University of Cincinnati

Date: 9/28/2017

I, Nithesh Venkata Ramana Surya Bommireddipalli, hereby submit this original work as part of the requirements for the degree of Master of Science in Computer Engineering.

It is entitled:
Tutorial on Elliptic Curve Arithmetic and Introduction to Elliptic Curve Cryptography (ECC)

Student’s name: Nithesh Venkata Ramana Surya Bommireddipalli

This work and its defense approved by:

Committee chair: Carla Purdy, Ph.D.
Committee member: Wen-Ben Jone, Ph.D.
Committee member: George Purdy, Ph.D.
Tutorial on Elliptic Curve Arithmetic and Introduction to Elliptic Curve Cryptography (ECC)

A Thesis submitted to the
Graduate School at the
University of Cincinnati
In Partial Fulfillment of the
Requirements for the Degree of
MASTERS OF SCIENCE
In the Department of Electrical Engineering and Computing Systems
Of the College of Engineering and Applied Sciences

By

Nithesh Venkata Ramana Surya Bommireddipalli

Bachelor of Technology (B. TECH.), 2014
GITAM University, Visakhapatnam, India

Thesis Advisor & Committee Chair: Dr. Carla Purdy
Abstract

This thesis focuses on elliptic curve arithmetic over the prime field GF (p) and elliptic curve cryptography (ECC). ECC over GF(p) has its own arithmetic which is done over elliptic curves of the form $y^2 \equiv x^3 + ax + b \pmod{p}$, where $p$ is prime. ECC is gaining importance in security because it uses smaller keys to provide the same security level as the popular RSA. It is the superior cryptographic scheme based on time efficiency and resource utilization. It is more suitable than RSA for DNSSEC and IoT systems and devices.

Unlike RSA, which is easily understood, ECC is complicated because of the arithmetic involved. It is not widely understood. We provide a tutorial on elliptic curve arithmetic and also explain the working of the ElGamal cryptosystem. We also describe general hardware-efficient methods to implement ECC such as Montgomery multiplication and projective coordinates. These methods are challenging to understand. Essentially, projective coordinates help reduce the number of inversions required in doing scalar multiplication. If Montgomery multiplication is used, a time-consuming operation like reduction modulo a prime $p$ can be simplified. In this work, we also present a user-friendly Java GUI application to provide education in elliptic curve arithmetic and its applications in cryptosystems. Lastly, we provide a module of questions and solutions to do the same and also enable senior students and graduate students to use ECC in their project work.
Acknowledgements

I would like to thank many people who have helped in my Masters journey. Without their help, I wouldn’t have reached this far. They have helped me stay motivated, focused, and dedicated to this work. They have made my experience at University of Cincinnati a great one.

Firstly, I am extremely thankful to my advisor Dr. Carla Purdy. She was very supportive and encouraging throughout. Her constant advice and technical expertise was invaluable. She helped me to find the area of focus for my research and clear all the obstacles I faced to achieve my goal. She pushed me to reach greater heights in this project and created a rich learning experience for me at the University of Cincinnati.

I would like to express immense gratitude to the thesis committee members Dr. Wen-Ben Jone and Dr. George Purdy for their active participation in my defense.

I would like to thank my family for allowing me to chase my dreams and keeping me motivated throughout the project. Lastly, I would like to thank my friends for making my experience at University of Cincinnati an enjoyable one.
# Table of Contents

1 Introduction .........................................................................................................................1

1.1 Motivation .........................................................................................................................1

1.2 Thesis Goals ......................................................................................................................3

1.3 Thesis Outline ..................................................................................................................3

2 Background ..........................................................................................................................5

2.1 Diffie-Hellman ..................................................................................................................5

2.2 Merkle’s Puzzles ..............................................................................................................7

2.3 RSA ...................................................................................................................................9

2.4 ElGamal Scheme ..............................................................................................................11

2.5 Applications of Public Key Cryptography ........................................................................12

2.5.1 SSH ............................................................................................................................13

