Elliptic Curve Cryptosystems: A Survey

Tallman Zacharia Nkgau
School of Computer Science
McGill University, Montreal
August 1998

A Thesis Submitted to the Faculty of Graduate Studies and
Research in partial fulfilment of the requirements of the degree of
Master of Science in Computer Science
Copyright ©1998 Tallman Zacharia Nkgau
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
Abstract

Elliptic curves have been a subject of much mathematical study since early in the past century. Recently, through the work of Koblitz and Miller, they have found application in the area of public-key cryptography. The basic reason is that, elliptic curves over finite fields provide an abundance of finite abelian groups which could be used as a basis for public-key cryptosystems. The objective of this thesis is to survey the field of elliptic curve public-key cryptography as it exists now, with an attempt to identify key ideas and contributions. We are particularly interested in elliptic curve cryptosystems defined over $\mathbb{Z}_p$ ($p > 3$ and prime) and the ring $\mathbb{Z}_n$ ($n$ is a product of two large distinct primes).
Résumé

Beaucoup d'attention a déjà été porté sur l'étude mathématique des courbes elliptiques durant les siècles passés. Récemment, grâce aux travaux de Koblitz et Miller, des applications dans le domaine de la cryptographie basée sur les clés publiques ont été trouvées. La raison principale de ces avancées est la facilité des courbes elliptiques basées sur des corps finis à fournir en quantité des groupes abéliens finis qui peuvent être à la base de système de cryptage à clé publique. Le but de cette thèse est de passer en revue l'état actuel de la connaissance dans le domaine de la cryptographie à clé publique basée sur les courbes elliptiques et d'en faire ressortir les idées et les contributions majeures. Nous porterons tout particulièrement notre attention sur les systèmes de cryptage basés sur les courbes elliptiques définies sur $\mathbb{Z}_p$ ($p > 3$ et $p$ est un nombre premier) et l'anneau $\mathbb{Z}_n$ ($n$ étant un produit de deux grands nombres premiers qui ne sont pas égaux).
Acknowledgements

Many thanks go to my supervisor, David Avis, for his patience, guidance, and motivation throughout the writing of this thesis. Special thanks to Nigel Smart for sending me his paper on anomalous curves, Kenji Koyama for helpful suggestions, and Chris Caldwell for sending me a list of prime (proved prime) for my experiments, and to Chrislain Razafimahefa for translating the abstract to French and for being such a friend. I am also grateful to Steve Robbins, Kenji Imasaki, Ian Garton, Vijay Sundaresan, Mike Soss, and Prasad Kakulavarapu for finding time to hang out and make McGill and Montreal the place to be. Also, thanks to Xiaoming Zhong and Ling Yong for all those Sunday morning dim sums. Lastly, a salute to a young man, Ame Louis Tebogo-Nkgau, for all the inspiration, patience, and understanding all the years I have been away.
Contents

1 Introduction

2 Mathematical Preliminaries
   2.1 Number Theory ........................................ 5
   2.2 Algorithms and their Complexity ....................... 7
   2.3 More Number Theory ..................................... 9
   2.4 Group and Field Theory ................................ 14
   2.5 Core Number-Theoretic Algorithms ..................... 24
   2.6 Public-Key Cryptography ................................ 28
      2.6.1 RSA ............................................. 30
      2.6.2 Attacks on RSA .................................... 32
      2.6.3 ElGamal .......................................... 33
      2.6.4 Attacks on ElGamal ................................ 34
      2.6.5 The Diffie-Hellman Key Exchange Protocol ........ 34
      2.6.6 Attacks on the Diffie-Hellman Key Exchange .... 37

3 Elliptic Curves
   3.1 Introduction to Elliptic Curves ......................... 38
   3.2 Addition Rule .......................................... 39
   3.3 Computing the number of points on an elliptic curve, \#E . 44
   3.4 The Elliptic Curve Discrete Logarithm Problem (ECDLP) . 46
   3.5 Elliptic Curves Over \(\mathbb{Z}_n\) ......................... 48
CONTENTS

4 Elliptic Curve Cryptosystems 51
   4.1 Elliptic Curve Cryptosystems Over $\mathbb{Z}_p$ ....................... 51
   4.2 Elliptic Curve Cryptosystems Over $\mathbb{Z}_n$ ....................... 58
   4.3 Attacks on Elliptic Curve Cryptosystems ......................... 61
   4.4 Speeding Up Elliptic Curve Computations ....................... 62
   4.5 Generating Computationally Secure Curves ..................... 69

5 Conclusion 72
List of Figures

2.1 Algorithm EUCLID ..................................................... 10
2.2 Algorithm EXTENDED-EUCLID ...................................... 13
2.3 Algorithm MOD-EXP .................................................. 18
2.4 Decision Problem QUADRATIC-RESIDUE ................................. 19
2.5 Decision Problem COMPOSITE ......................................... 21
2.6 Algorithm MILLER-RABIN ............................................. 23
2.7 Decision Problem PRIMITIVE ......................................... 24
2.8 Problem: FACTORING .................................................. 24
2.9 The Discrete Logarithm Problem (DLP) .................................. 25
2.10 Algorithm SQUARE-ROOT(p) ........................................... 27
2.11 Computing Square Roots Modulo n ..................................... 28
2.12 The RSA Public-Key Cryptosystem ..................................... 31
2.13 The ElGamal Public-Key Cryptosystem .................................. 33
2.14 Diffie-Hellman Key Exchange Protocol .................................. 35
2.15 The Diffie-Hellman problem ............................................. 35

3.1 Geometric description of adding two distinct elliptic curve points .......... 40
3.2 Geometric description of doubling an elliptic curve point ................. 41
3.3 Algorithm MULTIPLE-ADDITION ........................................ 47
3.4 The Elliptic Curve Discrete Logarithm Problem (ECDLP) ..................... 48

4.1 The Analog of the ElGamal Public Key Cryptosystem .......................... 52
4.2 The Menezes-Vanstone Cryptosystem ..................................... 54
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3</td>
<td>Menezes-Vanstone encryption</td>
<td>55</td>
</tr>
<tr>
<td>4.4</td>
<td>Menezes-Vanstone decryption</td>
<td>56</td>
</tr>
<tr>
<td>4.5</td>
<td>The Elliptic Curve Analog of the Diffie-Hellman Key Exchange Protocol</td>
<td>57</td>
</tr>
<tr>
<td>4.6</td>
<td>The KMOV-scheme</td>
<td>59</td>
</tr>
<tr>
<td>4.7</td>
<td>The Meyer et al. Scheme</td>
<td>60</td>
</tr>
<tr>
<td>4.8</td>
<td>Three speed up levels for elliptic curve operations</td>
<td>63</td>
</tr>
<tr>
<td>4.9</td>
<td>The modified signed-digit (MSD) algorithm for computing ( kP )</td>
<td>65</td>
</tr>
<tr>
<td>4.10</td>
<td>Jedwab and Mitchell minimum MSD algorithm</td>
<td>67</td>
</tr>
<tr>
<td>4.11</td>
<td>Laih and Kuo's algorithm for computing ( kP )</td>
<td>68</td>
</tr>
<tr>
<td>4.12</td>
<td>Koblitz's random curve selection method, RANDOM-CURVE</td>
<td>70</td>
</tr>
<tr>
<td>4.13</td>
<td>Koblitz's algorithm for generating curves of prime order, RANDOM-CURVE-PRIME</td>
<td>70</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Time complexities of primitive arithmetic operations. ........................................ 9
2.2 Factoring Algorithms .................................................................................. 25
2.3 Algorithms for solving DISCRETE-LOG ..................................................... 26
3.1 Points on the curve $E : y^2 = x^3 + x + 19$ over $\mathbb{Z}_{23}$ ......................... 43
3.2 Multiples of $P = (4, 8)$ ........................................................................... 44
3.3 Notational correspondence between $\mathbb{Z}_p^*$ and $E_p$ ................................. 47
3.4 Points on the curve $E : y^2 = x^3 + x + 12$ over $\mathbb{Z}_{35}$ ......................... 49
Chapter 1

Introduction

Cryptography is the art and science of communicating in the presence of adversaries. Two people, we will call Bob and Alice in keeping with tradition, wish to communicate privately, so that an adversary, Oscar, cannot know what was communicated. Bob and Alice have to send messages to each over some communication channel, which could either be a telephone line, a computer network or any other feasible means of communication. Cryptography, for a long time has always been within the realms of governments and the military. This changed in 1976 with the seminal paper of Diffie-Hellman [19], which introduced cryptography to the world of academia. One solution to this need for communicating privately is for Bob and Alice to exchange a secret key, $e_K$. Now, when Bob wishes to send a message $m$ (called “plaintext”) to Alice, he sends $c = e_K(m)$. The effect of computing $e_K(m)$ is to transform the original message into “ciphertext” (something unintelligible). Alice retrieves the message by computing $m = e^{-1}_K(e_K(m))$. As of necessity, the secret key could be thought of as a one-to-one function from the plaintext space to the ciphertext space. Computing $e_K$ and $e^{-1}_K$ is referred to as “encryption” and “decryption”, respectively. This scheme of communication is usually referred to as a “private-key cryptosystem”. Private-key cryptosystems, however, have two drawbacks:

1. The secret key $e_K$ must be communicated over a secure channel, which might not always be available.

2. In a network of $n$ users, $\frac{n(n-1)}{2}$ secret keys could be required. Privacy of keys could easily be
compromised.

To overcome these deficiencies, Diffie and Hellman [19] proposed the use of “public-key cryptosystems”. In these schemes, each user’s encryption key is put in a public directory of keys (like phone numbers). Anybody can send a message to Bob by using his public key, say $e_B$. Of course, this implies that $e_B$ should be computationally infeasible to invert at any point of its domain, unless you have some special information (the “trap-door”), which makes it easy to invert $e_B$. The trapdoor should only be known by Bob. Encryption functions satisfying these conditions are known as “trap-door one-way functions” (TOF), and it was Diffie and Hellman who first proposed their use in cryptography.

In their 1976 paper, Diffie and Hellman did not propose any public-key cryptosystem, this task was left to another seminal paper, this time of Rivest, Shamir, and Adleman [69]. They proposed a public-key cryptosystem, which now bears their names, RSA, based on factoring a large composite integer. This now famous cryptosystem goes by the name of RSA. Soon after, there was an explosion of research in to public-key cryptography. A lot of cryptosystems were proposed and many were broken [84, 73]. Amongst all these cryptosystems, the one proposed by ElGamal [20], based on exponentiation in finite fields stood out from the rest. Exponentiation in finite fields and factoring large composite numbers are now the basis of many public-key cryptosystems.

The use of elliptic curves (in particular, the theory of elliptic curves defined over finite fields) in public-key cryptography was discovered, independently, by Koblitz [29] and Miller [56] barely a decade ago. Elliptic curves, in contrast, have been a subject of much mathematical study since the dawn of the 19th century. Miller and Koblitz observed that elliptic curves defined over finite fields provide an abundance of finite abelian groups. Some cryptosystems, for example, the ElGamal cryptosystem, are based on multiplicative groups of finite fields, and in many ways elliptic curves are natural analogs of these groups. However, elliptic curves offer several advantages over their finite field counterparts:

- More flexibility in choosing an elliptic curve than in choosing a finite field.
- Elliptic curve public-key cryptosystems (over $\mathbb{Z}_p(p > 3$ and prime )) offer the highest strength-per-bit of any public-key cryptosystem.
Only elementary arithmetic operations are required, leading to faster implementations in hardware and software, especially for elliptic curves over \( \mathbb{F}_2^n \).

The main disadvantage in implementing elliptic curve cryptosystems is usually the fact that a thorough knowledge of the theory of elliptic curves over finite fields is required. A daunting task indeed.

The objective of this thesis is to survey the field of elliptic curve cryptography as it exists now, with an attempt to identify key ideas and contributions. We are particularly interested in elliptic curves cryptosystems defined over \( \mathbb{Z}_p(p > 3 \text{ and prime}) \) and the ring \( \mathbb{Z}_n \) (\( n \) is a product of two large distinct primes). For elliptic curve cryptosystems over \( \mathbb{F}_2^n \), the reader is referred to Menezes [52].

A survey like this can not hope to contain all the important results in the field, and some results will only be glossed over. However, we hope that what we have included is enough to please everybody.

So far, we only know of two surveys that have been published, Menezes [52] and Saeki [71]. Menezes' survey was published in 1993 and with the rate at which research is going at now, it is now being eclipsed by new results in the field. In fact, some of the curves that were proposed at that time for use in cryptosystems have now been found not to be that secure [83]. Also, the emphasis is on curves over \( \mathbb{F}_2^n \) and it is more inclined to the theory of elliptic curves. Saeki's (1996) survey does not take in to account many results that were published around that time. It also only discusses the analog of the ElGamal cryptosystem. It is our opinion that the two surveys fall short of making the field accessible (either because of a leaning towards theory or insufficient details) to senior undergraduates of computer science and do not, for example, adequately discuss the speeding up of elliptic curve operations, which is of paramount importance. It is our intention, with this survey, to address this inadequacies. As such, our survey plows a fine line between theory and practical implementation so as to make it accessible to the intended audience and at the same time provide sufficient details enabling it to be used in an undergraduate course in cryptography.

We begin by reviewing the mathematics underlying the cryptosystems discussed in this thesis. A level of advanced undergraduate in computer science is assumed of readers, however, a certain amount of mathematical maturity in handling mathematical concepts is desirable. We also give a brief introduction to the field of public-key cryptography. Chapter 2 introduces elliptic curves and their properties necessary for a full understanding of the elliptic curve cryptosystems discussed in Chapter 4 of this thesis. Chapter 4 also gives summaries of attacks on elliptic curve cryptosystems.
as well as how to generate elliptic curves suitable for use in cryptosystems and how to speed up elliptic curve arithmetic operations. We conclude by summarizing recent results and looking into the future of elliptic curve cryptosystems in the field of public-key cryptography.
Chapter 2

Mathematical Preliminaries

Elliptic curve public key cryptography depends heavily on number-theoretic concepts. In this chapter we introduce the necessary mathematical concepts used in this thesis. Specifically, we give a brief overview of concepts from number, group and field theories and the arithmetic of elliptic curves. Burton [11], Cohen [14], and Koblitz [29] have a more extensive review of elementary number theory. For field and group theory, Roman [70] is an excellent source. We also give a brief review of public-key cryptography. The proofs of most of the results in this chapter are omitted since they are now considered folklore and can be found in the references cited.

2.1 Number Theory

The set of all integers will be denoted by \( \mathbb{Z} \). \( \mathbb{N}_i \) will denote the set of integers greater than \( i \), that is, \( \{i+1, i+2, \ldots\} \); \( \mathbb{N}_0 \), in this case, will then represent the positive integers. The cardinality of a set \( S \) will be denoted by \( |S| \).

Definition An equivalence relation on a set \( S \) is a binary relation \( \sim \) on \( S \) such that for any \( x, y, z \in S \), the following is true:

1. \( x \sim x \) (reflexivity)
2. if \( x \sim y \) then \( y \sim x \) (symmetry)
3. if \( x \sim y \) and \( y \sim z \) then \( x \sim z \) (transitivity)
Let ~ be an equivalence relation on a set S. Then \( P = \{ [s] \mid s \in S \} \) where \( [s] = \{ u \in S \mid s \sim u \} \), form a partition of S in the sense that,

1. for each \( X \in P \), \( X \neq \emptyset \)
2. if \( X, Y \in P \), then \( X = Y \) or \( X \cap Y = \emptyset \)
3. \( \bigcup_{X \in P} X = S \)

An element \( X \in P \) is called an equivalence class of the partition \( P \). It is assumed that the reader is familiar with many elementary facts about integers; one of which is the Well-Ordering Principle, which states that, every non-empty set \( S \) of nonnegative integers (\( \mathbb{N}_{\geq 1} \)) contains a least element; that is, there is some integer \( u \in S \) such that \( u \leq s \) for all \( s \in S \).

**Theorem 2.1.1 (Division Algorithm)**

Let \( a, b \in \mathbb{Z}, b > 0 \), then there exist unique \( q, r \in \mathbb{Z} \) such that

\[
a = q \cdot b + r, \quad 0 \leq r < b
\]

The integers \( q \) and \( r \) are called the quotient and the remainder in the division of \( a \) and \( b \), respectively.

**Corollary 2.1.2** if \( a, b \in \mathbb{Z}, b \neq 0 \), then there exist unique \( q, r \in \mathbb{Z} \) such that \( a = q \cdot b + r \),

\[
0 \leq r < |b|.
\]

If \( r = 0 \), we say \( b \) divides \( a \), and we denote this by \( b \mid a \). Otherwise, we say \( b \) does not divide \( a \), which we denote by \( b \nmid a \).

**Definition** Let \( a, b \in \mathbb{Z}, |a| + |b| \neq 0 \). The greatest common divisor of \( a \) and \( b \), denoted by \( \text{GCD}(a, b) \), is the positive integer \( d \) satisfying

1. \( d \mid a \) and \( d \mid b \).
2. if \( c \mid a \) and \( c \mid b \), then \( c \leq d \).

