Diffie-Hellman Key Exchange Protocol, Its Generalization and Nilpotent Groups.

by
Ayan Mahalanobis

A Dissertation Submitted to the Faculty of
The Charles E. Schmidt College of Science
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Florida Atlantic University
Boca Raton, Florida
August 2005

Acknowledgements

My doctoral studies started in New Zealand under the supervision of Douglas Bridges. From there I had to move to the United States to work under the supervision of Fred Richman in constructive mathematics. Though I changed my research topic from constructive mathematics and Fred is no longer my dissertation director, he stayed the course with me as a guide and a mentor. I take this opportunity to thank him for his kindness and generosity and above all for giving me the opportunity to work with him.

Ronald Mullin has been a very good friend, philosopher and guide for me. He not only initiated me into cryptography but stayed with me as a member of my dissertation committee. It was a real honor to work with him not to mention the Erdös number I got because of him.

Spyros Magliveras was the dissertation director for this dissertation. I enjoyed long stimulating conversations with him. He helped me both emotionally and financially through the process of writing this dissertation. I take this opportunity to thank him.

I thank Tomas Schonbek and Heinrich Niederhausen for their help and support in the process of writing my dissertation. Special thanks to my parents for their support and my friends for keeping me sane.

ABSTRACT

Author:	Ayan Mahalanobis
Title:	Diffie-Hellman Key Exchange Protocol, its Generalization and Nilpotent Groups
Dissertation Advisor:	Dr. Spyros Magliveras
Degree:	Doctor of Philosophy
Year:	2005

This dissertation has two chapters. In the first chapter we talk about the discrete logarithm problem, more specifically we concentrate on the Diffie-Hellman key exchange protocol. We survey the current state of security for the Diffie-Hellman key exchange protocol. We also motivate the reader to think about the Diffie-Hellman key exchange in terms of group automorphisms.

In the second chapter we study two key exchange protocols similar to the Diffie-Hellman key exchange protocol using an abelian subgroup of the automorphism group of a nonabelian group. We also generalize group no. 92 of the Hall-Senior table, for arbitrary prime p and study the automorphism group of these generalized group. We show that for those groups, the group of central automorphisms is an abelian group. We use these central automorphisms for the key exchange we are studying. We also develop a signature scheme.

Table of Contents

Chapter 1

The Discrete Logarithm Problem

1.1 Introduction

It is reasonable to assume that people in any civilization, anywhere in this world tried to conceal information in written form as soon as writing was developed. This is probably the first and primitive form of encryption but is only one half of cryptography, the other half is the ability to recreate the original message from its concealed form. Cryptography is not hiding a message, so that no one can find it, but rather to leave the message in public in such a way that no one except the intended recipient understands the message. The first recorded use of cryptography for correspondence was by the Spartans who (as early as 600 BC) employed a cipher device called "the scytale" to send secret communications between military commanders. The scytale consisted of a wooden baton wrapped with a piece of parchment inscribed with the message. Once unwrapped the parchment shrunk and appeared to contain some incomprehensible marks; however, when wrapped around another baton of identical dimensions the original text appears.

Military uses of cryptography were the main motivations behind the study of cryptography in the old days. It was a secret endeavor, mostly undertaken by big governments, who could hide all the efforts and create smokescreens necessary to hide a lot of people, activities and active researchers.

In those days most of the cryptosystems were *private or symmetric* key cryptosystems. In this two users Alice and Bob select a key in advance, which is their private key, then they use the key in a private key cryptosystem to communicate data over the public channel. Military establishments and diplomatic offices normally have staffs, procedures and protocols in place to handle this key selection by two users and ways to change these keys periodically. Secret or private key cryptography is still the backbone of modern day cryptography, but it falls short of todays needs. We explain this with an example.

Let us assume that BankAtlantic™in Boca Raton, U.S.A wants to transfer a sum of money to a rather obscure bank, State Bank of India™, Chittaranjan, India, online. It is impossible that BankAtlantic™ has already negotiated with all banks on earth private keys for secret communication and has procedures in place to change those keys periodically. So the only other alternative is to send a trusted courier to Chittaranjan, India with the key before these two banks can exchange the money. This is clearly a major problem for online commerce, *How can two entities unknown to each other agree on a key?*.

The answer to the question raised above is the *public key cryptography*. We explore public key cryptography in terms of *the Discrete Logarithm Problem*, or more specifically, in terms of *the Diffie-Hellman Key Exchange Protocol*, which is the most primitive idea behind public key cryptography. In the Diffie-Hellman key exchange protocol, two users unknown to each other can set up a private but random key for their symmetric key cryptosystem. This way there is no need for Alice and Bob to meet in advance, or use a secure courier, or use some other secret means, to select a key.

Using the discrete logarithm problem involves computing g^n from g for some given $n \in \mathbb{N}$ in a group G. It is normally seen as an operation in the group $G = \langle g \rangle$. In this thesis we propose a "change in attitude", instead of computing g^n from g, we will compute the function $g \mapsto g^n$. This might seem silly at first, but our fundamental contribution in this thesis is to show that once we settle with the understanding that the discrete logarithm problem refers to a function in general and to an automorphism in most cases, there is an easy and obvious way to generalize the Diffie-Hellman key exchange protocol to non-abelian nilpotent groups. The reader will notice that from now onwards we will start switching between an operation in a group and function between the same group every now and then. This is to point out that they are the same thing in context of the Diffie-Hellman key exchange protocol and to prepare and motivate the reader for the later chapter, where we exclusively talk about functions(automorphisms).

A *one way function* between sets A and B is a function $f : A \to B$ such that it is easy to compute $f(a)$ for any given $a \in A$, but given $f(a)$ it is computationally infeasible to compute a. The most famous one way function is *the exponentiation*, in which a function $f : G \to G$ is defined as $f(g) = g^a \quad a \in \mathbb{N}$, where $G = \langle g \rangle$ is a cyclic group. Notice that the representation of the group is important, because for a finitely presented cyclic group $G = \langle g \mid g^n = 1 \rangle$, an automorphism $f : G \to G$ is given by $f(g) = g^k$, where $\gcd(n, k) = 1$. If someone makes g^k public then k is clearly visible and hence the automorphism becomes easy to find. On the other hand if one represents the group G, as a group of matrices(say), then g^k is a matrix and k is not visible. The most commonly used groups are the multiplicative groups of a finite field and a cyclic component of the group of an elliptic curve, we discuss these later in this chapter.

A trap door function $f : A \to B$ is a function between a set of plaintexts A and a set of ciphertexts B which have two sets of information, public information and private information. With only the public information the function f behaves like an one way function. With both the public and the private information, it is easy to compute the image and the preimage for the function. The famous trap door function RSA uses factoring integers.
There are four major issues with any public key cryptosystem:

Confidentiality: A message sent from Alice to Bob cannot be read by anyone else.

Authenticity: Bob knows that only Alice could have sent the message he just received.

Integrity: Bob knows that the message from Alice has not been tampered with in transit.

Non-repudiation: It is impossible for Alice to turn around later and say she did not send the message.

To see why these four properties are important, think of Bob as Alice's stock broker. Alice sends Bob an instruction to sell one thousand stocks when the stock hits a certain price. This information should remain confidential, because not only it reveals Alice's personal information but also stock holdings of Alice and the price she chose. Bob has to be sure that the message came from Alice not from an imposter, he might get in trouble later for selling Alice's stock without her permission. Bob should be sure that the message has not been altered, i.e., Alice wants to sell one thousand stocks not one hundred or ten thousand. It should be impossible for Alice to turn around later and say, "I never said sell". In other words we require transactions to take place between *two mutually distrusting parties over a public network*.

In this chapter our main concern is the discrete logarithm problem and the cryptosystems using the discrete logarithm problem (DLP for short), more specifically, the Diffie-Hellman key exchange protocol. The essential idea behind the Diffie-Hellman key exchange is: once two entities unknown to each other can *securely and trustfully* establish a key for a secret key cryptosystem between them (like the *Advanced Encryption Standard*). They can then transfer data using a secret key cryptosystem.

1.2 The Discrete Logarithm Problem

Let G be a cyclic group generated by g. Then it is easy to compute g^n for any positive integer n in $O(\log n)$ steps, using the repeated squaring method[1]. In many instances finding n from g and g^n is an exceptionally hard problem with exponential complexity. The degree of difficulty or the computational complexity of the problem depends mostly on the representation of the group, as finding n is trivial in the cyclic group $(\mathbb{Z}_k, +)$. It is interesting to note that though any two finite cyclic groups of same order are isomorphic, i.e., there is a isomorphism between G and $\mathbb{Z}_k$ for a suitable $k \in \mathbb{N}$, computing the image of an element g^n in G under this isomorphism entails solving the discrete logarithm problem. A large variety of groups are used and studied for use in the discrete logarithm problem:

a. Subgroups of $\mathbb{Z}_p^\star$ for some prime p.

b. Subgroups of $\mathbb{F}_{p^n}^\star$ for prime p, especially when $p = 2$.

c. A cyclic subgroup of the group of an elliptic curve $E_{a,b}(\mathbb{F}_p)$ over the finite field $\mathbb{F}_p$ with equation $y^2 = x^3 + ax + b$, $a, b \in \mathbb{F}_p$, also see [8].

d. The natural generalization of the group of an elliptic curve to the Jacobian of a hyperelliptic curve.

e. Abelian Varieties.

f. Ideal class group of an algebraic number field.

1.2.1 Diffie-Hellman key exchange

In 1976, Whitfield Diffie and Martin Hellman [18] introduced a key exchange protocol using the discrete logarithm problem. In this protocol Alice and Bob will set up a random secret key for their private key system, using a public but authenticated channel. They decide on a cyclic group G of order n and a generator g of the group in public. To set up a key Alice chooses a random integer $a \in [1, n]$ and sends Bob g^a, similarly Bob computes g^b for random $b \in [1, n]$ and sends it to Alice. The secret key is g^{ab}, which Alice computes by computing $\left(g^b\right)^a$ and Bob by computing $(g^a)^b$. Notice that an adversary Oscar notices the set $\{g, g^a, g^b\}$, which is now public information. If Oscar can somehow compute or have some non-negligible information about g^{ab} from the public information then the scheme is broken.

[1] Though it is easy to compute g^n, it is often not easy enough for practical purposes. Then special purpose bases in $\mathbb{F}_{2^n}$ for a positive integer n called the optimal normal bases and their alternates are used [51, 52].

This author noticed a misconception in literature; it comes from the implication that the security of the discrete logarithm problem and hence the Diffie-Hellman key exchange protocol is tied somehow to the fact that the discrete logarithm problem is an one way function. This is not true. Though exponentiation is an one-way function, and one way functions have many important uses,[2] it is not the property of being one-way that provides the security to the discrete logarithm problem. The claim of security in one-way comes from given f and $f(a)$ find a, but in the discrete logarithm problem, we already know g and g^n. The challenge in the discrete logarithm problem is finding n, i.e., to find f from a and $f(a)$. There is another serious challenge to the Diffie-Hellman key exchange protocol, given g, g^a and g^b find g^{ab}, known as *the Diffie-Hellman Problem*, DHP for short. We write this in terms of functions. Let A be a non empty set and $\mathcal{H}$ is a set of functions $f : A \to A$ such that $f \circ g = g \circ f$ (we will henceforth denote $f \circ g$ by fg) for all $f, g \in \mathcal{H}$. Then the Diffie-Hellman key exchange protocol can be expressed as follows: choose $a \in A$, and make it public. If Alice and Bob want to decide on a key using A, a and $\mathcal{H}$ then Alice chooses a random $f \in \mathcal{H}$ and sends Bob $f(a)$, similarly Bob chooses a random $g \in \mathcal{H}$ and sends $g(a)$ to Alice. They both compute $f(g(a)) = g(f(a))$ which is their private key or shared secret. In this case an adversary sees a, $f(a)$ and $g(a)$. Let $\mathcal{X} = \{b : \ f(a) = b \text{ for some } f \in \mathcal{H}\}$. Then the challenge is to compute $g(y)$ for all $y \in \mathcal{X}$ from the action of g on a, i.e., $g(a)$. If we think of A as a group and $\mathcal{H}$ a subgroup of its automorphism group, then $\mathcal{X}$ is the orbit of a under the action of $\mathcal{H}$ on A. In case of a A being a group of prime order and a a generator of A. The orbit of a under $\mathcal{H}$, the group of automorphisms of A, is A. This fact provides some evidence in the direction that for cyclic groups DHP is equivalent to DLP. There is another major issue with the Diffie-Hellman key exchange protocol, known as *bit security*. Suppose a 128 bit AES key is to be established, then how does one ensure that no information about these bits are computable from the public information g, g^a and g^b? The importance of these thoughts is easy to see. If an adversary can compute half of the bits then there is no point in establishing a key. We shall go in detail on this issue in the next section.

It is probably prudent to point out here that, Bob and Alice need to be sure about the identity of the other person, i.e., they must use an authenticated channel. It is possible that Oscar pretending to be Alice might start the protocol and get a secret key established with Bob, who gives the secret to Oscar, an imposter, instead of Alice.

Security of the Diffie-Hellman key exchange.

The three important security concepts on which the security of the Diffie-Hellman key exchange protocol depends. They are written below with decreasing computational strength. In this section, let G be a cyclic group of order n generated by g.

Discrete Logarithm Problem: If from g and g^a Oscar an adversary can compute a, then he can compute g^{ab} and the scheme is broken.

Diffie-Hellman Problem: Suppose from the information g, g^a and g^b with or without solving the discrete logarithm problem, Oscar can compute g^{ab} then the protocol is broken. It is still an open problem if DHP is equivalent to DLP.

Decision Diffie-Hellman Problem: Suppose we are given g, g^a, g^b and g^c, DDH is to answer the question, deterministically or probabilistically, Is $ab = c \mod n$?

[2]Consider a multi-user computer system where each user needs to authenticate himself with a password, to log in. If the passwords are stored in the computer then the file containing those passwords needs to be heavily protected. On the other hand if we take a one way function f and store $f(a)$ for a password a, and then each time a user types in his password b we compute $f(b)$ and match it with the password file. Then the password file is not that important, because all one sees in the password file is $f(a)$ and it is hard to find a from that.

Clearly any solution to the discrete logarithm problem implies a solution to the Diffie-Hellman problem and any solution to the Diffie-Hellman problem implies a solution to the decision Diffie-Hellman problem. So, the decision Diffie-Hellman problem is the weakest in terms of computational complexity and is currently the most researched attack.