2.5.2 SSL/TLS ......................................................................................................................13

2.5.3 S/MIME ......................................................................................................................14

2.5.4 PGP ................................................................................................................................14

2.6 Comparison of RSA and ECC .........................................................................................14

3 Elliptic Curve Arithmetic .....................................................................................................19

3.1 Group ...............................................................................................................................19

3.2 Field ..................................................................................................................................20

3.2.1 Modular Arithmetic .....................................................................................................21

3.2.1.1 Modular Addition, Modular Subtraction, and Modular Multiplication ..................22

3.2.1.2 Modular Inverse .......................................................................................................22

3.2.2 Field $\mathbb{Z}_p$ (=GF(p)) ............................................................................................23
3.3 Elliptic Curves over Real Numbers and GF(p) .......................................................... 24
3.3.1 Elliptic Curves over Real Numbers ........................................................................ 25
3.3.1.1 Point Addition .................................................................................................. 26
3.3.1.2 Adding a Point to Its Negative ......................................................................... 26
3.3.1.3 Adding a Point to the Point at Infinity .............................................................. 27
3.3.1.4 Point Doubling .................................................................................................. 27
3.3.1.5 Scalar Multiplication ....................................................................................... 28
3.3.2 Elliptic Curves over GF(p) .................................................................................... 28
3.3.2.1 Point Addition .................................................................................................. 29
3.3.2.2 Adding a Point to Its Negative ......................................................................... 30
3.3.2.3 Adding a Point to the Point at Infinity .............................................................. 30
3.3.2.4 Point Doubling .................................................................................................. 30
3.3.2.5 Scalar Multiplication ....................................................................................... 32
3.3.3 Example of Elliptic Curve over GF(p) ................................................................... 32
3.3.4 NIST-Recommended Elliptic Curves over Prime Fields .................................. 33
3.3.5 Elliptic Curve Digital Signature Algorithm (ECDSA) .................................... 34
3.4 ElGamal Cryptosystem ............................................................................................... 34

4 Efficient Methods of Implementing Elliptic Curve Arithmetic in Hardware ........... 37
4.1 Montgomery Multiplication Method ........................................................................ 37
4.2 Projective Coordinates ............................................................................................. 42
4.2.1 Elliptic Curve Point Addition and Doubling Using Projective Coordinates ....... 43
4.3 Combining Montgomery Multiplication and Projective Coordinates ..................... 47

5 GUI Application for Elliptic Curve Arithmetic and ElGamal Cryptosystem .......... 50
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 ElGamal Cryptosystem Mode</td>
<td>50</td>
</tr>
<tr>
<td>5.2 Arithmetic Mode</td>
<td>52</td>
</tr>
<tr>
<td>6 Conclusions and Future Work</td>
<td>57</td>
</tr>
<tr>
<td>References</td>
<td>58</td>
</tr>
<tr>
<td>Appendix A-User Manual for GUI Application</td>
<td>63</td>
</tr>
<tr>
<td>A.1 Instructions for Opening the Application</td>
<td>63</td>
</tr>
<tr>
<td>A.2 Instructions for Using Application</td>
<td>63</td>
</tr>
<tr>
<td>A.2.1 Instructions for ElGamal Cryptosystem Mode</td>
<td>63</td>
</tr>
<tr>
<td>A.2.2 Instructions for Arithmetic Mode</td>
<td>65</td>
</tr>
<tr>
<td>Appendix B-Module</td>
<td>69</td>
</tr>
<tr>
<td>B.1 Questions</td>
<td>69</td>
</tr>
<tr>
<td>B.2 Solutions</td>
<td>72</td>
</tr>
</tbody>
</table>
List of Figures

Figure 1: RSA vs ECC Comparison [25]..........................................................16

Figure 2: Comparison of Total Time for Encryption and Decryption of RSA and ECC in Smart Card [5], [26].................................................................17

Figure 3: Point Addition on the Curve $y^2 = x^3 - 7x$ [17]...........................................26

