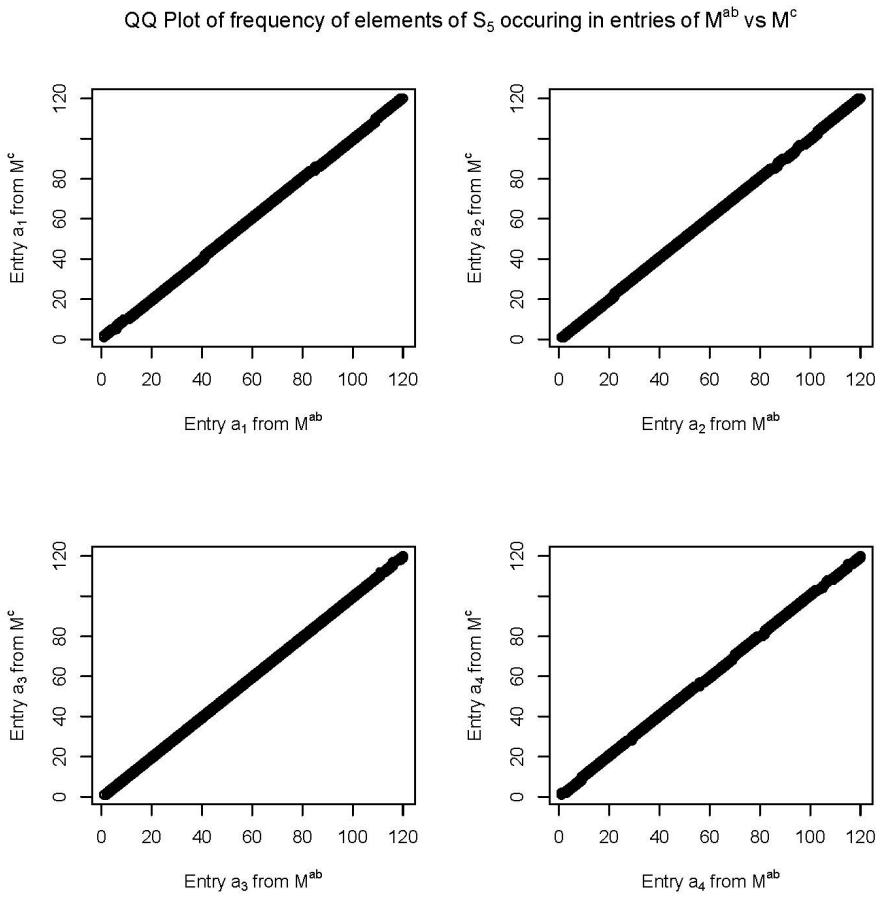

Figure 4.1: DDH results for M^{ab} vs. M^c

We also need to test the possibility of information about a being given away by publishing (M, M^a), for a given choice of M and a. In a similar fashion to the one described above we carried out an experiment to detect if any information leaked about a. In particular we wanted to test if a random matrix N and a

randomly generated key M^a were indistinguishable. We again carried out 500 experiments where we chose random $a \in [10^{44}, 10^{55})$, random M and N and then plotted the resulting Q-Q plots of the frequency distributions of elements of S_5 showing up in the polynomials entries of the matrices M^a versus N, see figure 4.2.

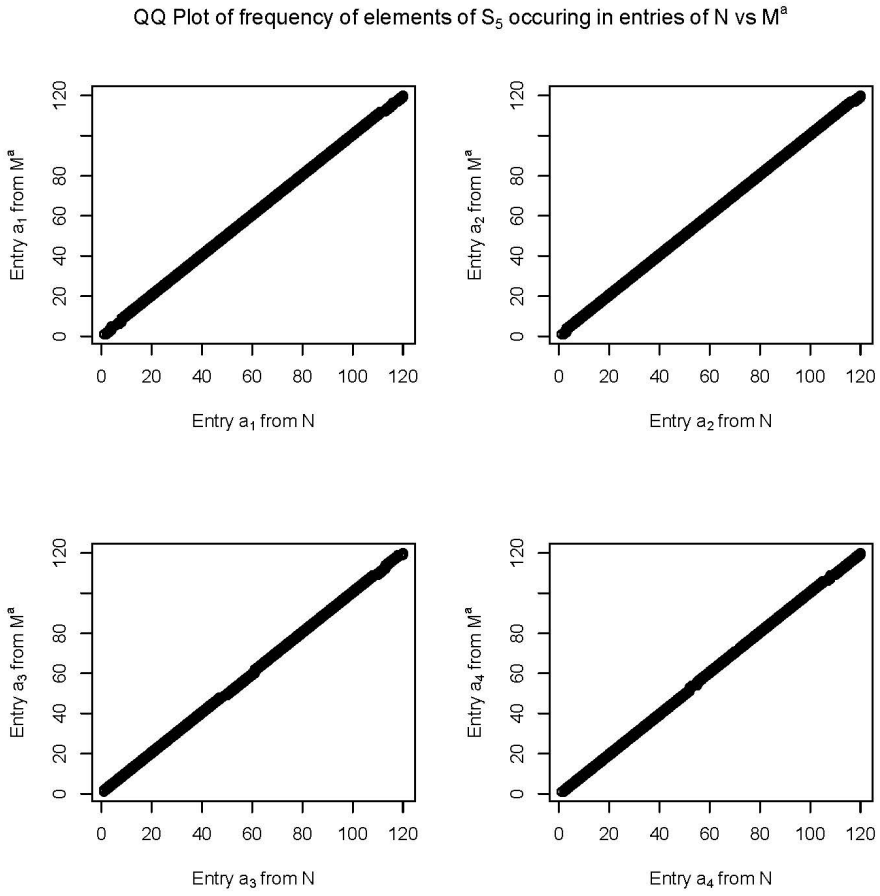

Figure 4.2: Security results for M^{ab} vs. N

From the above plot it is again clear that a random matrix N

and M^a are in fact indistinguishable.

4.4.2 Results concerning low orbits

Experimentally identifying low orbits is a laborious task. One has to compute all powers of the matrix M up to M^a. Each power of the matrix needs to be stored in some form and then this entire series needs to be compared with itself to find duplicates. Currently this is both prohibitive in terms of both time and space.

We propose to cut down on time and space required for these comparisons by only storing the numbers of terms of the polynomials in each entry of the matrix. This way we only end up storing 4 integers per matrix (for 2×2 matrices) instead of the full matrix. Hence we've reduced the problem of checking if two matrices are identical to checking if there are any repeats in a list of 4-dimensional vectors. There is one obvious caveat to this approach, we cannot tell with certainty if we have hit an orbit solely based on 2 vectors being identical, as the matching of vectors does not translate to identical matrices. We can however use this method to determine if there may be a match, i.e. if we find two identical vectors, then we can actually recompute those powers of M that produced those vectors and check for a match

there. In reality we waited for 3 consecutive matches, i.e. the set of vectors for $\{M^i, M^{i+1}, M^{i+2}\}$ must match the set of vectors for $\{M^j, M^{j+1}, M^{j+2}\}$ in order, before we computed the powers of the matrices, as it could be a simple coincidence to have 2 random vectors being equal.

This method allowed us to significantly improve the running time of our algorithm, however we still needed to sacrifice more for time and space considerations. The exponent a forces us to store every vector for M^a hence creating a list of vectors from $[1, a]$. While the list may not be too long to handle, we still had to search through this list and find duplicate vectors. In a simple list of numbers which is a long, we would need $(a-1)!$ comparisons to determine that there wasn't a match in the list. Because we are dealing with vectors we need to match all 4 components of a vectors so in fact the comparisons one must carry out are much more. With this in mind we decided to work with $a = 10,000$ and $a = 1,000,000$.

We ran this experiment with our variations on the ring size and matrix size. For each exponent we run 250 scenarios. The results of these experiments can be seen in table 4.2.

From these first results one can see that increasing the order

Table 4.2: Orbit match search

Group Ring	Exponent of Matrix	Percent Matches
$M_2(\mathbb{Z}_2[S_5])$	10^4	86.4%
$M_2(\mathbb{Z}_3[S_5])$	10^4	6.8%
$M_2(\mathbb{Z}_5[S_5])$	10^4	0.4%
$M_2(\mathbb{Z}_7[S_5])$	10^4	0.0%
$M_3(\mathbb{Z}_2[S_5])$	10^4	26.4%
$M_3(\mathbb{Z}_5[S_5])$	10^6	2.6%
$M_3(\mathbb{Z}_7[S_5])$	10^6	0.0%

of the ring (algebra in this case) decreases collisions the most, by about a factor of 10. Furthermore, increasing the size of the matrices used further reduces the number of matches by about a fourth. We only worked with 10^4 for the smaller matrices as this gave us a quick lower bound for the number of matches when the parameters change. We see furthermore from this table that $\mathbb{Z}_5$ and $\mathbb{Z}_7$ look like good candidates for the the choice of ring. As storing en element of $\mathbb{Z}_5$ or $\mathbb{Z}_7$ requires the same amount of bits we propose using the latter as the ring.