We should point out at this point that the greatest common divisor of any collection of integers, not all zero, always exists.
CHAPTER 2. MATHEMATICAL PRELIMINARIES

Theorem 2.1.3 (Extended Division Algorithm)
Given \(a, b \in \mathbb{Z}, |a| + |b| \neq 0\), there exists \(x, y \in \mathbb{Z}\) such that \(\text{GCD}(a, b) = a \cdot x + b \cdot y\).

Definition An integer \(n > 1\) is called a prime number if its only positive divisors are 1 and \(n\), otherwise it is called composite.

Definition An integer \(n\) is said to be \(y\)-smooth if all its prime factors are less or equal to \(y\).

From this point onwards, we will follow the convention that all variables are integers unless otherwise specified.

Theorem 2.1.4 If \(a|b \cdot c\) and \(\text{GCD}(a, b) = 1\), then \(a|c\).

Corollary 2.1.5 If \(p\) is prime and \(p|a \cdot b\), then \(p|a\) or \(p|b\).

Definition Let \(a, b \in \mathbb{Z}, |a| + |b| \neq 0\), then \(a\) and \(b\) are said to be relatively prime whenever \(\text{GCD}(a, b) = 1\).

The following theorem finally gives us the ammunition we require to calculate the greatest common divisor of two integers and can be easily be extended to a collection of integers, not all zero.

Theorem 2.1.6 Let \(a, b \in \mathbb{Z}\). If \(a = q \cdot b + r\) for some \(q \in \mathbb{Z}\), then \(\text{GCD}(a, b) = \text{GCD}(b, r)\).

Before giving any algorithms, we regress to give some definitions and notation we shall use in discussing algorithms. More detailed information on algorithms can be found in Cormen et al. [17], Rawlins [67], Aho et al. [3], and Wilf [91].

2.2 Algorithms and their Complexity

An algorithm, informally, is a list of steps for solving a problem. The following properties of an algorithm are usually of interest:

- Time Complexity, \(T(n)\)
- Space Complexity, \(S(n)\)
The argument $n$ is the size of the input to the algorithm.

**Definition (Big-Oh, $O$)**

Let $f$ and $g$ be two functions $f, g : \mathbb{N}_0 \rightarrow \mathbb{R}^+$. We say $f(n) = O(g(n))$ if there exists $c, n_0 \in \mathbb{N}_0$ such that $f(n) \leq cg(n)$ for all $n \geq n_0$. When $f(n) = O(g(n))$ we say that $g(n)$ is an (asymptotic) upper bound for $f(n)$.

**Definition (Small-oh, $o$)**

Let $f$ and $g$ be two functions $f, g : \mathbb{N}_0 \rightarrow \mathbb{R}^+$. We say $f(n) = o(g(n))$ if $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$.

**Definition** We say the running time of an algorithm is *polynomial* if its time complexity $T(n) = O(n^c)$, for a constant $c \geq 1$ (if $c = 1$ we usually say it is *linear* and if $c = 2$, we say it is *quadratic*); and we say it is *exponential* if $T(n) = O(r^n)$, for a constant $r > 1$.

We are mostly interested in the asymptotic behavior of $T(n)$, that is, its limiting behavior as the input size of the algorithm increases ($n \rightarrow \infty$). The functions $T(n)$ and $S(n)$ depends heavily on the computation model used and on the input encoding scheme used. Fortunately, it turns out that all "reasonable" input encoding schemes differ at most polynomially from one another. The input to our algorithms will exclusively be integers, hence their binary equivalents will be used as input to the algorithms. The "size" of the input will then be the number of binary digits used to encode the input.

Since we will be working with rather "large" integers, the following bounds, given in Table 2.1, for the primitive arithmetic operations will be used; where the input is two $k$-bit integers. Certainly better algorithms, with better time complexity bounds can be used, but these bounds suffice for us.

We will, on numerous occasions discuss algorithms that depend only on the random random choices they make. These kinds of algorithms are usually referred to as randomized algorithms. The randomness usually arises from the assumption that the algorithm has access to a random number generator. This of course is ever hardly the case. However, to simplify the analysis of the *expected* time complexity of the algorithm, it is usually assumed that the algorithm has access to a "true" random number generator. There are two different types of randomized algorithms:
1. **Las Vegas algorithms**—they always give the correct solution (if they terminate)

2. **Monte Carlo algorithms**—they sometimes give incorrect solutions, however, we are able to bound the probability of failure.

The one good property of Monte Carlo algorithms is that, by running them repeatedly, with independent random choices each time, the failure probability can be made arbitrarily small. Of course, usually at the expense of running time.

### 2.3 More Number Theory

We are now ready to present our first algorithm. The following algorithm, given in Figure 2.1, attributed to Euclid [28], is often used to compute the greatest common divisor of two integers.

The notation $\lfloor x \rfloor$ denotes the greatest integer less or equal to $x$. Euclid's algorithm is ancient and much has been written about it. Its time complexity can be shown to be $T(n) = O(\lg^3 n)$, see for example, Knuth [28] and Koblitz [29]. However, a more careful analysis of the operations involved can lower this to $O(\lg^2 n)$ [28]. We shall use $\lg x$ to denote $\log_2 x$.

**Definition** We say $m \in \mathbb{N}_0$ is the *least common multiple* of $a, b \in \mathbb{N}_0$, denoted by $\text{LCM}(a, b)$, if it satisfies

1. $a|m$ and $b|m$

2. if $a|c$ and $b|c$, with $c \in \mathbb{N}$, then $m \leq c$. 

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time Complexity, $T(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addition, +</td>
<td>$O(\lg n)$</td>
</tr>
<tr>
<td>Subtraction, -</td>
<td>$O(\lg n)$</td>
</tr>
<tr>
<td>Multiplication, *</td>
<td>$O(\lg^2 n)$</td>
</tr>
<tr>
<td>Division, /</td>
<td>$O(\lg^2 n)$</td>
</tr>
</tbody>
</table>

Table 2.1: Time complexities of primitive arithmetic operations.
Input: $r_0, r_1 \in \mathbb{N}_0, r_0 > r_1$

Output: $d = \text{GCD}(r_0, r_1)$

1. $q \leftarrow \lfloor r_0 / r_1 \rfloor$

2. $r \leftarrow r_0 - q \cdot r_1$

3. While ($r > 0$) Do
   
   \{ 
   
   $r_0 \leftarrow r_1$
   
   $r_1 \leftarrow r$
   
   $q \leftarrow \lfloor r_0 / r_1 \rfloor$
   
   $r \leftarrow r_0 - q \cdot r_1$
   
   \}

4. Return ($r_1$)

Figure 2.1: Algorithm EUCLID
**Theorem 2.3.1** For any \(a, b \in \mathbb{N}\), \(\text{GCD}(a, b) \bullet \text{LCM}(a, b) = a \cdot b\).

Next we introduce the notion of congruence.

**Definition** Let \(a, b \in \mathbb{Z}\) and \(n \in \mathbb{N}_1\). Then, we say \(a\) and \(b\) are congruent modulo \(n\), denoted by \(a \equiv b \pmod{n}\), if \(n \mid (a - b)\). The integer \(n\) is called the *modulus*.

It is easily seen that \(a \equiv b \pmod{n}\) if and only if \(a\) and \(b\) leave the same nonnegative remainder when divided by \(n\). Hence, every integer is congruent modulo \(n\) to exactly one of \(0, 1, \ldots, n - 1\). This set, of residues modulo \(n\), is usually denoted by \(\mathbb{Z}_n\), and the set \(\mathbb{Z}_n - \{0\}\), is denoted by \(\mathbb{Z}^*_n\).

The next theorems give some of the properties of modular arithmetic.

**Theorem 2.3.2** Let \(n \in \mathbb{N}_1\) be fixed and \(a, b, c, d \in \mathbb{Z}\). Then the following properties are satisfied.

1. \(a \equiv a \pmod{n}\)

2. if \(a \equiv b \pmod{n}\) then \(b \equiv a \pmod{n}\)

3. if \(a \equiv b \pmod{n}\) and \(b \equiv c \pmod{n}\) then \(a \equiv c \pmod{n}\)

4. \(a \equiv b \pmod{n}\) and \(c \equiv d \pmod{n}\) then \(a + c \equiv b + d \pmod{n}\) and \(a \cdot c \equiv b \cdot d \pmod{n}\)

5. if \(a \equiv b \pmod{n}\) then \(a + c \equiv b + c \pmod{n}\) and \(a \cdot c \equiv b \cdot c \pmod{n}\)

6. if \(a \equiv b \pmod{n}\) then \(a^k \equiv b^k \pmod{n}\) for any \(k \in \mathbb{N}_1\)

The first three properties imply that \(\equiv\) is an *equivalence relation* on \(\mathbb{Z}_n\) for a fixed \(n \in \mathbb{N}_1\).

**Theorem 2.3.3** If \(c \cdot a \equiv c \cdot b \pmod{n}\), then \(a \equiv b \pmod{(n/d)}\), where \(d = \text{GCD}(c, n)\).

**Corollary 2.3.4** If \(c \cdot a \equiv c \cdot b \pmod{n}\) and \(\text{GCD}(c, n) = 1\), then \(a \equiv b \pmod{n}\).

**Definition** Let \(x \in \mathbb{Z}_n\). Then an element \(\bar{x} \in \mathbb{Z}_n\) such that \(x \bar{x} \equiv 1 \pmod{n}\) is called the *multiplicative inverse* of \(x\), and is denoted by \(x^{-1}\).

The next theorem theorem tells us when this multiplicative inverse exists.
Theorem 2.3.5 Let \( x \in \mathbb{Z}_n \). Then, \( x \) has a multiplicative inverse if and only if \( \text{GCD}(n, x) = 1 \). Furthermore, this multiplicative inverse is unique.

Corollary 2.3.6 If \( n \) is prime, then every \( x \in \mathbb{Z}_n^* \) has a multiplicative inverse.

Most important to us, is computing the inverse of \( x \pmod{n} \) if it exists. Algorithm EXTENDED-EUCLID, from Stinson [84], given in Figure 2.2, is often used to compute the inverse. It is based on algorithm EUCLID.

The extended Euclidean algorithm, as its name suggests, is an extension of Euclid's algorithm to compute the greatest common divisor of two integers. As a result, its time complexity can be shown to be \( T(n) = O(\log^3(n)) \), the same as Euclid's algorithm. Stinson [84] and Knuth [28] discuss it in detail.

Theorem 2.3.7 Let \( a, b \in \mathbb{Z}_n \), \( n > 1 \) and fixed. The linear congruence \( ax \equiv b \pmod{n} \) has a solution if and only if \( d \mid b \), where \( d = \text{GCD}(a, n) \). If \( d \mid b \), then there are \( d \) mutually incongruent solutions modulo \( n \).

When \( b = 1 \), we have the special case of computing \( a^{-1} \).

Corollary 2.3.8 If \( \text{GCD}(a, n) = 1 \), then the linear congruence \( ax \equiv b \pmod{n} \) has a unique solution modulo \( n \).

The next theorem extends the result of Theorem 2.3.7 to a system of linear congruence equations.

Theorem 2.3.9 (Chinese Remainder Theorem)

Let \( n_1, n_2, \ldots, n_r \in \mathbb{N}_1 \) such that \( \text{GCD}(n_i, n_j) = 1 \) for \( 1 \leq i \neq j \leq r \). Then the system of linear congruences

\[
\begin{align*}
x \equiv a_1 & \pmod{n_1} \\
x \equiv a_2 & \pmod{n_2} \\
\vdots \\
x \equiv a_r & \pmod{n_r}
\end{align*}
\]

has a simultaneous solution, which is unique modulo \( n = n_1 \cdot n_2 \cdots n_r \), namely, \( \bar{x} = a_1 \cdot N_1 \cdot x_1 + a_2 \cdot N_2 \cdot x_2 + \cdots + a_r \cdot N_r \cdot x_r \pmod{n} \), where \( N_i = n/n_i \) and \( x_k \) is the solution to \( N_k \cdot x \equiv 1 \pmod{n_k} \).
**Input:** \(a, n \in \mathbb{N}_0\), such that \(a \in \mathbb{Z}^*_n\) and \(\text{GCD}(a, n) = 1\).

**Output:** \(u = a^{-1}\) (i.e. \(u \cdot a \equiv 1 \pmod{n}\)).

1. \(n_0 \leftarrow n\)
2. \(a_0 \leftarrow a\)
3. \(t_0 \leftarrow 0\)
4. \(t \leftarrow 1\)
5. \(q \leftarrow \lfloor n_0/a_0 \rfloor\)
6. \(r \leftarrow n_0 - q \cdot a_0\)
7. While \((r > 0)\) Do
   
   \[
   \begin{align*}
   \text{temp} & \leftarrow t_0 - q \cdot t \\
   \text{if} \ (\text{temp} \geq 0) \ \text{then} \\
   \text{temp} & \leftarrow \text{temp} \pmod{n} \\
   \text{else} \\
   \text{temp} & \leftarrow n - (-\text{temp}) \pmod{n} \\
   t_0 & \leftarrow t \\
   t & \leftarrow \text{temp} \\
   n_0 & \leftarrow a_0 \\
   q & \leftarrow \lfloor n_0/a_0 \rfloor \\
   r & \leftarrow n_0 - q \cdot a_0
   \end{align*}
   \]
8. Return \((t \pmod{n})\)

Figure 2.2: Algorithm EXTENDED-EUCLID
We will, however, employ the following corollary on numerous occasions.

**Corollary 2.3.10** The mapping defined by

$$\pi: \mathbb{Z}_n \rightarrow \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \cdots \times \mathbb{Z}_{n_r},$$

where \( n \) and \( n_1, \ldots, n_r \) are as in the theorem and \( \pi(x) = (x \mod n_1), (x \mod n_2), \ldots, (x \mod n_r) \), is a bijection.

Next, we give some well known facts from group and field theory that will be useful for a full understanding of this thesis.

## 2.4 Group and Field Theory

**Definition** A *binary operation* on a set \( A \) is a map from \( A \times A \) to \( A \), \( \pi: A \times A \rightarrow A \).

**Definition** A *group* is a nonempty set \( G \), together with a binary operation, \( \ast \), on elements of \( G \) satisfying the following properties:

1. *(associativity)* for all \( g, h, k \in G \), \( (g \ast k) \ast h = g \ast (h \ast k) \)

2. *(identity)* for each \( g \in G \), there exists an element \( e \in G \) such that \( g \ast e = e \ast g \)

3. *(Inverses)* for each \( g \in G \), there exists an element \( g^{-1} \in G \) such that \( g \ast g^{-1} = g^{-1} \ast g = e \)

**Definition** A group \( G \) is *abelian*, or *commutative*, if \( g \ast h = h \ast g \) for all \( g, h \in G \).

The identity element is often denoted by \( 1 \). When \( G \) is abelian, the group operation is often denoted by \( + \) and the identity by \( 0 \). A group \( G \) is said to be finite if it contains only a finite number of elements. The cardinality of a finite group \( G \) is called its *order* and is denoted by \( |G| \) or \( \text{ord}(G) \).

**Definition** A *subgroup* \( S \) of a group \( G \) is a subset of \( G \) that is also a group under the same operation defined on \( G \). We denote the fact that \( S \) is a subgroup of \( G \) by writing \( S < G \).

Interesting subgroups, of a group \( G \), are those generated by an element, \( g \in G \). That is, the set \( \langle g \rangle = \{g^n | n \in \mathbb{Z}\} \). The set \( \langle g \rangle \) is usually referred to as the *cyclic subgroup generated by g*.

**Definition** Let \( G \) be a group.
1. If \( g \in G \), and if \( g^k = e \) for some \( k \in \mathbb{Z} \), we say that \( k \) is an exponent of \( g \).

2. An \( m \in \mathbb{Z} \) for which \( g^m = e \) for all \( g \in G \) is called the exponent of \( G \).

The smallest positive exponent for \( g \in G \) is called the order of \( g \), denoted by \( \text{ord}(<g>) \).

**Definition** A finite group \( G \) is cyclic if and only if it has an element of order \( \text{ord}(G) \).

One way of looking at this is that, \( G \) is cyclic if and only if \( G = <g> \) for some \( g \in G \). In this case we say \( g \) generates \( G \). The following theorem gives us more information about the exponents.

**Theorem 2.4.1** Let \( G \) be a group, \( g \in G \). Then \( k \) is an exponent of \( g \) if and only if \( k | \text{ord}(<g>) \).

Similarly, the exponents of \( G \) are precisely the multiples of the smallest positive exponent of \( G \).

In this review, we will mainly be working with finite abelian groups, so the next two theorems characterize the smallest positive exponent for finite abelian groups.

**Theorem 2.4.2** Let \( G \) be a finite abelian group.

1. If \( m \) is the maximum order of all elements in \( G \), then \( g^m = e \) for all \( g \in G \). Thus, the smallest positive exponent of \( G \) is equal to the maximum order of all elements of \( G \).