Figure 4: Adding a Point to Its Negative on the Curve $y^2 = x^3 - 6x + 6$ [17].........................27

Figure 5: Point Doubling on Curve $y^2 = x^3 - 3x + 5$ [17]...........................................28

Figure 6: Setup for ElGamal Cryptosystem Mode .........................................................51

Figure 7: Output of Application for ElGamal Cryptosystem Mode ...............................52

Figure 8: Setup for Point Addition ..............................................................................53

Figure 9: Output of Application for Point Addition ......................................................53

Figure 10: Setup for Point Doubling ............................................................................54

Figure 11: Output of Application for Point Doubling ....................................................55

Figure 12: Setup for Scalar Multiplication ....................................................................56

Figure 13: Output of Application for Scalar Multiplication .........................................56

Figure A.1: Main Menu Screen ..................................................................................63

Figure A.2: Instructions for ElGamal Cryptosystem Mode .............................................64

Figure A.3: ElGamal Cryptosystem Mode Screen .........................................................65

Figure A.4: Instructions for Arithmetic Mode ...............................................................66

Figure A.5: Arithmetic Mode Screen ............................................................................67
List of Tables

Table 1: Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis [4] ........15
Table 2: ECC Encryption and Decryption Time in Smart Card [5], [26] ..........................16
Table 3: RSA Encryption Time and Decryption Time in Smart Card [5], [26] ....................17
Table 4: Comparison of Power Required for Implementing ECDH and RSA Algorithms in
Synopsys [32] .............................................................................................................18
Table 5: Comparison of Area Required for Implementing ECDH and RSA Algorithms in
Synopsys [32] .............................................................................................................18
Table 6: Checking for Group Properties with Set of Integers, Z, along with + and * Operators..20
Table 7: Modular Addition Table for Z5 .................................................................23
Table 8: Modular Multiplication Table for Z5 ..........................................................24
Table 9: ECC Abelian Group Properties ...............................................................25
Table 10: Point Addition Table for Curve $y^2 = x^3 + 2x + 4 \pmod{5}$ .......................33
1. Introduction

1.1 Motivation

Elliptic curve cryptography is a form of public key cryptography that is gaining importance. It can be used to improve browser security and mobile security [6], [30], [33], [34]. Some of its popular applications include bitcoins, multipurpose smart cards, and IoT applications. ECC can be considered as a replacement to the popular RSA public key cryptography scheme. When ECC is compared to RSA, ECC provides the same level of security as RSA with smaller key sizes [1], [2], [3], [4], [25]. It is superior in terms of time efficiency and resource utilization [5], [24], [26]. It is more suitable than RSA for providing security for DNSSEC and IoT systems and devices [30], [31], [32].

ECC can be used in DNS Security Extensions (DNSSEC) to make DNS servers more secure. DNS or Domain Name System maps domain names to IP addresses. DNS Security Extensions were developed to protect the DNS servers and transactions from attacks such as Distributed Denial of Service (DDOS) attacks. With ECC, DNSSEC can also protect the DNS servers from attacks like amplification attacks and packet fragmentation [6], [30].

ECC has promise in two-factor authentication in mobile devices, based on comparison of two-factor authentication with ECC and without ECC. The authentication protocol with ECC is resistant to replay attacks, man-in-the-middle attacks, foreign agent impersonation attacks, and user impersonation attacks. It can protect against stolen smartcard attacks. It can defend against offline and online password-guessing attacks as well as session-key compromised attacks. This
protocol has been verified with the Automated Validation of Internet Security Protocols and Applications (AVISPA) tool [6], [33].

ECC can also be used to improve security of MANETs (Mobile Ad hoc Networks). MANET is a wireless network of mobile nodes enabling peerless communication between the nodes without reliance on a fixed infrastructure [7]. ECC and Enhanced Adaptive Acknowledgement have been used to improve security of MANETs [6], [34].

Security for IoT devices can be provided with ECC. A one-time password scheme with ECC has been developed for IoT devices. The scheme that was developed is resource-efficient compared to competing schemes [6], [31].

