Cryptographic Systems Using Linear Groups

by

Xiaowei Xu

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City Univerty of New York.

2006

UMI Number: 3204959

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

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

----------	Gilbert Baumslag
Date	Chair of Examining Committee
----------	Ted Brown
Date	Executive Officer
	Michael Anshel
	Benjamin Fine
	Sergei N. Artemov
	Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Abstract

Cryptographic Systems Using Linear Groups

by

Xiaowei Xu

Advisor: Professor Gilbert Baumslag

The objective of this thesis is to propose and discuss some new public-key encoding and zero-knowledge protocols based on matrix groups over polynomial rings. One of the key aspects of these protocols is the use of a rewriting system, known as the method of Reidemeister and Schreier, which is common in group theory but not used until now in cryptography. In this thesis, we will explore how to use groups of matrices over polynomial rings in cryptography. This will involve solving various group-theoretic problems and the difficulty of solving various polynomial equations. This will allow us to propose various new public-key encoding and zero-knowledge protocols.

Instead of using protocols based on number theory, we will devise protocols based on groups of matrices, whose entries consist of polynomials in a number of variables. The groups involved will be free groups and we use the free generators as letters in an alphabet which permit the encoding of messages in terms of matrices. We then transform these matrices into 2×2 integral matrices of determinant 1. The Euclidean algorithm and the method of Reidemeister-Schreier alluded to above, then enable us to decode the given messages.

We describe two cryptography schemes based on combinatorial group theory. We will discuss the underlying theory and provide some explicit examples. We suggest that combinatorial group theory could be the next generation platform in the develop-

ment of cryptography because it is more general and more flexible to take advantage of dynamic algebraic structures instead of static arithmetic. Our various schemes are based on more than one secret element and breaking one doesn’t break the whole scheme.

Acknowledgments

I would like to express my gratitude to all those who gave me the possibility to complete this thesis. I want to thank the Department of Computer Science of the University of New York for giving me permission to commence this thesis in the first instance, to do the necessary research work. I have furthermore to thank my committee, Prof Gilbert Baumslag, Prof Michael Anshel, Prof Ted Brown, Prof Sergei N. Artemov and Prof Benjamin Fine, who gave and confirmed this permission and encouraged me to go ahead with my thesis. I am bound to the Department of Computer Science of City College for their stimulating support.

I am deeply indebted to my supervisor Prof Gilbert Baumslag from the City College whose help, stimulating suggestions and encouragement helped me in all the time of research for writing of this thesis. It is an impossible mission to accomplish this thesis without his line-by-line double checking. My special thank should give to Prof Benjamin Fine and prof Michael Anshel. I thank Prof Benjamin for all those mathematics and cryptography discussion. I thank Prof Michael Anshel for bringing me a broad view of the whole cryptography industry.

I was working with Prof Shankar Subash for some artificial intelligence projects and I was working with Prof Eskicioglu M. Ahmet with watermark scheme. I thank

them for their suggestions and helps to improve my research skills. My colleagues from the Center for Algorithms and Interactive Scientific Software supported me in my research work. I want to thank them for all their help, support, interest and valuable hints. Especially I am obliged to Bernice Ravitz, Peggy Dean, Delaram Kahrobaei, Marianna Papaleo and Marcos Zyman. I also want to thank Kwan Andis, Sean Zhang, Ke Tang and other colleagues from the Group of Cryptography for all their assistance on the topic discussion.

Especially, I would like to give my special thanks to my daddy and mommy whose patient love enabled me to complete this work.

Contents

Chapter 1

Introduction

The objective of this thesis is to propose and discuss some new public-key encoding and zero-knowledge protocols based on matrix groups over polynomial rings. One of the key aspects of these protocols is the use of a rewriting system, known as the method of Reidemeister and Schreier, which is common in group theory but not used until now in cryptography.

1.1 General remarks

Public-key cryptography and zero-knowledge cryptography are two important fields within cryptography. Public-key cryptography is widely applied in key management and other fields because it allows involved parties to communicate secretly in non-secure channels. Zero-knowledge cryptography is frequently used in identification applications and in distributed computation. It differs from other cryptographic protocols in that it is provably secure.

Most public key cryptography systems presently in use are based on number the-

ory and they often depend on unraveling the structure of finite abelian groups. Recently, there has been a flurry of activity in the use of non-abelian groups in cryptography. Most of these proposals depend on the complexity of the conjugacy problem in the braid group. We believe that there exist a lot of variations of this idea by replacing the braid group with a variety of infinite, non-abelian groups. In this thesis, we will explore how to use groups of matrices over polynomial rings in cryptography. This will involve solving various group-theoretic problems and various polynomial equations. This will allow us to propose various new public-key encoding and zero-knowledge protocols.

1.2 Basic objective

Instead of using protocols based on number theory, we will devise protocols based on groups of matrices, whose entries consist of polynomials in a number of variables. The groups involved will be free groups and we use the free generators of matrix groups as letters to encode messages. The Euclidean algorithm and the method of Reidemeister-Schreier alluded to above, then enable us to decode the given messages.

For the public-key encoding protocol, we use multi-variable polynomials to generate the entries of the matrices that we will use to encode messages. These polynomials have the property that substituting values for the variables give rise to matrices, which can be uniquely re-expressed so that the resultant expressions turn into the original messages. The variable substitution will be the secret key of the protocol and is used to decode messages. Further, we will replace the underlying groups involved with some of their isomorphic copies and also use the addition of “noise” to

strengthen the protocol. Our scheme has flexibility that the choice of the underlying group can be kept secret during the communication.

For the zero-knowledge protocols, our protocols will make use of the conjugacy problem. We will discuss how to build the use of matrices over polynomial rings and to replace groups with their isomorphic copies in our zero-knowledge protocols.

1.3 Results

We describe two cryptography schemes based on combinatorial group theory. We will discuss their underlying theory and provide some explicit examples. We suggest that combinatorial group theory could be the next generation platform in the development of cryptography because it is more general and more flexible to take advantage of dynamic algebraic structures instead of static arithmetic. Our various schemes are based on more than one secret element and breaking one doesn't break the whole scheme. We have developed a polynomial matrix software package and utilize our software to investigate our protocols.

1.4 Overview of thesis

In Chapter 2 we recall some basic terminology, review some fundamental combinatorial group theory and introduce some general notions pertaining to cryptography and zero-knowledge.

In Chapter 3 we propose a public key encoding system based on matrix groups over polynomial rings. The general encoding scheme and some explicit examples will be given. We discuss the fundamental difference between our proposed scheme

and traditional schemes. At the same time, the weaknesses of our schemes and potential attacks are briefly addressed. Further possible techniques for improving our schemes are also discussed.

In Chapter 4 we introduce a zero knowledge scheme based on matrix groups. We informally prove that the scheme is complete and sound. We construct a simulator for the scheme to prove that the scheme is a zero knowledge scheme. At the same time, the scheme weaknesses and potential attacks for our scheme are briefly addressed. Further possible improvements are also discussed.

In Chapter 5 we take a close look at matrix groups. We investigate the group $SL(2,\mathbb{Z})$ and some of its free subgroups. We are especially interested in the construction of free subgroups with an arbitrarily large rank. We propose four methods to construct isomorphic groups of $SL(2,\mathbb{Z})$. This will lead to complicated representations of free groups which we will make use of.

In Chapter 6 we describe a procedure to decompose various matrices into products of given matrices. We review the Reidemeister-Schreier rewriting process and use this method to decompose elements in various subgroups of $SL(2,\mathbb{Z})$ into canonical forms.

In Chapter 7 we extend the methods in Chapter 6 and we show how to decompose various matrices using some underlying isomorphism problems. These problems and methods of decomposing the matrices are investigated according to three construction methods introduced in Chapter 3.

In Chapter 8 we introduce a methods to randomly generate polynomial matrices. Here we make use of the random coin flip-flop system. We also introduce constraints on the polynomials that are involved and show how to use these constraints to make

matrices more complex.

In Chapter 9 we discuss how to strengthen our public key-encoding scheme using noise. We propose changes to the subgroups that are involved and explain the addition of our notion of noise here. We also discuss how to eliminate noise in the communication channel while keeping it undetectable to attackers. We also discuss the method of the elimination of noise for the parties who are communicating.

In Chapter 10 we informally review the complexity of the conjugacy problem for matrix groups. We strengthen our zero knowledge protocols using isomorphisms and polynomials. We also discuss zero knowledge protocols of a different kind and describe how to construct other zero-knowledge schemes using matrix groups.

In Chapter 11 we compare our proposed scheme with other traditional schemes. We analysis the properties of our scheme and examine the possible attacks. We also briefly explain the testing environment we have implemented.

Chapter 2

Preliminary

2.1 Combinatorial group theory

Our universe is a very amazing system and we never stop trying to recognize and explore our universe. When we are exploring, one of our beliefs is that everything can be made up out of a finite number of basic elements. We thought that everything was made up of Water, Fire, Wood, Soil and Gold thousands of years ago. Between 1868 and 1870, in the process of writing his book, The Principles of Chemistry, Mendeleev created The Periodic Table of Elements. People were beginning to believe that everything was made from those 113 basic elements, now 118. We called these basic elements atoms and we found later that an atom is made out of three different types of particles: protons, neutrons, and electrons. More research showed that protons and neutrons are made of two types of quark. These quarks are said to have different flavors: up and down. These up and down quarks are the only quarks that are found in normal matter and they are known as first generation quarks.

Another example of such an attempt to break a particular phenomenon up into some basic more primitive ones arises in the study of an object's movement. For instance, if we want to study movement of the object in the solar system, we can decompose it into two basic movements, movement of the earth and movement of the object on the earth. The earth's movement in the solar system consists of its rotation around the sun and its self-rotation. The movement of object on the earth consists of translational motion and rotation.

We call these basic elements or basic movements generators in combinatorial group theory. As its name suggests, "generators" indicate that everything connected to them can be built up out of them.

On the other hand, there are a lot of relations between basic elements. A proton is made from two up quarks and a down quark. A neutron is made from two down quarks and an up quark. The basic movements have some intrinsic properties too. When an object is rotated by 360°, the object will go back to its original place, while the translation has another property, an object can go to infinity by translation. We call these relations among basic elements or basic movements relators in combinatorial group theory.

2.2 Alice's Bestbuy's nightmare

Alice's computer crashed because of a recent HDD virus. She decided to get a new computer from Bestbuy. She picked up her favorite Sony Vaio because of its cute design. She went to the checkout counter:

"How much is the notebook?"

”You are the lucky one, it is on sale right now. It is just fifteen hundred.”

”Wow! That’s a good deal, I will take it. ”

”Cash or credit”

”Do you accept Visa?”

”Sure, but I have to make a copy of your ID and credit card since it is a big transaction.”

Alice hesitated for a while and gave the salesman her License ID and credit card.

”Here are my license and credit card.”

”Thank you! You are all set. Enjoy your notebook!”

One month later, Alice received her credit card bill. To her surprise, she found a lot of strange transactions that she had no idea about. She decided to call the credit card customer service.

...

”My name is Mike, How can I help you today?”

”I have a lot of unknown transactions on my bill.”

”let me take a look at your bill. I am sorry, Alice! All those transactions were accompanied with sufficient information, such as your credit card number and your driver license number.”

”But I am pretty sure that I have never made those transactions.”

”I don’t know, there are a lot of identity problems these days. We can report this to police if you like.”

”Can I cancel those transactions?”

”Sorry, they are made online for downloading software, so we can not trace the transactions and the transactions can not be rolled back. So I am afraid that you have

to pay your bill. And after that, I suggest you cancel your credit card."

The problem here is when you try to prove you are who you are, other people can steal your information. One way to avoid it is to use so called zero-knowledge. Zero-knowledge proofs are proofs that yield nothing beyond the validity of the assertion. A simple zero knowledge scheme, due to Fiat and Shamir, is introduced in Chapter 4.

2.3 Overview of this chapter

In the next section we establish some notation and formally define group, ring, free group, group presentation, matrix group and related concepts. In section 5 we present a preliminary overview of some cryptography definitions. In section 6 we discuss zero knowledge interactive proof systems. Readers with a good background in group theory and zero knowledge interactive proofs may choose to skip all these sections.

2.4 Group theory

We recall some basic definitions in group theory. If the reader is not familiar with these concepts please refer to our references [9], [10], [39], [62], [63], [64], [85].

Definition 1 *: A group is a pair (G,*) where G is a non-empty set and * is a binary operation on G satisfying the following conditions.*

1) For any $a, b, c \in G$, $(a * b) * c = a * (b * c)$.

2) There is an element $e \in G$ such that $g * e = e * g = g$ for every $g \in G$.

3) For every $g \in G$, there exists a unique element h such that $g * h = h * g = e$. The element h is termed the inverse of g and is usually denoted by g^{-1}.

Usually the symbol * is omitted and we write ab for $a * b$.

If for every pair $a, b \in G$, $a * b = b * a$, we say that the group G is abelian.

Let (G,*) be a group and let H be a subset of G. If H is a group with respect to the binary operation * restricted to H, then we say H is a subgroup of G, denoted as $H \subseteq G$,

Definition 2 *Let G be a group, $H \subseteq G$. Then for a fixed $g \in G$, a right coset of H in G is*

$$Hg = \{hg | h \in H\}.$$

A left coset of H in G is

$$gH = \{gh | h \in H\}.$$

The right cosets of a subgroup have the following properties:

1. $Hg_1 = Hg_2$ if and only if $g1g2^{-1} \in H$;
2. Either $Hg_1 = Hg_2$ or else $Hg_1 \cap Hg_2 = \emptyset$;
3. Every element $g \in G$ is an element of some right coset.

Definition 3 *A subgroup H of a group G is called a normal subgroup of G if* $aH = Ha$ *for all* $a \in G$.

Definition 4 *Let N be a normal subgroups of group G and consider the set of all left cosets . This set will be denoted by G/N. If we define aN*bN=(aN)(bN)=(ab)N for any* $a, b \in G$, *we claim that this defines a binary operation on* G/N *and that this turns* G/N *into a group, called the factor group of G with respect to N.*

Definition 5 *A ring is a set R with two binary operations + and * satisfying the following axioms:*

1. *(a + b) + c = a + (b + c) for all a, b, c in R;*
2. *(ab)c = a(bc) for all a, b, c in R, where here we denote x*y simply by xy;*
3. *a + b = b + a for all a, b in R;*
4. *There exists an element 0 in R such that 0 + a = a for all a in R;*
5. *For every a in R, there corresponds an element -a in R, satisfying*

$$a + (-a) = 0;$$

6. *a(b + c) = ab + ac and (a + b)c = ac + bc for all a, b, c in R.*

The ring R is termed commutative if a b = b a for all a, b in R.

2.4.1 Free groups

If G is a group and X is a subset of G, we denote the smallest subgroup of G containing X by gp(X). It follows that

$$gp(X) = \{x_1^{\epsilon_1}...x_n^{\epsilon_n} | x_i \in X, \epsilon_i = \pm 1\}.$$

If $x_1^{\epsilon_1}...x_n^{\epsilon_n}$ and $y_1^{\eta_1}...y_n^{\eta_n}$ are two products with $x_i, y_i \in X$,$\epsilon_i = \pm 1$,$\eta_i = \pm 1$, then they are said to be identical if $n = m$,$x_i = y_i$ and $\epsilon_i = \eta_i$ for i=1, ..., m. Two products are said to be different if they are not identical. It is easy to see that two different products can give rise to the same element of G.

A product $x_1^{\epsilon_1}...x_n^{\epsilon_n}$, where $x_i \in X$ and $\epsilon_i = \pm 1$, is said to be a reduced X-product or a reduced product if $x_i = x_{i+1}$ implies $\epsilon_i \neq \epsilon_{i+1}$.

Definition 6 *A group G is said to be a free group freely generated by the set $X \subseteq G$ if*

(1) gp(X)=G;

(2) two different reduced X-products define different non-unit elements of G.

2.4.2 Group presentations

A group G is said to be generated by the set $X \subseteq G$ if gp(X)=G. The definition of gp(X) is the same as in Section 2.4.1. All elements in this set are called generators of G.

Given a group G and a set X=$\{g_1, g_2, ..., g_n\}$ of generators, an unworked out product of generators is called an X–word or simply a word of G. Usually we denote it as $W(g_1, g_2, ..., g_n)$. Sometimes we just use W if it is clear what is meant from the context.

Let $\{g_1, g_2, ..., g_n\}$ generate the group G. A word $R(g_1, g_2, ..., g_n)$ which defines

the identity element 1 in group G is called a relator. The equation

$$R(g_1, g_2, ..., g_n) = S(g_1, g_2, ..., g_n)$$

is called a relation if the word RS^{-1} is a relator. A set of relators $\{r_1, r_2, ..., r_n\}$ is a set of defining relators if all other relators are consequences of this set, i.e., follow from them the group laws or, in other words, can be derived from them.

Describing a group in terms of a set of generators X and a set of defining relators R is called presentation of a group. Suppose X=$\{g_1, g_2, ...\}$ and R=$\{r_1, r_2, ...\}$, we write $G =< x_1, x_2, ...; r_1, r_2, ... >$.

We say a presentation is finitely generated(related) if the number of generators(relators) in it is finite. If a presentation is both finitely generated and finitely related, we say that the presentation is finite. The relation between groups and presentations is shown by the following theorem:

Theorem 1 *Given a set of distinct symbols a, b, c, ... and a set(possibly empty) of words P, Q, R ,... in a, b, c, ... there is a unique group with presentation [64]*

$$< a, b, c, ...; P, Q, R, ... >$$

For example: $< a, b >$ is a free group with two generators. The number of elements in a free set of generators of a free group, is an invariant of the free group, termed its rank. Thus $< a, b >$ is a free group of rank 2. $< a, b; aba^{-1}b^{-1} >$ is an abelian group with two generators.

2.4.3 Matrix groups

Our schemes will involve free subgroups of various matrix groups. Recall that if R is a ring with identity 1, then the set R_n of all $n \times n$ matrices

$$M = \{ \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ ... & & & \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix} \}$$

becomes a ring with identity under the usual matrix operations of addition and multiplication.

We denote the set of all invertible elements of R_n by $GL(n, R)$. It follows that GL(n,R) is a group under matrix multiplication. The subset of all elements of $GL(n, R)$ of determinant 1 is a subgroup of GL(n,R), which we denote by SL(n,R). We shall be concerned with the special case that $R = \mathbb{Z}$ and $n = 2$, which we will discuss more fully in Chapter 5.

2.5 Cryptographic background

2.5.1 Basic computational complexity theory

We will begin with the definition of a few complexity classes.

Definition 7 *P: P is the set of decision problems solvable in polynomial time.*

NP: NP is the set of decision problems whose YES answers are checkable in polynomial time.

BPP: We say that language L has an algorithm with error probability ϵ if there is a non–deterministic Turing machine M such that

1) For all $x \in L$, $Prob[M\ accepts\ x] \leq 1 - \epsilon$, and

2) For all $x \in L$, $Prob[M\ accepts\ x] \geq \epsilon$.

One way function

Definition 8 *Informally, a function f(x) is a one-way function if*

1). The description of f is publicly known and does not require any secret information for its computation.

2). Given x, it is easy to compute f(x).

3). Given y, in the range of f, it is hard to find an x such that f(x)=y.

Somehow, the following are usually considered (have not been proven) to give rise to one-way functions:

1). The factoring problem: f(p,q)=pq, for randomly chosen primes p, q.

2). The discrete logarithm problem: f(N,g,x)=g^x(mod N), where N, g and x are integers.

3). The RSA one way function.

4). The subset sum problem.

5). The quadratic residue problem.

Computational indistinguishability

A central notion in modern cryptography is that of ”effective similarity”. The underlying theory is that we do not care whether or not objects are equal; all we care

about is whether or not a difference between the objects can be observed by a feasible computation. In case the answer is negative, the two objects are equivalent as far as any practical application is concerned.

The asymptotic formulation of computational indistinguishability refers to (pairs of) probability ensembles, which are infinite sequences of finite distributions, rather than to (pairs of) finite distributions. Specifically, we consider sequences indexed by strings (rather than by integers (in unary representation)). For $S \subseteq \{0,1\}^*$, we consider the probability ensembles X = $\{X_\alpha\}_{\alpha \in S}$ and Y = $\{Y_\alpha\}_{\alpha \in S}$, where each X_α (respectively, Y_α) is a distribution that ranges over strings of length polynomial in $n = |\alpha|$. We say that X and Y are computationally indistinguishable if for every feasible algorithm A the difference

$$dA(n) = max_{\alpha \in \{0,1\}^n} |Pr[A[X_\alpha] = 1] - Pr[A[Y_\alpha] = 1]|$$

is a negligible function in $|\alpha|$. That is:

Definition 9 *Computational indistinguishability [43] We say that $X = \{X_\alpha\}_{\alpha \in S}$ and $Y = \{Y_\alpha\}_{\alpha \in S}$ are computationally indistinguishable if for every family of polynomial-size circuits {Dn}, every polynomial p, all sufficiently large n and every $\alpha \in \{0,1\}^{poly(n)} \cap S$,*

$$|Pr[Dn[X_\alpha] = 1] - Pr[Dn[Y_\alpha] = 1]| < \frac{1}{p(n)}$$

where the probabilities are taken over the relevant distribution (i.e., either X_α or Y_α). That is, we think of D = {Dn} as of somebody who wishes to distinguish two distribu-

tions (based on a sample given to it), and think of 1 as of D's verdict that the sample was drawn according to the first distribution. Saying that the two distributions are computationally indistinguishable means that if D is an efficient procedure then its verdict is not really meaningful (because the verdict is almost as often 1 when the input is drawn from the first distribution as when the input is drawn from the second distribution).

2.5.2 Public key system

Symmetric cryptography, or secret key cryptography has been in use for thousands of years. It involves the use of a secret key known only to the participants of the secure communication. If Bob wants to send the message M securely over a public channel to Alice, he uses the key K and the encryption algorithm $E_K(M)$ to send the cipher text to Alice. Alice will decrypt the received ciphertext and use the decryption algorithm $D_K(C)$ to gain access to the message. DES, 3DES and AES [23], [24], [40] are examples of symmetric cryptography.

Asymmetric cryptography, also called public key cryptography, is a relatively new field. It was invented by Diffie and Hellman in 1976 [29]. The essential difference to symmetric cryptography is that this kind of algorithm uses two different keys for encryption and decryption.