2. The smallest positive exponent of \( G \) is equal to \( \text{ord}(G) \) if and only if \( G \) is cyclic.

3. If \( G \) is cyclic, then every subgroup of \( G \) is cyclic.

4. If \( G = <g> \) is a cyclic group of order \( n \), then
   
   1. For \( 1 \leq k \leq n \), \( \text{ord}(<g^k>) = \frac{n}{\text{GCD}(n,k)} \)
   
   2. If \( d | n \), then \( \text{ord}(g^k) = d \iff k = r \frac{n}{d} \) where \( \text{GCD}(r, d) = 1 \).

**Theorem 2.4.3** Let \( G \) be a finite group.

1. (Lagrange) For any \( S < G \), \( \text{ord}(S) | \text{ord}(G) \)

2. For any \( g \in G \), \( \text{ord}(<g>) | \text{ord}(G) \)

3. If \( G \) is a finite abelian group and if \( k | \text{ord}(G) \), then \( G \) has a subgroup of order \( k \).
4. (Cauchy) If \( \text{ord}(G) \) is divisible by a prime \( p \), then \( G \) contains an element of order \( p \).

5. If \( p \) is a prime and \( \text{ord}(G) \) is divisible by \( p^n \), then \( G \) contains a subgroup of order \( p^n \).

**Definition** A ring is a nonempty set \( R \), together with two binary operations on its elements, called *addition* (denoted by \( + \)), and *multiplication* (denoted by \( \cdot \)), satisfying the following properties.

1. \( R \) is an abelian group under \( + \)

2. (Associativity) for all \( g, h, k \in R \), \( (g \cdot k) \cdot h = g \cdot (h \cdot k) \)

3. (Distributivity) for all \( g, h, k \in R \), \( (g + h) \cdot k = g \cdot k + h \cdot k \) and \( h \cdot (g + k) = h \cdot g + h \cdot k \)

The ring \( R \) is called a *ring with identity* if there exists an element \( 1 \in R \) for which \( r \cdot 1 = 1 \cdot r = r \) for all \( r \in R \). From this point onwards we will denote the identity of the group by \( 1 \). That is, \( e \equiv 1 \).

**Definition** Let \( R \) be a ring with identity. \( R \) is called *field* if the nonzero elements of \( R \) form an abelian group under multiplication.

**Definition** A *subring* \( S \) of a ring \( R \) is a nonempty subset of \( R \) that is also a ring under the same operations defined on \( R \).

**Definition** A *subfield* \( E \) of a field \( F \) is a nonempty subset of \( F \) that is also a field under the same operations defined on \( F \). In this case, we say that \( F \) is an *extension* of \( E \) and we write \( E < F \) or \( F > E \).

**Definition** Let \( F \) be a field. The *characteristic*, \( \text{char}(F) \), of \( F \) is the smallest positive integer \( k \) (if it exists) for which \( k \cdot 1 = 1 + 1 + \cdots + 1 \) (\( k \) times) = 0. If it doesn't exist we say \( F \) has characteristic 0. In the former case \( F \) contains a copy of \( \mathbb{Z}_k \), and in the latter, \( F \) contains a copy of \( \mathbb{Q} \), the rational numbers.

The notion of finiteness extends to fields as well. Finite fields play a very important role in cryptography, and below we give some of the properties of finite fields that will be useful to us.

**Theorem 2.4.4** Let \( F \) be a finite field.
1. $F$ has prime characteristic (we denote this by $\mathbb{F}_p$)

2. $F^*$, the multiplicative group of all nonzero elements of $F$, is cyclic

3. $F$ has size $q = p^n$, for some prime $p$ and $n \in \mathbb{N}_0$. In this case we denote $F$ by $\mathbb{F}_q$ or $GF(q)$, where the symbol $GF$ stands for Galois Field, in honor of Evariste Galois.

**Definition** An element $\alpha \in \mathbb{F}_q$ that generates the cyclic group $\mathbb{F}_q^*$ is called a **primitive element** of $\mathbb{F}_q$.

It is not difficult to ascertain that $\mathbb{Z}_p$ is a field under addition and multiplication modulo $p$. Next, we present some number theoretic results whose understanding benefits from the results above.

**Definition** (Euler phi function, $\phi$)

Let $n \in \mathbb{N}_0$, then the Euler phi function is defined to be

$$\phi(n) = | \{ k \in \mathbb{N}_0 | 1 \leq k < n \text{ and } \gcd(k, n) = 1 \} |.$$

The next theorem gives some properties of the Euler phi function, in fact, the two properties completely determine $\phi$.

**Theorem 2.4.5**

1. The Euler phi function is multiplicative, that is $\phi(mn) = \phi(m)\phi(n)$, when $\gcd(m, n) = 1$.

2. $\phi(p^n) = p^{n-1}(p - 1)$, $p$ prime, $n \in \mathbb{N}_0$.

Connected to Euler's phi function is the following theorem, also due to Euler, and its corollary. Their proofs rely heavily on Theorem 2.3.5 and Corollary 2.3.6.

**Theorem 2.4.6 (Euler's Theorem)**

If $\alpha, n \in \mathbb{Z}$ and $\gcd(\alpha, n) = 1$ then $\alpha^{\phi(n)} \equiv 1 \pmod{n}$.

**Corollary 2.4.7 (Fermat's Theorem)**

If $p$ is a prime not dividing $\alpha \in \mathbb{N}$, then $\alpha^p \equiv \alpha \pmod{p}$. 
Input: $x, k, n$ where $k, n \in \mathbb{N}_0, x \in \mathbb{Z}_n$

Output: $y \equiv x^k \pmod{n}$ Let $k_{i-1} \cdots k_1 k_0$ be the binary representation of $k$.

1. $y \leftarrow 1$

2. For $i \leftarrow l - 1$ downto 0 Do
   
   \{
   
   $y \leftarrow y^2 \pmod{n}$

   If $k_i = 1$ Then
   
   $y \leftarrow y \cdot x \pmod{n}$

   \}

3. Return ($y \pmod{n}$)

Figure 2.3: Algorithm MOD-EXP

So far, our results have been largely theoretical rather than the practical. However, we wish to be able to compute $y \equiv x^k \pmod{n}$, with $x, k, n \in \mathbb{N}_0$ known. This is known as modular exponentiation. The algorithm given in Figure 2.3 can be used to compute $y \equiv x^k \pmod{n}$ for $k > 0$ and $x \in \mathbb{Z}_n$.

In many situations it is useful to have solutions of the equation $x^n = 1$ in $\mathbb{F}_q$. This solutions (roots) are usually called the $n$th roots of unity. Of interest, is the special case $n = 2$. First though, a few definitions to get us started.

**Definition** Let $p$ be an odd prime, and $a \neq 0 \in \mathbb{Z}$ such that $\gcd(a, p) = 1$. Then we say $a$ is a quadratic residue modulo $p$, denoted by $a \in \text{QR}(p)$, where

$$\text{QR}(p) = \{ x \mid x \in \mathbb{Z}_p \text{ and there exist } y \in \mathbb{Z}_p \text{ such that } y^2 \equiv x \pmod{p} \},$$

if the quadratic congruence $x^2 \equiv a \pmod{p}$ has a solution in $\mathbb{Z}_p$. Otherwise, $a$ is called a quadratic nonresidue modulo $p$.

It is easy to see that if $p \equiv 3 \pmod{4}$ and $a \in \text{QR}(p)$ then also $p - a \in \text{QR}(p)$. 

Problem Instance: Field \( \mathbb{Z}_p \) and \( \alpha \in \mathbb{Z}_p \)

Question: Is \( \alpha \in \text{QR}(p) \)?

Figure 2.4: Decision Problem QUADRATIC-RESIDUE

**Definition** Let \( p \) be an odd prime. For \( x \in \mathbb{N} \), we define the Legendre symbol \( L(\frac{x}{p}) \) as follows:

\[
L(\frac{x}{p}) = \begin{cases} 
0 & \text{if } p \mid x \\
1 & \text{if } x \in \text{QR}(p) \\
-1 & \text{if } x \notin \text{QR}(p)
\end{cases}
\]

Often we want to answer the question: Is \( \alpha \in \mathbb{Z}_p \) a quadratic residue modulo \( p \)? To answer this question and others of interest, we first define the notion of a **decision problem**.

**Definition** A decision problem, \( \Pi \), is a problem whose solution is either “yes” or “no”.

The above question can now be cast as a decision problem, Figure 2.4.

At first glance, it seems that to solve the decision problem QUADRATIC-RESIDUE, we will have to try all \( x \in \mathbb{Z}_p \), which could be time consuming for large \( p \). But luckily, a result known as **Euler’s criterion** reduces this to just a modular exponentiation problem, which we have already seen can be performed in \( O(\log^3 p) \) time.

**Theorem 2.4.8 (Euler’s criterion)**

*Let \( p \) be prime. Then \( \alpha \in \text{QR}(p) \) if and only if*

\[
\alpha^{\frac{p-1}{2}} \equiv 1 \pmod{p}.
\]

Euler’s criterion also gives us a way to compute the Legendre symbol, since it can easily be shown that \( L(\frac{x}{p}) \equiv \alpha^{\frac{x-1}{2}} \pmod{p} \); see, for example, Koblitz [29]. We also need to be able to generate quadratic residues, unfortunately, there is no known deterministic polynomial time algorithm to do this. But if the so called **Riemann Hypothesis** holds, then generating a quadratic residue could be done in deterministic polynomial time. We have to resort to a randomized algorithm, in which we
randomly choose, independently, numbers in \( \mathbb{Z}_p^* \) and test them using Euler's criterion. There are precisely \( \frac{p-1}{2} \) quadratic residues in \( \mathbb{Z}_p^* \). This means that the probability of a failure is \( \frac{1}{2} \). Next, we define the generalization of the Legendre symbol.

**Definition** Let \( n \) be an odd positive integer, and the prime factorization of \( n \) is \( p_1^{e_1} \cdots p_k^{e_k} \). Let \( x \in \mathbb{N}_{-1} \). The Jacobi symbol \( J(\frac{x}{n}) \) is defined to be

\[
J(\frac{x}{n}) = \prod_{i=1}^{k} (L(\frac{x}{p_i}))^{e_i}
\]

The Jacobi symbol, although looks imposing to compute, can in fact be computed in polynomial time using the generalization of the Law of Quadratic Reciprocity [29, 61] given below [84].

**Properties of the Jacobi symbol**

Let \( n \) be an odd positive number, \( a, b \in \mathbb{N}_0 \).

1. \( J(\frac{-1}{n}) = (-1)^{\frac{n-1}{2}} \)

2. \( J(\frac{ab}{n}) = J(\frac{a}{n}) \cdot J(\frac{b}{n}) \)

3. If \( a \equiv b \) (mod \( n \)), then \( J(\frac{a}{n}) = J(\frac{b}{n}) \)

4. \[
J(\frac{2}{n}) = \begin{cases} 
1 & \text{if } n \equiv \pm 1 \pmod{8}, \\
-1 & \text{if } n \equiv \pm 3 \pmod{8} 
\end{cases}
\]

5. **(Law of Quadratic Reciprocity)** If \( m \) is an odd positive integer, then

\[
J(\frac{m}{n}) = \begin{cases} 
-J(\frac{n}{m}) & \text{if } n \equiv m \equiv \pm 3 \pmod{4}, \\
J(\frac{n}{m}) & \text{otherwise}
\end{cases}
\]

Using these properties, an \( O(\log^3 n) \) time algorithm can be devised to compute the Jacobi symbol. Note that only modular reductions and factoring out powers of two (using property 1) are required in the computation of the Jacobi symbol, as can be seen from the example below. It is easy to see that \( O(\log n) \) modular reductions are performed.
Problem Instance: $n \in \mathbb{N}_2$

Question: Is $n$ composite?

Figure 2.5: Decision Problem COMPOSITE

Example

\[
J\left(\frac{415}{283}\right) = J\left(\frac{132}{283}\right) \text{ by property 2} \\
= J\left(\frac{2^2}{283}\right)J\left(\frac{33}{283}\right) \text{ by property 1} \\
= (J\left(\frac{2}{283}\right))^2J\left(\frac{2}{283}\right) \text{ by property 1} \\
= (-1)^2J\left(\frac{283}{33}\right) \text{ by properties 3, 4} \\
= J\left(\frac{19}{33}\right) \text{ by property 2} \\
= J\left(\frac{33}{19}\right) \text{ by property 4} \\
= J\left(\frac{14}{19}\right) \text{ by property 2} \\
= J\left(\frac{2}{19}\right)J\left(\frac{7}{19}\right) \text{ by property 1} \\
= (-1)J\left(\frac{7}{19}\right) \text{ by property 3} \\
= (-1)(-1)J\left(\frac{19}{7}\right) \text{ by property 4} \\
= J\left(\frac{5}{7}\right) \text{ by property 2} \\
= J\left(\frac{7}{5}\right) \text{ by property 4} \\
= J\left(\frac{2}{5}\right) \text{ by property 2} \\
= -1 \text{ by property 3}
\]

Our next decision problem, COMPOSITE, presented in Figure 2.5, has been studied for quite a long time. As a result, good randomized algorithms now exist for it.
CHAPTER 2. MATHEMATICAL PRELIMINARIES

Just like QUADRATIC-RESIDUE, the decision problem COMPOSITE has no known deterministic polynomial time algorithm for its solution. As already mentioned, it has been widely studied over the centuries, though usually in the form of a primality testing decision problem. Several randomized algorithms exist for it, though here we only briefly describe the one by Miller and Rabin [84], Figure 2.6.

The MILLER-RABIN algorithm is perhaps the most widely used and fast in practice. It can be shown, Miller [55], that it has a failure probability of at most $\frac{1}{4}$, and usually it suffices to only repeat it 32 times.

Another decision problem of interest to us is knowing whether $\alpha \in \mathbb{Z}_p^*$ is a primitive element of $\mathbb{Z}_p$. It is presented in Figure 2.7. The decision problem PRIMITIVE also has no known deterministic polynomial time algorithm. However, the following result [28], which we record as a theorem proves to be useful.

**Theorem 2.4.9** Let $p$ be prime and $p - 1 = p_1^{e_1} \cdots p_k^{e_k}$. Then we have that $\alpha \in \mathbb{Z}_p^*$ is a primitive element of $\mathbb{Z}_p$ if and only if

$$\alpha^{\frac{p-1}{p_j}} \neq 1 \pmod{p}, 1 \leq j \leq k.$$  

It is fairly easy to show that there are $\phi(p-1)$ primitive elements in $\mathbb{Z}_p^*$. This follows from the fact that every $\alpha \in \mathbb{Z}_p^*$ can be written as $\alpha = g^i$, where $g \in \mathbb{Z}_p^*$ is a generator of $\mathbb{Z}_p^*$ and $0 \leq i \leq p-2$. Then, by Theorem 2.4.2, $ord(\alpha) = \frac{p-1}{\text{GCD}(p-1,i)}$, which implies that $\alpha$ is a primitive element if and only if $\text{GCD}(p-1, i) = 1$. Thus, it follows that there are $\phi(p-1)$ primitive elements in $\mathbb{Z}_p^*$. In order to use Theorem 2.4.9, we have to know the factorization of $p - 1$. As will be shown in the next section, this is no easy task for $p$ large. Fortunately for us, in designing cryptosystems, we can choose these large primes. Therefore, we would choose $p$ so that the factorization of $p - 1$ is known.

**Example** We would choose a prime $p = 2p_1 + 1$, where $p_1$ is prime. Then for $\alpha \in \mathbb{Z}_p^*$, $\alpha \neq \pm 1 \pmod{p}$ it can easily be verified that $\alpha$ is a primitive element if and only if $\alpha^{\frac{p-1}{2}} \neq 1 \pmod{p}$.

Thus, if we know the factorization of $p - 1$, a Las Vegas algorithm can be employed to generate a primitive element with the probability of a failure of $\frac{\phi(p-1)}{p-1}$. In our example above, this probability is approximately $\frac{1}{2}$ when $p$ is large. Moreover, each $\alpha \in \mathbb{Z}_p^*$ can be tested for primitivity in polynomial time since there are $O(\lg p)$ prime factors of $p - 1$. 

Input: $n \in \mathbb{N}_2$

Output: 1 if $n$ is composite and 0, if prime.

1. $\text{result} \leftarrow 1$

2. Decompose $n - 1$ as $2^k \cdot m$

3. Choose a random $\alpha \in \mathbb{Z}_n^*$

4. $b \leftarrow \alpha^m \pmod{n}$

5. If $(b \equiv 1 \pmod{n})$ Then
   
   $\text{result} \leftarrow 0$

   Else
   
   $i \leftarrow 0$
   
   While $i \leq k - 1$ Do
   
   $\text{If } (b \equiv -1 \pmod{n}) \text{ Then}$
   
   $\text{result} \leftarrow 0$
   
   $i \leftarrow k$

   $\text{Else}$
   
   $b \leftarrow b^2 \pmod{n}$
   
   $i \leftarrow i + 1$

   $\text{End}$

6. Return $(\text{result})$

Figure 2.6: Algorithm MILLER-RABIN
Problem Instance: Field $\mathbb{Z}_p$ and $\alpha \in \mathbb{Z}_p^*$

Question: Is $\alpha$ a primitive element of $\mathbb{Z}_p$?