1.3 Attacks on the Discrete Logarithm Problem.

Let G be a cyclic group of order n, generated by g. Let $g^x = a$ for $x \in [1, n]$. We are given g and a, the DLP (the discrete logarithm problem) is to find $x = \log_g a$. Here g is called the *base* for the discrete logarithm problem. It is often customary to define the DLP in a cyclic group, but it can be defined in any group G. Fix a base $g \in G$ and work in the cyclic group $\langle g \rangle$. The easiest of the attacks is to produce an ordered list $\{g^k : \ k = 1, 2, \cdots, n\}$ and compare a with elements in the list to find x. This attack takes $O(n)$ space and at least n operations to compute the list, i.e., time complexity is $O(n)$. Any attack has to beat this space-time complexity. There are two kinds of attack to solve the discrete logarithm problem:

1. Generic Attacks: These attacks work for any cyclic group, i.e., these attacks treat the group as a finitely presented group [9, 36, 44, 63]. There is another way to look at generic attacks. We think of the group G as an oracle, which can compute the product gh of two elements g and h in G, it can compute the inverse g^{-1} for any element $g \in G$ and it can test equality for any two elements in the group. So in this context the algorithm for a generic attack makes oracle calls to perform group operations. The following are the known generic attacks[3]:

    a. Shanks baby-step giant-step.

    b. Silver-Pohlig-Hellman.

    c. Pollard's ρ method.

    d. λ-method.

2. Special Attacks: These attacks are not generic because they need more information than is provided by the oracle, i.e., these attacks depend on the particular group or a family of groups in which exponentiation is working for that particular cryptosystem. A good example is index calculus attacks on the discrete logarithm problem on $\mathbb{Z}_p^*$ or $\mathbb{F}_q^*$, where p is a prime and $q = p^n$. In this case the attack uses the representation of the group.

1.3.1 Generic Attacks.

The most common generic attack to the discrete logarithm problem is based on the *index search algorithm*, popular in computer science. Suppose there is an ordered list of $n + 1$ elements $\{g_i\}_{i=0}^n$. Let us try to find the index of a in this list. We further assume that $\sigma : g_i \mapsto g_{i+1} \mod n$ is efficiently computable. Then choose a positive integer M and compute and store the table

$$\left\{a, \sigma(a), \sigma^2(a), \cdots, \sigma^{M-1}(a)\right\}.$$

[3]The work with generic attacks is almost complete with Victor Shoup's paper [60], who found a tight lower bound for the complexity of a generic attack in Silver-Pohlig-Hellman attack. However there is still some interest with parallelization of these attacks as in [66].

Then $g_0, g_M, g_{2M} \cdots$ is computed one after the other and compared with the table above, if there is a collision of g_{iM} with $\sigma^j(a)$ then the index of a is $iM - j$.

In practice the integer M is chosen close to $\sqrt{n}$ and then the space complexity is $O(\sqrt{n})$ and the time complexity is $O(\sqrt{n}\log n)$. Shanks baby-step giant-step algorithm for solving the discrete logarithm problem is a particular case of the index search algorithm. In this case take g as the generator of the group and the index is the exponent of g, σ is defined as multiplication by g. So index search applies to solve the discrete logarithm problem, but is exponential in both space and time complexity.

One can reduce the space complexity to nothing by going probabilistic using Pollard's ρ method, which is far more practical. Still the heuristic time estimate is the same as for the baby-step giant-step algorithm [9, 44]. It is straightforward to see that Shank's algorithm can be used to calculate a multiple of the order of g in time $O(\sqrt{n}\log n)$ by solving $\log_g 1$ where 1 is the identity of the group. Now suppose that

$$n = \prod_{i=1}^{k} p_i^{\alpha_i} \quad \text{where} \quad p_1 < p_2 < \cdots < p_k \tag{1.1}$$

are the prime factors of n, and α_i is the largest power of p_i dividing n. The x in the discrete logarithm problem is computed modulo n, hence if one can compute x modulo $p_i^{\alpha_i}$ for each i, then using the Chinese remainder theorem one can compute x modulo n. Let $x' = x \mod p^\alpha$ and we compute x'. Let

$$x' = x_0 + x_1 p + x_2 p^2 + \cdots + x_{\alpha-1} p^{\alpha-1} \mod p^\alpha \tag{1.2}$$

be the p radix expansion of x' for some $p \in \{p_1, p_2, \cdots, p_k\}$ and α the corresponding α_i. Clearly $0 \le x_i < p$ for $i = 0, 1, \cdots, \alpha - 1$. Since $x = x' + p^\alpha t$ for some integer t. Notice now that

$$a^{\frac{n}{p}} = \left(g^{x' + p^\alpha t}\right)^{\frac{n}{p}} = g^{\frac{nx'}{p} + np^{\alpha-1}t}. \tag{1.3}$$

Since

$$\frac{nx'}{p} = n\left(\frac{x_0}{p} + x_1 + x_2 p + \cdots + x_{\alpha-1}p^{\alpha-2} + p^{\alpha-1}t\right), \tag{1.4}$$

we have

$$g^{\frac{nx}{p} + np^{\alpha-1}t} = g^{\frac{nx_0}{p}} \quad \text{since the order of } g \text{ is } n. \tag{1.5}$$

Hence one computes $a^{\frac{n}{p}}$ and $\zeta = g^{\frac{n}{p}}$ and then use Shanks baby-step giant-step to find x_0. Clearly, the order of ζ is p, and hence the complexity of the discrete logarithm is that of a group of order p. Once x_0 is computed one computes

$$ag^{-x_0} = g^{x_1 p + \cdots + p^{\alpha t}},$$

hence $(ag^{-x_0})^{\frac{n}{p^2}} = g^{\frac{nx_1}{p}}$. Then finding x_1 is the same as computing the discrete logarithm in ζ as n and p^2 is known. Similarly, one can proceed to find x_i for all i by solving the discrete logarithms in[4] ζ. Further details of this process (the Silver-Pohlig-Hellman algorithm) can be found in [36, Chapter IV] or [9, 40, 44, 54, 63].

The Silver-Pohlig-Hellman algorithm shows us that groups in which all the prime factors are "small" should be avoided for use in the discrete logarithm problem. In other words the groups acceptable for use in any cryptosystem, assuming the discrete logarithm problem to be a hard problem, should have at least one large prime factor in their order. The largest prime factor of the order of the

[4] Is there an efficient parallel implementation of Silver-Pohlig-Hellman algorithm? This question is interesting because to compute x_i one needs to compute x_{i-1}, making the algorithm recursive by nature.

group is going to provide the security as is clear from the Silver-Pohlig-Hellman algorithm. This is one argument in favor of using only subgroups of prime order, for a large enough prime. This algorithm is the reason that once $F^*_{2^n}$, where $2^n - 1$ is prime, was thought ideal for cryptosystems using the discrete logarithm problem. Victor Shoup in [60] shows that Silver-Pohlig-Hellman is the best generic algorithm one can expect with respect to running time.

It is clear that the above algorithm only works if the order of the group is known. We now show that any algorithm to compute the discrete logarithm can be used to compute the order of the base element g. Let us assume that we can compute the discrete logarithm to the base $g \in G$ and we prove that there is a non-deterministic algorithm to compute the order of g.

Choose an integer m; it helps if one can make a guess and choose m to be bigger than the order of g. Then pick a random $y_0 \in \{m, m+1, m+2, \cdots, 2m\}$ and compute g^{y_0} in G and then compute $x_0 = \log_g g^{y_0}$. If $n_0 = x_0 - y_0 = 0$ then the choice of m was too small, make $m := 2m$ and start all over again. If $n_0 \neq 0$, then choose another y_0 at random and find n_1. Then the order of g is a factor of the $\gcd(n_0, n_1)$. After several computations of n_i's their GCD will yield the order[5] of g.

We just proved that computing the discrete logarithm is as hard as computing the order of an element. Hence if we can find a group G such that computing order of an element g is a hard problem and can build a cryptosystem whose security depends on computing that order, then that cryptosystem is at least as secure as computing the discrete logarithm in the cyclic group generated by g.

This idea can serve as a motivation for the work of Wei, Trung, Magliveras and Hoffman in [69], though they didn't mention this as their motivation. The idea behind their cryptosystem is not new. It is very much similar to the idea of Kevin McCurley in [43]. The central idea is to work in the $\mathbb{Z}_n^*$, where $n = pq$, p and q are primes and $\mathbb{Z}_n^*$ is the group of units in the ring $\mathbb{Z}_n$. Then they find an element $\alpha \in \mathbb{Z}_n^*$, such that to find the order of α one needs to factor n. The claim of Wei *et. al.* is that the cryptosystem is as secure as RSA and the discrete logarithm in a prime field, taken together. We will see later RSA is "a little less" secure than the discrete logarithm problem in a prime field with the same modulus that of RSA. Hence the primes to be chosen have to be at least 1024 bits, making n large. On the other hand, since the security depends on two theoretical problems with the same complexity, this cryptosystem is like using RSA twice or DLP in a prime field twice.

1.3.2 Special Attacks: Index Calculus Methods.

The special attack we have in mind is known as the *Index Calculus Method*, it works for the multiplicative subgroups of the finite fields and the class groups of imaginary quadratic number fields. We describe here the attack for $\mathbb{Z}_p^*$ and $\mathbb{F}_{2^k}^*$ where k is a positive integer. This method is normally attributed to Kraitchik, who wrote about it in the 1920's [44, 58], but the modern version was rediscovered by Adleman in [1]. This method is probabilistic rather than deterministic.

Let G be a group generated by g of order n. Let $\mathbb{S} := \{g_1, g_2, \cdots, g_m\}$ be a set of elements of G. Then the index calculus methods involves two precomputations. First we compute equations of the form

$$\prod_{j=1}^{m} g_j^{a_{ij}} = g^{b_i} \tag{1.6}$$

[5] It is worth pointing out here the relevance of non-deterministic algorithms in computational mathematics, a deterministic algorithm of the above would have almost certainly included in it an algorithm for the well ordering principle for naturals, no one believes that such an algorithm exists.

or equivalently

$$\sum_{j=1}^{m} a_{ij} \log_g g_j = b_i \mod n \tag{1.7}$$

where a_{ij} and b_i are positive integers. The computation in this step is the same as finding b_i, such that g^{b_i} factors in the set $\mathbb{S}$. Clearly g^{b_i} is a random element of the group, for a randomly chosen b_i. So, in other words we are choosing random elements from the group and trying to factor them into $\mathbb{S}$. In the second stage we solve the set of linear equations for $\log_g g_j$, i.e., find the discrete logarithm for each g_i. This completes the precomputations. Now suppose we want to find $\log_g a$ then we construct relations of the form

$$\prod_{j=1}^{m} g_j^{e_j} = ag^e \tag{1.8}$$

where e_j and e are positive integers and e is chosen randomly. This is equivalent to saying that we find an integer e such that ag^e factors in the set $\mathbb{S}$. Equation 1.8 implies that $\sum_{j=1}^{m} e_j \log_g g_j = e + \log_g a$, further implying

$$\log_g a = \sum_{j=1}^{m} e_j \log_g g_j - e \tag{1.9}$$

Two questions arise automatically,

1. How to find $g_1, g_2, \cdots, g_m$, such that equations of the form of Equation 1.6 can be formed effectively and efficiently? This step will be mentioned henceforth as the *database*, because we are creating a database of linear equations in $g_1, g_2, \cdots, g_m$.

2. How to solve the set of linear equations, i.e., the database created above with equations like Equation 1.7, for $\log_g g_i$?

Forming the database and solving the set of linear equations is known as the precomputations, in this step the discrete logarithms of the elements $g_1, g_2, \cdots, g_m$ are found, which are needed only once for each new logarithm found later and hence can be stored in a slow device.

The first of these questions limits the index calculus method mostly to the multiplicative group of finite fields, where we know how to effectively generate these $g_1, g_2, \cdots, g_m$. The second question is more intriguing. Parallel to the index calculus method in finite fields there is a factoring algorithm for integers, but we will not explore factoring algorithms in this thesis. The precomputation has to be done only once for a particular group, so complexity of the precomputation is one of the security conditions in the discrete logarithm problem. Once the precomputation stage is computed the computation of the final stage i.e., Equations 1.8 and 1.9 is not that tedious or time consuming.

Clearly the larger the number m, the greater the probability that Equations of the form (1.6) and (1.8) can be formed. But if m is too large then, since there have to be significantly more than m equations of the form (1.6), solving that many linear equations adds to the complexity of the index calculus algorithm. There are two bottlenecks in this algorithm:

a. Forming enough equations of the form of Equation 1.6.

b. Solving the above mentioned system of equations.

Index calculus in $\mathbb{Z}_p^*$

Let us assume that $\mathbb{Z}_p^*$ is generated by g. To form equations of the form 1.6, take a set of m primes $\{p_1, p_2, \cdots, p_m\}$. The usual practice is to take the first m primes. Take an arbitrary random integer $r \in [1, p-1]$ and compute the least integer $z = g^r \mod p$. If $z = \prod_{i=1}^{m} p_i^{e_i}$, where $e_i \in \{0, 1, 2, \cdots\}$ then we have an equation as in Equation 1.6. If z doesn't factor into the set then we throw away that z and pick another random r and proceed as before.

It is clear at this stage that the bigger the m, the probability of finding more equations of the form of Equation 1.6, increases. On the other hand while solving the system of linear equations as in Equation 1.7, the bigger the m, the higher the cost of computation. The number of equations, like Equation 1.6, should be greater than m. There are many strategies available to solve the set of linear equations, we refer the reader to [44, Section 5.1] or probably the best survey written on the discrete logarithm problem [53] or the paper by Kevin S. McCurley in [55]. We mention a few facts about the complexity of the discrete logarithm problem in $\mathbb{Z}_p^*$. This might not be the best complexity analysis of the facts, to date, but historically whenever there is a new method for factoring integers, the same method can be adapted into an index calculus method for prime fields. It has been the case that the discrete logarithm for prime field is always "a little more" secure than RSA, where the prime in the prime field is of the same size as the modulus of RSA.

Let us define

$$L(p) = \exp(\sqrt{\log p \log \log p}). \tag{1.10}$$

Then McCurley proves in [44, Page 62] that if trial division is used in the index calculus method and $2m$ equations are generated then the time complexity is

$$L(p)^{2c+1/(2c)+o(1)}. \tag{1.11}$$

Carl Pomerance in his paper titled Factoring in [55] uses the quadratic sieve factoring method and states heuristically that the running time to factor an integer n such as $n = pq$, where p and q are of the same size is

$$\exp(1 + o(1)(\sqrt{\log n \log \log n})) \tag{1.12}$$

After establishing a relationship between the complexity of RSA and that of the discrete logarithm in prime fields, it is easy to find suitable primes for a secure Diffie-Hellman key exchange in prime fields, once one accepts that RSA with modulus of same size is secure. These days, the industry standard for a modulus for RSA is 1024 bits. So using a prime of 1024 bits should provide an adequate security to any cryptosystem using the discrete logarithm problem, for example, the Diffie-Hellman key exchange. However in [54], Odlyzko claims that for long lasting security one needs to amend the size of the modulus of RSA to 2048 bits.

Index calculus in $\mathbb{F}_{2^k}^*$ Cryptologic protocols are most interesting over an extension of the binary field. The reasons are easy hardware as well as software implementations of the protocol. There is a lot of interest these days on implementations of finite fields of characteristic 2 (see for example [45, 51, 52]), as a hardware model. In this section we will not talk about representations of finite fields; rather, we will talk about the index calculus method in $\mathbb{F}_{2^k}^*$. There is the usual index calculus whose running time can be analyzed completely, as in [53]. Then there are improvements made by Blake, Fuji-Hara, Mullin and Vanstone in [6, 7] and by Coppersmith in [15]. To emphasize the improvements we quote a paragraph from [15].