Each participant in a secure communication owns a unique pair of keys, a public key p and a secret key s. The keys p and s are mathematically dependent on each other. A requirement of the asymmetric algorithm is that p can be computed easily from s while obtaining s from p is computationally infeasible. This property allows

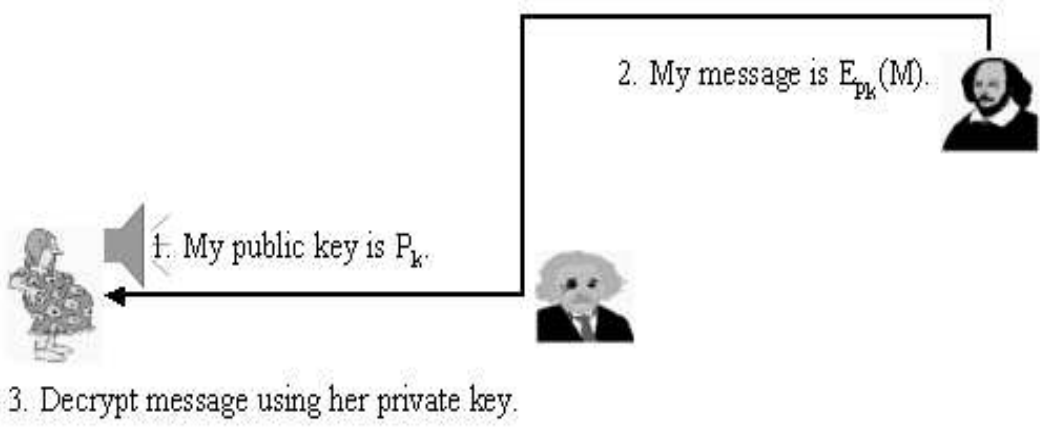

Figure 2.1: Public Cryptography Example.

one to make p publicly known while s must be kept secret by its owner.

The following example illustrates the framework of a typical public-key cryptographic system:

Suppose that Bob wants to send a message M to Alice over an insecure channel. Bob knows Alice's public key p_B in advance. Bob uses the encryption algorithm E_{p_B} and Alice's public key p_B to make the cipher text $c = E_{p_B}(M)$.

Alice uses the decryption algorithm D_{s_B} and her secret key s_B to compute Bob's message $m = D_{s_B}(c)$. Because s_B is known only to Alice and cannot be obtained from her public key p_B, this procedure is secure.

The RSA algorithm, the Diffie-Hellman algorithm and the method using elliptic curves are typical public key cryptographic systems.

2.6 Zero-knowledge proof systems

We recall some basic definitions in zero-knowledge. If the reader is not familiar with these concepts, please refer to our references [43], [42], [44], [46].

A mathematical proof is a sequence consisting of statements that are either axioms or are derived from previous statements via accepted deduction rules. However, in a zero knowledge scheme, the notion of a ”proof” is not a fixed object but rather a process by which the validity of an assertion is established. For example, the cross-examination of a witness in court is considered a proof in law, and failure to answer a rival’s claim is considered a proof in philosophical, political and sometimes even technical discussions. We will be examining zero-knowledge proof systems. In this section we introduce the basic concepts of particular importance of the distinction between a proof in the mathematical sense and a proof in a zero-knowledge scheme.

A proof system consists of a proof machine and a verification machine. We usually use P to stand for proof machine and V to stand for verification machine.

Two fundamental properties of a proof system are its soundness and completeness. The soundness property asserts that the verification procedure cannot be ”tricked” into accepting false statements. On the other hand, completeness captures the ability of some prover to convince the verifier of true statements.

Both properties are essential to the very notion of a proof system. We are now ready to define the concept of perfect zero-knowledge.

Definition 10 *Let (P, V) be an interactive proof system for some language L. We say that (P, V) is perfect zero-knowledge if for every probabilistic polynomial time interactive machine V* there exists an (ordinary) probabilistic polynomial time algo-*

rithm M so that for every $x \in L$ the following two random variables are identically distributed*

a) (P, V^*) (x) (i.e., the output of the interactive machine V after interacting with the interactive machine P on common input x;*

b) M^* (x) (i.e., the output of machine M^* on input x).

Related to perfect zero-knowledge is black-box simulation of perfect zero-knowledge.

Definition 11 *Denoting the running time of machine V^* when interacting with P on input x as $TimePV^*(x)$, an interactive proof system for a language L is black-box simulation zero-knowledge if there exists a probabilistic polynomial-time algorithm Mu such that for every polynomial Q the distribution ensembles $P(x), V^*(x)_{(x,V^*)\in D}$ and $MuV^*(x)_{(x,V^*)\in D}$ are polynomially indistinguishable even when the distinguishers are allowed black access to V^*, where*

$$D = \{(x, V^*) | x \in L \; and \; TimePV^*(x) \leq Q(x)\}$$

.

Chapter 3

Public-key encoding schemes based on matrix groups

3.1 The history of cryptography

Private key cryptography, also called symmetric key cryptography, is perhaps the most traditional method of cryptography. Two parties have a secret key that only they know. They also have a function T used to encrypt messages. Perhaps the earliest cryptographic system was developed by the Greek historian Polybios.

In 1977 the National Bureau of Standards created the Data Encryption Standard (DES) which was quite revolutionary at the time. DES was the first attempt at creating a universal encryption standard. In the January 1997 edition of the Federal Register, the U.S. National Institute of Standards and Technology (NIST) published a request for information regarding the creation of a new Advanced Encryption Standard (AES) for non-classified government documents.

The idea of public key cryptography, also referred to as asymmetric key cryptography, was first presented by Martin Hellman, Ralph Merkle, and Whitfield Diffie at Stanford University in 1976 [29]. The main idea behind public key cryptography is that not only can one make his/her algorithm public, one can make his/her key public as well. A person might publish their public key in a directory, and anyone who wishes to send that person secure information can encrypt that information using the public key and send it over insecure channels. Then when the cipher text gets to the receiver, she/he can decrypt it using her/his private key. This way, anyone can encrypt information, but only the person with the private key can decrypt it. Public key cryptography is very much like looking up someone's phone number in a large telephone directory. If you know the person's name, like having the private key, it is easy to find out her/his information. If, however, you only have her/his phone number you would have to search each entry one by one to find her/his name. Thus, the bigger the number of names, the harder and harder it would become to find a person.

Most public key cryptographic systems presently in use, such as the RSA algorithm, the Diffie-Hellman algorithm and the method using elliptic curves are based on number theory and hence depend a part on the structure of finite abelian groups. One of the earliest descriptions of a free group cryptographic system as well as a homomorphic version of it was in a paper by W. Magnus in the early 1970s [63]. In 1999, the Arithmetical key exchange [3] was introduced by Anshel, Anshel and Goldfeld. By contrast with other public schemes, it made use of a family of infinite non-abelian groups knows as braid groups and combinatorial group theory. A lot of cryptographic systems based on braid groups have been proposed and investigated since then [3], [28], [48], [59], [78].

We propose the following public-key encoding cryptographic system using matrix groups over polynomial rings. There are a number of aspects of this cryptographic system, which we will describe in detail in the latter chapters. This chapter is devoted to evolve basic steps of our public-key cryptographic system.

3.2 Cryptographic scheme based on matrix groups

We notice the following design properties of various encoding systems. The English language is made up from various letters, quotation marks, commas, etc. The decimal digit system uses digits 0,1,2,...9. The computer encoding system depends on two bits: 0 and 1. The basic idea behind the encoding system is that they all have some elementary symbols and a different arrangement of these symbols stands for a different message. This allows us to use matrices to encode messages as we will see below.

Recall that binary representation uses two bits: 0 and 1. 0 and 1 are just two symbols here and we can replace them with any two arbitrary symbols. Let's use $\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$ in place of 0 and use $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$ in place of 1. Under the new encoding system, we can use the sequence

$$\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$$

to stand for the corresponding binary 01010.

It turns out that if we now multiply these matrices using matrix multiplication,

the result is uniquely determined by the sequence we start out with. Moreover given the matrix we have now obtained, we can decompose it uniquely as a product of the matrices given at the outset. The decomposition details will be given in Chapter 6.

Based on this idea, we have the following public-key cryptographic system based on matrix groups. Our scenario is that Bob wants to send Alice a message securely over a non-secure public channel. We will describe our method in three sections: key setup, encryption and decryption.

3.2.1 Key setup

Alice chooses an appropriate commutative ring S and a free subgroup G of GL(n,S). G will be freely generated by a carefully chosen set $\{M_1, M_2, ..., M_k\}$ of k matrices where k is chosen in accordance with the scheme to be discussed. It follows that each M_i will take the form

$$M_i = \begin{pmatrix} a_{11}^i & a_{12}^i & ... & a_{1n}^i \\ a_{21}^i & a_{22}^i & ... & a_{2n}^i \\ ... & & & \\ a_{n1}^i & a_{n2}^i & ... & a_{nn}^i \end{pmatrix}.$$

We emphasize that G is free and is freely generated by the matrices $M_1, M_2, ..., M_k$. The public key then is taken to be the set $\{M_1, M_2, ..., M_k\}$. This set will be the "alphabet" used to encode messages.

3.2.2 Encryption

To encrypt a message M, Bob follows the following procedure:

1. Bob composes his message using the above matrices.

2. Bob multiplies the matrices resulting in the matrix $M^{'}$.

3. Bob sends $M^{'}$ to Alice.

3.2.3 Decryption

1. Alice receives the matrix $M^{'}$.

2. Alice decomposes the matrix $M^{'}$ resulting in a unique product of the given matrices. Again, the decomposition algorithm will be discussed in Chapter 6.

Here we assume that the process involved, i.e. re–expressing $M^{'}$ as a product of the given matrices M_i, is known to everyone. We need some secret key to complete our cryptographic system. In our scheme, we choose R to be the ring of polynomials in a specified number of variables over the original commutative ring S. So each entry in the matrices is a multivariable polynomial with coefficients from S.

The basic idea is to replace each of the entries a^i_{jk} in the matrices M_i with a multi-variable polynomial. One can substitute elements coming out of S for each of the variables in these polynomials. This will transform a matrix over the polynomial ring over S into a matrix over the ring S. We call this set of variable values a variable substitution. The original matrices and the variable substitution are chosen in such a way as to ensure that the end result of this process leads to a matrix that can be uniquely rewritten in a manner which allows us to recapture the original message. First we will introduce our general public-key cryptographic system over polynomial rings, then we will use an example to illustrate our scheme.

3.3 Cryptographic system based on matrix groups over polynomial rings

We will describe our scheme in three sections: key setup, encryption and decryption.

3.3.1 Key setup

1. Alice chooses an appropriate commutative ring S and a free subgroup G of GL(n,S). G will be generated by a carefully chosen set $M_1, M_2, ..., M_k$ of k matrices where k is chosen in accordance with the scheme to be discussed. It follows that each M_i will take the form

$$M_i = \begin{pmatrix} a^i_{11} & a^i_{12} & ... & a^i_{1n} \\ a^i_{21} & a^i_{22} & ... & a^i_{2n} \\ ... & & & \\ a^i_{n1} & a^i_{n2} & ... & a^i_{nn} \end{pmatrix}$$

   We emphasize that G will be free and freely generated by the set $M_1, M_2, ..., M_k$. We assume that given a matrix $M \in G$, there exists an algorithm whereby M can be re–expressed as a product of the matrices $M_1, M_2, ..., M_k$ and their inverses.

2. Alice randomly generates a set of kn^2 polynomials $f^i_{uv}(x_1, x_2, ..., x_m)$ in a chosen set of variables $x_1, x_2, ..., x_m$ over S. The generation of the polynomials is not completely random but subject to the fact that there exist elements

$v_1, v_2, ..., v_m \in S$ such that

$$f_{jk}^{(i)}(v_1, v_2, ..., v_m) = a_{jk}^{(i)}$$

Exactly how this is accomplished will be discussed in Chapter 8.

3. The public key then is taken to be the set:

$$Alphabet = \{ \begin{pmatrix} f_{11}^1 & f_{12}^1 & ... & f_{1n}^1 \\ f_{21}^1 & f_{22}^1 & ... & f_{2n}^1 \\ ... & & & \\ f_{n1}^1 & f_{n2}^1 & ... & f_{nn}^1 \end{pmatrix}, \begin{pmatrix} f_{11}^2 & f_{12}^2 & ... & f_{1n}^2 \\ f_{21}^2 & f_{22}^2 & ... & f_{2n}^2 \\ ... & & & \\ f_{n1}^2 & f_{n2}^2 & ... & f_{nn}^2 \end{pmatrix}, ..., \begin{pmatrix} f_{11}^k & f_{12}^k & ... & f_{1n}^k \\ f_{21}^k & f_{22}^k & ... & f_{2n}^k \\ ... & & & \\ f_{n1}^k & f_{n2}^k & ... & f_{nn}^k \end{pmatrix} \}$$

This set will be the "alphabet" used to encode messages. The substitution $v_1, v_2, ..., v_m$ will be the secret key.

3.3.2 Encryption

To encrypt a message M, Bob adopts the following procedure:

1. Bob composes a message using the matrices above whose entries are polynomials over S.

2. Bob multiplies the polynomial matrices to get a matrix $M^{'}$.

3. Bob sends $M^{'}$ to Alice.

3.3.3 Decryption

1. Alice receives the matrix $M^{'}$.

2. Alice replaces the variables $x_1, x_2, ..., x_m$ with the elements $v_1, v_2, ..., v_m$ from S, resulting in the matrix $M^{''}$.

3. Alice decomposes the matrix $M^{''}$ uniquely as a product of the matrices M_is and their inverses. The decomposition algorithm will be discussed in Chapter 6.

4. Alice replaces each of the matrices M_i with the corresponding matrices whose entries are polynomials over S. The result is the matrix $M^{'}$ which means that Alice can now determine the message being sent by Bob.

Notice that solving the following system of equations will steal the secret key of the scheme.

$$f_{jk}^{(i)}(x_1, x_2, ..., x_m) = a_{jk}^{(i)}.$$

To achieve the complexity of equation solving, we need to make sure that the sets of polynomials involved are sufficiently complicated. If we choose n polynomials at the outset, we usually choose the number of variables $m = n$. The reason for doing this, is that the fewer the number of variables the more redundant is the information. On the other hand, if there are more variables, it is, in general, easier to solve the system of equations.

Another possible attack is to try to find a way of decomposing a matrix over a polynomial ring directly. In this case, the potential security hole lies in the polynomial coefficients and their variable indexes. We will introduce polynomials based on

equation constraints in Chapter 8 to strengthen the above scheme.

3.4 A simple example of a cryptographic system based on matrix groups over polynomial rings

In this section we choose G=$SL(2,\mathbb{Z})$ to demonstrate our protocol. All these steps are generated by our testing software which will be introduced in Chapter 11.

3.4.1 Key Setup

Alice randomly generates the following three items:

1. A variable list X=$\{x_0, x_1, ..., x_7\}$.
2. A variable substitution $\{$1889,5162,4551,3617,-5638,-5318,-3552,3743$\}$.
3. Eight polynomials

$$f_{11}^1(X) = 2442x_1x_2x_3x_5 + 1103487294499323625,$$

$$f_{12}^1(X) = 3039x_2x_3x_4x_5x_6 + 7524x_1x_4x_5x_7 + 5323251031017389764129,$$

$$f_{21}^1(X) = 7264x_0x_3 + 4193x_0x_1x_4x_5x_6 - 7476x_1x_2x_3x_5x_7 - 8290437592330653722896,$$

$$f_{22}^1(X) = 7738x_0x_2x_3x_4x_6 - 3809x_1x_4x_5x_7 - 4816317737431479020647$$

$$f_{11}^2(X) = 1326x_5 + 2368x_3x_4x_5x_6 + 912171645948393878$$

$$f_{12}^2(X) = -8366x_0x_2x_5 - 382476702683531,$$

$$f_{21}^2(X) = 345x_1x_2x_3x_4x_5x_6x_7 + 116858048197288692333345381121\,and$$

$$f_{22}^2(X) = -3011x_1x_3x_4x_5x_7 + 6309144817376852301129$$

Now Alice will publish $\{M_1 = \begin{pmatrix} f_{11}^1 & f_{12}^1 \\ f_{21}^1 & f_{21}^1 \end{pmatrix}, M_2 = \begin{pmatrix} f_{11}^2 & f_{21}^2 \\ f_{21}^2 & f_{22}^2 \end{pmatrix}\}$. These two matrices will be the "alphabet" used to encode messages. Notice that M_1 and M_2 freely generate a free group. To see this, first observe that M_1 and M_2 are invertible. Second that the variable substitution induces a homomorphism of $GL(2, \mathbb{Z}[X])$ into $GL(2, \mathbb{Z})$. Since $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$ freely generate a free group, so do M_1 and M_2.

3.4.2 Encryption

To encrypt a message M, Bob adopts the following procedure:

1. Bob composes the message using M_1 and M_2.
2. Bob multiplies the matrices resulting in $M^{'}$.
3. Bob sends $M^{'}$ to Alice.

3.4.3 Decryption

1. Alice receives the matrix $M^{'}$.
2. Alice replaces the variables substitution. The result is the matrix $M^{''}$.

3. Using the procedure described in Chapter 6, Alice decomposes M'' into a unique reduced product of the matrices $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$.

4. Alice then replaces $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, $\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$ with $\begin{pmatrix} f_{11}^1 & f_{12}^1 \\ f_{21}^1 & f_{21}^1 \end{pmatrix}$, $\begin{pmatrix} f_{11}^2 & f_{12}^2 \\ f_{21}^2 & f_{21}^2 \end{pmatrix}$ respectively, thereby deciphering the message being sent by Bob.

3.5 Conclusion

The chapter introduced our public-key encoding protocol based on matrix groups over polynomial rings. Compared with other cryptographic systems based on polynomials [5], [50], [82], [70], we propose to use matrices as an alternative encoding tool. The benefit from this matrix encoding scheme is that some traditional attacks are very hard to launch. The weakness and advantage of the scheme will be discussed in Chapter 11.

This chapter proposes our public-key encoding scheme in the following steps:

1. The chapter starts with using matrices to represent messages. We use a set of matrix generators which form a free group. A few methods to construct free groups of matrices will be presented in Chapter 5.

2. Use matrix composition to encrypt messages and the decomposition of the resultant matrices to decrypt messages. The matrix decomposition algorithm will be discussed in Chapter 6.

3. Use polynomial variable substitution to provide secret keys. The random gen-

eration of polynomials and the scheme variants will be investigated in Chapter 8.

4. We claim that our scheme hides the underlying matrix group. This is accomplished by replacing each matrix entry with a polynomial. Further group hiding techniques using various isomorphism methods will be presented in Chapter 7.

Chapter 4

Zero knowledge scheme based on matrix groups

4.1 Introduction

Zero-knowledge proofs are proofs that yield nothing beyond the validity of the assertion. We start with Alice's Bestbuy nightmare and use the Fiat-Shamir zero-knowledge scheme to help her.

Let's restate Alice's situation:

Alice has a secret confirmation number S of her credit card. She tries to prove that she knows her magic number S. At the same time, she does not want to tell Bestbuy her number S. Now suppose that Bestbuy has the number S^2, which is called public key and is published by the credit card company. We assume that both Alice and Bestbuy have a calculator. This calculator can randomly generate a number and do multiplication. It cannot compute square roots. In another words, it is impossible to

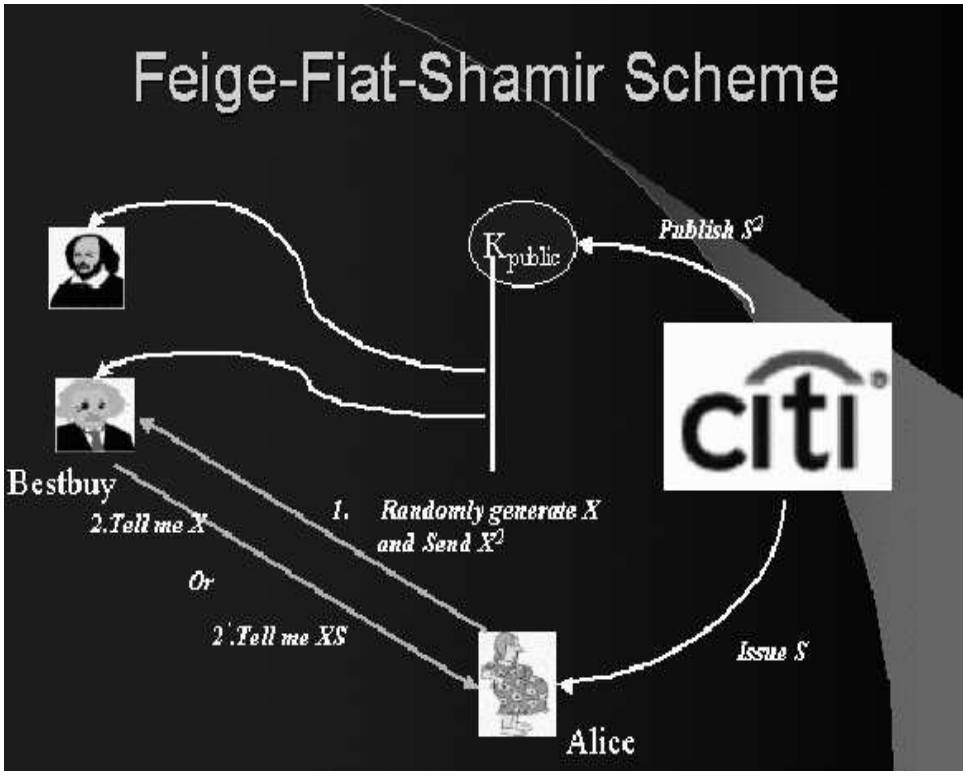

Figure 4.1: Simple Zero-knowledge Scheme

compute the value of X from X^2 while it is possible to compute X^2 from X.

Now Alice and Bestbuy follow the following protocol:

Step 1: Alice comes up with a random number X and tells Bestbuy the number X^2. All these can be done using her calculator.

Step 2: Bestbuy can ask Alice one and only one of the following two questions:

i What is the value of X?

ii What is the value of X*S?

Step 3: Alice tells Bestbuy the value of X if question i) is asked;

Alice tells Bestbuy the value of X*S if question ii) is asked.

Step 4: After repeating step 1-3 a few times, Bestbuy accepts her credit card if Alice can always answer the chosen questions.

Otherwise, Bestbuy will reject Alice's use of her credit card. Notice that Bestbuy can verify Alice's answer using its calculator.

There are three aspects of this protocol: soundness, completeness and zero knowledge. We have defined these three aspects in Chapter 2 and we just want to point out three things for this protocol:

1. Alice can answer both questions if and only if she knows the value of X and X*S. It follows that she knows the value of S because she can compute the value S as $S = \frac{X*S}{X}$.

2. Every time Alice tells Bestbuy the value of X, Bestbuy does not get any information about the value of S. The informal argument is that Bestbuy can randomly generate X, compute X^2, ask the value of X and answer the question all by itself. And this X is nothing to do with S.

3. Every time Alice tells Bestbuy the value of X*S, Bestbuy does not get any information of the value of S either. The informal argument is that Bestbuy can randomly generate a value Y and assume $X^2 = \frac{Y^2}{S^2}$. Then as the above simulation, Bestbuy can follow the protocol with question ii) by itself.

In this chapter we are going to introduce a zero-knowledge scheme based on matrix groups and a zero-knowledge scheme based on matrix groups over polynomial rings. We use the conjugacy searching problem to construct our schemes.

