Tightest Lower Bound for the Elliptic Curve Diffie-Hellman Problem and New Attacks on the Discrete Logarithm Problem

A thesis

submitted in partial fulfillment of the requirements

of the degree of

Doctor of Philosophy

by

Prabhat Kumar Kushwaha

ID: 20123167

INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH PUNE

February, 2017

Dedicated to
My Parents

Abstract

The elliptic curve discrete logarithm problem(ECDLP) is one of the most widely used primitives in various public key cryptosystems. Hardness of ECDLP is an absolute security necessity, but not sufficient, for these cryptosystems and the actual security depends on the elliptic curve Diffie-Hellman problem(ECDHP). Hence, it is imperative to study hardness of ECDLP as well as of ECDHP on the elliptic curve parameters recommended for practical implementations. Our work contributes in both the directions. We have given the **tightest** lower bound for ECDHP on the elliptic curve parameters most widely used in practical applications. These lower bounds ensure the security of all those protocols which rely on ECDHP for their security. We also present a novel *generic* algorithm which uses the multiplicative group of a finite field as *auxiliary group* probably for the *first* time. Our algorithm also indicates some security issues in NIST curves which are used for USA federal government for extremely secure communications.

Certificate

Certified that the work incorporated in the thesis entitled "*Tightest Lower Bound for the Elliptic Curve Diffie-Hellman Problem and New Attacks on the Discrete Logarithm Problem*", submitted by *Prabhat Kumar Kushwaha* was carried out by the candidate, under my supervision. The work presented here or any part of it has not been included in any other thesis submitted previously for the award of any degree or diploma from any other university or institution.

Date: February 27, 2017

Dr. Ayan Mahalanobis
Thesis Supervisor

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that violation of the above will be cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: February 27, 2017 *Prabhat Kumar Kushwaha*

Roll Number: 20123167

Acknowledgements

To my parents, Kamalesh Kushwaha and Archana Kushwaha: you never stopped me from pursuing things I was interested in and never forced your decisions on me, academically or otherwise. I always enjoyed your unconditional support. Whatever I have achieved in my life so far, I owe it all to you. A big thank you!

To my late grandmother, Rajmani Devi: I will always cherish our long-lasting chats which had a big impact on me. You have been, and still are, one of my guiding angels.

To my younger brother Prabodh and elder sister Shweta: you two have been my source of primary support system, emotional and moral. Lucky to have siblings like you by my side.

To my teachers, especially Prof. A. K. Srivastav and Dr. Shantanu Bhattacharya: the reason I chose a career in higher studies is you two. The faith you had in my capabilities encouraged and convinced me to follow that path. Indebted for life!

To all my friends, especially Ravi Maurya, Ashutosh Singh, Shivendra Yadav, Abhishek Juyal, Mayank Srivastav, Santosh Maurya, Akanksha Sharma and Anant Srivastava: Each one of you have made vital contribution at some point of time in my life, knowingly or unknowingly. Thanks for all the support and help, and making sure that I am on the right track. Special mention to Anurag Arya: you are like a brother to me. Perhaps the most honest, stoic

and righteous person I have seen so far. You effortlessly brought so many positive and permanent changes in me. An absolute fun and real treat to be with, cannot possibly ask for more!

To my colleagues at IISER Pune: the last five years of my life would not have been so much memorable and eventful if not for all of you nice people at Department of Mathematics, especially Pralhad Shinde, Manidipa Pal, Sushil Bhunia. Also, thanks a ton to all my teammates for those unforgettable football memories. You guys will be missed quite a bit.

I am also thankful to the following institute staff: Suvarna Bhardwaj and Tushar Kurulkar for their unfailing support and assistance.

A special gratitude goes out to University Grant Commission(UGC) for providing the funding for my research work.

And finally, last but by no means least, my supervisor Dr. Ayan Mahalanobis: I have learned a lot from you, and not just academically. The journey would not have been so much productive and satisfactory without you. Your positive energy and conviction in me kept me pushing and made me strive for better results. Thanks a lot for your unwaivering support, excellent guidance, and for being there whenever I needed.

Thanks for all your encouragement!

Prabhat Kumar Kushwaha

Contents

Chapter 1

Introduction

1.1 Cryptography

Word cryptography comes from the Greek root words *kryptos*, meaning hidden, and *graphikos*, meaning writing. So, cryptography was referred to as the methodology of concealing the content of messages before the computer age which started around mid-1970s. However, in the present digital age, this is just one of the goals of modern cryptography. Objectives of modern cryptography can be broadly divided into following four:

1. *Confidentiality*: This is to keep the content of data secret from all but those authorized. Encryption schemes are the cryptographic tools that are used for confidentiality(secrecy).

2. *Data integrity*: This is to address the unauthorized alteration of data and *hash functions* are used to achieve this.

3. *Authentication*: This is related to authenticate the source of the message as well as the message itself. Digital signatures ensure the integrity(authenticity) of the source and the message.

4. *Non-repudiation*: It prevents a party from denying previous commit-

ments or actions. Hash functions along with digital signatures enable us to deal with such situations.

Therefore, the overall goal of modern cryptography can be described as adequately addressing these four areas in both theory and practice.

Given the astronomical scale of web communications happening per second all over the world, it would not be wrong to say that we are living in an 'online' world where everyone is connected with one another via an insecure channel of communication. From secure exchange of personal messages to extremely sensitive communications between the governments of two countries, from online grocery purchase to safety of financial transactions worth thousands of billions of dollars, from our basic cellphone's password to the password designed to launch a nuclear weapon etc., are just some of the examples where cryptography is at the very core of their existence. In fact, the online world we are living in at present would not have been possible without the invention of cryptography. Therefore, the role of modern cryptography in maintaining peace, safety and order of present online world can not possibly be overstated.

1.2 Private Key Cryptography

Private key cryptography and public key cryptography are the two branches of cryptography. Until the mid-1970s, there was simply private key cryptography and it dates back to even the time of Roman emperor Julius Caesar. There is some evidence that Caesar employed a simple shift cipher to hide a secret message from enemy army. Shift cipher, substitution ciphers are simple examples of private key cryptosystems. Suppose Alice and Bob want to exchange encrypted messages. In this type of cryptosystems, the sender and the receiver of the message **must** agree on a secret key before they start

communicating. Using that shared secret key, any of the two parties can encrypt and other party can easily decrypt with the knowledge of the shared secret key. Thus, both parties have equal(or symmetric) powers and abilities. That is why private key cryptography is also called *symmetric key cryptography*. To give an idea on how it works, we take the example of a simple substitution cipher:

- The *secret key* k is a permutation f of the English alphabet $\{a, b, c, ..., z\}$ which is known to both Alice and Bob.

- *Encryption*: Using the encryption function $e_k = f$, the sender can encrypt a message m by computing $e_k(m) = f(m) = c$. c is the encrypted message or ciphertext.

- *Decryption*: To decrypt the ciphertext c, the receiver would use the decrypting function $d_k = f^{-1}$(easily computed from f) to extract the original message m as follows: $d_k(c) = f^{-1}(c) = f^{-1}(f(m)) = m$.

Modern symmetric key cryptosystems consist of block ciphers(more popular) and stream ciphers. Popular block ciphers are Data Encryption Standard (DES), triple-DES and Advanced Encryption Standard (AES). AES is the current world standard for symmetric key cryptography. At present, the most common implementations of AES encrypt and decrypt blocks of 128-bits, using keys also of 128-bits. A stream cipher, in its simplest form, requires that Alice and Bob agree on a pseudo-random bit generator and a seed. The seed acts as the secret key. A currently popular stream cipher is RC4[2].

1.3 Public Key Cryptography

Even though modern symmetric cryptosystems are safe and fast, there is this inherent problem that Alice and Bob need to agree on the secret key before they start communicating. What if they never met before or never had a chance to communicate over a secure channel to agree on the secret key? These are actual situations that occur all the time in the modern digital communications. For example, think about the practical situation where you want to encrypt your credit card details for a web purchase from a distant vendor. We can easily think of a number of similar circumstances in our daily lives. So, it might seem that symmetric key cryptography is of no use here and we are stuck. Fortunately, this problem was solved in the mid-1970s using public key cryptography by Whitfield Diffie and Martin Hellman[17]. Their paper, entitled "New Directions in Cryptography", formulated the concept of public key cryptosystems and made a number of groundbreaking contributions to this new area. However, it came to the light later that the concept of public key cryptosystems was already discovered by James Ellis in 1969 while working at the British Government Communications Headquarters (GCHQ) but his discoveries remained classified until his death in 1997. There is one more interesting thing about this story. It is now known that Malcolm Williamson, a fellow researcher at GCHQ, discovered Diffie-Hellman key exchange before it was rediscovered by Diffie and Hellman while another fellow researcher, Clifford Cocks, invented the RSA public key cryptosystem before Rivest, Shamir and Adleman did.

The paper[17] of Diffie and Hellman was revolutionary and it aptly begins with the following sentence:

We stand today on the brink of a revolution in cryptography.

Indeed, their paper formulated the basic definitions and goals of a new field, public key cryptography, of mathematics and computer science at a

time when the existence of this new field was dependent on then growing age of the digital computer.

This problem of not having a shared secret key but still be able to communicate can be solved by public key cryptography in two ways:

- Use public key exchange protocol to first get a shared secret key and then use symmetric key encryption.
- Directly use public key encryption scheme to communicate which does not require knowledge of a shared key beforehand.

In public key cryptosystems, each party has their own private key k_{priv} which they use to create a public key k_{pub} in polynomial time. However, there is no known polynomial time algorithm to get the private key from the public key. In fact, one need to solve some computationally hard number-theoretic problem to get the private key from the public key. Two computationally hard problems most widely used in public key cryptography are factorization of large integers(used in RSA) and discrete logarithm problem (DLP) in the multiplicative group of a finite field or the group of points on an elliptic curve over a finite field. We now describe the Diffie-Hellman key exchange protocol[30, Section 2.3] which is based on DLP.

Let $\mathbb{G}$ be a cyclic additive group generated by P and its order is n. Given $Q \in \mathbb{G}$, the **discrete logarithm problem (DLP)** in $\mathbb{G}$ is to compute the integer $x \pmod n$ such that $Q = xP$. The integer x is called the *discrete logarithm of Q with respect to P*. A very closely related problem to DLP is the **Diffie-Hellman problem (DHP)** in $\mathbb{G}$ which is to compute the element xyP from given elements xP and yP. Assume that DLP is hard in the group $\mathbb{G}$. Suppose Alice and Bob want to share a secret key to be used in a symmetric encryption scheme, but their only means of communication is insecure. Using the intractability of **DLP** in the group $\mathbb{G}$, they can solve this problem as follows:

- Alice picks a secret integer a and publishes $Q = aP$. Here, a is her private key and $Q = aP$ is her public key.
- Similarly, Bob chooses a secret integer b and makes $R = bP$ public. Thus, b is the private key of Bob and $R = bP$ is his public key.
- Alice uses her private key a and computes $aR = abP$.
- Bob uses his private key b and computes $bQ = abP$.
- Thus, $S = abP$ is the *shared secret key* between Alice and Bob.

Note that how hardness of DLP in $\mathbb{G}$ makes sure that no one else except Alice and Bob can possibly have access to a and b respectively, and thus provides a way for them to get the secret key. Moreover, there are public key encryption schemes such as ElGamal public key encryption[19] which can be used directly for encryption, without requiring a shared secret key. So, we saw how public key cryptography solved the problem with symmetric key cryptography, and hence led to the gigantic scale of deployment of modern cryptography in current digitally connected world. If not for public key cryptography, we would never have been able to reap the extremely rich rewards modern cryptography is bestowing upon us in a number of ways. In simple terms, whenever we provide a secret piece of information, something which is known only to you such as password/ATM pin etc., to be able to start the various web communications done on a daily basis, this should remind us that public key cryptography is playing its role there. This is enough to signify the massive and indispensable role of public key cryptography in modern cryptography for secure and astronomical web communications performed on a daily basis all around the globe. Perhaps public key cryptography has transformed the world in a way that has never happened before and it is fair to say it is at the root cause of the present online world we are living in.

1.4 Our contribution

As mentioned above, factorization, used in RSA, and DLP are the two major primitives used in practical public key cryptography. However, factorization and DLP in the multiplicative group of a finite field can be solved in sub-exponential time whereas there are no known attacks that solves DLP in the group of points on elliptic curves over finite fields faster than exponential time, barring some special and a very small class of elliptic curves which can easily be avoided for practical implementations. Thus, DLP in elliptic curve over finite field require far less input size than RSA or than DLP in the multiplicative group of a finite field to provide same level of security assurance. This reduces the cost of implementations significantly, explaining the widespread employment of elliptic curves in practice.

Hardness of elliptic curve discrete logarithm problem (ECDLP) is a security necessity but it might not be sufficient to assure the security of protocols based on ECDLP. For example, one can easily check that Diffie-Hellman key exchange protocol described above, as well as ECDSA[33] and many more protocols are the examples where an attacker just needs to solve elliptic curve Diffie-Hellman problem (ECDHP), even if ECDLP is hard. Hence, it is imperative to study hardness of ECDLP as well as of ECDHP on the elliptic curve parameters recommended for practical implementations. Our work contributes in both the directions:

1. We present the **first** reduction of DLP to DHP that *uses* $\mathbb{F}_p^\times$ *as the auxiliary group* to estimate the lower bound on DHP. Moreover, using our reduction algorithm, we have given the **tightest** lower bound for the ECDHP on the curves in *SEC2 standard*[52]. These lower bounds ensure the security of all those protocols, like ECDSA[33], Diffie-Hellman key exchange protocol etc, which rely on ECDHP for their security. To give an idea about the significance of this result, we mention that

SEC2 standard[52] of elliptic curve parameters is recommended by *Standard of Efficient Cryptography Group (SECG)* at Certicom Corporation and the SEC2 standard also include *NIST curves*[48](used by USA federal government for extremely secret communications) as well as most widely used curves from ANSI[1]. Also, digital signature protocol ECDSA in bitcoin is done using one of the curves in SEC2 standard, viz SECP256K1[7].

2. We also present a novel *generic* algorithm to solve DLP, again *using* $\mathbb{F}_p^\times$ *as the auxiliary group*. A remarkable revelation coming out of our new attack is that it indicates some security lapses in ECDLP on NIST curves. We are well aware that hardness of ECDLP is an absolute necessity for the security of protocols like ECDSA[33], Diffie-Hellman key exchange[30, Section 2.3] etc. Therefore, any new attack indicating any kind of weakness in ECDLP would have immense cryptographic importance and this is exactly where this work contributes.

1.5 Overview

We give an overview of the chapters of this thesis. Chapter 2 describes ECDLP and known attacks which are relevant to our work. In chapter 3, we present our work in achieving the tightest lower bound on ECDHP for the important elliptic curves in SEC2 standard. Our new attack on DLP, indicating some weakness in NIST curves is presented in chapter 4. Appendices and references are provided at the end.

Going over our work described in these chapters, one can say that our work has contributed towards the development of elliptic curve public key cryptography.

Chapter 2

Elliptic Curve Discrete Logarithm Problem and Attacks

2.1 Introduction

Discrete logarithm problem and factorization of integers are two most widely used primitives in the public key cryptography as discussed in chapter 1. However, in this chapter, we will only be interested in the discrete logarithm problem and various attacks to solve the problem. Let $\mathbb{G}$ be a cyclic additive group generated by P and its order is n. Given $Q \in \mathbb{G}$, the **discrete logarithm problem (DLP)** in $\mathbb{G}$ is to compute the integer $x \pmod{n}$ such that $Q = xP$. The integer x is called the *discrete logarithm of Q with respect to P*. DLP in certain groups is a computationally hard number theoretic problem, and therefore it is used as building blocks for various public key protocols that we come across relentlessly in our daily web communications. Hardness of DLP assures the desired security of such protocols, signifying the indispensable role of DLP towards the security of modern web communications.

We already described that in detail in chapter 1.

Having said that, we also want to point out that hardness of DLP depends on the group structure and DLP is not hard in all groups. For example, DLP in $< \mathbb{Z}_n, + >$ is very easy because solving DLP in $< \mathbb{Z}_n, + >$ boils down to finding inverse module n which can be done efficiently using extended Euclid algorithm[30, Theorem 1.11]. Of course, only those groups where DLP is hard are used in public key cryptography. The multiplicative group of a finite field $\mathbb{F}_q^\times$ and the group of points on an elliptic curve $\mathbb{E}(\mathbb{F}_q)$ over a finite field $\mathbb{F}_q$ are two main candidates of such groups for DLP-based protocols. The DLP on elliptic curves is also known as *elliptic curve discrete logarithm problem (ECDLP)*. Now, there exists index calculus attack, a non-generic attack, which solves DLP in $\mathbb{F}_q^\times$ and has sub-exponential complexity. On the other hand, there are only generic attacks known, except some special classes of elliptic curves which we describe briefly in section 2.2, to solve ECDLP on $\mathbb{E}(\mathbb{F}_q)$ and these generic attacks have exponential time complexity, much slower than sub-exponential. As a result, $\mathbb{E}(\mathbb{F}_q)$ requires far less input size than $\mathbb{F}_q^\times$ to provide same level of security and thus, reduces the cost of implementing the protocols. Therefore, $\mathbb{E}(\mathbb{F}_q)$ is extensively used in practice for DLP-based protocols and that is why the attacks on ECDLP are studied extensively.