Figure 2.7: Decision Problem PRIMITIVE

Problem Instance: $n \in \mathbb{N}_2$ and odd.

Task: Find all the prime factors of $n$.

Figure 2.8: Problem: FACTORING

2.5 Core Number-Theoretic Algorithms

In this section we discuss three number-theoretic algorithms that are central to most of the cryptosystems discussed in this survey. We do not give details about these algorithms since a lot has been written about them, and refer the reader to the references cited. The complexity of each algorithm is expressed with the following function:

$$L[x, c, \alpha] = O(e^{(c+o(1))((\log x)^{\alpha}((\log \log x)^{1-\alpha})},$$

where $x$ is the input size, $c$ is a constant and $0 < \alpha < 1$. Any algorithm having this time complexity is said to be subexponential. Note that, if $\alpha = 0$ than the time complexity is polynomial in $\log x$, and if $\alpha = 1$ the time complexity is polynomial in $x$, thus fully exponential in $\log x$. The notation, $o(1)$, stands for a function $f(n)$ such that $f(n) \to 0$ as $n \to \infty$. The first problem we discuss, given in Figure 2.8, is as old as the branch of number theory itself.

The problem, FACTORING, has been studied intensively for a long time, but so far it has managed to remain elusive, even in the face of modern technology. However, several important and most practical algorithms, given in Table 2.2, have been designed for the problem.

The quadratic sieve algorithm, developed by Pomerance [66], is suitable for factoring general integers consisting of two large primes of about the same size. It has a heuristic running time of
Table 2.2: Factoring Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quadratic Sieve</td>
<td>$O(e^{(1+o(1))\sqrt{\log n \log \log n}})$</td>
</tr>
<tr>
<td>Elliptic Curve</td>
<td>$O(e^{(1+o(1))\sqrt{\log \log p \log \log \log p}})$</td>
</tr>
<tr>
<td>Number Field Sieve</td>
<td>$O(e^{(1.92+o(1))\frac{1}{3}(\log n)^{\frac{4}{3}}(\log \log n)^{\frac{1}{3}}})$</td>
</tr>
</tbody>
</table>

Problem Instance: Finite group $\mathbb{Z}_p$, where $p$ is prime, $\alpha \in \mathbb{Z}_p^*$, a primitive element, and $\beta \in \mathbb{Z}_p^*$.

Task: Find the unique integer $a \in \mathbb{Z}_{p-1}$ such that $\alpha^a \equiv \beta \pmod{p}$.

The integer $a$ is called the discrete log of $\beta$, and is denoted by $\log_\alpha \beta$.

Figure 2.9: The Discrete Logarithm Problem (DLP)

$L[n, 1, \frac{1}{2}]$. The elliptic curve method of factorization, invented by Lenstra Jr. [21], is an analogue of Pollard's $(p - 1)$-method [43, 84] which is effective when $n$ has a prime factor $p$ such that $p - 1$ is $y$-smooth for small $y$. It has a heuristic time complexity of $L[y, \sqrt{2}, \frac{1}{2}]$, where $p$ is the smallest prime factor of $n$. It is effective when there is an integer "close to" $p$ that is $y$-smooth for small $y$. Hence, it is more useful when the prime factors of $n$ are of differing size. In fact, the largest factor ever found with the elliptic curve method has 47 digits [82]. The number field sieve [44] is the most recent of the three algorithms and has a better asymptotic time complexity of $L[n, 1.92, \frac{1}{3}]$. It is particularly suited for factoring integers having around 150 digits or more and is fastest for numbers of the form $r^e \pm s$ (Cunningham numbers), with $r$ and $s$ small. All these algorithms can be parallelized, making them well suited for distributed computing. In fact, RSA-130, a composite integer with 130 digits, was factored in 1996 by a distributed implementation of the number field sieve [42]. This is the current record for factoring algorithms.

The second problem, DISCRETE-LOG, given in Figure 2.9, has also been intensely studied and is now believed to be equally as difficult to compute discrete logarithms modulo $p$ as it is to factor an integer $n$ of the same size [50]. This is only from comparing the running times of the best algorithms for both problems.
Several algorithms now exist for the DLP. We list some of them in Table 2.3. Shank's time-memory trade-off algorithm [78] computes a database of $O(\sqrt{p})$ logarithms, requiring a storage space of that amount. It quickly becomes impractical for large $p$. The Pohlig-Hellman algorithm [64] also has a time complexity in the order of $\sqrt{p}$, but it is more effective when $p - 1 = p_1^{e_1} \cdots p_t^{e_t}$ is $y$-smooth for small $y$. The index calculus method [43, 84, 29] bears a lot of resemblance to many of the best factoring algorithms and even has the same heuristic time complexity of $L[p, 2, \frac{1}{2}]$. Recently, some variants of the index calculus method has been developed to solve the DLP. This includes the linear sieve [16, 43] and Coppersmith's variant for the DLP over $\mathbb{F}_{2^n}$ [15]. In fact, Coppersmith's algorithm is asymptotically the best in fields of characteristic 2. It has a heuristic time complexity of $L[2^n, c, \frac{1}{3}]$, where $1.3507 \leq c \leq 1.4047$. To date, the number field sieve is the fastest (asymptotically) algorithm known. It has a time complexity similar to the its factoring counterpart. Another important algorithm is the Gaussian integer scheme [16], which is similar to the number field sieve but works only with quadratic number fields. A variant of the quadratic sieve algorithm was recently used by Lercier and Joux [46] to compute discrete logarithms modulo a 90 digits prime. They used a distributed implementation of the algorithm.

The third problem we discuss, SQUARE-ROOT($p$), is related to the decision problem QUADRATIC-RESIDUE. We have already seen that the decision problem can be solved in polynomial time for $p$ prime. However, this only tells us whether $a \in \mathbb{Z}_p^*$ is quadratic residue or not, and not how to find a solution to the congruence $x^2 \equiv a \pmod{p}$. There are several algorithms to solve this congruence for $p$ prime, but we shall be content with describing the algorithm from Koblitz [29], presented in Figure 2.10.
Input: $p$, prime and $a, b \in \mathbb{Z}_p^*$ such that $a \in QR(p)$ and $b \notin QR(p)$.

Output: $x \in \mathbb{Z}_p^*$ such that $x^2 \equiv a \pmod{p}$.

1. Compute $\alpha$ and $s$ such that $p - 1 = 2^\alpha \cdot s$ where $s$ odd.

2. $b_1 \leftarrow b^4 \pmod{p}$

3. $r \leftarrow a^{(\frac{p+1}{2})} \pmod{p}$
   Compute $j = j_0 + 2j_1 + 2^2j_2 + \cdots + 2^{\alpha-2}j_{\alpha-2}$ such that
   $x = b_1^j r$ is the desired square root of $a \pmod{p}$, and $j_i = 0$ or 1.

4. $\text{bit} \leftarrow (\frac{t}{a}2^{\alpha-2}) \pmod{p}$

5. If (bit = 1) Then
   $j_0 \leftarrow 0$
   Else
   $j_0 \leftarrow 1$

6. For $i \leftarrow 1$ to $\alpha - 2$ Do
   {
   
   \begin{align*}
   \text{bit} & \leftarrow \left(\frac{\{b_1j_0 + 2j_1 + \cdots + 2^i-1j_{i-1}r\}^2}{a}2^{\alpha-1-2}\right) \pmod{p}
   \end{align*}
   
   If (bit = 1) Then
   $j_0 \leftarrow 0$
   Else
   $j_0 \leftarrow 1$
   \}

7. Return $(x = b_1^j \cdot r \pmod{p})$

Figure 2.10: Algorithm SQUARE-ROOT(p)
1. Compute the square root, \( r_i \), of \( a \) modulo \( p_i \), \( 1 \leq i \leq k \).

2. "Lift" \( r_i \) to a square root, \( \bar{r}_i \), of \( a \) modulo \( p_i^{e_i} \).

3. Use Theorem 2.3.9 and the \( \bar{r}_i \)'s to compute a square root of \( a \) modulo \( n \).

Figure 2.11: Computing Square Roots Modulo \( n \)

The algorithm SQUARE-ROOT(\( p \)) can be shown to have \( O(\lg^4 n) \) time complexity [29]. The only downside to the algorithm is that it requires a quadratic nonresidue, and we have already seen that there is no known deterministic polynomial time algorithm for generating one, unless the Riemann Hypothesis holds. For more details on the algorithm, including its correctness, the reader is referred to Koblitz [29]. However, at the same time, Bach [6] has showed the probability of a failure is \( O\left(\frac{\ln p}{\sqrt{p}}\right) \) when the random numbers are chosen using a linear congruential function. He also showed that the expected number of trials is about \( O\left(\frac{1}{\ln p}\right) \). In experiments we performed, we were able to find quadratic nonresidues in about 2 trials on average. We note that if \( p \equiv 3 \pmod{4} \), then the square roots of \( a \pmod{p} \) are easily given by \( x = \pm a^{\frac{p+1}{4}} \pmod{p} \). The general case of computing square roots \( \pmod{n} \) is more complicated. In fact, it can be shown Motwani et al. [61] that computing square roots modulo any \( n \) is as hard as factoring \( n \). But, if the prime factors of \( n \) are known, say \( n = p_1^{e_1} \cdots p_k^{e_k} \), then, a square root of \( a \pmod{n} \) could be computed by the following procedure in Figure 2.11.

Step 2 can be done in randomized polynomial time; the reader is referred to Koblitz [29] and Motwani et al. [61] for more details.

2.6 Public-Key Cryptography

In this section we give a brief review of public-key cryptography. A more thorough exposition can be found in Stinson [84], Schneier [73], Rivest [68], and Salomaa [72]. Cryptography, informally, is the art and science of communicating in the presence of adversaries. Two people, usually referred to as Bob and Alice wish to communicate over an insecure channel in such a way that the adversary,
Oscar cannot understand what is being said. The information that Bob wants to send to Alice is called "plaintext". Bob would then encrypt the plaintext, using a predetermined key, and send the resulting "ciphertext" to Alice. Since the ciphertext is sent over an insecure communication channel, Oscar can easily acquire a copy by eavesdropping, but cannot determine what the plaintext was; only Alice, who knows the encryption key, can decrypt the ciphertext and reconstruct the plaintext. We begin, first, with some definitions and notation.

**Definition** A function, $f(n)$, is said to be one-way if, given $f(i) = z$, it is computationally infeasible to find any $j$ (including $i$) such that $f(j) = z$. That is, it is difficult to compute $f^{-1}(z)$.

A publicly available one-way function has a number of useful applications, including storing of passwords on a time-shared system and in cryptography, as we shall soon demonstrate.

**Definition** A trapdoor one-way function (TOF), $f(n)$, is like a one-way function except that there also exist secret information (the trapdoor) that makes it easy to invert $f$ at any point.

Factoring and exponentiation modulo $p$ are examples of TOF's. As a result they form the basis of many cryptosystems.

**Definition** A cryptosystem is a five-tuple $(P, C, K, E, D)$ where

1. $P$ is a finite set of possible plaintexts
2. $C$ is a finite set of possible ciphertexts
3. $K$ is a finite set of possible keys, also called the *keyspace*
4. $E$ is a set of encryption rules, for each $K \in K$, there is an encryption rule, $e_K \in E$, and a decryption rule $d_K \in D$. Each $e_K: P \rightarrow C$ and $d_K: C \rightarrow P$ are 1-1 functions such that $d_K(e_K(x)) = x$ for all $x \in P$

A *public-key* cryptosystem is a cryptosystem where the encryption rule, $e_K$, is made public, and the decryption rule, $d_K$ is kept private. To use the system, Bob publishes his encryption key in the "directory of public keys" (as in telephone directory) but keeps his decryption key private. Now, if
Alice wants to send a message, $m$, to Bob, she first looks up Bob's encryption key from "directory of public keys", computes the ciphertext, $c = e_K(m)$, and sends it to Bob. Only Bob can decrypt the message since he is the only one who knows the decryption key, $d_K$. For the cryptosystem to be secure, no one should be able to derive Bob's decryption key in any feasible amount of time. Hence, the encryption function should necessarily be a TOF, with the trapdoor information built into the decryption key. Anderson and Needham [4] discusses robustness principles for public-key cryptosystems.

Next we present three public-key cryptosystems which forms the basis of elliptic curve cryptosystems to be discussed in the following chapters.

### 2.6.1 RSA

In 1977, Rivest, Shamir, and Adleman proposed a public-key cryptosystem (RSA) based on the factoring problem [69]. This came after the seminal paper of Diffie and Hellman in 1976 [19], who were the first to come up with the notion of a public-key cryptosystem. The RSA cryptosystem is presented in Figure 2.12.

To show the correctness of the cryptosystem, we need to show that encryption and decryption are inverse operations. Since $ed \equiv 1 \pmod{\phi(n)}$, we have $ed = k\phi(n) + 1$ for some $k \geq 1$. Now suppose $x \in \mathbb{Z}_n - \{0\}$, then

$$x^{ed} \equiv x^{k\phi(n)} \pmod{p}$$

$$\equiv x(x^{p-1})^{k(q-1)} \pmod{p}$$

$$\equiv x(1)^{k(q-1)} \pmod{p} \text{ using Fermat's Theorem}$$

$$\equiv x \pmod{p}.$$ 

If $x = 0$, then clearly, $x^{ed} \equiv x \pmod{p}$. Similarly, $x^{ed} \equiv x \pmod{q}$. Thus by Theorem 2.3.9, $x^{ed} \equiv x \pmod{n}$.

Encryption and decryption are just instances of modulo exponentiation, and can be done in $O(\log^3 n)$ time. The set-up step can be done in probabilistic polynomial time, by using a Monte Carlo prime
Set-up

1. Generate two "large" primes $p, q$ such that $p \neq q$.

2. Compute $n = pq$ and $\phi(n) = (p - 1)(q - 1)$.

3. Find $e$ such that $\text{GCD}(e, \phi(n)) = 1$.

$\mathcal{P} = \mathcal{C} = \mathbb{Z}_p^*, \mathcal{K} = \{(n, p, q, e, d): n = pq, p, q \text{ prime, } ed \equiv 1 \pmod{\phi(n)}\}$

**Public Key:** $(e, n)$

**Private Key:** $(d, p, q)$ where $d = e^{-1} \pmod{\phi(n)}$

**Encryption**

Given key, $K = (n, p, q, e, d)$ and message, $m \in \mathcal{P}$, define

$$e_K(m) = m^e \pmod{n}$$

**Decryption**

Given ciphertext, $c \in \mathbb{Z}_n^*$ define

$$d_K(c) = c^d \pmod{n}$$

Figure 2.12: The RSA Public-Key Cryptosystem
testing algorithm and the application of the Prime Number Theorem, which states that, the number of primes not exceeding \( N \) is approximately \( \frac{N}{\log N} \). Thus, we need only randomly sample \( \lg N \) odd numbers to find one that is a probable prime. Silverman [82] discusses the generation of random, strong RSA primes. The method discussed only uses probabilistic primality testing algorithms. Hence, it is feasible to generate “primes” for RSA and other public-key cryptosystems.

### 2.6.2 Attacks on RSA

The obvious attack on the RSA cryptosystem is to factor the modulus \( n \). By carefully selecting the two primes \( p \) and \( q \) used in RSA, factoring algorithms can be rendered ineffective. In particular, \( p \) and \( q \) should roughly be of the same size (about 100 decimal digits) and neither \( p - 1 \) and \( q - 1 \) should be \( y \)-smooth for small \( y \) (they should have at least one large prime factor). This will guard against the Pollard rho-method and the elliptic curve method of factoring. Since \( n \) has almost 200 decimal digits, this also guards against the quadratic sieve algorithm. It is still not known whether breaking RSA is equivalent to factoring the modulus \( n \). Another avenue for attacking RSA would be to compute \( \phi(n) \) efficiently. But, computing \( \phi(n) \) is polynomial-time equivalent to factoring \( n \) since, if we knew \( \phi(n) \) we could solve for \( p \) and \( q \) from the equations

\[
\phi(n) = (p - 1)(q - 1) \quad \text{and} \quad n = pq.
\]

In fact, these are the roots of \( p^2 - (n - \phi(n) + 1)p + n = 0 \). Hence, we can not hope to make any head way by trying to compute \( \phi(n) \). Yet another avenue is through computing the decryption exponent \( d \). However, it has been shown that any algorithm which computes the decryption exponent \( d \) can be used as a subroutine (oracle) in a probabilistic algorithm to factor the modulus \( n \) [84, 72]. Huber [24], by considering cycle length of the recursion \( c \leftrightarrow c^{\phi(n)-1} + 1 \pmod{n} \) (for suitable \( c \)), gives two conditions which “safe” RSA modulus \( n \) must fulfill, by way of Fibonacci numbers \( (F_0 = 0, F_1 = 1, \text{ and } F_{j+1} = F_j + F_{j-1}, j = 1, 2, \ldots) \). Lastly, we mention another possible attack if RSA with a small encryption exponent \( e \) is used to send the same message to several recipients. Hastad [23], proved that if the encryption exponent is \( e = 3 \) and the same message \( m \) is sent to \( k \geq 7 \) recipients, then it is possible to recover \( m \). Currently, as already mentioned, the best attempt at factoring comes from using networked computers, where the computation load is distributed over a number of computers. Odlyzko's [63] article on the future of integer factorization is worth reading.
CHAPTER 2. MATHEMATICAL PRELIMINARIES

Set-up

1. Generate a "large" primes $p$ such that the DLP in $Z_p$ is intractable

2. Choose a primitive element, $\alpha \in Z_p^*$

$P = Z_p^*, C = Z_p^* \times Z_p^*, K = \{(p, \alpha, a, \beta); \beta \equiv \alpha^a \pmod{p}, a \in Z_p^*, \text{ randomly chosen}\}$

Public Key: $(p, \alpha, \beta)$

Private Key: $(a)$

Encryption

Given key, $K = (p, \alpha, a, \beta)$ and message, $m \in P$, define

$e_K(m, k) = (y_1, y_2) = (\alpha^k, m\beta^k)(\mod{p})$ where $k \in Z_{p-1}^*$ is a secret random number

Decryption

Given ciphertext, $c = (y_1, y_2) \in C$ define

$d_K(c) = y_2(y_1^a)^{-1}(\mod{p})$

Figure 2.13: The ElGamal Public-Key Cryptosystem

2.6.3 ElGamal

In 1985, ElGamal proposed a public-key cryptosystem based on the discrete logarithm problem [20]. This cryptosystem is presented in Figure 2.13.