Definition 12 *Given a group G and two elements $x, y \in G$, the conjugacy searching problem is the problem of finding an $\alpha \in G$ such that $y = \alpha^{-1} x \alpha$.*

In our scheme, Alice is a prover and Bob is a verifier. We will describe our scheme in general followed by an example. We are going to introduce our zero-knowledge scheme in two parts: key setup and zero-knowledge proof. We will also give an informal argument about completeness, soundness and zero-knowledge of our proposed scheme.

4.2 Zero-knowledge scheme based on matrix groups

4.2.1 Key setup

Alice will set up her secret key and her public key in this stage.

1. Alice chooses an appropriate commutative ring R and a subgroup G of GL(n,R). G will be generated by a carefully chosen set $M_1, M_2, ..., M_k$ of k matrices where k is chosen in accordance with the scheme to be discussed. Each M_i takes the form

$$M_i = \begin{pmatrix} a_{11}^i & a_{12}^i & ... & a_{1n}^i \\ a_{21}^i & a_{22}^i & ... & a_{2n}^i \\ ... & & & \\ a_{n1}^i & a_{n2}^i & ... & a_{nn}^i \end{pmatrix}.$$

   We emphasize that G is not necessary free for our zero-knowledge scheme, compared to our public-key encoding scheme.

2. Alice randomly generates two elements $x \in G$ and $\alpha \in G$.

3. Alice calculates $y = \alpha^{-1} x \alpha$.

4. Alice publishes x and y as her public keys. At the same time, she will keep α as her secret key.

4.2.2 Zero-knowledge proof

In this stage, Alice's objective is to convince Bob that she knows $\alpha \in G$ without revealing any information about α. The protocol is obtained by repeating the following basic steps n times.

1. Alice randomly generates $\beta \in G$, calculates $z = \beta^{-1} y \beta$ and send z to Bob.

2. Bob randomly chooses one of two questions:

   i What is the value of β?

   ii What is the value of $\alpha\beta$?

3. Alice sends the answer δ as follows:

   $\delta = \beta$ if the question i) is chosen in step 2;

   $\delta = \alpha\beta$ if the question ii) is chosen in step 2.

4. Bob verifies Alice's answer:

   $z = \delta^{-1} y \delta$ if the question i) is chosen in step 2;

   $z = \delta^{-1} x \delta$ if the question ii) is chosen in step 2.

We argue informally that this protocol is a complete, sound and perfect zero knowledge proof.

1. We first show completeness:

   if the question i) is chosen in step 2, Alice can cause Bob to accept the proof by sending $\delta = \beta$.

   if the question ii) is chosen in step 2, Alice can cause Bob to accept the proof by sending $\delta = \alpha\beta$. Thus a faithful prover, who knows α, can always follow this protocol.

2. We next show soundness:

   If Alice is always able to answer both questions i) and ii) in step 2, she has to know the value of β and $\alpha\beta$. This means that she can calculate $\alpha = \alpha\beta\beta^{-1}$. In another word, if Alice can follow the protocol, she must know the value of α.

3. Finally we show that it is zero-knowledge:

   We construct a simulator M which simulates the whole protocol without knowledge of α. The complete simulation is constructed by performing the following simulation for basic steps.

   The simulator M guesses the question i) or ii) and chooses a random matrix $\beta' \in G$. The simulator also generates a value r according to the guessed question:

   $r = \beta'^{-1} y \beta'$ if the question i) is guessed.

   $r = \beta'^{-1} x \beta'$ if the question ii) is guessed.

If V's challenge at step 2 is the guessed question, M sends the value of $z = r$. If the guess is not correct, then M restarts the simulation with a freshly guessed question.

To prove that our scheme is zero knowledge, r's in the above simulation program and z's in the above zero-knowledge proof should have the same distribution. In another word, the following two pairs should have the same distribution.

(a) $\beta^{-1}y\beta$ and $\beta'^{-1}y\beta'$.

(b) $\beta^{-1}y\beta$ and $\beta'^{-1}x\beta'$.

β and β' are chosen randomly, so obviously the first pair have the same distribution. The same distribution for the second pair can be shown as follows:

(a) $\beta^{-1}y\beta$=$\beta^{-1}\alpha^{-1}x\alpha\beta$.

(b) $\alpha\beta$ and β' have the same uniform distribution over G because that G is a group.

We further use the following example to demonstrate the above zero-knowledge scheme. The purpose of this example is to show the main idea and the conjugacy problem used in the following example can be trivially solved.

4.3 An example of zero-knowledge based on matrices

Our simple example uses the conjugacy searching problem of a free group G freely generated by $\{\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}\}$. We describe our scheme in two phases: key setup phase and proof phase.

4.3.1 Key setup procedure

Alice chooses a secret key $\alpha \in G$, a public key $x \in G$ and computes $y = \alpha^{-1}x\alpha$ in this stage. For example:

$$x = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 26 & 15 \\ 19 & 11 \end{pmatrix}.$$

$$\alpha = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 8 & 3 \\ 5 & 2 \end{pmatrix}.$$

$$y = \alpha^{-1}x\alpha = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}\begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}\begin{pmatrix} 26 & 15 \\ 19 & 11 \end{pmatrix}\begin{pmatrix} 8 & 3 \\ 5 & 2 \end{pmatrix} = \begin{pmatrix} -55 & -21 \\ 241 & 92 \end{pmatrix}.$$

Alice publishes x and y as her public keys which will be used in the coming proof stage. Now the scenario is that Alice tries to prove to Bob that she knows α without giving out the value of α.

4.3.2 The zero-knowledge proof

Step 1: Alice randomly generates

$$\beta = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 3 & 1 \\ 2 & 1 \end{pmatrix},$$

calculates

$$z = \beta^{-1} y \beta = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} -55 & -21 \\ 241 & 92 \end{pmatrix} \begin{pmatrix} 3 & 1 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} -1114 & -409 \\ 3135 & 1151 \end{pmatrix}$$

and send z to Bob.

Step 2: Bob randomly chooses one of two questions:

i What is the value of β?

ii What is the value of $\alpha\beta$?

Step 3: Alice sends the answer δ:

$\delta = \beta = \begin{pmatrix} 3 & 1 \\ 2 & 1 \end{pmatrix}$ if the question i) is chosen in step 2.

$\delta = \alpha\beta = \begin{pmatrix} 8 & 3 \\ 5 & 2 \end{pmatrix} \begin{pmatrix} 3 & 1 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} 30 & 11 \\ 19 & 7 \end{pmatrix}$ if the question ii) is chosen in step 2.

Step 4: Bob verifies Alice's answer:

$z = \delta^{-1} y \delta$ if the question i) is chosen in step 2.

$z = \delta^{-1} x \delta$ if the question ii) is chosen in step 2.

Same as our public-key cryptographic scheme, we are going to introduce a zero-knowledge scheme based on matrix groups over polynomial rings. In another words, we choose R to be the ring of polynomials in a specified number of variables over a second commutative ring S. So each of the entries in the matrices is a multivariable polynomial.

4.4 Zero-knowledge based on matrix groups over polynomial rings

4.4.1 Key setup

1. Alice chooses an appropriate commutative ring S and a free subgroup G of GL(n,S). G will be generated by carefully chosen set $M_1, M_2, ..., M_k$ of k matrices where k is chosen in accordance with the scheme to be discussed. Each M_i takes the form

$$M_i = \begin{pmatrix} a_{11}^i & a_{12}^i & ... & a_{1n}^i \\ a_{21}^i & a_{22}^i & ... & a_{2n}^i \\ ... & & & \\ a_{n1}^i & a_{n2}^i & ... & a_{nn}^i \end{pmatrix}.$$

   We emphasize that G is not necessary free for our zero-knowledge scheme, compared to our public-key encoding scheme.

2. Alice randomly generates a set of kn^2 polynomials $f^i_{uv}(x_1, x_2, ..., x_m)$ in a chosen set of variables $x_1, x_2, ..., x_m$. The generation of polynomials is not completely random but subject to the fact that there exists elements $v_1, v_2, ..., v_m \in R$ such that

$$f^{(i)}_{jk}(v_1, v_2, ..., v_m) = a^{(i)}_{jk}$$

   Exactly how this is accomplished will be discussed in Chapter 8.

3. Notice that the set of matrices over polynomial rings:

$$\{\begin{pmatrix} f1_{11} & f1_{12} & ... & f1_{1n} \\ f1_{21} & f1_{22} & ... & f1_{2n} \\ ... & & & \\ f1_{n1} & f1_{n2} & ... & f1_{nn} \end{pmatrix}, \begin{pmatrix} f2_{11} & f2_{12} & ... & f2_{1n} \\ f2_{21} & f2_{22} & ... & f2_{2n} \\ ... & & & \\ f2_{n1} & f2_{n2} & ... & f2_{nn} \end{pmatrix}, ..., \begin{pmatrix} fk_{11} & fk_{12} & ... & fk_{1n} \\ fk_{21} & fk_{22} & ... & fk_{2n} \\ ... & & & \\ fk_{n1} & fk_{n2} & ... & fk_{nn} \end{pmatrix}\}$$

   generates a group, we denote it as GM, this group is isomorphic to the original group G.

   Alice randomly generates two elements $x \in GM$ and $\alpha \in GM$.

4. Alice calculates $y = \alpha^{-1} x \alpha$.

5. Alice publishes x and y as her public keys.

4.4.2 Zero-knowledge proof

In this stage, Alice tries to convince Bob that she knows $\alpha \in GM$ without revealing any information about α.

The protocol is obtained by repeating the following basic steps n times.

1. Alice randomly generates $\beta \in GM$, calculates $z = \beta^{-1} y \beta$ and send z to Bob.

2. Bob randomly chooses one of two questions:

   i What is the value of β?

   ii What is the value of $\alpha\beta$?

3. Alice sends the answer δ:

   $\delta = \beta$ if the question i) is chosen in step 2;

   $\delta = \alpha\beta$ if the question ii) is chosen in step 2.

4. Bob verifies Alice's answer:

   $z = \delta^{-1} y \delta$ if the question i) is chosen in step 2;

   $z = \delta^{-1} x \delta$ if the question ii) is chosen in step 2.

This protocol is still a complete, sound and perfect zero knowledge proof because the informal argument for our previous zero-knowledge scheme still holds. In fact, our previous argument is true for any arbitrary group.

4.5 Conclusion

Besides the review of Fiat-Shamir zero-knowledge scheme, this chapter also introduces our two zero-knowledge schemes. Our first scheme is based on matrix groups. The second one is based on matrix groups over polynomial rings. We use the matrix

decomposition and conjugacy problems as our one way functions in both schemes. The soundness, completeness and zero-knowledge of our schemes are informally proven. Compared to traditional zero-knowledge schemes, our scheme is based on infinite non-abelian groups and depends on a totally different arithmetic structure.

Some variants of zero knowledge schemes will be proposed in Chapter 10.

Chapter 5

Matrix groups

5.1 Introduction

In previous chapters we use matrix groups to construct our public-key protocols and zero-knowledge protocols. In this chapter we are going to introduce a few methods to construct matrix groups. Then we will talk about how to decompose matrices in the next chapter.

Definition 13 *Given a matrix M in a matrix group G, M is called as a scalar matrix if and only if*

1. *M's diagonal entries all contain the non-zero same element.*

2. *M's other entries are all 0's.*

In fact, all scalar matrices in a matrix group G form a normal subgroup of G. Considering SL(n,R) group, we have the following definition of PSL(n,R):

Definition 14 *The projective special linear group PSL(n,R) is the factor group* $\frac{SL(n,R)}{DL(n,R)}$*, where DL(n,R) is the scalar matrix group of SL(n,R).*

At first, the chapter investigates the matrix group $GL(2,\mathbb{Z})$. We discuss relations among $GL(2,\mathbb{Z})$, $SL(2,\mathbb{Z})$ and $PSL(2,\mathbb{Z})$. Secondly, the chapter introduces two free subgroups of $PSL(2,\mathbb{Z})$ and two free subgroups of $SL(2,\mathbb{Z})$. We claim that these subgroups can be used in our public-key encoding scheme. We also propose four methods to construct matrix groups based on the isomorphism problem. In another words, given a matrix group G, we use our methods to construct another matrix group and the new matrix group is isomorphic to the original group G. Finally, we take a quick review of a few methods to construct new groups from existing ones. We claim that these methods can be used to implement our zero-knowledge protocols.

5.2 $GL(2,\mathbb{Z})$ and $SL(2,\mathbb{Z})$

Recall that $SL(2,\mathbb{Z})$ consists of all 2×2 matrices on $\mathbb{Z}$ whose determinant is 1. By definition, group $GL(2,\mathbb{Z})$ consists of all invertible 2×2 matrices on $\mathbb{Z}$. In fact, $GL(2,\mathbb{Z})$ consists of all 2×2 matrices over $\mathbb{Z}$ whose determinant is 1 or -1.

Besides the above observation, we also observe the following relation between $SL(2,\mathbb{Z})$ and $PSL(2,\mathbb{Z})$:

5.2.1 $SL(2,\mathbb{Z})$ and $PSL(2,\mathbb{Z})$

There exists only two scalar matrices in $SL(2,\mathbb{Z})$, the identity matrix and $\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$.

Notice that

$$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} -a & -b \\ -c & -d \end{pmatrix} = - \begin{pmatrix} a & b \\ c & d \end{pmatrix}.$$

By the definition of $PSL(2,\mathbb{Z})$, two matrices M and -M are treated as different elements in $SL(2,\mathbb{Z})$ while they are treated as the same element in $PSL(2,\mathbb{Z})$. Based on the above observation, two matrices $M_1, M_2 \in SL(2,\mathbb{Z})$ are the same element in $PSL(2,\mathbb{Z})$ if and only if $M_1 = M_2$ or $M_1 = -M_2$ in $SL(2,\mathbb{Z})$.

The following sections describe a presentation for $PSL(2,\mathbb{Z})$ and use this presentation to construct free subgroups of $PSL(2,\mathbb{Z})$.

5.2.2 Presentation for matrix group $PSL(2,\mathbb{Z})$

Based on the decomposition algorithm we are going to describe in Chapter 6, any matrix in $PSL(2,\mathbb{Z})$ can be decomposed into a product of $\pm \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$, $\pm \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and their inverses, i.e. $PSL(2,\mathbb{Z})$ is generated by $\pm \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$ and $\pm \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$.

Further, $PSL(2,\mathbb{Z})$ can be defined by two relators: $(\pm \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix})^2$,$(\pm \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \pm \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix})^3$ [4], [60].

In another words, $PSL(2,\mathbb{Z})$ has the following presentation:

$PSL(2,\mathbb{Z}) =< s,t; s^2, (st)^3 >$. In more detail, if F is the free group on s, t and

ϕ is the homomorphism of F onto $PSL(2,\mathbb{Z})$ defined by

$$s \to \pm \begin{pmatrix} 0,1 \\ -1,0 \end{pmatrix},$$

$$t \to \pm \begin{pmatrix} 1,1 \\ 0,1 \end{pmatrix}.$$

Then the kernel of ϕ is the smallest normal subgroup of F containing s^2 and $(st)^3$. We often abuse notation by setting

$$s = \pm \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix};$$

$$t = \pm \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

The following section describes two free subgroups of $PSL(2,\mathbb{Z})$ from the point of combinatorial group theory view.

5.2.3 Free subgroups of $PSL(2,\mathbb{Z})$

Considering a group G presented as follows: $G =< s,t; s^2, (st)^3 >$. Then $T = gp(t, a = sts)$ is free on t and a. It follows that $H = gp(t, a^{-1}ta, a^{-2}ta^2, a^{-3}ta^3, a^{-4}ta^4, ..., a^{-n}ta^n)$ is also free on the group generators(see the appendix). Actually any torsion-free subgroup is free. Hence randomly generating matrices will almost always produce free subgroups [38].

In view of the matrix representation given for $PSL(2,\mathbb{Z})$, let $t = \pm \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, $a = \pm \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$. ThenT is a subgroup of $PSL(2,\mathbb{Z})$ and it is free on t and a.

Similarly we can identify H with $gp(\pm \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \pm \begin{pmatrix} 2 & 1 \\ -1 & 0 \end{pmatrix}, ...)$

Because n is an arbitrary number, we can construct a free matrix group with any number of group generators.

5.2.4 Free subgroups of $SL(2,\mathbb{Z})$

In the above section, we introduced a method to construct free subgroups of $PSL(2,\mathbb{Z})$. If a set of matrices $\{\pm M_1, \pm M_2, ..., \pm M_n | \pm M_i \in PSL(2,\mathbb{Z})\}$ freely generate a free group, then the corresponding set of matrices $\{M_1, M_2, ..., M_n | M_i \in SL(2,\mathbb{Z})\}$ also freely generate a free subgroup of $SL(2,\mathbb{Z})$. This follows from the fact that the canonical homomorphism from $SL(2,\mathbb{Z})$ onto $PSL(2,\mathbb{Z})$ will map a reduced word in $M_1, M_2, ..., M_n$ to a reduced word in $\pm M_1, \pm M_2, ..., \pm M_n$. Since such a reduced word in $PSL(2,\mathbb{Z})$ is not 1, the same is true of the original word in $SL(2,\mathbb{Z})$.

Recall that in our public-key cryptographic scheme, we use a free matrix group with n generators. The above group construction provides a simple way to implement our encoding scheme using an arbitrary number of generators. In the following section, we describe three methods to construct new matrix groups from existing matrix groups.

5.3 Matrix group constructions using isomorphisms

In this section, our objective is to construct some new matrix groups from existing ones. Our purpose is to make the new group more complicated. We will not formally discuss the complexity of groups. Instead, we provide a lot of flexible constructions of various groups. When this is applied to our protocol, attackers will not know which groups are being used. The details of scheme will be discussed in Chapter 7.

5.3.1 Constructing new matrix groups by increasing their sizes

Our first method is to build new matrix groups by increasing their sizes. As we noticed, there doesn't exist any algorithm to decompose a matrix in $SL(3,\mathbb{Z})$ or $SL(4,\mathbb{Z})$ into a product of its group generators. We believe that a higher dimension matrix is more complex than a lower dimension one.

Given a free group G_1 freely generating by a set of $m \times m$ matrices over ring R so that every element in the set takes the following form: $\begin{pmatrix} a_{11} & a_{12} & ....a_{1m} \\ a_{21} & a_{22} & ....a_{2m} \\ \dots & & \\ a_{m1} & a_{m2} & ....a_{mm} \end{pmatrix}$, we can increase the matrix group dimension by mapping any element in the set to the

following nXn matrix form:

$$\begin{pmatrix}
1 & 0 & ... & 0 & 0 & ... & 0 & 0... & 0 & 0 \\
0 & 1 & ... & 0 & 0 & ... & 0 & 0... & 0 & 0 \\
... & & & & & & & & & \\
0 & 0 & ... & 1 & 0 & ... & 0 & 0... & 0 & 0 \\
0 & 0 & ... & 0 & a_{11} & .... & a_{1m} & 0... & 0 & 0 \\
0 & 0 & ... & 0 & a_{21} & .... & a_{2m} & 0... & 0 & 0 \\
... & & & & & & & & & \\
0 & 0 & ... & 0 & a_{m1} & .... & a_{mm} & 0... & 0 & 0 \\
0 & 0 & ... & 0 & 0 & ... & 0 & 1... & 0 & 0 \\
... & & & & & & & & & \\
0 & 0 & ... & 0 & 0 & ... & 0 & 0... & 1 & 0 \\
0 & 0 & ... & 0 & 0 & ... & 0 & 0... & 0 & 1
\end{pmatrix}$$

In another words, we introduce one identity matrix in the left-top and right-bottom respectively. It is obvious that the new matrix group is also a free group.

For example, suppose G is a free group freely generated by the following two 2×2 matrices:

$A = \begin{pmatrix} a1 & b1 \\ c1 & d1 \end{pmatrix}$ and $B = \begin{pmatrix} a2 & b2 \\ c2 & d2 \end{pmatrix}$.

Replace A with C and replace B with D, where

$$C = \begin{pmatrix} a1 & b1 & 0 & 0 \\ c1 & d1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix};$$

$$D = \begin{pmatrix} a2 & b2 & 0 & 0 \\ c2 & d2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Then C and D will also freely generate a free group isomorphic to G.

5.3.2 Construct a matrix group by matrix transformations

The second method is to construct new matrix groups from existing ones with the same size. The construction is based on the following lemma:

Lemma 1 *Given a group G, if a free subgroup $H \subseteq G$ is freely generated by $\{g_1, g_2, ..., g_n\}$ and y is chosen from G, then the set $\{y^{-1}g_1y, y^{-1}g_2y, ..., y^{-1}g_ny\}$ freely generates a new free group H' and H' is isomorphic to H.*

Apply this to the matrix representation, we have the following method to construct a new matrix group.

Given any $n \times n$ free matrix group $H =< g_1, g_2, ...g_n >$, we choose a $Y \in G$ and construct a new matrix group

$$H' =< Y^{-1}g_1Y, Y^{-1}g_2Y, ..., Y^{-1}g_nY > .$$

By the above lemma, the new group H' is also a free group.

5.3.3 Construct matrix groups by free group isomorphisms

Our third method is based on the following definition and theorem:

Definition 15 *Consider a free group F freely generated by a set of generators X,*

1. *For any x in X, let α_x be the automorphism carrying x into x^{-1} and leaving X-x fixed.*

2. *For any x, y in X, where $x \neq y$, let β_{xy} be the automorphism carrying x into xy and leaving X-x fixed.*

Theorem 2 *Let Aut(F) be the group of all automorphism of the free group F and let $Aut_f(F)$ be the group generated by all of its automorphisms composed by α_x and β_{xy}. Then $Aut_f(F) = Aut(F)$ if F is finitely generated [62].*

Now suppose that $M_1, M_2, ..., M_n$ freely generate a group G of matrices. We can generate an automorphism α of the free group on $M_1, M_2, ..., M_n$ using the automorphism described in Definition 15. We will make use of Theorem 2 in Chapter 7 where we will discuss a new version of our public key cryptography.

5.3.4 Further research on group constructions

As we mentioned, our zero-knowledge doesn't require that its underlying group is a free group. We also can use traditional group construction methods to form new groups, such as direct products, semidirect products, free products, amalgamated products, HNN extensions and factor groups.

The following issues but not limited to these issues still need further research:

1) What is the complexity relation between new groups and existing ones? The reference [66] gives some surveys about this issue.

2) As for group constructions, some programming algorithms are needed for our research purposes. Specifically, how can we use a few parameters to flexibly provide a large variety of groups?

3) The specific protocol related problem should be addressed. In our case, the decomposition algorithm should be addressed for newly constructed groups.

5.4 Conclusion

This chapter examines relations among group $GL(2,\mathbb{Z})$, $SL(2,\mathbb{Z})$ and $PSL(2,\mathbb{Z})$. We introduce presentation for $PSL(2,\mathbb{Z})$, matrix representation for $PSL(2,\mathbb{Z})$ and its free subgroups. We also informally discuss presentation, matrix generators and its subgroups for $SL(2,\mathbb{Z})$. We claim that these subgroups can be used to implement our public-key encoding protocols. We describe three methods to construct new matrix groups from existing ones.