In this chapter, we discuss baby-step giant-step (BSGS) algorithm and its improved versions as our work presents a new BSGS-type algorithm. The overview of this chapter is as follows: basic notations of $\mathbb{E}(\mathbb{F}_q)$ and special attacks known for $\mathbb{E}(\mathbb{F}_q)$ are discussed in section 2. Section 3 deals with basic BSGS algorithm. In section 4, Pollard's 'interleaving' BSGS is given. In section 5, we discuss the improvement made in BSGS algorithm using negation map of $\mathbb{E}(\mathbb{F}_q)$. Section 6 discusses the best theoretical model of BSGS-type algorithm, and also the efforts towards such algorithm and some

open questions.

2.2 Basic Notations on Elliptic Curves and Special Attacks

Let $\mathbb{F}_q$ be a finite field with q elements(q is some prime-power). An elliptic curve over $\mathbb{F}_q$(char$\neq 2, 3$) in *Weierstrass model* is given by the solution of the equation

$$\mathbb{E} : Y^2 = X^3 + AX + B$$

where $A, B \in \mathbb{F}_q$ such that $4A^3 + 27B^2 \neq 0$. This condition makes sure that the curve is *non-singular*. We denote it by $\mathbb{E}(\mathbb{F}_q)$. If $P = (x_0, y_0)$ is a point on $\mathbb{E}(\mathbb{F}_q)$, its inverse is given by $-P = (x_0, -y_0)$. Note that P and $-P$ both have same X-coordinate but their Y-coordinate are negative of each other. In section 5, we will see how this property of *negation map* on elliptic curve can be used to make faster attacks on ECDLP. There are other models of elliptic curves such as Montgomery model [46], twisted Edwards model[4]. These models are useful for efficient implementation but from the point of view of the ECDLP, various models do not play such a prominent role because we can usually switch between them whenever needed. Moreover, we do not worry too much about elliptic curve models because we are mainly interested in generic attacks on ECDLP and these generic attacks work just the same on any model.

For a point $P \in \mathbb{E}(\mathbb{F}_q)$ and a non-negative integer k we define kP to be $P + P + \cdots\cdots\cdots + P(k$ times). One can extend this definition to all $k \in \mathbb{Z}$ using $kP = (-k)(-P)$. The **ECDLP** is: Given $P, Q \in \mathbb{E}(\mathbb{F}_q)$, find an integer x, if it exists, such that $Q = xP$. As mentioned above, best attacks to solve ECDLP are, in general, generic attacks such as BSGS and Pollard's rho attack[49]. However, there are some special instances of elliptic curves

where far more efficient attacks are known on ECDLP. We list some of the important ones below.

- Elliptic curves $\mathbb{E}(\mathbb{F}_p)$ where p is a prime and $|\mathbb{E}(\mathbb{F}_p)| = p$ are called anomalous curves. Anomalous curves are an extremely special case and ECDLP on such curves can be very easily solved in linear time using p-adic logarithm map [15].

- When elliptic curve is not anomalous, the Weil and Tate-Lichtenbaum pairings transform the ECDLP in $\mathbb{E}(\mathbb{F}_q)$ into an instance of discrete logarithm problem in the multiplicative group of a finite field $\mathbb{F}_{q^k}$(MOV attack)[15, 44] where k is the *embedding degree* of the elliptic curve. For certain special elliptic curves such as *supersingular curves*, the finite field $\mathbb{F}_{q^k}$ is small enough, therefore resulting instance of DLP in $\mathbb{F}_{q^k}$ can be solved using index calculus algorithm. However, this attack on ECDLP may not be practical in general because it is applicable to only very special class of curves.

- The Xedni calculus algorithm proposed by Silverman is a variant of index calculus attack and it is based on lifting elliptic curves from finite fields to number fields. However, this approach is also not likely to work, see [32].

- There have been many other ideas to give an attack on ECDLP, viz. [23, 26] by Galbraith *et al.*, [20, 21] by Frey, [31] by Huang and Raskind. All these approaches to solve ECDLP bring to light deep connections within number theory, but at present none of them seem to have any practical impact.

- After Semaev proposed the idea of summation polynomials[53], there has been tremendous research going on to construct non-generic index

calculus like attacks on ECDLP using summation polynomials. The work of Gandry [29] and Diem[16] did propose such attacks but they have their limitations. Extending the scope of such attacks has been a very active area of research.

Since the class of elliptic curves where efficient attacks are known is very small and can easily be avoided for cryptographic purposes, the best attacks on ECDLP for elliptic curves used in practice are generic attacks. Shank's BSGS[54] and Pollard's rho attack[49] are two most important attacks on ECDLP. But we focus on BSGS-type attacks in this chapter because we contribute in that direction.

2.3 Basic Baby-Step Giant-Step Algorithm

If one wants to solve DLP in any group $\mathbb{G}$ with order n, then it is enough to find the discrete logarithm modulo the largest prime divisor of n because of Pohlig Hellman algorithm [30, section 2.9]. Therefore, one can assume the order of group $\mathbb{G}$ to be a prime p, and hence cyclic, when one wants to solve DLP in $\mathbb{G}$. Keeping this in mind, we assume that elliptic curve $\mathbb{E}(\mathbb{F}_q)$ is cyclic, generated by P and its order is a prime p. Then, given $Q \in \mathbb{E}(\mathbb{F}_q)$, ECDLP is to find the integer $x \pmod p$ such that $Q = xP$.

The basic baby-step giant-step (BSGS) introduced by Shank[54] uses the simple idea of division algorithm to solve ECDLP as follows: let $M = \lceil \sqrt{p} \rceil$ where $\lceil \cdot \rceil$ is the ceiling function. Then, division algorithm gives us unique non-negative integers a_0, a_1 such that $x = a_1 M + a_0$ with $0 \leq a_0 < M$ where a_1 is the quotient and a_0 is the remainder when x is divided by M. Since $0 \leq a_0 < M$, it follows that

$$0 \leq a_1 = \frac{x - a_0}{M} < \frac{x}{M} < \frac{p}{M} \leq M$$

Now, $x = a_0 + a_1 M$ implies that $xP = a_0 P + (a_1 M)P$ or equivalently,

$$Q - (a_1 M)P = a_0 P$$

for $0 \leq a_0, a_1 < M$. Therefore, we make following two lists:

1. **Baby steps**: We compute the elements $\{a_0 P : a_0 = 1, 2, ..., M\}$ and store them.

2. **Giant steps**: We compute MP just once and then, we compute giant steps $Q - a_1(MP)$ for $a_1 = 1, 2, ..., M$ and seek a match from the baby steps. These steps are called 'giant steps' because of the big step of MP.

The above match between two lists is guaranteed to occur because of the division algorithm. The clever choice of $M \approx \sqrt{p}$ makes sure that both lists have equal number of elements. Once we have the match between two lists, we get a_0, a_1 which gives us the desired discrete logarithm as $x = a_1 M + a_0$.

- **Space complexity** : Since we have to store elements only from the baby-steps, the space complexity for the basic BSGS is $\lceil \sqrt{p} \rceil$ group elements.

- **Time complexity**[14]

  - Worst-case: $2\lceil \sqrt{p} \rceil$ group operations.
  - Average-case: $\frac{3}{2}\lceil \sqrt{p} \rceil$ group operations.
  - Asymptotic: $O(\sqrt{p})$ group operations.

Note that the asymptotic time complexity $O(\sqrt{p})$ of this algorithm is optimal due to Shoup's lower bound [55] on the complexity of any "generic algorithm" that solves DLP. Therefore, one can not possibly develop a generic algorithm to solve DLP that is faster than $O(\sqrt{p})$ asymptotically. However, it does not negate the possibility of having a generic algorithm which has better average-case complexity than basic BSGS.

2.4 Pollard's Interleaving BSGS

Pollard proposed a different way of computing the two list by 'interleaving'[50, section 3]. In Pollard's method, we compute the elements *alternatively* from both lists and store them all. The difference from the basic BSGS is that we compute all the baby steps first and then we start computing giant steps. To find a match in Pollard's method, the element computed at the k^{th} step in one list is compared with all of the already computed elements of the other list i.e. kP is compared with $Q-i(MP)$ for $i = 1, 2, ..., k-1$ and $Q-k(MP)$ is compared with jP for $j = 1, 2, ..., k-1$.

Pollard's BSGS is slightly faster than basic BSGS, but may require up to twice the storage. On average, this method would require only $\frac{4}{3}\lceil\sqrt{p}\rceil \approx 1.33\lceil\sqrt{p}\rceil$ group operations which is less than the average-case $1.5\lceil\sqrt{p}\rceil$ of basic BSGS[14]. Pollard only indicated the reason behind this slight gain [50, section 3] and we describe it in detail here. The constant $\frac{4}{3}$ arises for the following fact: If two random numbers y and z are chosen in the interval $[0, 1]$, then expected value of $max(y, z)$ is $\frac{2}{3}$. We now describe the role of this fact towards the average-case complexity of this method.

Recall that division algorithm implies that there exists a match between a_0P and $Q - a_1(MP)$ for $0 \leq a_0, a_1 < M = \lceil\sqrt{p}\rceil$. Since the discrete logarithm x is random, it implies that a_0, a_1 are random as well. Different values of a_0, a_1 give different values of $x = a_0 + a_1M$ for the fixed $M = \lceil\sqrt{p}\rceil$ and we take all possible pairs of (a_0, a_1) in the basic BSGS to arrive at the desired, but random, discrete logarithm $x = a_0 + a_1M$.

In other words, randomness of x results in the randomness of a_0 and a_1. Since a_0, a_1 are random, $\frac{a_0}{M}$ and $\frac{a_1}{M}$ are also random and this is where above fact was utilized by Pollard to give slightly faster average-case time complexity. Taking $y = \frac{a_0}{M}$ and $z = \frac{a_1}{M}$, we see that y, z belong to $[0, 1]$. Then

above fact gives us,

$$\text{Exp}\left(max\left(\frac{a_0}{M}, \frac{a_1}{M}\right)\right) = \frac{2}{3}$$

It follows that $\text{Exp}(a_0) \leq \frac{2}{3}M$ and $\text{Exp}(a_1) \leq \frac{2}{3}M$ and thus, we can *expect* a collision between the following two lists:

- $\{a_0P : a_0 = 1, 2, ..., \frac{2}{3}M\}$.
- $\{Q - a_1(MP) : a_1 = 1, 2, ..., \frac{2}{3}M\}$.

Therefore, Pollard's BSGS have average-case time complexity is $\frac{4}{3}M = \frac{4}{3}\lceil\sqrt{p}\rceil \approx 1.33\lceil\sqrt{p}\rceil$ group operations.

2.5 Negation Map and Improved BSGS for ECDLP

Basic BSGS and Pollard's BSGS work in any group. However, negative map($P \mapsto -P$) in elliptic curves can be used to further improve the BSGS attack on ECDLP in elliptic curves[24, section 4]. Recall that for $P = (x_0, y_0)$, its inverse in the Weierstrass model is given by $-P = (x_0, -y_0)$. Thus, $-P$ can be easily computed from $P = (x_0, y_0)$ and both have same X-coordinate. This fact can be exploited to further improve Pollard's BSGS. The main idea is to divide $0 \leq a_0 < M$ into two relatively smaller and special sub-cases, $0 \leq a_0 < \frac{M}{2}$ and $-\frac{M}{2} \leq a_0 < 0$. This use of negation map was mentioned briefly in the article[24, section 4] and we explain this method step by step below.

Fix any $M > 0$. Then, $x = a_0 + a_1M$ with $0 \leq a_0 < M$ implies that $a_1 = \frac{x-a_0}{M} \leq \frac{x}{M} < \frac{p}{M}$. Such $0 \leq a_0 < M$ and $0 < a_1 < \frac{p}{M}$ are unique and always exist. We can always break $0 \leq a_0 < M$ into two equal sub-cases, either $0 \leq a_0 < \frac{M}{2}$ or $\frac{M}{2} \leq a_0 < M$.

1. If $0 \leq a_0 < \frac{M}{2}$, then $0 \leq a_1 < \frac{p}{M}$.

2. If $\frac{M}{2} \leq a_0 < M$, then rewrite $x = a_0 + a_1 M$ as $x = (a_0 - M) + (a_1 + 1)M$. Now, it follows from $\frac{M}{2} \leq a_0 < M$ that $-\frac{M}{2} \leq (a_0 - M) < 0$ and then,

$$0 \leq (a_1 + 1) = \frac{x - (a_0 - M)}{M} < \frac{x + \frac{M}{2}}{M} < \frac{x}{M} + \frac{1}{2} < \frac{p}{M} + \frac{1}{2}$$

That is, $(a_1 + 1) < \frac{p}{M}$ in this sub-case as well.

Combining above two sub-cases, we see that for any a_0 such that $-\frac{M}{2} \leq a_0 \leq \frac{M}{2}$, there exists $0 \leq a_1 < \frac{p}{M}$ such that $x = a_0 + a_1 M$.

The motivation behind changing the interval $0 \leq a_0 < M$ into $-\frac{M}{2} \leq a_0 < \frac{M}{2}$ comes from the fact that X-coordinate of P and $-P$ are same i.e. $\mathrm{X}(P) = \mathrm{X}(-P)$. The advantage of doing so is that we just have to compute $a_0 P$ for $0 \leq a_0 < \frac{M}{2}$ and compute $Q - a_1(MP)$ for $0 \leq a_1 < \frac{p}{M}$, and then look for the match between their X-coordinates, i.e.

$$\mathrm{X}(a_0 P) = \mathrm{X}(Q - a_1(MP))$$

where $0 \leq a_0 < \frac{M}{2}$ and $0 \leq a_1 < \frac{p}{M}$.

Again, we have two lists and to make both sides having equal number of elements, we need to have $\frac{M}{2} \approx \frac{p}{M}$ which implies that $M \approx \sqrt{2p}$. Therefore, $M = \lceil \sqrt{2p} \rceil$ is necessary to start with in this method. Also, note that $\frac{M}{2} \approx \sqrt{\frac{p}{2}}$. In all, we compute the following two lists:

1. $\{a_0 P : a_0 = 0, 1, ..., \frac{M}{2} \approx \sqrt{\frac{p}{2}}\}$($\sqrt{p}$ in basic BSGS).

2. $\{Q - a_1(MP) : a_1 = 0, 1, ..., \frac{M}{2} \approx \sqrt{\frac{p}{2}}\}$ ($\sqrt{p}$ in basic BSGS).

And look for a match on X-coordinate of elements in above two lists which means $Q - a_1(MP) = \pm a_0 P$. Thus, the worst-case time complexity of solving ECDLP using negative map is $2 \cdot \sqrt{\frac{p}{2}} \approx \sqrt{2p} \approx 1.414\sqrt{p}$ group operations which was $2\sqrt{p}$ in the basic BSGS. The average-case time complexity to solve ECDLP using negation map is $\frac{3}{2}\sqrt{\frac{p}{2}} \approx 1.061\sqrt{p}$ group operations which was

$\frac{3}{2}\sqrt{p}$ in the basic BSGS. Therefore, negation map improved the basic BSGS by a factor of $\sqrt{2}$. Similarly, using negation map property with Pollard's BSGS, the average-case time complexity to solve ECDLP would be $\frac{4}{3}\sqrt{\frac{p}{2}} \approx 0.943\sqrt{p}$ group operations[24], again an improvement over Pollard's BSGS $\frac{4}{3}\sqrt{p}$ by a factor of $\sqrt{2}$.

Remark 2.1. *Note that both intervals* $0 \leq a_0 < M$ *and* $-\frac{M}{2} \leq a_0 < \frac{M}{2}$ *have same length,* M*. One can also think about divide* $0 \leq a_0 < M$ *into something like* $-\frac{M}{4} \leq a_0 < \frac{3M}{4}$ *but then the negation map* $P \mapsto -P$ *would not be of much help because we have to compute* a_0P *for* $0 \leq a_0 < \frac{3M}{4}$*.*

2.6 Best Possible BSGS

Shoup [55] already gave the asymptotic lower bound on the number of group operations for any generic algorithm to solve DLP. A natural question is what would be the best possible algorithm of BSGS type, i.e. to give a tight lower bound on the average-case time complexity for any algorithm of BSGS type. Note that the basic BSGS algorithm is not optimal because DLP can only be solved when there is a match between baby steps and giant steps. For example, suppose one has computed a list of k elements in total. So, $\frac{k}{2}$ elements are there in baby steps and $\frac{k}{2}$ elements are in giant steps. Therefore, at most $\frac{k}{2} \cdot \frac{k}{2} = \frac{k^2}{4}$ instances of DLP can be solved from the k elements computed in the basic BSGS algorithm. It follows that the worst-case complexity of solving the desired DLP is the value of k for which $\frac{k^2}{4} = p$, i.e $k = 2\sqrt{p}$ is the worst-case complexity of basic BSGS, as already discussed.