> Throughout this paper we will use for our example the field GF(2^{127}). The primitive polynomial involved is $P(x) = x^{127}+x+1$. The Diffie-Hellman key exchange algorithm,

as described above, has been implemented in this field. To build the database necessary to take logarithms in this field, Adleman's algorithm seems to take two weeks; a modification due to Blake, Fuji-Hara, Mullin and Vanstone takes about nine hours, and the present scheme takes eleven minutes.

Blake *et. al.* [6] and Coppersmith [15] attacks the first bottleneck of Adleman's algorithm [1], i.e., trying to find conditions where one finds polynomials which are easier to factor completely into irreducible polynomials of "low" degree. The clever point made by Blake *et. al.* is: the probability that a polynomial of small degree will factor completely into irreducible polynomials of low degree is much higher than that for a polynomial of much higher degree. Hence if one could find a way to choose randomly polynomials of lower degree, then the complexity of forming relations for precomputations goes down quite a bit.

In [6] the authors use the extended Euclidean algorithm to find for any polynomial $g(x)$ of degree at most $(n-1)$, two polynomials $t(x)$ and $r(x)$, such that degrees of $t(x)$ and $r(x)$ are less than or equal to $\frac{n-1}{2}$ and $g(x) = \frac{t(x)}{r(x)}$. They then prove heuristically that the probability that $t(x)$ and $r(x)$ will factor into irreducible polynomials of smaller degree is much higher than that of factoring $g(x)$ into irreducible polynomials of low degree.

In other words, if they choose a database D of all irreducible polynomials of degree less than or equal to b, then using the above method it is much faster to compute the linear equations as in Equations 1.6 and 1.7. This is a contributions to the practical side of the index calculus algorithm by Blake *et. al.* . There is a serious theoretical contribution made in [6] which motivated Coppersmith in [15], known as *systematic equations*. It follows from the following theorem:

Theorem 1.3.1. *Let $f(x)$ be an irreducible polynomial of degree n, over $F = \mathrm{GF(q)}$ and let $g(x)$ be an arbitrary polynomial over $\mathrm{GF(q)}$. If $m(x)$ is any divisor of $f(g(x))$ then the degree of $m(x)$ is a multiple of n.*

Proof. See [6] page 283. •

Let $P(x)$ be the defining polynomial for the field $F = \mathrm{GF}(2^{\mathrm{n}})$. Further assume that $P(x) = x^n + Q(x)$ where $Q(x)$ is a polynomial of low degree. Then one can write

$$x^k = x^{k-n}Q(x) \mod P(x) \tag{1.13}$$

where k is the smallest integer greater than n of the form 2^l, $l \in \mathbb{N}$. Then for any polynomial g in the field $g\left(x^k\right) = g(x)^k$, since F is of characteristic 2. Thus

$$g\left(x^{k-n}Q(x)\right) = g(x)^k \tag{1.14}$$

Now take an irreducible polynomial $g(x)$ of small degree in F, then there is a high probability that the polynomial $g\left(x^{k-n}Q(x)\right)$ factors in the database and from the above equation there are many linear equations of the form of Equation 1.7. Blake *et. al.* work to create a database for $\mathrm{GF}(2^{127})$. In that case there were not enough systematic equations found by the above rule. They also failed to give any systematic approach to create these systematic equations. There are some better ways known today than to go for trial division of $g(x)$, for example see [53]. Coppersmith found a way to to create systematic equations of degree less than equal to $n^{\frac{2}{3}}$, see [15, Section IV].

1.4 The Diffie-Hellman Problem

It is clear that if one can solve the discrete logarithm problem then one can solve the Diffie-Hellman problem. Is the other direction true? This has been one of the fundamental questions concerning the security of the Diffie-Hellman key exchange protocol. In this section we talk about some of the ideas described in [12, 39]. We won't go to the explicit details in describing the "Black Box Fields", neither do we feel that that description is important. It is a more graphic description of a field, other than that it serves no purpose. The question we begin with is, is the Diffie-Hellman problem as secure as the discrete logarithm problem? This is the same as asking does solution of the Diffie-Hellman problem yields to a solution of the discrete logarithm problem? We feel that the problem is not well posed as what does solution of the discrete logarithm problem or solution of the Diffie-Hellman problem means. After all they all have solutions, it is finding the solution that matters. That is where computational complexity of the algorithm that finds the solution comes into play. We qualify these algorithms in terms of $\log n$ where n is the order of the group $G = \langle g \rangle$. We follow Boneh and Lipton in [12] to show that if the discrete logarithm problem is exponential in $\log n$ then the Diffie-Hellman problem is also exponential in $\log n$. This is very encouraging news for the elliptic curve cryptography, in which no subexponential algorithm is known for the discrete logarithm problem, hence in current understanding of the problem, the Diffie-Hellman problem is also exponential. In other words if one believes in the security of the discrete logarithm problem, then one has every reason to believe in the security of Diffie-Hellman problem in the group of elliptic curves.

Let G be a cyclic group generated by g which is of order p, where p is a prime. Then every element of G can be expressed as g^n, $n \in [1, p]$. Addition and multiplication in G are defined respectively as follows:

$$g^n + g^m = g^{n+m} \tag{1.15}$$

$$g^n \cdot g^m = g^{nm} \tag{1.16}$$

Clearly $(G, +, \cdot)$ is a field. Notice that computing the sum is the same as the operation in the cyclic group G. Hence assuming that one can compute the Diffie-Hellman problem, or there is a Diffie-Hellman oracle which when given g^a, g^b computes g^{ab} with out any computational cost, we can define a field on G and compute sum and product for any two elements in G. We will denote $(G, +, \cdot)$ by "the field G". Corresponding to the field G there is a map $\tau : G \to \mathbb{F}_p$ defined as $g^x \mapsto x$. Hence τ computes the discrete logarithm of g^x. On the other hand it is easy to compute g^x corresponding to any $x \in \mathbb{F}_p$.

With the use of the Diffie-Hellman oracle, the field G for all computational purposes behaves like a field of order p. Hence one can define an elliptic curve $y^2 = x^3 + ax + b$ over the field G, we denote the elliptic curve over the field G by $E_{a,b}$. It is clear that all algorithms for an elliptic curve that use the operations of sum, product and testing of equality of the field can be used in this scenario. In particular Schoof's algorithm to compute $|E_{a,b}|$ can be used.

We select a and b such that a curve of smooth order is found, i.e., the largest prime divisor of $|E_{a,b}|$ is less than or equal to $\exp \sqrt{\log p \log \log p}$. Boneh and Lipton in [12] prove that it can be expected after $\exp\left(\left(\frac{1}{2} + o(1)\right) \sqrt{\log p \log \log p}\right)$ tries. Then they use the fact that the abelian group corresponding to the elliptic curve has at most two minimal set of generators. They further prove that the probability that two arbitrarily chosen points generate the whole abelian group is at least $\Omega(1/\log^2 p)$.

Then they provide an algorithm [12, Theorem 3.1] to compute the discrete logarithm, i.e., τ. The algorithm is subexponential in $\log p$. Now assume there is an oracle that solves the Diffie-Hellman problem in subexponential time in $\log p$ since the algorithm for computing τ is subexponential hence one can solve the discrete logarithm problem in subexponential time. We know that is not the case in many groups, for example, the groups of a elliptic curve. Then there is every reason to believe that there is no subexponential algorithm for the Diffie-Hellman problem, i.e., the Diffie-Hellman oracle with subexponential time can't be built. Of course, this whole analysis fails as soon as one finds a subexponential algorithm for the discrete logarithm problem in elliptic curves, but till then the Diffie-Hellman problem is as secure as the discrete logarithm problem for groups like the group of an elliptic curve where there is no subexponential algorithm for the discrete logarithm problem known. Boneh and Lipton in [12, Theorem 4.4] states these facts more formally in the language of computational complexity, we refer an interested reader to that.

1.5 The El-Gamal Signature Scheme

We start this section quoting the abstract from *FIPS PUB 186* - Digital Signature Standard (DSS) available online at *http://security.isu.edu/pdf/fips186.pdf*.

> This standard specifies a Digital Signature Algorithm (DSA) which can be used to generate a digital signature. Digital signatures are used to detect unauthorized modification of data and to authenticate the identity of the signatory. In addition, the recipient of signed data can use a digital signature in proving to a third party that the signature was in fact generated by the signatory. This is known as non-repudiation since the signatory cannot, at a later time, repudiate the signature.

This chapter began with the requirements for any public key cryptosystem. We see that Diffie-Hellman key exchange protocol with a secure signature scheme satisfies this requirement. In a signature scheme there is a public list, like a phone book, available. Each user chooses a public and a private key. The public key is made available along with the name or some other authentication of a person or an organization, in the public list. Now if the user Alice wants to sign a message m, then she first uses a hash function to reduce the message space to a fixed and small message size. Then she signs the message using her private key. She then sends m and the hashed signed message to Bob. Bob on receiving the message verifies it with the public key of Alice. If the message is authentic then the verification algorithm is true, otherwise it is false. From the verification algorithm it should be clear if the message is authentic and signed by Alice. We present now the El-Gamal signature scheme on which DSS is developed. The security of the scheme is based on the discrete logarithm problem.

1.5.1 The El-Gamal Signature Scheme

Let p be a prime and α be a primitive element of $\mathbb{Z}_p^*$. Then the public information of Alice is α and β where $\alpha^a = \beta$. The secret key is a. To sign a message m, Alice selects a random integer $k \in \mathbb{Z}_{p-1}^*$, where $\mathbb{Z}_n^*$ is the ring of units in $\mathbb{Z}_n$. To compute the signature for m, Alice does the following:

$$\begin{aligned} \gamma &= \alpha^k \mod p. \\ \delta &= (m - a\gamma)k^{-1} \mod (p-1). \end{aligned}$$

Then the signature for the message m is the pair (γ, δ). As described earlier the usual practice is to hash m with a public hash algorithm rather than using m. It makes computing signature faster but the hash function might add vulnerabilities to the signature scheme. The integer k should never be made public, if it becomes public then it is easy to compute

$$a = (m - k\delta)\gamma^{-1} \mod (p-1).$$

Once a is known the system is completely broken.

To verify the signature Bob gets (p, α, β). Then he does the following:

Check to see if	$0 < \gamma < p$	otherwise reject.
Compute	$v_1 = \beta^\gamma \gamma^\delta$.	
Compute	$v_2 = \alpha^m \mod p$.	
If	$v_1 = v_2$	accept, otherwise reject.

To see why the signature scheme works, notice that

$$m = k\delta + a\gamma \mod (p-1)$$

Hence,

$$\begin{aligned} \alpha^m &= \alpha^{k\delta + a\gamma} \\ &= \alpha^{k\delta}\alpha^{a\gamma} \\ &= \gamma^\delta \beta^\gamma. \end{aligned}$$

Because, $\beta = \alpha^a \mod p$ and $\gamma = \alpha^k \mod p$.

Once Alice signs a message and it is verified by her public key, she can't refuse that she signed the message. The public key in this case is very similar to "her own signature". On the other hand verification of the signature involves computing α^m. Hence if the message m is different from the one she signed then the verification algorithm will reject the message. So if the verification algorithm accepts the message then two facts are established; the fact that Alice sent the message and that the message is authentic.

1.6 Conclusion

We began this chapter with the need for a public key cryptosystem and the necessary conditions a public key cryptosystem must satisfy like, confidentiality, authenticity, integrity and non-repudiation.

We have shown that the Diffie-Hellman key exchange protocol along with the El-Gamal signature scheme satisfies all these needs. This makes a complete cryptosystem.

There is one major development in this chapter. We noted that, "the operation of exponentiation in a cyclic group of prime order is an automorphism". Since the DLP involves exponentiation, hence we can see the whole concept of the DLP in a group of prime order as computing the image of an automorphism of the group. In the next chapter we show that this idea leads us to an easy generalization of the Diffie-Hellman key exchange protocol to nilpotent groups.

There are couple of interesting concepts about the security of the DLP which we didn't had the opportunity to explore in this dissertation. They are the decision Diffie-Hellman problem and the differential power analysis.

Chapter 2

Diffie-Hellman key exchange protocol and non-abelian nilpotent groups.

2.1 Introduction

In this chapter we generalize the Diffie-Hellman key exchange protocol from a cyclic group to a finitely presented non-abelian nilpotent group of class 2. Similar efforts were made in [3, 4, 34] to use braid groups, a family of finitely presented non-commutative groups [5, 17], in key exchange. Our efforts are not solely directed to construct an efficient and fast key exchange protocol. We also try to understand the conjecture, "the discrete logarithm problem in a cyclic group is equivalent to the Diffie-Hellman problem in a cyclic group". We develop and study protocols where, at least theoretically, non-abelian groups can be used to share a secret or exchange random private keys between two people over an insecure channel. This development is significant because nilpotent or, more specifically p-groups, have nice presentations and computations in those groups are fast and easy [62, Chapter 9]. So our work can be seen as a nice application of the advanced and developed subject of p-groups and computations with p-groups.

The frequently used public key cryptosystems are slow and use mainly number theoretic complexity. The specific "cryptographic primitive" that we have in mind is "The Discrete Logarithm Problem", DLP for short. DLP is general enough to be defined in an arbitrary cyclic group as follows. Let $G = \langle g \rangle$ be a cyclic group generated by g and let $g^n = h$. We are given g and h, DLP is to find n [63, Chapter 6]. The security of the discrete logarithm problem depends on the representation of the group. It is trivial in $\mathbb{Z}_n$, but is much harder (no polynomial time algorithm is known) in the multiplicative group of a finite field and even harder (no subexponential time algorithm known) in the group of elliptic curves which are not supersingular [8]. However with the invention of sub-exponential algorithms for breaking the discrete logarithm problem, like the index calculus and Coppersmith's algorithm, multiplicative groups of finite fields are no longer that attractive, especially the ones of characteristic 2.

Discrete exponentiation is also used in many other groups like in elliptic curves, in which case a cyclic group or a big enough cyclic component of an abelian group is used. In this chapter we propose a generalization of the DLP or more specifically the Diffie-Hellman key exchange protocol in situations where the group has more than one generator, i.e., in a finitely presented nonabelian group. Let f be an automorphism of a finitely presented group G generated by $\{a_1, a_2, \ldots, a_n\}$. If one knows the action of f on $a \in G$, i.e., $f(a)$, it is still difficult to tell the action of f on any other $b \in G$ i.e., $f(b)$. We describe this in detail later under the name "general discrete logarithm problem". In this chapter we work with finitely presented groups in terms of generators and relations and do not consider any representation of that group. However, this seems to be a good idea for future research.