We emphasize that most of our methods are based on non-abelian infinite groups which are different from other traditional cryptography schemes. Our purpose is to demonstrate that matrix groups can provide a good fundamental arithmetical structure to construct cryptography systems. Further research needs to be done about how to apply group constructions to create new cryptography systems.

Chapter 6

Matrix decomposition

6.1 Introduction

Let G be a group of matrices generated by $M_1, M_2, ..., M_n$ and matrix $M \in G$, it is not easy to express M in terms of these generators. We refer to the process of writing in terms of $M_1, M_2, ..., M_n$ as a decomposition algorithm if it exists. In general, such algorithm doesn't exist (e.g. see [66]).

At first, we introduce a decomposition algorithm for $PSL(2, \mathbb{Z})$. We will refer to this algorithm as the $PSL(2, \mathbb{Z})$ decomposition algorithm. Then we use the Reidemeister-Schreier Rewriting method to decompose subgroups of $PSL(2, \mathbb{Z})$. We refer to this method as the Reidemeister-Schreier matrix decomposition algorithm.

6.2 Decomposition algorithms for the groups $PSL(2,\mathbb{Z})$ and $SL(2,\mathbb{Z})$

In Chapter 5, we claim that the group $PSL(2,\mathbb{Z})$ can be generated by the following two matrices:

$$s = \pm \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix},$$

$$t = \pm \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Given a matrix $T \in PSL(2,\mathbb{Z})$, our objective is to rewrite it in terms of the generators: s, t. At first we will talk about a decomposition algorithm for $PSL(2,\mathbb{Z})$ followed by a decomposition algorithm for $SL(2,\mathbb{Z})$.

6.2.1 Decomposition algorithm for the group $PSL(2,\mathbb{Z})$

This can be described in the following manner. Let s and t be defined as above and let

$$T = \pm \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

be in $PSL(2,\mathbb{Z})$. Then

$$sT = \pm \begin{pmatrix} c & d \\ -a & -b \end{pmatrix} = \pm \begin{pmatrix} -c & -d \\ a & b \end{pmatrix}.$$

$$t^k T = \begin{pmatrix} a + kc & b + kd \\ c & d \end{pmatrix}.$$

Therefore by multiplying on its left by s and then by an appropriate t^k we can make the lower left entry of T positive and smaller. Continuing in this way we find that

$$t^{k_1} s t^{k_2} s ... t^{k_n} T = \pm \begin{pmatrix} \alpha & \gamma \\ 0 & \beta \end{pmatrix}.$$

Since $\alpha\beta = 1$ this implies that the right hand side

$$\pm \begin{pmatrix} \alpha & \gamma \\ 0 & \beta \end{pmatrix} = \pm \begin{pmatrix} 1 & \gamma \\ 0 & 1 \end{pmatrix} = t^{\gamma}.$$

This then expresses the matrix T as a word $W(s,t)$.

This algorithm also proves that $PSL(2,\mathbb{Z})$ can be generated by s and t.

6.2.2 Decompose $SL(2,\mathbb{Z})$ using the $PSL(2,\mathbb{Z})$ decomposition algorithm

Given a matrix $M \in SL(2,\mathbb{Z})$, we view M as an element of $PSL(2,\mathbb{Z})$ and work now on $PSL(2,\mathbb{Z})$. Notice that

$$M \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} M$$

and

$$\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Given a matrix T' in $SL(2,\mathbb{Z})$, it can be decomposed into a product of $SL(2,\mathbb{Z})$ generators as follows:

1. Transform the matrix T' into corresponding form T of $PSL(2,\mathbb{Z})$ matrix.
2. Decompose T into a product of $PSL(2,\mathbb{Z})$ generators.
3. Transform the product of $PSL(2,\mathbb{Z})$ generators into corresponding $SL(2,\mathbb{Z})$ form.
4. If necessary, put $\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$ in front of the product.

Given a subgroup of $PSL(2,\mathbb{Z})$, if each of its elements can be decomposed into a product of a given set of generators of that subgroup. Then we can use the above algorithm to decompose any element in the corresponding subgroup of $SL(2,\mathbb{Z})$.

6.3 Decomposition algorithm for the elements of subgroups of $PSL(2,\mathbb{Z})$

As discussed in Chapter 5, it is possible to construct a free group on any number of free generators. This section uses the Reidemeister-Schreier Rewriting Process to decompose an element in a given subgroup of $PSL(2,\mathbb{Z})$ as a product of its given generators. At first, we describe the Reidemeister-Schreier theorem and the Reidemeister-Schreier rewriting process. Then we will apply this method to decompose subgroups of $PSL(2,\mathbb{Z})$.

6.3.1 Introduction to the Reidemeister-Schreier method

Definition 16 *Let G be a group and $H \subseteq G$. Then a complete set of representatives of the right cosets Hg of H in G is a set R consisting of one element from each coset. The element in R coming from the coset Hg is termed the representative of Hg or sometimes the representative of any element x in Hg. If $1 \in R$, R is termed a right transversal of H in G.*

Given an element $x \in G$ and a right transversal, we denote the representative of x by $\overline{x}$.

Theorem 3 *Let G be a group generated by a set X, $H \subseteq G$ and let S be a right transversal of H in G. Then*

1. $H = gp(\delta(s,x) = sx\overline{sx}^{-1} | s \in S, x \in X)$.

2. *Given a set of R of defining relators for G, then H can be defined by the set of relators of the form* $\{\rho(srs^{-1})|s \in S, r \in R\}$ *where* $\rho(srs^{-1})$ *is a word in the generators* $\delta(s,x)$ *of H.*

To prove the above theorem, the interested reader can refer to [10].

6.3.2 The Reidemeister-Schreier rewriting process

Let G be a group generated by X, $H \subseteq G$ and let S be a right transversal of H in G. We can find a presentation for the group H from the presentation for G according to the above theorem.

Suppose $w = a_1a_2...a_m \in H$ ($a_1, a_2, ...a_m \in X \bigcup X^{-1}$), we can also rewrite w as a product $\delta(s,x)$ as follows:

1. Put S_i=representative in S of $w_i = a_1a_2...a_i$ and denoted by $\overline{w_i}$, so $w_iH = \{w_ih|h \in H\}$.
2. Replace a_i by $S_{i-1}a_i\overline{S_{i-1}a_i}^{-1}$ if $a_i \in X$;
3. Replace a_i by $(S_ia_i^{-1}\overline{S_ia_i^{-1}}^{-1})^{-1}$ if $a_i \in X^{-1}$.

The resultant expression rewrites w as a product of terms which take the form $sx\overline{sx}^{-1}$ or $(sx\overline{sx}^{-1})^{-1}$.

This process is referred as the rewriting method of Reidemeister-Schreier.

6.3.3 Decomposition of elements in given subgroups of $PSL(2,\mathbb{Z})$

In the following section, we use the Reidemeister-Schreier rewriting process to decompose elements in given subgroups of $PSL(2,\mathbb{Z})$.

Example 1:

let G be presentation $< s,t; s^2, (sts)^3 >$ for $PSL(2,\mathbb{Z})$ and $H =< t, a = sts >$($\subset PSL(2,\mathbb{Z})$).

$$a = sts = \pm \begin{pmatrix} -1 & 0 \\ 1 & -1 \end{pmatrix}.$$

$$t = \pm \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

The right transversal S of $H \in PSL(2,\mathbb{Z})$ is $\{1, s\}$. To see this,

1. s is not in H. So s is in a different coset than 1.

2. Any member of H will take form $t^{i_0} s t^{i_1} s t^{i_2} ... s t^{i_n}$.

   (a) if $i_1 > 0$, we transform st^{i_1} as $st^{i_1} = stsst^{i_1-1} = (sts)^{i_1} s$.

   (b) if $i_1 < 0$, we transform st^{i_1} as $st^{i_1} = st^{-1}sst^{i_1+1} = (sts)^{-1} st^{i_1+1} = (sts)^{i_1} s$.

3. Repeat the above step for $i_2, ..., i_n$.

Now given a matrix M in H, we can use the following procedure to decompose it into a product of generators of H:

1. Decompose M as a product of $PSL(2,\mathbb{Z})$ matrix generators.

2. Map $PSL(2,\mathbb{Z})$ matrix generators to its combinatorial symbols.

3. Use the Reidemeister-Schreier rewriting process to rewrite a $PSL(2,\mathbb{Z})$ generator product into a H generator product.

4. Map H combinatorial symbols into matrix generators.

Example 2: The right transversal S of $H =< a, bab^{-1}, ..., b^{n-1}ab^{1-n}, ... >$ in $G =< a, b >$ is:

$$S = \{b^i | i \geq 0\}.$$

In fact, H is the kernel subgroup of ϕ where $\phi : G \to C =< x >$ by $a \to 1, b \to x$.

Example 3: Given a word $babab^{-2}abab^{-1} \in H$, rewrite it as a product of H generators:

1. Rewrite b as $1b\overline{1b}^{-1} = 1$.
2. Rewrite ba as $ba\overline{ba}^{-1} = bab^{-1}$.
3. Rewrite bab as $bab^{-1}b\overline{b^2}^{-1} = bab^{-1}$.
4. Rewrite $baba$ as $bab^{-1}b^2a\overline{b^2a}^{-1} = bab^{-1}b^2ab^{-2}$.
5. Rewrite $babab^{-1}$ as $bab^{-1}b^2ab^{-2}(b\overline{bbb}^{-1})^{-1} = bab^{-1}b^2ab^{-2}$.
6. Rewrite $babab^{-2}$ as $bab^{-1}b^2ab^{-2}(b\overline{bbb}^{-1})^{-1} = bab^{-1}b^2ab^{-2}$.
7. Rewrite $babab^{-2}a$ as $bab^{-1}b^2ab^{-2}1a\overline{1a} = bab^{-1}b^2ab^{-2}a$.
8. Rewrite $babab^{-2}ab$ as $bab^{-1}b^2ab^{-2}1b\overline{1b} = bab^{-1}b^2ab^{-2}a$.
9. Rewrite $babab^{-2}aba$ as $bab^{-1}b^2ab^{-2}ba\overline{ba}^{-1} = bab^{-1}b^2ab^{-2}bab^{-1}$.
10. Rewrite $babab^{-2}abab^{-1}$ as $bab^{-1}b^2ab^{-2}bab^{-1}(b\overline{bbb}^{-}1)^{-}1 = bab^{-1}b^2ab^{-2}bab^{-1}$.

Given a matrix in H, we can decompose it as a product of G generators. Then use the Reidemeister-Schreier rewriting process to transform the product into a product of H generators.

6.4 Conclusion

This chapter describes two decomposition methods for matrix groups: the decomposition algorithm for $PSL(2, \mathbb{Z})$ and the Reidemeister-Schereier rewriting method. Combined with isomorphism construction methods that we introduced in Chapter 5, we also can apply the $PSL(2, \mathbb{Z})$ decomposition algorithm to decompose high dimension matrix groups. The Reidemeister-Schereier method can be used to decompose matrix subgroup H as long as the original group G's decomposition algorithm and the right transversal of H in G is known.

We emphasize that decomposition methods themselves cannot be used to construct public-key cryptographic system. Some of decomposition problems can be solved by Stallings folding method [54], [79], [80], [81]. Combined with random polynomial matrix generations or isomorphism constructions, the decomposition can provide a platform to build cryptographic systems. As far as we noticed, this is the first time to apply these two decomposition algorithms in cryptography systems.

Chapter 7

Public-key cryptographic system based on isomorphisms

7.1 Introduction

Two groups, S and T, are isomorphic if and only if there exists a mapping $\phi : S \to T$, which satisfies the following three properties:

1. If s_1, $s_2 \in S$ and $\phi(s_1) = \phi(s_2)$, then $s_1 = s_2$.

2. For any $t \in T$, these exists $s \in S$ such that $\phi(s) = t$.

3. If $\phi(s_1) = t_1$ and $\phi(s_2) = t_2$, then $\phi(s_1 s_2) = t_1 t_2$.

Given two groups G_1 and G_2, the isomorphism problem is to find an algorithm to tell whether G_1 and G_2 are isomorphic.

In combinatorial group theory, people are mostly interested in three problems first raised by Dehn in 1911: the word problem, the conjugacy problem and the isomor-

phism problem. The isomorphism problem is the hardest one. In particular, there is no algorithm to decide whether or not any finitely presented group is free. This leads us to investigate some cryptographic applications by using various isomorphisms of certain free groups to other free groups.

Firstly, this chapter introduces a method to construct matrix groups isomorphic to free groups and proposes a rewriting algorithm for the newly constructed groups. After that, we use this decomposition algorithm to build a new version of our public-key cryptographic system. Finally, we use matrix groups over polynomial rings to strengthen our proposed scheme.

7.2 Isomorphic matrix group construction

In Chapter 5 we have introduced three methods to construct free matrix groups. In this section we are going to combine these three methods together to construct new matrix groups. The construction algorithm is as follows:

1. Take an initial matrix group $H \subset SL(2, \mathbb{Z})$ as in Chapter 5.

2. Randomly use one of three construction methods in section 5.3 to obtain a new matrix group $G^{'}$.

3. Repeat step 2) arbitrary number of times.

The final resulting matrix group is isomorphic to the original matrix group. This is true because the isomorphic property is transitive. In another words, if G_1 is isomorphic to G_2 and G_2 is isomorphic to G_3, then G_1 is isomorphic to G_3.

7.3 Decomposition for isomorphic groups

Suppose that we are given a pair of free groups F and G and an isomorphism ϕ from F to G. If X is a free set of generators of F and $f \in F$, then $\phi(f)$ can be expressed in terms of any free set Y of generators of G. If we take $g = \phi(f)$, then we get one expression for $\phi(f)$. But if we choose a different set of free generators for G, we get a different expression for $\phi(f)$. We shall take advantage of this idea and refer to the various processes for decomposing $\phi(f)$ as the decomposition problem. In fact this allows us to encrypt information in a complicated way, which we will discuss in due course.

7.4 Public-key cryptographic system based on isomorphisms

Now we are ready to build our new version of a public-key cryptographic system. As usual, we introduce our scheme in three parts: key setup, encryption and decryption.

7.4.1 Key setup

1. As we described in section 7.2, Alice constructs a free matrix group G_n from an existing free group G_0 whose rewriting algorithm is known. The new free group will be represented by its matrix generators and every generator is produced by the construction algorithm.

2. Publish the matrix generators in the group G_n, which is our ”alphabet”.

7.4.2 Encryption

Multiply the letters(matrix generators) and send over the resulting matrix $M^{'}$.

7.4.3 Decryption

1. Based on the isomorphism construction steps in section 7.2, Alice transforms $M^{'}$ into a matrix M in G_0. This can be done by carrying out every inverse construction step in reversing order.

2. Use the rewriting algorithm introduced in Chapter 6 to rewrite M as a product of G_0 generators.

3. Transform the product of G_0 generators to the product of G_n generators by following the construction steps for every G_0 generators.

The correctness of the whole system can be easily verified and the decoder Alice has two secrets to distinguish herself from attackers. Firstly, only Alice can reverse the construction steps because only she knows those chosen construction steps. Secondly, nobody but Alice knows the original group G_0 while everyone knows the group G_n. In this sense, we say that the whole scheme hides the underlying groups. Specifically, we believe that it is hard to figure out the underlying group structure giving a few complicated matrix generators. To demonstrate our scheme, we have the following example:

7.5 An example of public-key scheme based on the isomorphisms

7.5.1 Key setup

Alice chooses $G_0 =< \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} >$. She uses the following methods to construct the group G_3 and published generators of G_3.

1. Extend the dimension of G_0 and obtain $G_1 =< \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} > .$

2. Conjugate each of the generators in G_1 by $M = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ and obtain

   $G_2 =< \begin{pmatrix} 0 & -1 & -1 \\ 2 & 3 & 2 \\ 1 & 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} > .$

3. Use an automorphism construction α to obtain $G_3 =< \begin{pmatrix} -1 & -2 & -1 \\ 4 & 5 & 2 \\ 2 & 2 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} >$

   . by replacing $\begin{pmatrix} 0 & -1 & -1 \\ 2 & 3 & 2 \\ 1 & 1 & 1 \end{pmatrix}$ with $\begin{pmatrix} 0 & -1 & -1 \\ 2 & 3 & 2 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$ and keeping

$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$ unchanged.

7.5.2 Encryption

Bob composes a message as $\begin{pmatrix} -1 & -2 & -1 \\ 4 & 5 & 2 \\ 2 & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -2 & -3 & -1 \\ 6 & 7 & 2 \\ 3 & 3 & 1 \end{pmatrix}$ and sends this matrix N to Alice.

7.5.3 Decryption

Alice decodes Bob's message as follows:

1. At first, she carries out every inverse of the isomorphism construction steps in a reversing order. She keeps N unchanged for the automorphism step because that only changes the generator set. Then she applies conjugation by M^{-1} to N to obtain $\begin{pmatrix} 1 & 0 & 0 \\ 0 & 4 & 1 \\ 0 & 3 & 1 \end{pmatrix}$. Finally she reduces the dimension by removing the top row and the left-most column yielding $\begin{pmatrix} 4 & 1 \\ 3 & 1 \end{pmatrix}$.
2. She then decomposes the matrix $\begin{pmatrix} 4 & 1 \\ 3 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$.

3. The she follows the construction step for each generator resulting in the message Bob has. The concrete steps are as follows:

$$\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

$$\Longrightarrow$$

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

$$\Longrightarrow$$

$$\begin{pmatrix} 0 & -1 & -1 \\ 2 & 3 & 2 \\ 1 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\Longrightarrow$$

$$\begin{pmatrix} -1 & -2 & -1 \\ 4 & 5 & 2 \\ 2 & 2 & 1 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

The final result is the message Bob composed.

7.6 Conclusion

In a private-key cryptographic system, the relation between clear texts and cipher texts is an isomorphism between two string structures [1], [23], [86]. This is also true for some public-key cryptographic systems, such as RSA, Diffie-Hellman. The

contribution of this thesis is to introduce a method to flexibly construct isomorphisms and use these isomorphisms to construct cryptographic systems.

Specifically, this chapter describes an algorithm to flexibly construct new groups and a method to rewrite any word in the new group into a product of generators. Then we make use of the new matrix groups to construct public-key cryptographic systems. We also propose two methods (details in the appendix) to enhance this system using polynomial rings. Different from other traditional cryptographic systems, we argue that our scheme hides the underlying groups.

In Chapter 10 we will discuss how to apply isomorphisms into zero-knowledge schemes.

Chapter 8

Random polynomial generation

8.1 Introduction

In Chapter 3 we have proposed our public-key cryptographic system using matrix groups over polynomial rings. In Chapter 4 we have described a zero-knowledge scheme based on matrix groups. In Chapter 7 we used matrices over polynomial rings to strengthen our public-key cryptographic scheme based on the use of various isomorphisms. All of these protocols have a common underlying problem: How to generate a system of multi-variable equations which has solutions of a particular kind to be used later. The chapter is devoted to solve this problem. We will make two assumptions:

1) Little is known in general about solutions of multi-variable equations. The theoretical complexity involved is beyond the scope of this thesis and the interested readers can refer to [22], [26], [53].

2) We can use an ideal coin-flip system to generate bits randomly [61], [68], [71].

Hence, we can generate a random sequence of bits. It is obvious that we also can generate a random integer under this assumption.

First of all, this chapter describes an algorithm to randomly generate a system of equations. Secondly, an example is provided for our algorithm. The chapter also proposes a method to strengthen our previous scheme based on matrix groups over polynomial rings.

8.2 Random polynomial generation algorithm

In this chapter our goal is to randomly generate arbitrary number of equations. In order to do so, we choose a finite set of variables and a finite set of monomials. The left-hand side is a sum of monomials and we call the sum of its variable exponents its index. The right-side of each equation is a constant. Besides these equations, we need to ensure that we can generate specific solutions to these equations.

8.2.1 Description of the algorithm

At the outset we choose a finite set of variables, a set of possible monomials and randomly generate a variable assignment.

Secondly, the algorithm is to repeat the following steps to generate the designed number of equations.

1. Randomly generate coefficients for all the chosen monomials. Sometimes it is necessary to control the computational complexity. One way to achieve this is to manage the distribution of coefficients.

2. Calculate the value of the polynomial by replacing the variables by their assignments.

3. Add a constant monomial as defined by the equation.

8.2.2 Example of the algorithm

Assume our aim is to randomly generate an equation with 4 variables where the right side is 2. Suppose that the monomials involved are $x_2, x_3, x_1x_2, x_2x_4, x_2^2, x_3^2, x_4^2$. We now randomly generate four integers, say 45, -13, 29, 87. We put $x_1 = 45$, $x_2 = -13$, $x_3 = 29$, $x_4 = 87$. We also randomly generate coefficients for monomials as 12, -5, 24, -67, 3, -34, 31. Our random polynomial is then

$$12x_2 - 5x_3 + 24x_1x_2 - 67x_2x_4 + 3x_2^2 - 34x_3^2 + 31x_4^2 - 267987.$$

If we substitute variables with the given values, we get the value 267989 for the polynomial. Then $12x_2 - 5x_3 + 24x_1x_2 - 67x_2x_4 + 3x_2^2 - 34x_3^2 + 31x_4^2 - 267987$ will have defined properties, i.e has value 2 on substituting variables with their assignments.

By modifying the above method, we introduce the following strengthening version of our public-key cryptographic system.

8.3 Equivalence class of polynomials

Definition 17 *An index of monomial is the sum of its variable exponents.*
We also define the index of an equation as the maximum indices of the monomials involved.

Given n variables x_1, x_2, ..., x_n, we can modify the algorithm in section 8.2 and make the generated system of equations satisfy the following conditions:

1. The index of any equation is the same as the indices of the other ones.

2. Every equation has and only has one monomial whose index equals the index of the equation. We call this monomial the leading monomial of the equation.

3. The leading monomial of the ith equation has form of x_i^m, where m is the index of the equation.

The following system of three equations with three variables is an example of our designed equations.

$$f_1(x_1, x_2, x_3) = x_1^4 - x_x^2 x_2 + x_2 x_3 - x_3^3 - 6 = 1$$

$$f_2(x_1, x_2, x_3) = x_2^4 - x_1^2 x_3 - x_1 x_2^2 + x_3^2 + 17 = 1$$

$$f_3(x_1, x_2, x_3) = x_3^4 - x_1^3 - x_2^2 x_3 - x_2^3 - 204 = 1$$

In all our proposed public-key schemes based on matrix groups over polynomial rings, the encryption algorithm is always just to multiply the polynomial matrices. If we generate a system of equations as above and take m to be the index of polynomials. Then we can use the following encryption algorithm:

1. Use the published ”alphabet” to compose the message.

2. Multiply all matrix generators to obtain a matrix M.

3. For every entry of M, if that entry has a monomial in which a variable's exponent is larger than m-1 and the variable involved is x_j, then the equation

satisfied by x_j^m allows us to replace x_j^m by a polynomial of smaller index. E.g. In the case where j=2 and $x_2^4 - x_1^2x_3 - x_1x_2^2 + x_3^2 + 17 = 1$, then we replace x_2^4 by $x_1^2x_3 + x_1x_2^2 - x_3^2 - 17 + 1$. We call this whole step the reducing step.