In search of *best possible* BSGS type algorithm, Chateauneuf, Ling and Stinson[9] gave an unrealistic model where computing an arbitrary element $aP + bQ$ is counted as a single operation. The main idea in this unrealistic computational model is to compute one list of elements like $\{a_iP + b_iQ\}$

such that every collision between two distinct elements $a_iP + b_iQ = a_jP + b_jQ$ solves an instance of DLP. Thus, a list of k elements will have $\binom{k}{2} \approx \frac{k^2}{2}$ distinct pairs, or a list of k elements would solve $\frac{k^2}{2}$ instances of DLP. Note that the number of DLP instances $\frac{k^2}{2}$ solved by this unrealistic model is twice as compared to the basic BSGS $\frac{k^2}{4}$. As a result, the worst-case complexity to solve desired DLP under this "best possible" BSGS model would be $k = \sqrt{2p} \approx 1.414\sqrt{p}$ which follows from $\frac{k^2}{2} = p$. To find the average-case complexity, one looks at the probability of solving a given instance of DLP under this model which is roughly $\frac{\frac{k^2}{2}}{p} = \frac{k^2}{2p}$ because there are p instances of DLP in total, and the expected running time is therefore roughly[22]

$$\sum_{k=1}^{\sqrt{2p}} \left(1 - \frac{k^2}{2p}\right) \approx \int_0^{\sqrt{2p}} \left(1 - \frac{x^2}{2p}\right) dx = \frac{2\sqrt{2p}}{3} \approx 0.943\sqrt{p}$$

group operations. For elliptic curves, when using negation map, the lower bound on average-case would become $\frac{2\sqrt{p}}{3}$ group operations. The interesting thing is that both these lower bounds are better than what could be expected from any method (for example, Pollard's rho[49]) based on the birthday paradox.

To get one step closer towards the "best possible" algorithm, Bernstein and Lange [5] proposed "two grumpy giants and a baby" which is also a variant of BSGS algorithm. As the name suggests, they compute three lists in their method: $\{a_0P : a_0 = 0, 1, ...\}$, $\{Q + a_1(MP) : a_1 = 0, 1, ...\}$ and $\{2Q - a_2((M+1)P) : a_2 = 0, 1, 2, ...\}$, where $M \approx \frac{\sqrt{p}}{2}$ and a match between two elements from any pair of the lists can lead to a solution to DLP. Therefore once k elements are computed, we have three lists of size $\frac{k}{3}$ and we can hope to solve about $3(\frac{k}{3})^2 = \frac{k^2}{3}$ instances of DLP. Since $\frac{k^2}{4} < \frac{k^2}{3} < \frac{k^2}{2}$, we can expect this method be better than basic BSGS but not to match the best "ideal" lower bound of the best possible BSGS. The name comes from that fact that latter two walks are "giant steps" but in opposite directions.

The worst-case time complexity is $\sqrt{3p} \approx 1.732\sqrt{p}$ group opera-

Table 2.1: Time complexity of BSGS type attacks

Algorithms	Average-case	Worst-case
Basic BSGS	$\frac{3}{2}\sqrt{p} \approx 1.5\sqrt{p}$	$2\sqrt{p}$
BSGS with negation map	$\frac{3}{2\sqrt{2}}\sqrt{p} \approx 1.061\sqrt{p}$	$\sqrt{2p} \approx 1.414\sqrt{p}$
Pollard's BSGS	$\frac{4}{3}\sqrt{p} \approx 1.333\sqrt{p}$	$2\sqrt{p}$
Pollard's BSGS with negation map	$\frac{4}{3\sqrt{2}}\sqrt{p} \approx 0.943\sqrt{p}$	$\sqrt{2p} \approx 1.414\sqrt{p}$
Best possible BSGS	$\frac{2\sqrt{2}}{3}\sqrt{p} \approx 0.943\sqrt{p}$	$\sqrt{2p} \approx 1.414\sqrt{p}$
Best possible BSGS with negation map	$\frac{2}{3}\sqrt{p} \approx 0.667\sqrt{p}$	$\sqrt{p}$
Two grumpy giants BSGS	$\frac{2\sqrt{3}}{3}\sqrt{p} \approx 1.155\sqrt{p}$	$\sqrt{3p} \approx 1.732\sqrt{p}$

tions. Even though they did not give any precise statement about the average-case asymptotic performance, the theoretic analysis and experimental result[5] suggest that the average-case time complexity of this method is $\frac{2\sqrt{3p}}{3} \approx 1.155\sqrt{p}$ group operations which is slightly better than Pollard's rho method[49]. Galbraith, Wang and Zhang [27] analyzed the two grumpy giants method in case of elliptic curves and using negation map. They also gave efficient ways to perform baby-step giant-steps on elliptic curves by computing "blocks" of points and sharing inversions using the Montgomery trick. However, it remains an open question to determine the exact average-case time complexity of the "two grumpy giants and one baby" algorithm and to develop algorithms of BSGS type that have time complexity closer to the best possible theoretical lower bound.

The time complexity of the BSGS type attacks mentioned in this chapter in the average-case as well as in the worst-case is summarized in the Table 2.1.

Chapter 3

Improved Lower Bound for the Diffie-Hellman Problem on Important Elliptic Curves

3.1 Introduction

For public key protocols based on the discrete logarithm problem (DLP), computational difficulty of solving DLP is a security necessity. The group of points on an elliptic curve over finite field is one such example because there is no efficient algorithm, in general, to solve the (elliptic curve)discrete logarithm problem (ECDLP). However, interesting thing about these DLP-based protocols is that, the security of many such protocols does not exactly rely on the hardness of DLP. For example, the ElGamal public key cryptosystem [19] is secure if and only if the Diffie-Hellman problem (DHP) is hard [30, Proposition 2.10]. Thus, it is enough for an attacker to solve DHP to break the ElGamal cryptosystem. The Diffie-Hellman key exchange[30, Section 2.3], tripartite Diffie-Hellman key exchange[34], ECDSA[33], pairing-based cryptosystems [18], identity-based encryption schemes[10], BLS short signa-

ture scheme[6] and many more public key protocols are some other examples where the security of the protocols depends mainly on hardness on DHP. Since elliptic curves over finite field are widely used in practice to implement the above mentioned protocols, along with many others, it is of paramount importance to investigate the hardness of the elliptic curve Diffie-Hellman problem (ECDHP) and this is the central idea of this chapter.

A brief outline of the chapter is the following: in section 2, we will see the traditional methods and known results on the hardness of DHP on any general group. Section 3 presents some known results on minimum number of group operations required to solve ECDHP on the elliptic curve groups over finite fields. In section 4, we discuss some previous results related to hardness of DHP. In section 5, we present our work which improves the lower bound on ECDHP for various elliptic curves used in practice. Conclusion is given in section 6.

3.2 Reduction of DLP to DHP

To study the hardness of DHP the traditional method, and the only method known so far, involves reduction of DLP to DHP. In such reductions, one tries to solve DLP efficiently(in polynomial time of the input size), using the solution of DHP as sub-routine for polynomially many times in the input size. If there exists such a reduction algorithm, we say that **DLP reduces to DHP** in polynomial time. Moreover, it would imply that DHP is at least as hard as DLP, or equivalently, DLP is no harder than DHP. This is exactly the motivation behind reducing DLP to DHP in polynomial time. Clearly, existence of any such reduction algorithm in case of elliptic curve groups would imply that ECDHP is hard, since ECDLP is hard to solve, in general.

The first, and the only, reduction algorithm known so far which reduces DLP to DHP was proposed by Maurer in his seminal paper [39]. He intro-

duced the technique of implicit representation of elements of a finite field and indicated the use of an auxiliary group in constructing such a reduction algorithm. Before discussing these concepts, we formally define a DH-oracle.

Definition 3.1. *Let $\mathbb{G}$ be a cyclic group written additively and generated by $P \in \mathbb{G}$. A **DH-oracle** is a function that takes $xP, yP \in \mathbb{G}$ as inputs and returns $xyP \in \mathbb{G}$ as output. We write it as $\mathcal{DH}(xP, yP) = xyP$.*

3.2.1 Implicit representation of elements of $\mathbb{F}_p$

Let $\mathbb{G}$ be a cyclic group written additively and generated by $P \in \mathbb{G}$ and p be a prime that divides $|\mathbb{G}|$. We wish to solve the DLP in this group $\mathbb{G}$, assuming that we have access to a DH-oracle in the group $\mathbb{G}$. With this in mind, Maurer and Wolf[42] defined *implicit representation* of a finite field element $y \in \mathbb{F}_p$ as follows:

Let $y \in \mathbb{F}_p$. Then, $A = y'P \in \mathbb{G}$ is called an *implicit representation(with respect to $\mathbb{G}$ and P)* of the element $y \in \mathbb{F}_p$ if $y \equiv y' \pmod{p}$. That is, each element y of $\mathbb{F}_p$ can be attached to a subset of the group $\mathbb{G}$,

$$y \leftrightarrow \{(y + kp)P \,|k \in \mathbb{Z}\}$$

Basically, all elements($A = y'P$) of $\mathbb{G}$ whose discrete logarithm(with respect to P) is congruent to $y \pmod{p}$ are implicit representation of $y \in \mathbb{F}_p$. We denote this by $y \rightsquigarrow A$. It is easy to see that $y \rightsquigarrow A$ is unique if and only if $|\mathbb{G}| = p$. Without loss of generality, we can assume that $|\mathbb{G}| = p$ as it is enough to find the discrete logarithm with respect to prime divisors of $|\mathbb{G}|$ because of the Pohlig-Hellman algorithm[30, Section 2.9]. Thus, we assume that $|\mathbb{G}| = p$ from now onwards. As a result, every finite field element $y \in \mathbb{F}_p$ is implicitly represented by a group element $yP \in \mathbb{G}$ *uniquely*.

Let $yP, zP \in \mathbb{G}$ be implicit representations of $y, z \in \mathbb{F}_p$ respectively. Then basic *algebraic* operations in $\mathbb{F}_p$ can be performed efficiently on implicit representations, using group operations and a DH-oracle in $\mathbb{G}$ and the results

are also in implicit form:

- **Equality testing**: $y \equiv z \pmod{p}$ if and only if $yP = zP$.
- **Addition**: $y + z \rightsquigarrow yP + zP$ (1 group operation in $\mathbb{G}$).
- **Subtraction**: $y - z \rightsquigarrow yP - zP$($O(\log p)$ group operations in $\mathbb{G}$).
- **Multiplication**: $y \cdot z \rightsquigarrow yzP = \mathcal{DH}(yP, zP)$ (1 call to DH-oracle).
- **Inversion**: $y^{-1} = y^{p-2} = \underbrace{y \cdots y} \rightsquigarrow y^{p-2}P$ ($O(\log_2 p)$ DH-oracle calls by using binary expansion).

Observe that the DH-oracle is used only for multiplication and inversion in $\mathbb{F}_p$. Therefore, number of DH-oracle calls required in the reduction algorithm increases with the increase in number of multiplication and inversions needed in the algorithm. We will see the importance of this fact in later sections.

We call the above five operations in $\mathbb{F}_p$ *algebraic operations*. Implicit representations of finite field elements was a novel idea for reducing DLP to DHP because any efficient computation in $\mathbb{F}_p$ can be performed equally efficiently on implicit representations whenever it makes use only of algebraic operations. Examples of such efficient computations in $\mathbb{F}_p$ are:

1. Quadratic reciprocity of an element $y \in \mathbb{F}_p$ can be checked by $(y^{\frac{p-1}{2}})P = P$, since $y^{\frac{p-1}{2}} \equiv 1 \pmod{p}$(the reciprocity condition) if and only if $(y^{\frac{p-1}{2}})P = P$.
2. Computation of square roots using some standard algorithms which use only algebraic operations. For example, yP can be computed from y^2P using an implicit version of the Tonelli-Shanks algorithm[14].

In short, whenever we see the term *implicit representation* in context of a DLP to DHP reduction algorithm, we can assume that following things are

given to us: a prime p and a cyclic group $\mathbb{G}$ of order p and a DH-oracle in $\mathbb{G}$ where elements of finite field $\mathbb{F}_p$ are implicitly represented by elements of group $\mathbb{G}$.

3.2.2 Auxiliary Groups over $\mathbb{F}_p$

In addition to implicit representation of finite field elements, Maurer and Wolf came up with the idea of *auxiliary groups* over $\mathbb{F}_p$ in order to construct a DLP to DHP reduction algorithm in the group $\mathbb{G}$ of order p. As the name suggests, any group(other than the group $\mathbb{G}$) is called an auxiliary group if it is used to achieve the targeted goal of an algorithm. Here our goal is to construct a DLP to DHP reduction algorithm, i.e., to solve DLP, using implicit representation and with the help of a DH-oracle. To achieve this goal, auxiliary group $\mathbb{H}$ defined by Maurer and Wolf[42] must have some specific properties which can essentially be summarized into following two properties:

- Elements of $\mathbb{H}$ can be represented as m-tuples of elements of $\mathbb{F}_p$ for some $m \geq 1$.
- Group operation in this auxiliary group $\mathbb{H}$ can be defined using *small* number of algebraic operations in $\mathbb{F}_p$.

For us, any group $\mathbb{H}$ that satisfies above two properties is called **auxiliary group over** $\mathbb{F}_p$. Maurer and Wolf[42] also mentioned two classes of possible auxiliary groups, satisfying above requirements: elliptic curves $E_0(\mathbb{F}_p)$ where $m = 2$ and subgroups of $\mathbb{F}_{p^n}^{\times}$ for some $n \geq 1$ where $m = n$. They called these groups *applicable auxiliary groups over* $\mathbb{F}_p$. The motivation behind the essential properties of auxiliary groups over $\mathbb{F}_p$ is that any computation in $\mathbb{H}$(for example, equality testing, exponentiation in $\mathbb{H}$) can also be performed explicitly in $\mathbb{G}$, via implicitly representation of elements of $\mathbb{F}_p$. We take the

example of $(\mathbb{F}_{p^2}^{\times}, \cdot)$ to illustrate how its multiplication operation and equality testing can be performed using implicit representations with the help of group operations and the DH-oracle in $\mathbb{G}$.

Assume that $t^2 - \alpha$ is irreducible in $\mathbb{F}_p[t]$ for some $\alpha \in \mathbb{F}_p$ and assume that θ is some element in its extension field which satisfies $\theta^2 = \alpha$. Thus, $\mathbb{F}_p(\theta) \cong \mathbb{F}_{p^2}$. Therefore, $a + b\theta$ represents a general element of $\mathbb{F}_{p^2}^{\times}$ for some $a, b \in \mathbb{F}_p$ with either $a \neq 0$ or $b \neq 0$. Hence, after θ is fixed, each non-zero tuple $(a, b) \in \mathbb{F}_p \times \mathbb{F}_p$ can be thought of as an element of $\mathbb{F}_{p^2}^{\times}$. Now, the multiplication operation in $\mathbb{F}_{p^2}^{\times}$ also gives a non-zero tuple because,

$$(a + b\theta) \cdot (c + d\theta) = (ac + bd\theta^2) + (bc + ad)\theta = (ac + bd\alpha) + (ad + bc)\theta$$

That is, above multiplication results in the non-zero tuple $(ac + bd\alpha, ad + bc)$. With the above discussion in mind, we consider elements of $\mathbb{F}_{p^2}^{\times}$ as non-zero tuples like (a, b). We call the tuple $(aP, bP) \in \mathbb{G} \times \mathbb{G}$ the implicit representation of $(a, b) \in \mathbb{F}_{p^2}^{\times}$.

Now, to compute implicit representation of the multiplication of two elements (a, b) and (c, d) of $\mathbb{F}_{p^2}^{\times}$, we only need to compute the tuple $((ac + bd\alpha)P, (ad + bc)P)$ which can be computed using the tuples (aP, bP), (cP, dP), the group operation in $\mathbb{G}$ and the DH-oracle in $\mathbb{G}$. For example, the DH-oracle gives us acP from aP, cP and bdP from bP, dP. Then, one computes $\alpha(bdP)$ because α is known. Lastly, one computes $acP + \alpha(bdP)$ by using group operations in $\mathbb{G}$ which gives us $(ac + bd\alpha)P$, the first coordinate of the desired element. Similarly, the second coordinate $(ad + bc)P$ can also be computed. This is how the implicit representation of $\mathbb{F}_p$ helps us in computing the implicit representation of the multiplication operation in the auxiliary group $\mathbb{F}_{p^2}^{\times}$.