We also propose an additional method of tackling the low orbit question. This involves looking at the "type" of elements in the chosen matrix M. We will denote by e group ring elements with an even number of terms, and by o group ring elements with an odd number of terms. We can rephrase the question of orbits as

such, do there exist $n, k \in \mathbb{N}$ with $n+k << |M_2(\mathbb{Z}_7[S_5])|$ such that

$$M^{n+k} = M^n \tag{4.4.1}$$

$$\Rightarrow M^n(M^k - I) = 0. \tag{4.4.2}$$

Suppose the matrix M we start with is of the form $M \sim \left(\begin{smallmatrix} o & o \\ e & e \end{smallmatrix}\right)$. If we compute powers of this matrix will always end up with the same form as the original matrix, i.e. $M^n \sim \left(\begin{smallmatrix} o & o \\ e & e \end{smallmatrix}\right)$. Hence, observing that the identity matrix $I \sim \left(\begin{smallmatrix} o & e \\ e & o \end{smallmatrix}\right)$ and the zero matrix $O \sim \left(\begin{smallmatrix} e & e \\ e & e \end{smallmatrix}\right)$, the form of (4.4.2) above becomes

$$\begin{aligned} &\left(\begin{smallmatrix} o & o \\ e & e \end{smallmatrix}\right)\left(\left(\begin{smallmatrix} o & o \\ e & e \end{smallmatrix}\right) - \left(\begin{smallmatrix} o & e \\ e & o \end{smallmatrix}\right)\right) = \left(\begin{smallmatrix} e & e \\ e & e \end{smallmatrix}\right) \\ \Rightarrow &\left(\begin{smallmatrix} o & o \\ e & e \end{smallmatrix}\right)\left(\begin{smallmatrix} e & o \\ e & o \end{smallmatrix}\right) = \left(\begin{smallmatrix} e & e \\ e & e \end{smallmatrix}\right) \\ \Rightarrow &\left(\begin{smallmatrix} e & e \\ e & e \end{smallmatrix}\right) = \left(\begin{smallmatrix} e & e \\ e & e \end{smallmatrix}\right). \end{aligned}$$

Hence in choosing our matrix M we may want to exclude matrices which have a form that leads to a valid equation. After a quick analysis of the 16 different matrix forms for 2×2 matrices, there are in fact some possible candidates for which the orbit question has an obvious negative answer, as the matrix on the left hand side is not of the form of the zero matrix.

Some of these remaining matrices however have powers that

cycle through different forms. For example, if $M \sim \left(\begin{smallmatrix} o & o \\ o & e \end{smallmatrix}\right)$, then

$$M \sim \left(\begin{smallmatrix} o & o \\ o & e \end{smallmatrix}\right) \tag{4.4.3}$$

$$M^2 \sim \left(\begin{smallmatrix} e & o \\ o & o \end{smallmatrix}\right) \tag{4.4.4}$$

$$M^3 \sim \left(\begin{smallmatrix} o & e \\ e & o \end{smallmatrix}\right) \tag{4.4.5}$$

$$M^4 \sim \left(\begin{smallmatrix} o & o \\ o & e \end{smallmatrix}\right) \tag{4.4.6}$$

One can indeed work through the details and verify that equation (4.4.2) is not satisfied is M^n and M^k are of the same form as M^2 above. Hence while we can somewhat simplify the question of low orbits, the problem can still be investigated further.

4.5 Attack Methods for Public Key Exchange Protocols

In this section we briefly discuss three of the standard method of attacking the classical discrete logarithm problem and we further discuss why these methods will not work with our choice of platform semigroup.

4.5.1 Baby-step Giant-step Algorithm

A well known method of attacking the classical discrete logarithm problem, due to Shank's, is the baby-step giant-step algorithm,

(see 1.5.1). This algorithm computes discrete logarithms in a group of order q in $O\left(\sqrt{q}\ \text{polylog}(q)\right)$ time, where $\text{polylog}(q)$ is $O((\log(q))^c)$ for some constant c. If adapted to our situation, this algorithm would look as follows.

Baby-step giant-step algorithm

Input: M, $A \in M_3(\mathbb{Z}_7[S_5])$, $n = |M_3(\mathbb{Z}_7[S_5])|$
Output: $x \in \mathbb{N}, \ni M^x = A$

Set $s := \lceil \sqrt{n} \rceil$
Set $t := \lceil n/s \rceil$
for $i = 0$ to s
    compute and store (i, AM^i)
for $j = 0$ to t
    compute $M_j = M^{js}$
    if $M_j = AM^i$, for some i, **return** $js - i$

Table 4.3: Adapted Baby-step giant-step algorithm

As the algorithm stands there are a couple of points which need to be discussed. The first is that we need to provide a good method of storing the matrices the algorithm uses. This could be possible with a hash function, in which case the insertion and lookup times are constant in time. However, as our matrices are fairly complex objects, and we need to take into account the storage requirements of the algorithm.

One may point out that it is unlikely that the order of a chosen matrix M with be the same as the order of a group ring. So one may get away with using a smaller value of $n4$. However, this requires a priori knowledge of the order of M. Since little is known about the structure of this group, the order is hard to find, and we're not guaranteed that such an order even exists. We're essentially back to looking for (low order) orbit collisions as in the previous section. Our best guess in this case would have to be the order of the semigroup.

Storage requirements are also problem in this algorithm. Each polynomial entry in the matrix can be represented by a sequence of 120 three-bit coefficients. Hence we can use a 360 bit string to encode an element of $\mathbb{Z}_7[S_5]$. This means for each matrix we will require 360×4 bits for storage. In the setup of this algorithm we are required to store $\sqrt{|M_3(\mathbb{Z}_7[S_5])|} = \sqrt{7^{540}} \sim 10^{456}$ matrices. This works out to be a huge amount of space that we need to store all this information. Thus it seems that this algorithm is already infeasible in terms of the space needed to store the data. Of course one may optimize the method of storing the matrices however, given the sheer number of matrices one needs to store, this algorithm seems prohibitive.

One approach which has often been suggested to decrease space requirements is to decrease the value of s, hence increasing t. This would have the algorithm running instead of in $O(\sqrt{n})$ time, it would run in $O(n/t)$ time. Every time we reduce by half the storage requirements of this algorithm, we end up doubling its running time. In the end though, regardless of what s and t are chosen to be we will still need to perform $s + t$ group operations in the two loops of the algorithm. Based on this algorithm the number of group operations is minimized when $s = t = \sqrt{n}$. Hence we will still need at least 10^{457} group operations to run this algorithm. In the end it seems that both space and time requirements make this algorithm unsuitable for attacking the discrete logarithm for our semigroups.

4.5.2 Pohlig-Hellman Algorithm

Another classical algorithm for attacking the discrete logarithm problem is the Pohlig-Hellman algorithm, (see 1.5.2). This algorithm relies on the order of the group element and the generalized Chinese Remainder theorem to break the problem into smaller subproblems.

The problem with this algorithm for our setup is that the order

of a matrix in $M_3(\mathbb{Z}_7[S_5])$ does not relate to the size of the actual semigroup. In fact, as this is a semigroup the number of invertible elements is assumed to be very small. Additionally, since the order of this semigroup is 7^{1080} the Chinese Remainder theorem, were it to apply, doesn't really help in breaking this problem into smaller parts. If one does manage though to break the problem into smaller subproblems, we still need to solve the discrete logarithm problem for our semigroup, which so far as we know can only be done via brute force.

4.5.3 Pollard's Rho Algorithm

Finally, the last classical algorithm which can be used to attack the DH scheme is Pollard's rho algorithm, (see 1.5.3). We recall that the inputs to this algorithm are the group elements M and N, and the output is an integer n such that $M^n = N$. The algorithm stops when it has reached an orbit of form $M^a N^b = M^c N^d$, for a, b, c and $d \in \mathbb{N}$. It then takes logarithms with base M to determine $n = \frac{a-c}{d-b}$.

The problem here is that in applying Floyd's cycle-finding algorithm in Pollard's rho attack, the knowledge of the order of the

cyclic group generated by M is essential. However, in our situation, not only is the order of a random matrix M unknown, but more importantly, since a random matrix M is not going to be invertible with overwhelming probability, other considerations are not applicable, orders don't make sense (rather orbits do). Finally, even if one were able to find such a match, compute logarithms in our group is still a brute force attack. Hence it appears that with not much difficulty one can produce public and shared keys that cannot be attacked using this algorithm.

4.5.4 Quantum Algorithms

It is a well known problem that current cryptographic protocols are vulnerable to quantum algorithm attacks. The Diffie-Hellman problem in particular is susceptible to attacks based on Shor's algorithm. Shor's algorithm allows one to rephrase the discrete logarithm problem as a hidden subgroup problem (HSP) and then leverage the existing quantum algorithms developed for the HSP to recover the secret exponents.

We propose that our protocol is even secure against such quantum algorithm attacks.. We know that the HSP relies on the existence of a function $f : G \to S$ such that f is constanct on cosets

of the unknown subgroup $H \leq G$ and also takes on distinct values for each coset. In our setup we need to recast the the discrete logarithm problem as a HSP, as such we define $f : \mathbb{Z}_N \times \mathbb{Z}_N \to G$, such that $f(a, b) = g^a x^b$, where $a, b \in \mathbb{Z}_N$, $g, x \in G$, $g^\alpha = x$ and $|g| = N$. One then rewrites this as $f(a, b) = g^{a+b*log_g x}$, and hence f is constant on the sets $L_c := \{(a, b)|a + b \log_g x = c\}$.