Security enhancements with ECC can be made in vehicular communication. Autonomous self-driving cars can communicate wirelessly, so it is important to prevent false messages being transmitted by a hacker, leading the vehicles astray. Secure vehicular communication was difficult previously because of larger key sizes that cryptographic schemes used. Since ECC has smaller key sizes, it can more easily be used to provide secure communication between autonomous self-driving vehicles. There is a scheme that comes with a low computation cost and provides mutual authentication and confidentiality. The scheme is secure against brute force and man-in-the-middle attacks. These are not successful given only polynomial time due to complexity of the discrete logarithm problem [6], [35].

Due to promising results in various security applications, it is very important to understand elliptic curve arithmetic and ECC-based cryptosystems. Use of elliptic curve arithmetic makes ECC unique among all the existing cryptographic schemes. The arithmetic uses modular arithmetic, including modular multiplication, addition, and inversion. Points are taken on an
elliptic curve and different operations such as point addition, point doubling, and scalar multiplication are performed on those points.

1.2 Thesis Goals

Analyzing an ECC-based cryptosystem requires an understanding of elliptic curve arithmetic. Understanding elliptic curve arithmetic requires knowledge of modular arithmetic. This includes knowledge of group and field properties and knowledge of how to perform modular addition, subtraction, multiplication, and inversion. In addition to this, knowledge of operations on elliptic curve points is required. This includes addition of points, doubling points, and scalar multiplication. Given a field GF(p) and a point P, scalar multiplication is equivalent to calculating aP where a is an integer greater than or equal to 2 and p is a prime number. If a is 2, that is a unique type of scalar multiplication called point doubling.

The goal of this thesis is to provide all the technical knowledge required to understand an ECC-based cryptosystem, particularly the ElGamal cryptosystem. A user application in Java has been developed to make grasping this technical knowledge much easier. In addition to this, a module has been developed containing questions and their solutions to assist with same and also enable senior students and graduate students to use ECC in their project work.

1.3 Thesis Outline

The thesis is organized into six chapters.

In chapter 2, different public key cryptographic schemes are explained including ElGamal scheme and RSA scheme. A comparison between RSA and ECC is made, which shows ECC is clearly the better scheme.
Chapter 3 explains elliptic curve arithmetic and the ElGamal cryptosystem thoroughly. It defines the entities group, Abelian group, field, and finite field and the properties that they satisfy. It explains how points on elliptic curves over real and prime fields along with the point at infinity form an Abelian group. Field arithmetic, including modular addition, modular subtraction, modular multiplication, and modular inversion, is explained in great detail. The chapter also covers ECC operations like point addition, point doubling, and scalar multiplication. An example of an elliptic curve over a prime field is given and results of various operations on points of the curve are presented.

Chapter 4 describes efficient methods such as Montgomery multiplication and projective coordinates used to implement ECC in hardware.

Chapter 5 presents our Java implementation of ECC. It can be used to understand elliptic curve arithmetic as well as the ElGamal cryptosystem. Its functionality and various applications are explained.

In chapter 6, we give our conclusions and describe possible future work.
2. Background

Cryptography is the method of protecting information or providing confidentiality to information by transforming information into ciphertext. Only those who possess a secret key can decipher the information into plaintext. Cryptography is divided into two types, symmetric key cryptography and asymmetric key cryptography. In symmetric key cryptography, one key is used for both encryption and decryption. In asymmetric key cryptography, two keys are used, one for encryption and one for decryption. Asymmetric key cryptography is also known as public key cryptography. Two public key cryptography schemes are RSA and ElGamal schemes [9], [10]. We explain the RSA and ElGamal schemes and describe some key applications of public key cryptography in this chapter.

RSA is the most popular public key cryptography scheme today, but we compare ECC and RSA schemes in this chapter to show that ECC can replace RSA [1], [2], [3], [4], [5], [24], [25], [26]. We show that ECC surely has an important role to play in the future of security and also show how important it is to understand elliptic curve arithmetic operations and an ECC-based cryptosystem.