Moreover, we can also check the equality testing in the auxiliary group $\mathbb{F}_{p^2}^{\times}$ by looking at two equalities in $\mathbb{G}$ as follows:

$$a + b\theta = c + d\theta \quad \Leftrightarrow \quad a \equiv c \pmod{p}, \;\; b \equiv d \pmod{p}$$

$$\Leftrightarrow aP = cP,\ \ bP = dP$$

Thus, we have seen that implicit representation of the group operation of the auxiliary group as well as equality testing in the auxiliary group can very well be done by working only with the group operation in $\mathbb{G}$ and the DH-oracle in $\mathbb{G}$.

Now, we present the motivation and main idea behind using the concepts of implicit representation and auxiliary groups for reducing DLP to DHP.

3.2.3 General Idea of Reducing DLP to DHP using Implicit Representation and Auxiliary Group

Given $Q = xP \in \mathbb{G}$ and the generator P of $\mathbb{G}$, our aim is to compute the integer $x \pmod p$ using the DH-oracle in $\mathbb{G}$. Let us try to understand the motivation behind auxiliary groups over $\mathbb{F}_p$ and implicit representations of $\mathbb{F}_p$ in this context of reducing DLP to DHP. With the help of these two, Maurer and Wolf's general method[42, Proof of Theorem 2] can be described in the following four steps:

- **Step 1**. Choose some suitable cyclic auxiliary group $\mathbb{H}$ over $\mathbb{F}_p$ that satisfies the properties mentioned above. Assume that it is generated by ζ_0.

- **Step 2**. The next step is to **embed** x implicitly into $c \in \mathbb{H}$. This element c must be related to the discrete logarithm x. The main purpose of embedding x implicitly into $c \in \mathbb{H}$ is that it should lead to some implicit equations in $\mathbb{H}$, involving c(thus x) and its generator ζ_0. We call such equations in $\mathbb{H}$ *implicit* because they involve the discrete logarithm x which is unknown.

- **Step 3**. Once we have some implicit equations in $\mathbb{H}$ involving x, c and ζ_o, then we *explicitly* compute the discrete logarithm of c with respect

to ζ_0 in $\mathbb{H}$. Since each implicit equation in $\mathbb{H}$ corresponds to an explicit equation in $\mathbb{G}$, we use implicit representation computation to write the explicit equations in $\mathbb{G}$(corresponding to those implicit equations in $\mathbb{H}$) and those explicit equations helps us to explicitly find the discrete logarithm of c with respect to ζ_0. Observe that all computations(equality testing and exponentiation) actually take place in the original group $\mathbb{G}$ and computing implicit representations of finite field elements is exactly the place where the DH-oracle is needed.

- **Step 4**. At the end, since x and c are related, we **extract** the unknown x from the discrete logarithm of c with respect to ζ_0 found in the previous step.

We put some more light on step 3 above because it contains most of the computations that take place in the algorithm. To find the discrete logarithm of c with respect to ζ_0 explicitly, we write the explicit equations in $\mathbb{G}$ (using the implicit representation computation) corresponding to those implicit equations in $\mathbb{H}$. In other words, one can say that the main contribution of implicit representations is to write explicit equations in the original group $\mathbb{G}$ from the corresponding implicit equations in $\mathbb{H}$ while the role of auxiliary group over $\mathbb{F}_p$ can be thought of as giving us those implicit equations in $\mathbb{H}$. We are calling the equations in $\mathbb{G}$ *explicit* because we can explicitly find both sides of these equations. Solving the explicit equations(equality testings) thus obtained in $\mathbb{G}$ yields us the discrete logarithm of c with respect to ζ_0. The original group $\mathbb{G}$ is the place where any equation solving(equality testing) takes place and to be able to do equality testing in $\mathbb{G}$, we must have explicit equations in $\mathbb{G}$. That is exactly what the implicit representation does, to help us obtain explicit equations in $\mathbb{G}$.

To illustrate these steps, we take the case of an elliptic curve over $\mathbb{F}_p$ of smooth order being used as the auxiliary group:

1. Suppose we have an elliptic curve $E_0(\mathbb{F}_p)$ which is given by $Y^2 = X^3 + aX + b$; $a, b \in \mathbb{F}_p$. Assume that it is generated by $P_0 = (x_0, y_0)$. Moreover, $|E_0(\mathbb{F}_p)| = N$ with N smooth.

2. The algorithm **embeds** the unknown discrete logarithm x implicitly into $c = Q_0 = (x, y)$ for some (unknown)$y \in \mathbb{F}_p$ such that $y^2 = x^3 + ax + b$ i.e. $Q_0 = (x, y) \in E_0(\mathbb{F}_p)$.

3. Now, the algorithm computes the discrete logarithm k of $Q_0 = (x, y)$ with respect to $P_0 = (x_0, y_0)$ *explicitly*. This is done by computing k modulo each prime power of N first(using Pohlig-Hellman algorithm) and then, Chinese Remainder Theorem yields us $k \pmod N$. The interesting thing is that Pohlig-Hellman algorithm is being used in the auxiliary group $E_0(\mathbb{F}_p)$ and this is exactly the place where implicit representation helps us because it transfers an equality in $\mathbb{F}_p$ into the corresponding equality in $\mathbb{G}$.

4. The last step is to **extract** x from k by computing kP_0 because $kP_0 = Q_0$ and the abscissa of the point Q_0 gives us the desired discrete logarithm x. Observe that $Q_0 = (x, y)$ was not *explicitly* known before we computed k. However, once we have computed k, we can compute Q_0 *explicitly* using P_0 and k as $Q_0 = kP_0$.

For more detailed information on how the above steps work, one can see [47].

A natural question one may ask: what is the significance of N being smooth? Or what does this reduction algorithm(with an elliptic curve over $\mathbb{F}_p$ as auxiliary group) tell us about the hardness of DHP? Maurer and Wolf showed that the reduction algorithm described above requires only polynomial number of DH-oracle calls as well as only polynomial number of group operations of $\mathbb{G}$ to solve DLP in the group $\mathbb{G}$ of prime order p if we can

construct an elliptic curve over $\mathbb{F}_p$ of smooth order, say N[42, Theorem 3]. Since their reduction algorithm has to perform the Pohlig-Hellman reduction step for each prime factor of N, the smoothness on N is precisely the reason behind the polynomial sizes of DH-oracle calls and the group operations required in the algorithm. In other words, DLP reduces to DHP in polynomial time if we can construct such an elliptic curve or if we know that such an elliptic curve exists. Therefore, it implies that DHP is at least as hard as DLP in the group $\mathbb{G}$, given the existence of such an elliptic curve. In particular, it would imply that ECDHP is hard because ECDLP is hard, under the same assumption. This shows us the importance and the motivation behind reducing DLP to DHP as well as knowing the existence of smooth order elliptic curve over $\mathbb{F}_p$, since it provides us a tool to study the hardness of DHP.

Unfortunately, existence of smooth order elliptic curve over $\mathbb{F}_p$ is yet to be proved because it depends on some number-theoretic conjecture on the existence of smooth numbers in an interval[42, Section 4.3.2]. Moreover, it is extremely hard to construct smooth order elliptic curve over $\mathbb{F}_p$ for large p and the difficulty increases exponentially with the increase in the size of p. As a result, above reduction algorithm of Maurer and Wolf with elliptic curve over $\mathbb{F}_p$ as auxiliary group fails, in general, to prove the computational equivalence of DLP and DHP in the group $\mathbb{G}$ of order p.

Now, if one fails to prove the computational equivalence of DLP and DHP, what could be the next best thing to decide how hard DHP is? Muzereau *et al.*[47] argued that the next best thing would be to estimate the **minimum** number of group operations required to solve DHP and we describe their approach in the next section.

3.3 Lower Bound of DHP

Since it is hard to prove computational equivalence of DLP and DHP in general, Muzereau *et al.* restricted themselves only to the particular case of group of points on an elliptic curve over a finite field i.e. $\mathbb{G} = \mathbb{E}(\mathbb{F}_q)$ for some finite field $\mathbb{F}_q$. Moreover, we assume that $|\mathbb{G}| = p$ and $q \approx p$. The hardness of DHP on these elliptic curve groups carries a lot of practical significance because of their large scale deployment in the industry. This makes the study of ECDHP more exciting and worthwhile.

To study the hardness of DHP in the particular case of these elliptic curve groups, Muzereau *et al.* [47] proposed an interesting idea of estimating the **minimum** number of group operations required to solve DHP. To achieve this, Muzereau *et al.* re-visited the reduction algorithm of Maurer and Wolf with elliptic curve as auxiliary group, but impose a condition on number of group operations required by the reduction algorithm in their model. Their model assumes that the number of group operations required in the reduction algorithm is almost insignificant when compared with the cost of DLP. Then, the model estimates the minimum number of group operations required to solve DHP as the ratio of the cost of solving DLP and the number of DH-oracle calls. More precisely, their model for a general group $\mathbb{G}$(not necessarily $\mathbb{E}(\mathbb{F}_q)$ as it can be applied to any general group provided the reduction algorithm satisfies the requirements of the model) can be described as follows:

Let C_{DLP}, C_{DHP} be the minimum cost of solving DLP and DHP in $\mathbb{G}$ respectively. Therefore, in the view of a general DLP to DHP reduction algorithm, we get $C_{DLP} = n \cdot C_{DHP} + M$ where n is the number of calls to the DH-oracle and M is the number of group operations required in the reduction algorithm. Now, assuming that $M \ll C_{DLP}$, we get,

$$C_{DHP} = \frac{C_{DLP} - M}{n} \approx \frac{C_{DLP}}{n}$$

Set

$$T_{DH} = \frac{C_{DLP}}{n}$$

Then, the number T_{DH} is exactly what gives the minimum number of group operations needed by any algorithm to solve DHP. This is how Muzereau *et al.* [47] estimated the minimum number of group operations required by any algorithm that solves DHP in any general group $\mathbb{G}$. Note that the condition $M \ll C_{DLP}$ is crucial in their model.

Of course, the aim would be to make n as small as possible to make the value of T_{DH} as large as possible. A large enough value of T_{DH}(of the order which is assumed to be beyond the reach of modern computation power) would ensure the security of those protocols which depend on DHP for their security. This underlines the motivation and strength of their model to study the hardness of DHP.

It is clear that their model works fine to estimate the lower bound on DHP in any general group $\mathbb{G}$ for which there exists a reduction algorithm that satisfies $M \ll C_{DLP}$. For the specific case of $\mathbb{G} = \mathbb{E}(\mathbb{F}_q)$, $|\mathbb{E}(\mathbb{F}_q)| = p$, Muzereau *et al.* applied the same reduction algorithm of Maurer and Wolf we discussed above but showed that there exists (auxiliary) elliptic curve over $\mathbb{F}_p$ of *suitable* order ($\approx \sqrt[3]{p}$) which ensured that $M = O\left(\sqrt[3]{p}\right)$ in the reduction algorithm. Since we are working with elliptic curve group of prime order p, we assume that the best algorithm to solve DLP on this group takes at least $\sqrt{p}$ group operations, i.e. $\sqrt{p} \leq C_{ECDLP}$. Then, it follows that $M \ll C_{ECDLP}$ and thus, they used their model to estimate the lower bound on ECDHP.

More precisely, assuming that the best algorithm to solve DLP on an elliptic curve of order p takes at least $\sqrt{p}$ group operations, Muzereau *et al.*

gave the following estimate on the lower bound on ECDHP [47, Theorem 4]:

Theorem 3.1 (Muzereau *et al.*, 2004)**.** *Let p be a prime. Assuming in the interval $[p+1-\sqrt{p}, p+1+\sqrt{p}]$ there is an integer which is product of three primes of roughly equal size, then there exists a string S which implies that the best algorithm to solve the ECDHP for an elliptic curve of order p takes time at least*

$$O\left(\frac{\sqrt{p}}{(log_2 p)^2}\right)$$

group operations.

The string S in above theorem refers to the information regarding the complete list of divisors of some (nice)integer in the interval $[p+1-\sqrt{p}, p+1+\sqrt{p}]$ which guarantees the existence of an ECDLP to ECDHP reduction algorithm.

Recall that T_{DH} represents the minimum number of group operations to solve DHP, thus the above ratio is equal to T_{DH} for the elliptic curve group of order p where the numerator $\sqrt{p}$ refers to the C_{ECDLP} while denominator $(log_2 p)^2$ refers to the number of DH-oracle calls n. Since $\sqrt{p}$ is a constant, one must minimize the number of DH-oracle calls to achieve a larger and better value of T_{DH}.

Not only that, Muzereau *et al.* explicitly constructed suitable auxiliary elliptic curve $E_0(\mathbb{F}_p)$ for each elliptic curve $\mathbb{E}(\mathbb{F}_q)$, $|\mathbb{E}(\mathbb{F}_q)| = p$ of **SEC2** standard[52] and gave exact estimate on T_{DH} for each elliptic curve $\mathbb{E}(\mathbb{F}_q)$ of the SEC2 standard [47, Table 1, Table 2]. The parameters of those auxiliary elliptic curves were explicitly given [47, Appendix].

The cryptographic importance of these elliptic curves in the SEC2 standard[52] is that they are recommended for practical implementation by *Standard for Efficient Cryptography Group* at *Certicom Corporation* and the standard include all NIST curves[48] and the most used ones in the ANSI standard[1]. Therefore, the elliptic curves in SEC2 standard carry immense

practical significance for practical implementation and we will call these curves **SECG curves** throughout this chapter.

In 2005, Bentahar[3] applied the ideas similar to Muzereau *et al.* but constructed different auxiliary elliptic curves over $\mathbb{F}_p$[3, Appendix C]. Moreover, he used projective coordinates, instead of affine coordinate as in the work of Muzereau *et al.*. By introducing these changes, he reduced the number of DH-oracle calls required in the reduction algorithm. As a consequence, Bentahar improved the estimates [3, Table 1, Table 2] on T_{DH} given by Muzereau *et al.* for each of the SECG curves.

Of course, if we want to improve the estimates on T_{DH} given by Bentahar, we must reduce the number of DH-oracle calls further. That is what we did through a different reduction algorithm. Before we describe that result in section 5 of this chapter, we mention a couple of related work that motivated us towards our reduction algorithm.

3.4 Static Diffie-Hellman Problem and DLP-wAI

Let $\mathbb{G}$ be any cyclic group generated by P and of order p. Then, the DHP is to compute the xyP from the any two *random* elements xP, yP while **static Diffie-Hellman Problem (SDHP)** is to solve a subset of instances of the Diffie-Hellman Problem(DHP), namely those instances where one of the input elements is *fixed*.

Definition 3.2 (SDHP)**.** *Given P and fixed $Q = xP$, and random yP, the static Diffie-Hellman problem (SDHP) in $\mathbb{G}$ is to compute the element xyP.*

Definition 3.3 (SDH_Q-oracle)**.** *SDH_Q-oracle in group $\mathbb{G}$ is a function that takes a random element $yP \in \mathbb{G}$ as input and returns xyP as output where $Q = xP$. We denote it as $\mathcal{SDH}_Q(Q, yP) = xyP$.*

Basically, SDH_Q-oracle solves the SDHP when the fixed element is Q. It is easy to see that SDHP in no harder than DHP because SDHP is a special case of DHP. Therefore, any lower bound on SDHP would also be valid for DHP as well.

Brown and Gallant presented a novel algorithm[8, Theorem 1] that computes the discrete logarithm x of the (fixed)element Q by using SDH_Q-oracle in the group $\mathbb{G}$ of prime order p. They also discussed how their algorithm can be interpreted as a reduction of the DLP to SDHP and gave a rough estimate on the minimum cost of solving the SDHP in the group $\mathbb{G}$ under some assumptions [8, Section 4.1]. The key step in their algorithm is to compute the element x^dP where d is some divisor of $p-1$ and that is exactly where SDH_Q-oracle comes into picture because the element x^dP is computed by making d calls to SDH_Q-oracle in an iterative fashion, i.e. one needs $x^{n-1}P$ to compute x^nP using the SDH_Q-oracle as $\mathcal{SDH}_Q(Q, x^{n-1}P) = x^nP$ for any $n \geq 2$. Thus, one has to make large number of SDH_Q-oracle calls to compute x^dP for large value of d.

Note that Brown and Gallant did mention the work of Muzereau *et al.* on the exact lower bound on DHP for various SECG curves but it remains unclear if their algorithm can be of any help in improving those lower bound estimate on DHP given by Muzereau *et al.*. Nevertheless, Brown and Gallant did give lower bound on SDHP, hence this can also be taken as lower bound on DHP because SDHP is no harder than DHP. When viewed this way, the lower bound on DHP given by Brown and Gallant as well as those given by Muzereau *et al.* in the case of elliptic curve groups are comparable, and Brown and Gallant also remarked that.