This means that in our instance, we are looking for the following hidden subgroup

$$H := L_0 = \{(0, 0), (\log_g x, -1), (2 \log_g x, -2), \cdots, (N \log_g x, -N)\}.$$

Now in order to apply the HSP algorithms one would need to know the order of a matrix. However, this isn't known a priori and it's also the case that invertible matrices are sparse in our setup and we furthermore aim to avoid using them. Thus is appears that in our our setup the function f is ill-defined.

Finally, for a given random non-invertible matrix it is highly improbable that the function f will be distinct on cosets of the subgroup H or even constant on the different cosets. To demonstrate this assume that M is a non-invertible matrix. Then we know that M will either end up in an orbit or will become the zero matrix. If M enters an orbit, let's use an example where $M^9 = M^{15}$ and the secret exponent we are seeking is $a = 12$. The

subgroup we are trying to identify then is

$$H = \{(0,0), (12,-1), (24,-2), (36,-3), \cdots\}.$$

Based on our hypothetical setup we obtain that $(36,-3) \sim (18,-3)$, but $(18,-3) \notin H$, for if it were then $(36,-3)-(18,-3) = (18,0) \in H$, which is a contradiction. On the other hand, assume that M ends up as the zero matrix, say $M^{20} = 0$ and again $a = 12$. If this happens then f is no longer constant on the subgroup H as $0 = f(24,-2) \neq f(12,-1) = I$.

4.6 Conclusions

In summary, it appears that our group provides simple and tractable keys. It is easy to compute with them and the group satisfies numerous cryptographic assumptions needed to prove security of the protocol carried out over this group. We have shown that traditional methods of attacking the DH protocol do not apply to our scheme and that even the current quantum algorithms (which are not yet widely deployed or used) are not able to provide much traction either.

We end with the note that the platform semigroup can be further enhanced (from a security standpoint) by changing the underlying group. This is achieved by instead of using the full symmetric group, switching instead to the alternating subgroup A_5 of S_5. This effectively bars all group theoretic attacks. The new choice of platform group obviously decreases the overall semigroup size, but one can counter this by either increasing the size of the ring, or the size of the matrices used.

Chapter 5

CCA-2 Security of Cramer-Shoup With Matrices Over Group Rings

5.1 Introduction

One of the often sought results for cryptosystems is security against adaptive chosen ciphertext attacks (CCA-2 security). The Cramer-Shoup cryptosystem is a generalization of ElGamal's protocol which satisfies this requirement. The proof of security relies only on a standard intractability assumption, namely, the hardness of the Diffie-Hellman decision problem in the underlying group (see [7], [32]), and a hash function H whose output can be interpreted as a number in $\mathbb{Z}_q$ (where q is a large prime number). Finally, one needs an additional requirement, that is, it should be hard to find collisions in H. It was shown in the original paper that in fact with

a fairly minor increase in cost and complexity of the algorithms involved, we can completely eliminate H. We will discuss first the classical Cramer-Shoup cryptosystem and then present our work in [17], which addresses the same cryptosystem but instead uses matrices over group rings as the platform group.

5.1.1 Definition of provable security against adaptive chosen ciphertext attack

Through a series of papers by Naor and Yung, Rackoff and Simon, Dolev, Dwork and Naor the formal definition of security against active attacks was developed. The driving ideas behind this notion were alternatively called *chosen ciphertext security* or, equivalently, *non-malleability*. An intuitive explanation of this definition is that even if an adversary is allowed to have arbitrary ciphertexts of his choice decrypted, he still gets no partial information about other encrypted messages. For more information see [7], [32].

One defines the following game, which is played by the adversary. First, one runs the encryption scheme's key generation algorithm, with the necessary input parameters. (In particular, one can input a binary string in $\{0,1\}^n$, which describes the group G on which the algorithm is based.) The adversary is then allowed

to make arbitrary queries to the decryption oracle, decrypting ciphertexts which he has chosen.

The adversary then chooses two messages, m_0 and m_1, and submits these to the encryption oracle. The encryption oracle chooses a random bit $b \in \{0, 1\}$ and encrypts m_b. The adversary is then given the ciphertext, without knowledge of b.

Upon receipt of the ciphertext from the encryption oracle, the adversary is allowed to continue querying the decryption oracle. Of course the adversary is not allowed to submit the output ciphertext of the encryption oracle.

Finally, at the end of the game, the adversary must output $b' \in \{0, 1\}$, which is the adversary's best guess as to the value of b. Define the probability that $b' = b$ to be $1/2 + \epsilon(n)$, $\epsilon(n)$ is called the adversary's advantage, and $n \sim |G|$.

A cryptosystem is CCA-2 secure if the advantage of any polynomial-time adversary is negligible. Note here that we define a negligible function as one which grows slower than any inverse polynomial, n^{-c}, for any particular constant c and large enough n.

5.1.2 The Cramer-Shoup Scheme

The following is the general Cramer-Shoup cryptosystem protocol as defined over a group G: **Secret Key:** random $x_1, x_2, y_1, y_2, z \in \mathbb{Z}_q$

Public Key:

$$\text{group } G;\ \ g_1, g_2 \neq 1 \text{ in } G$$
$$c = {g_1}^{x_1}{g_2}^{x_2},\ \ d = {g_1}^{y_1}{g_2}^{y_2}$$
$$h = {g_1}^{z}.$$

Encryption of $m \in G$: $\ E(m) = (u_1, u_2, e, v)$, where

$$u_1 = {g_1}^{r}, u_2 = {g_2}^{r}, e = h^r m, v = c^r d^{r\alpha}, \text{ where } r \in \mathbb{Z}_q \text{ is random,}$$
$$\text{and}$$
$$\alpha = H(u_1, u_2, e).$$

Decryption of (u_1, u_2, e, v):

$$\text{If } v = {u_1}^{x_1+\alpha y_1}{u_2}^{x_2+\alpha y_2}, \text{ where } \alpha = H(u_1, u_2, e),$$
$$\text{then } m = e/{u_1}^{z}$$
$$\text{else "reject"}$$

Theorem 5.1.1. *[7] The Cramer-Shoup cryptosystem is secure against adaptive chosen ciphertext attacks assuming that (1) the hash function H is chosen from a universal one-way family, and (2) the Diffie-Hellman decision problem is hard in the group G.*

The proof of this theorem can be deduced easily from the same theorem as applied to our platform semigroup. The only difference arises from the two experimental facts we use to validate CCA-2 security for our semigroup, whereas for general groups one needs a more robust proof, which relies the ability to compute inverses in the chosen platform group (which is also why we needed a different technique for the proof), see [7] for the full proof.

5.2 A CCA-2 secure cryptosystem using matrices over group rings

In the previous chapter we proposed a public key exchange scheme using matrices over group rings. The public key exchange protocol was essentially the usual Diffie-Hellman, but we used matrices over a group ring of a (rather small) symmetric group as the platform and discussed it's security by addressing the Decision Diffie-Hellman (DDH) and Computational Diffie-Hellman (CDH) problems for this platform.

In this section we propose using the same platform (although a smaller subset thereof) and show that a scheme similar to the Cramer-Shoup scheme is CCA-2 secure. Our adapted protocol is

as follows:

Secret Key: random $x_1, x_2, y_1, y_2, z \in \mathbb{Z}_n$

Public Key: 3×3 non-identity matrices $M_1, M_2 \in M_{3\times 3}(\mathbb{Z}_7[S_5])$ such that M_1 is invertible and

$$M_1M_2 = M_2M_1, \qquad h = {M_1}^z$$
$$c = {M_1}^{x_1}{M_2}^{x_2}, \qquad d = {M_1}^{y_1}{M_2}^{y_2}.$$

Encryption of a message $N \in M_{3\times 3}(\mathbb{Z}_7[S_5])$: Generate $E(N) = (u_1, u_2, e, v)$, where

$$u_1 = {M_1}^r, \qquad u_2 = {M_2}^r,$$
$$e = h^r N, \qquad v = c^r d^{r\alpha},$$
$$r \in \mathbb{Z}_n \text{ is random, and} \qquad \alpha = H(u_1, u_2, e).$$

Decryption of (u_1, u_2, e, v): If

$$v = {u_1}^{x_1+\alpha y_1}{u_2}^{x_2+\alpha y_2},$$

where $\alpha = H(u_1, u_2, e)$, then

$$N = ({u_1}^z)^{-1}e$$

(Note that u_1 is invertible since M_1 is chosen to be invertible.)

else "reject".

Remarks: M_1 must always be chosen to be an invertible matrix, whereas M_2 is just any matrix such that $M_1M_2 = M_2M_1$. One

must also decide what group $\mathbb{Z}_n$ to use, i.e., n must be specified. Details for specifications of M_1, M_2 and n are outlined in a later section.