4. We repeat the above reducing steps until no actions are possible.

The above encryption algorithm doesn't require us to change its peer decrypting algorithm because the procedure doesn't change the value of polynomials with variable assignments. In another words, if we substitute the variables in reduced and unreduced polynomials with our generated variable assignments, they will result in the same value and this is the critical point of decryption algorithm.

At the same time, the index of a polynomial will be decreased by at least 1 in every reducing step. This guarantees that our algorithm will not continue for ever. The advantage for these reduced forms is that we can control the complexity of polynomials. Specifically, the index of polynomials will be no more than $(m-1) \times n$ where n is the number of variables.

8.4 Conclusion

This chapter solves the previous pending question: How to randomly generate a system of equations and a matrix over polynomial rings? This chapter describes an algorithm to accomplish this and uses an example to demonstrate the procedure. The second part of this chapter proposes a method to manage the complexity of polynomials and uses this method to make our public-key cryptographic system stronger. The further research can be done on the following issues:

1. How to evaluate the complexity of a randomly generated system of equations?

2. How to factor a polynomial [41], [77] and how hard is it to factor a polynomial?

3. How to use unsolvable equations to construct cryptographic systems?

Chapter 9

Extensions of the encoding scheme

9.1 Introduction

It is believed that it is dangerous for a cryptographic system to depend on only one scheme [11], [12], [37], [51], [52]. This chapter discusses some problems of combining our scheme with others and proposes protocols by inserting useless messages among original messages. Because our scheme is based on non-abelian groups which are different from the fundamental mathematical structures of other schemes, we believe that the combination of our scheme and other cryptographic schemes will benefit the whole system.

At first, this chapter reviews the RSA scheme which is widely studied by researchers [15], [16], [27], [67]. Secondly, we introduce a protocol to embed the RSA public-key cryptographic scheme into our public-key scheme. In the second part of this chapter, we make use of noise to strengthen our scheme. At the end of this chapter, we will talk about when noise is detectable and how to detect noise.

9.2 A public-key system using RSA and matrix groups over polynomial rings

9.2.1 An abstract of the RSA public-key scheme

RSA is a public-key cryptographic algorithm, which uses prime factorization as its one-way function. Let p, q be large primes and

$$n = (p-1) * (q-1).$$

Also define a private key d and a public key e such that d and e are relatively prime and

$$de = 1 (mod\ n).$$

Let a message be converted to a number M which is less than pq. The encryption is done by using the public key e:

$$E = M^e (mod\ pq).$$

To decode, the receiver decrypts the cipher text E using the private key d:

$$E^d = (M^e)^d = M^{ed} = M^{an+1} = M (mod\ pq) = M.$$

9.2.2 The RSA public-key scheme with matrix groups

Let's say that we have two public-key cryptographic systems. The encryption algorithms are E_1 and E_2, respectively. The decryption algorithms are D_1 and D_2, respectively. Further we assume that both clear texts and cipher texts are numbers. Given a message M, we can use E_1 and E_2 in this order to encrypt M. The decoder can use D_2 and D_1 in this order to decrypt the message.

Following the above basic idea, we use the RSA public-key scheme to strengthen our public-key scheme based on matrix groups over polynomial rings. The details of the combined scheme is as follows:

1. Key setup

   The public key: Polynomial matrix generators, the RSA public key e and pq.

   The Private Key: The variable substitution and the RSA private key d.

2. Encryption

   1) Encrypt a message M into a polynomial matrix $M^{'}$ using our public-key scheme.

   2) Use the RSA public key and the RSA encryption algorithm to encrypt every coefficient of any polynomial in $M^{'}$.

3. Decryption

   1) Receive the encrypted matrix and use the RSA decryption algorithm to decrypt every coefficient.

   2) Use the variable substitution and our proposed decomposition algorithm to decrypt the matrix.

It is possible to make some variants of the above scheme. For example, we can use different RSA keys for each entry of matrix $M^{'}$ [83], [84].

9.3 Public-key cryptographic scheme with noise

If Alice and Bob can have some private talks before their public communication, they can make an agreement on some encoding methods. For example: they can agree that they will discard the first and the third message in the future communication. In this example, we can regard these useless messages as noise. This will not affect the communication between the parties while it will mislead attackers who cannot distinguish noise from original messages.

Our scheme has a lot of flexibility in the choices of the groups and the polynomials. It is very easy to add some noise agreements as described above. The following is one such scheme with noise. Unlike the above discussion, our following scheme doesn't need private talks and so satisfies the requirement of a public-key cryptographic system.

9.3.1 Description of the scheme

Assume that we have a matrix group G and $G_1, G_2, ..., G_n$ are free subgroups of G such that $G_1 \cap G_i = 1$ for $1 \leq i \leq n$. We describe our scheme in three parts: key setup, encryption and decryption.

Key setup:

As in Chapter 3, at first we make a set of matrix generators over a carefully chosen polynomial rings for each of the subgroups: $G_1, G_2, ..., G_n$. We call the generators

of G_1 message generators and the generators of the other groups noise generators.

The public key will be the message generators and the noise generators. The public will know exactly which are the message generators and which are the noise generators.

The private key will be the variable substitution.

Encryption:

As in Chapter 3, the encoder will create his message as a sequence of the matrix generators. Before he multiplies these matrices, the encoder will randomly insert some noise generators among the message generators. The encrypted matrix M will be the product of all these matrix generators.

Decryption

1. Transform the encrypted polynomial matrix M into a matrix $M^{'}$ using the variable substitution.

2. Decompose the matrix $M^{'}$ into a product of G generators.

3. Rewrite this product of G generators into a product of the generators of $G_1, ..., G_n$. Then the decoder can retrieve the message by eliminating the noise generators. The following section discusses details about this step.

9.3.2 How to detect the noise?

One question arising from the above scheme is: Is there any way we can detect the inserted noise generators effectively and efficiently? Basically, we have two problems:

Firstly, there may exist more than one message that can be recovered from a given encrypted noisy message. For example, suppose that $G =< a, b, c >$, $G_1 =< a, c >$, $G_2 =< b >$, $G_3 =< ab >$ and $G_4 =< bc^{-1} >$. For an encrypted message abc, Alice can treat b as a noise generator, so the recovered message will be ac. Alice can also think ab as noise, then she will recover the message as c. She can also think abc=$abc^{-1}cc$ and take bc^{-1} as noise, she will get the recovered message as acc. This is a problem.

Another problem is whether the message is always recoverable. For example, if $G =< a, b >$, $G_1 =< a^{-1}ba >$ and $G_2 =< a^{-1} >$. For an encrypted message $a^{-1}ba$, the message is supposed to be noise-free. But if we treat the first part a^{-1} of $a^{-1}ba$ as noise, the message is not recovable any more because ba is not a message, nor is it noise.

Fortunately, we can add some agreements into the above scheme to solve these two problems. At first we introduce the following definition.

Definition 18 *Given a group generated by a set of generators X. A group word w = $y_1y_2...y_n$ ($y_i \in X \cup X^{-1}$) is reduced if and only if $y_i \neq y_{i+1}^{-1}$ for any $1 \leq i \leq n-1$. It follows that any word in a free group has a unique reduced form.*

Our agreements are as follows:

1. Each message generator, noise generator and message are all in reduced forms.

2. When inserting a noise generator g into a message M_1M_2 to form M_1gM_2, both M_1M_2 and M_1gM_2 should be in reduced forms.

3. No generator of a subgroup begins with a generator in another subgroup.

4. Any noise subgroup intersects the group G_1 in the identity as stated earlier.

Now given an encrypted message $a_1a_2...a_n$, we try to find the shortest subword $a_1a_2...a_i$ such that it belongs to one of subgroups $G_1, G_2, ..., G_n$. This can be done by applying the rewriting methods for the subgroups to a_1,a_1a_2,$a_1a_2a_3$, ... until one such subword belongs to one of subgroups. After the first subword recovered, we can continue the procedure to recover the rest. The following example shows this procedure:

9.3.3 Example of noise detection

Let $G =< a, b >, G_1 =< b^{-1}ab, b^{-2}ab^2 >, G_2 =< a >$. Given a word $babab^{-2}abab^{-1}$, we can use the Reidemeiser-Schreier methods for $H =< a, bab^{-1}, ..., b^{n-1}ab^{1-n}, ... >$ as Example 3 in Chapter 6 to rewrite this word as product of generators of $G_1 and G_2$:

1. Replace b with $1b\overline{1b}^{-1}$ =1. Because $b \neq 1$, b is not a member in G_1 or G_2.
2. Replace ba with $ba\overline{ba}^{-1} = bab^{-1}$. Because $ba \neq bab^{-1}$, ba is not a member in G_1 or G_2.
3. Replace bab with $bab^{-1}bb\overline{b^2}^{-1} = bab^{-1}$ and bab is not a member in G_1 or G_2.
4. Replace $baba$ with $bab^{-1}b^2a\overline{b^2a}^{-1} = bab^{-1}b^2ab^{-2}$ and $baba$ is not a member in G_1 or G_2.
5. Replace $babab^{-1}$ with $bab^{-1}b^2ab^{-2}(bb\overline{bb}^{-1})^{-1} = bab^{-1}b^2ab^{-2}$ and $babab^{-1}$ is not a member in G_1 or G_2.
6. Replace $babab^{-2}$ with $bab^{-1}b^2ab^{-2}(bb\overline{bb}^{-1})^{-1} = bab^{-1}b^2ab^{-2}$ and $babab^{-2}$ is a member of G_1.

7. Replace a with $1a\overline{1a} = a$ and a is a member of G_2.

8. Replace b with $1b\overline{1b} = 1$ and b is not a member in G_1 or G_2.

9. Replace ba with $ba\overline{ba}^{-1} = bab^{-1}$ and ba is not a member in G_1 or G_2.

10. Replace bab^{-1} with $bab^{-1}(bb\overline{bb}^{-}1)^{-}1 = bab^{-1}$ and $ba^b - 1$ is a member in G_1.

This finishes the rewriting procedure.

9.4 Conclusion

The chapter introduces a method to strengthen our public-key cryptographic system using the RSA public-key encoding scheme. It is easy to build other similar schemes like the one in section 9.1. Because our scheme is based on a different arithmetical structure from other schemes, we believe that the combination with other schemes will make the whole system stronger.

We also propose a public-key scheme with noise. As we noticed, this is the first time that noise is used in a public key scheme. The main challenge to use noise in a public-key scheme is making noise detectable. This chapter uses examples to discuss the problem and uses some predefined agreements to solve this challenge.

Chapter 10

Extension of zero-knowledge schemes

10.1 Introduction

So far, we have talked about a few variants of public-key cryptographic systems. This chapter proposes various zero-knowledge protocols using the isomorphisms and matrix groups over polynomial rings.

Since zero-knowledge was introduced by A.Shamir in 1986, a few other zero-knowledge protocols have been defined and proposed. The composition of multiple zero-knowledge schemes has been discussed in references [35], [36], [45]. The concurrent zero-knowledge schemes with time constraint [33] or without time constraint [25], [33] have been proposed by a few authors. Both theoretical scheme complexity and practical scheme complexity issues have been addressed literally [34], [56], [72]. Non-interactive zero-knowledge [6], [13], [14], [55], non-malleable zero-knowledge [7], [8], [17], [30] , [47], [74] and recently resettable zero-knowledge [19], [20], [57], [65], [73] have been investigated.

First this chapter discusses the complexity of the conjugacy problem. Then we try to use two methods to solve the conjugacy problem of matrix groups. After the investigation, we propose a few methods to improve our zero-knowledge protocols. At the end of this chapter, we will propose a non-interactive zero-knowledge scheme and a non-malleable zero-knowledge scheme using matrix groups.

10.2 A brief discussion about the complexity of conjugacy problem

In Chapter 4 we use the conjugacy problem of matrix groups to construct our zero-knowledge scheme. We have proven that our scheme is sound, complete and zero-knowledge. It is unknown how hard the conjugacy problem is for matrix groups? In this section we use two methods to informally examine the complexity of this problem.

10.2.1 Solve the conjugacy problem using the system of equations

Given a pair x,y of matrices which are known to be conjugate. The problem is to find a α such that $y = \alpha^{-1}x\alpha$. A brute force approach is to choose α to be a matrix where the entries are variables. Then in order to find α, we have to solve a system of equations. We use the matrix group $SL(2, \mathbb{Z})$ as an example to demonstrate this method.

The inverse of a square matrix A with a non-zero determinant is the adjoint matrix

divided by its determinant, this can be written as

$$A^{-1} = \frac{1}{det(A)} adj(A).$$

As for the 2X2 matrix,

$$adj(A) = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

$$det(A) = ad - bc.$$

So if $A \in SL(2, \mathbb{Z})$, then

$$A^{-1} = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

Given x=$\begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix}$ and $y = \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix}$, the conjugacy searching problem is to find an α such that $y = \alpha^{-1} x \alpha$. Assuming that $\alpha = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, we have

$$y = \begin{pmatrix} -bcx_{11} - bdx_{13} + acx_{12} + adx_{14} & abx_{11} + b^2x_{13} - a^2x_{12} - abx_{14} \\ -cdx11 - d^2x_{13} + c^2x_{12} + cdx_{14} & adx_{11} + bdx_{13} - acx_{12} - bcx_{14} \end{pmatrix} = \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix}.$$

So it is a system of four equations with four variables. The equations are so complicated that the above method does not appear to be practical for general matrix groups.

10.2.2 Solve the conjugacy problem using group theory

Another method to solve the conjugacy problem for matrix groups is to use group theory. In another words, given one matrix group G and two conjugate matrices $x, y \in G$, we can rewrite x and y as a product of G generators. If we can solve the conjugacy problem in the group G, then we can solve this conjugacy problem for matrices x and y. For example, given a free group $G =< a = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}, b = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} >$ and two matrices $x = \begin{pmatrix} 26 & 15 \\ 19 & 11 \end{pmatrix}$, $y = \begin{pmatrix} -55 & -21 \\ 241 & 92 \end{pmatrix}$, we can use the decomposition algorithm introduced in Chapter 6 to rewrite $x = ababa$ and $y = b^{-1}a^{-1}a^{-1}ababaaab$. Then it is easy to figure out that x and y are conjugate by $aab = \begin{pmatrix} 8 & 3 \\ 5 & 2 \end{pmatrix}$.

Based on the above discussion, we have two methods to improve the robustness of our zero-knowledge scheme. We can make the decomposition problem for the matrix group harder. Another choice is to make the conjugacy problem harder.

10.3 Extensions of zero-knowledge based on matrix groups

The following sections discuss how to improve our zero-knowledge protocols using isomorphisms and matrix groups over polynomial rings.

10.3.1 Zero-knowledge based on isomorphisms

In Chapter 7, we have proposed a public-key scheme based on isomorphisms. We can also use this method to construct zero-knowledge schemes. The key setup and zero-knowledge proof procedure are the same as our zero-knowledge scheme in Chapter 4. The only change is the underlying matrix group. In the new scheme, we can use any free subgroup of $SL(2, \mathbb{Z})$ and the isomorphisms discussed in Chapter 7 to build new matrix groups. We will propose a non-malleable zero-knowledge scheme based on this idea later in this chapter.

10.3.2 Zero-knowledge based on matrix group over polynomial rings

We can also use polynomial rings to make decomposing matrices harder. As our public-key scheme, we can replace every matrix entry with a polynomial. The key setup and zero-knowledge proof procedure are the same as our zero-knowledge scheme as shown in Chapter 4. The only change is the underlying matrix group. We can also use equation constraints to strengthen the scheme. The idea of equation constraints is introduced in Chapter 9.

To further demonstrate the usage of polynomial rings, we will introduce a non-interactive scheme based on matrix groups over polynomial rings later in this chapter.

10.4 Other variants of zero-knowledge

Since the zero-knowledge cryptography was introduced by A.Shamir in 1986, a few other zero-knowledge protocols have been defined and proposed. This section uses a non-malleable zero-knowledge scheme and a non-interactive zero-knowledge scheme to demonstrate how to use matrix groups.

10.4.1 Non-malleable zero-knowledge scheme

Suppose that researcher Alice has obtained a proof $P \neq NP$ and wishes to communicate this fact to professor Bob. Clearly Alice will initially do this in a zero-knowledge fashion, but professor Bob may try to steal credit for this result by calling eminent professor E and acting as a transparent prover. Any question posted by professor E to professor Bob is relayed by the latter to Alice, and Alice's answers to professor Bob are then relayed in turn to professor E.

One solution is to use the non-malleable zero-knowledge which can prevent the exact-copy attack. In our proposed scheme, we assign every party a matrix for purpose of identification. The procedure is as follows:

Private Keys:

1. Alice has a private identification key M_A;
2. Alice has a private key α for zero-knowledge proof;
3. Bob has a private identification key M_B;

Public Keys:

1. A matrix g is given out as public.

2. A second public key is $y = \alpha^{-1} g \alpha$.

3. The public key for Alice is $P_A = M_A^{-1} y M_A$.

4. The public key for Bob is $P_B = M_B^{-1} y M_B$.

All these matrices are in the matrix group G constructed using the method in Chapter 7.

Zero-knowledge Proof:

Alice wants to prove that she knows α.

The proof procedures:

1. Alice randomly generates a matrix β and sends Bob $\beta^{-1} P_A \beta$;

2. Bob randomly choose one of the following two questions:

   (a) What is the value of β?

   (b) What is the value of $\alpha M_A \beta$?

3. Alice answers the chosen question using the value of α, β, M_A.

4. Bob verifies the answer.

if Bob wants to claim that he knows the value of α, the question 2.b posted by other people will be "What is the value of $\alpha M_B \beta$". Bob can not answer this question until he knows the value of α, β.

10.4.2 Non-interactive zero-knowledge Scheme

Think of Alice and Bob as two mathematicians. After having played heads and tails for a while, Alice leaves for a long trip around the world. Alice continues her math-

ematical investigations during her trip. Whenever she wants to send Bob a message, she writes a postcard to Bob proving the validity of her identity in zero-knowledge. Notice that this is necessarily a non-interactive process. In fact, Bob couldn't answer Alice because she has no fixed address.

One of solutions is to use non-interactive zero-knowledge. We propose the following non-interactive zero-knowledge scheme based on matrix groups over polynomial rings:

Public Key:

1. Matrix group generators over polynomial rings.

2. A sequence of random polynomial matrices $m_1, m_2, ...$ in the group.

Alice's secret key:

Variable substitution to decompose the product of matrix generators.

Alice's zero-knowledge proof that she knows the variable substitution.

1. Use the variable substitution to decompose the matrix.

2. Send out the result.

It is easy for Bob to verify Alice's result, but that will not help him to figure out the variable substitution. The key to this protocol is that Alice and Bob can agree on some predetermined questions and only the secret-holder can answer these question. We use an example to illustrate this protocol.

10.4.3 Example of non-interactive zero-knowledge

Let us use the same key setup as section 3.4.1. That is to say, Alice has her public keys $\{M_1 = \begin{pmatrix} f_{11}^1 & f_{12}^1 \\ f_{21}^1 & f_{21}^1 \end{pmatrix}, M_2 = \begin{pmatrix} f_{11}^2 & f_{21}^2 \\ f_{21}^2 & f_{22}^2 \end{pmatrix}\}$ which generate a free group G. She also has her secret key {1889,5162,4551,3617,-5638,-5318,-3552,3743} to decompose any matrix in the group G into a product of M_1 and M_2. Before Alice's departure, Bob and Alice will randomly generate a series of matrices in the group G by multiplying products of M_1 and M_2. After that, whenever Alice wants to prove herself in zero-knowledge style, she will decompose a few matrices using her secret key and reveal the products to Bob.

10.5 Conclusion

This chapter uses the conjugacy in matrix groups from two aspects: Solve systems of equations and solve the conjugacy problem. Unfortunately, the conjugacy problem is solvable in some matrix groups. Hence we propose the following methods to strengthen our zero-knowledge schemes:

1) Use the isomorphisms and hide the underlying group.

2) Use matrix groups over polynomial rings to improve the robustness.

3) Use more complicated groups instead of free groups to construct zero-knowledge protocols.

This chapter also proposes a non-interactive zero-knowledge scheme and a non-malleable zero-knowledge. We use examples to demonstrate that matrix groups can be applied in advanced zero-knowledge systems.

Chapter 11

Scheme Robustness and Testing Platform

11.1 Introduction

A cryptographic system is always a fight between protocol proposers and attackers [15]. A cryptographic system is incomplete if no possible attacks have been considered. This has led us to modify our original scheme a lot of times and we have kept doing so in this thesis. We believe that this is the first try to construct cryptographic systems using matrix group over polynomial rings. It is hard to discuss all aspects of our schemes. The objective of this chapter is to informally discuss robustness of our proposed schemes and to compare our schemes with other traditional schemes.

At first this chapter examines our schemes using traditional encryption characteristics. Then we discuss robustness of systems in the quantum computation scenario. At the end of this chapter, we introduce our testing platform for our schemes.

11.2 Robustness discussion for encoding schemes

Security, key length and speed are three most important characteristics for a cryptographic system. Besides using diffusion and confusion to measure the security of our schemes, we will also discuss the robustness of our schemes from a quantum computation angle. We will use key space to estimate the key length for our schemes. Finally, we give a brief discussion about implementation efficiency of our schemes.

11.2.1 Confusion and diffusion

Two important criteria to measure an encoding system are confusion and diffusion [75]. Confusion obscures the relationship between plain texts and cipher texts. A system with good confusion should only have few redundancies and statistical patterns. For example, substitution is a way to introduce confusion. Our scheme has good confusion because the result of multiplication of two matrices is very different from the original ones. In the polynomial scenario, the encrypted matrix is very different from the start-up matrices. If we use other methods in this thesis, a better confusion result can be achieved.

Diffusion dissipates the redundancy of the plaintext by spreading it out all over the ciphertext. In another words, changing one symbol in the clear text should change its whole cipher text. The simplest way to cause diffusion is transposition. Our scheme has good diffusion because changing one matrix generator in messages will change every entry of its final encrypted matrix. When we make use of polynomial rings, changing one item of one polynomial will change every item of each matrix entry.

11.2.2 Key space and code space

The second issue is the key length. Besides scheme security, computation efficiency, the ideal size of the key depends on key space and code space. In cryptography, a system's key space refers to all possible keys that can be used to accomplish the scheme. A well-designed cryptographic system should be highly computationally expensive when trying to brute-force through all possible key values. A system's code space refers to all possible messages that can appear.

In our scheme using polynomials, the key space is all possible value substitutions. Furthermore in our scheme using isomorphisms s, the key space is all possible isomorphic groups. Our code space in each case is the matrix group generated by matrix generators. Given the good confusion and diffusion achieved by our cryptographic system, the key length can be significantly shorter than other traditional systems. Another factor to reduce the key size is the computational complexity involved for each possible key. This will be further illustrated in the next section.