Now suppose for a moment that $G = \langle g \rangle$ is a cyclic group and that we are given g and g^n where $\gcd(n, |G|) = 1$. DLP is to find n. Notice that in this case the map $x \mapsto x^n$ is an automorphism. If we conjecture that finding the automorphism is finding n then one way to see DLP, in the language of group theory, is to find the automorphism from its action on one element. This is the central idea that we want to generalize to nonabelian finitely presented groups, especially nilpotent group of class 2. This explains our choice of the name "general discrete logarithm problem".

To work with a finitely presented group and its automorphisms the following properties of the group are needed.

- A consistent and natural representation of the elements in the group.
- Computation in the group should be fast and easy.
- The automorphism group should be known and the automorphisms should have a nice enough presentation so that images can be computed quickly.

We note at this point that for a p-group the first two requirements are satisfied [62, Chapter 9].

2.2 Some Notation and Definitions

We now describe some of the definitions and notation that will be used in this paper. The notation used are standard:

- G will denote a finite group. $Z = Z(G)$ denotes the center of the group G.
- $G' = [G, G]$ is the commutator subgroup of G.
- $\mathrm{Aut}(G)$ and $\mathrm{Aut}_c(G)$ are the group of automorphisms and the group of central automorphisms of G respectively.
- $\Phi(G)$ is the Frattini subgroup of G, which is the intersection of all maximal subgroups of G.
- We denote the commutator of a, b by $[a, b]$ where $[a, b] = a^{-1}b^{-1}ab$.
- The exponent of a p-group G, denoted by $\exp(G)$, is the exponent of the largest power of p that is the order of an element in G.

The following commutator formulas hold for any element a,b and c in any group G.

(a) $a^b = a[a, b]$

(b) $[ab, c] = [a, c]^b[b, c] = [a, c][a, c, b][b, c]$; it follows that in a nilpotent group of class 2, $[ab, c] = [a, c][b, c]$

(c) $[a, bc] = [a, c][a, b]^c = [a, c][a, b][a, b, c]$; it follows that in a nilpotent group of class 2, $[a, bc] = [a, b][a, c]$

(d) $[a, b]^{-1} = [b, a]$

The proofs of these formulas follow from direct computation or can be found in [32].

Definition 2.2.1 (Miller Group). *A group G is called a Miller group if it has an abelian automorphism group, in other words, if Aut(G) is commutative then the group G is called a Miller Group.*

Definition 2.2.2 (Central Automorphisms). *Let G be a group, then $\phi \in Aut(G)$ is called a central automorphism if $g^{-1}\phi(g) \in Z(G)$ for all $g \in G$. Alternately, one might say that ϕ is a central automorphism if $\phi(g) = gz_g$ where $z_g \in Z(G)$ depends on g.*

Apart from inner automorphisms, central automorphisms are second best in terms of having a nice description. So they are very attractive for cryptographic purposes, since it is easy to describe the automorphisms and compute the image of an arbitrary element.

Theorem 2.2.3. *The centralizer of the group of inner automorphisms is the group of central automorphisms. Moreover a central automorphism fixes the commutator subgroup elementwise.*

This theorem first appears in [21] who refers to [26] and [70].

Definition 2.2.4 (Polycyclic Group). *Let G be a group, a finite series of subgroups in G*

$$G = G_0 \trianglerighteq G_1 \trianglerighteq G_2 \trianglerighteq G_3 \trianglerighteq \ldots \trianglerighteq G_n = 1$$

is a polycyclic series if G_i/G_{i+1} is cyclic and G_{i+1} is a normal subgroup of G_i. Any group with a polycyclic series is called a polycyclic group.

It is easy to prove that finitely generated nilpotent groups are polycyclic and so any finitely generated p-group is polycyclic. Let a_i be an element in G_i whose image generates G_i/G_{i+1}. Then the sequence $\{a_0, a_1, a_2, \ldots, a_n\}$ is called a polycyclic generating set. It is easy to see that $g \in G$ can be written as $g = a_0^{\alpha_0} a_1^{\alpha_1} a_2^{\alpha_2} \ldots a_n^{\alpha_n}$, where α_i are integers. If $g = a_0^{\alpha_0} a_1^{\alpha_1} a_2^{\alpha_2} \ldots a_n^{\alpha_n}$ where $0 \le \alpha_i < m_i$, $m_i = |G_i : G_{i+1}|$ then the expression is a *collected word.* Each element $g \in G$ can be expressed by a unique collected word. Computation with these collected words is easy and implementable by computer, for more information on this topic see [62, Section 9.4] and also [23, polycyclic package].

2.3 Key Exchange

We want to follow the Diffie-Hellman key exchange protocol using a commutative subgroup of the automorphism group of a finitely presented group G. The security of the Diffie-Hellman key exchange protocol in a cyclic group rests on the following three factors:

The discrete logarithm problem.

The Diffie-Hellman problem.

The decision Diffie-Hellman Problem [9, 11, 22, 61, 67].

We have already described the discrete logarithm problem. The Diffie-Hellman problem is the following: Let $G = \langle g \rangle$ be a cyclic group of order n. One knows g, g^a and g^b, and the problem is to find g^{ab}. It is not known if DLP is equivalent to DHP. The decision Diffie-Hellman problem (DDH) is more subtle. Suppose that DHP is a hard problem, so it is impossible to compute g^{ab} from g^a, g^b and g. But what happens if someone can compute 80% of the shared secret from g^a, g^b and g, then the adversary will have 80% of the shared secret, that is most of the private key. This is clearly unacceptable. It is often hard to formalize DDH in exact mathematical terms, see [11, Section 3], the best formalism offered is a randomness criterion for the bits of the key. In DDH we ask the question: given the triple g^a, g^b and g^c; is $c = ab \mod n$? But there is no known link between DDH and any mathematically hard problem for the Diffie-Hellman key exchange protocol in cyclic groups.

Clearly, solving the discrete logarithm problem solves the Diffie-Hellman problem and solving the Diffie-Hellman problem solves the decision Diffie-Hellman problem.

As is usual, we denote by Alice and Bob, two people trying to set up a private key over an insecure channel to communicate securely and by Oscar an eavesdropping adversary. In this thesis the shared secret or the private key is an element of a finitely presented group G.

2.3.1 The General Discrete Logarithm Problem

Let $G = \langle a_1, a_2, \ldots, a_n \rangle$ and $f : G \to G$ be a non identity automorphism. Suppose one knows $f(a)$ and $a \in G$ then GDLP is to find $f(b)$ for any b in G. Assuming the word problem is easy or presentation of the group is by means of generators, GDLP is equivalent to finding $f(a_i)$ for all i which gives us a complete knowledge of the automorphism. So in other words the cryptographic primitive GDLP is equivalent to, "*finding the automorphism f from the action of f on only one element*".

2.3.2 The General Diffie-Hellman Problem

Let $\phi, \psi : G \to G$ be arbitrary automorphisms, such that $\phi\psi = \psi\phi$ and assume one knows a, $\phi(a)$ and $\psi(a)$. Then GDHP is to find $\phi(\psi(a))$. Notice that GDHP is a restricted form of GDLP, because in the case of GDHP one has to compute $\phi(\psi(a))$ for some fixed a, not $\phi(b)$ for an arbitrary b in G. We now describe two key exchange protocols and do some cryptanalysis. We denote by G a finitely presented group and S an abelian subgroup of $\mathrm{Aut}(G)$.

2.4 Key Exchange Protocol I

Alice and Bob want to set up a private key. They select a group G and an element $a \in G \setminus Z(G)$ over an insecure channel, i.e., a and G are public. Then Alice picks a random automorphism $\phi_A \in S$ and sends Bob $\phi_A(a)$. Bob similarly picks a random automorphism $\phi_B \in S$ and sends Alice $\phi_B(a)$. Both of them can now compute $\phi_A(\phi_B(a)) = \phi_B(\phi_A(a))$ which is their private key for symmetric transmission.

2.4.1 Comments on Key Exchange Protocol I

Though initially it might seem that we don't have enough information to know the automorphisms ϕ_A and ϕ_B, it turns out that if we are using inner automorphisms, then the security of the above scheme actually rests on the conjugacy problem.

Let $\phi_A(a) = x^{-1}ax$ for some x and let $\phi_B(a) = y^{-1}ay$. Then $\phi_A(\phi_B(a)) = (xy)^{-1}a(xy)$. Since, a, $\phi_A(a)$ and $\phi_B(a)$ are known, if the conjugacy problem is easy in the group then anyone can find x and y and break the system.

In the key exchange protocol I Oscar knows G and a. If the automorphisms in use are central automorphisms, then he also sees $\phi_A(a) = az_{\phi_A,a}$ and $\phi_B(a) = az_{\phi_B,a}$. Oscar can find $z_{\phi_A,a}$ and $z_{\phi_B,a}$. Now if G is a special p-group ($G' = Z(G) = \Phi(G)$) then $Z(G)$ is fixed elementwise by both ϕ_A and ϕ_B. Then

$$\phi_A(\phi_B(a)) = \phi_A(az_{\phi_B,a}) = az_{\phi_A,a}z_{\phi_B,a} \tag{2.1}$$

Oscar knows a and can compute $z_{\phi_A,a}$ and $z_{\phi_B,a}$ and can find the private key $\phi_A(\phi_B(a))$. In the literature all examples of Miller p-group with odd prime p are special (see Section 2.9) and the above key exchange is fatally flawed for those groups.

2.5 Key Exchange Protocol II

In this case Alice and Bob want to set up a private key and they set up a group G over an insecure channel. Alice chooses a random non-central element g and a random automorphism $\phi_A \in S$ and sends Bob $\phi_A(g)$. Bob picks another automorphism $\phi_B \in S$ and computes $\phi_B(\phi_A(g))$ and sends it back to Alice. Alice knowing ϕ_A computes ϕ_A^{-1} which gives her $\phi_B(g)$ and picks another random automorphism $\phi_H \in S$ and computes $\phi_H(\phi_B(g))$ and sends it back to Bob. Bob knowing ϕ_B computes ϕ_B^{-1} which gives him $\phi_H(g)$ which is their private key. Notice that Alice never reveals g in public.

2.5.1 Comments on Key Exchange Protocol II

Notice that for central automorphisms, ϕ_A and ϕ_B, $\phi_A(g) = gz_{\phi_A,g}$, since g is not known we don't know $z_{\phi_A,g}$ but if G is special ($Z(G) = G' = \Phi(G)$) then $\phi_B(gz_{\phi_A,g}) = gz_{\phi_B,g}z_{\phi_A,g}$ from which $z_{\phi_B,g}$ can be found. Then $\phi_H(\phi_B(g)) = gz_{\phi_B,g}z_{\phi_H,g}$, hence one can find $gz_{\phi_H,g}$ which is $\phi_H(g)$ and the scheme is broken. As one clearly sees, this attack is not possible if the group is not special.
The reader might have noticed at this point that all the attacks are GDHP. So certainly in some groups GDHP is easy.

As we know, any automorphism in G can be seen as the restriction of an inner automorphism in $\mathrm{Hol}(G)$, see [37, 68] for further details on the holomorph of a group, hence solving the conjugacy problem in $\mathrm{Hol}(G)$ will break the system for any automorphism. On the other hand the operation of $\mathrm{Hol}(G)$ is twisted so it is possible that the conjugacy problem in $\mathrm{Hol}(G)$ is difficult even though it is easy in G. Any cyclic group is a Miller group so success of the holomorph attack would prove insecurity to the DLP, so we believe that the holomorph attack won't be successful in many cases. Though more work needs to be done on this.

2.6 Key Exchange using Braid Groups

In [34] a similar key exchange protocol has been defined. In this section we mention some similarities of their approach with ours. We also mention how our system generalizes their system using braid groups. See also [14].

We define braid group as a finitely presented group, though there are fancy pictorial ways to look at braids and multiplication of braids. An interested reader can look in [5, 17]. The Braid Group B_n with n-strands is defined as:

$$B_n = \left\langle \sigma_1, \sigma_2, \ldots, \sigma_{n-1} : \ \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j \text{ if } |i-j| = 1, \ \sigma_i\sigma_j = \sigma_j\sigma_i \text{ if } |i-j| \geq 2 \right\rangle$$

In [34] the authors found two disjoint subgroups A and B of the group of inner automorphisms $\mathrm{Inn}(B_n)$, such that for $\phi \in A$ and $\psi \in B$, $\phi(\psi(g)) = \psi(\phi(g))$. Then the key exchange proceeds similar to key exchange protocol I above with the restriction that Alice chooses automorphisms from A and Bob chooses automorphisms from B. There is also a different approach to key exchange in braid group as in [3, 4].

In the same spirit as [34] we can develop a key exchange protocol similar to key exchange protocol I, where we take two subgroups A and B in $\mathrm{Aut}(G)$ such that for $\phi \in A$ and $\psi \in B$, $\phi(\psi(g)) = \psi(\phi(g))$. The use of inner automorphisms is only possible when the conjugacy or the generalized conjugacy problem (conjugator search problem) is known to be hard.

There are significant differences in our approach to that of the approach in [34]. In [34] the authors choose a group and then try to use that group in cryptography. We, on the other hand, take as fundamental concept the discrete logarithm problem, generalize it using automorphisms of

a non-abelian group and then look for groups favorable to us. The fact that the central idea in braid group key exchange turns out to be similar to ours is encouraging.
It is intuitively clear at this point that we should start looking for groups with an abelian automorphism group, i.e., Miller groups.

2.7 Some useful facts from group theory.

The term Miller Group is not that common in literature. It was introduced by Earnley in [19]. Miller was the first to study groups with abelian automorphism group in [48]. Cyclic groups are good examples of Miller groups. G.A. Miller also proved that no non-cyclic abelian group has an abelian automorphism group.

Charles Hopkins began a list of necessary conditions for a Miller group in 1927 [28]. He complained that very little is known about those groups. The same is true today. Except for some sporadic examples of groups with abelian automorphism groups, there is no sufficient condition known for a group to be Miller.
We now state some known facts about Miller groups which are available in the literature and which we shall need later. For proof of these theorems which we present in a rapid fashion, the reader can look in any standard text books like [32, 56] or the references there.

Proposition 2.7.1. *Let G be a non-abelian Miller group, then G is nilpotent and of class 2.*

Proof. It follows from the fact that the group of inner automorphisms is abelian and $G/Z(G) \cong \mathrm{Inn}(G)$. •

Since a nilpotent group is the direct product of its Sylow p-subgroups S_p, and $\mathrm{Aut}(A \times B) = \mathrm{Aut}(A) \times \mathrm{Aut}(B)$ whenever A and B are of relatively prime order, it is enough to study Miller p-groups for prime p.

Proposition 2.7.2. *Let G be a p-group of class 2, then $exp(G') = exp(G/Z(G))$.*

Proposition 2.7.3. *In a p-group of class 2, $(xy)^n = x^n y^n [y, x]^{\frac{n(n-1)}{2}}$. Furthermore if $exp(G') = n$ is odd, then $(xy)^n = x^n y^n$.*

By definition in a Miller group all automorphisms commute. Since central automorphisms are the centralizer of inner automorphisms, we have proved the following theorem.