5.3 Adaptive CCA security for matrices over group rings

We will show by essentially proving Theorem 5.1.1 for our platform , that if for invertible matrices over $M_{3\times3}\mathbb{Z}_7[S_5]$ the DDH problem is hard, then the previously mentioned cyrptosystem is secure against adaptive chosen ciphertext attack. That is to say,

Theorem 5.3.1. *The Cramer-Shoup cryptosystem using the semigroup $G = M_{3\times3}\mathbb{Z}_7[S_5]$ is secure against adaptive chosen ciphertext attack assuming that (1) the hash function H is chosen from a universal one-way family, and (2) the decision Diffie-Hellman problem is hard in the group G.*

Before beginning the proof of the theorem we will need be reassured that the following two facts hold with a certain level of certainty:

1. Given an invertible matrix $M \in G = M_{3\times3}\mathbb{Z}_7[S_5]$ and random integers a, b and $c \in \mathbb{N}$, it is not possible to distinguish

between the distributions generated by (M^a, M^b, M^{ab}) and (M^a, M^b, M^c) (DDH assumption)

2. Given an invertible matrix $M \in G = M_{3\times 3}\mathbb{Z}_7[S_5]$ and a random integer a, it is not possible to extract information about a from M^a and M. In other words, the distributions generated by a random matrix N and M^a are indistinguishable (no information is leaked when choosing parameters).

We offer the following two experiments as evidence for the plausibility of the above facts. For each of these tests we used invertible matrices over the group ring $M_{3\times 3}\mathbb{Z}_7[S_5]$. In the first test we chose a random invertible matrix M (see section 5.3.1) and random integers a, b and $c \in \mathbb{N}$. We chose a and b in the interval $[10^{22}, 10^{27})$ and c in the interval $[10^{44}, 10^{54})$ so that ab and c were roughly of the same size. For each pair of resulting matrices M^{ab} and M^c we counted the frequency of elements of S_5 appearing in each entry.

Repeating this 500 times for randomly chosen M, a, b and c, we obtained a frequency distribution of elements of the group ring in each entry of the two matrices. From this we created the QQ-plots for each of the 9 matrix entries. QQ-plots are a quick and easy way to test for identical distributions, in which case the plots should be straight lines. As we can see from Figure 5.1, it appears that

judging by the distributions which are generated via this process it is not possible to distinguish DH pairs (M^a, M^b, M^{ab}) from non-DH pairs (M^a, M^b, M^c).

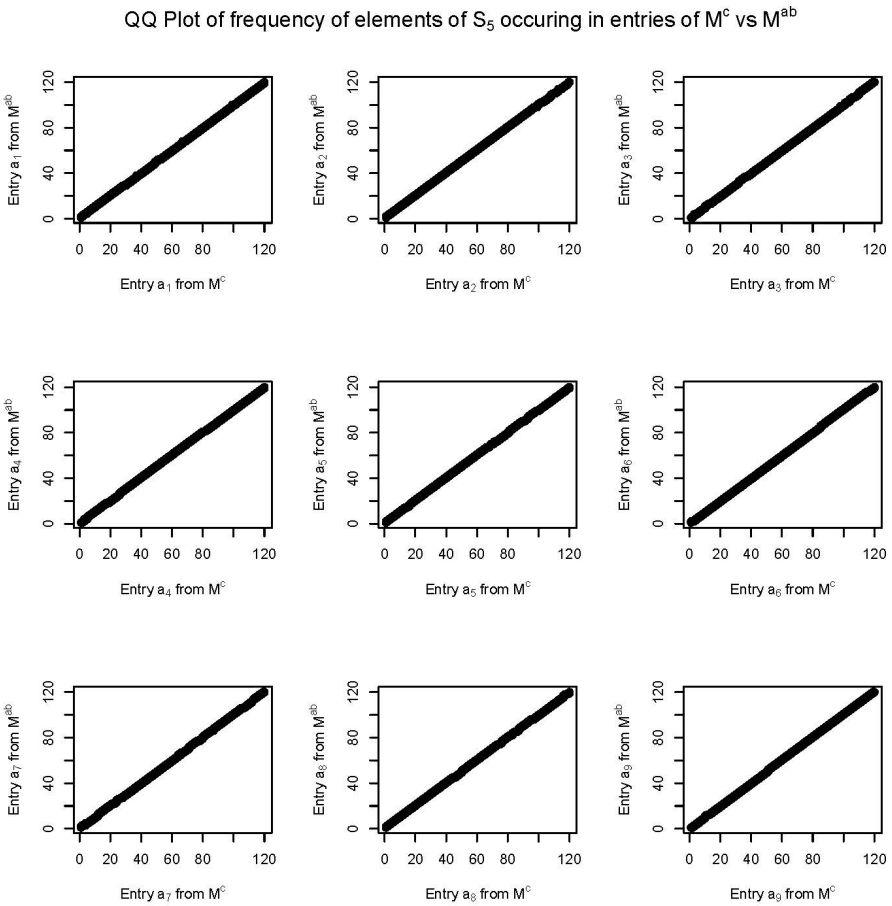

Figure 5.1: DDH results for M^{ab} vs. M^c

For experimental verification of the second fact, we conducted a similar experiment, i.e., for each of the 500 draws we varied all parameters N, M and a. We again generated QQ-plots as shown

in Figure 5.2, and these again show that no information about a is leaked from publishing M and M^a.

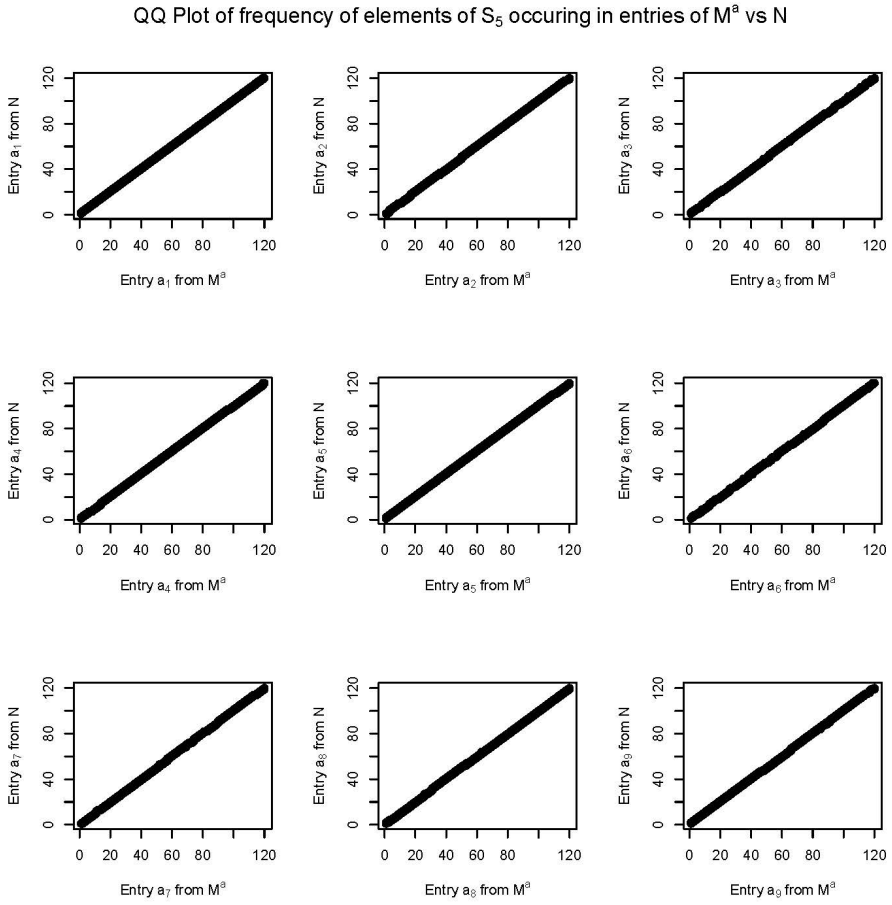

Figure 5.2: Security results for M^a vs. N

We are now in a position to prove Theorem 5.3.1. The proof will proceed in a similar fashion as Cramer-Shoup's original proof. We will begin by constructing an algorithm D to attack the DDH assumption. This algorithm relies on a probabilistic polynomial time

adversary A attacking our scheme, which succeeds with probability p, $\mathbb{P}_A(Success) = p$. Denote by DH the set of valid Diffie-Hellman tuples (M_1, M_2, M_1^r, M_2^r), and by R the set of all random tuples (M_1, M_2, M_3, M_4). Then the algorithm is constructed as follows:

- D receives input (M_1, M_2, M_3, M_4) from DH or R
- Pick $x_1, x_2, y_1, y_2, z \in \mathbb{Z}_n$ and a universal one-way hash function H
- The adversary A receives the public key, PK, which is

$$(M_1, M_2, c = M_1^{x_1} M_2^{x_2}, d = M_1^{y_1} M_2^{y_2}, h = M_1^z, H)$$

- The adversary picks two messages m_0, m_1 and publishes them
- D picks $b \in \{0, 1\}$ and passes to A

$$(M_3, M_4, M_3^z \cdot m_b, M_3^{x_1 + \alpha x_2} M_4^{y_1 + \alpha y_2}),$$

  where $\alpha = H(M_3, M_4, M_3^z \cdot m_b)$

- With this information A tries to determine b and returns its guess b'
- If $b = b'$ return "DH", else "R"

The proof is then verifying that this algorithm cannot attack the DDH problem. It is built from the following three claims.