2.1 Diffie-Hellman

The idea of public key cryptography was proposed by Whitfield Diffie and Martin Hellman in 1976 [27]. They defined the properties of a public key cryptosystem and a way of establishing a shared key over a public channel.

Diffie and Hellman define a public key cryptosystem as a pair of families \( \{ E_K \}_{K \in \mathbb{K}} \) and \( \{ D_K \}_{K \in \mathbb{K}} \) of algorithms representing invertible transformations,
on a finite message space \( \{M\} \), such that the following conditions are met.

(i) For every \( K \in \{K\} \), \( E_K \) is the inverse of \( D_K \).

(ii) For every \( K \in \{K\} \) and \( M \in \{M\} \), the algorithms \( E_K \) and \( D_K \) are easy to compute.

(iii) For almost every \( K \in \{K\} \), each easily computed algorithm equivalent to \( D_K \) is computationally infeasible to derive from \( E_K \).

(iv) For every \( K \in \{K\} \), it is feasible to compute inverse pairs \( E_K \) and \( D_K \) from \( K \).

\( K \) is key, \( M \) is message, and \( \{K\} \) is a finite set called key space [27].

Diffie-Hellman Key Exchange Protocol

1. Alice and Bob want to establish a secret shared key to send messages to each other. Alice and Bob agree on common parameters that don’t have to be kept hidden from everyone else. They are:

   (i) large prime \( q \) and

   (ii) primitive root \( r \) \((mod \ q)\).

where \( r \) is called a primitive root \( mod \ q \) if for every integer \( w \) coprime to \( q \), there is an integer \( k \) such that \( r^k \equiv w \ (mod \ q) \). After agreeing on these common parameters, Alice and Bob generate their own public keys. Alice selects a secret integer \( x < q \) and computes her public key \( A = r^x \ (mod \ q) \).
Similarly, Bob does the same. His public key is \( B = r^y \pmod{q} \) and \( y \) is his secret integer. Both publish their public keys so both know each other’s public key.

2. Both Alice and Bob now compute a common value, which they can use to encrypt messages. Bob computes \( A^y \pmod{q} \) using his private key \( y \). Alice computes \( B^x \pmod{q} \) using her private key \( x \). Both values computed are the same. This is the shared key or secret of Alice and Bob [8], [27].

### 2.2 Merkle’s Puzzles

Ralph Merkle provided a paradigm of establishing a shared key between two parties when the third party fully has full knowledge of the information being shared between the two parties in 1978 [29]. Until Merkle came up with this paradigm, a key was established between two parties when a third party had no knowledge of the information being sent. Keys were sent over a channel called key channel, which was a channel dedicated for keys. Messages were exchanged over a normal channel, which was a channel dedicated for exchanging normal information.

Merkle’s paradigm was based on one idea. If two parties X and Y wish to establish a key and there is a third party Z which wants to determine the key, the amount of work done by Z to figure out the key should be more than the work done by X and Y.

Merkle defined a “puzzle” in his paradigm. A puzzle is like a cryptogram, but a puzzle is meant to be solved. A puzzle is obtained by encrypting information with an encryption function that has restricted key space. With restricted key space, the puzzle becomes solvable. Merkle said a puzzle should only be solved through an exhaustive search of the key space. He also said
redundancy should be included in the puzzle so that the puzzle is not random. If the puzzle is random, cryptanalysis can be done, and the puzzle is solvable. Merkle suggested a known constant be attached to the information before encrypting and sending the information.

Puzzles are used to establish a key between two parties X and Y. The following was the method proposed by Merkle to establish a shared key.

Method:

X and Y agree upon a value N. X generates N puzzles and transmits these N puzzles to Y over the key channel. X chooses the size of the key space such that each puzzle requires O(N) efforts to break.

A puzzle consists of two pieces of information. One piece of information is a puzzle ID, which uniquely identifies each of the N puzzles. The IDs are assigned by X at random. The other piece of information in the puzzle is a puzzle key, i.e., one of the possible keys to be used in subsequent encrypted communications over the normal channel. Neither piece of information is readily available to anyone examining the puzzle.