Theorem 2.7.4. *In a Miller group G, all automorphisms are central.*

It follows that to show a group is not Miller, all we have to do is to produce a non-central automorphism.

Proposition 2.7.5. *If the commutator subgroup and the center of G coincide then every pair of central automorphisms commute.*

Proof. Let G be a group such that $G' = Z(G)$. Then let ϕ and ψ be central automorphisms given by $\phi(x) = xz_{\phi,x}$ and $\psi(x) = xz_{\psi,x}$ where $z_{\phi,x}, z_{\psi,x} \in G'$. Then

$$\psi(\phi(x)) = \psi(xz_{\phi,x}) = \psi(x)z_{\phi,x} = xz_{\psi,x}z_{\phi,x} = xz_{\phi,x}z_{\psi,x} = \phi(\psi(x)).$$

•

Definition 2.7.6 (Purely non-abelian group). *A group G is said to be a purely nonabelian group (PN group for short) if whenever $G = A \times B$ where A and B are subgroups of G with A abelian, then $A = 1$. Equivalently G has no abelian direct factor.*

Let $\sigma : G \to G$ be a central automorphism. Then we define a map $f_\sigma : G \to Z(G)$ as follows: $f_\sigma(g) = g^{-1}\sigma(g)$. Clearly this map defines a homomorphism. The map $\sigma \mapsto f_\sigma$ is clearly a one-one map. Conversely, if $f \in \mathrm{Hom}(G, Z(G))$ then we define a map $\sigma_f(g) = gf(g)$, $x \in G$. Clearly σ_f is an endomorphism. It is easy to see that

$$\mathrm{Ker}(\sigma_f) = \{x \in G : \; f(x) = x^{-1}\}$$

Hence it follows that σ_f is an automorphism if and only if $f(x) \neq x^{-1}$ for all $x \in G$ with $x \neq 1$.

Theorem 2.7.7. *In a purely non-abelian group G, the correspondence $\sigma \to f_\sigma$ is a one-one map of $Aut_c(G)$ onto $Hom(G, Z(G))$*

Proof. See [2]. •

Notice that for any $f \in \mathrm{Hom}(G, Z(G))$ there is a map $f' \in \mathrm{Hom}(G/G', Z(G))$ since $f(G') = 1$. Furthermore notice that corresponding to $f' \in \mathrm{Hom}(G/G', Z(G))$ there is a map $f : G \to Z(G)$ explained in the following diagram

$$G \xrightarrow{\;\eta\;} G/G' \xrightarrow{\;f'\;} Z(G)$$

where η is the natural epimorphism.

Let G be a p-group of class 2, such that $\exp(Z(G)) = a$, $\exp(G') = b$ and $\exp(G/G') = c$ and let $d = \min(a, c)$. Notice that from the fundamental theorem of abelian groups

$$G/G' = A_1 \oplus A_2 \oplus \ldots \oplus A_r \;\text{ where }\; A_i = \langle a_i \rangle$$

$$Z(G) = B_1 \oplus B_2 \oplus \ldots \oplus B_s \;\text{ where }\; B_i = \langle b_i \rangle$$

$r, s \in \mathbb{N}$ be the direct decomposition of G/G' and $Z(G)$. If the cyclic component $A_k = \langle a_k \rangle$ has exponent greater or equal to the exponent of $b \in Z(G)$, then one can define a homomorphisms $f : G/G' \to Z(G)$ as follows

$$f(a_i) = \begin{cases} b & \text{where } \; i = k \\ 1 & \text{where } \; i \neq k \end{cases}$$

From this discussion it is clear that for $f \in \mathrm{Hom}(G, Z(G))$, $f(G)$ generates the subgroup

$$\mathcal{R} = \{z \in Z(G) : \; |z| \leq p^d, \; d = \min(a, c)\}$$

Definition 2.7.8 (Height). *In any abelian p-group A written additively, there is a descending sequence of subgroups*

$$A \supset pA \supset p^2A \supset \ldots \supset p^nA \supset p^{n+1}A \supset \ldots$$

Then $x \in A$ is of height n if $x \in p^nA$ but not in $p^{n+1}A$. In other words the elements of height n are those that drop out of the chain in the $(n + 1)^{th}$ inclusion.

For further information on height see [31].

For a class 2 group we have

$$\exp(G/G') \geq \exp(G/Z(G)) = \exp(G')$$

it follows that $c \geq b$. Hence if $d = \min(a, c)$ then either $d = b$ or $d > b$.

Let height$(xG') \geq b$, then $xG' = y^{p^b}G'$ for some $y \in G$. Then for any $F \in \mathrm{Hom}(G, G')$, $F(yG')^{p^b} = 1$ implying $xG' \in F^{-1}(1)$. Conversely, let height$(xG') < b$. Then from the previous discussion it is clear that there is a $F' \in \mathrm{Hom}(G/G', G')$ such that xG' is not in the kernel of F', consequently there is a $F \in \mathrm{Hom}(G, G')$ such that $x \notin \ker(F)$. Combining these two facts we see that:

$$\mathcal{K} = \bigcap_{F \in \mathrm{Hom}(G,G')} F^{-1}(1) = \{x \in G : \ \text{height}(xG') \geq b\}$$

Proposition 2.7.9. $\mathcal{K} \subseteq \mathcal{R}$

Proof. In a class 2 group, if $x \in \mathcal{K}$ then $xG' = y^{p^b}G'$ for some $y \in G$ and $\exp(G/Z) = b$ and $G' \subseteq Z(G)$, hence $x \in Z(G)$.

Let $x \in \mathcal{K}$, then height$(xG') \geq b$, hence there is a $y \in G$ such that $y^{p^b}G' = xG'$ i.e. $x = y^{p^b}z$ where $z \in G'$ and $y^{p^c} \in G'$ and $c \geq b$. We have

$$x^{p^c} = (y^{p^b})^{p^c} z^{p^c} = (y^{p^c})^{p^b} = 1$$

Hence $|x| \leq \min(p^a, p^c)$ which implies that $x \in \mathcal{R}$. This proves that $\mathcal{K} \subseteq \mathcal{R}$. •

Proposition 2.7.10. *For a PN group G of class* 2*, if* $Aut_c(G)$ *is abelian then* $\mathcal{R} \subseteq \mathcal{K}$*.*

Proof. In a PN group, using theorem 2.7.7 and the notation there, two central automorphisms σ and τ commute if and only if $f_\sigma, f_\tau \in \mathrm{Hom}(G, Z(G))$ commute. Then for any $f \in \mathrm{Hom}(G, Z(G))$ and $F \in \mathrm{Hom}(G, G')$ we have that $f \circ F = F \circ f = 1$ because $f(G') = 1$. Then $F \circ f(G) = 1$ proving that $\mathcal{R} \subseteq \mathcal{K}$. •

Combining the above two propositions, we just proved that in a PN group G of class 2, if $\mathrm{Aut}_c(G)$ is abelian then $\mathcal{R} = \mathcal{K}$. As discussed earlier there are two cases $d = b$ and $d > b$. Adney and Yen prove that:

Proposition 2.7.11. *If G is a non-abelian p group of class* 2*, and* $Aut_c(G)$ *is abelian with* $d > b$*, then* $\mathcal{R}/G'$ *is cyclic.*

Proof. See [2, Theorem 3]. •

Theorem 2.7.12. *Adney and Yen [2].*
Let G be a purely non-abelian group of class 2*, p odd, let* $G/G' = \prod_{i=1}^{n} \langle x_i G' \rangle$*. Then the group* $Aut_c(G)$ *is abelian if and only if*

(i) $\mathcal{R} = \mathcal{K}$

(ii) either $d = b$ *or* $d > b$ *and* $\mathcal{R}/G' = \langle x_1^{p^b} G' \rangle$

Proof. See [2, Theorem 4]. •

From the proof of Proposition 2.7.5 it follows that in a group G with $Z(G) \leq G'$, the central automorphisms commute.

Theorem 2.7.13. *The group of central automorphisms of a p-group G, where p is odd, is a p-group if and only if G has no abelian direct factor.*

Proof. See [57, Theorem B] and its corollary. •

At this point we concentrate on building a cryptosystem. We note that Miller groups in particular have no advantage over groups with abelian central automorphism group. It is hard to construct Miller groups and there is no known Miller group for odd prime which is not special, so we now turn towards a group G such that $\mathrm{Aut}(G)$ may not be abelian but $\mathrm{Aut}_c(G)$ is abelian. We propose to use $\mathrm{Aut}_c(G)$ rather than $\mathrm{Aut}(G)$ in the key exchange protocols described earlier.

2.8 Signature Scheme based on the conjugacy problem

Assume that we are working with a group G with commuting inner automorphisms, for example, a group of class 2 with abelian inner automorphism group.

Alice publishes α and β where $\beta = a^{-1}\alpha a$, $a, \alpha \in G$ and keeps a a secret. To sign a plaintext $x \in G$ she picks an arbitrary element $k \in G$ and computes $\gamma = k\alpha k^{-1}$ and then computes δ such that $x = (\delta k)(a\gamma)^{-1}$. Now notice that

$$\begin{aligned}
x\alpha x^{-1} &= (\delta k)(a\gamma)^{-1}\alpha((\delta k)(a\gamma)^{-1})^{-1} \\
&= (\delta k)\gamma^{-1}a^{-1}\alpha a\gamma k^{-1}\delta^{-1} \\
&= \delta\gamma^{-1}a^{-1}k\alpha k^{-1}a\gamma\delta^{-1} && \text{Inner automorphisms commute} \\
&= \delta\gamma^{-1}a^{-1}\gamma a\gamma\delta^{-1} \\
&= \delta a^{-1}\gamma a\delta^{-1} \\
&= \delta(k\beta k^{-1})\delta^{-1} && \gamma = k\alpha k^{-1} \Rightarrow a^{-1}\gamma a = k\beta k^{-1}
\end{aligned}$$

So to sign a message $x \in G$ Alice computes δ as mentioned and sends $x, (k\delta)$. To verify the message one computes $L = x\alpha x^{-1}$ and $R = \delta k\beta(\delta k)^{-1}$. If $L = R$ then the message is authentic otherwise not.

There is a similar signature scheme in [33], where the authors exploit the gap between the computational version (conjugacy problem) and the decision version of the conjugacy problem (conjugator search problem) in Braid Groups. We followed the ElGamal signature scheme closely [63, Chapter 7].

2.8.1 Comments on the above Signature Scheme

If one can solve the conjugacy problem in the group then from the public information α and β he can find out a and our scheme is broken. The conjugacy problem is known to be hard in some groups and hence it seems to be a reasonable assumption at this moment. There is another worry: if Alice sends k and δ separately then one can find a from the equation $x = (\delta k)(a\gamma)^{-1}$, since γ is computable. However, this is circumvented easily by sending the product δk not δ and k individually and keeping k random.

2.9 An interesting family of p-groups

It is well known that cyclic groups have abelian automorphism groups. The first person to give an example of an non-abelian group with an abelian automorphism groups is G.A. Miller in [48] which was generalized by Struik in [64]. There are three non-abelian groups with abelian automorphism group in Hall-Senior table [25], they are nos. 91, 92 and 99. Miller's example is no. 99. In [29], Jamali generalized no. 91 and 92. His generalization of no. 91 is in one direction, it increases the exponent of the group.

Jamali in the same paper generalizes group no. 92 in two directions, the size of the exponent and the number of generators. His generalization was restrictive in that it works only for the prime 2. There are other examples of families of Miller p-groups in literature, the most notable one is the family of p-groups for any arbitrary prime p given by Jonah and Konisver in [30] which was generalized to an arbitrary number of generators by Earnley in [19]. There are examples by Martha Morigi in [50] and Heineken and Liebeck in [27] also. All these examples of Miller groups given in [19, 27, 30, 50] are special groups i.e., the commutator and the center are same. For special groups the key exchange protocols does not work as noted earlier. So there is no Miller p-group, readily available in literature, for arbitrary prime p which can be used right away in the construction of the protocol. The only other source are groups nos. 91, 92 and 99 in Hall Senior table [25] and their generalizations, notice that these groups are not special but are 2-group. Of the three generalizations, the generalization of no. 92 best fits our criterion since it has been generalized in two directions, *viz.* the number of generators and exponent of the center and moreover it is not special and $Z(G) = A \times G'$ where A is a cyclic group. So once we generalize it for arbitrary primes, it has "three degrees of freedom", the number of generators, the exponent of center and the prime; which makes it attractive for cryptographic purposes.

In the rest of the section we use Jamali's definition in [29] to define a family of p-groups for arbitrary prime. So this family is a generalization of Jamali's example and assuming transitivity of generalizations, ultimately a generalization of group no. 92 in the Hall-Senior table [25]. We study automorphisms of this group and show that the group is Miller if and only if $p = 2$, but groups in this family always have an abelian central automorphism group which is fairly large. We then attempt to build a key exchange protocol as described earlier using the central automorphisms. We start with a definition of the group.

Definition 2.9.1. *Let $G_n(m,p)$ be a group generated by $n+1$ elements $\{a_0, a_1, a_2, \ldots, a_n\}$, let p be any prime and $m \geq 2$ and $n \geq 3$ be integers. The group is defined by the following relations:*

$$a_1^p = 1, \quad a_2^{p^m} = 1, \quad a_i^{p^2} = 1 \quad \textit{for} \quad 3 \leq i \leq n, \quad a_{n-1}^p = a_0^p.$$
$$[a_1, a_0] = 1, \quad [a_n, a_0] = a_1, \quad [a_{i-1}, a_0] = a_i^p \quad \textit{for} \quad 3 \leq i \leq n.$$
$$[a_i, a_j] = 1 \quad \textit{for} \quad 1 \leq i < j \leq n.$$

Proposition 2.9.2. *We show that $[a_0, a_i^n] = [a_0, a_i]^n$ is true in G for $i \geq 2$.*

Proof. Clearly the proposition is true for $n = 1$. Assume that the proposition is true for $n = k$, i.e., $[a_0, a_i^k] = [a_0, a_i]^k$. Then

$$\begin{aligned}
&[a_0, a_i^{k+1}] \\
&\quad = \quad [a_0, a_i^k][a_0, a_i][[a_0, a_i^k], a_i] \\
&\quad = \quad [a_0, a_i]^{k+1}[[a_0, a_i]^k, a_i] \\
&\quad = \quad [a_0, a_i]^{k+1},
\end{aligned}$$

since $[a_i, a_j] = 1$ for $1 \leq i, j \leq n$. This proves the proposition by the principle of mathematical induction. •

It follows from the above discussion and the relations in the group $G_n(m,p)$ that $[a_0, [a_0, a_i]] = [a_0, a_k^p] = 1$ for all i where k is either $i+1$ or 1 depending on i. This implies that in the group G any commutator in the generators of weight 3 is zero.