11.2.3 Computational effi ciency

Traditional encryption schemes use binary logic computation to improve their computational efficiency. Due to the mathematical calculation in our scheme, it is hard to use traditional techniques to improve the computational efficiency. The major obstacles for a cryptographic system using polynomial matrices are inefficient utilization of code space, matrix computation and polynomial computation. The details are as follows:

a) Large portions of code space have not been used.

In most our schemes, we use subgroup generators to make messages. This leads to a waste of code space that is outside of the subgroup. For example: If we use $SL(2,\mathbb{Z})$, our scheme can not use any matrix whose determinant is not 1. If our scheme is based on polynomials and various isomorphisms, this waste of code space will be even more dramatic.

b) Computations with matrices over polynomial rings are very complicated [21], [58].

With the best-known matrix multiplication algorithm, one 2×2 matrix multiplication has to involve seven simple multiplications. Computation of inverse requires determinant calculation, submatrix transposition and submatrix determinant calculation. Polynomial multiplication and reduction make computation even more expensive. This computational complexity is a double-edge sword. On the one hand, this complexity makes our scheme take a long time to encrypt and decrypt messages. On the other hand, it makes attacking much harder, especially using brute force.

11.3 Robustness discussion in quantum computation

Quantum computers have gained widespread interest [18], [31] and a few physical experiments have been explored [2], [32]. Some problems of practical interest are known to be in BQP (the problem has polynomial complexity under quantum computation), but is suspected to be outside P. Currently, the following three problems are known yo be in BQP.

a) Integer factorization.

b) Discrete logarithm.

c) Simulation of quantum systems.

Most cryptography systems rely on the first two problems [49]. There is one common thing among these problems. The quantum algorithms developed for these problems are examples of procedures designed to crack a specific hidden subgroup problem [69]. Especially, we are interested in group problems which can not be reduced to hidden subgroup problems and our underlying one-way functions are not known to be solvable using quantum computation.

11.4 Testing and implementation

In order to examine our protocols from the implementation aspect, we have developed a testing environment for computing matrix over polynomial rings. The objective is to provide a platform to investigate some matrix properties and study some examples. Our calculation is symbolic by using the NTL library [76]. This limits our computational performance only to the computer hardware. The following is a brief description of our system:

Like some other mathematic software, our software uses command line mode. We have two categories of commands: configurational commands and computational commands. The configurational commands set the value of particular parameters, such as the number of variables, the maximum index of polynomials. Computational commands can randomly generate matrices and do matrix computations. The following are some examples of commands and their descriptions:

1. set variable-num 8

   (Use variables $x_1, x_2, ..., x_8$ to generate polynomials.)

2. set maximum-index 2

   (The maximum index of variables is 2.)

3. set max-random 10000

   (The maximum random integer is 10000.)

4. set-matrix M1 integer $\{1,1\}\{0,1\}$

   (Set M1 as $\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$.)

5. set-matrix M2 integer $\{2,1\}\{1,1\}$

   (Set M2 as $\begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}$.)

6. set-matrix M3 random M1

   (Randomly generate a polynomial for each entry of M1 and assign the new polynomial matrix to M3. The algorithm is given in Chapter 8.)

7. set-matrix M4 random M2

   (Randomly generate a polynomial for each entry of M2 and assign the new polynomial matrix to M4. The algorithm is given in Chapter 8.)

8. set-matrix M5 M3*M4

   (Set M5=M3*M4.)

11.5 Conclusion

This chapter examines a few characteristics of our proposed schemes based on matrix groups. We realize that computation is more complicated than other traditional schemes. On the other hand, our schemes have the following advantages:

1. Our schemes hide theirs underlying groups.

2. Our one-way functions are robust against both traditional attacks and quantum attacks.

3. Our scheme achieves good confusion and diffusion.

This chapter also describes our polynomial matrix computation system. To demonstrate the usage of our system, some simple examples have been given.

Appendix A

Cryptosystems Using Linear Groups

1. Introduction

Because of the increasing power of computing machinery, cryptosystems, both public key and classical, are becoming less secure. At the same time there is an increasing need for secure cryptosystems. This is clear from the increasing use of internet shopping, electronic financial transfers and so on.

Most common public key cryptosystems presently in use, such as the RSA algorithm, Diffie-Hellman, and elliptic curve methods are number theory based and hence depend on the structure of abelian groups. In particular these depend for the most part on solving equations and representing elements in abelian groups. In view of the fact that computing machinery is getting stronger, abelian groups are too easy to understand and hence not hard enough to use for encryption. Therefore from an algebraic point of view cryptography must increasingly rely on noncommutative objects, especially nonabelian groups. The important sources of nonabelian groups that

can be used in cryptosystems are combinatorial group theory and linear group theory. In particular combinatorial group theory allows the description of group elements in terms of words in a system of generators which can then be used for encryption. Linear groups allow for encryption methods using matrices over arbitary fields. There are many public key cryptosystems based on combinatorial group theory, for example braid group cryptography (see [2],[9]). The one way functions in these systems are based on the difficulty of solving various group theoretic decision problems such as the conjugacy problem. There have also been several (sometimes successful) attacks on such systems (see [6]) so new methods must be developed. Further there have been several cryptosystems devised using linear groups and homomorphisms into linear groups (see [6],[17]). As with most cryptosystems attacks have been devised against them.

In this paper we present various suggestions for developing cryptosystems, both classical and public key, using a combination of methods from combinatorial group theory and linear groups. As a type of one way function we use the relative difficulty of the Reidemeister-Schreier rewriting process (see [13]). We first describe a general schema for the types of cryptosystems that we are developing. We then show an implementation of these basic ideas in the classical modular group M. This system is quite secure for private communication and at present we are working on ways to alter it to become a true public key system. A cryptosystem using the the extended modular group $SL_2(\mathbb{Z})$ was developed by Yamamura ([17]) but was subsequently shown to have loopholes based on hyperbolic geometry (see [6] and [14]). Several potential attacks were described by Hall,Goldberg and Schneier [8] who pointed out that any closest-point cryptosystem would be vulnerable to the same sorts of attacks.

In the paper [1], Anderson and Needham pointed out that the same key and scheme for two different messages should be avoided. Here we give an additional enhanced modular group scheme, based on our general method, which addresses these problems and attempts to avoid these possible attacks. We subsequently present some further suggestions on extended implementations of this method.

The suggestions given in this paper raise many additional questions concerning both actual implementation and security. We present these questions in the present paper. They are being addressed both experimentally and theoretically in the thesis of the third author Xu [16].

2. Linear Free Group Cryptosystems

The basic idea in using combinatorial group theory for cryptography is that elements of groups can be expressed as words in some alphabet. If there is an easy method to rewrite group elements in terms of these words and further the technique used in this rewriting process can be supplied by a secret key then a cryptosystem can be created. The simplest example is perhaps a **free group cryptosystem**. This can be described in the following manner. We will use the books by Magnus, Karrass and Solitar [13] or Baumslag [3] as standard references for material on combinatorial group theory.

Consider a free group F on free generators $x_1, ...x_r$. Then each element g in F has a unique expression as a word $W(x_1, ...x_r)$. Let $W_1, ..., W_k$ with $W_i = W_i(x_1, ...x_r)$ be a set of words in the generators $x_1, ...x_r$ of the free group F. At the most basic level, to construct a cryptosystem, suppose that we have a plaintext alphabet $\mathcal{A}$. For example suppose $\mathcal{A} = \{\dashv, \lfloor, ...\}$ are the symbols needed to construct

meaningful messages in English. To encrypt, use a substitution ciphertext

$$\mathcal{A} \to \{\mathcal{W}_\infty, ...\mathcal{W}_\|\}.$$

That is

$$a \to W_1, b \to W_2, ...$$

Then given an word $W(a, b, ...)$ in the plaintext alphabet form the free group word $W(W_1, W_2, ...)$. This represents an element g in F. Send out g as the secret message.

In order to implement this scheme we need a concrete representation of g and then for decryption a way to rewrite g back in terms of $W_1, ...W_k$. This concrete representation is the idea behind **homomorphic cryptosystems** (see the article of Grigoriev and Ponomarenko [6]).

The decryption algorithm then depends on a very important idea that we will need later known as the **Reidemeister-Schreier rewriting process** (see [13] for full details). Assume $W_1, ...W_k$ are free generators for some subgroup H of F. A **Schreier transversal** for H is a set $\{h_1, ...h_t, ...\}$ of (left) coset representatives for H in F of a special form (again see [13] for particular details). Any subgroup of a free group has a Schreier transversal. The Reidemeister-Schreier process allows one to construct a set of generators $W_1, ...W_k$ for H by using a Schreier transversal. Further given the Schreier transversal from which the set of generators for H was constructed, the **Reidemeister-Schreier Rewriting Process** allows us to algorithmically rewrite an element of H. Given such an element expressed as a word $W = W(x_1, ...x_r)$ in the generators of F this algorithm rewrites W as a word $W^\star(W_1, ...W_k)$ in the generators of H. The actual algorithm is described in detail in [13] and [3]. What is impor-

tant from the point of view of making codes secure is that given a set of generators it is easy to multiply them as strings in the whole free group. On the other hand even though the membership problem in a free group is solvable, that is it can determined if an element is in a subgroup, it is relatively difficult to determine a Schreier transversal. Although there are algorithms to accomplish this these are based for the most part on the knowledge of a homomorphic image of the subgroup and make it harder to determine a transversal given a set of generators than to rewrite using a known transversal. Schreier transversals are usually computed algorithmically from homomorphisms onto either permutation groups or linear groups. Nothing is published about the relative complexity of determining a Schreier transversal versus the complexity of then rewriting words in the given subgroups. These questions, which are important in determining the security of cryptosystems using Reidemeister-Schreier rewriting, are being explored in the thesis of Xu [16] and in [4].

An important feature of the Reidmeister-Schreier rewriting process is that to rewrite a word

$$W = x_{i_1}^{\epsilon_{i_1}} x_{i_2}^{\epsilon_{i_2}} \dots x_{i_t}^{\epsilon_{i_t}}$$

where $\epsilon_{ij} \in \{-1, 1\}$ and $i_j \in \{1, 2, \dots r\}$, the algorithm rewrites letter by letter from left to right (see [13] for complete details). We will use this feature in the implementation of our Modular Group scheme.

One of the earliest descriptions of a free group cryptosystem as well as a homomorphic version of it was in a paper by W. Magnus in the early 1970's [12]. In this paper Magnus was studying rational representations of Fuchsian Groups and non-parabolic and Neumann subgroups of the classical modular group M. Recall that

$M = PSL_2(\mathbb{Z})$. That is M consists of the 2×2 projective integral matrices:

$$M = \{\pm \begin{pmatrix} a & b \\ c & d \end{pmatrix} : ad - bc = 1, a, b, c, d \in \mathbb{Z}\}.$$

Equivalently M can be considered as the set of integral linear fractional transformations with determinant 1:

$$z' = \frac{az + b}{cz + d}, ad - bc = 1, a, b, c, d \in \mathbb{Z}.$$

Magnus proved the following theorem.

Theorem 1 [12] The matrices

$$\pm \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}, \pm \begin{pmatrix} 1 + 4t^2 & 2t \\ 2t & 1 \end{pmatrix}, t = 1, 2, 3, ...$$

freely generate a free subgroup F of infinite index in M. Further distinct elements of F have distinct first columns.

Since the entries in the generating matrices are positive we can do the following. Choose a set

$$T_1, ...T_n$$

of projective matrices from the set above with n large enough to encode a desired plaintext alphabet $\mathcal{A}$. Any message would be encoded by a word

$$W(T_1, ...T_n)$$

with non-negative exponents. This represents an element g of F. The two elements in the first column determine W and therefore g. Receiving W then determines the message uniquely.

Pure free group cryptosystems are subject to various attacks and can be broken easily (see [6] and [14]). Therefore we would like to develop other encryption methods which utilize the same ideas but not subject to the same attacks.

3. A General Scheme for Reidemeister-Schreier Based Cryptosystems

Here we suggest a general approach using a combination of combinatorial group theory coupled with group respresentations.The basic schema is as follows:

We start with a finitely presented group

$$G =< X|R >$$

where $X = \{x_1, ...x_n\}$ and a faithful representation

$$\rho : G \to \overline{G}.$$

$\overline{G}$ can be any one of several different kinds of objects - linear group,permutation group, power series ring etc.

We assume that there is an algorithm to re-express an element of $\rho(G)$ in $\overline{G}$ in terms of the generators of G. That is is $g = W(x_1, ...x_n...) \in G$ where W is a

word in the these generators and we are given $\rho(g) \in \overline{G}$ we can algorithmically find g and its expression as the word $W(x_1, ..x_n)$. We note that if this algorithm moving between G and $\overline{G}$ is too simple then the representation ρ is unnecessary, When we discuss the implementation of this schema using the Modular Group M as G these comments will become clearer.

Once we have G we assume that we have two free subgroups K, H with

$$H \subset K \subset G.$$

We assume that we have fixed Schreier transversals for K in G and for H in K both of which are held in secret by the communicating parties Bob and Alice. Now based on the fixed Shreier transversals we have sets of Schreier generators (again see [13] for full details), constructed from the Reidemeister-Schreier process for K and for H.

$$k_1, ...k_m, ... \quad \text{for } K$$

and

$$h_1, ...h_t, ... \quad \text{for } H.$$

Notice that the generators for K will be given as words in $x_1, ...x_n$ the generators of G while the generators for H will be given as words in the generators $k_1, k_2, ...$ for K. We note further that H and K may coincide and that H and K need not in general be free but only have a unique set of normal forms so that the representation of an element in terms of the given Schreier generators is unique.

We will encode within H, or more precisely within $\rho(H)$. We assume that the number of generators for H is is larger than the set of characters within our plaintext

alphabet. Let $\mathcal{A} = \{\dashv, \lfloor, \rfloor ...\}$ be our plaintext alphabet. At the simplest level we choose a starting point i, within the generators of H, and enclode

$$a \to h_i, b \to h_{i+1}, ... \text{ etc.}$$

Suppose that Bob wants to communicate the message $W(a, b, c...)$ to Alice where W ia a word in the plaintext alphabet. Recall that both Bob and Alice know the various Schreier transversals which are kept secret between them. Bob then encodes $W(h_i, h_{i+1}...)$ and computes in $\overline{G}$ the element $W(\rho(h_i), \rho(h_{i+1}), ..)$ which he sends to Alice. This is sent as a matrix if $\overline{G}$ is a linear group or as a permutation if $\overline{G}$ is a permutation group and so on.

Alice uses the algorithm for $\overline{G}$ relative to G to rewrite $W(\rho(h_i), \rho(h_{i+1}), ..)$ as a word $W^\star(x_1, ...x_n)$ in the generators of G. She then uses the Shreier transversal for K in G to rewrite using the Reidemeister-Schreier process $W^\star$ as a word $W^{\star\star}(k_1, ...k_s..)$ in the generators of K. Since K is free or has unique normal forms this expression for the element of K is unique. Once she has the word written in the generators of K she uses the transversal for H in K to rewrite again, using the Reidemeister-Schreier process, in terms of the generators for H. She then has a word $W^{\star\star\star}(h_i, h_{i+1}, ...)$ and using $h_i \to a, h_{i+1} \to b, ...$ decodes the message.

In actual implementation an additional *random noise factor* is added. We will make this explicit in the next section when we discuss the inplementation of this process using for the base group G the classical modular group $M = PSL(2, \mathbb{Z})$.

Asa mentioned in the introduction no formal work has been done on the relative complexity of rewriting using a known transversal versus that of finding a transversal

say given solely some generators. This is important for determining the security of this system. Work on this theoretical question as well as experimental work on the security of these systems is being done in the thesis of Xu [16].

4. Modular Group Schemes

We now give an example of an implementation of the general schema of the previous section using the Modular Group M. Yamamura devised a cryptosystem using the extended modular group $SL_2(\mathbb{Z})$. His method depended on the free product with amalgamation structure of $SL_2(\mathbb{Z})$ (see [5]) and the geometric action of this group on the upper half plane. Recall that M is a Fuchsian group and hence acts as a group of hyperbolic isometries on the upper half-plane considered as a model of hyperbolic geometry. The regularity of this action by subgroups was found to be a potential loophole to the Yamamura scheme (see [14]). To construct our cryptosystem we need some essential facts about M.

M has a group presentation

$$M =< x, y; x^2 = y^3 = 1 >=< a, t; a^2 = (at)^3 = 1 >$$

where

$$x = \pm \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, y = \pm \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}, a = \pm \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, t = \pm \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Further $t = xy$. Group theoretically M is a **free product** of a cyclic group of order

2 and a cyclic group of order 3, that is $M = \mathbb{Z}_2 \star \mathbb{Z}_3$. In particular this implies that x and y as defined above are an independent system of generators and hence any element $g \in M$ has an essentially *unique* expression as a word $W = W(x, y)$.

There is an algorithm based on the Euclidean algorithm which given a projective integral matrix T can rewrite it in terms of the standard generators x, y. This can be described in the following manner. Let a and t be defined as above and let

$$T = \pm \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$

be a matrix in M. Then by computation we get that

$$aT = \pm \begin{pmatrix} \gamma & \delta \\ -\alpha & -\beta \end{pmatrix} = \pm \begin{pmatrix} -\gamma & -\delta \\ \alpha & \beta \end{pmatrix}$$

$$t^k T = \pm \begin{pmatrix} \alpha + k\gamma & \beta + k\delta \\ \gamma & \delta \end{pmatrix}.$$

Therefore by multiplying on the left by a and then by an appropriate t^k we can make the lower left entry positive and smaller. Continuing we get eventually

$$t^{k_1} a t^{k_1} a ... t^{k_n} a^{\epsilon} T = \pm \begin{pmatrix} \mu & \nu \\ 0 & \eta \end{pmatrix}$$

where $\epsilon = 1$ or 0. Since $\mu\eta = 1$ this implies that the right hand side

$$\pm \begin{pmatrix} \mu & \nu \\ 0 & \eta \end{pmatrix} = \pm \begin{pmatrix} 1 & \rho \\ 0 & 1 \end{pmatrix} = t^{\rho}.$$

This then expresses the matrix T as word $W(a,t)$. Then using that $t = xy$ we get T expressed as a word $W_1(x,y)$ in the standard generators x, y.

The final necessary fact is that any torsion-free subgroup of M is actually a free group. This is an easy consequence of the free product structure (see [5]) and has been generalized. We say that M has the **torsion-free subgroup property**.

Based on these ideas and following the general outline we give a simple encryption-decryption algorithm based on the Modular group. This is close to the Yamamura method which as mentioned has been broken. We will then enhance it and modify it to handle the various attacks.

The encryption algorithm is as follows. Start with a free subgroup H of M. Suppose that $h_1, ...h_t$ is a Schreier transversal for H in M and that $W_1, ...W_k$ is a set of generators for H constructed by the Reidemeister-Schreier method from $h_1, ...h_t$. It could be that t and k are infinite. The Schreier transversal will be kept secret. Suppose that $\mathcal{A} = \{\dashv, \lfloor, ...\}$ is our plaintext alphabet. Suppose further than $k > l$ where l is the size of the plaintext alphabet $\mathcal{A}$. We then, as in the free group cryptosystem, use the substitution ciphertext $a \rightarrow W_1, b \rightarrow W_2, ...$ and so on. Given a plaintext message $W(a,b,...)$ compute $W(W_1, W_2...)$ as an element of H and then express it

as an integral matrix

$$W(W_1, W_2, ...) = T = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}.$$

The secret message sent out is T. Notice that knowledge of the Shreier transversal is not necessary for encryption.

To decipher the message the decryption algorithm proceeds as in the general schema. Suppose we obtain the coded message T given as an integral matrix. Use the standard algorithm explained earlier to express T as a word in the standard generators x, y of M. We then have

$$T = W(x, y).$$

We now use the Shreier transversal and Reidemeister-Schreier rewriting to reexpress T as a word in W_1, W_2.... Applying the inverse substitution $W_1 \to a, W_2 \to b$... now decodes the message.

As for Yamamura's method [17] there are various problems with this code (see [6] and [14]). Since M acts discontinously on the upper half plane, subgroups of finite index behave somewhat regularly relative to this geometric action. An examination of the way the messages act can be used to uncover the subgroup H. Yamamura tried to disguise H by conjugating the message with an arbitrary matrix but this could also be broken. We now try a slightly different approach.

We propose the following potential cryptosystem. It is actually a type of polyalphabetic cipher like the Alberti code (see [7]). Suppose that we have a dictionary $H_1, ...H_n$ of finite index subgroups of M. In a practical implementation we assume

that n is large. For each H_i we have a Schreier transversal

$$h_{1,i}, ...h_{t(i),i}$$

and a corresponding ordered set of generators

$$W_{1,i}, ...W_{m(i),i}$$

constructed from the Schreier transversal by the Reidemeister-Schreier process. The Schreier transversal is kept secret between the communicating parties Bob and Alice. Notice this is a cryptosystem and not a key protocol. It is further assumed that each $m(i) >> l$ where l is the size of the plaintext alphabet, that is each subgroup has many more generators than the size of the plaintext alphabet.

The subgroups in this dictionary and their corresponding Schreier transversals can be chosen in a variety of ways. For example the commutator subgroup of the Modular group is free of rank 2 and some of the subgroups H_i can be determined from homomorphisms of this subgroup onto a set of finite groups. Finding a free subgroup and a representation was described in part in [6].

Suppose that Bob wants to send a message to Alice. Bob first chooses three integers (m, q, t) where

$$m = \text{ choice of the subgroup } H_m$$

$$q = \text{ starting point among the generators of } H_m$$

for the substitution of the plaintext alphabet

$$t = \text{ size of the message unit .}$$

We clarify the meanings of q and t. Once Bob chooses m, to further clarify the meaning of q, he makes the substitution

$$a \to W_{m,q}, b \to W_{m,q+1}, ...$$

Again the assumption is that $m(i) >> l$ so that starting almost anywhere in the sequence of generators of H_m will allow this substitution. The message unit size t is the number of coded letters that Bob will place into each coded integral matrix.

Once Bob has made the choices (m, q, t) he takes his plaintext message $W(a, b, ...)$ and groups blocks of t letters. He then makes the given substitution above to form integral matrices

$$T_1, ...T_s$$

We now introduce a *random noise factor*. After forming $T_1, ...T_s$ Bob then multiplies on the right each T_i by a random determinant one integral matrix R_{T_i} (different for each T_i). The only restriction on this random matrix R_{T_i} is that there is no free product cancellation in forming the product $T_i R_{T_i}$. This can be easily checked and ensures that the free product expression for $T_i R_{T_i}$ in terms of the free product generators of the Modular group M is just the concatenation of the expressions for T_i and R_{T_i}. Next he sends Alice the integral key (m, q, t) by some public key method (RSA, Anshel-Goldfeld etc.). He then sends the message as s integral matrices

$$T_1 R_{T_1}, T_2 R_{T_2}, ...T_s R_{T_s}.$$

Hence what is actually being sent out are not elements of the chosen subgroup H_m but rather elements of random right cosets of H_m in M. The purpose of sending coset elements is two-fold. The first is to hinder any geometric attack by masking the subgroup. The second is that it makes the resulting words in the the Modular Group generators longer - effectively hindering a brute force attack.