Claim 1: $|\mathbb{P}(D = DH|DH) - \mathbb{P}(D = DH|R)| < \epsilon$. This claim is trivially true since D is a PPT algorithm and the DDH assumption holds as verified previously.

Claim 2: $\mathbb{P}(D = DH|DH) = \mathbb{P}_A(Success)$. If we are given a DDH tuple, then all decryption queries succeed for A. Hence the output of A will match the choice of b with $\mathbb{P}_A(Success)$.

Claim 3: $|\mathbb{P}(D = DH|R) - \frac{1}{2}| < \epsilon$. Since $\mathbb{P}(D = DH) = \mathbb{P}(A = b)$, the proof of this claim relies on the proof of two pieces. We need to show that for all decryption queries where $u_1 = M_1^{r_1}$ and $u_2 = M_2^{r_2}$ with $r_1 \neq r_2$, the decryption verification fails with non-negligible probability. In addition to this, we must also show that assuming all invalid decryptions fail, the adversary A does not learn any additional information about z.

We first start with the latter piece. If all invalid decryptions fail, then the only additional information A receives is when valid decryptions are performed. Thus, at the onset of the attack A only has information available that is given to him from PK, namely $h = M_1^z$. If A submits a valid ciphertext (u_1', u_2', e', v'), where $u_1' = M_1^{r'}$, then A obtains that $h^{r'} = M_1^{z^{r'}}$. However, based on the results above, if we denote $M = M_1^z$, then $h^{r'} = M^{r'}$ and the distributions of any random matrix N and $M^{r'}$ generated by r' are

indistinguishable, hence nothing is revealed about z.

Furthermore, from the encryption information passed to A, the only additional information A has is $M_3^z \cdot m_b$, which leaves him with obtaining information from M_3^z and M_1^z, i.e. solving a Diffie-Helmann problem, which we assumed was difficult in our scheme setup.

We are now left with showing that decryption almost always fails for invalid ciphertexts. Suppose that the adversary submits an invalid ciphertext, $(u_1', u_2', e', v') \neq (u_1, u_2, e, v)$. Then we have the following cases:

Case 1: If $(u_1, u_2, e) = (u_1', u_2', e')$ and $v \neq v'$, then the hash values α and α' will be the same, however decryption will certainly be rejected.

Case 2: If $(u_1, u_2, e) \neq (u_1', u_2', e')$ but $a = a'$, then this means that A has found a collision in H. But we assumed H was collision resistant, and since A runs in polynomial time, this can only happen with negligible probability.

Case 3: If $H(u_1, u_2, e) \neq H(u_1', u_2', e')$, then we have the following system of equations where we denote by $\log = \log_{M_1}$ and

$w = \log(M_2)$, and $u_1 = M_1^{r_1}$, $u_1' = M_1^{r_1'}$, $u_2 = M_2^{r_2}$ and $u_2' = M_2^{r_2'}$:

$$\log c = x_1 + wx_2 \tag{5.3.1}$$

$$\log d = y_1 + wy_2 \tag{5.3.2}$$

$$\log v = r_1x_1 + wr_2x_2 + \alpha r_1y_1 + \alpha wr_2y_2 \tag{5.3.3}$$

$$\log v' = r_1'x_1 + wr_2'x_2 + \alpha' r_1'y_1 + \alpha' wr_2'y_2. \tag{5.3.4}$$

These equations are linearly independent as can be verified by looking at

$$det \begin{pmatrix} 1 & w & 0 & 0 \\ 0 & 0 & 1 & w \\ r_1 & wr_2 & \alpha r_1 & \alpha wr_2 \\ r_1' & wr_2' & \alpha' r_1' & \alpha' wr_2' \end{pmatrix} = w^2(r_2 - r_1)(r_2' - r_1')(\alpha - \alpha')$$

The above determinant is nonzero since we are considering bad decryptions and hence

$$r_1 \neq r_1', r_2 \neq r_2', \alpha \neq \alpha'.$$

Therefore, almost surely any bad decryption queries of this form will be rejected.

Thus we have shown from Claim 3 that the adversary A is unable to correctly determine b given a random tuple, which we saw is equivalent to our algorithm not being able to distinguish a

random tuple from a DH tuple when given a random tuple. This together with Claim 1 shows that our algorithm cannot distinguish between tuples no matter what the input was. And finally, from Claim 2, we get that the adversary is unable to attack our scheme with an adaptive chosen ciphertext attack. □

5.3.1 Parameters for the Cramer-Shoup-like scheme using matrices over group rings

In this section we will now address two problems relevant to key generation in our scheme, namely, (1) how to sample invertible matrices and (2) how to sample commuting matrices.

Invertible matrices

Sampling invertible matrices can be done in a multitude of ways, employing various techniques. The first method is to construct a matrix which is a product of elementary matrices,

$$M = \prod_{i=1}^{n} E_i,$$

where E_i is any elementary matrix chosen at random from $M_{3\times 3}(\mathbb{Z}_7[S_5])$. Elementary matrices can be of one of the three types we describe below.

A matrix that either switches row i with row j, $T_{i,j}$, or multiplies row i by u, $T_i(u)$, or adds u times row j to row i, $T_{i,j}(u)$, (note that u should be an invertible element of $\mathbb{Z}_7[S_5]$). The general form of each matrix is given below.

$$T_{i,j} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 0 & & 1 & & \\ & & & \ddots & & & \\ & & 1 & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}, \quad T_i(u) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & u & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$$

$$T_{i,j}(v) = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & \ddots & & & \\ & & v & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$$

We can then easily compute M^{-1} as

$$M^{-1} = \prod_{i=1}^{n} E_{n-i+1}^{-1}$$

The drawback of generating an invertible matrix this way is that we do not have a good grasp of the randomness embedded in this

process. In particular, how large must n be to generate a truly random matrix? Given that there are 3 different types of elementary matrices, does it matter in which order they we construct and multiply this product and does the number of elementary matrices of each form matter? These are questions that have not been addressed and may influence the final invertible matrix generated in unknown ways.

The results in the previous sections are based on a different method. Instead of the sampling random elementary matrices, we propose an alternative generation method. We start with an already "somewhat random" matrix, for which it is easy to compute the inverse. An example of such a matrix is a lower/upper triangular matrix, with invertible elements on the diagonal:

$$M = \begin{pmatrix} u_1 & g_1 & g_2 \\ 0 & u_2 & g_3 \\ 0 & 0 & u_3 \end{pmatrix}.$$

Constructing the inverse of this matrix involves solving a matrix

equation,

$$M \cdot M^{-1} = I$$

$$\Rightarrow \begin{pmatrix} u_1 & g_1 & g_2 \\ 0 & u_2 & g_3 \\ 0 & 0 & u_3 \end{pmatrix} \cdot \begin{pmatrix} u_1^{-1} & g_4 & g_5 \\ 0 & u_2^{-1} & g_6 \\ 0 & 0 & u_3^{-1} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$\Rightarrow g_4 = -u_1^{-1} g_1 u_2^{-1}$$

$$g_5 = u_1^{-1} g_1 u_2^{-1} g_3 u_3^{-1} - u_1^{-1} g_2 u_3^{-1}$$

$$g_6 = -u_2^{-1} g_3 u_3^{-1}.$$

We then propose taking a random product of such invertible upper and lower triangular matrices. Given that these matrices are by construction more complex than elementary matrices, it seems reasonable to assume that we arrive at a more uniform distribution sooner than by simply using elementary matrices. In our experiments we used a product of 20 random matrices, where each term of the product was chosen randomly as either a random invertible upper or lower triangular matrix.

As mentioned previously, the benefits of this method are that inverses are easy to compute and that the chosen matrix already

has a large degree of randomness built in. In particular, any element of $\mathbb{Z}_7[S_5]$ can be used off the diagonal, and any invertible elements of the group ring can be used on the diagonal. These of course include elements such as $nu \in \mathbb{Z}_7[S_5]$, where $u \in S_5$ and $n \in \mathbb{Z}_7$.

Finally, we note that the order of the group $GL_3\mathbb{Z}_7[S_5]$ of invertible 3×3 matrices over $\mathbb{Z}_7[S_5]$ is at least 10^{313}. Indeed, if we only count invertible upper and lower triangular matrices that we described above, then we already have $(7 \cdot 120)^3(7^{120})^3 \sim 10^{313}$ matrices.

Commuting matrices

Now that we have sampled an invertible matrix (M_1 in our notation – see Section 5.2), we have to sample an arbitrary (i.e., not necessarily invertible) matrix M_2 that would commute with M_1.

For a given matrix $M_1 \in G$, define $M_2 = \sum_{i=1}^{k} a_i M_1^i$, where $a_i \in \mathbb{Z}_7$ are selected randomly. Then it is clear that $M_1M_2 = M_2M_1$. A reasonable choice for k is about 100 as this would yield $7^{100} \sim 10^{85}$ choices for M_2, which is a sufficiently large key space.

Additional Parameters

In the introduction of the Cramer Shoup algorithm adapted to our group rings we mentioned that we needed to specify the value of n for $\mathbb{Z}_n$. Based on experiments in our previous chapter we suggest $n \sim 10^{100}$. This seemed a reasonable choice of exponent since it both allowed quick computations and ensured that the power a matrix was raised to could not be obtained via brute force methods alone.