We take an example to demonstrate that in $G_n(m, p)$ for generators $x_1, y_1, x_2, y_2 \in \{a_0, a_1, a_2, \ldots, a_n\}$

$$[x_1x_2, y_1y_2] = [x_1, y_1][x_1, y_2][x_2, y_1][x_2, y_2].$$

Notice that

$$[x_1x_2, y_1y_2] = [x_1, y_1y_2][[x_1, y_1y_2], x_2][x_2, y_1y_2]$$

Now

$$[x_1, y_1y_2] = [x_1, y_2][x_1, y_1][[x_1, y_2], y_1] = [x_1, y_2][x_1, y_1]$$

Then clearly, $[x_1, y_1y_2][[x_1, y_1y_2], x_2][x_2, y_1y_2]$ becomes $[x_1, y_1y_2][x_2, y_1y_2]$. Using similar arguments as before it follows that $[x_1, y_1y_2] = [x_1, y_1][x_1, y_2]$ and $[x_2, y_1y_2] = [x_2, y_1][x_2, y_2]$. The rest follows from direct computation. We used commutator identities from Section 2.2.

Lemma 2.9.3. *The group $G_n(m, p)$ is a PN group.*

Proof. We denote $G_n(m, p)$ by G. Let $G = A \times B$ where A is an abelian group. Then $G/B \cong A$ an abelian group, hence $G' \subseteq B$. From the fact that $G = A \times B$, it follows that $g \in G$ has a unique expression of the form $g = ab$ where $a \in A$ and $b \in B$. Then for any $x \in A$

$$xg = xab = axb = abx = gx$$

i.e., $x \in Z(G)$. This implies that $A \subseteq Z(G)$. Now recall that in G, $Z(G) = \langle a_2^p \rangle \times G'$, from which it follows that $Z(G) \subseteq G^p = A^p \times B^p$. Since $A \subseteq Z(G)$ so $A \subseteq A^p \times B^p$, this implies that $A \subseteq A^p$ which proves that $A = 1$. •

The following facts about the group $G_n(m, p)$ follows:

a $G_n(m, p)'$ the derived subgroup of $G_n(m, p)$ is an elementary abelian group $\langle a_1, a_3^p, \ldots a_n^p \rangle \simeq \mathbb{Z}_p^{n-1}$.

b $Z(G_n(m, p)) = \langle a_2^p \rangle \times G'$.

c $G_n(m, p)$ is a p-group of class 2.

d $G_n(m, p)$ is a PN group.

Proposition 2.9.4. *$G_n(m, p)$ is a polycyclic group and every element $g \in G_n(m, p)$ can be uniquely expressed in the form $g = a_0^{\alpha_0} a_1^{\alpha_1} a_2^{\alpha_2} a_3^{\alpha_3} \ldots a_n^{\alpha_n}$, where $0 \le \alpha_i < p$ for $i = 0, 1$; $0 \le \alpha_2 < p^m$, $0 \le \alpha_i < p^2$ for $i = 3, 4, \ldots, n$.*

Proof. Let us define $G_0 = G_n(m, p) = \langle a_0, a_1, a_2, \ldots, a_n \rangle$, $G_1 = \langle a_1, a_2, \ldots a_n \rangle$ and similarly $G_k = \langle a_k, a_{k+1}, \ldots, a_n \rangle$ for $k \le n$. Since G_1 is a finitely generated abelian group, it is a polycyclic group [62, Proposition 3.2]. It is fairly straightforward to see that

$$G_1 \rhd G_2 \rhd \ldots \rhd G_n \rhd \langle 1 \rangle$$

is a polycyclic series and $\{a_1, \ldots, a_n\}$ a polycyclic generating sequence of G_1.

It is easy to see from the relations of the group that G_1 is normal in G_0 and G_0/G_1 is cyclic and generated by $\langle a_0G_1 \rangle$. It follows that $\langle a_iG_{i+1} \rangle = G_i/G_{i+1}$ and $|a_iG_{i+1}| = |a_i|$ for $i = 0, 1, 2, \cdots, a_n$ and hence any element of the group has a unique representation of the form $g = a_0^{\beta_0} a_1^{\beta_1} a_2^{\beta_2} \ldots a_n^{\beta_n}$ where $0 \le \beta_0, \beta_1 < p$, $0 \le \beta_2 < p^m$ and $0 \le \beta_i < p^2$ for $3 \le i < p$. We would call an element represented in the above form a *collected word*. See also [62, Chapter 9, Proposition 4.1]. •

Computation with $G_n(m, p)$: In our group $G_n(m, p)$, which is of class 2, i.e. commutators of weight 3 are the identity, computations become real nice and easy. Let us demonstrate the product of two collected words $g = a_0^{\alpha_0} a_1^{\alpha_1} a_2^{\alpha_2} a_3^{\alpha_3} a_4^{\alpha_4}$ and $h = a_0^{\beta_0} a_1^{\beta_1} a_2^{\beta_2} a_3^{\beta_3} a_4^{\beta_4}$. To compute gh we use concatenation and form the word $a_0^{\alpha_0} a_1^{\alpha_1} a_2^{\alpha_2} a_3^{\alpha_3} a_4^{\alpha_4} a_0^{\beta_0} a_1^{\beta_1} a_2^{\beta_2} a_3^{\beta_3} a_4^{\beta_4}$ and note that a_i's commute except for a_0 hence one tries to move a_0 towards the left using the identity

$$a_i a_0 = a_0 a_i [a_i, a_0] = \begin{cases} a_0 a_i a_{i+1}^p & \text{for} \quad 1 \leq i < n \\ a_0 a_i a_1 & \text{for} \quad i = n \end{cases}$$

Further note that since commutators are in the center of the group, a_{i+1}^p or a_1 can be moved anywhere. Once a_0 is moved to the extreme left the word formed is the collected word of gh. This process in the literature is often referred to as "collection". Computing the inverse of an element can be similarly achieved.

We now prove that the group of central automorphisms of the group $G_n(m, p)$ for an arbitrary prime p is abelian. For the sake of simplicity we denote $G_n(m, p)$ by G for the rest of the chapter, and use notation from Theorem 2.7.12.

Lemma 2.9.5. *In G, $\mathcal{R} = Z(G) = \mathcal{K}$.*

Proof. Using the notation from theorem 2.7.12, we see that in G, $a = m - 1$, $b = 1$ and $c = m$ hence $d = m - 1$. Clearly, $\mathcal{R} = Z(G)$ hence $\mathcal{K} \subseteq Z(G)$.
Let $x \in Z(G)$, if $x \in G'$ then height$(xG') = \infty$ and we are done. If not then $x = z_1 z_2$ where $z_1 \in \langle a_2^p \rangle$ and $z_2 \in G'$. Then $xG' = z_1 G'$ and hence height$(xG') \geq 1$. •

It is easy to see that $\mathcal{R}/G' = Z(G)/G' = \langle a_2^p G' \rangle$ and hence from theorem 2.7.12 we have proved the following theorem:

Theorem 2.9.6. *$Aut_c(G)$ is abelian.*

2.9.1 The Automorphisms of $G_n(m, p)$

In this section we describe the automorphisms of groups of this kind. The discussion is in, more than one way, an adaptation of Jamali's work in [29].

From Proposition 2.7.3 it follows that replacing a_0 by $a_{n-1}^{-1} a_0$ we have that $a_0^p = (a_{n-1}^{-p}) a_0^p = 1$ for an odd prime p. Since all the commutator relations remains the same we have a new representation for the group $G_n(m, p)$, for an odd prime p as follows:

$$a_1^p = 1, \quad a_2^{p^m} = 1, \quad a_i^{p^2} = 1 \quad \text{for} \quad 3 \leq i \leq n, \quad a_0^p = 1.$$
$$[a_1, a_0] = 1, \quad [a_n, a_0] = a_1, \quad [a_{i-1}, a_0] = a_i^p \quad \text{for} \quad 3 \leq i \leq n.$$
$$[a_i, a_j] = 1 \quad \text{for} \quad 1 \leq i < j \leq n.$$

This representation proves that $G_n(m, p)$ is a semidirect product of its subgroups $H = \langle a_1, a_2, \ldots, a_n \rangle$ and $\langle a_0 \rangle$. We will use this new representation as it simplifies some relations in the group $G_n(m, p)$.

We in our understanding of the automorphisms would only consider the case $p > 2$. The case for $p = 2$ is already been taken care of by Jamali in [29]. Henceforth p refers to an odd prime.

We now look at $H = \langle a_1, a_2, \ldots, a_n \rangle$ an abelian group of maximal order in G, i.e., of index p. Now assume that K is another abelian subgroup of maximal order in G, we show that $H = K$. Since in a p-group subgroups of maximal order are normal, both H and K are normal. Let $x \in K$ and $x \notin H$, then $\langle x, H \rangle = G$ and $Z(G) = C_G(x) \cap C_G(H)$. Now from [56, Theorem 5.41] we know that for a maximal abelian subgroup H of a p group G, $C_G(H) = H$, hence we have that $Z(G) = C_G(x) \cap H$. Since $x \in K$ and K is an abelian subgroup hence $K \subseteq C_G(x)$ hence $H \cap K \subseteq C_G(x) \cap H = Z(G)$. Now notice that

$$|HK| = \frac{|H||K|}{|H \cap K|}$$

and because the index of $Z(G)$ in H is greater than p we have that

$$|HK| \geq \frac{|H||K|}{|Z(G)|} > |G|.$$

This contradicts the fact that $x \notin H$ and hence $K \subseteq H$, from the maximality of K it follows that $K = H$.

This proves that $H = \langle a_1, a_2, a_3, \dots a_n \rangle$ is the unique maximal abelian normal subgroup of G and hence is a characteristic subgroup. It follows that the H^p is also characteristic subgroup. Corresponding to H we define two decreasing sequences of characteristic subgroups $\{K_i\}_{i=0}^{n-1}$ such that

$$K_0 = H \text{ and } K_i/K_{i-1}^p = Z(G/K_{i-1}^p) \ \ (1 \leq i \leq n-1)$$

and $\{L_i\}$ such that

$$L_0 = H \text{ and } L_i = \{h : \ h \in H, \ h^p \in [G, L_{i-1}]\} \ (1 \leq i \leq n-1).$$

It follows from the fact $H = \langle a_1 \rangle \oplus \langle a_2 \rangle \oplus \dots \oplus \langle a_n \rangle$ that $H^p = \langle a_2^p, a_3^p, \dots, a_n^p \rangle$. From the relations in the group $G_n(m, p)$ it follows that all the nontrivial commutator of $G_n(m, p)$ belongs to H^p except for the $[a_n, a_0] = a_1$. Hence $K_1 = \langle a_1, a_2, \dots, a_n^p \rangle$. A similar analysis reveals that:

$$K_i = \langle a_1, a_2, \dots, a_{n-i}, a_{n-i+1}^p, \dots, a_n^p \rangle \ 1 \leq i \leq n-1$$

$$L_1 = \langle a_1, v, a_3, \dots, a_n \rangle$$

$$L_i = \langle a_1, v, a_3^p, \dots, a_{i+1}^p, a_{i+2}, \dots, a_n \rangle \quad 2 \leq i \leq n-1$$

where $v = a_2^{p^{m-1}}$. For $3 \leq i \leq n$ we have

$$K_{n-i} \cap L_{i-2} = \langle a_1, v, a_3^p, \dots a_{i-1}^p, a_i, a_{i+1}^p, \dots a_n^p \rangle = \langle v, a_i, G' \rangle.$$

Also $K_{n-2} \cap L_0 = \langle a_2, G' \rangle$.

Since $\langle v, a_i, G' \rangle$ and $\langle a_2, G' \rangle$ are characteristic subgroups, for any $\theta \in \mathrm{Aut}(G)$,

$$\begin{aligned} \theta(a_2) &= a_2^{k_2} z \quad \text{where } z \in G' \text{ and } k_2 \in \mathbb{N} \\ \theta(a_i) &= a_i^{k_i} v^{r_i} z \quad \text{where } z \in G', \ k_i \in \mathbb{N}, \quad 0 \leq r_i < p \quad i = 3, 4, \dots, n \end{aligned} \tag{2.2}$$

There are some conditions on k_2 and k_i. To begin with, if θ is an automorphism then the order of g is equal to the order of $\theta(g)$, which implies that $\gcd(k_i, p) = 1$ for all k_i, and we may choose k_i, such that $0 < k_i < p$ for $i = 3, 4, \dots, n$.

Let $\theta(a_0) = a_0^{\beta_0} a_1^{\beta_1} a_2^{\beta_2} \dots a_n^{\beta_n}$ where $0 \leq \beta_0, \beta_1 < p$, $0 \leq \beta_2 < p^m$ and $0 \leq \beta_i < p^2$ for $3 \leq i \leq n$. Since $\theta(a_0^p) = 1$ we have

$$1 = a_2^{p\beta_2} a_3^{p\beta_3} \dots a_n^{p\beta_n} \quad \text{since the order of } a_0 \text{ and } a_1 \text{ is } p$$

implies that $p^{m-1} | \beta_2$ and $p | \beta_i$ for $i = 3, 4, \dots, n$. Hence $\theta(a_0) = a_0^{k_0} v^r z$ where $0 < k_0 < p$, $0 \leq r < p$ and $z \in G'$.

Notice the relation $[a_i, a_0] = a_{i+1}^p$ for $i = 2, 3, \ldots, (n-1)$ implying that $[\theta(a_i), \theta(a_0)] = \theta(a_{i+1})^p = a_{i+1}^{pk_{i+1}}$. It follows from Equation 2.2 and Section 2.2 that $[a_i^{k_i} z_1, a_0^{k_0} z_2] = a_{i+1}^{pk_{i+1}}$ for $z_1, z_2 \in Z(G)$ which is the same as $[a_i^{k_i}, a_0^{k_0}] = a_{i+1}^{pk_{i+1}}$, which implies that $[a_i, a_0]^{k_0 k_i} = a_{i+1}^{pk_{i+1}}$. Recall that G is a p-group of class 2.

Theorem 2.9.7. *Let $\theta : G \to G$ be a map then a necessary and sufficient condition for $\theta \in Aut(G)$ is*

$$\begin{array}{rcl} \theta(a_0) & = & a_0^{k_0} v^{r_0} z_0 \quad 0 < k_0 < p \quad 0 \le r_0 < p \quad \textit{where} \quad z \in G' \\ \theta(a_2) & = & a_2^{k_2} z_2 \quad \textit{where} \quad \gcd(k_2, p) = 1; \quad 0 < k_2 < p^m \\ \theta(a_i) & = & a_i^{k_i} v^{r_i} z_i \quad \textit{where} \quad 0 \le r_i < p, \quad z_i \in G' \\ \theta(a_1) & = & a_1^{k_1} \quad \textit{where} \quad k_1 = k_0 k_n \mod p \end{array}$$

where k_i satisfy the equation $k_0 k_i = k_{i+1} \mod p$, $i = 2, 3, \ldots, (n-1)$, $0 < k_0 < p$ and $\gcd(k_2, p) = 1$ *and θ extended to all of G.*

Proof. It follows from the earlier discussion that the above conditions are necessary.