To decode the message Alice first uses public key decryption to obtain the integral keys (m, q, t). She then knows the subgroup H_m, the ciphertext substitution from the generators of H_m and how many letters t each matrix encodes. She next uses the Modular group algorithm to express each $T_i R_{T_i}$ in terms of the standard generators x, y of M say $W_{T_i}(x, y)$. She has knowledge of the Schreier transversal, which is held secretly by Bob and Alice, so now uses the Reidemeister-Schreier rewriting process to start expressing $W_{T_i}(x, y)$ in terms of the generators of H_m. Recall that Reidemeister-Schreier rewriting is done letter by letter from left to right. Hence when she reaches t of the free generators she stops. Notice that the string that she is rewriting is longer than what she needs to rewrite in order to decode as a result of the random matrix R_{T_i}. This is due to the fact that she is actually rewriting not an element of the subgroup but an element in a right coset. This presents a further difficulty to an attacker. Since these are random right cosets it makes it difficult to pick up statistical patterns in the generators even if more than one message is intercepted. In practice the subgroups should be changed with each message.

The initial key (m, q, t) is changed frequently. Hence as mentioned above this method becomes a type of polyalphabetic cipher. Polyalphabetic ciphers have historically been very difficult to decode (see [7]).

We note that in the above Modular group scheme the use of the representation of

M in terms of integral matrices is in essence unnecessary since the algorithm to go back and forth between group elements and matrices is relatively easy. However it is the suggestion of the method that is important as it is part of the general idea of homomorphic cryptosystems (see [6]). In addition sending matrices instead of symbols is a convenient way of transmitting the code which presents a further difficulty to a purely random attacker.

Initial small experiments show that this code is quite secure for private key communications. The details and further experiments are being pursued in the thesis of Xu [16] and in [4]. The problems in extending this to a pure public key system lie in determining exactly what should be made public.

5. Some Algebraic Linear Group Extensions

As explained in the general outline in section 2, three things are necessary for the suggested potential cryptosystems; an abundant supply of free subgroups; an algorithm to go back and forth between group generators and matrices; finally Reidemeister-Schreier rewriting. Hence besides the Modular group other linear groups of different sizes over different ground rings can be used as a base group for a similar cryptosystem - provided the three ideas just mentioned can be implemented. For linear groups over fields of characteristic zero it is known that there is an abundant supply of free subgroups. For example from the theorem of Tits [15] any finitely generated linear group is either virtually solvable (has a solvable subgroup of finite index) or contains free subgroups of arbitrary rank. Hence working with free subgroups in a linear group does not pose a major restriction. What is difficult is moving between

group elements and representing matrices. As explained in the last section the use of matrices with the Modular group is unnecessary however this may not be the case with other representations.

A similar approach to the Modular group cryptosystem can be used if we take one of the Euclidean Bianchi groups Γ_d as the starting group. Recall (see [5]) that the Bianchi groups Γ_d are the groups $PSL_2(\mathcal{O}_\lceil)$ where $\mathcal{O}_\lceil$ is the ring of integers in the quadratic imaginary number field $Q(\sqrt{-d})$. If $d = 1, 2, 3, 7, 11$ these rings have a Euclidean algorithm and the corresponding groups are the Euclidean Bianchi groups. In these groups there is an algorithm, similar to that in M to express a projective matrix in terms of a standard set of generators. The subgroups, while not necessarily free, have a large subset of free generators which can be used as in the scheme of the last section. For concreteness we describe the process in the Picard group $\Gamma_1 = PSL_2(Z[i])$. Hence this is the analog of the Modular group for the Gaussian integers.

It is known ([5]) that Γ_1 can be generated by the projective matrices

$$a = \pm \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, t = \pm \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, u = \pm \begin{pmatrix} 1 & i \\ 0 & 1 \end{pmatrix}.$$

Further the same type of algorithm as in M allows us to rewrite a matrix $T \in \Gamma_1$ in terms of the generators a, t, u.

Γ_1 has been proved ([5]) to have a nice group theoretical amalgam structure that, while not as simple as M can be handled in the same manner as M. In particular

$$\Gamma_1 = G_1 \star_M G_2$$

where

$$G_1 = S_3 \star_{\mathbb{Z}_3} A_4, G_2 = S_3 \star_{\mathbb{Z}_2} D_2.$$

Here S_3 is the symmetric group on 3 symbols, A_4 the alternating group on 4 symbols and M is the Modular group.

What is important is that this amalgam decomposition gives a unique normal form for each element of Γ_1. While Γ_1 does not have the torsion-free subgroup property, that is torsion-free subgroups need not be free, any subgroup must have an amalgam structure and hence there is an independent amalgam basis that can be used as in M. The group Γ_1 is not a Fuchsian group but rather acts on 3-dimensional hyperbolic geometry. This would make a geometric attack on a Picard group code more difficult.

One can also use different representations of a free group in order to develop new cryptosystems. For example let

$$\Lambda = \mathbb{Z} << x_1, ...x_q >>$$

be the ring of integral power series in the non-commuting variables $x_1, ...x_q$. Further let h be a large positive integer. Suppose that we add the relations

$$x_1^h = x_2^h = ... = x_q^h = 0$$

to Λ giving us a new power series ring Γ. Then Magnus has proved that the elements

$$f_1 = 1 + x_1, f_2 = 1 + x_2, ...f_q = 1 + x_q$$

freely generate a free group $F = F(f_1, .., f_q)$ within Γ. The inverses of the f_i within

Γ are only known if one knows the value of h. This value of h will then become part of the secret key. Then as before we can encode any message $\mathcal{M}$ in terms of the generators of F. Alice will send out the message as

$$M' = uMv$$

where u and v are finite sums of monomials in Γ chosen so that u^{-1} and v^{-1} exist and are again finite sums of monomials. These inverses will exists because of the relations given in terms of the integer h. The integer h and the elements u, v are known only to Bob and Alice. Upon receiving the message Bob can decode to get $\mathcal{M}$ from $\mathcal{M}'$. This method and its complexity and security will be explored further in [4].

6. Some Further Ideas

There are various further extensions of the ideas of the previous sections. An idea being developed is the following (in [4]). Consider a polynomial ring $R[x_1, ...x_n]$ over an arbitrary ring R in a large number of variables. Let

$$A = \begin{pmatrix} 1 & f \\ 0 & 1 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 \\ f & 1 \end{pmatrix}$$

with $f \in R$. In most cases A, B will be generators for a free group of rank 2. Take an infinite index subgroup of $< A, B >$ to obtain a free subgroup with infinitely many

generators

$$W_1(A,B), W_2(A,B), ...W_k(A,B), ...$$

and make a plaintext alphabet substitution as in the Magnus scheme

$$a \to W_1, b \to W_2, ...$$

Now choose polynomials $f_1, f_2, f_3, f_4, g_1, g_2, g_3, g_4 \in R[x_1, ...x_n]$ and set

$$\overline{A}(x_1, .., x_n) = \begin{pmatrix} f_1 & f_2 \\ f_3 & f_4 \end{pmatrix}, \overline{B}(x_1, ...x_n) = \begin{pmatrix} g_1 & g_2 \\ g_3 & g_4 \end{pmatrix}.$$

Suppose we have chosen the polynomials above so that there is a substitution

$$\overline{r} = (r_1, ...r_n)$$

such that under this substitution

$$\overline{A}(r_1, ...r_n) = A \text{ and } \overline{B}(r_1, ...r_n) = B.$$

Given a plaintext word $W(a, b, ...)$ encode it with $W(W_1, W_2, ..)$ as before. However what is sent out is

$$W(\overline{W_1}, \overline{W_2}....)$$

where if $W_i = W_i(A, B)$ then

$$\overline{W_i} = W_i(\overline{A}, \overline{B}).$$

The receiver Alice has the secret substitution key $\overline{r} = (r_1, ...r_n)$ which she uses to rewrite

$$W(\overline{W_1}, \overline{W_2}....)$$

as

$$W(W_1, W_2, ...)$$

to decode.

A variation of this idea is to use a variable version of Magnus' original scheme. Recall from Magnus' result that the matrices

$$\pm \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}, \pm \begin{pmatrix} 1+4t^2 & 2t \\ 2t & 1 \end{pmatrix}, t = 1, 2, 3, ...$$

freely generate a free subgroup F of infinite index in M. Let

$$A(t,s) = \pm \begin{pmatrix} 1+4t^2 & 2t \\ 2t & 1 \end{pmatrix}, B(t,s) = \pm \begin{pmatrix} 1+4s^2 & 2s \\ 2s & 1 \end{pmatrix}.$$

These will then generate a free subgroup of rank 2 in the linear group $PSL_2(Z[s,t])$. Now do the substitutions as described above to send out coded messages as matrices in $PSL_2(Z[s,t])$. The secret deciphering code would be an integral pair (s_0, t_0).

7. Public Key Protocols: Group Theoretic Diffie-Hellman

Encrypting within a group can be also be used to develop a public key protocol. Braid groups have been used for public key protocols (see [2] and [9]). The following is a group theoretic version of the Diffie-Hellman public key scheme. Recall that the Diffie-Hellman method uses the difficulty of the discrete log problem coupled with the fact that exponents commute (see [10]). This can be translated into general group theory as follows. Suppose Bob wants to communicate with Alice via an open airway. They have decided to encrypt their messages within a group G and have decided on an encryption method - perhaps one described in the previous sections. Further we suppose that A is a large abelian subgroup of G which is held secret by Bob and Alice.

Suppose now that Bob wants to send the message $W \in G$ to Alice. He chooses two random elements $B_1, B_2 \in A$ and sends Alice the message (in encrypted form) B_1WB_2. Alice now chooses two random elements A_1, A_2 also in A and sends Bob back $A_1B_1WB_2A_2$. Since A is abelian

$$A_1B_1WB_2A_2 = B_1A_1WA_2B_2.$$

Further since Bob knows his chosen elements B_1 and B_2 he can multiply by their inverses to obtain A_1WA_2 which he then sends back to Alice. Since Alice knows her chosen elements A_1, A_2 she can multiply by their inverses to get the message W. It is assumed that for each message Bob and Alice would choose different pairs of

random elements of A.

References

[1] R. Anderson and R. Needham, Robustness Principle for Public Key Protocols, Crypto 95.

[2] I.Anshel,M.Anshel and D.Goldfeld, An Algebraic Method for Public Key Cryptography, Math.Res. Lett 6, Springer Verlag, 287-291, 1999.

[3] G.Baumslag, Topics in Combinatorial Group Theory, Birkhauser, 1993.

[4] G. Baumslag, B.Fine, and X.Xu, Free Group and Linear Group Public Key Cryptosystems, in progress.

[5] B.Fine, The Algebraic Theory of the Bianchi Groups, Marcel Dekker, 1990.

[6] D.Grigoriev and I. Ponomarenko, Homomorphic Public-Key Cryptosystems Over Groups and Rings, Quaderni di Matematica, 2005 to appear.

[7] P.Hoffman, Archimedes Revenge, Fawcett-Crest, 1988.

[8] C.Hall, I. Goldberg and B. Schneier, Reaction attaacks Against Several Public Key Cryptosystems, Proceedings of Information and Communications Security ICICS 99, Springer-Verlag, 1999, 2-12

[9] K.H. Ko,J.Lee,J.H. Cheon,J.W. Han, J.Kang and C.Park, New Public-Key Cryptosystem Using Braid Groups, Advances in Cryptology - CRYPTO 2000 Santa Barbara CA - Lecture Notes in Computer Science, Springer,vol 1880, 2000, 166-183, Springer Verlag.

[10] N.Koblitz, Algebraic Methods of Cryptography, Springer, 1998.

[11] R.C. Lyndon and P.E. Schupp, Combinatorial Group Theory, Springer-Verlag, 1977.

[12] W. Magnus, Rational Representations of Fuchsian Groups and Non-Parabolic

Subgroups of the Modular Group, Nachrichten der Akad Gottingen, 1973, 179-189.

[13] W. Magnus, A. Karass and D. Solitar, Combinatorial Group Theory, Wiley Interscience,New York, 1968.

[14] R. Steinwandt, Loopholes in two public key cryptosystems using the modular groups, preprint Univ. of Karlsruhle, 2000.

[15] J.Tits, Free Subgroups in Linear Groups, J. Algebra 20, 1972, 250-270.

[16] Xiaowei Xu, Cryptographic Systems using Linear Groups– Ph.D. Thesis.

[17] A.Yamamura, Public Key cryptosystems using the modular groups, Lecture Notes in Comput. Sci. 1431, 1998, 203-216.

Appendix B

A Proposed Public Key Cryptosystem Using the Modular Group

1. Introduction

In [BFX] we described a general schema for developing encryption methods using a combination of combinatorial group theory and linear groups. The decryption algorithm was based on the Reidemeister-Schreier rewriting system. Experiments [X] indicated that the system was indeed secure as a classical cryptosystem. In the present paper we propose to make this system public key in the sense that we publish the encryption method so that anyone can send an encrypted message. However only those with knowledge of certain private keys can decrypt in a reasonable amount of time.

The general method outlined in [BFX] went as folows: We start with a given a finitely presented group $G =< X, R >$ and a homomorphism $\rho : G \to \overline{G}$ where $\overline{G}$ is a linear group, together with an algorithm so that elements of $\rho(G)$ can be writ-

ten in terms of a generating system $g_1, ...g_n$ of G. We then take a subgroup K of G and a free subgroup H of K together with Schreier transversals for K in G and for H in K (see [BFX],[B] or [MKS]). Further we have sets of Schreier generators based on these transversals for H which are expressed in terms of the generators of G and Schreier generators for K expressed in terms of the generators of H. The encryption is done in ρ via multiplication of matrices. The decryption is then a combination of the algorithm to rewrite an element of $\rho(H)$ in terms of the generators of G followed by Reidemeister-Schreier rewriting. The details are in [BFX]. There, a suggested platform group for G is the classical modular group $M = PSL_2(\mathbb{Z})$ and it is within this platform that we propose a public key system. A public key system using the extended modular group $SL_2(\mathbb{Z})$ was proposed by Yamamura [Y] but this was subsequently shown to be breakable ([GP],[S]). Yamamura used however only one masked subgroup. We propose to use a collection of subgroups that are protected from the type of attacks that break the Yamamura system.

Recall that a public key system has two components - an encryption system and a public key exchange protocol describing which encryption ssytem to use. In the proposed system in this paper there are three integral keys. Without knowledge of these keys there are only brute force attacks on a given encrypted message, even knowing the method. Therefore decrypting the message is computationally infeasible (see section 4) and hence the security of the system lies in the security of the key exchange protocol. Further, here , as we will show, knowledge of all three keys, is essential.

In the next section we review the basic ideas of a public key system. In section 3 we describe the basic encryption method using the classical modular group as the

platform group. We then propose a method to make this public key. As we will see although decryption is done via Reidemesiter-Schreier rewriting this is not essential if one knows the subgroup in terms of its generators. The Stallings Folding method (see [KM]) can also be utilized to rewrite by an attacker. However, as we will show, this is computationally infeasible. Further, work with Grobner bases shows that in terms of complexity it is more difficult to find a Grobner basis than to solve the explicit membership problem when a given Grobner basis is known. Translating this to group theory indicates that it is harder in terms of complexity to find a Schreier transversal for a subgroup than to rewrite using a known Schreier transversal for that subgroup.

2. Public Key Cryptosystems

Recall that a public key cryptosystem has an open encryption system but a secret decryption system. That is, anyone can send encrypted messages but the decryption algorithm is known only to those who have the key. As is well-known, a secure public key cryptosystem depends on a good one-way function or one-way process. This one-way function is easy to apply but very difficult to invert.

A public key cryptosystem actualy has three parts: an encryption algorithm that is made public, a key exchange protocol that goes with the encryption algorithm, that is kept secret between the communicating parties and finally a decryption technique that depends on the hidden key. The security of such a system depends on the difficulty of decoding a given message knowing the encryption method but not the secret key. Decoding deosn't have to be impossible for an attacker just timewise infeasi-

ble. For example in the standard RSA algorithm the difficulty of decoding depends on the relative difficulty of factoring a large integer that is aproduct of two primes versus the difficulty of determing large primes (see [K]). It is theoretically possible, but computationally infeasible, to break RSA by a brute force attack, that is try all possible factorizations.

As we will show, this is precisely the situation in our proposed cryptosystem. We will make public generating systems for a large collection of subgroups of the Modular Group M. The secret integral keys will describe which subgroup to use and certain other parameters. Only a knowledge of each of these parameters allows for decryption. A brute force attack going through each subgroup can determine the subgroup and attack a given message. This is computationally infeasible, analogous to the brute force factoring approach attack to RSA.

3. The Modular Group Public Key Cryptosystem

We first describe the basic encryption system and then show the public key version. Our platform base group G as described in the general scheme will be the classical modular group $M = PSL_2(\mathbb{Z})$. As we will see it has all the necessary ingredients to fit the general schema:

1. M is finitely presented and there is a faithful representation in $PSL_2(\mathbb{R})$.

2. There is an algorithm to move back and forth between the presentation of M and the representing matrices

3. There is an abundant supply of free subgroups and it is easy to check on free

cancellation

4. The explicit membership problem for subgroups is easily solvable if you know the subgroup and/or know a Schreier transversal

We descibe each of these and then the encryption algorithm. Details on the Modular Group can be found in the book [F]. Recall that the Modular Group $M = PSL_2(\mathbb{Z})$. That is M consists of the 2×2 projective integral matrices:

$$M = \{\pm \begin{pmatrix} a & b \\ c & d \end{pmatrix} : ad - bc = 1, a, b, c, d \in \mathbb{Z}\}.$$

Equivalently M can be considered as the set of integral linear fractional transformations with determinant 1:

$$z' = \frac{az + b}{cz + d}, ad - bc = 1, a, b, c, d \in \mathbb{Z}.$$

M has a group presentation

$$M =< x, y; x^2 = y^3 = 1 >=< a, t; a^2 = (at)^3 = 1 >$$

where

$$x = \pm \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, y = \pm \begin{pmatrix} 0 & 1 \\ -1 & 1 \end{pmatrix}, a = \pm \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, t = \pm \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

Further $t = xy$. Group theoretically M is a **free product** of a cyclic group of order 2 and a cyclic group of order 3, that is $M = \mathbb{Z}_2 \star \mathbb{Z}_3$. In particular this implies that

x and y as defined above are an independent system of generators and hence any element $g \in M$ has an essentially *unique* expression as a word $W = W(x, y)$.

There is an algorithm based on the Euclidean algorithm which given a projective integral matrix T can rewrite it in terms of the standard generators x, y. This can be described in the following manner. Let a and t be defined as above and let

$$T = \pm \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$

be a matrix in M. Then by computation we get that

$$aT = \pm \begin{pmatrix} \gamma & \delta \\ -\alpha & -\beta \end{pmatrix} = \pm \begin{pmatrix} -\gamma & -\delta \\ \alpha & \beta \end{pmatrix}$$

$$t^k T = \pm \begin{pmatrix} \alpha + k\gamma & \beta + k\delta \\ \gamma & \delta \end{pmatrix}.$$

Therefore by multiplying on the left by a and then by an appropriate t^k we can make the lower left entry positive and smaller. Continuing we get eventually

$$t^{k_1} a t^{k_1} a ... t^{k_n} a^{\epsilon} T = \pm \begin{pmatrix} \mu & \nu \\ 0 & \eta \end{pmatrix}$$

where $\epsilon = 1$ or 0. Since $\mu\eta = 1$ this implies that the right hand side

$$\pm \begin{pmatrix} \mu & \nu \\ 0 & \eta \end{pmatrix} = \pm \begin{pmatrix} 1 & \rho \\ 0 & 1 \end{pmatrix} = t^{\rho}.$$

This then expresses the matrix T as word $W(a,t)$. Then using that $t = xy$ we get T expressed as a word $W_1(x,y)$ in the standard generators x, y. Hence the Modular Group M has the necessary faithful representation to be able to serve as our platform group in the general outlined encryption scheme.

It is known,and an easy consequence of the free product structure (see [F]), that any torsion-free subgroup of M is actually a free group. We say that M has the **torsion-free subgroup property**.

Based on these ideas and following the general outline we give a simple encryption-decryption algorithm based on the Modular group. This is close to the Yamamura method which as mentioned has been broken. The public key version enhances and modifies this basic scheme in order to handle the various attacks.

The encryption algorithm is as follows. Start with a free subgroup H of M. Suppose that $h_1, ...h_t$ is a Schreier transversal for H in M and that $W_1, ...W_k$ is a set of generators for H constructed by the Reidemeister-Schreier method from $h_1, ...h_t$. It could be that t and k are infinite. The Schreier transversal will be kept secret. Suppose that $\mathcal{A} = \{\dashv, \lfloor, ...\}$ is our plaintext alphabet. Suppose further than $k > l$ where l is the size of the plaintext alphabet $\mathcal{A}$. We then, as in the free group cryptosystem, use the substitution ciphertext $a \rightarrow W_1, b \rightarrow W_2, ...$ and so on. Given a plaintext message $W(a,b,...)$ compute $W(W_1, W_2...)$ as an element of H and then express it as an integral matrix

$$W(W_1, W_2, ...) = T = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}.$$

The secret message sent out is T. Notice that knowledge of the Shreier transversal is not necessary for encryption.

To decipher the message the decryption algorithm proceeds as in the general schema. Suppose we obtain the coded message T given as an integral matrix. Use the standard algorithm explained earlier to express T as a word in the standard generators x, y of M. We then have

$$T = W(x, y).$$

We now use the Shreier transversal and Reidemeister-Schreier rewriting to reexpress T as a word in W_1, W_2.... Applying the inverse substitution $W_1 \to a, W_2 \to b$... now decodes the message.

As for Yamamura's method [Y] there are various problems with this code (see [GP] and [S]). Since M acts discontinously on the upper half plane, subgroups of finite index behave somewhat regularly relative to this geometric action. An examination of the way the messages act can be used to uncover the subgroup H. Yamamura tried to disguise H by conjugating the message with an arbitrary matrix but this could also be broken. Yamamura used only a single masked subgroup. We now try a slightly different approach.

To make this public key we make a polyalphabetic cipher based on a collection of free subgroups of M. Suppose that we have a indexed collection $H_1, ...H_n$ of free subgroups of M. In a practical implementation we assume that n is large. We make no assumptions on the index of each H_i but assume that each is finitely generated. Let k_i denote the number of generators of H_i. Suppose that $\mathcal{A}$ is the plaintext alphabet we wish to encode. We make the assumption that k_i is much greater that l where l is the size $\mathcal{A}$. This is to ensure that we can start almost anywhere within the system of

generators of H_i for a letter by letter encryption. In practice k_i could be a fixed large integer k so that each H_i is a free subgroup of rank k.