The correctness of the ElGamal cryptosystem easily follows from the following.

\[
\begin{align*}
d_K(y_1, y_2) &= y_2(y_1^a)^{-1}(\mod{p}) \\
&= m\beta^k((\alpha^k)^a)^{-1}(\mod{p}) \\
&= m(\alpha^a)^k\alpha^{-ka}(\mod{p}) \\
&= m(\mod{p})
\end{align*}
\]
We should note here that, the ElGamal encryption is not deterministic as it uses a randomly selected integer \( k \) in the encryption process. So it's possible that a message unit could encrypt to different ciphertext units. Clearly, encryption and decryption could be performed in \( O(\log^3 p) \) since they only involve multiplication, modular exponentiation, and computing inverses modulo \( p \). The set-up step requires generating a prime \( p \) such that the DLP in \( Z_p^* \) is intractable. We already know, from setting up RSA, that generating "probable" primes of a specified size is feasible, hence all that we need do is to verify that the DLP is indeed intractable in the underlying field. This will be covered when we discuss attacks on the ElGamal cryptosystem.

### 2.6.4 Attacks on ElGamal

Attacks on the ElGamal cryptosystem mainly come from attempts to solve the DLP in finite groups. In our case, the group is the multiplicative group \( Z_p^* \). Shank's algorithm and the Pollard rho-method are exponential and therefore not practical for large \( p \). The Pohlig-Hellman algorithm requires \( p - 1 \) to be \( y \)-smooth for small \( y \) and also the prime factorization of \( p - 1 \). Otherwise, it too becomes exponential. The index-calculus method, though subexponential, becomes impractical for large \( p \). Therefore, to thwart these attacks, \( p \) should be large and \( p - 1 \) should not be \( y \)-smooth for small \( y \).

In 1984, Coppersmith [15], designed an efficient variant of the index calculus method for the fields \( \mathbb{F}_{2^m}^* \). But, it too becomes impractical for \( m \geq 512 \). A more general algorithm for the DLP over all finite fields was designed by Adleman and Demarrais [1]. It is subexponential, but for \( \mathbb{F}_{2^m}^* \), Coppersmith's algorithm is more "efficient".

### 2.6.5 The Diffie-Hellman Key Exchange Protocol

The Diffie-Hellman key exchange protocol [19], as its name suggests, is used by two users of a private-key cryptosystem (encryption and decryption rules are only known to the two users) to exchange the private encryption key they are going to use. It is presented in Figure 2.14.

At the end of the protocol, both Bob and Alice have computed the same key. The protocol is based on the Diffie-Hellman problem, Figure 2.15.
Set-up
1. Choose a "large" prime $p$ such that the DLP in $\mathbb{Z}_p$ is intractable
2. Choose a primitive element, $\alpha \in \mathbb{Z}_p^*$

Public Key: $(p, \alpha)$

Key Exchange:
1. Bob chooses $b \in \mathbb{Z}_{p-1}^*$ at random

2. Bob computes $\alpha^b \pmod{p}$ and sends it to Alice

3. Alice chooses $a \in \mathbb{Z}_{p-1}^*$ at random

4. Alice computes $\alpha^a \pmod{p}$ and sends it to Bob

5. Bob computes
   
   $K = (\alpha^a)^b \pmod{p}$

   and Alice computes
   
   $K = (\alpha^b)^a \pmod{p}$

Figure 2.14: Diffie-Hellman Key Exchange Protocol

Problem Instance: Finite group $\mathbb{Z}_p$, where $p$ is prime, $\alpha \in \mathbb{Z}_p^*$, a primitive element, $\alpha^a$, and $\alpha^b$.

Task: Compute $K = \alpha^{ab}$

Figure 2.15: The Diffie-Hellman problem
The Diffie-Hellman key exchange protocol only requires generating a large prime such that the DLP in the underlying field is intractable, as a result, the same precautions as for the ElGamal cryptosystem applies. The following theorem relates the security of the ElGamal cryptosystem to that of the Diffie-Hellman problem.

**Theorem 2.6.1** Breaking the ElGamal cryptosystem is equivalent to solving the Diffie-Hellman problem.

**Proof** First, let us recall the ElGamal encryption and decryption. The key is given by

\[ K = (p, \alpha, a, \beta) \]

where \( \beta \equiv \alpha^a \pmod{p} \), \( a \in \mathbb{Z}_p^* \) (\( a \) is secret and \( (p, \alpha, \beta) \) from the public key), and the encryption is

\[ e_K(m, k) = (y_1, y_2) = (y_1 \equiv \alpha^k \pmod{p}, y_2 \equiv m\beta^k \pmod{p}), \]

and the decryption is

\[ d_K(y_1, y_2) = y_2(y_1^a)^{-1} \pmod{p}. \]

Now, suppose we have an algorithm \( \text{ADH} \) to solve the Diffie-Hellman problem and we are given an ElGamal encryption \( (y_1, y_2) \). We will compute

\[ \text{ADH}(p, \alpha, y_1, \beta) = \text{ADH}(p, \alpha, \alpha^k, \alpha^a) = \alpha^{ka} \pmod{p} = \beta^k \pmod{p}. \]

Then, the decryption of \( (y_1, y_2) \) can easily be computed as

\[ x = y_2\beta^{k-1} \pmod{p}. \]

Suppose we now have an algorithm \( \text{AE} \) that performs ElGamal decryption. That is, given \( p, \alpha, \beta, y_1, y_2 \), it computes

\[ x = y_2(y_1^a)^{-1} \pmod{p}. \]
Now, given inputs $p, \alpha, \beta,$ and $\gamma$ we compute

$$AE(p, \alpha, \beta, \gamma, 1) = 1(\gamma^\alpha)^{-1} \pmod{p}$$

$$= \gamma^{-\alpha} \pmod{p}.$$

Computing the inverse, $AE(p, \alpha, \beta, \gamma, 1)^{-1}$, we get $\gamma^\alpha$, which is what we want. Note that $\alpha = \log_{\alpha,\beta}$. 

2.6.6 Attacks on the Diffie-Hellman Key Exchange

It is easy to see that if we could compute discrete logarithms efficiently, then we could break the Diffie-Hellman key exchange protocol by solving two instances of the DLP. Therefore, attacks on ElGamal can also be used against the Diffie-Hellman key exchange protocol. However, it is unknown whether breaking the Diffie-Hellman key exchange protocol is equivalent to computing discrete logarithms in the underlying field. In this direction, Maurer [49], has shown that under certain conditions this is possible.
Elliptic curves have been a subject of much mathematical study for the last century and there is a vast amount of literature accumulated on them. Silverman [80] and Tate and Silverman [81] are excellent sources for the theory of elliptic curves. Recently, they have found application in the areas of primality proving [5, 14, 29], integer factorization, cryptography [30, 56], pseudorandom bit generation [27], and also played a role in proving Fermat’s Last Theorem [89, 90]. In this chapter, we give a brief introduction to elliptic curves and some of their properties, necessary for understanding the rest of this thesis.

3.1 Introduction to Elliptic Curves

**Definition** (Non-Homogeneous Coordinates)

An elliptic curve, $E$, over a field $F$ is the set

$$\{(x, y) \in F \times F : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\}$$

where $a_1, \ldots, a_6 \in F$, together with the point at infinity, $\mathcal{O}$.

If the field $F = \mathbb{Z}_p$, for $p \neq 2$ or 3 prime, the curve defined in the set (3.1) can be reduced to $y^2 \equiv x^3 + ax + b$, $a, b \in \mathbb{Z}_p$, by a linear change of variables. There exist corresponding equations for the cases $p = 2$ or 3. In particular, the case $p = 2^m$, is of special interest since elliptic curve operations over $F_{2^m}$ can be efficiently implemented [52]. For the rest of the thesis we only consider
CHAPTER 3. ELLIPTIC CURVES

the case \( p > 3 \), prime. The point at infinity, \( O \), is best explained through \textit{homogeneous} coordinates. In homogeneous coordinates, the equation of the curve becomes

\[ Y^2Z \equiv X^3 + aX^2Z + bZ^3 \pmod p. \]

Points on this curve are defined as equivalence classes of triples \((X, Y, Z)\) satisfying the equation, where the equivalence relation \( \sim \) is defined as

\[(X, Y, Z) \sim (U, V, W)\]

if and only if there exists \( \lambda \in F \) such that \( X = \lambda U, Y = \lambda V \) and \( Z = \lambda W \). Then the point at infinity is represented by \((0, 1, 0)\), because this represents the only solution of the homogeneous equation for \( Z = 0 \). To go back to non-homogeneous coordinates, we use the transformation \( x = X/Z \) and \( y = Y/Z \). We also define two properties of \( E \) pertaining to our case. The \textit{discriminant} of \( E \) is \( \Delta = -16(4a^3 + 27b^2) \) and the \textit{j-invariant} is \( j = 1728 \frac{(4a)^3}{\Delta} \). The number of points on \( E \), including \( O \), is denoted by \#\( E \). Two elliptic curves over the algebraic closure of \( \mathbb{F}_p \) are said to be isomorphic if and only if they have the same \( j \)-invariant.

\textbf{Definition} An elliptic curve, \( E \), is said to be \textit{singular} if \( \Delta = 0 \), otherwise it is said to be \textit{non-singular}.

\textbf{Definition} An elliptic curve \( E_q \) over the field \( \mathbb{F}_q \) of \( q \) elements is said to be \textit{anomalous} if \( \#E_q = q \).

We are interested only in non-singular curves, so we will assume that \( 4a^3 + 27b^2 \neq 0 \) (which corresponds to the condition \( 4a^3 + 27b^2 \neq 0 \pmod p \)). We will also use the notation \( E, E_p \) or \( E_p(a, b) \) to represent an elliptic curve over \( \mathbb{Z}_p \) (\( p > 3 \)). There is a rule for adding two points on an elliptic curve \( E \) to give a third elliptic curve point. Historically, it has been called \textit{addition}, and denoted by \( + \). Together with this addition rule, the set of points on \( E \) forms an abelian group, with \( O \) serving as its identity. Elliptic curve cryptosystems are constructed based on this group.

### 3.2 Addition Rule

The addition rule is best explained geometrically. Let \( P = (x_1, y_1) \) and \( Q = (x_2, y_2) \) be two distinct points on an elliptic curve \( E \). Then \( R = (x_3, y_3) = P + Q \), is defined as follows. Let \( l \) be
the line through $P$ and $Q$; this line intersects the curve in a third point we call $I$. Then $R$ is just the reflection of $I$ in the $x$–axis, as depicted in Figure 3.1.

If $P = Q$, then the double of $P$, $R = (x_3, y_3) = 2P$, is defined as follows. Here, $kP = P + \cdots + P$, denotes the addition of a point $P$ to itself $k$ times. Let $l$ be the tangent line to the elliptic curve at $P$. This line intercepts the elliptic curve in a second point we call $I$. Then $R$ is just the reflection of $I$ in the $x$–axis, as depicted in Figure 3.2. The following algebraic formulae for adding points on an elliptic curve, $E$, can now be easily derived from the geometric description. For example, see Koblitz [29]. Let $P = (x_1, y_1), Q = (x_2, y_2) \in E$.

**Case $P = \mathcal{O}$ or $Q = \mathcal{O}$**
Figure 3.2: Geometric description of doubling an elliptic curve point
1. \(-P = \mathcal{O}\) (negative of \(P\))

2. \(P + Q = Q\)

**Case \(P \neq \mathcal{O}\) and \(Q \neq \mathcal{O}\)**

1. \(-P = -(x_1, y_1) = (x_1, -y_1)\) (negative of \(P\)) and \(P + (-P) = \mathcal{O}\)

2. if \(Q = -P\) then \(P + Q = \mathcal{O}\), otherwise

3. \(P + Q = (x_3, y_3)\) where
   \[
   x_3 = \lambda^2 - x_1 - x_2, \\
   y_3 = \lambda(x_1 - x_3) - y_1, 
   \]

   \[
   \lambda = \begin{cases} 
   \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q; \\
   \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q. 
   \end{cases} 
   \]

It can be proven that the above addition rule indeed makes the points on an elliptic curve an abelian group. But, proving associativity is not easy. We refer the reader to Tate and Silverman [81] for the proofs. The following example demonstrates the addition rule on an elliptic curve.

**Example** Let \(E\) be the elliptic curve \(y^2 = x^3 + x + 19\) over \(\mathbb{Z}_{23}\). The number of points, \(\#E\), on \(E\) can easily be determined by simply looking at each possible \(x \in \mathbb{Z}_{23}\), computing \(y^2 = x^3 + x + 19\) (mod 23) and solving for \(y\). Theorem 2.4.8 is used to test whether \(x^3 + x + 19\) (mod 23) is a quadratic residue modulo \(p(= 23)\). That is, whether \((x^3 + x + 19\) (mod 23) \(\in QR(23))\). The results of the computations are given in Table 3.1.

We see that \(\#E = 19\) (actually 18 plus the point at infinity, \(\mathcal{O}\)), and since \(\#E\) is prime, the group of points on \(E\) is cyclic and any point on \(E\), except for \(\mathcal{O}\), is a generator. For example, \(P = (4, 8)\) is a generator as shown in Table 3.2. For addition, let \(P = (3, 16)\) and \(Q = (11, 2)\), then \(R = (x_3, y_3) = P + Q\) is computed as:

\[
\lambda = \frac{2 - 16}{11 - 3} = \frac{-14}{8} = 9(8)^{-1} = 9(3) = 4 \pmod{23} 
\]

thus,

\[
x_3 = 4^2 - 3 - 11 = 2 \pmod{23} 
\]
### Table 3.1: Points on the curve $E : y^2 = x^3 + x + 19$ over $\mathbb{Z}_{23}$

<table>
<thead>
<tr>
<th>$x$</th>
<th>$x^3 + x + 19 \pmod{23}$</th>
<th>QR(23)?</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>19</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>yes</td>
<td>12,11</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>yes</td>
<td>16,7</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
<td>yes</td>
<td>8,15</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>yes</td>
<td>1,22</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>21</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>17</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>yes</td>
<td>2,21</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>21</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>17</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>14</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>4</td>
<td>yes</td>
<td>2,21</td>
</tr>
<tr>
<td>18</td>
<td>4</td>
<td>yes</td>
<td>2,21</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>12</td>
<td>yes</td>
<td>9,14</td>
</tr>
<tr>
<td>21</td>
<td>9</td>
<td>yes</td>
<td>3,20</td>
</tr>
<tr>
<td>22</td>
<td>17</td>
<td>no</td>
<td></td>
</tr>
</tbody>
</table>
Table 3.2: Multiples of $P = (4, 8)$

\[
\begin{array}{ccc}
P &=& (4, 8) \\
2P &=& (18, 21) \\
3P &=& (17, 21) \\
4P &=& (3, 16) \\
5P &=& (11, 2) \\
6P &=& (20, 9) \\
7P &=& (7, 22) \\
8P &=& (21, 20) \\
9P &=& (2, 11) \\
10P &=& (2, 12) \\
11P &=& (21, 3) \\
12P &=& (7, 1) \\
13P &=& (20, 14) \\
14P &=& (11, 21) \\
15P &=& (3, 7) \\
16P &=& (17, 2) \\
17P &=& (18, 2) \\
18P &=& (4, 15) \\
19P &=& \mathcal{O}
\end{array}
\]

and

\[y_3 = 4(3 - 2) - 16 = -12 = 11 \pmod{23}.
\]

### 3.3 Computing the number of points on an elliptic curve, $\# E$

Knowing the number of points on an elliptic curve is central to the design of elliptic curve cryptosystems, as it can often give useful information on the structure of the group, useful to cryptanalysts. If the prime $p$ is small enough, brute force methods could be used to compute $\# E$. But this becomes impractical for $p$ with around 35 digits. There is a deterministic algorithm due to Schoof [75], that computes $\# E$ in $O(\log^2 p)$ operations. Even with recent improvements to the algorithm [48, 46], the algorithm becomes impractical for $p$ with around 150 digits. The following Theorem, due to Hasse, gives bounds on $\# E$.