Morever, one can also consider the reduction algorithm of Brown and Gallant into the model of Muzereau *et al.* to estimate the lower bound on SDHP. In the model of Muzereau *et al.*, the reduction of Brown and Gallant

must satisfy $M \ll C_{DLP}$. It is an easy observation that d must be quite large to satisfy this condition. Now, for large d, their reduction algorithm needs a large number of SDH_Q-oracle calls to compute the element x^dP as already discussed above. But for the large number of SDH_Q-oracle calls, the lower bound on SDHP thus obtained would be even worse than the one given by Brown and Gallant [8, Corollary 2]. Therefore, one could conclude that the reduction algorithm of Brown and Gallant when viewed in the model of Muzereau *et al.*, does not improve their lower bound on SDHP, and hence lower bound on DHP. Lastly, it also does not improve the lower bound on DHP given by Muzereau *et al.* because lower bound estimate on DHP given by Muzereau *et al.* and those given by Brown and Gallant are comparable.

In another related work in 2006 by Cheon[11, Theorem 1], the idea used was similar to that of Brown and Gallant, but in a different perspective. Brown and Gallant first computed x^dP in their reduction algorithm by using a SDH_Q-oracle and then solved the DLP; and then, they tried to estimate the hardness of SDHP on the basis of their reduction. On the other hand, Cheon was interested in solving the DLP when some additional(auxiliary) input is also given to us. The auxiliary input, in addition to P, $Q = xP$, was the element $x^dP(d|p-1)$ and the problem of computing the secret x from P, $Q = xP$ and x^dP is called the *discrete logarithm problem with auxiliary input* (**DLPwAI**)[12]. Cheon showed that cost of certain instances of DLPwAI is significantly less than the DLP and thus, analyzed the security loss in presence of auxiliary inputs which was the main aim of his paper[11].

This leaves us with the following question: is it possible to incorporate the work of Cheon[11, Theorem 1], and Brown and Gallant[8, Theorem 1] to improve the lower bound for DHP on elliptic curve groups given by Muzereau *et al.*[47, Theorem 4, Table 1, Table 2]. The answer of above question is affirmative and we were able to discover a way that combines all the work

discussed so far to achieve a better and tighter lower bound for DHP known so far on those elliptic curves. We present our work in next section.

3.5 Our Contribution

Reductions of Muzereau *et al.* and Bentahar were based on Maurer and Wolf's general idea of reducing DLP to DHP with elliptic curves as the auxiliary group. Using their reduction algorithm, Muzereau *et al.* presented exact estimates of lower bound for DHP on SECG curves which was further improved by Bentahar. Although, algorithms of Brown and Gallant, and of Cheon both solve DLP (but under different assumptions), it is entirely unclear if their algorithm can be used to construct such a DLP to DHP reduction that can improve Bentahar's lower bound for DHP on SECG curves. This is precisely where our work contributes as we have given such a DLP to DHP reduction algorithm. The contribution of our work can be divided into two parts:

1. Taking motivation from the work of Brown and Gallant[8, Theorem 1], and Cheon[11, Theorem 1], we have constructed a DLP to DHP reduction which requires very small number of DH-oracle calls. Small number of DH-oracle calls is exactly what one aims for to have a tighter estimate of the lower bound for DHP, in the model of Muzereau *et al.*. Moreover, we have discovered a wonderful thing about our reduction algorithm: it fits perfectly well in the general idea of reducing DLP to DHP using implicit representation and auxiliary groups. The auxiliary group used in our reduction algorithm is the **multiplicative group of a finite field**. Note that previous reduction algorithm of Muzereau *et al.* and Bentahar used elliptic curves as the auxiliary group. Hence, our algorithm is the **first** DLP to DHP reduction that uses the multi-

plicative group of a finite field as the auxiliary group to estimate lower bound for DHP. Under an assumption, our algorithm also *improves* the asymptotic lower bound for ECDHP given by Muzereau *et al.*[47, Theorem 4].

2. Even though our reduction algorithm is fundamentally different from the ones in Muzereau *et al.* and Bentahar, a nice thing about our reduction algorithm is that it is still applicable to almost all SECG curves[52]. And the difference between our algorithm and previous algorithms as well as the change in the auxiliary group from an elliptic curve over finite field to the multiplicative group of a finite field is mainly the reason that our reduction algorithm requires very small number of DH-oracle calls. As a consequence, our reduction algorithm improves the previous best known estimates of T_{DH} for SECG curves, given by Bentahar[3, Table 1, Table 2]. Therefore, we have obtained the **tightest** estimates of T_{DH} for SECG curves known so far.

We now present our DLP to DHP reduction algorithm which uses $\mathbb{F}_p^\times$ as auxiliary group.

3.5.1 Reduction Using $\mathbb{F}_p^\times$ as Auxiliary Group

We first present a lemma that is needed in our reduction algorithm.

Lemma 3.1. *Let $\mathbb{G}$ be a cyclic additive group of prime order p, generated by $P \in \mathbb{G}$. Suppose we are given another element $Q = xP \in \mathbb{G}$(x unknown), then $x^dP \in \mathbb{G}$ can be computed by making at most $2\lceil log_2 d \rceil$ calls to a DH-oracle for any positive integer d where $\lceil . \rceil$ is the ceiling function.*

Proof. We start by observing that if $d = 2^r$, it takes exactly r calls to the DH-oracle for computing $x^dP \in \mathbb{G}$.

$$x^2P = \mathcal{DH}(xp, xP)(1\,call)$$
$$x^4P = x^{2^2}P = \mathcal{DH}(x^2P, x^2P)(2\,calls)$$
$$x^8P = x^{2^3}P = \mathcal{DH}(x^4P, x^4P)(3\,calls)$$
$$x^{16}P = x^{2^4}P = \mathcal{DH}(x^8P, x^8P)(4\,calls)$$

So, we see that continuing the same steps we can compute $x^dP = x^{2^r}P \in \mathbb{G}$ for $d = 2^r$ in exactly r calls to the DH-oracle.

Let d be any integer, not necessarily some power of 2. Let k be the unique integer such that $2^{k-1} \leq d < 2^k$. So, we can write the binary representation of d as follows:

$$d = a_0 + a_1 2^1 + a_2 2^2 + ... + a_{k-1} 2^{k-1}$$

where a_i's are 0 or 1 with $a_{k-1} = 1$.

Above remark shows that elements $x^{2^1}P, x^{2^2}P, x^{2^3}P, ..., x^{2^{k-1}}P \in \mathbb{G}$ can be computed by making at most $k-1$ calls to the DH-oracle. Using these elements, it is easy to check that x^dP can be computed by making at most $k-1$ additional calls to the DH-oracle as the number of non-zero a_i's can be at most $k-1$ for $i = 0, 1, ..., k-2$. Therefore, we need to make at most $2(k-1) = 2\lceil \log_2 d \rceil$ calls to the DH-oracle to compute $x^dP \in \mathbb{G}$ since $k - 1 = \lceil \log_2 d \rceil$. This completes the proof of the lemma. □

Example 3.1. We illustrate this by an example. Suppose $d = 13 = 1 + 2^2 + 2^3$, then $k = 4$. Then,

$$x^dP = x^{13}P = \mathcal{DH}(xP,\ \mathcal{DH}(x^{2^2}P,\ x^{2^3}P)\ )$$

We have to make at most 3 calls to compute the elements $x^{2^2}P, x^{2^3}P$. Since there are two non-zero coefficients (other than the leading coefficient) in the binary representation of $d = 13$, we need exactly $3 + 2 = 5$ calls to DH-oracle to compute $x^{13}P$. Note that $5 < 2(k-1) = 6$. However, $x^{15}P$ requires exactly $2(k-1) = 6$ calls as $15 = 1 + 2 + 2^2 + 2^3$

$$x^{15}P = \mathcal{DH}(xP,\ \mathcal{DH}(x^2P, \mathcal{DH}(x^{2^2}P, x^{2^3}P)\ )\ ).$$

Now, we present our DLP to DHP reduction algorithm using implicit representations and $\mathbb{F}_p^\times$ as the auxiliary group. This reduction algorithm is motivated by the work of Gallant and Brown[8, Theorem 1] and Cheon [11, Theorem 1]. The central idea used in both of them is almost the same once one *particular* element is known. Gallant and Brown used SDH_Q-oracle to compute that *particular* element, thus their work yields a reduction algorithm of DLP to SDHP. On the other hand, Cheon assumed the existence of that *particular* element, hence it leads us towards solving DLPwAI. Whereas in our case, Lemma 1 gives us that *particular* element using a small number of DH-oracle calls which leads to our DLP to DHP reduction algorithm presented in Lemma 2.

Much to our surprise, we discovered that our algorithm fits perfectly well into the general idea of Maurer and Wolf reducing DLP to DHP using implicit representation but $\mathbb{F}_p^\times$ as the auxiliary group.

Lemma 3.2. *Let $\mathbb{G}$ be an additive cyclic group generated by $P \in \mathbb{G}$ and the order of P is a prime number p. Let $Q = xP \in \mathbb{G}$. Then, x can be computed using at most $2log_2p\left(\left\lceil\sqrt{\frac{p-1}{d}}\right\rceil + \left\lceil\sqrt{d}\right\rceil\right)$ group operations and by making at most $2\lceil log_2 d\rceil$ calls to the DH-oracle. Here d is a positive divisor of $p-1$.*

Proof. As already mentioned, the proof is based on implicit representation of elements of $\mathbb{F}_p$ using $\mathbb{H} = \mathbb{F}_p^\times$ as the auxiliary group. Recall the unknown x will be implicitly represented by $Q = xP \in \mathbb{G}$. Furthermore, $\mathbb{F}_p^\times$ is a cyclic group with $\phi(p-1)$ generators, where ϕ is the Euler totient function. Since a random element in $\mathbb{F}_p^\times$is a generator with probability

$$\frac{\phi(p-1)}{p-1} > \frac{1}{6log(log(p-1))}$$

which is large enough(see [11]), it is easy to choose a generator of $\mathbb{F}_p^\times$.

Let ζ_0 be a generator of $\mathbb{H} = \mathbb{F}_p^\times$, then

$$x = \zeta_0^{i_0} \ (mod\ p)$$

for some integer i_0 such that $1 \leq i_0 \leq p-1$.

We want to compute i_0 *explicitly* and then x can be computed using above equation. Let $\zeta = \zeta_0^d \, (mod\, p)$. Since $d|(p-1)$, there exists unique cyclic subgroup, $\mathbb{K}$ of $\mathbb{H} = \mathbb{F}_p^\times$ of order $\frac{p-1}{d}$, generated by ζ. Now as $(x^d)^{\left(\frac{p-1}{d}\right)} = 1$, it implies that $x^d \in \mathbb{K}$. Therefore, there exists unique non-negative integer j with $1 \leq j \leq \left(\frac{p-1}{d}\right)$ such that

$$x^d = \zeta^j \qquad (mod\, p) \tag{3.1}$$

Let $d_1 = \left\lceil \sqrt{\frac{p-1}{d}} \right\rceil$. Since j is between 1 and $\frac{p-1}{d}$, there exist unique non-negative integers u_1, v_1 with $0 \leq u_1, v_1 \leq d_1$ such that $j = u_1 d_1 - v_1$. Plugging this value of j in Equation 3.1, we get

$$x^d = \zeta^{u_1 d_1} \zeta^{-v_1} \qquad (mod\, p)$$

which implies,

$$\zeta^{v_1} x^d = (\zeta^{d_1})^{u_1} \qquad (mod\, p) \tag{3.2}$$

Recall that equality of two field elements can also be checked on their implicitly represented elements as follows: $y = z$ *(in* $\mathbb{F}_p^\times$*) is equivalent to* $yP = zP$ *in* $\mathbb{G}$. Therefore, above **implicit equation** in $\mathbb{F}_p^\times$ is equivalent to following **explicit equation** in $\mathbb{G}$:

$$\zeta^{v_1}(x^d P) = (\zeta^{d_1})^{u_1} P \tag{3.3}$$

By Lemma 1 above, we can compute implicit representation $x^d P$ of x^d by making at most $2\lceil \log_2 d \rceil$ calls to the DH-oracle. Looking at Equation 3.3, it is clear that the elements on the left-hand side can be computed using $x^d P$ for any value of v_1 for $0 \leq v_1 \leq d_1$, by repeated addition of previous terms by ζ-times. Similarly, the elements on the right-hand side can be computed for any value of u_1 for $0 \leq u_1 \leq d_1$ using P by repeated addition of previous terms by ζ^{d_1}-times. So, we compute $\zeta^{v_1}(x^d P)$ for each v_1 with $0 \leq v_1 \leq d_1$ and store them. Then, we compare them with each of right-hand side terms(similar to Baby-Step Giant-Step (BSGS) algorithm [45]) to find a match and it yields the integer $j = u_1 d_1 - v_1$.

Note that the non-negative integer $j = u_1 d_1 - v_1$ in Equation 3.1 is

nothing but i_0 modulo $\frac{p-1}{d}$. Now to compute i_0 modulo $(p-1)$ from this integer j, we apply division algorithm on i_0 with divisor $\frac{p-1}{d}$ to get a relation between i_0 and j and it gives us, $i_0 = \left(\frac{p-1}{d}\right)t + j$ for some non-negative integer t. Observe that $0 \leq t < d$, otherwise $i_0 \geq p-1$, a contradiction. Therefore, the integer t can be written uniquely as $t = u_2\left\lceil\sqrt{d}\right\rceil - v_2$ for some $0 \leq u_2, v_2 \leq \left\lceil\sqrt{d}\right\rceil$, again by the division algorithm. Thus, we get the following **implicit equation** in $\mathbb{H} = \mathbb{F}_p^\times$,

$$x = {\zeta_0}^{i_0} = {\zeta_0}^{j+t\left(\frac{p-1}{d}\right)} = {\zeta_0}^{j}{\zeta_0}^{\left(\frac{p-1}{d}\right)\left(u_2\left\lceil\sqrt{d}\right\rceil - v_2\right)}$$

Or

$$\left({\zeta_0}^{\frac{p-1}{d}}\right)^{v_2} x = \left({\zeta_0}^{\left(\frac{p-1}{d}\right)\left\lceil\sqrt{d}\right\rceil}\right)^{u_2} {\zeta_0}^{j}$$

The last **implicit equation** in $\mathbb{H} = \mathbb{F}_p^\times$ is equivalent to the following **explicit equation** in $\mathbb{G}$,

$$\left({\zeta_0}^{\frac{p-1}{d}}\right)^{v_2} (xP) = \left({\zeta_0}^{\left(\frac{p-1}{d}\right)\left\lceil\sqrt{d}\right\rceil}\right)^{u_2} ({\zeta_0}^{j}P) \tag{3.4}$$

As xP and ${\zeta_0}^{j}P_0$ are known, we can solve for u_2, v_2 by finding a match between two sides of the Equation 3.4, just as we found u_1, v_1 above. This solution for u_2, v_2 would give us,

$$i_0 = \left(\frac{p-1}{d}\right)\left(u_2\left\lceil\sqrt{d}\right\rceil - v_2\right) + j$$

Thus, we have **explicitly** computed i_0. Lastly, we **extract** the original discrete logarithm x from this i_0 and the relation $x = {\zeta_0}^{i_0} \pmod p$.

It is easy to see that it takes at most $2\log_2 p\left(\left\lceil\sqrt{\frac{p-1}{d}}\right\rceil\right)$ group operations to find the match in Equation 3.3 and at most $2\log_2 p\left(\left\lceil\sqrt{d}\right\rceil\right)$ group operations to find the match in Equation 3.4. Therefore, we have computed the discrete logarithm x using at most $2\log_2 p\left(\left\lceil\sqrt{\frac{p-1}{d}}\right\rceil + \left\lceil\sqrt{d}\right\rceil\right)$ group operations and by making at most $2\lceil\log_2 d\rceil$ calls to the DH-oracle. This completes the proof. □

Remark 3.1. One can get rid of the factor $\log_2 p$ from the above time complexity using KKM improvement [36]. Then, above time complexity reduces

to $2\left(\left\lceil\sqrt{\frac{p-1}{d}}\right\rceil+\left\lceil\sqrt{d}\right\rceil\right)$.

Remark 3.2. Observe that x^d is unknown in Equation 3.2 because x is unknown. This makes Equation 3.2 an **implicit equation** in $\mathbb{H}=\mathbb{F}_p^\times$. This is exactly the place where implicit representation computation comes into play, to compute implicit representation x^dP of x^d.

Remark 3.3. Our algorithm follows the general idea of DLP to DHP reduction algorithm of Maurer and Wolf with $\mathbb{H}=\mathbb{F}_p^\times$ as the auxiliary group where the unknown x is *embedded implicitly* into itself, i.e. $c=x$. To the best of our knowledge, our algorithm is the **first DLP to DHP reduction algorithm** that uses $\mathbb{H}=\mathbb{F}_p^\times$ as auxiliary group but *does not* use the Chinese Remainder Theorem to compute the discrete logarithm.

We already saw Theorem 3.1 of Muzereau *et al.* on the lower bound for ECDHP, assuming that the best algorithm to solve ECDLP on an elliptic curve of order p takes at least $\sqrt{p}$ group operations. Under the same assumption as above, our reduction algorithm will prove the following theorem[38, Theorem 2] which improves the lower bound for ECDHP given by Muzereau *et al.*.