We additionally use a hash function H in our algorithm as in the original Cramer and Shoup paper. The only requirements of H are that it is drawn from a family of a universal one-way hash functions. This requirement is in fact less stringent than requiring the hash function to be chosen from a family of collision resistant hash functions. The latter implies that upon choosing a hash function H, it is infeasible for an adversary to find two different inputs x and y such that $H(x) = H(y)$. The weaker notion implies that upon choosing an input x and drawing a random hash function H, it is infeasible to find a different input y such that $H(x) = H(y)$.

Again we point out that in their paper Cramer and Shoup also provide details of their same scheme without requiring the use of any hash functions. The modified algorithm is only slightly more

complicated but relies on the same principles and can be applied to our setup as well.

Chapter 6

Digital Signatures

6.1 Introduction

Digital signature schemes are often cited as one of the most fundamental and useful inventions of modern cryptography. Such signature schemes are already quite widespread today - they are of particular use in email (see [3]). A common explanation of digital signatures uses the analogy of a signed letter (message, document, etc.) from the non-digital world. In this setup a person signs a document, seals it in an envelope and then mails it to a recipient. When the recipient receives the envelope they then open and examine the document. In particular they pay close attention to the signature, which allows them to verify the authenticity of the document and that the author was in fact the expected sender of the envelope.

In a similar fashion a digital signature scheme provides a way for any particular user to sign messages so that their signatures can later be verified by anyone else. To be specific, each user wishing to sign a message creates a matched pair of private and public signatures for the message (using the signer's public key). The person receiving the message can then verify and convince themselves that the message contents have not been altered since the message was signed. Furthermore, the signer can not later deny having signed the message, as no one but the signer possesses their private key. The recipient can perform the inverse operations of opening the letter and verifying the signature.

6.2 Ingredients of Digital Signatures

We follow the notation of Goldwasser and Bellare in their MIT lecture notes (for further reading and definitions see [3]). A digital signature scheme within the public key framework, is defined as a tuple of algorithms (G, σ, V).

- The key generation algorithm G is a probabilistic, polynomial-time algorithm which on input a security parameter 1^k, produces pairs (P, S) where P is called a public key and S a secret key. (We use the notation $(P, S) \in G(1^k)$ to indicate

that the pair (P, S) is produced by the algorithm G.)

- The signing algorithm σ is a probabilistic polynomial time algorithm which is given a security parameter 1^k, a secret key S in the range $G(1^k)$, and a message $m \in \{0,1\}^k$, produces as output a string s which we call the signature of m. (We use the notation $s \in \sigma(1^k, S, m)$ if the signing algorithm is probabilistic and $s = \sigma(1^k, S, m)$ otherwise. As a shorthand when the context is clear, the secret key may be omitted and we will write $s \in \sigma(S, m)$ to mean that s is the signature of message m.)

- The verification algorithm V is a probabilistic polynomial time algorithm which given a public key P, a digital signature s, and a message m, returns 1 (i.e "true") or 0 (i.e "false") to indicate whether or not the signature is valid. We require that $V(P, s, m) = 1$ if $s \in \sigma(m)$ and 0 otherwise. (We may omit the public key and abbreviate $V(P, s, m)$ as $V(s, m)$ to indicate verifying signature s of message m when the context is clear.)

- The final characteristic of a digital signature system is its security against a probabilistic polynomial time forger.

Note that if V is probabilistic, we can relax the requirement on V to accept valid signatures and reject invalid signatures with high probability for all messages m, all sufficiently large security parameters k, and all pairs of keys $(P, S) \in G(1^k)$. The probability is taken over the coins of V and S. Note also that the message to be signed may be plain text or encrypted, because the message space of the digital signature system can be any subset of $\{0,1\}^*$.

6.3 Classical digital signatures

We briefly mention a some of the classical digital signatures, again following [3].

6.3.1 RSA Digital Signature Scheme

The RSA Digital Signature Scheme is based on the RSA cryptosystem. The public key consists of a pair of integers (n, e) where n is the product of two large primes, e is relatively prime to $\phi(n)$ (ϕ is Euler's totient function). The secret key, d, is chosen such that $ed = 1 \mod \phi(n)$. One signs a message by computing the signature $\sigma(m) = m^d \mod n$. To verify that this is a valid signature one raises the signature to the power e and compares it to the original message.

6.3.2 El Gamal Digital Signature Scheme

The El Gamal Digital Signature Scheme is based on the Diffie-Hellman key exchange (DHKE) problem, and the difficulty of solving this problem. Presently, it is suggested that the best approach to tackling the DHKE problem is to first solve the discrete log problem. However, it is unknown whether computing a discrete log is as hard as solving the Diffie-Hellman problem.

The scheme works as follows:

- Public key: A triple (y, p, q), where $y = g^x \bmod p$, where p is prime and g is a generator of $\mathbb{Z}_p^*$.
- Secret key: $x \in \mathbb{Z}$ such that $y = g^x \bmod p$.
- Signing: The signature of message m is a pair (r, s) such that $0 \neq r, s \neq p-1$ and $g^m = y^r r^s \bmod p$.
- Verifying: Check that $g^m = y^r r^s \bmod p$ actually holds.

In order to generate a pair (r, s) which constitutes a valid signature, the signer begins by choosing a random number k such that $0 \neq k \neq p-1$ and $GCD(k, p-1) = 1$. Define $r = g^k \bmod p$. Next we need to determine an s such that $g^m = y^r r^s = g^{xr+ks} \bmod p$. Looking at the exponents in this equation we have the relationship $m = xr + ks \pmod{p-1}$. Solving this for s yields $s = (m - xr)k^{-1}$

mod $p-1$. The signature of m is then the pair (r, s). It is obvious that if an attacker could solve the discrete logarithm problem, he could break the scheme completely by computing the secret key x (or y) from the public information. Moreover, care needs to be taken in choosing the pseudo random number generator, since if an attacker finds k for one message, he can solve the discrete logarithm problem.

6.3.3 Schnorr Digital Signature Scheme

The Schnorr signature algorithm's security is is based on the intractability of certain discrete logarithm problems [30]. This signature scheme is considered one of the simplest digital signature schemes to be provably secure in a random oracle model. It is both efficient and allows for the generation of short signatures.

The scheme proceeds as follows:

- Public key: The tuple (q, g, y) where g is a generator of the group G with prime order q and $y = g^x$.
- Secret key: $x \in \mathbb{Z}_q^*$.
- Signing: Choose $k \in \mathbb{Z}_q^*$, let $r = g^k$, $e = H(M||r)$, where $||$ denotes string concatenation and H is a cryptographic hash function. Then set $s = k - xe$.The signature is thus (s, e).

- Verifying: Let $r_v = g^s y^e$, $e_v = H(M||r_v)$. If $e_v = e$ then the signature is verified.

6.4 Non-commutative digital signatures using non-commutative groups and rings

6.4.1 Braid Groups

Ko, Choi, Cho and Lee in 2002 [20] proposed a digital signature using braid groups where they assume the conjugacy search problem is hard, but the conjugacy decision problem is feasible.

To describe the algorithm we will need to work through some of their definitions. For each particular braid $b \in B_n$, there exists a canonical representation $b = \Delta^u \pi_1 \pi_2 \cdots \pi_l$, where Δ is the fundamental braid, π_i are permutation braids and u is an integer. Hence we can naturally define $\inf(b) = u, \sup(b) = u + l$ and $l(b) = l$. We can also use $\sim$ to denote the equivalence relation of conjugacy among braids. Furthermore, one can define the distance between two braids x and y as

$$d(x, y) = \min\{l(b) | y = b^{-1} x b\}.$$

Define the super summit set SSS(x) of a braid x by

$$SSS(x) = \{b \in B_n | b \sim x, \inf(b) = \text{maxinf}(x), \sup(b) = \text{minsup}(x)\},$$

where maxinf(x) is the maximum among infimums of all braids conjugate to x and minsup(x) is the minimum among supremums of all braids conjugate to x. Define $B_n(l) = \{b \in B_n | 0 \leq \inf(b), \sup(b) \leq l\}$.

With the notion of a distance we can also define the $d-neighborhood$ $S(x,d)$ of x in SSS(x) as follows

$$S(x,d) = \{y \in SSS(x) | d(x,y) \leq d\}.$$

Finally, they describe an algorithm which generates a random braid in $S(x,d)$. The random super summit braid generator

$$RSSBG(x,d) = (x', a),$$

upon input a braid x and an integer d, outputs a random braid $x' \in S(x,d)$ and a braid $a \in B_n(d)$ such that $x' = a^{-1}xa$.

We can now describe their braid signature scheme:

Key generating algorithm

- Select a braid $x \in B_n(l)$ such that $x \in SSS(x)$
- Choose (x', a) by running $RSSBG(x,d)$
- Return the public key $p = (x, x')$, and a remains secret

Signing algorithm

- Choose (α, b) by running $RSSBG(x,d)$ again

- Compute $h = H(m||\alpha)$ for a message m and let $\beta = b^{-1}hb$ and $\gamma = b^{-1}aha^{-1}b$
- Return the signature $\sigma = (\alpha, \beta, \gamma)$

Verifying algorithm

- Compute $h = H(m||a)$
- Accept if and inly if $\alpha \sim x$, $\beta \sim h$, $\gamma \sim h$, $\alpha\beta \sim xh$ and $\alpha\gamma \sim x'h$

In 2009 Wang and Hu [34] proposed a new digital signature based on a non-commutative group. Their signature scheme is based on the root extraction problem over braid groups.