**Theorem 3.3.1 (Hasse's)**

*Let $\# E$ be the number of points on an elliptic curve defined over $\mathbb{Z}_p$, $p > 3$ prime. Then*

\[
|\# E - (p + 1)| \leq 2\sqrt{p}.
\]

In fact, we can still say more about the number of points on an elliptic curve. Let $E_{pr}$ be an elliptic curve over $\mathbb{F}_{pr}$, $p > 3$ prime and $r \geq 1$, and $N_r = \# E_{pr}$. Then, from Hasse's theorem, we know
that the number of points on $E_p$ over $\mathbb{F}_p (= \mathbb{Z}_p)$ is

$$\#E_p = N_1 = p + 1 - a, a^2 < 4p.$$ 

It can be shown, Silverman [80], that,

$$N_r = p^r + 1 - \alpha^r - \beta^r, r = 1, 2, \ldots,$$

where $\alpha$ and $\beta$ satisfy $1 - ax + px^2 = (1 - \alpha x)(1 - \beta x)$. In other words, it is easy to compute $N_r$ once $N_1$ is known.

**Definition** Let $\#E_q = q + 1 - t$ denote the order of an elliptic curve $E_q$ over $\mathbb{F}_q$ ($q = p^k, k \geq 1$), where $t$ is as in Hasse's theorem. Then $E_q$ is said to be *supersingular* if $p \mid t$.

It is often of interest to cryptanalysts not only to know the number of points on an elliptic curve, but also the structure of the group. The following theorem, Cassel [12], gives considerable information on the group structure of $E$.

**Theorem 3.3.2** Let $E$ be an elliptic curve over $\mathbb{Z}_p$, $p$ prime. Then, there exist $n_1$ and $n_2$ such that $E$ is isomorphic to $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$, where $n_1 n_2 = \#E, n_2 \mid n_1$, and $n_2 \mid (p - 1)$.

The above theorem says that, $E$ is either cyclic or a product of two cyclic groups. Unfortunately, computing the structure of $E$ is not always easy. However, for some elliptic curves over $\mathbb{Z}_p$, we can say more about their structure. The next two lemma's from Koyama et al. [35] illustrates this point. These lemma's are used in proving the correctness of elliptic curve cryptosystems proposed by Koyama et al. [35].

**Lemma 3.3.3** Let $p(> 3)$ be a prime, $p \equiv 2 \pmod{3}$. Then for $b \in \mathbb{Z}_p^*$, the elliptic curve $E: y^2 \equiv x^3 + b$ over $\mathbb{Z}_p$ is cyclic and $\#E = p + 1$.

**Proof** First we show that $\#E = p + 1$. Now, for $p \equiv 2 \pmod{3}$, the mapping $x \mapsto x^3$ is a permutation on $\mathbb{Z}_p$. Hence, for every $b$ there are exactly $\frac{p-1}{2}$ numbers $x \in \mathbb{Z}_p$ for which $x^3 + b \in \text{QR}(p)$. Also, for each such $x$ there are two points on $E$. This points, together with $O$ and $(-b, 0)$, we get $\#E = p + 1$. To prove that $E$ is cyclic, we assume the contrary. Then, by Theorem 3.3.2, we have that $E$ is isomorphic to $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$, where $n_1 n_2 = \#E = p + 1$ and
CHAPTER 3. ELLIPTIC CURVES

\( n_2 \mid (p - 1) \). A bit of algebra and Corollary 2.1.5 reveals that \( n_2 = 2 \) and \( n_1 \) is even. Thus, the group \( \mathbb{Z}_{n_1} \times \mathbb{Z}_2 \) must have four elements \( P \) for which \( -P = P \). But, only the points \( P = O \) and \( P = (\sqrt{(-b)}, 0) \) have \(-P = P\), since the only points on \( E \) for which \(-P = P\) are those points \((x, y)\) with \( y = 0 \) (Note that \(-1 \equiv 1 \pmod{2}\)). This contradicts the fact that \( E \) is isomorphic to a product of two cyclic groups, hence, \( E \) must be cyclic. ■

**Lemma 3.3.4** Let \( p(> 3) \) be a prime such that \( p \equiv 3 \pmod{4} \). Then, for \( a \in \mathbb{Z}_p^* \), the elliptic curve \( E : y^2 \equiv x^3 + ax \) over \( \mathbb{Z}_p \) has \( \#E = p + 1 \). Furthermore, \( E \) is cyclic if \( a \in QR(p) \), otherwise, \( E \) is isomorphic to \( \mathbb{Z}_{p-1} \times \mathbb{Z}_2 \).

**Proof** Let \( f(x) = x^3 + ax \). Since \( f(-x) = f(x) \), \( f(x) \) is an odd function. Following our discussion of algorithm SQUARE-ROOT \((p), p \equiv 3 \pmod{4} \) implies that for every \( x \in \mathbb{Z}_p^* \), exactly one of \( x \) or \( -x \in QR(p) \). Note that \(-1 \notin QR(p) \). Consider the pairs \([x, -x]\) for \( 0 < x \leq \frac{p-1}{2} \). For every such pair, either \( f(x) = f(-x) = 0 \) or \( f(x) \in QR(p) \) or \( f(-x) \in QR(p) \). In each case, there are two points on \( E \) associated with the pair \([x, -x]\), namely, \((\pm x, 0), (x, \pm \sqrt{f(x)})\) or \((-x, \pm \sqrt{-f(x)})\), respectively. This makes \( p - 1 \) points on \( E \). Adding the points \( O \) and \((0, 0)\) gives \( \#E = p + 1 \). The proof of the last claim is similar to the proof given for Lemma 3.3.3. ■

There are many analogies between the group of points on an elliptic curve, \( E \), over \( \mathbb{Z}_p \) and the multiplicative group \( \mathbb{Z}_p^* \). We list some of these in Table 3.3.

The elliptic curve analog of exponentiating by \( k \) in \( \mathbb{Z}_p^* \) is repeated addition of a point to itself \( k \) times, as seen in Table 3.3. We have seen how to compute \( g^k \pmod{p} \) efficiently by using algorithm MOD-EXP. The same algorithm can be adapted to compute \( kP \), where \( P \in E_p \), in \( O(\lg k \, \lg^3 p) \). This algorithm, MULTIPLE-ADDITION, is presented in Figure 3.3.

### 3.4 The Elliptic Curve Discrete Logarithm Problem (ECDLP)

The DLP, originally posed over \( \mathbb{Z}_p^* \), can in fact be posed over any group. Since the points on an elliptic curve, \( E_p \), form a group (an abelian group, to be more specific), they provide a perfect setting for posing the DLP. This version of the DLP, called the elliptic curve discrete logarithm problem (ECDLP), is presented in Figure 3.4.
### Table 3.3: Notational correspondence between $Z_p^*$ and $E_p$

<table>
<thead>
<tr>
<th>Group</th>
<th>Multiplicative Group</th>
<th>Elliptic Curve Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elements</td>
<td>$Z_p^*$</td>
<td>$E$ or $E_p$</td>
</tr>
<tr>
<td></td>
<td>${1, 2, \ldots, p-1}$</td>
<td>Points $(x, y) \in E$ plus $\mathcal{O}$</td>
</tr>
<tr>
<td>Operation</td>
<td>Multiplication modulo $p$</td>
<td>Addition modulo $p$</td>
</tr>
<tr>
<td></td>
<td>Elements: $g, h$</td>
<td>Elements: $P, Q$</td>
</tr>
<tr>
<td></td>
<td>Multiplication: $gh$</td>
<td>Addition: $P + Q$</td>
</tr>
<tr>
<td></td>
<td>Inverse: $h^{-1}$</td>
<td>Negative: $-P$</td>
</tr>
<tr>
<td></td>
<td>Division: $g/h$ or $g(h)^{-1}$</td>
<td>Subtraction: $P - Q$</td>
</tr>
<tr>
<td></td>
<td>Exponentiation: $g^a$</td>
<td>Multiple: $aP$</td>
</tr>
<tr>
<td>Discrete logarithm problem</td>
<td>Given $g$ and $h = g^a$, find $a$</td>
<td>Given $P$ and $Q = aP$, find $a$</td>
</tr>
</tbody>
</table>

**Input:** $k, P$ where $k, \in Z_p, P \in E_p$

**Output:** $kP$

Let $k_{l-1} \cdots k_1k_0$ be the binary representation of $k$.

1. $Q \leftarrow \mathcal{O}$

2. For $i \leftarrow l - 1$ downto 0 Do
   
   \[
   \begin{align*}
   &\{ \\
   &Q \leftarrow 2Q \\
   &Q \leftarrow Q + k_iP \\
   \}
   \end{align*}
   
3. Return $(Q)$

**Figure 3.3: Algorithm MULTIPLE-ADDITION**
Problem Instance: An elliptic curve $E_p$, a point $P \in E_p$ of order $N$ and a point $Q \in E_p$.

Task: Find $a \in Z_N$ (if it exists) such that $Q = aP$.

Figure 3.4: The Elliptic Curve Discrete Logarithm Problem (ECDLP)

The ECDLP has received much attention over the past decade from leading scholars around the world, and no significant weaknesses have been reported. In fact, it is also conjectured to be harder than the DLP and the factoring problem [88].

3.5 Elliptic Curves Over $Z_n$

We now consider elliptic curves over the ring $Z_n$, where $n = pq$ is an odd squarefree integer, for two distinct primes $p$ and $q$ (both $> 3$). An elliptic curve, $E_n(a, b)$, over $Z_n$ is defined to be the set of points $(x, y) \in Z_n \times Z_n$ such that $y^2 \equiv x^3 + ax + b \pmod n$, together with the point at infinity, $O$. An addition operation can be defined on the points of $E_n(a, b)$ in the same way as addition on $E_p(a, b)$ by simply replacing operations in $Z_p$ with operations in $Z_n$. But, since division is not always defined in $Z_n$, points on $E_n(a, b)$ does not form a group. However, these problems can be overcome so as to allow us to construct elliptic curve cryptosystems on $E_n(a, b)$. By Theorem 2.3.9 (more specifically, Corollary 2.3.10), any $c \in Z_n$ can be uniquely represented by a pair of integers $[c_p, c_q]$ where $c_p \in Z_p$ and $c_q \in Z_q$. Thus, every point $P = (x, y) \in E_n(a, b)$ can be uniquely represented by a pair of points

$$[P_p, P_q] = [(x_p, y_p), (x_q, y_q)]$$

such that $P_p \in E_p(a, b)$ and $P_q \in E_q(a, b)$, with the convention that $O$ is represented by $[O_p, O_q]$, where $O_p$ and $O_q$ are the points at infinity on $E_p(a, b)$ and $E_q(a, b)$, respectively. It is now easy to easy that when it is defined, then addition operation on $E_n(a, b)$ is equivalent to the componentwise addition operation on $E_p(a, b) \times E_q(a, b)$.

Example Let $p = 7, q = 5$ (we use small primes just for illustration), and $n = pq = 35$. Let $E$ be
Consider the addition of points $P = (6, 34)$ and $Q = (11, 16)$. Since $P \neq Q$, we have to compute the inverse of $11 - 6$ in $\mathbb{Z}_{35}$. But $\text{GCD}(5, 35) = 5 > 1$. Hence, the inverse does not exist, and in fact we have found one of the factors of $n$. A simple calculation can show that the only points on the curve (considered over $\mathbb{Z}_5$) are those with $x = 1$ and $x = 2$. Similarly, when considered over $\mathbb{Z}_7$, the only points on the curve are those with $x = 1, x = 4$, and $x = 6$. Of course, the point at infinity is, by definition, always included. A cross-product of these points (and using the Chinese Remainder Theorem) gives precisely the points on Table 3.4.

Note that the addition operation on $E_n(a, b)$ is undefined precisely when one of $P_p$ and $P_q$ is $\mathcal{O}$. For large $p$ and $q$, it is highly (probability of it is very small) unlikely that the addition of two points on $E_n(a, b)$ is undefined. Note that if it were not negligible, trying to perform the undefined operation (an inversion) would give a nontrivial factor of $n$, and this would be an effective factoring algorithm, which is assumed not to exist. Although we do not use the properties of a finite group directly, the following lemma (from Koyama et al. [35], but with our proof) give us a property of $E_n(a, b)$ which is similar to that of a finite group.

**Lemma 3.5.1** Let $E_n(a, b)$ be an elliptic curve over $\mathbb{Z}_n$ such that $\text{GCD}(4a^3 + 27b^2, n) = 1$ and $n = pq (p, q (> 3) \text{prime})$. Let

$$N_n = \text{LCM}(\#E_p(a, b), \#E_q(a, b)).$$
CHAPTER 3. ELLIPTIC CURVES

Then, for any \( P \in E_n(a, b) \), and any \( k \in \mathbb{Z}_n \),

\[(k \cdot N_n + 1)P = P \text{ over } E_n(a, b).\]

**Proof** Since \( N_n = \text{LCM}(\#E_p(a, b), \#E_q(a, b)) \), we have, by the division algorithm

\[ N_n = u \cdot \#E_p(a, b) \text{ and } N_n = v \cdot \#E_p(a, b), \]

where \( u, v \in \mathbb{N}_0 \). Therefore, \( N_nP = u \cdot \#E_p(a, b)P = u \cdot (\#E_p(a, b)P) = O_p \). Similarly, \( N_nP = v \cdot \#E_q(a, b)P = O_q \). Hence it follows, by convention, that for any \( k \in \mathbb{Z}_n \)

\[ k \cdot N_nP = O \]

and therefore \( (k \cdot N_n + 1)P = P \text{ over } E_n(a, b). \)

It is on this curves that RSA-type elliptic curve cryptosystems are constructed, deriving their security, like RSA, from the factoring problem.
Chapter 4

Elliptic Curve Cryptosystems

The use of elliptic curves over finite fields for public key cryptography was first suggested by Koblitz [30] and Miller [56]. They did not, at that time, invent new public key cryptosystems, but rather presented analogs of the ElGamal cryptosystem and the Diffie-Hellman key exchange over elliptic curves, respectively. The security of these systems is derived from the difficulty of solving the elliptic curve versions of the discrete logarithm problem and the Diffie-Hellman problem. In 1991, Menezes et al. [35], proposed the use of elliptic curves over the ring $Z_n$, where $n$ is a product of two "large" and distinct primes. The security of the public key cryptosystems the proposed is derived from the difficulty of factoring the modulus $n$. In this chapter, we discuss elliptic curve public key cryptosystems over the field $Z_p$ and the ring $Z_n$. However, since there are many cryptosystems proposed, we shall only discuss a few we hope will be representative of the field. We first discuss elliptic curve cryptosystems over $Z_p$.

4.1 Elliptic Curve Cryptosystems Over $Z_p$

In this section, we present analogs of the ElGamal cryptosystem and the Diffie-Hellman key exchange. The ElGamal cryptosystems can be implemented in any group where the discrete logarithm problem is intractable. In fact, the group need not be abelian (of course the subgroup used is cyclic). The analog of the ElGamal cryptosystem is presented in Figure 4.1.
CHAPTER 4. ELLIPTIC CURVE CRYPTO SYSTEMS

Set-up

1. Choose a "large" prime $p$ and an elliptic curve $E_p$, such that the ECDLP in $E_p$ is intractable.

2. Choose a point $P \in E_p$ (called the base point).

3. Choose $a \in \mathbb{Z}_p^*$, and compute $Q = aP$.

4. Choose an invertible function $f$ such that, given $m \in \mathbb{Z}_p^*$, $f(m) \in E_p$. That is, given $m \in \mathbb{Z}_p^*$, the function $f$ embeds $m$ on $E_p$, deterministically.

$\mathcal{P} = \mathbb{Z}_p^*, \mathcal{C} = E_p \times E_p, \mathcal{K} = \{(p, E_p, a, P, Q, f): Q = aP\}$

Public Key: $(p, E_p, P, Q, f)$

Private Key: $(a)$

Encryption

Given key $K = (p, E_p, a, P, Q, f)$ and message $m \in \mathcal{P}$, define

$$e_K(m, k) = (c_1, c_2) = (kP, P_m + kQ)$$

where $k \in \mathbb{Z}_p^*$ is a secret random number $P_m = f(m)$

Decryption

Given ciphertext $c = (c_1, c_2) \in \mathcal{C}$ define

$$d_K(c) = f^{-1}(c_2 - ac_1)$$

Figure 4.1: The Analog of the ElGamal Public Key Cryptosystem
CHAPTER 4. ELLIPTIC CURVE CRYPTOGRAMS

The correctness is easy to verify since,

\[ c_2 - ac_1 = P_m + kQ - a(kP) = P_m + k(aP) - a(kP) = P_m. \]

Now, since the function \( f \) is invertible, we can derive the original message, \( m \), by computing \( m = f^{-1}(P_m) \).