Theorem 3.2. *For a prime p, assume that there exists a divisor d of $p-1$ of the size roughly equal to $\sqrt[3]{p}$. Then, any algorithm that solves ECDHP for an elliptic curve (sub-)group of order p requires at least*

$$O\left(\frac{\sqrt{p}}{log_2 d}\right)$$

group operations.

Proof. Since we are dealing with an elliptic curve of prime order p, we assume that the best algorithm to solve elliptic curve discrete logarithm problem will take at least $\sqrt{p}$ group operations. Recall the model of Muzereau *et al.* to estimate the minimum number of group operation required to solve DHP, discussed in section 3.3. We saw that Muzereau *et al.* proposed that the

number $T_{DH} = \frac{C_{DLP}}{n}$ is exactly what gives the minimum number of group operations needed by any algorithm to solve DHP, assuming $M \ll C_{DLP}$ where C_{DLP} denotes the minimum cost of solving DLP and n is the number of calls to the DH- oracle and M is the number of group operations required in the reduction algorithm. Clearly, the aim would be to make n as small as possible to have the value of T_{DH} as large as possible.

In case of an elliptic curve group of prime order p, one can take $C_{ECDLP} \approx \sqrt{p}$ under our assumption. Since there is a divisor d of $p-1$ such that $d \approx \sqrt[3]{p}$, then it is easy to check that

$$M \leq \left(\left\lceil \sqrt{\frac{p-1}{d}} \right\rceil + \left\lceil \sqrt{d} \right\rceil \right) \approx \sqrt[3]{p}$$

satisfying the condition $M \ll C_{ECDLP}$ because $\sqrt[3]{p} \ll \sqrt{p}$ and $\sqrt{p} \approx C_{ECDLP}$.

Since $n \leq 2\lceil \log_2 d \rceil$, we finally get,

$$T_{DH} = O\left(\frac{\sqrt{p}}{log_2 d}\right).$$

This implies that the minimum number group operation to solve ECDHP on any elliptic curve group of prime order p by any algorithm is asymptotically $O\left(\frac{\sqrt{p}}{log_2 d}\right)$ if there exists a divisor d of $p-1$ of size approximately $\sqrt[3]{p}$. This completes the proof. □

Remark 3.4. If we assume that a divisor d of $p-1$ of size approximately $\sqrt[3]{p}$ exists, then the above result shows that the cost of ECDHP is getting closer to the cost ECDLP.

Remark 3.5. Note that the total number of group operations, M needed in the reduction algorithms of Muzereau *et al.* [47] and Bentahar [3] was also of the same order i.e. $M \approx \sqrt[3]{p}$. This indicates the importance of such a divisor d of size approximately $\sqrt[3]{p}$ in our reduction algorithm.

3.5.2 Advantage of $\mathbb{F}_p^\times$ over $E_0(\mathbb{F}_p)$ as the auxiliary group

The difference between algorithms of Muzereau *et al.*, Bentahar and our algorithm as well as the change of auxiliary group from $E_0(\mathbb{F}_p)$ to $\mathbb{F}_p^\times$ both have their implications on the number of DH-oracle calls, consequently affecting the value of T_{DH}. Since algorithm used in previous reductions of Muzereau *et al.* and Bentahar required several iterations of Pohlig-Hellman algorithm, one had to compute a large number of implicitly represented elements in those reduction algorithms. Therefore, a large number of DH-oracle calls were needed in previous reductions. On the other hand, our reduction algorithm requires **only** one implicitly represented element x^dP of $x^d \in \mathbb{F}_p^\times$. This element can be computed by using at most $n \leq 2\lceil \log_2 d \rceil$ DH-oracle calls which can further be made really small by taking small value of d.

Recall that the addition operation in $E_0(\mathbb{F}_p)$ requires many multiplications in $\mathbb{F}_p$(one multiplication in $\mathbb{F}_p$ means one DH-oracle call to compute implicit representation) and many inversions in $\mathbb{F}_p$ (one inversion in $\mathbb{F}_p$ means on average $\frac{3}{2}\lceil \log_2 p \rceil$ calls to the DH-oracle to compute implicit representation). Thus, in terms of DH-oracle calls, computing the sum of elements in $E_0(\mathbb{F}_p)$ is much more expensive than multiplying elements in $\mathbb{F}_p^\times$.

Since our main aim through this reduction algorithm is to increase the value of T_{DH} which is inversely proportional to number of DH-oracle calls n, it will be nice to reduce the number of DH-oracle calls as much as possible. That is exactly what our reduction algorithm does by using $\mathbb{F}_p^\times$ as auxiliary group. This shows that the advantage of our reduction algorithm in achieving the improved value of T_{DH}, over the previous reduction algorithms which used $\mathbb{H} = E_0(\mathbb{F}_p)$ as the auxiliary group.

3.5.3 Tightest Value of T_{DH} for SECG Curves

In this section, we study the lower bound for ECDHP on various important elliptic curves parameters[52] and show the improvement made by our reduction algorithm on the lower bound for ECDHP on those curves. These curves are recommended in SEC 2 by *Standard of Efficient Cryptography Group (SECG)* at Certicom Corporation to be used for practical purposes and we are calling those curves **SECG** curves. These SECG curves are divided into two sub-categories: curves over prime fields of large odd characteristic and curves over binary fields. The prime p denotes the order of those SECG curves defined over prime fields of odd characteristic. For remaining SECG curves defined over binary fields, p denotes the prime divisor of the order of the curve, with a very small co-factor of either 2 or 4.

It should also be noted that SECG curves [52] include all curves recommended by NIST [48] and the most used ones in ANSI [1]. These covers the most commonly used curves in practice. Thus, these are important curves from the point of view of public key cryptography.

Muzereau *et al.*[47, Table 1, Table 2] used the value of T_{DH} as the lower bound for group operations to break DH-protocol and also gave the estimates for T_{DH} on various SECG curves. Thereafter, Bentahar [3, Table 1, Table 2] improved the previous values of T_{DH} given by Muzereau*et al.* and his estimates remained the best estimates till date.

Now, in our algorithm, with $\mathbb{F}_p^\times$ as the auxiliary group, we have $n \leq 2\lceil \log_2 d \rceil$ and $M \leq 2\left(\left\lceil \sqrt{\frac{p-1}{d}} \right\rceil + \left\lceil \sqrt{d} \right\rceil\right)$ where d is some divisor of $p-1$. As per the discussion above, to achieve a tighter(larger) value of T_{DH} using our reduction algorithm, one should try to make $n \leq 2\lceil \log_2 d \rceil$ as small as possible, which forces d to be small as well. On the other hand, we have to make sure that $M \approx \sqrt[3]{p}$, so that it does not violate $M \ll C_{ECDLP} \leq \sqrt{p}$. It

is not hard to see that for really small value of d, M is inversely proportional to d. Therefore, too small value of d must not be used to avoid the violation of $M \ll C_{ECDLP} \leq \sqrt{p}$. Also note that $d \approx \sqrt[3]{p}$ yields $M \approx \sqrt[3]{p}$ in our reduction algorithm.

Keeping all these in mind, we factored $p-1$ and found that most of SECG curves contain divisors d which are between $\sqrt[3]{p}$ and $\sqrt{p}$ and we have taken the smallest such d in the range $\sqrt[3]{p}$ and $\sqrt{p}$ to compute the values in Table 1 and 2 given below. For those curves where such a divisor d does not exist, we have chosen the largest d less than $\sqrt[3]{p}$ to compute the values in the tables. For exact values of the d, see appendix A and appendix B.

For those choices of d, we calculated *exact* number of the DH-oracle calls, $n \leq 2\lceil \log_2 d \rceil]$ using binary expansion of d. The values of n thus achieved are significantly small as compared with the values of n shown by Bentahar [3, Table 1,Table 2](and much smaller than those in the work of Muzereau *et al.* [47, Theorem 1, Theorem 2]). Consequently, these significantly small values of n resulted in much tighter(larger)values of T_{DH} for all SECG curves. Therefore, it implies that we have given the **tightest lower bound**, known so far, for ECDHP on these SECG curves[52] (except SECP224K1). In other words, our results shows the gap between the cost of ECDHP and ECDHP to be the **least**(known so far) for these curves and it leads us *one step closer* towards the computational equivalence of ECDHP and ECDLP for these important curves.

Moreover, for curves SECP521R1, SECT409R1, SECT571R1, SECT571K1, Bentahar was unable to construct the auxiliary elliptic curves. However, we had no problem applying our algorithm to these curves and the lower bound estimates of T_{DH} on these curves are also given here. One additional advantage of our algorithm is that the values of M in our algorithm are less than

Table 3.1: Summary of results for curves of large prime characteristic

SECP Curve	$\log_2\sqrt{\lvert E\rvert}$	$\log_2 M$	$\log_2 n$	$\log_2 T_{DH}$	ADV
SECP112R1	55.89	48.34	4.59	51.30	6.90
SECP112R2	54.90	37.54	5.88	49.01	5.51
SECP128R1	64.00	43.45	6.02	57.98	5.58
SECP128R2	63.00	48.23	5.49	57.51	6.11
SECP160K1	80.00	48.39	6.55	73.45	5.45
SECP160R1	80.00	53.85	6.30	73.70	5.70
SECP160R2	80.00	47.53	6.70	73.30	5.30
SECP192K1	96.00	84.31	5.36	90.64	6.84
SECP192R1	96.00	55.51	6.97	89.03	5.23
SECP224R1	112.00	98.50	5.55	106.45	6.85
SECP224K1	-	-	-	-	-
SECP256K1	128.00	86.12	7.00	121.00	5.60
SECP256R1	128.00	86.06	7.00	121.00	5.60
SECP384R1	192.00	141.33	7.33	184.67	5.87
SECP521R1	260.50	196.26	7.67	252.83	6.03

or of almost same order as the ones given by Bentahar[3, Table 1, Table 2] for most of SECG curves.

Table 3.1 and Table 3.2 present the key values, $\log_2 M$, $\log_2 n$ and $\log_2 T_{DH}$ for various SECG curves. The tables also have the value of $\log_2\sqrt{|E|}$ which refers to the assumed minimum cost of solving DLP in that particular SECG curve E. The column under ADV shows the number of security bits gained by the values of T_{DH} in our algorithm over the previous best known values of T_{DH} given by Bentahar[3]. Moreover, the present algorithm works for the curves SECP521R1, SECT571R1, SECT571K1 as well which were out of reach in previous work due to inability to construct auxiliary elliptic curves, and Table 3.1 and Table 3.2 give the key data for these curves as well.

It should also be remarked that the current algorithm fails for the curve SECP224K1 as there does not exist any divisor of $p-1$ of appropriate size. Therefore, Bentahar's result still gives the tightest value of T_{DH} for this

curve.

To understand the advantage gained by our result over the work of Bentahar[3], as an example we consider the security of ECDHP for SECP256R1, see Table 3.1. SECP256R1 is the NIST curve **P-256** which is a randomly chosen elliptic curve defined over a finite field of 256 bit. Thus, the best known algorithm at present to solve ECDLP on this curve takes on an average 2^{128} group operations. Now, our algorithm implies that ECDHP can not be solved in less than 2^{121} group operations, in contrast to $2^{115.4}$ group operations from the work of Bentahar[3, Table 1]. This shows that there is a gain factor of $2^{5.6}$ over the previous best $2^{115.4}$ given by Bentahar for the curve SECP256R1. If we assume that today's computational power is incapable of performing 2^{121} group operations(which is considered to be true by many), then ECDHP on the curve SECP256R1 is secure and any cryptography protocol which rely on DHP for its security can safely be implemented on the curve SECP256R1, under the assumption above.

3.6 Conclusion

In this chapter, we presented a DLP to DHP reduction algorithm that uses implicit representation technique of Maurer and Wolf but uses $\mathbb{F}_p^\times$ as the auxiliary group. This is the *first* reduction algorithm which estimates the lower bound for DHP but uses $\mathbb{F}_p^\times$ as the auxiliary group. Previous such reductions used elliptic curve over finite field as the auxiliary group. Our reduction algorithm provides us a very important cryptographic tool to study the hardness of DHP because it yields the **tightest** estimate of the lower bound for ECDHP for recommended SECG elliptic curves[52] which are widely used for practical implementation. This shows the practical and cryptographic significance of our work.

Table 3.2 : Summary of results for curves of even characteristic

SECT Curve	$\log_2\sqrt{\lvert E\rvert}$	$\log_2 M$	$\log_2 n$	$\log_2 T_{DH}$	ADV
SECT113R1	56.00	38.06	5.67	50.33	5.73
SECT113R2	56.00	38.17	5.76	50.25	5.65
SECT131R1	65.00	58.75	4.46	60.54	7.24
SECT131R2	65.00	51.57	5.43	59.57	6.27
SECT163K1	81.00	54.56	6.36	74.64	5.64
SECT163R1	81.00	54.69	6.36	74.64	5.64
SECT163R2	81.00	67.16	5.56	75.45	6.45
SECT193R1	96.00	61.74	6.76	89.25	5.45
SECT193R2	96.00	56.08	6.99	89.01	5.21
SECT233K1	115.50	79.89	6.77	108.73	5.73
SECT233R1	116.00	77.72	6.92	109.08	5.58
SECT239K1	116.00	79.70	6.87	111.63	5.63
SECT283K1	140.50	94.51	7.15	133.35	5.65
SECT283R1	141.00	94.61	7.18	133.82	5.62
SECT409K1	203.50	150.09	7.44	196.07	5.87
SECT409R1	204.00	136.70	7.66	196.34	5.64
SECT571K1	284.50	190.46	8.08	276.41	5.71
SECT571R1	285.00	190.77	8.15	276.85	5.65

Chapter 4

A Probabilistic Baby-Step Giant-Step Algorithm

4.1 Introduction

The elliptic curve discrete logarithm problem (ECDLP) is one of the most used primitives in the public key cryptography because there is no efficient algorithm to solve it, in general. A number of public key protocols such as ECDSA[33] are based on this problem and hardness of this problem is an absolute necessity for the security of such protocols. Therefore, any new algorithm that indicates any kind of weakness in ECDLP would be of huge cryptographic importance and that is exactly where this chapter contributes.

In this chapter, a new *generic* algorithm to solve the DLP, and thus ECDLP as well, is presented which is motivated by *implicit representation, auxiliary groups* and the usual baby-step giant-step(hereafter, BSGS) algorithm. Our algorithm is perhaps the *first* of its kind which uses *implicit representation* and *auxiliary group* to solve DLP. Moreover, using randomization with parallelized collision search, our algorithm indicates some weakness in NIST curves over prime fields which are considered to be the most

conservative and safest curves among all NIST curves.

This chapter is organized as follows: section 2 discusses BSGS algorithm and the motivation behind our algorithm. Section 3 presents our algorithm to solve DLP. In section 4, we will analyze the security of NIST curve P-256 by applying our algorithm on P-256. We conclude with section 5.

4.2 BSGS and Motivation

Let G be an additive cyclic group of prime order p and generated by P. Recall that the *discrete logarithm problem* (**DLP**) in G is to compute the integer x from the given element $Q = xP \in \mathbb{G}$ and if the group is an elliptic curve, we call this the elliptic curve discrete logarithm problem (ECDLP). This integer x is called the discrete logarithm of Q with the base P. The generic algorithms of Shank's BSGS[54] and Pollard's rho[49, 50] are the best known attacks on ECDLP. As discussed in chapter 2, the time complexity of these attacks are exponential $O(\sqrt{p})$(see Table 2.1), thus they are not efficient.

In this chapter, we develop and study a different version of BSGS algorithm which is much faster than Shank's BSGS in some specific cases. Shank's BSGS in the group $\mathbb{G}$ treats the discrete logarithm x as just an integer and exploits the size of x to find it. On the other hand, our algorithm treats x as an element of the group $\mathbb{F}_p^\times$ and exploits the order of x in the cyclic group $\mathbb{F}_p^\times$. The novelty of our approach comes from *implicit representation* using $\mathbb{F}_p^\times$ as the *auxiliary group*. Maurer and Wolf introduced the concepts of implicit representation of elements of $\mathbb{F}_p$ and auxiliary group over $\mathbb{F}_p$ to reduce DLP to DHP and so far, these concepts have been used for this purpose only. We have already described them in detail in the section 3.3. The motivation for our attack on DLP comes from the work of Brown and Gallant[8, Theorem 1] and Cheon [11], already discussed in the section 3.4.