The information published is the index n of the braid group, an integer $e \geq 2$ and a collision-free one way hash function H. Key generation involves randomly chosen braids $b_1, ..., b_k, r$ such that b_i and b_j commute, for all i and j. The signer computes

$$\alpha_i = rb_i^e r^{-1}, \text{ for } i = 1, ..., k$$

The public key is then $(\alpha_1, ..., \alpha_k)$, and the private key is $(b_1, ..., b_k, r)$.

To sign a chosen message m, the signer chooses a random braid s and calculates $H(m) = h_1...h_k$, $t = sr^{-1}$ and

$$u = s\left(\prod_{i=1}^{k} b_i^{h_i}\right)s^{-1}$$

The signature for the message m is (u, t).

The verify the signature one computes

$$v = \prod_{i=1}^{k} \alpha_i^{h_i}$$

and checks that the following equation holds

$$u^e = tvt^{-1}$$

They further proved that an existential forgery of the implementation under no message attack gives a solution to a variation of the conjugacy search problem; see [34].

We note that in general braid group based schemes are susceptible to Length-Based Attacks (see [26], [11] and [13]) and as such may not be suitable as platforms for non-commutative digital signatures.

6.4.2 Division Semirings

Another example of generating digital signatures over non-commutative algebraic objects was given by Anjaneyulu, Reddy and Reddy in 2008 [1]. They consider polynomials over non-commutative division semirings. They assume that the computational Diffie-Hellman problem is hard in their setup. Additionally their signature also relies on the difficulty of the Generalized Symmetrical

Decomposition (GSD) problem as applied to their rings.

Let R be a non-commutative division semiring, define $P_a = \{f(a)|f(x) \in \mathbb{Z}_{>0}[x]\}$, where $a \in R$. Then the GSD problem applied here is called the Polynomial Symmetrical Decomposition (PSD) problem. I.e., given $(a, x, y) \in R^3$ and $m, n \in \mathbb{Z}$, find $z \in P_z$ such that $y = z^m x z^n$.

The public information is the tuple (S, m, n, M, H), where S is the non-commutative division semiring, $m, n \in \mathbb{Z}$ and H is a cryptographic hash function which maps the message space M to S.

Alice publishes (p, q, y) where p, q are chosen randomly from S and y is generated by choosing a random polynomial $f(x) \in \mathbb{Z}_{>0}[x]$ such that $0 \neq f(p) \in S$, and computing $y = f(p)^m q f(p)^n$.

To generate a signature Alice choses another random $h(x) \in \mathbb{Z}_{>0}[x]$ such that $h(p) \in S$ and she defines

$$u = h(p)^m q h(p)^n.$$

She then computes

$$r = f(p)^m H(M) u f(p)^n$$

$$s = h(p)^m r h(p)^n$$

$$\alpha = h(p)^m r f(p)^n$$

$$\beta = f(p)^m H(M) h(p)^n$$

$$v_1 = h(p)^m H(M) h(p)^n$$

She then publishes $(u, s, \alpha, \beta, v_1)$ as the signature of the message M.

Bob verifies this is a valid signature by computing

$$v_2 = \alpha y^{-1} \beta.$$

He accepts the signature if and only if

$$u^{-1} v_1 = s^{-1} v_2.$$

The authors propose in their paper that their signature is secure against data forging of the message and also against existential forgery. However, upon further inspection we believe that both these claims may be incorrect. In particular, if we replace a valid message M with a forged message M_f, then the signature already sent would be valid. Although M is used in creating the signature, it is not needed in verification of the signature. Hence the verification test will succeed regardless of the message passed.

For existential forgery one is required to produce a valid signature for any message of their choosing. As such, after some inspection of the private and public keys involved one can easily choose their own parameters that satisfy their verification algorithm, i.e. the equations are easily solvable for arbitrary choices of messages and simple parameters.

6.4.3 General Non-Commutative Rings

In order to limit the ability of a third party from verifying the validity of a signature, Chaum and van Antwerpen [6] introduced the notion of *undeniable signatures*. Just as in the construction of a digital signature, undeniable signatures depend on the signer's public key in addition to the message being signed. However, verification can only be achieved by interacting with the legitimate signer through a *confirmation protocol*.

Additionally, this method allows the signer to deny the signature. In particular, if the signer refuses to deny, or fails to deny the signature, then the signature is assumed to be legitimate. Furthermore, as the signer's cooperation is required for verification of the signature they are protected from verification attempts by unauthorized third parties.

Chapter 7

A Non-Commutative Digital Signature

7.1 Introduction

We will proceed to define a new non-commutative digital signature which has some advantages over existing schemes following our work in [16]. The signature draws elements, from amongst other sources, the Diffie-Hellman key exchange and the Schnorr digital signature. The novelties of the signature allow once to chose any infinite non-commutative group for which the conjugacy search problem is infeasible as the platform group. Furthermore, we propose that both existential and data forgery are not possible with this signature, with the caveat that periodically one may have to update and change their public key.

7.2 A new non-commutative digital signature

Let G be an infinite group, which is finitely presented with exponential growth rate, such that the search conjugacy problem is exponential in time. In this signature we use f to represent a simple mapping function $f : G \to \{0,1\}^*$, which maps our group to some binary representation that can be digitally encoded. We will also be using a collision-free hash function H which maps into G. We note that for our algorithm Alice's public key will have to be updated/changed periodically depending on the number of messages she transmits.

- **Setup** The signer, Alice, chooses a group element g, a private key $s \in G$ and an integer $n \in \mathbb{N}$. Here we require $n = \prod_{k=1}^{l} p_k^{e_k}$, where p_k are prime and $e_k \in \mathbb{N}$. She then computes $x = g^{ns}$ and publishes x. Note, when exponentiating with an element of $h \in G$ we are representing conjugation, $g^h = h^{-1}gh$. Furthermore, the centralizer of g should be trivial.

- **Key generation:** The signer wishes to sign the message m. She picks t uniformly at random from G, a random factorization of $n = n_i n_j$. and computes the key $y = g^{n_i t}$.

- **Signature:** To generate the signature σ compute the following:

$$h = H(m||f(y))$$
$$\alpha = t^{-1}shy$$

  Alice then publishes her signature $\sigma = (y, \alpha, n_j)$ and the message m.

- **Verification:** To verify the signature compute $h' = H(m||f(y))$. The signature is valid and accepted if and only if

$$y^{n_j\alpha} = x^{h'y}$$

7.3 Security Analysis of the Signature Protocol

We note that some of the security of this algorithm is achieved by using elements of Schnorr's digital signature for commutative groups. In particular, the use of string concatenation and a hash function were borrowed from this scheme.

7.3.1 Completeness

Assume we are given a valid signature generated by Alice (y, α, n_j), and the public key x. It is trivial to check that Bob will always accept the signature as valid following the verification algorithm.

First Bob computes $h' = H(m||f(y)) \equiv h$. He then must verify the equality $y^{n_j\alpha} = x^{h'y}$. The left hand side yields

$$y^{n_j\alpha} = y^{n_j t^{-1} shy} = g^{n_i n_j t t^{-1} shy} = g^{nshy} = x^{hy}.$$

As $h' = h$ the equation is valid, hence the protocol is complete.

7.3.2 Data forging

Suppose that the forger Eve replaces a valid message which was signed, m, with a forged message, m_f. When Bob computes $h' = H(m_f||f(y)) \neq H(m||f(y)) \equiv h$ he wont be able to verify that $y^{n_j\alpha} = x^{h'y} \neq x^{hy}$. This equation in general doesn't hold unless there is a collision in the hash function for the particular choices of (m, y, f), which is unlikely given our assumptions about H.

Alternatively for this equation to hold Eve would have to be able to generate a forged message m'_f which produces $h'' = H(m'_f||f(y))$ such that the following is valid $y^{n_j\alpha} = x^{h''y}$. This is a difficult task as this then involves existential forgery, which we discuss in the next section. Furthermore, the message Eve signs m''_f will likely not have any valid interpretation as it was chosen in order to satisfy the equation and not the other way around.

7.3.3 Existential Forgery

Assume now that Eve wishes to sign a forged message m_f. She must then find a method to generate a valid signature without knowledge of the secret keys Alice has been using. To be precise she needs to produce $\sigma = (y_f, \alpha_f, n_{j_f})$ which passes the verification algorithm. This is where it is crucial that one incorporates the exponent n into the algorithm. For if $n = 1$ then the verification step reads

$$y^{\alpha} = x^{h'y} \Rightarrow y^{\beta h'y} = x^{h'y} \Rightarrow y^{\beta} = x.$$

Hence, choosing β determines y, which in turn gives us h and thus α. Combining all this yields an existentially forged signature for any m_f.

Repeating the above calculation in our case ($n \neq 1$) yields

$$y^{n_j \beta} = x.$$

In order to solve this equation y, β and n_j must be determined. A priori it is not clear how this may be done. One may proceed by choosing 2 of the 3 unknowns and solving for the third.