Encryption and decryption can be performed in \( O(lg(max(a,k))lg^3p) \). If we knew the size of the subgroup generated by \( P \), then this bound could be replaced by \( O(lg^4p) \). The difficulty lies in finding a curve \( E_p \) in which the ECDLP is intractable. However, by using some byproducts of the elliptic curve primality proving algorithm of Atkin and Morain [5], Morain [59] showed that such curves could be constructed in \( O(lg^6p) \) (from heuristic analysis). The following papers also discuss the construction of elliptic curves suitable for cryptosystems [31, 32, 57, 58, 13, 41].

There are some practical difficulties in implementing this cryptosystem. Firstly, it has a message expansion factor of 4 (4 elements of \( Z_p^* \) are generated from 1 (plaintext)). When the cryptosystem is implemented in \( Z_p^* \), it only has a message expansion factor of 2. The second problem is that, we require a function (algorithm) to embed plaintext on \( E_p \) before encrypting it. Unfortunately, there is no known convenient method for deterministically generating points on \( E_p \). In order to overcome this shortcomings, Menezes and Vanstone [53], invented a more efficient variation, presented in Figure 4.2.

The Menezes-Vanstone cryptosystem only uses the elliptic curve to "mask" the plaintext, as opposed to embedding it on the curve. Plaintext can be any element of \( Z_p^* \times Z_p^* \), thus vastly increasing the plaintext space. It has a message expansion factor of 2, the same as in the original ElGamal cryptosystem over \( Z_p^* \). Its correctness can easily be verified, and has the same time complexity as first variation. Figure 4.3 and Figure 4.4 show the results encrypting the message "public" using Ame Louis'\(^1\) public key \( p = 33333331 \), \( E_p: y^2 \equiv x^3 + 5202x + 5379352, P = (33331773, 8538162), a = 16839, Q = (27518663, 635646) \).

That is, the message is intended for Ame Louis. Notice how quickly the ciphertext becomes cumbersome.

Next, we discuss the Diffie-Hellman analog, presented in Figure 4.5.

---

\(^1\)See an implementation of the Menezes-Vanstone scheme at http://www.cs.mcgill.ca/~nkgautz/crypt.html
CHAPTER 4. ELLIPTIC CURVE CRYPTO SYSTEMS

Set-up

1. Choose a "large" prime $p$ and an elliptic curve $E_p$, such that $E_p$ contains a cyclic subgroup $H$, in which the ECDLP in $E_p$ is intractable.

2. Choose a point $P \in E_p$ (called the base point).

$$ P = \mathbb{Z}_p^* \times \mathbb{Z}_p^*, C = E_p \times \mathbb{Z}_p^* \times \mathbb{Z}_p^*, K = \{(p, E_p, a, P, Q,): Q = aP, a \in \mathbb{Z}_p^*\}, $$

$$ H = \langle P \rangle $(cyclic subgroup generated by $P$)

Public Key: $(p, E_p, P, Q)$

Private Key: $(a)$

Encryption

Given key $K = (p, E_p, a, P, Q)$ and message $m = (m_1, m_2) \in C$, define

$$ e_K(m, k) = (y_0, y_1, y_2) $$

where

$$ k \in \mathbb{Z}_p^* \text{ is a secret random number,} $$

$$ (c_1, c_2) = kQ, $$

$$ y_1 = c_1m_1 \pmod{p}, $$

and

$$ y_2 = c_2m_2 \pmod{p}. $$

Decryption

Given ciphertext $c = (y_0, y_1, y_2) \in C$ define

$$ d_K(c) = (y_1c_1^{-1}(\pmod{p}), y_2c_2^{-1}(\pmod{p})), \text{ where } ay_0 = (c_1, c_2) $$

Figure 4.2: The Menezes-Vanstone Cryptosystem
Figure 4.3: Menezes-Vanstone encryption
<table>
<thead>
<tr>
<th>CipherText</th>
<th>PublicKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>615725 8078534 32377558 28505026 23530021 31843338 12083137 12987531 29722125 17106630 21721869 1223307</td>
<td></td>
</tr>
</tbody>
</table>

Send To

Thandile Mpopi
Tlotlo Neo

| Clear | Send | Decode |

Figure 4.4: Menezes-Vanstone decryption
Set-up

1. Choose a finite field $\mathbb{F}_q$ and an elliptic curve $E_p$ defined over it

2. Choose a point $P \in E_q$ (of large order)

Public Key: $(q, E_q, P)$

Key Exchange:

1. Bob chooses $b \in \mathbb{Z}_q^*$ at random

2. Bob computes $bP$ and sends it to Alice

3. Alice chooses $a \in \mathbb{Z}_q^*$ at random

4. Alice computes $aP$ and sends it to Bob

5. Bob computes

$$K = b(aP)$$

and Alice computes

$$K = a(bP)$$

Figure 4.5: The Elliptic Curve Analog of the Diffie-Hellman Key Exchange Protocol
After executing the protocol, both Bob and Alice have computed the same key, $K = (ab)P$. Notice that, as in the original protocol, we do not need to know the order of the base point $P$, nor do we need to know that it is a generator. The security of this system is derived from the analog of the Diffie-Hellman problem. That is, given $U = bP$ and $V = aP$, compute $Q = (ab)P$, where $P$ is a point on an elliptic curve $E_q$. As of present, there is no known algorithm for solving this problem other than solving two instances of the ECDLP.

### 4.2 Elliptic Curve Cryptosystems Over $\mathbb{Z}_n$

In this section we describe cryptosystems proposed by Koyama et al. [35] and Meyer et al. [54] over the ring $\mathbb{Z}_n$. First we discuss the KMOV-scheme of Koyama et al.. This scheme is presented in Figure 4.6. The KMOV-scheme utilizes Lemma 3.3.3. We note here that the curves used are supersingular and care should be exercised if the KMOV-scheme is to be used. See comments on algorithm RANDOM-CURVE-PRIME in Section 4.5.

The correctness of the algorithm easily follows from the facts that $ed \equiv 1 \pmod{N_n}$ and that the addition formula is independent of both $a$ and $b$ while the doubling formula is independent of $b$. Encryption and decryption can clearly be performed in polynomial time since only elliptic curve arithmetic is performed. Note that the minimum possible value for $e$ is 5 because $2 \mid N_n$ and $3 \mid N_n$ (from Lemma 3.3.3). Also, note that the KMOV-scheme is not based on a single group but on a large class of groups (with the same order), and each curve is determined by the plaintext. The security of the system is derived from the factoring problem, however, like in the RSA system, it is not known whether breaking the system is equivalent to factoring. Demytko [18] proposed a scheme which has very little restriction on the types of elliptic curves and types of primes used, and has a message expansion factor of 1. However, this flexibility comes at the cost of having to use Schoof's algorithm to compute the order of the curve.

Let $P$ be a point on an elliptic curve $E$, then $x(P)$ and $y(P)$ would represent the $x$-coordinate and $y$-coordinate of the point $P$, respectively. Also, $\text{type}(x) = 1$ if $J(\frac{x}{P}) = +1$ and $\text{type}(x) = 0$ if $J(\frac{x}{P}) = -1$, where $J(\frac{x}{P})$ is the Jacobi symbol. We also denote the least significant bit of $x$ by $\text{lsb}(x)$. We are now ready to discuss the Meyer et al. scheme. It is presented in Figure 4.7.
Set-up

1. Choose large primes \( p \) and \( q \) such that \( p \equiv q \equiv 2 \pmod{3} \)

2. Compute \( n = pq \) and \( N_n = \text{LCM}(\#E_p(0, b), \#E_q(0, b)) \) where \( b \) is to be determined by the plaintext to be transmitted.

3. Choose \( e \) such that \( \gcd(e, N_n) = 1 \).

\( \mathcal{P} = \mathbb{Z}_n \times \mathbb{Z}_n, \mathcal{C} = E_n(0, b), \mathcal{K} = \{(p, q, N_n, d) : d \equiv e^{-1}(\pmod{N_n})\} \)

Public Key: \((n, e)\)

Private Key: \((d, p, q, N_n)\)

Encryption

Given key \( K = (p, q, N_n, d) \) and message \( m = (m_x, m_y) \in \mathcal{P} \), define

\[ e_K(m) = e \cdot m \text{ over } E_n(0, b), \ m \in E_n(0, b) \text{ and } b \equiv m_y^2 - m_x^3(\pmod{n}) \]

Decryption

Given ciphertext \( c \in \mathcal{C} \) define

\[ d_K(c) = d \cdot c \text{ over } E_n(0, b) \]

Figure 4.6: The KMOV-scheme
Set-up

1. Choose large primes p and q such that $p \equiv q \equiv 11 \pmod{12}$

$P = \mathbb{Z}_n^*, C = E_n \times \mathbb{Z}_n^* \times \mathbb{Z}_2^* \times \mathbb{Z}_2^*, K = \{(p, q, n) : n = pq \text{ and } p \equiv q \equiv 11 \pmod{12}\}$

Public Key: $(n)$
Private Key: $(p, q)$

Encryption

Given key $K = (p, q, n)$ and message $m \in P$, define

$$e_K(m, \lambda) = (E_n(a, b), x(2 \cdot P), \text{type}(y(2 \cdot P)), \text{lsb}(y(2 \cdot P)))$$

where

$$\lambda \in \mathbb{Z}_n^* \text{ is chosen at random,}$$

$$P = (m^2, \lambda m^3),$$

$$a = \lambda^3 \text{ and } b = (\lambda^2 - 1)m^6 - am^2$$

Decryption

Given ciphertext $c = (E_n(a, b), x_Q, t, l)$ define

$$d_K(c) = (y(P_1))^3(x(P_1))^{-4}a^{-1}$$

where

$$Q = (x_Q, y_Q \equiv \sqrt{x_Q^3 + ax_Q + b} \pmod{n})$$

such that $\text{type}(y_Q) = t$ and $\text{lsb}(y_Q) = l$, and $I = \{1 \leq i \leq s : a^2 = (y(P_i))^6(x(P_i))^{-9}\}$ and $P_i \in E_n(a, b), 1 \leq i \leq s$ such that $2 \cdot P_i = Q$
In this scheme the Jacobi symbol and the least significant bit are used to identify the proper square root of $x \in \mathbb{Z}_n^*$, since by the Chinese Remainder Theorem, each square has exactly four square roots. Some remarks are in order for this scheme.

**Remark**

1. The encryption step may fail if $\text{GCD}(4a^3 + 27b^2, n) > 1$. In which case $4a^3 + 27b^2 \equiv 0 (\text{mod } n)$ or a factor of $n$ has been found. However, as already indicated, the probability of this event is negligible.

2. The decryption step requires computing all points $P_i$ such that $2 \cdot P_i = Q$. This involves finding roots of a degree 4 polynomial in $\mathbb{Z}_q$ and $\mathbb{Z}_p$.

3. The set $I$ must have only one element, and as such, $P_I$ denotes the point with the index in $I$.

In fact, Meyer et al. [54] showed that the probability that $|I| > 1$ and $n$ can not be factored is at most $\frac{118^2}{n-1}$. Since $n$ is large, the probability is negligible.

The correctness of the cryptosystem follows directly from the fact that $2 \cdot P = Q$ and $a^2 = (y(P_i))^6(x(P_i))^{-9}$ (from the encryption step). The encryption step is clearly polynomial and the decryption step can be done in $O(\log^3 n)$ probabilistic time. This is probabilistic as a randomized root finding algorithm has to be employed, see for example, Shoup [79]. The decryption step, compared to the KMOV-scheme decryption, seems to be a bit more involved.

Other RSA-like cryptosystems have been proposed based on singular cubic curves $y^2 + axy \equiv x^3 (\text{mod } n)$, Koyama [34], and curves of smooth order, Vanstone and Zuccherato [86]. Koyama's analysis of his cryptosystems indicate that they are twice as fast as RSA, whereas the other systems are slower than RSA.

### 4.3 Attacks on Elliptic Curve Cryptosystems

First, we discuss attacks on elliptic curve cryptosystems whose basis for security is the ECDLP. The best algorithms that are known for solving the ECDLP are the square root methods that apply to any finite group $G$, and have a running time that is proportional to $\sqrt{p}$, where $p$ is the square root of the
largest prime factor of \(| G |\). These are just the analogs of the algorithms for the DLP. Hence, to guard against these attacks, the group order should be prime or \(| G | - 1\) should not be \(y\)-smooth for small \(y\). However, Miller [56] argues that, it is unlikely that the index calculus methods could be extended to the ECDLP, leaving only exponential algorithms for attacking the ECDLP. In 1993, Menezes et al. [51] invented an algorithm, MOV-reduction, that could use the Weil pairing [80] to reduce the ECDLP in a curve \(E_q\) over the field \(F_q\) to the DLP in a suitable extension field \(F_{q^k}\) of \(F_q\). The DLP can then be solved by using the subexponential index calculus method. But the reduction method is only applicable to supersingular curves. Hence, only non-supersingular curves should be used.

Another class of curves to avoid are the anomalous curves. In 1997, Smart [83] showed that anomalous are not safe for use in cryptosystems by describing a linear time algorithm to solve the DLP under the assumption that one knows the order of the group, which by definition is equal to the order of the underlying group.

Currently, the most effective algorithm against the ECDLP seem to be the Pollard rho-method, which takes about \(\sqrt{\frac{n!}{2}}\) (\(n\) is the order of the group generated by the base point) elliptic curve addition steps [65]. In 1993, van Oorschot and Wiener [85] showed how to parallelize the Pollard rho-method so that if \(r\) processors are used, then the expected number of steps taken by each processor before a single discrete logarithm is obtained is \(\frac{\sqrt{n!}}{r}\). Recently, a distributed implementation of the Schoof-Elkies-Atkins algorithm [48] was used by Lercier and Joux [47] to compute the number of points on an elliptic curve over \(F_{2^{1663}}\). This shows the power of distributed computing in solving this seemingly intractable problem.

In the case of RSA-type elliptic curve cryptosystems, all attacks on the original RSA cryptosystem are applicable, since they too derive their security from the factoring problem. Against the Hastad attack, Kurosawa et al. [37] and Koyama [38] have shown that RSA-type elliptic curve cryptosystems are more secure.

### 4.4 Speeding Up Elliptic Curve Computations

Elliptic curve cryptosystems, being public-key cryptosystems, suffer from the same deficiencies affecting all other public-key cryptosystems. Specifically, they have relatively low bandwidth (slow
In fact, this is the reason why up to now, conventional cryptosystems are still preferred over public-key cryptosystems for bulk data encryption. Public-key cryptosystems are relegated for use in digital signatures [69, 84] (as opposed to hand written signatures) and distribution of secret keys for use in conventional cryptosystems [68, 84]. However, it has not been proven that low bandwidth is a necessary characteristic of public-key cryptosystems. But for now it seems that this is true since no public-key cryptosystems has achieved the speed of conventional cryptosystems like the Data Encryption Standard (DES) (see Stinson [84] for more information on DES). A lot of research has been done and a lot written on improving public-key cryptosystems. In this section we give a summary of methods used to improve the speed of elliptic curve cryptosystems.

Speeding up elliptic curve operations can be divided in to three levels (or categories), as can be seen in Figure 4.8.