However, the context we are interested in is quite different: our algorithm does solve DLP in $\mathbb{G}$ but does not require a static DH-oracle in $\mathbb{G}$ or any auxiliary input from the group $\mathbb{G}$ to solve DLP. On the other hand, their algorithm can solve DLP only when either a static DH-oracle is given(Brown and Gallant) or some auxiliary input is given(Cheon). Therefore, this is the **first** time that implicit representation and auxiliary group are used to solve DLP but without requiring any help from a DH-oracle in $\mathbb{G}$. Our algorithm has some similarity with the four steps discussed in section 3.2.3 except that we no longer need the DH-oracle to do the implicit computations and still, we can solve DLP. This is a very pleasant situation because our algorithm can be employed as a *practical* attack on DLP, unlike DLP to DHP reduction which is a theoretical model to study the hardness of DHP.

We present our algorithm in the next section. Our approach leads to a way to reduce the discrete logarithm problem to a problem in $\mathbb{F}_p^\times$. The advantage of this approach is that $\mathbb{F}_p^\times$ has many subgroups and one can exploit the rich and well understood subgroup structure of $\mathbb{F}_p^\times$.

4.3 Our Contribution

Let G be an additive cyclic group of prime order p and generated by P. Recall the definition of implicit representation given in section 3.2.1. Basically, for $y \in \mathbb{F}_p$, $yP \in G$ is called the implicit representation of $y \in \mathbb{F}_p$(with respect to G and P). For auxiliary group, see section 3.2.2. The following lemma comes from the idea of implicit representation of a finite field, proposed by Maurer and Wolf [42].

Lemma 4.1. *Let a, b be any two integers. Then $a = b \pmod p$ if and only if $aP = bP$ in G.*

Proof. Assume that $a = b \pmod p$, then $a = tp + b$ for some integer t. Then

$aP = tpP + bP = bP$. Conversely, assume that $aP = bP$, then $(a-b)P = 0$ in G and this means $p|(a-b)$ which implies that $a = b \pmod{p}$. □

The usefulness of this lemma is to be able to decide on the equality in $\mathbb{F}_p^\times$ by looking at the equality in G. The following algorithm to solve the discrete logarithm problem uses the order of the discrete logarithm in the multiplicative group of a finite field. This algorithm is different from the baby-step giant-step [30] as it uses the implicit representation with multiplicative group of a finite field as auxiliary group.

Theorem 4.1. *Let G be an additive cyclic group generated by P and order of P is a prime p. Let $Q = xP$ be another given element of G(x is unknown). For a given divisor d of $p-1$, let H be the unique subgroup of $\mathbb{F}_p^\times$ of order d. Then, one can decide whether or not x belongs to H in $O(\sqrt{d})$ steps. Furthermore, if x belongs to H, the same algorithm will also find the discrete logarithm x in $O(\sqrt{d})$ steps where each step is an exponentiation in the group G.*

Proof. Since H is a subgroup of the cyclic group $\mathbb{F}_p^\times$, we assume that it is generated by some element ζ. If the generator of H is not given to us, we can compute it using a generator of $\mathbb{F}_p^\times$ and d. The proof of whether x belongs to H or not follows from the well-known baby-step giant-step algorithm [30, Proposition 2.22] to compute the discrete logarithm.

Let n be the smallest integer greater than $\sqrt{d}$. Then $x \in H$ if and only if there exists an integer k with $0 \le k \le d$ such that $x = \zeta^k \pmod{p}$. Note that any integer k between 0 and d can be written as $k = an - b$ for unique integers a, b with $0 \le a, b \le n$, by division algorithm. Therefore, $x \in H$ if and only if there exist two integers a, b with $0 \le a, b \le n$ such that $x = \zeta^{an-b} \pmod{p}$, or equivalently $\zeta^b x = \zeta^{na} \pmod{p}$. Using the lemma above, we see that $x \in H$ if and only if there exist two integers a, b with $0 \le a, b \le n$ such that $\zeta^b xP = \zeta^{na}P$, equivalently $\zeta^b Q = (\zeta^n)^a P$ as $Q = xP$.

Now, we create a list $\left\{\zeta^b Q : 0 \leq b \leq n\right\}$. Then we generate elements of the form $(\zeta^n)^a P$ for each integer a in $[0, n]$ and try to find a collision with the earlier list. When there is a collision, i.e., $\zeta^b Q = (\zeta^n)^a P$ for some $0 \leq a, b \leq n$, it means that $x \in H$. Otherwise, $x \notin H$.

Moreover, if $x \in H$ then $\zeta^b Q = (\zeta^n)^a P$ for some $0 \leq a, b \leq n$. So, we use the integers a and b to compute $\zeta^{an-b} \pmod p$ which is nothing but the discrete logarithm x. Since the two lists require computation of at most $2n$ exponentiations, the worst case time complexity of the algorithm to check whether or not $x \in H$, as well as to compute x(if $x \in H$) would be $O(n) \approx O(\sqrt{d})$ steps. This completes the proof. □

Thus, we saw that above theorem can solve the discrete logarithm problem in some cases while using implicit representation and $\mathbb{F}_p^\times$ as auxiliary group. Two things come out of this theorem:

(A) The theorem can be used to check if the secret key x belongs to some small (fixed)subgroup of $\mathbb{F}_p^\times$ or not. If it does, then the algorithm also finds the secret key x in far less time than the best generic attacks on DLP.

(B) If somehow it is known to an attacker that the secret key is in some subgroup H of $\mathbb{F}_p^\times$, that information can be used to develop a much faster attack than best known generic attack on DLP, for example, Pollard's rho.

Both of these occurrences have their cryptographic relevance which signifies the importance of above theorem. There is another security issue that above theorem brings to the fore. We take the example of NIST curves defined over prime fields of different size viz. P-192, P-224, 256, P-384, P-521 and p denotes the respective prime order of the curves. Since the above algorithm depends on d and $p-1$ factors into small divisors, the above theorem

is applicable to each of the five NIST curves[48]. Although, one can say that probability of randomly chosen secret x being inside a particular subgroup of $\mathbb{F}_p^\times$ can be very small, the availability of so many divisors d of $p-1$ of different sizes itself is not a desirable security feature from the cryptographic point of view and it is always a sound security practice to exclude any such probability, however small. Therefore, as a security necessity, it is highly **recommended** that $p-1$ should be of the form $k \cdot p'$ for a very small value of k and some prime p' so that above algorithm does not provide any faster attack on DLP than the generic attacks.

Remark 4.1. We already discussed that above algorithm is applicable on all the five prime order NIST curves [48] viz. P-192, P-224, P-256, P-384, P-521. Although the probability of a randomly chosen secret key x being inside a particular subgroup of $\mathbb{F}_p^\times$ can be very small, one should always do a security check on the secret key x for its safety and our algorithm provides such a security check on the secret key x as follows: check, using our algorithm for each curve, if the secret key x belongs to any of two (large enough)subgroups whose orders are mentioned in the appendix C. If it does, we *discard* the secret key because our algorithm finds the secret key much faster that generic attacks.

Suppose that $p-1$ has large enough(but a lot smaller than $p-1$) divisor d and H is the unique subgroup of $\mathbb{F}_p^\times$ of order d. One can argue that the drawback of the deterministic algorithm given in Theorem 4.1 is that it might fail to solve DLP because the probability of x belonging to H is very small. To deal with such a scenario, one would try to increase the probability. One way to increase the probability is to increase the size of d, if such d exists. Clearly, this is not a desirable solution because the computational cost depends on the size of the subgroup.

In this situation, the question remains, what happens if no information

about the secret x is known. We develop a probabilistic algorithm (Theorem 4.2) to expand our attack. To understand this probabilistic attack properly, we study it on the curve P-256. This is an NIST recommended curve over a prime field and is considered secure. Our study, which we present in details in section 4.4 indicates some weakness in this curve.

The above algorithm can be parallelized which helps us overcome this obstacle by increasing the probability. We have *randomized* the above algorithm where the random inputs will be running on parallel processes or threads. This parallelization along with collision algorithm (based on birthday paradox) [30, Theorem 5.38] yields a randomized probabilistic algorithm which can solve DLP with a given probability.

Collision Theorem: An urn contains N balls, of which n balls are red and $N-n$ are blue. One randomly selects a ball from the urn, replaces it in the urn, randomly selects a second ball, replaces it, and so on. He does this until he has looked at a total number of m balls. Then, the probability that he selects at least one red ball is

$$\Pr(\textit{at least one red ball}) = 1 - \left(1 - \frac{n}{N}\right)^m \geq 1 - e^{\frac{-mn}{N}}.$$

Theorem 4.2. *Let G be an additive cyclic group generated by P and the order of P is a prime p. Let $Q = xP$ be another given element of G(x is unknown). For a given divisor d of $p-1$, let H be the unique subgroup of $\mathbb{F}_p^\times$ of order d. Then, x can be computed in $O(\sqrt{d})$ steps with probability at least $1 - e^{\left(\frac{-dm}{p-1}\right)}$ if one has access to m parallel threads.*

Proof. The main idea is to run the algorithm from Theorem 4.1 on each of m threads as follows. We randomly selects m elements $y_1, y_2, .., y_m$ in $\mathbb{F}_p^\times$ and compute corresponding m elements $Q_1 = y_1Q = (y_1x)P$,...,$Q_m = y_mQ = (y_mx)P$ of G. Now, we run the above algorithm on each of m parallel threads, with element $Q_i = (y_ix)P$ running on i^{th} thread. Let $z_i = y_ix \pmod p$ for $i = 1, .., m$. If $z_i \in H$ for some i, $1 \leq i \leq m$; then the algorithm on that

thread returns z_i. Once we have z_i for some i, we compute $z_i \cdot {y_i}^{-1} \pmod{p}$ which is nothing but the discrete logarithm x.

The collision theorem above tells us about the probability of at least one z_i belonging to H for $1 \leq i \leq m$. In present case, $\mathbb{F}_p^\times$ with $p-1$ elements is the urn, so $N = p-1$. The elements of H are red balls, so $n = d$. Since we are randomly selecting m elements $y_1, .., y_m$ from $\mathbb{F}_p^\times$, it implies that $z_1, z_2, .., z_m$ also are random elements of $\mathbb{F}_p^\times$. Therefore, probability that at least one of z_i would belong to H is at least $1 - e^{\left(\frac{-dm}{p-1}\right)}$, by the collision theorem. In other words, with probability at least $1 - e^{-\frac{dm}{p-1}}$, one can compute z_i for some i, $1 \leq i \leq m$ if one has access to m threads. Since the number of steps performed on each thread before we have computed z_i for some i is at max $2\sqrt{d}$, we conclude that it takes $O(\sqrt{d})$ steps to compute x with the probability at least $1 - e^{\left(\frac{-dm}{p-1}\right)}$ if m threads are available. This completes the proof. □

Remark 4.2. It follows from Theorem 4.2 that if there exist divisors d of $p-1$ of suitable sizes, then DLP can be solved in time much less than the square root of the group size but with a probability which increases with the number of parallel threads used. A practical importance of Theorem 4.2 lies in the fact that such divisors of $p-1$ do exist for all NIST curves [48] as well as most of SEC2 curves [52]. This gives us precise estimates about the number of group operations and threads needed to solve DLP with a given probability. We illustrate this by an example in the next section.

Remark 4.3. Note that the probability of solving the DLP in above theorem is proportional to the product $m \cdot d$. It follows that if we fix a probability, this product is constant. Therefore, for a fixed probability of solving the DLP, there is a trade-off between the number of steps and number of threads needed in Theorem 4.2. Increasing one of the two would decrease the other and vice-a-versa.

4.4 Security analysis of NIST curve P-256

As discussed earlier, our probabilistic algorithm is applicable to NIST curves. In this section, we will demonstrate the implication of our algorithm on NIST curves. We will do that only on the NIST curve P-256 but similar conclusions hold for other four NIST curves over prime field as well, see appendix C.

The NIST curve P-256 is defined over the prime field $\mathbb{F}_q$ and the order of P-256 is a prime p given below.

$q = 115792089210356248762697446949407573530086143415290314195533631308867097853951$

$p = 115792089210356248762697446949407573529996955224135760342422259061068512044369$

$p - 1 = 2^4 \cdot 3 \cdot 71 \cdot 131 \cdot 373 \cdot 3407 \cdot 17449 \cdot 38189 \cdot 187019741 \cdot 622491383 \cdot 1002328039319 \cdot 26247475503338692784167739 53$

Since $p-1$ factors into many relatively small integers, we have the following divisors of $p-1$ of various sizes.

$d_1 = 53442744950329414596399414364097097310204741237882641297199829 \approx 2^{201.73}$.

$d_2 = 10688548990065882919279882872819419462040948247576528259439658 \approx 2^{202.73}$.

$d_3 = 16032823485098824378919824309229129193061422371364792389159487 \approx 2^{203.32}$.

$d_4 = 18207943204577231552993280473847881053586755339746615889955457403 \approx 2^{213.47}$.

Table 4.1

	$\log_2 d_1 = 201.73$ $\log_2(\sqrt{d_1}) = 101.86$	$\log_2 d_2 = 202.73$ $\log_2(\sqrt{d_2}) = 101.36$	$\log_2 d_3 = 203.32$ $\log_2(\sqrt{d_3}) = 101.66$
$\log_2 m = 45$	0.00162	0.00324	0.00486
$\log_2 m = 50$	0.05064	0.098711	0.14435
$\log_2 m = 52$	0.18768	**0.34013**	0.46398
$\log_2 m = 53$	**0.34013**	**0.56458**	0.71268
$\log_2 m = 54$	**0.56458**	**0.81040**	0.91745
$\log_2 m = 55$	**0.81040**	**0.96405**	0.993184
$\log_2 m = 56$	**0.96405**	0.99871	0.99995

$d_5 = 2385240559799617333442119742074072418019864949506806681584164919793 \approx 2^{220.50}$.

For above sizes of subgroups and various number of parallel threads m, the following three tables give the probability to solve DLP. The second column of the Table 4.1 shows the probabilities when the subgroup size is $d_1 \approx 2^{201.73}$ bits. For example, if we have $m = 2^{54}$ parallel threads, then our algorithm would solve DLP in $2^{101.86}$ steps with probability 0.56458 which is the intersection of the fifth row(corresponding to $m = 2^{54}$) and the second column(corresponding to $d_1 \approx 2^{201.73}$). Other entries(probabilities) of the tables can be understood similarly.

If we go across a row in the Table 4.1, we see the probabilities getting increased with the size of subgroup d. If we move along a column, probabilities increase with the number (m) of parallel threads. Table 4.1 also exhibits the trade-off between d and m for equal probability. For equal probability, highlighted diagonally in the second and third column, we see that increasing the subgroup size by 1-bit(d_1 and d_2 differ by 1-bit) results in a decrease of 1-bit in the number of parallel threads m. As an example, to achieve the probability 0.56458, the subgroup of order d_1 requires 2^{54} parallel threads while the subgroup of order d_2 requires 2^{53}.

	$\log_2 d_4 = 213.47$ $\log_2(\sqrt{d_4}) = 106.78$
$\log_2 m = 41$	0.29234
$\log_2 m = 42$	0.49921
$\log_2 m = 43$	0.74921
$\log_2 m = 44$	0.93710

Table 4.2

	$\log_2 d_5 = 220.50$ $\log_2(\sqrt{d_5}) = 110.25$
$\log_2 m = 33$	0.16218
$\log_2 m = 34$	0.29805
$\log_2 m = 35$	0.50727
$\log_2 m = 36$	0.75721
$\log_2 m = 37$	0.94106

Table 4.3

Entries of the Table 4.2 and Table 4.3 correspond to d_4 and d_5 respectively, and can similarly be understood as Table 4.1. From Table 4.3, we can see that DLP on the curve P-256 can be solved in $2^{110.25}$(with a significant reduction from 2^{128}) steps with probability greater than 0.5, while using 2^{35} parallel threads. This indicates a weakness of NIST curve P-256 if one assumes that 2^{35} parallel threads are within the reach of modern distributed computing. Similar conclusions can be drawn for other NIST curves P-192, P-224, P-384 and P-521.

Moreover, most of the curves in SEC2(version 2) standard [52] which also include all other ten NIST curves [48]over binary field, $p-1$ factors into small divisors. Therefore, our algorithm for solving DLP on those curves in SEC2 [52] can *similarly* be applied.

Lastly, we understand that our result is not as good as parallelized version [56] of Pollard's rho attack, and in practice it may be impractical or impossible to have 2^{35} parallel threads. However, the most important thing about our algorithm is the novel approach to attack DLP and it certainly shows us a way how to use parallelization to increase the probability so that Theorem 4.1 can then be applied successfully.

4.5 Conclusion

In this chapter, we presented a novel idea of using the implicit representation with $\mathbb{F}_p^\times$ as auxiliary group to solve the discrete logarithm problem in a group $\mathbb{G}$ of prime order p. We modified the most common generic algorithm, the baby-step giant-step algorithm for this purpose and studied it further for NIST curves over prime fields which indicates some security issues in the NIST curves. Moreover, this algorithm that we developed brings to the spotlight the structure of the auxiliary group for the security of the discrete logarithm problem in $\mathbb{G}$ of prime order p. Even though our algorithm does not match the best known attacks using parallelization, it does present a novel approach to attack DLP and this aspect is probably reported for the first time. We hope that like most of the new ideas, this would lead to more useful and efficient attacks on the discrete logarithm problem.