For each H_i we have a Schreier transversal

$$h_{1,i}, ...h_{t(i),i}$$

and a corresponding indexed set of generators

$$W_{1,i}, ...W_{m(i),i}$$

constructed from the Schreier transversal by the Reidemeister-Schreier process. Each of these Schreier generators corresponds uniquely to a 2×2 unimodular projective integral matrix.

What is made public are these systems of generators, given in terms of integral matrices. Each Schreier transversal is kept secret between the communicating parties Bob and Alice. In the case where anyone can send an encoded message only the universal receiver, say Alice, knows the Schreier transversals.

The subgroups in this collection and their corresponding Schreier transversals can be chosen in a variety of ways. For example the commutator subgroup of the Modular group is free of rank 2 and some of the subgroups H_i can be determined from homomorphisms of this subgroup onto a set of finite groups. Practically however it can be shown that if k is a large integer then the probability is 1 that the subgroup generated by k randomly chosen 2×2 integral matrices is free with those as a basis. Hence in practice we can fix k choose n randomly chosen sets of k integral matrices each, and after checking that each is actually free (and discarding those that are not),

use these as the generators for the H_i. After these are chosen we determine determine the Schreier transversals and Schreier generators. We assume that Alice will receive messages from anyone.

Suppose that an arbitary person, Bob, wants to send a message to Alice. Bob first chooses three integers (m, q, t) where

$$m = \text{ choice of the subgroup } H_m$$

$$q = \text{ starting point among the generators of } H_m$$

$$\text{for the substitution of the plaintext alphabet}$$

$$t = \text{ size of the message unit .}$$

We clarify the meanings of the integral parameters q and t. Suppose that Bob has chosen the index of the subgroup m, and the second integral parameter q. He now makes the ciphering substitution

$$a \to W_{m,q}, b \to W_{m,q+1}, ...$$

Again the assumption is that k_i or in practice the fixed k is much larger than l so that starting almost anywhere in the sequence of generators of H_m will allow this substitution. The message unit size t is the number of coded letters that Bob will place into each coded integral matrix. Hence if $t = 3$ he will make the letter by letter substitutions into integral matrices then multiply together groups successive groups of three integral matrices.

Once Bob has made the choices (m, q, t) he takes his plaintext message $W(a, b, ...)$ and groups blocks of t letters. He then makes the given substitution above to form integral matrices

$$T_1, ...T_s.$$

We now introduce a *random noise factor*. After forming $T_1, ...T_s$ Bob then multiplies on the right each T_i by a random determinant one integral matrix R_{T_i} (different for each T_i). The only restriction on this random matrix R_{T_i} is that there is no free product cancellation in forming the product $T_i R_{T_i}$. Within the Modular group this can be easily checked and ensures that the free product expression for $T_i R_{T_i}$ in terms of the free product generators of M is just the concatenation of the expressions for T_i and R_{T_i}. Next he sends Alice the integral key (m, q, t) by some public key method (RSA, Anshel-Goldfeld etc.). He then sends the message as s integral matrices

$$T_1 R_{T_1}, T_2 R_{T_2}, ...T_s R_{T_s}.$$

Hence what is actually being sent out are not elements of the chosen subgroup H_m but rather elements of random right cosets of H_m in M. The purpose of sending coset elements is two-fold. The first is to hinder any geometric attack by masking the subgroup. The second is that it makes the resulting words in the the Modular Group generators longer - effectively hindering a brute force attack.

To decode the message Alice first uses public key decryption to obtain the integral keys (m, q, t). She then knows the subgroup H_m, the ciphertext substitution from the generators of H_m and how many letters t each matrix encodes. She next uses the Modular group algorithm to express each $T_i R_{T_i}$ in terms of the standard generators

x, y of M say $W_{T_i}(x, y)$. She has knowledge of the Schreier transversal, so now uses the Reidemeister-Schreier rewriting process to start expressing $W_{T_i}(x, y)$ in terms of the generators of H_m. The Reidemeister-Schreier rewriting process is done letter by letter from left to right. Hence when she reaches t of the free generators she stops. Notice that the string that she is rewriting is longer than what she needs to rewrite in order to decode, as a result of the random matrix R_{T_i}. This is due to the fact that she is actually rewriting not an element of the subgroup but an element in a right coset. An attacker, not knowing t will continue rewriting even if the attacker determined a Schreier transversal. Further, since these are random right cosets it makes it difficult to pick up statistical patterns in the generators even if more than one message is intercepted. In practice the subgroups should be changed with each message.

The initial key (m, q, t) is changed frequently. Hence as mentioned above this method becomes a type of polyalphabetic cipher. Polyalphabetic ciphers have historically been very difficult to decode (see [7]).

An attacker has access to a string of integral matrices and knows that these are from random right cosets of one of the published subgroups. In the next section we describe how an attacker might proceed and show that it is computationally infeasible. Frist we given a simple example.

EXAMPLE We want to encode the message

STOP THE WAR

We choose the integral keys $(256, 6, 3)$. That is we have chosen subgroup number 256 on our published list and we will start the enciphering at generator number 6, that is

$$a \to W_{256,6}, b \to W_{256,7}...$$

Finally we will group letters in sets of 3.

In reality the full list of subgroups have not yet been constructed so these matrices were chosen as a fixed subgroup of the free subgroup generated by

$$A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix} \text{ and } B = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

.

Assume the subgroup $H_{256} =< B^{-1}AB, B^{-2}AB^2, B^{-3}AB^3, ... >$ we have the encryption scheme

$$a \to B^{-6}AB^6, b \to B^{-7}AB^7, ...$$

Therefore we form by multiplying the encrypting matrices

$$T_1 = STO = \begin{pmatrix} -2995 & -74 \\ 135544 & 3349 \end{pmatrix}, T_2 = PTH = \begin{pmatrix} -4919 & -74 \\ 222618 & 3349 \end{pmatrix},$$

$$T_3 = EWA = \begin{pmatrix} -3883 & -74 \\ 175732 & 3349 \end{pmatrix}, T_4 = R\{\}\{\} = \begin{pmatrix} -6251 & -74 \\ 282900 & 3349 \end{pmatrix}$$

Next we choose four random noise matrices

$$R_1 = ABA = \begin{pmatrix} 5 & 12 \\ 2 & 5 \end{pmatrix}, R_2 = BA = \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix},$$

$$R_3 = A = \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, R_4 = AB = \begin{pmatrix} 5 & 2 \\ 2 & 1 \end{pmatrix}$$

For each $i = 1, 2, 3, 4$ we check that there is no free product cancellation in forming $T_i R_i$. Essentially this means the following. Suppose x, y are the generators of the Modular Group of order 2 and order 3 respectively. If T_i ends in x when written as a word in x, y then R_i cannot begin with x. If T_i ends with y then R_i cannot begin with y^2 and if T_i ends in y^2 then R_i cannot begin with y.

After doing this we form the four matrix products

$$T_1 R_1 = \begin{pmatrix} -15123 & -36310 \\ 684418 & 1643273 \end{pmatrix}, T_2 R_2 = \begin{pmatrix} -5067 & -10208 \\ 229316 & 461981 \end{pmatrix},$$

$$T_3 R_3 = \begin{pmatrix} -3883 & -7840 \\ 175732 & 354813 \end{pmatrix}, T_4 R_4 = \begin{pmatrix} -31403 & -12576 \\ 1421198 & 569149 \end{pmatrix}$$

and transmit these to Alice along with with secret integral key $(256, 6, 3)$.

Alice first expresses each of these four matrices as words in x, y. From the secret key she knows the appropriate subgroup and hence has a Schreier transversal. She starts rewriting $R_1 T_1$. After she obtains three generators from H_{256} from this rewriting she stops although the word for $R_1 T_1$ may be much longer. She then does the same for each of the transmitted matrices. She now has each of the four matrices written as a product of three generators from H_{256}. Using the second key 6 she knows the letter by letter substitution on these generators and can use this to decode.

4. Security and Potential Attacks

Here we examine basic attacks on the proposed cryptosystem and show that the security is entirely in the key exchange. That is there is no feasible attack on the coded message directly.

An attacker has access to the list of encoding subgroups. Hence the attacker has the integral matrices and can easily turn them into words in the standard generators of the modular group. However the attacker must determine all three integral keys to actually decode. Since we are sending coset elements of different cosets there is no geometric pattern in the action of transmitting matrices on the upper half-plane. This handles a protion of the attacks on the Yamamura scheme. An attacker then has no recourse but a brute force attack to determine the correct encoding subgroup. But even here there is a problem.

The Schreier transversal is kept secret by Alice and allows her to easily decode but this presents no problem to the attacker. A method known as Stallings Foldings (the paper [KM] is now the standard reference on this technqiue) allows one, given a known system of generators for a subgroup, to develop a subgroup graph, and then utilize this graph to rewrite in terms of the given generators. There is nothing published however on the relative complexity of rewriting using a Schreier transversal versus rewriting using Stallings Foldings so more work is necessary. However the difficulty of rewriting doesn't play an essential role in the security of the scheme.

However we are sending out not subgroup elements but random right coset element. In principle, the Stallings Folding method will also tell an attacker what coset a given matrix is in relative to a given subgroup. Hwoever a given element of the modular group will lie in a right coset of almost all (perhaps all) of the subgroups on

our list. Hence unless an attacker exactly determines the subgroup this method gives him no information. Further even if they determine the subgroup and the message siye they must do a statistical analysis to obtian the letter by letter substituion. Here finding a single letter suffices. However the messaeg siyes are assumed small. We could even mask the encryption further by instead of using just q and then a sequential substitution used a permutation σ on the generators of the subgroup

There seems to be no trapdoor given the encoded message without finding the subgroup. Hence the security is based on the integral keys (m, q, t). The brute force method an attacker can uitliye is to find a set of Schreier transversals for each of the subgroups on the published list and then rewrite each matrix in terms of each subgroup. Then do a statistical analysis on the letters to find one that makes sense. As pointed out though this is even more difficult since each matrix in a message will be in right cosetss for arbitrarily large sets of subgroups. Hence finding the message size T is necessary.

References

[AN] R. Anderson and R. Needham, Robustness Principle for Public Key Protocols, Crypto 95.

[AAG] I.Anshel,M.Anshel,D.Goldfeld, An Algebraic Method for Public Key Cryptography, Math.Res. Lett 6, 1999, 287-291, Springer Verlag.

[B] G.Baumslag, Topics in Combinatorial Group Theory, Birkhauser, 1993.

[BFX] G. Baumslag, B.Fine, and X.Xu, Free Group and Linear Group Public Key Cryptosystems, in progress.

[F] B.Fine, The Algebraic Theory of the Bianchi Groups, Marcel Dekker, 1990.

[GP] D.Grigoriev and I. Ponomarenko, Homomorphic Public-Key Cryptosystems

Over Groups and Rings, Quaderni di Matematica, 2005 to appear.

[8] C.Hall, I. Goldberg, B. Schneier, Reaction attaacks Against Several Public Key Cryptosystems, Proceedings of Information and Communications Security ICICS 99, Springer-Verlag, 1999, 2-12.

[K] N.Koblitz, Algebraic Methods of Cryptography, Springer, 1998.

[MKS] W. Magnus, A. Karass and D. Solitar, Combinatorial Group Theory, Wiley Interscience, New York, 1968.

[S] R. Steinwandt, Loopholes in two public key cryptosystems using the modular groups. preprint Univ. of Karlsruhle, 2000.

[X] Xiaowei Xu, Cryptographic Systems using Linear Groups— Ph.D. Thesis.

[Y] A.Yamamura, Public Key cryptosystems using the modular groups, Lecture Notes in Comput. Sci. 1431, 1998, 203-216

Bibliography

[1] M. L. Akkar and C. Giraud. An implementation of des and aes secure against some attacks. *Proceedings of CHES 2001*, pages 309–318, 2001.

[2] C. G. Amelino, G. Dumitru, and R. G. Graham. The effect of yukawa couplings on unification predictions and the nonperturbative limit. *Nuclear Physics B, Volume 528*, pages 35–58, 1998.

[3] I. Anshel, M. Anshel, and D. Goldfeld. An algebraic method for public-key cryptography. *MRLETS: Mathematical Research Letters*, pages 1–5, 1999.

[4] T. M. Apostol. Modular Functions and Dirichlet Series in Number Theory. *Springer*, 1990.

[5] D. Augot and M. Finiasz. A public-key encryption scheme based on the polynomial reconstruction problem. *Proceedings of Eurocrypt*, pages 551–565, 2003.

[6] M. Balare and M. Yung. Certifying permutations: non-interactive zero-knowledge based on any trapdoor permutation. *Journal of Cryptology*, pages 149–166, 1996.

[7] B. Barak. How to go beyond the black-box simulation barrier. *IEEE Symposium on Foundations of Computer Science*, pages 106–115, 2001.

[8] B. Barak. Constant-round coin-tossing with a man in the middle or realizing the shared random string model. *Proceeding of the 43rd Symposium of Foundations of Computer Science*, pages 345–355, 2002.

[9] B. Baumslag and B. Chandler. Group Theory. *McGRAW Hill*, 1968.

[10] G. Baumslag. Topics in Combinatorial Group Theory. *Birkhauser Verlag*, 1993.

[11] E. Biham. New types of cryptanalytic attacks using related keys. *Proceedings Eurocrypt 93*, pages 398–409, 1993.

[12] E. Biham and A. Shamir. Differential cryptanalysis of des-like cryptosystems. *Journal of Cryptology*, pages 3–72, 1991.

[13] M. Blum, P. Feldman, and S. Micali. Non-interactive zero-knowledge and its applications. *In 20th ACM Symposium on the Theory of Computing*, pages 103–118, 1988.

[14] M. Blum, A. D. Santis, S. Micali, and G. Persiano. Non-interactive zero knowledge. *SIAM. J. Computing*, pages 1084–1118, 1991.

[15] D. Boneh. Twenty years of attacks on the rsa cryptosystem. *Notices of the American Mathematical Society*, pages 203–213, 1999.

[16] D. Boneh and R. Venkatesan. Breaking rsa may be easier than factoring. *Nyberg*, pages 59–71, 1998.

[17] G. Brassard and C. Crepeau. Zero-knowledge simulation of boolean circuits. *Proceedings on Advances in cryptology*, pages 223–233, 1987.

[18] G. Brassard, C. Crepeau, R. Jozsa, and D. Langlois. A quantum bit commitment scheme provably unbreakable by both parties. *Proceedings of 34th Annual Symposium on Foundations of Computer Science*, pages 362–371, 1993.

[19] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable zero-knowledge. *Electronic Colloquium on Computational Complexity*, 2000.

[20] R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Black-box concurrent zero-knowledge requires (almost) logarithmically many rounds. *SIAM Journal on Computing, Volume 32*, pages 1 – 47, 2003.

[21] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *J. Symbolic Comput*, pages 251–280, 1990.

[22] N. Courtois, A. Klimov, J. Patarin, and A. Shamir. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. *Eurocrypt 2000*, pages 392–407, 2000.

[23] N. T. Courtois and G. Castagnos. What do des s-boxes say to each other? *Cryptology ePrint Archive*, 2003.

[24] J. Daemen and V. Rijmen. Aes proposal: Rijndael. *csrc.nist.gov*, 1999.

[25] I. Damgard. Efficient concurrent zero-knowledge in the auxiliary string model. *Eurocrypt*, pages 418–430, 2000.

[26] S. David. History of Mathematics. *Dover*, 1958.

[27] J. M . de Laurentis. A further weakness in the common modulus protocol for the rsa cryptoalgorithm. *Cryptologia*, pages 253–259, 1984.

[28] P. Dehornoy. Braid-based cryptography. *Contemporary Mathematics*, pages 5–33, 2004.

[29] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, pages 644–654, 1976.

[30] D. Dolev, C. Dwork, and M. Nao. Non-malleable cryptography. *SIAM Review*, pages 727–784, 2003.

[31] M. Dominic. The trouble with quantum bit commitment. *eprint*, 1995.

[32] G. Dumitru and R. G. Graham. Unification and extra space-time dimensions. *Physics Letters B, Volume 442*, pages 165–172, 1998.

[33] C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. *Lecture Notes In Computer Science; Vol. 1462*, pages 442 – 457, 1998.

[34] C. Dwork and A. Sahai. Concurrent zero-knowledge : reducing the need for timing constraints. *Lecture Notes in Computer Science*, pages 442–457, 1998.

[35] U. Feige. Feige's thesis. 1998.

[36] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. *Annual ACM Symposium on Theory of Computing*, pages 416–426, 1998.

[37] E. Filiol. A new statistical testing for symmetric ciphers and hash functions. *Cryptology ePrint Archive*, 2002.

[38] B. Fine. The Algebraic Theory of the Bianchi Groups. *Marcel Dekker*, 1990.

[39] C. Florian. A History of Elementary Mathematics. *Macmillan*, 1930.

[40] Gaithersburg. Data encryption standard. *National Bureau of Standards*, 1999.

[41] P. Gemmell and M. Sudan. Highly resilient correctors for multivariate polynomials. *Information Processing Letters*, pages 169–174, 1992.

[42] O. Goldreich. The Foundations of Cryptography Volume 1. *Cambridge University Press*, 2001.

[43] O. Goldreich. Zero-knowledge twenty years after their invention. 2002.

[44] O. Goldreich. The Foundations of Cryptography Volume 2. *Cambridge University Press*, 2004.

[45] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing, Volume 25*, pages 169–192, 1998.

[46] O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, pages 1–32, 1994.

[47] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Jounral on Computing*, pages 281–308, 1998.

[48] D. Grigoriev. Homomorphic public-key cryptosystems over groups and rings. *Quaderni di Matematica*, pages 305–326, 2005.

[49] M. Hirvensalo. Quantum Computing. *Springer-Verlag*, 2001.

[50] B.Fisherand D. Goldfeld I. Anshel, M. Anshel. New key agreement protocols in braid group cryptography. *Proceedings of the 2001 Conference on Topics in Cryptology: The Cryptographer's Track at RSA*, pages 13–27, 2001.

[51] L.R. Knudsen J. Daemen and V. Rijmen. Linear frameworks for block ciphers. *Design, Codes and Cryptography*, pages 65–87, 2001.

[52] T. Jakobsen and L. R. Knudsen. The interpolation attack on block ciphers. *Fast Software Encryption*, pages 28–40, 1997.

[53] E. Kaltofen. Polynomial factorization: a success story. *International Symposium on Symbolic and Algebraic Computation*, pages 3–4, 2003.

[54] I. Kapovich and A. Myasnikov. Stallings foldings and subgroups of free groups. *Fast Software Encryption*, pages 28–40, 1997.

[55] J. Kilian and E. Petrank. An efficient non-interactive zero-knowledge proof system for np with general sssumptions. *Journal of Cryptology*, pages 1–27, 1998.

[56] J. Kilian and E. Petrank. Concurrent zero-knowledge in poly-logarithmic rounds. *Annual ACM Symposium on Theory of Computing*, pages 560–569, 2001.

[57] J. Kilian, E. Petrank, and R. Richardson. On concurrent and resettable zero-knowledge proofs for np. *eprint*, 2001.

[58] D. E. Knuth. The art of computer programming. *Seminumerical algorithms*, pages 251–280, 1981.

[59] K. H. Ko, S. J. Lee, J. H. Cheon, J. W. Han, and J. Kang. New public-key cryptosystem using braid groups. *CRYPTO*, pages 166–183, 2000.

[60] J. Lehner. Discontinuous groups and automorphic functions. *American Mathematical Society*, pages 230–234, 1964.

[61] M. Luby and C. Racko. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, pages 373–386, 1988.

[62] R. C. Lyndon and P. E. Schupp. Combinatorial Group Theory. *Springer-Verlag*, 1977.

[63] W. Magnus. Rational representations of fuchsian groups and non-parabolic subgroups of the modular group. *Nachrichten der Akad Gottingen*, pages 179–189, 1973.

[64] W. Magnus and A. Karrass. Combinatorial Group Theory : Presentations of Groups in Terms of G *Dover Publications*, 1976.

[65] D. Micciancio and E. Petrank. Efficient and concurrent zero-knowledge from any public coin hvzk protocol. *The Electronic Colloquium on Computational Complexity*, 2002.

[66] C. F. Miller. On Group-theoretic Decision Problems and Their Classification. *Princeton University Press*, 1971.

[67] G. L. Miller. Riemanns hypothesis and test for primality. *Journal of Computer and Systems Science*, pages 300–317, 1976.

[68] M. Naor and O. Reingold. On the construction of pseudo-random permutations: Luby-racko revisited. *Journal of Cryptology*, pages 29–66, 1999.

[69] M. A Nielsen and I. L Chuang. Quantum Computation and Quantum Information. *Cambridge University Press*, 2000.

[70] J. Patarin. Hidden fields equations and isomorphisms of polynomials: Two new families of asymmetric algorithms. *EUROCRYPT*, pages 33–48, 1996.

[71] J. Patarin. Improved security bounds for pseudorandom permutations. *4th ACM Conference on Computer and Communications Security*, pages 142–150, 1997.

[72] M. Prabhakaran and A. Sahai. Concurrent zero knowledge proofs with logarithmic round -complexity. *Proceedings of the 43rd Symposium of Foundations of Computer Science*, pages 366–375, 2002.

[73] L. Reyzin. Min-round resettable zero-knowledge in the public-key model. *Lecture Notes in Computer Science*, pages 373–393, 2001.

[74] A. Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 543–553, 1999.

[75] B. Schneier. Applied Cryptography: Protocols, Algorithms, and Source Code in C, second editio *Wiley*, 1995.

[76] V. Shoup. Ntl: A library for doing number theory. *www.shoup.net*.

[77] V. Shoup. A fast deterministic algorithm for factoring polynomials over finite fields of small characteristic. *International Symposium on Symbolic and Algebraic Computation*, pages 14–21, 1991.

[78] V. Shpilrain and G. Zapata. Combinatorial group theory and public key cryptography. *eprint*, 2004.

[79] J. R. Stallings. Topology of finite graphs. *Inventiones Mathematicae*, pages 551–565, 1983.

[80] J. R. Stallings. Foldings of g-trees. *Arboreal group theory*, pages 355–368, 1988.

[81] J. R. Stallings and A. R. Wolf. The todd-coxeter process using graphs. *Combinatorial group theory and topology*, pages 157–161, 1987.

[82] H. Stamer and F. Otto. On oleshchuks public-key cryptosystem. *Cryptology ePrint Archive*, 2004.

[83] V. Varadharajan and R. Odoni. Extension of rsa cryptosystem to matrix rings. *Cryptologia*, pages 140–153, 1985.

[84] V. Varadharajan, R. Odoni, and P. Sanders. Public key distribution in matrix rings. *IEE Electronic Letters*, pages 386–387, 1984.

[85] P. Witold and R. G. Graham. Flat directions, string compactification and three generation models. *Nuclear Physics B, Volume 551, Issue 3*, pages 515–548, 1998.

[86] A. C. Yao. Theory and application of trapdoor functions. *Proceeding of 23th IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.