At level 0, we are mainly concerned with speeding up primitive operations: addition, subtraction, multiplication, and computing greatest common divisors and inverses modulo p. This requires the use of fast algorithms to perform the primitive operations modulo p. For example, Jebelean's
algorithm can be employed to perform long integer division in a field to speed up computing GCD's and computing inverses [25]. Jebelean's algorithm is about two times slower than Karatsuba multiplication (a practical algorithm to perform long integer multiplication and asymptotically fast). However, it is still quadratic in the worst case. Weber [87], describes an accelerated integer GCD algorithm. Level 1 is composed mainly of computing point addition and doubling on the curve efficiently. In particular, the goal is to minimize the number of inversions performed as they are costly. One way of doing this on elliptic curve was shown by Menezes [52], Beth and Schaeffer [7], and Schroeppel et al. [77] in the fields $\mathbb{F}_{2^n}$ using homogeneous coordinates. Recall that in homogeneous coordinates the equation of the curve becomes

$$Y^2Z \equiv X^3 + aX^2Z + bZ^3 \pmod{p}.$$

Let $P_1 = (X_1, Y_1, Z_1)$ and $P_2 = (X_2, Y_2, Z_2)$ be on an elliptic curve $E$ over $\mathbb{F}_p$ ($p > 3$). The addition formulae for adding computing $P_3 = (X_3, Y_3, Z_3) = P_1 + P_2, P_1 \neq P_2$ now becomes [36]

$$X_3 = VA$$
$$Y_3 = U(V^2X_1Z_2 - A) - V^3Y_1Z_2$$
$$Z_3 = V^3Z_1Z_2$$

where $U = Y_2Z_1 - Y_1Z_2, V = X_2Z_1 - X_1Z_2, A = U^2Z_1Z_2 - V^2T, \text{ and } T = X_2Z_1 + X_1Z_2$. The doubling formulae becomes

$$X_3 = 2SH$$
$$Y_3 = W(4F - H) - 8E^2$$
$$Z_3 = 8S^3$$

where $S = Y_1Z_1, W = 3X_1^2 + aZ_1^2, E = Y_1S, F = X_1E, \text{ and } H = W^2 - 8F$. By a careful counting of the multiplications performed (ignoring the multiplication of a point by a small constant), one can show that we require 15 multiplications when $P_1 \neq P_2$ and 12 multiplications otherwise. These scheme only uses addition and multiplication to add two points on a curve. Only one inversion is required at the end to get a unique representation of the point (non-homogeneous
Input: $k, P$ where $k, P \in \mathbb{Z}_p, P \in E_p$

Output: $kP$

Let $k = k_i 2^l + k_{i-1} 2^{l-1} + \cdots + k_1 2^1 + k_0$ be the modified signed-digit representation of $k$, where $k_i \in \{-1, 0, 1\}, 0 \leq i \leq l$

1. $Q \leftarrow k_i P$

2. For $i \leftarrow l - 1$ downto 0 Do

   \{ 
   \begin{align*}
   Q & \leftarrow 2Q \\
   Q & \leftarrow Q + k_i P
   \end{align*}
   \}

3. Return $(Q)$

Figure 4.9: The modified signed-digit (MSD) algorithm for computing $kP$

coordinates). Of course, there is a memory trade-off in this scheme as storage space is required for intermediate results. Note that addition and subtraction of points on an elliptic curve are equivalent. Another speed up option is to speed up inverse computations. Schroeppel et al. [77] give a fast algorithm to compute inverses in $\mathbb{F}_{2^{165}}$. How this impacts inverse computations in other fields remains to be seen.

At level 2, our goal is to minimize the number of point addition and doublings used to compute $kP$. Recall from our addition formulae for curves over $\mathbb{F}_p (p > 3)$ that, when $P \neq Q$, we perform 2 multiplications and 1 inversion, and when $P = Q$, we perform 3 multiplications and 1 inversion. Now, computing $kP$ using algorithm MULTIPLE-ADDITION, we use $\lfloor \lg k \rfloor$ doublings and $\omega(k) - 1$ additions, where $\omega(x)$ is the number of nonzero bits in the binary representation of $x$. The following algorithm, from Laih and Kuo [39], presented in Figure 4.9, improves on this by using the fact already mentioned that, subtraction of points on an elliptic curve is as easy as addition, to reduce the number of additions performed on computing $kP$.

It is easy to see that the algorithm uses $l$ doublings and $\omega(k) - 1$ additions on $E_p$, where $\omega(k) =$
The goal now is to minimize $\omega(k)$ so as to decrease the number of additions performed. This stems from the fact that the modified signed-digit representation (addition-subtraction chain) of an integer is not unique. Finding the minimum weight representation of an integer has been studied by several researchers. Jedwab and Mitchell \cite{26} designed an algorithm to find a minimum weight representation of any integer $k$. It is presented in Figure 4.10.

Notice that the Jedwab and Mitchell algorithm returns a sparse representation. That is, no two adjacent elements are nonzero.

**Example** For $k = 15$ (01111 in binary), the algorithm returns $1000(-1)$ as the minimum MSD representation of 15 after just one iteration of the while loop. This corresponds to computing $15P = (2(2(2(2(2P)))))) - P$. The binary method would compute $15P = (P + 2(P + 2(P + (2P))))$.

Morain and Olivos \cite{60} used the MSD representation of $k$ to compute $kP$ using addition and subtraction operations. They construct two integers $k_-$ and $k_+$ such that $k = k_+ - k_-$ and computing $k_+P$ and $k_-P$ require less operations than that of $kP$. They found their first algorithm to be relatively 8.33% faster than the ordinary binary algorithm. Laih and Kuo \cite{39} combined Jedwab and Mitchell's algorithm with the M-ary algorithm of Koyama and Tsuruoka \cite{36} to speed up computations of $kP$. The M-ary \cite{36} algorithm consists of four phases

1. Finding a minimum weight MSD representation of $k$.
2. Splitting the representation into segments (windows).
3. Computing all the segments.
4. Concatenating all the segments.

Of course, Laih and Kuo's algorithm sacrifices storage space to achieve significant speed up since precomputations of $dP$ for small values of $d$ are performed and stored. Their algorithm is presented in Figure 4.11.

They found their algorithm to be 12% faster on average than the ordinary addition-subtraction chains. Bos and Coster \cite{9} and Koblitz \cite{32} also discuss these methods. All of these algorithms
**Input:** $k \in \mathbb{N}_0$

**Output:** $\text{MSD}(k)$ (minimum weight MSD representation of $k$)

1. $k \leftarrow m_r2^r + m_{r-1}2^{r-1} + \cdots + m_12^1 + m_0$ (ordinary binary representation of $k$)

2. While (consecutive pairs of nonzero elements remain) Do
   
   { 
   $s \leftarrow$ least integer for which $m_s$ and $m_{s+1}$ are nonzero
   
   if $(m_s \neq m_{s+1})$ then
     
     { 
     $m_s \leftarrow -m_s$
     
     $m_{s+1} \leftarrow 0$
     
     }
   
   else
     
     { 
     $t \leftarrow$ least integer for which $m_t \neq m_s$ and $m_{t-1} = m_{t-2} = \cdots = m_s$
     
     if $(m_t = 0)$ then
       
       $m_t \leftarrow m_s$
     
     else
       
       $m_t \leftarrow 0$
       
       $m_s \leftarrow -m_s$
     
     For $i \leftarrow s + 1$ to $t - 1$ Do
       
       $m_i \leftarrow 0$
     
     }
   
   }

3. Return $\text{MSD}(k)$

Figure 4.10: Jedwab and Mitchell minimum MSD algorithm
**Input**: $k, P$ where $k \in \mathbb{Z}_p, P \in E_p$

**Output**: $kP$

Let $k = m_r 2^r + m_{r-1} 2^{r-1} + \cdots + m_1 2^1 + m_0$ be the minimum MSD representation of $k$, where $m_i \in \{-1, 0, 1\}, 0 \leq i \leq r$.

1. Choose an appropriate window size $w \leq r + 1$ and precompute all elements of the form $dP$ such that $-2^w \leq d \leq 2^w$, where $d$ is a sparse MSD representation.

2. Let $t = r - w + 1$ and let $\tilde{m} = \sum_{j=0}^{w-1} m_{t+j} 2^j$ be a span of $w$ bits from $k$. Find $s$ and $d$ such that $\tilde{m} = 2^s d$ where $d$ is odd. Set $Y = 2^s(dP)$ using one table look-up followed by $s$ doublings.

3. Stop if $t = 0$. Otherwise, if $m_{t-1} = 0$ the set $t \leftarrow t - 1$, $Y \leftarrow 2Y$ and repeat step 3.

4. If $t \geq w$, then set $t \leftarrow t - w$, otherwise set $w \leftarrow t$ and $t \leftarrow 0$.

5. Compute $\tilde{m}, s$, and $d$ as in step 2 and set $Y \leftarrow 2^s(2^w \cdot sY + dP)$ and go to step 3.

6. Return $(Y)$

---

Figure 4.11: Laih and Kuo's algorithm for computing $kP$
can benefit from specially built processors to speed the computations. Menezes and Vanstone [53], have noted that using optimal normal bases, Mullin et al. [62], arithmetic in the field $\mathbb{F}_{2^n}$ can be significantly sped up.

The usually relatively high message expansion factor of elliptic curves can be lowered if instead of transmitting the $x$ and $y$ coordinates of a point, one transmits only the $x$ coordinate and a single bit from the $y$ coordinate. Since, as explained in Menezes and Vanstone [53], the $y$ coordinate can be recovered from that information. This would, for example, lower the message expansion factor of the Menezes-Vanstone cryptosystem from 2 to $\frac{3}{2}$.

### 4.5 Generating Computationally Secure Curves

In this section we consider the construction of elliptic curves suitable for use in elliptic curve cryptosystems. We only consider curves over $\mathbb{Z}_p(p > 3)$ and refer the reader to Lercier [45] and Beth and Schaefer [7] for curves over $\mathbb{Z}_{2^n}$. To avoid attacks from Shanks’s algorithm, the Pollard rho-method, and the Pohlig-Hellman algorithm, the order of the subgroup generated by the base point $P$ (cryptosystems are implemented over the cyclic subgroup generated by the base point) should be large and not be $y$-smooth for small $y$. Of course, supersingular and anomalous curves should be avoided. The first algorithm we consider, RANDOM-CURVE, is due to Koblitz [29]. It is presented in Figure 4.12.

Koblitz’s random curve selection method only makes sure that the curve is not singular, otherwise, everything is left to chance. This, is clearly not adequate if we want to have confidence in the security of our cryptosystem. But, despite this fact, it can still be used for generating curves for the Diffie-Hellman and ElGamal schemes, as we do not need to know the number of points on the curve and the order of the subgroup generated by the base point. Such schemes, however, should not be used for encrypting sensitive data. To address these shortcomings, Koblitz [33], designed a procedure to generate curves immune against the exponential algorithms (Shanks’s, Pollard rho-method, Pohlig-Hellman) and the MOV-reduction. This algorithm, RANDOM-CURVE-PRIME, is presented in Figure 4.13.
Input: A large prime $p$

Output: $(a, b)$ such that $E_p(a, b)$ is an elliptic curve over $\mathbb{Z}_p$ and $P \in E_p(a, b)$

1. Repeat
   
   Randomly choose $a, x, y \in \mathbb{Z}_p$
   
   $b \leftarrow y^2 - (x^3 + ax) \pmod{p}$
   
   Until $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$

2. Set base point $P \leftarrow (x, y)$ and $E_p(a, b)$ to $y^2 = x^3 + ax + b$

3. Return $((a, b), P)$

Figure 4.12: Koblitz's random curve selection method, RANDOM-CURVE

---

Input: A large prime $p$

Output: $(a, b)$ such that $E_p(a, b)$ is an elliptic curve over $\mathbb{Z}_p$ and $P \in E_p(a, b)$

1. Repeat
   
   $((a, b), P) \leftarrow$ RANDOM-CURVE($p$)
   
   Calculate $N = \#E_p(a, b)$ by Schoof's algorithm
   
   Until $N$ is prime and $p^j \not\equiv 1 \pmod{N}$, $1 \leq j \leq \lg^2 p$

2. Return $((a, b), P)$

Figure 4.13: Koblitz's algorithm for generating curves of prime order, RANDOM-CURVE-PRIME
Algorithm RANDOM-CURVE-PRIME's advantages are that:

- It produces curves $E_p$, of prime order. Hence, every point (except $O$) is a generator, making the subgroup generated by the base point immune to the exponential algorithms when $p$ is large.

- The index-calculus method requires exponential time, since it can be shown that the index-calculus method in $\mathbb{F}_{p^k}$ becomes fully exponential if $k \geq \lg^2 p$, where $k$ is the extension degree required by the MOV-reduction [2]. Hence, the generate curve is immune against the reduction attack.

However, the algorithm uses Schoof's algorithm to compute the number of points on the curve. As we have already mentioned, Schoof's algorithm is $O(\log^3 p)$, and even after several improvements by Morain, Atkin, and others, it is still awkward to use when $p$ is large. Therefore, it seems prudent to avoid Schoof's algorithm. In particular, generating curves of known order (without using Schoof's algorithm) seems more attractive. In fact, algorithms have been designed specifically to build curves of known order, although there is a price to be paid. It turns out that the algorithms require advanced technical details that are beyond the scope of the thesis and most have exponential running time. Interested readers should see [59, 41, 13, 7, 57, 40].
Chapter 5

Conclusion

As already stated, elliptic curve public-key cryptosystems suffer from the same deficiency affecting all other public-key cryptosystems, namely, that of low bandwidth. As such, they are slower than private-key cryptosystems, and as a result, are currently used only in digital signatures and key exchange schemes. However, they have offer several advantages over their counterparts:

- They offer the highest-strength-per-bit of any known public-key cryptosystem.
- The ECDLP seems, at the present time, to be stronger than both the DLP and factoring. The result of this is that elliptic curve cryptosystems can offer the same level of cryptographic security as, for example, RSA, with smaller key size.
- More flexibility in choosing an elliptic curve than in choosing a finite field.
- Elliptic curves, especially those defined over \( \mathbb{F}_2^n \), lend themselves to efficient implementation both in software and hardware [77, 31, 2, 57].

The smaller key sizes in elliptic curve cryptosystems result in smaller system parameters, smaller public-key certificates, bandwidth savings, and most importantly, low cost implementations are feasible in restricted computing environments such as smart cards and wireless devices. For a performance comparison of public-key cryptosystems, see Wiener [88].

More recently, in 1988, Newbridge Microsystems Inc. in conjunction with Cryptech Systems Inc., Canada, manufactured a single chip device that implements various public-key cryptosystems.
based on arithmetic in the field $\mathbb{F}_{2^{103}}$. Also, a VLSI device, requiring 11,000 gates, was built for performing arithmetic operations in the field $\mathbb{F}_{2^{155}}$ [2]. Harper et al. [22], describes a software implementation of the ElGamal cryptosystem over the finite field $\mathbb{F}_{2^{104}}$. They reported encryption rates of 2Kbits/sec on a SUN-2 SPARC-station with public keys of 105 bits. Miyaji [57] presents methods for selecting elliptic curves over prime fields suitable for implementing Schnorr's [74] signature scheme on smart cards.

As a result of this success, standards for elliptic curve systems are currently being drafted by various accredited standards bodies around the world; some of this work is summarized below.

1. Elliptic curves over $\mathbb{Z}_p$ and $\mathbb{F}_{2^n}$ are included in the draft IEEE P1363 standard (Standard Specifications for Public-Key Cryptography), which includes encryption, signature, and key agreement schemes.

2. Two work items by the National Standards Institute (ANSI) ASC X9 (Financial Services) are on drafting elliptic curve systems: ANSI X9.62, The Elliptic Curve Digital Signature Algorithm (ECDSA); and ANSI X9.63, Elliptic Curve Key Agreement and Transport Protocols.

3. The ATM (Asynchronous Transfer Mode) Forum Technical Committee's Phase I ATM Security Specification draft document aims at providing security mechanisms for ATM networks. Security services provided include confidentiality, authentication, data integrity, and access control. Elliptic curves systems are included in the variety of systems to be supported.

Certainly the use of elliptic curve systems by providers of information security is going to increase as these drafts become officially adopted.

Advances in factoring and solving the ECDLP will always cast a dark shadow of these schemes. These two problems' computational complexity is as yet still unresolved. Advances in computational complexity would either result in the resolution of these problems or provide convincing evidence of the strength of either, perhaps establishing, in the process a new paradigm for judging cryptographic schemes. Recently, though, our ability to carry out large arithmetic computations has grown steadily, using parallel machines and the novel idea of distributed computing, and now permits us to factor numbers with around 100 decimal digits and compute discrete logarithms in $\mathbb{Z}_p$ with $p$ having around 97 digits. At the same time, this implies that our current schemes are only vul-
nerable to a dramatic breakthrough in factoring and solving the ECDLP. On the hand, Schroeppe1 et al. [77] observed that, “As computing power increases, and the search capabilities of opponents improve accordingly, it is cheaper to improve the security of elliptic curve methods than to improve the security of modp methods.”

We have seen that the selection of elliptic curves for use in these cryptosystems must be performed prudently, as some curves are vulnerable to attacks.

The exact nature of the relationship between the DLP and the ECDLP is still unresolved. So far, only the MOV-reduction has shed light in to this problem, though only for supersingular curves. It is imperative that more research be done to resolve the nature of this relationship. At the same time, the search for other groups in which the DLP is intractable should continue. New methods for speeding up arithmetic on elliptic curves should be investigated, particularly, that of computing inverses in finite fields. A careful study of elliptic curves being proposed for use in cryptosystems should be performed so as to avoid pitfalls in the future. Anomalous and supersingular curves come to mind.

It is also possible that new cryptosystems would be discovered that will supersede elliptic curve systems, or a new method of computation developed that would render factoring and the ECDLP obsolete. In fact, Boneh and Lipton [8] extended a result of Shor’s [76] to show that the ECDLP can be solved in quantum polynomial time on a quantum computer. Shor [76] had previously shown that factoring and the DLP are solvable in random quantum polynomial time on a quantum computer. A “quantum computer” is a computing device based on principles of quantum mechanics, for more information see Brassard [10].

But, at the present moment, elliptic curve cryptosystems are a good step towards achieving the ultimate goal of information technology: a paperless office.
Bibliography


BIBLIOGRAPHY


BIBLIOGRAPHY


BIBLIOGRAPHY