To see that the conditions are sufficient first notice that for a θ as defined above

$$\begin{array}{rcl} \theta(a_{i+1})^p = \theta[a_i, a_0] & = & [\theta(a_i), \theta(a_0)] \\ & = & [a_i^{k_i} z_1, a_0^{k_0} z_2] \quad z_1, z_2 \in Z(G) \\ & = & [a_i, a_0]^{k_i k_0} \ \text{ since } G \text{ is a } p\text{-group} \\ & = & a_{i+1}^{pk_{i+1}} = \left(a_{i+1}^{k_{i+1}} v^{r_{i+1}} z_{i+1}\right)^p \end{array}$$

This shows that the nontrivial commutator relation in G is satisfied for $i = 2, 3, \ldots, (n-1)$. The case for $i = n$ is similar. The order of the image of a generator is the same as that of the order of the generator because $\gcd(k_i, p) = 1$ and the order of $v^{r_i} z_i$ is p. The other relation of commutativity of the generators is also satisfied. So we just showed that if a potentially multi-valued map θ satisfies the above relations between k_i then it is an endomorphism of G.

We now assume that θ is an endomorphism that satisfies the relations in the theorem, then consider the subgroup of G defined as

$$G^\# = \langle a_0^{k_0} v^{r_0} z_0, a_1^{k_1}, a_2^{k_2} z_2, a_3^{k_3} v^{r_3} z_3, \ldots, a_n^{k_n} v^{r_n} z_n \rangle$$

where k_i, v, r_i and z_i are as defined in the statement of the theorem. We propose to show that the cardinality of $G^\#$ is the same as the cardinality of G and since the image of θ is $G^\#$ that should prove θ is an automorphism.

Let us define a subgroup of $G^\#$ as follows:

$$G_1^\# \quad = \quad \langle a_1^{k_1}, a_2^{k_2} z_2, a_3^{k_3} v^{r_3} z_3, \ldots, a_n^{k_n} v^{r_n} z_n \rangle$$

Since all elements of $G_1^\#$ are words in $a_1, a_2, \ldots, a_n$ hence $G_1^\# \subseteq H$. We now propose to show that $H = G_1^\#$ by showing that $a_1, a_2, \ldots, a_n \in G_1^\#$. Since $\gcd(k_1, p) = 1$ hence if m_1 is the multiplicative inverse of k_1 mod p then $\left(a_1^{k_1}\right)^{m_1} = a_1$ implying that $a_1 \in G_1^\#$. Now we show that $a_i^p \in G_1^\#$ for $i \ge 1$. Notice that $a_i^{pk_i} \in G_1^\#$ then since $\gcd(k_i, p) = 1$ hence by computing the inverse of k_i to the appropriate power of p and then raising $a_i^{pk_i}$ to that power of the inverse we get $a_i^p \in G_1^\#$. This proves that $a_i^{k_i} \in G_1^\#$. Since $\gcd(k_i, p) = 1$ hence there are integers a, b such that $ak_i + bp = 1$. This gives us $a_i \in G_1^\#$ for all $i \ge 1$.

It is clear that $a_0^{k_0} v^{r_0} z_0 \notin H$ otherwise $a_0 \in H$ hence we see that H is a proper subgroup of $G^\#$ and since H is of prime index in $G_n(m, p)$ hence we have that $G^\# = G$. •

From the definition of central automorphisms we see that an automorphism is a central automorphism if and only if $k_0 = 1, k_2 = 1 \mod p$ for $i = 2, 3, 4, \ldots, n$.

We now provide an algorithm to compute an automorphism in G. Choose k_0 such that $0 < k_0 < p$ and k_2 such that $0 < k_2 < p^m$ and $\gcd(k_2, p) = 1$ and then define $k_{i+1} = k_0 k_i \mod p$ for $i = 2, 3, 4, \ldots, (n-1)$. Then use the above theorem to define the automorphism.

Consider the following automorphisms:

$$\begin{aligned} \theta(a_0) &= a_0^{k_0} \quad 0 < k_0 < p \\ \theta(a_2) &= a_2^{k_2} \quad 0 < k_2 < p \\ \theta(a_i) &= a_i^{k_i} \end{aligned}$$

Where k_i satisfy the relation in the above theorem. Then we see that any automorphism is a automorphism of the above type composed with a central automorphism. Hence the order of the automorphism group is $(p-1)^2 |\mathrm{Aut}_c(G)|$.

In [29, Proposition 2.3] Jamali proves that for $p = 2$, all automorphisms of G are central. We just proved that for $p \neq 2$ there is a noncentral automorphism, take $k_0 > 1$ above, hence we have the following theorem.

Theorem 2.9.8. *The group $G_n(m, p)$ is Miller if and only if $p = 2$.*

2.9.2 Description of the Central Automorphisms

Notice that since G is a PN group, hence there is a one-one correspondence between $\mathrm{Aut}_c(G)$ and $\mathrm{Hom}(G, Z(G))$. Since, $Z(G) = \langle a_2^p \rangle \times G'$. Hence $\mathrm{Hom}(G, Z(G)) = \mathrm{Hom}(G, \langle a_2^p \rangle) \times \mathrm{Hom}(G, G')$. It follows: $\mathrm{Aut}_c(G) = A \times B$ where

$$A = \{\sigma \in \mathrm{Aut}_c(G) : x^{-1}\sigma(x) \in \langle a_2^p \rangle\}$$

$$B = \{\sigma \in \mathrm{Aut}_c(G) : x^{-1}\sigma(x) \in G'\}$$

Elements of A can be explained in a very nice way. Pick a random integer k such that $k = lp + 1$ where $0 \leq l \leq p^{m-2}$ and a random subset R (could be empty) of $\{0, 3, 4, \ldots n\}$, and then an arbitrary automorphism in A is

$$\begin{gathered} \sigma(a_1) = a_1 \\ \sigma(a_2) = a_2^k \\ \sigma(a_i) = \begin{cases} a_i & \text{if } i \notin R \\ a_i \left(a_2^{p^{m-1}}\right)^{r_i} & \text{if } i \in R \end{cases} \end{gathered} \tag{2.3}$$

We use indexing in $\{0, 3, 4, \ldots, n\}$ to order R and $0 < r_i < p$ is an integer corresponding to $i \in R$. Conversely, any element in A can be described this way.

The automorphism $\phi \in B$ is of the form

$$\phi(x) = \begin{cases} a_1 & \text{if } \quad x = a_1 \\ a_i z & \text{if } \quad x = a_i \quad i \in \{0, 2, 3, \ldots, n\} \end{cases} \tag{2.4}$$

where $z \in G'$.

2.10 Using key-exchange protocol I

Let us briefly recall the key-exchange protocol described earlier. Alice and Bob decide on a group G and a non-central element $g \in G \setminus Z(G)$ over an insecure channel. Alice then chooses an arbitrary automorphism ϕ_A and sends Bob $\phi_A(g)$. Similarly Bob picks an arbitrary automorphism ϕ_B and sends Alice $\phi_B(g)$. Since the automorphisms commute, both of them can compute $\phi_A(\phi_B(g))$, which is their private key. The most devastating attack on the system is the one in which Oscar looking at g, $\phi_A(g)$ and $\phi_B(g)$ can predict with some degree of certainty what $\phi_A(\phi_B(g))$ will look like, i.e., a GDHP attack.

Definition 2.10.1 (Parity condition for elements in G). *Let $g = d_0^{\beta_0} d_1^{\beta_1} d_2^{\beta_2} d_3^{\beta_3} \ldots d_n^{\beta_n}$ be an arbitrary element of G, i.e. $0 \leq \beta_0 < p$, $0 \leq \beta_1 < p$, $0 \leq \beta_2 < p^m$ and $0 \leq \beta_i < p^2$ for $3 \leq i \leq n$. Then the vector $v := (\beta_0, \beta_3, \beta_4, \ldots, \beta_n)$ is called the parity of g. Two elements g and g' are said to be of same parity condition if $v = v' \mod p$, where v' is the parity of g'.*

Lemma 2.10.2. *Let $g \in G$ and $\phi : G \to G$ be any central automorphism then g and $\phi(g)$ have the same parity condition.*

Proof. Notice that an automorphism ϕ either belongs to A or B or is of the form $\phi(g) = g f_\phi(g) g_\phi(g)$ where $f_\phi \in \mathrm{Hom}(G, Z(G))$ and $g_\phi \in \mathrm{Hom}(G, G')$. So we might safely ignore elements from A, since they only affect the exponent of a_2. Also note that a_1 being in the commutator subgroup remains fixed under any central automorphism.
So we need to be concerned with elements of B. From the description of B, from the fact that each commutator is a word in p-powers of the generators and from the fact that $G' \subset Z(G)$, the lemma follows. •

Now let us understand what an element in A does to an element $g \in G$. We use notation from Equation 2.3.

Lemma 2.10.3. *Let $g = d_0^{\beta_0} d_1^{\beta_1} d_2^{\beta_2} d_3^{\beta_3} \ldots d_n^{\beta_n}$, $\phi \in A$ and if*
$\phi(g) = d_0^{\beta'_0} d_1^{\beta'_1} d_2^{\beta'_2} d_3^{\beta'_3} \ldots d_n^{\beta'_n}$ then $\beta_i = \beta'_i$ for $i \neq 2$ and
$\beta'_2 = k\beta_2 + p^{m-1} \sum_{i \in R} r_i \beta_i \mod p^m$ where $k = lp + 1$, $l \in [0, p^{m-2}]$.

Proof. Notice that from Equation 2.3, it is clear that elements of A only affect the exponent of a_2, so $\beta'_i = \beta_i$ for $i \neq 2$ follows trivially. From the definition of A and simple computation it follows that $\beta'_2 = k\beta_2 + p^{m-1} \sum_{i \in R} r_i \beta_i \mod p^m$. •

In the key exchange protocol I, we will only use automorphisms from[1] A. As noted earlier there are two kinds of attacks, GDLP (the discrete logarithm problem in automorphisms) and GDHP (Diffie-Hellman problem in automorphisms). We have earlier stated that GDLP is equivalent to finding the automorphism from the action of the automorphism on one element. It seems that for one to find the automorphism discussed in the previous lemma, one has to find k, R and r_i. Notice that $\beta'_2 = k\beta_2 + p^{m-1} \sum_{i \in R} r_i \beta_i \mod p^m$, is a knapsack in β_2 and p^{m-1}, but solving that knapsack is not enough to compute the image of any element, because R is not known so the β_i's are not known. We shall show in a moment that the security of the key exchange protocol depends on the difficulty of this knapsack, whose security is still an open question, but this doesn't help Oscar to find the automorphism, just partial information about the automorphism comes out.

[1]In light of Lemma 2.10.2, we believe that adding automorphisms from B is not going to add to the security of the system.

Next we show that though it seems to be secure under GDLP, if the knapsack is solved then the system is broken by GDHP. This proves that GDHP is a weaker problem than GDLP in $G_n(m, p)$. Let $g = a_0^{\beta_0} a_1^{\beta_1} a_2^{\beta_2} a_3^{\beta_3} \dots a_n^{\beta_n}$, then as discussed before for $\phi, \psi \in \mathrm{Aut}_c(G)$, there are $k_i \in \mathbb{N}$ for $i = 3, 4, \dots, n$:

$$\phi(g) = a_0^{\beta_0} a_1^{\beta_1} a_2^{k_2\beta_2 + p^{m-1} \sum_{i\in R} r_i\beta_i} a_3^{\beta_3 + k_3 p} \dots a_n^{\beta_n + k_n p} \tag{2.5}$$

$$\psi(g) = a_0^{\beta_0} a_1^{\beta_1} a_2^{k'_2\beta_2 + p^{m-1} \sum_{i\in R'} r'_i\beta_i} a_3^{\beta_3 + k'_3 p} \dots a_n^{\beta_n + k'_n p} \tag{2.6}$$

From direct computation it follows that the exponent of a_2 in $\phi(\psi(g))$ is

$$k_2 \left(k'_2\beta_2 + p^{m-1} \sum_{i\in R'} r'_i\beta_i \right) + p^{m-1} \sum_{i\in R} r_i\beta_i \tag{2.7}$$

The exponent of a_0, a_1 stays the same and the exponent of a_i will be $\beta_i + (k_i + k'_i)p \mod p^2$ for $3 \leq i \leq n$. As mentioned before since we are using only automorphisms from A, i.e., ϕ and ψ are in A, it follows that $k_i = k'_i = 0$ for $i = 3, 4, \dots, n$.

Notice that g, Equations 2.5 and 2.6 are public, so Oscar sees those. Since the exponents of $a_0, a_1, a_3, \dots, a_n$ are predictable, hence the key, Alice and Bob want to establish the exponent of a_2 in $\phi(\psi(g))$, which is given by Equation 2.7. Since Oscar sees Equations 2.5 and 2.6, if he can compute k_2 from $k_2\beta_2 + p^{m-1} \sum_{i\in R} r_i\beta_i \mod p^m$, then he can compute $p^{m-1} \sum_{i\in R} r_i\beta_i$ and the scheme is broken. But, $k_2 = lp + 1$ for some $l \in [0, p^{m-2}]$ hence

$$k_2\beta_2 + p^{m-1} \sum_{i\in R} r_i\beta_i \mod p^m$$

reduces to

$$\beta_2 + lp\beta_2 + p^{m-1} \sum_{i\in R} r_i\beta_i \mod p^m.$$

Since β_2 is public, Oscar can compute $lp\beta_2 + p^{m-1} \sum_{i\in R} r_i\beta_i \mod p^m$. Notice that finding k_2 is equivalent to finding l, hence one of the security assumptions is that there is no polynomial time algorithm to find l from

$$lp\beta_2 + p^{m-1} \sum_{i\in R} r_i\beta_i \mod p^m. \tag{2.8}$$

In one instance, if the parameters β_i, l, r_i and R are so chosen that $lp\beta_2 < p^{m-1}$ and $lp\beta_2 + p^{m-1} \sum_{i\in R} r_i\beta_i < p^m$, then one can divide the whole expression in Equation 2.7 by p^{m-1} and the remainder is $lp\beta_2$ from which l can be found, since p and β_2 are public. It seems that the best choice for each of l and β_i, $i = 1, 2, \dots, n$ is a power of p such that $lp\beta_2$ is greater than p^{m-1}.

2.11 Using key exchange protocol II

We briefly recall the key exchange protocol II. In this protocol Alice picks a random noncentral element $g \in G$ and $\phi_A \in A$ and sends Bob $\phi_A(g)$. Bob selects randomly $\phi_B \in A$ and sends Alice $\phi_B(\phi_A(g))$. Alice then computes $\phi_B(g)$ by computing $\phi_A^{-1}(\phi_B(\phi_A(g)))$ and picks another random automorphism $\phi_H \in A$ and computes $\phi_H(\phi_B(g))$ and sends it to Bob. Bob computes $\phi_B^{-1}(\phi_H(\phi_B(g))) = \phi_H(g)$ which is their private key. Notice that Oscar never sees g.