Assuming one follows this method, if β is the last parameter left to be determined, then we must solve a CSP problem, which we assumed to be difficult for our given platform group. Thus it seems that one must first choose β and one of the remaining parameters.

Suppose we decide to choose n_j next, thus we need to solve

$$y^{n_j} = x^{\beta^{-1}}$$

for y. In this case however we are not generally guaranteed a solution to this equation, as this would imply both, the existence of and the ability to, compute n_j^{th} roots in the underlying group. This is neither computationally feasible nor guaranteed if the platform group is chosen appropriately.

Hence it seems we are forced to choose both β and y first. We are now tasked with solving a DH problem which may or may not have a solution and as we have already discussed, and additionally is assumed to be computationally hard.

Based on the above analysis we believe that existential forgery of this protocol is not possible, unless one already knows a particular root of x. It is here where the need for a composite n is necessary. It turns out that some roots of x can in fact be computed once Alice has sent out a message and its signature. In effect, one can determine an n_j^{th} root of x by computing $y^{\alpha y^{-1} h^{-1}}$.

In order to hinder Eve from forging a message using an already used and published n_j Alice needs to keep a public list updated with the n_j's she has used thus far. She could for example update and publish this list after every time she signs a message. Thus, if

we receive a message with an n_j which has already been used then we know it can not be from Alice.

Another option is an adaptive chosen ciphertext attack, where Eve gets to submit messages of her choosing for signing. Again this method of attack is unlikely to succeed as the most Eve will obtain is a distribution of $t^{-1}s$, which is random and should yield no information about s nor t. Via this attack method Eve will be receiving information about n as well (she will be receiving various divisors of n), however, as suggested before, Alice should periodically update her secret keys. For example, once Alice has exhausted a small list of factorizations we recommend switching to a new x and n. This switch can be prolonged if careful choices are made in the factorization of n. In particular we can specifically choose not to include certain prime factors of n when choosing the integers n_j that are published. Or one might possibly also restrict the size of the exponents for the primes used in when choosing n_j.

7.3.4 Soundness

An additional method of breaking the security of this protocol requires the eavesdropper to recover s, t or n. Since neither g nor n were ever published, nor were they needed, there isn't a clear

method of starting a CSP attack. One would either have to be able to attack the random algorithm which generates t and hence obtain information about $t^{-1}s$, or there would have to be some method of attacking the hash function. Thus it appears that the security of this signature generation protocol relies on the appropriate choice of a hash function and the method by which one generates random group elements. We additionally note that care must be taken as to how the elements are transmitted. Since an eavesdropper can always read back $s^{-1}t$ from the publicly available information, for all random t, we must ensure that $s^{-1}t$ doesn't leak any information about s.

7.4 Proposed Platforms

As for most digital security schemes the choice of a platform group is a huge part of the security and success of the protocol. For this particular digital signature we obviously need to use platform groups for which the Conjugacy Search Problem is hard. Such non-commutative groups have been discussed in [25]. In particular, any group which has been deemed secure against Length Based Attacks and other attacks may be used. In lieu of this, we advocate using

polycyclic groups as they have been previously proposed for cryptography in [9], [14] and [15]. Furthermore, Garber, Kahrobaei and Lam in [12] have carried out some experiments which show that well-chosen polycyclic groups, with high Hirsch length, are secure against length based attacks. For a survey on non-commutative group-based cryptography see [10] and [25].

Bibliography

[1] G.S.G.N. Anjaneyulu, P.V. Reddy, and U.M. Reddy, *Secured digital signature scheme using polynomials over non-commutative division semirings*, International Journal of Computer Science and Network Security **8** (2008), no. 8, 278–284.

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

[3] M. Bellare and S. Goldwasser, *Lecture notes on cryptography*, 2008.

[4] Dan Boneh, *The decision Diffie-Hellman problem*, Algorithmic Number Theory Proc of ANTSIII (Joe P Buhler, ed.), Lecture Notes in Computer Science, vol. 1423, Springer-Verlag, 1998, pp. 48–63.

[5] S. Chakraborti and J. D. Gibbons, *Nonparametric statistical inference by*, Biometrics **67** (2011), no. 3, 1182–1183.

[6] D. Chaum and H. van Antwerpen, *Undeniable signatures*, Advances in Cryptology - CRYPTO 89, Lecture Notes in Computer Science **435** (1990), 212–216.

[7] R. Cramer and V. Shoup, *A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack*, Proc. Crypto (1998).

[8] W. Diffie and M. E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory **IT-22** (1976), 644–654.

[9] B. Eick and D. Kahrobaei, *Polycyclic groups: A new platform for cryptology?*, math.GR/0411077 (2004), 1–7.

[10] B. Fine, M. Habeeb, D. Kahrobaei, and G.Rosenberger, *Survey and open problems in non-commutative cryptography*, JP Journal of Algebra, Number Theory and Applications **21** (2011), 1–40.

[11] D. Garber, S. Kaplan, M. Teicher, B. Tsaban, and U. Vishne, *Length-based conjugacy search in the braid group*, Contemp. Math. Amer. Math. Soc. **418** (2006), 75–87.

[12] David Garber, Delaram Kahrobaei, and Ha Lam, *Analyzing length based attacks on polycyclic groups*, In Preparation (2012).

[13] J. Hughes and A. Tannenbaum, *Length-based attacks for certain group based encryption rewriting systems*, Workshop SECI02 Sèecuritè de la Communication sur Internet (2002).

[14] D. Kahrobaei and M. Anshel, *Decision and search in non-abelian Cramer-Shoup cryptosystem*, 2009, pp. 217–225.

[15] D. Kahrobaei and B.Khan, *A non-commutative generalization of the El Gamal key exchange using polycyclic groups*, Proceeding of IEEE, GLOBECOM (2006), 1–5.

[16] D. Kahrobaei and C. Koupparis, *Non-commutative digital signatures*, Journal of Groups, Complexity, Cryptology (to appear in 2012).

[17] D. Kahrobaei, C. Koupparis, and V. Shpilrain, *A cryptosystem secure against chosen ciphertext attack using matrices over group-rings*, Journal of Groups, Complexity, Cryptology (to appear in 2013).

[18] _____, *Public key exchange using matrices over group rings*, Submitted.

[19] Emilia Kasper, *Fast elliptic curve cryptography in openssl*, Financial Cryptography and Data Security: FC 2011 Workshops, RLCPS and WECSR, 2011.

[20] K. Ko, D. Choi, M. Cho, and J. Lee, *New signature scheme using conjugacy problem*, Cryptology ePrint Archive: Report 2002/168 (2002).

[21] K. Ko, S. Lee, J. Cheon, J. Han, J. Kang, and C. Park, *New public-key cryptosystem using braid groups*, Advances in cryptology—CRYPTO 2000 (Santa Barbara, CA), Lecture Notes in Comput. Sci., vol. 1880, Springer, Berlin, 2000, pp. 166–183.

[22] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, and Scott A. Vanstone, *Handbook of applied cryptography*, 2001.

[23] C.P. Milies and S.K. Sehgal, *An introduction to group rings*, Algebras and Applications, Kluwer Academic Publishers, 2002.

[24] A. Myasnikov, V. Shpilrain, and A. Ushakov, *Group-based cryptography*, Advanced Courses in Mathematics. CRM Barcelona. Birkhuser Verlag, Basel, 2008.

[25] ———, *Non-commutative cryptography and complexity of group-theoretic problems*, Amer. Math. Soc. Surveys and Monographs **177** (2011).

[26] A. Myasnikov and A. Ushakov, *Length based attack and braid groups: Cryptanalysis of anshel-anshel-goldfeld key exchange protocol*, Lecture Notes in Computer Science, Springer, Public Key Cryptography - PKC 2007 (2007).

[27] Moni Naor and Omer Reingold, *Number-Theoretic Constructions of Efficient Pseudo-Random Functions*, Proofs **51** (1997), no. 2, 458–467.

[28] S. Pohlig and M. Hellman, *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (Corresp.)*, Information Theory, IEEE Transactions on **24** (1978), no. 1, 106–110.

[29] J M Pollard, *Monte Carlo methods of Index computation (mod p)*, Mathematics of Computation **32** (1978), no. 143, 918–924.

[30] C P Schnorr, *Efficient signature generation by smart cards*, Journal of Cryptology **4** (1991), no. 3, 161–174.

[31] Daniel Shanks, *Class Number, a Theory of Factorization, and Genera*, Proceedings Symposia in Pure Mathematics (Donald J Lewis, ed.), (20), vol. 20, AMS, 1971, pp. 415–440.

[32] V. Shoup, *Why chosen ciphertext security matters*, IBM Research Report RZ 3076 (1998).

[33] M A Stadler, *Publicly verifiable secret sharing*, Advances in Cryptology EuroCrypt 96 (Ueli Maurer, ed.), Lecture Notes in Computer Science, vol. 1070, International Association for Cryptologic Research, Springer-Verlag, 1996, pp. 190–199.

[34] B.C. Wang and Y.P. Hu, *Signature scheme based on the root extraction problem over braid groups*, IET Information Security **3** (2009), 53–59.