When Y is presented with a menu of N puzzles, he selects a puzzle at random, and then spends the amount of effort required to solve the puzzle (O(N)). Y then transmits the ID back to X over the key channel, and uses the puzzle key found in the puzzle as the key for further encrypted communications over the normal channel.

X, Y, and Z all know the N puzzles. They also know the ID because Y transmitted the ID over the key channel. Y knows the corresponding puzzle key because Y solved the correct puzzle. X knows the corresponding puzzle key, because X knows which puzzle key is associated with the ID that Y sent. Z knows only the ID, but does not know the puzzle key. Z cannot know which
puzzle contains the puzzle key that Y selected, and which X and Y are using, even though he knows the ID. To determine which puzzle is the correct one, he must break puzzles at random until he encounters the correct one.

If Z desires to determine the key which X and Y are using, then, on an average, Z will have to solve N/2 puzzles before reaching the puzzle that Y solved. Each puzzle is constructed so that it requires O(N) effort to break, so Z must spend, on an average, O(N^2) effort to determine the key. Y, on the other hand, needs to only spend O(N) effort to break the one puzzle he selected, while X needs only spend O(N) effort to create the N puzzles. Thus, both X and Y will only put in O(N) effort. So, Z puts more effort than X and Y [29].

2.3 RSA

Ron Rivest, Adi Shamir, and Leonard Adelman came up with the first public key encryption and decryption scheme in 1978 [9]. It was a method making use of two keys, a public key and private key. The public key can be used to encrypt the message and the private key can be used to retrieve the original message from the encrypted message. The RSA scheme consists of three parts (i) key generation, (ii) encryption, and (iii) decryption [8].

Steps for Key Generation, Encryption, and Decryption

Key Generation

Each user generates their public and private keys by following these steps.

(i) Two large primes, p and q are selected. Typically, these are chosen at random.

(ii) System modulus (n) is computed. n = p*q.
(iii) Encryption key e is chosen where $1 < e < \phi(n)$.

$e$, $\phi(n)$ satisfy the relationship $\gcd(e, \phi(n)) = 1$ where $\phi(n)$ is Euler’s totient function and $\phi(n) = (p-1)(q-1)$.

(iv) An equation is then solved to obtain the decryption key d. The equation is

$d = e^{-1} \pmod{\phi(n)}$, $0 < d < n$.

Public key is $\{e, n\}$ and decryption or private key is $\{d, n\}$. Each user publishes their public key but keeps their private key hidden. $p$, $q$, and $\phi(n)$ must be kept secret because knowing them enables the computation of $d$.

Encryption

Suppose Alice wants to send a message $m$, written as an integer, to Bob using this cryptography scheme. The selected message $m$ has to be less than the system modulus $n$.

(i) Alice gets the public key of Bob, which is $\{e, n\}$.

(ii) She computes $c = m^e \pmod{n}$, where $0 < m < n$.

Decryption

To decipher the message $c$, this is what Bob does. Bob computes $c^d = (m^e)^d \equiv m \pmod{n}$ using his private key $\{d, n\}$ and recovers the original message. $c^d$ equates to the message $m$ based on the following.
Since \( e^d = 1 \pmod{\varphi(n)} \), \( e^d = 1 + k\varphi(n) \) for some integer \( k \).

\[
c^d \equiv m^{exd} \pmod{n} \\
\equiv m^{1+k\varphi(n)} \pmod{n} \\
\equiv m^{1\cdot(m^{\varphi(n)})^k} \pmod{n} \\
\equiv m^{1\cdot1^k} \pmod{n} \quad \text{(By Euler’s Theorem)} \\
\equiv m^1 \pmod{n} \\
\equiv m \pmod{n}
\]

Here, we are using Euler’s totient theorem, which states that if \( a \) and \( n \) are positive integers and if \( \text{gcd}(a,n) = 1 \), then \( a^{\varphi(n)} \equiv 1 \pmod{n} \) [15].