Appendices

Appendix A

Elliptic curve domain parameters over prime field

The following data present several SECG curves[52] which are defined over some prime field of characteristic not equal to 2 and are used for practical purposes. For these curves, prime p denotes the order of the elliptic curve group and d is the suitable divisor of $p-1$ which is used by us for various computation in Table 3.1 of chapter 3.

A.1 SECP112R1

$p = 4451685225093714776491891542548933$

$d = 140876$

A.2 SECP112R2

$p = 1112921306273428674967732714786891$

$d = 110852811870$

A.3 SECP128R1

$p = 340282366762482138443322565580356624661$

$d = 9476076960994$

A.4 SECP128R2

$p = 85070591690620534603955721926813660579$

$d = 3101689558$

A.5 SECP160K1

$p = 1461501637330902918203686915170869725397159163571$

$d = 42918291593381467397$

A.6 SECP160R1

$p = 1461501637330902918203687197606826779884643492439$

$d = 22167198845997443$

A.7 SECP160R2

$p = 1461501637330902918203685083571792140653176136043$

$d = 142004808588765074419$

A.8 SECP192K1

$p = 6277101735386680763835789423061264271957123915200845512077$

$d = 43818996$

A.9 SECP192R1

$p = 6277101735386680763835789423176059013767194773182842284081$

$d = 95646823139138600591956 69$

A.10 SECP224K1

$p = 26959946667150639794667015087019640346510327083120074548994958668279$

Appropriate size of divisor d of $p - 1$ not available

A.11 SECP224R1

$p = 26959946667150639794667015087019625940457807714424391721682722368061$

$d = 533642580$

A.12 SECP256K1

$p = 11579208923731619542357098500868790785283756427907490438260516314151816149433 7$

$d = 65709355417112419152054124$

A.13 SECP256R1

$p = 11579208921035624876269744694940757352999695522413576034242225906106851204436 9$

$d = 71482998987075857096374359$

A.14 SECP384R1

$p = 39402006196394479212279040100143613805079739270465446667946905279627659399113263569398956308152294913554433653942643$

$d = 128955808797897620607830395927026$

A.15 SECP521R1

$p = 6864797660130609714981900799081393217269435300143305409394463459185543183397655394245057746333217197532963996371363321113864768612440380340372808892707005449$

$d = 1898873518475180724503002533770555108536$

Appendix B

Elliptic curve domain parameters over $\mathbb{F}_{2^m}$

The following data present several SECG curves[52] which are defined over a binary field and are used for practical purposes. For these curves, prime p is the largest divisor of the order of that particular elliptic curve group(with a very small co-factor of either 2 or 4) and d is the appropriate divisor of $p-1$ used by us for various computation in Table 3.2 of chapter 3.

B.1 SECT113R1

$p = 5192296858534827689835882578830703$

$d = 253877289037$

B.2 SECT113R2

$p = 5192296858534827702972497909952403$

$d = 215851796187$

B.3 SECT131R1

$p = 136112946768375385389393275568536556065 3$

$d = 23348$

B.4 SECT131R2

$p = 136112946768375385387953504341281286798 3$

$d = 485524729$

B.5 SECT163K1

$p = 5846006549323611672814741753598448348329118574063$

$d = 33118034411893094$

B.6 SECT163R1

$p = 5846006549323611672814738465098798981304420411291$

$d = 27744064547201903$

B.7 SECT163R2

$p = 5846006549323611672814742442876390689256843201587$

$d = 859825042$

B.8 SECT193R1

$p = 627710173538668076383578942326954805369157518605104019719 3$

$d = 1697589986603916123127$

B.9 SECT193R2

$p = 6277101735386680763835789423314955362437298222279840143829$

$d = 4345632155805272808276901$

B.10 SECT233K1

$p = 3450873173395281893717377931138512760570940988862252126328087024741343$

$d = 11064269030135607689238$

B.11 SECT233R1

$p = 6901746346790563787434755862277025555839812737345013555379383634485463$

$d = 443484653691663066996649$

B.12 SECT239K1

$p = 2208558830972980411979121875928648149482165613217098488874802192153622 13$

$d = 912013207122974008798076$

B.13 SECT283K1

$p = 3885337784451458141838923813647037813284811733793061324295874997529815829704422603873$

$d = 19578145037471479248182334822$

B.14 SECT283R1

p = 77706755689029162836778476272940756265696259243769048891091965267700442777873786928 71

d = 34107744933314238426752172695

B.15 SECT409K1

p = 33052798439512429947595765401638551991420234148214060964232439502288071128924919105067325845777745801409636659061773135867 1

d = 57244322287026111360919333305789 0

B.16 SECT409R1

p = 66105596879024859895191530803277103982840468296428121928464879830415777482737480520814372376217911096597986728836656752677 1

d = 13303514230748105710830031415444654372433 8

B.17 SECT571K1

p = 19322687615086291723476759454659936721494636648532174993286176257257595711447802122681339785227067118347067128008253514612736749740666173119296824216170925035555733685276673

d = 16508360322752105262554680590633369145542494978266766319 16

B.18 SECT571R1

$p = 38645375230172583446953518909319873442989273297064349986572352514515191422895604245361439993894157730831338811219269444862468724628168130702345282883033324113931911052857034$

$d = 21606773965882205526514379463389966056990432774077550969190$

Appendix C

NIST Curves Over Prime Field

For each of these five NIST curves of order prime p, two subgroups of $\mathbb{F}_p^\times$ with (large enough)orders d_1, d_2 are given such that $d_1 \cdot d_2 = p - 1$ and $\gcd(d_1, d_2) = 1$, see Remark 4.1.

C.1 P-192

$p = 6277101735386680763835789423176059013767194773182842284081$

$p - 1 = 2^4 \cdot 5 \cdot 2389 \cdot 956468231391386005919566 9 \cdot 3433859179316188$
682119986911
$d_1 = 65627916635090998092677189843032 0 \approx 2^{109.02}$
$d_2 = 95646823139138600591956 69 \approx 2^{82.98}$

C.2 P-224

$p = 26959946667150639794667015087019625940457807714424391721682722368061$

$p - 1 = 2^2 \cdot 3^6 \cdot 5 \cdot 2153 \cdot 5052060625887581870$

74708601532876667009176960999333893515 07

$d_1 = 505206062588758187074708601532876667009176960999333893515 07 \approx 2^{195.01}$

$d_2 = 533642580 \approx 2^{28.99}$

C.3 P-256

$p = 115792089210356248762697446949407573529996955224135760342422$
259061068512044369

$p - 1 = 2^4 \cdot 3 \cdot 71 \cdot 131 \cdot 373 \cdot 3407 \cdot 17449 \cdot 38189 \cdot 187019741 \cdot 622491383 \cdot$
$1002328039319 \cdot 26247475503338692784167739 53$

$d_1 = 14891532244080672251707533164156494935 84 \approx 2^{130.13}$
$d_2 = 77757001302792844776776389119582520177 \approx 2^{125.87}$

C.4 P-384

$p = 394020061963944792122790401001436138050797392704654466679 4$
$690527962765939911326356939895630815229491355443365394264 3$

$p - 1 = 2 \cdot 3^2 \cdot 7^2 \cdot 13 \cdot 1124679999981664229965379347 \cdot$
$305546578814035200273394690614456109064124960616040788436539197970492 9$
268480326390471

$d_1 = 116779902422724253544491450752845124884308559947450789340445281 4$
$643223966413180746438016 2 \approx 2^{292.55}$
$d_2 = 1124679999981664229965379347 \approx 2^{89.86}$

C.5 P-521

$p = 6864797660130609714981900799081393217269435300143305409394463$
$4591855431833976553942450577463332171975329639963713633211138647 6$
$8612440380340372808892707005449$

$p - 1 = 2^3 \cdot 7 \cdot 11 \cdot 1283 \cdot 1458105463 \cdot 164778191592198069046859 9 \cdot$
$3615194794881930010216942559103847593050265703173292383701371712350878926821$
$6612437559338354268960584185097598801719 43$

$d_1 = 4166083869350854498586791068944823620942931357552596820305098954973$
$6942712923152533496543294196006831576365431086302108142568219817 52 \approx$
$2^{440.55}$
$d_2 = 164778191592198069046859 9 \approx 2^{80.45}$

Bibliography

[1] X9.62: Public key cryptography for the financial services industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). 1999.

[2] Riddhipratim Basu, Shirshendu Ganguly, Subhamoy Maitra, and Goutam Paul. A complete characterization of the evolution of RC4 pseudo random generation algorithm. *Journal of Mathematical Cryptology*, 2(3):257–289, 2008.

[3] Kamel Bentahar. The equivalence between the DHP and DLP for elliptic curves used in practical applications, revisited. In *Cryptography and Coding*, pages 376–391. Springer, 2005.

[4] Daniel J Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted Edwards curves. In *International Conference on Cryptology in Africa*, pages 389–405. Springer, 2008.

[5] Daniel J Bernstein and Tanja Lange. Computing small discrete logarithms faster. In *International Conference on Cryptology in India*, pages 317–338. Springer, 2012.

[6] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532. Springer, 2001.

[7] Joppe W Bos, J Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig, and Eric Wustrow. Elliptic curve cryptography in practice. In *International Conference on Financial Cryptography and Data Security*, pages 157–175. Springer, 2014.

[8] Daniel RL Brown and Robert P Gallant. The static Diffie-Hellman problem. *IACR Cryptology ePrint Archive*, 2004:306, 2004.

[9] M Chateauneuf, Alan CH Ling, and Douglas R Stinson. Slope packings and coverings, and generic algorithms for the discrete logarithm problem. *Journal of Combinatorial Designs*, 11(1):36–50, 2003.

[10] Sanjit Chatterjee and Palash Sarkar. *Identity-based encryption*. Springer Science & Business Media, 2011.

[11] Jung Hee Cheon. Security analysis of the strong Diffie-Hellman problem. In *Advances in Cryptology-EUROCRYPT 2006*, pages 1–11. Springer, 2006.

[12] Jung Hee Cheon. Discrete logarithm problems with auxiliary inputs. *Journal of Cryptology*, 23(3):457–476, 2010.

[13] H. Cohen. *A course in computational algebraic number theory*. Springer-Verlag, GTM 138, 1993.

[14] Henri Cohen. *A course in computational algebraic number theory*, volume 138. Springer Science & Business Media, 2013.

[15] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC press, 2005.

[16] Claus Diem. On the discrete logarithm problem in elliptic curves. *Compositio Mathematica*, 147(01):75–104, 2011.

[17] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[18] Ratna Dutta, Rana Barua, and Palash Sarkar. Pairing-based cryptographic protocols: A survey. *IACR Cryptology ePrint Archive*, 2004:64, 2004.

[19] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[20] Gerhard Frey. Applications of arithmetical geometry to cryptographic constructions. In *Finite Fields and Applications*, pages 128–161. Springer, 2001.

[21] Gerhard Frey. *On the relation between Brauer groups and discrete logarithms*. Citeseer, 2005.

[22] Steven Galbraith. Online notes "algorithms of the ECDLP". *See https://www.math.auckland.ac.nz/ sgal018/talks.html*, pages 6–8, 2015.

[23] Steven D Galbraith. Constructing isogenies between elliptic curves over finite fields. *LMS Journal of Computation and Mathematics*, 2:118–138, 1999.

[24] Steven D Galbraith and Pierrick Gaudry. Recent progress on the elliptic curve discrete logarithm problem. *Designs, Codes and Cryptography*, 78(1):51–72, 2016.

[25] Steven D Galbraith and Shishay W Gebregiyorgis. Summation polynomial algorithms for elliptic curves in characteristic two. In *International Conference in Cryptology in India*, pages 409–427. Springer, 2014.

[26] Steven D Galbraith, Florian Hess, and Nigel P Smart. Extending the GHS Weil descent attack. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 29–44. Springer, 2002.

[27] Steven D Galbraith, Ping Wang, and Fangguo Zhang. Computing elliptic curve discrete logarithms with improved baby-step giant-step algorithm. *IACR Cryptology ePrint Archive*, 2015:605, 2015.

[28] Robert Gallant, Robert Lambert, and Scott Vanstone. Improving the parallelized Pollard lambda search on anomalous binary curves. *Mathematics of Computation*, 69(232):1699–1705, 2000.

[29] Pierrick Gaudry. Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem. *Journal of Symbolic Computation*, 44(12):1690–1702, 2009.

[30] Jeffrey Hoffstein, Jill Pipher, Joseph H Silverman, and Joseph H Silverman. *An Introduction to Mathematical Cryptography*. Springer, 2008.

[31] Ming-Deh Huang and Wayne Raskind. Global duality, signature calculus and the discrete logarithm problem. *LMS Journal of Computation and Mathematics*, 12:228–263, 2009.

[32] Michael J Jacobson, Neal Koblitz, Joseph H Silverman, Andreas Stein, and Edlyn Teske. Analysis of the xedni calculus attack. *Designs, Codes and Cryptography*, 20(1):41–64, 2000.

[33] Don Johnson, Alfred Menezes, and Scott Vanstone. The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security*, 1(1):36–63, 2001.

[34] Antoine Joux. A one round protocol for tripartite diffie–hellman. In *International Algorithmic Number Theory Symposium*, pages 385–393. Springer, 2000.

[35] Neal Koblitz and Alfred Menezes. A riddle wrapped in an enigma. *IACR Cryptology ePrint Archive*, 2015:1018, 2015.

[36] Shunji Kozaki, Taketeru Kutsuma, and Kazuto Matsuo. Remarks on Cheon's algorithms for pairing-related problems. In *Pairing-Based Cryptography–Pairing 2007*, pages 302–316. Springer, 2007.

[37] Prabhat Kushwaha. Improved Lower Bound on DHP: Towards the Equivalence of DHP and DLP for Important Elliptic Curves Used for Implementation. *ArXiv e-prints, 1610.01354*, October 2016.

[38] Prabhat Kushwaha. Towards the equivalence of Diffie-Hellman problem and Discrete Logarithm problem for Important Elliptic Curves Used in Practice. In *IEEE ISEA Asia Security and Privacy Conference 2017*, pages 9–12, 2017.

[39] Ueli M Maurer. Towards the equivalence of breaking the Diffie-Hellman protocol and computing discrete logarithms. In *Advances in Cryptology-CRYPTO'94*, pages 271–281. Springer LNCS 839, 1994.

[40] Ueli M Maurer and S Wolf. On the difficulty of breaking the Diffie-Hellman protocol. Technical report, 24, Department of Computer Science, ETH Zurich, 1996.

[41] Ueli M Maurer and Stefan Wolf. Diffie-Hellman oracles. In *Advances in Cryptology - Crypto 96*, pages 268–282. Springer LNCS 1109, 1996.

[42] Ueli M Maurer and Stefan Wolf. The relationship between breaking the Diffie–Hellman protocol and computing discrete logarithms. *SIAM Journal on Computing*, 28(5):1689–1721, 1999.

[43] Ueli M Maurer and Stefan Wolf. The Diffie–Hellman Protocol. *Designs, Codes and Cryptography*, 19:147–171, 2000.

[44] Alfred J Menezes, Tatsuaki Okamoto, and Scott A Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39(5):1639–1646, 1993.

[45] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of Applied Cryptography*. CRC press, 1996.

[46] Peter L Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, 1987.

[47] Antoine Muzereau, Nigel Smart, and Frederik Vercauteren. The equivalence between the DHP and DLP for elliptic curves used in practical applications. *LMS Journal of Computation and Mathematics*, 7:50–72, 2004.

[48] FIPS NIST. 186.2 Digital Signature Standard (DSS). *National Institute of Standards and Technology (NIST)*, 2000.

[49] John M Pollard. Monte Carlo methods for index computation (mod p). *Mathematics of Computation*, 32(143):918–924, 1978.

[50] John M Pollard. Kangaroos, monopoly and discrete logarithms. *Journal of Cryptology*, 13(4):437–447, 2000.

[51] H.-G. Rück. A note on elliptic curves over finite fields. *Mathematics of Computation*, 49:301–304, 1987.

[52] SECG. SEC 2. : Recommended Elliptic Curve Domain Parameters. *See http://www. secg.org/, 2000.*

[53] Igor Semaev. Summation polynomials and the discrete logarithm problem on elliptic curves. *IACR Cryptology ePrint Archive*, 2004:31, 2004.

[54] Daniel Shanks. Class number, a theory of factorization, and genera. In *Proc. Symp. Pure Math*, volume 20, pages 415–440, 1971.

[55] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Advances in Cryptology EUROCRYPT'97*, pages 256–266. Springer, 1997.

[56] Paul C Van Oorschot and Michael J Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1–28, 1999.

[57] Michael J Wiener and Robert J Zuccherato. Faster attacks on elliptic curve cryptosystems. In *International Workshop on Selected Areas in Cryptography*, pages 190–200. Springer, 1998.