Using notation from Equation 2.3 for the automorphisms in A, we see that with the exchange of $\phi_A(g)$, $k_2\beta_2 + p^{m-1} \sum_{i\in R} r_i\beta_i \mod p^m$ is revealed. Then with the exchange of $\phi_B(\phi_A(g))$,

$$k'_2\left(k_2\beta_2 + p^{m-1}\sum_{i\in R} r_i\beta_i\right) + p^{m-1}\sum_{i\in R'} r'_i\beta_i = k'_2k_2\beta_2 + p^{m-1}\sum_{i\in R} r_i\beta_i + p^{m-1}\sum_{i\in R'} r'_i\beta_i$$

is revealed. When Alice computes $\phi_B(g)$ she computes $k'_2\beta_2 + \sum_{i\in R'} r'_i\beta_i$. With the exchange of $\phi_H(\phi_B(g))$ Oscar sees

$$k''_2\left(k'_2\beta_2 + p^{m-1}\sum_{i\in R'} r'_i\beta_i\right) + p^{m-1}\sum_{i\in R''} r''_i\beta_i = k''_2k'_2\beta_2 + p^{m-1}\sum_{i\in R'} r'_i\beta_i + p^{m-1}\sum_{i\in R''} r''_i\beta_i$$

and the key is

$$k''_2\beta_2 + p^{m-1}\sum_{i\in R''} r''_i\beta_i.$$

All the above operations are done $\mod p^m$. Since in the above key exchange protocol, g is never revealed in public, so β_2 is not a public information, hence the knapsack attack similar to key exchange protocol I is not possible.

2.12 Conclusion

In this chapter we studied a key exchange protocol using commuting automorphisms in a non-abelian p-group. Since any nilpotent group is a direct product of its Sylow subgroups, so for our work nilpotent groups can be reduced to p-groups. We argued that this is a generalization of the Diffie-Hellman key exchange and hence a generalization of the discrete log problem. Other public key systems like the El-Gamal cryptosystem using discrete logarithm might be adaptable to our methods. This is the first attempt to generalize discrete logarithm in the way we did. So there are more questions than there are answers.

We should try to find other groups and try our system in terms of GDLP and GDHP. As we noted earlier, GDHP is a subproblem of the GDLP, and we saw in $G_n(m, p)$, GDHP is a much easier problem than GDLP. Our example was of the form $d > b$ in Theorem 2.7.12. The next step is to look at groups where $d = b$. We note from theorem 2.7.13, if a p-group G is a PN group then $\mathrm{Aut}_c(G)$ is a p-group and since p-groups have nontrivial centers, one can work in that center with our scheme. In this case we would be generalizing to arbitrary nilpotentcy class but keep working with central automorphisms.

Lastly we note that, if we were using some representation for this finitely presented group G, say for example, matrix representation of the group over a finite field $\mathbb{F}_q$, then the security of the system in $G_n(m, p)$ becomes the discrete logarithm problem [46, 47]. Since the discrete logarithm problem in matrices is only as secure as the discrete logarithm problem in finite fields there is no known advantage to go for matrix representation, but there might be other representations of interest.
There is one conjecture that comes out of this work and we end with that.

Conjecture 2.12.1. *Let G be a Miller p-group for odd prime p, then G is special.*

Bibliography

[1] L. Adleman, *A subexponential algorithm for the discrete logarithm problem with application to cryptography*, Proceedings IEEE 20th annual symposium on foundations of computer science, 1979, pp. 55–60.

[2] A.E. Adney and Ti Yen, *Automorphisms of p-group*, Illinois Journal of Mathematics **9** (1965), 137–143.

[3] I. Anshel, M. Anshel, B. Fisher, and D. Goldfield, *New key agreement protocols in braid group cryptography*, CT-RSA 2001, Lecture Notes in Computer Science, no. 2020, Springer, 2001, pp. 1–15.

[4] I. Anshel, M. Anshel, and D. Goldfeld, *An algebraic method for public-key cryptography*, Math. Research Letters **6** (1999), 287–291.

[5] Joan S. Birman, *Braids, links and mapping glass groups*, Annals of Mathematics Studies, no. 82, Princeton University Press, 1974.

[6] I. F. Blake, R. Fuji-Hara, R. C. Mullin, and S. A. Vanstone, *Computing logarithms in finite fields of characteristic two*, SIAM Journal on Matrix Analysis and Applications **5** (1984), no. 2, 276–285.

[7] I. F. Blake, R. C. Mullin, and S. A. Vanstone, *Computing logarithms in* GF(2^n), Advances in cryptology (Santa Barbara, Calif., 1984), Lecture Notes in Computer Science, no. 196, Springer, Berlin, 1985, pp. 73–82.

[8] Ian Blake, Gadiel Seroussi, and Nigel Smart, *Eliptic curves in cryptography*, London Mathematical Society, Lecture Note Series, no. 265, Cambridge University Press, 1999.

[9] Ian F. Blake and Theo Garefalakis, *On the complexity of the Discrete Logarithm and Diffie-Hellman problems*, Journal of Complexity **20** (2004), 148–170.

[10] Manuel Blum and Silvio Micali, *How to generate cryptographically strong sequence of pseudo random bits?*, Siam Journal of Computing **13** (1984), no. 4, 850–864.

[11] Dan Boneh, *The Decision Diffie-Hellman problem*, Algorithmic number theory (Portland, OR, 1998), Lecture Notes in Computer Science, no. 1423, Springer, Berlin, 1998, pp. 48–63.

[12] Dan Boneh and Richard Lipton, *Searching for elements in black box fields and applications*, Crypto '96, Lecture notes in Computer Science, vol. 1109, Springer-Verlag, 1996, pp. 283–297.

[13] Dan Boneh and Ramaratham Venkatesan, *Hardness of computing the most significant bits of secret keys in diffie-hellman and related schemes*, Crypto '96, Lecture notes in Computer Science, vol. 1109, Springer-Verlag, 1996, pp. 129–142.

[14] Jung Hee Cheon and Byungheup Jun, *A polynomial time algoritm for the braid Diffie-Hellman conjugacy problem*, Advances in cryptography – CRYPTO 2003, Lecture Notes in Computer Science, no. 2729, Springer, Berlin, 2003, pp. 212–225.

[15] Don Coppersmith, *Fast evaluation of logarithms in fields of characteristic two*, Transactions on Information Theory **30** (1984), no. 4, 587–594.

[16] Don Coppersmith, Andrew M. Odlzyko, and Richard Schroeppel, *Discrete logarithms in* GF(p), Algorithmica **1** (1986), no. 1, 1–15.

[17] Patrick Dehornoy, *Braid-based cryptogrpahy*, Contemporary Mathematics **360** (2004), 1–33.

[18] Whitfield Diffie and Martin Hellman, *New directions in cryptography*, Institute of Electrical and Electronics Engineers., vol. IT-22, Transactions on Information Theory, no. 6, 1976, pp. 644–654.

[19] Bruce E. Earnley, *On finite groups whose group of automorphisms is abelian*, Ph.D. thesis, Wayne State University, 1975.

[20] Taher Elgamal, *A public key cryptosystem and a signature scheme based on discrete logarithms.*, Lecture Notes in Comput. Sci., 196, Springer, Berlin. (1985), 10–18.

[21] T.A. Fournelle, *Elementary abelian p-groups as automorphism group of infinite group*, I. Math. Z. **167** (1979), 259–270.

[22] Steven Galbraith and Victor Rotger, *Easy decision Diffie-Hellman groups*, LMS Journal of Computation and Mathematics **7** (2004), 201–218.

[23] The GAP Group, *GAP – Groups, Algorithms, and Programming, Version 4.3*, 2002, (`http://www.gap-system.org`).

[24] Theodoulos Garefalakis and Daniel Panario, *The index calculus method using non-smooth polynomials*, Mathematics of Computation **70** (2001), no. 235, 1253–1264.

[25] M. Hall and J.K. Senior, *The groups of order* 2^n $(n \leq 6)$, Macmillan, 1964.

[26] P. Hall, *The Edmonton notes on nilpotent groups*, Queen Mary college mathematics notes, Cambridge, 1969.

[27] Hermann Heineken and Hans Liebeck, *The occurrence of finite groups in the automorphism group of nilpotent groups of class* 2, Archives of Mathematics **25** (1974), 8–16.

[28] Charles Hopkins, *Non-abelian groups whose groups of isomorphism are abelian*, Ann. of Math **29** (1927), no. 1-4, 508–520.

[29] Ali-Reza Jamali, *Some new non-abelian 2-groups with abelian automorphism groups*, Journal of Group Theory **5** (2002), 53–57.

[30] D. Jonah and M. Konvisser, *Some non-abelian p-groups with abelian automorphism groups*, Archives of Mathematics **26** (1975), 131–133.

[31] Irving Kaplansky, *Infinite abelian groups*, The University of Michigan Press, 1969.

[32] E.I. Khukhro, *p-automorphisms of finite p-groups*, London Mathematical Society, Lecture Note Series, no. 246, Cambridge University Press, 1997.

[33] Ki Hyoung Ko, Doo Ho Choi, Mi Sung Cho, and Jang Won Lee, *New signature scheme using conjugacy problem*, http://eprint.iacr.org/2002/168, 2002.

[34] Ki Hyoung Ko, Sang Jin Lee, Jung Hee Cheon, Jae Woo Han, Ju sung Kang, and Choonsik Park, *New public-key cryptosystem using braid groups*, Advances in Cryptology – CRYPTO 2000 (Mihir Bellare, ed.), Lecture Notes in Computer Science, no. 1880, 2000, pp. 166–183.

[35] Neal Koblitz, *A course in number theory and cryptography*, second ed., Graduate Texts in Mathematics, no. 114, Springer-Verlag, New York, 1994.

[36] ______, *Algebraic aspects of cryptography*, Algorithms and Computation in Mathematics, no. 3, Springer-Verlag, Berlin, 1998.

[37] A.G. Kurosh, *The theory of groups*, vol. 1 & 2, Chelsea Publishing Company, 1960.

[38] B. A. LaMacchia and A. M. Odlyzko, *Computation of discrete logarithms in prime fields*, Design Codes and Cryptogrpahy **1** (1991), 46–62.

[39] Ueli Maurer and Stefan Wolf, *On the complexity of breaking the Deffie-Hellman protocol*, Advances in Cryptology - CRYPTO '96, Lecture Notes in Computer Science, vol. 1109, Springer-Verlag, 1996, pp. 268–282.

[40] ______, *Secret-key agreement over unauthenticated public channels – part I: Definitions and a completeness result*, IEEE Transactions on information thoery **49** (2003), no. 4, 822–831.

[41] ______, *Secret-key agreement over unauthenticated public channels – part II: The simulatability condition*, IEEE Transactions on information thoery **49** (2003), no. 4, 832–838.

[42] ______, *Secret-key agreement over unauthenticated public channels – part III: Privacy amplification*, IEEE Transactions on information thoery **49** (2003), no. 4, 839–851.

[43] Kevin McCurley, *A key distribution system equivalent to factoring*, Journal of cryptology **1** (1988), 95–105.

[44] ______, *The discrete logarithm problem*, Proceedings of synopsis in applied mathematics **42** (1990), 49–74.

[45] Alfred J. Menezes (ed.), *Applications of finite fields*, Kluwer Academic Publishers, 1993.

[46] Alfred J. Menezes and Scott A. Vanstone, *A note on cyclic group, finite fields and discrete logarithm problem*, Applicable Algebra in Engineering, Communication and Computing **3** (1992), no. 1, 67–74.

[47] Alfred J. Menezes and Yi-Hong Wu, *The discrete logarithm problem in* $\mathrm{GL}(n, q)$, Ars Combinatoria **47** (1997), 23–32.

[48] G.A. Miller, *A non-abelian group whose group of isomorphism is abelian*, Messenger Math. **43** (1913), 124–125.

[49] M.J.Curran, *Semidirect product groups with abelian automorphism groups*, J. Austral. Math, Soc. **Series A** (1987), no. 42, 84–91.

[50] Martha Morigi, *On p-groups with abelian automorphism group*, The Mathematical Journal of the University of Padova **92** (1994), 47–58.

[51] Ronald C. Mullin and Ayan Mahalanobis, *An alternative representation of finite fields*, preprint.

[52] ______, *Dickson bases and finite fields*, Tech. Report CORR 2005-04, University of Waterloo, 2005.

[53] Andrew Odlyzko, *Discrete logarithms in finite fields and their cryptographic significance*, Advances in Cryptology: Proceedings of EUROCRYPT 84 (T. Beth, Cot N., and I. Ingemarsson, eds.), Lecture notes in computer science, no. 209, Springer-Verlag, 1985, pp. 224–314.

[54] Andrew Odlyzko, *Discrete logarithms: The past and the future*, Designs Codes and Cryptography **19** (2000), 129–145.

[55] Carl Pomerance (ed.), *Cryptology and computational number theory*, Proceedings of symposia in Aplied Mathematics, vol. 42, American Mathematical Society, 1990.

[56] Joseph J. Rotman, *An introduction to the theory of groups*, Springer-Verlag, 1994.

[57] P. R. Sanders, *The central automorphism of a finite group*, J. London Math. Soc. **44** (1969), 225–228.

[58] Oliver Schirokauer, Damian Weber, and Thomas Denny, *Discrete logarithms: the effectiveness of the index calculus method*, Algorithmic number theory (Talence, 1996), Lecture Notes in Computer Science, no. 1122, Springer, Berlin, 1996, pp. 337–361.

[59] W.R. Scott, *Group theory*, Dover, 1964.

[60] Victor Shoup, *Lower bounds for discrete logarithm and related problems*, EUROCRYPT '97, Lecture Notes in Computer Science, vol. 1233, Springer, 1997, pp. 256–266.

[61] Igor E. Shparlinski, *Security of polynomial transformations of Diffie-Hellman key*, Finite fields and their applications **10** (2004), 123–131.

[62] Charles Sims, *Computation with finitely presented groups*, Cambridge University Press, Cambridge, 1994.

[63] Douglas Stinson, *Cryptography: Theory and practice*, 2 ed., CRC Press, 2002.

[64] Ruth R. Struik, *Some non-abelian 2-groups with abelian automorphism groups*, Archives of Mathematics **39** (1982), 299–302.

[65] Edlyn Teske, *Square root algorithms for discrete logarithm problem (a survey)*, Public-Key Cryptography and Computational Number Theory, Walter de Gruyter, Berlin - New York, 2001, pp. 283–301.

[66] ______, *Computing discrete logarithm with the parallelized kangaroo method*, Discrete Applied Mathematics **130** (2003), 61–82.

[67] Maria Isabel González Vasco and Igor E. Sharlinski, *On the security of Diffie-Hellman bits*, Cryptography and computational number theory, Progress in Computer Science and Applied Logic, Birkhäuser, Basel, 2001, pp. 257–268.

[68] Maria S. Voloshina, *On the holomorph of a discrete group*, Ph.D. thesis, University of Rochester, 2003.

[69] Wandi Wei, Tran van Trung, Spyros Magliveras, and Frederick Hoffman, *Cryptographic primitive based on groups of hidden order*, Tatra Mountains Mathematical Publications **29** (2004), 147–155.

[70] H. Zassenhaus, *The theory of groups*, Chelsea, New York, 1958.