### 2.4 ElGamal Scheme

This scheme was developed by Taher ElGamal in 1985 [10]. It also consists of three components: (i) key generation, (ii) encryption, and (iii) decryption [8].

Steps for Key Generation, Encryption, and Decryption

**Key Generation**

Users select a large prime \( p \) and primitive root \( \alpha \pmod{p} \). Both of these are part of the user’s public key. Each user selects random \( a \in [0,p-1] \) and makes the computation \( \alpha^a \pmod{p} \). The user’s public key is \((p,\alpha,\alpha^a)\). Private key is \( a \). If a user wants to send a message to another user,
for example, if Bob wants to send a message to Alice, the message has to be in the form of a number modulo p.

Encryption

Bob generates the message $m$ modulo $p$ that he wants to send to Alice. He gets Alice’s public key $(p, \alpha, \alpha^a)$. He chooses $b \in [0, p-1]$ randomly. He computes $\alpha^b$ and $m(\alpha^a)^b \pmod{p}$. Then, he sends $c = (\alpha^b, m\alpha^{ab})$ to Alice.

Decryption

Alice receives $\alpha^b$ and $m\alpha^{ab}$. To recover $m$, she computes $(\alpha^b)^{p-a} \equiv (\alpha^b)^{p-1-a} \pmod{p}$ because $\alpha^{p-1} \pmod{p} \equiv 1$ from Fermat’s Little Theorem [15].

She then computes

$$(\alpha^b)^{p-1-a} \equiv m\alpha^{ab}$$

$$\equiv m\alpha^{p-1} \pmod{p}$$

$$\equiv m \pmod{p}$$ to recover the message $m$.

It can be observed that the sender of the message sends two parameters in case of ElGamal scheme and one parameter in case of RSA scheme. This doubles the transmission data compared to RSA.

2.5 Applications of Public Key Cryptography

Public key cryptography has a wide range of applications. It is used in protocols like SSH, PGP, S/MIME, and SSL/TLS.
2.5.1 SSH

SSH (Secure Shell) is a network protocol that provides administrators a secure way to access a remote computer. It allows secure data communication among two computers connecting over an insecure network like the internet. It is used to manage systems and applications remotely by network administrators. Administrators can log into another computer over a network, execute commands, and move files from one computer to another [11].

Public key cryptography is used to establish a secure connection between server (remote machine) and client (user or machine). The server sends its host key to the client. The client encrypts the session key with the server’s public key and sends the encrypted session key to the server along with the encryption algorithm used. The server decrypts the encrypted session key with its private key to obtain the session key. A confirmation message is encrypted using the session key and sent to the client. Further communication is done by using a symmetric key cryptographic algorithm and the session key.

2.5.2 SSL/TLS

These are cryptographic protocols used in applications such as web browsing, email, instant messaging, and voice-over-IP (VoIP). Websites use TLS to secure all communication between their servers and web browsers. The identities of communicating parties, client and server, are usually authenticated using public key cryptography and public key cryptography is also used to establish a session key between client and server [12].
2.5.3 S/MIME

S/MIME is a secure method of sending e-mails. It is based on asymmetric cryptography and protects emails from unwanted access. Emails are encrypted with the recipient’s public key. Emails can only be decrypted with the corresponding private key, which is in sole possession of recipient. Unless someone figures out the private key, only the intended recipient can access the sensitive data in mails [13].

2.5.4 PGP

PGP is an encryption program used to increase security of emails. It is also used for encrypting and decrypting texts, e-mails, files, directories, and whole disk partitions [14]. It employs public key cryptography to securely set up session keys. OpenPGP is the open-source version of PGP supported by the Internet Engineering Task Force (IETF).

2.6 Comparison of RSA and ECC

In this section, we show that ECC is a better scheme than RSA, especially in resource-constrained devices. We also show that ECC is more suitable than RSA for providing security for DNSSEC and IoT systems and devices [30], [31], [32].

According to [1], [2], [3], [4], and [25], ECC provides the same security level as RSA for smaller key size. Number of bits of security provided by the key size defines the security strength or security level provided by the scheme. More number of bits of security means higher security level for the scheme. It also means more the computational effort required to break the scheme. Security levels of RSA and ECC schemes for different key sizes as well as security levels of different symmetric key algorithms are provided in Table 1 [4].
<table>
<thead>
<tr>
<th>Security Strength in bits</th>
<th>Symmetric Key Algorithms</th>
<th>RSA (Minimum size of public key in bits)</th>
<th>ECC (Minimum size of public key in bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤80</td>
<td>2TDEA</td>
<td>1024</td>
<td>160</td>
</tr>
<tr>
<td>112</td>
<td>3TDEA</td>
<td>2048</td>
<td>224</td>
</tr>
<tr>
<td>128</td>
<td>AES-128</td>
<td>3072</td>
<td>256</td>
</tr>
<tr>
<td>192</td>
<td>AES-192</td>
<td>7680</td>
<td>384</td>
</tr>
<tr>
<td>256</td>
<td>AES-256</td>
<td>15360</td>
<td>512</td>
</tr>
</tbody>
</table>

Table 1: Comparable Key Sizes in Terms of Computational Effort for Cryptanalysis [4]

Computational effort in cryptography is usually expressed in MIPS years. One MIPS year is the amount of work performed in one year by a computer operating at a rate of one million operations per second. Figure 1 [25] compares what key lengths of RSA and ECC schemes will provide a level of security measured by time in MIPS years to break the security. It shows smaller keys in ECC require the same computational effort as larger keys in RSA to break the scheme. This is the same as saying that ECC provides the same security level as RSA for smaller key sizes.
ECC is more time-efficient. That was found out when implemented in smart cards. The total time required for encryption and decryption for RSA was greater than the same for ECC for all the comparable key sizes as shown in Figure 2 [5, 26], which plots data from Table 2 [5, 26] and Table 3 [5, 26]. The difference in total time required for encryption and decryption between RSA and ECC grew for larger keys.

![Figure 1: RSA vs ECC Comparison [25]](image)

Table 2: ECC Encryption and Decryption Time in Smart Card [5], [26]
Table 3: RSA Encryption Time and Decryption Time in Smart Card [5], [26]

<table>
<thead>
<tr>
<th>Length Key in RSA</th>
<th>RSA Decryption time (ms)</th>
<th>RSA Encryption time (ms)</th>
<th>RSA signature generation</th>
<th>RSA signature verification</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>8.60</td>
<td>0.46</td>
<td>75.19</td>
<td>4.67</td>
</tr>
<tr>
<td>1536</td>
<td>11.83</td>
<td>0.63</td>
<td>111.57</td>
<td>7.12</td>
</tr>
<tr>
<td>2048</td>
<td>12.46</td>
<td>0.85</td>
<td>152.45</td>
<td>9.63</td>
</tr>
<tr>
<td>3072</td>
<td>24.39</td>
<td>1.29</td>
<td>228.07</td>
<td>16.62</td>
</tr>
<tr>
<td>7680</td>
<td>64.41</td>
<td>3.39</td>
<td>544.92</td>
<td>43.54</td>
</tr>
<tr>
<td>15360</td>
<td>96.73</td>
<td>45.95</td>
<td>1137.2</td>
<td>122.15</td>
</tr>
</tbody>
</table>

Figure 2: Comparison of Total Time for Encryption and Decryption of RSA and ECC in Smart Card [5], [26]

ECC requires less memory and less resources compared to RSA. So, ECC is more suitable for resource-constrained devices. In [24], 1024-bit key RSA and 163-bit key ECC are compared for a Java card. The ECC algorithm needs fewer resources in hardware and uses less memory. It also needs less computational overhead, so it is more efficient and suitable for devices with smaller memory [24].

DNSSEC suffers from IP fragmentation and amplification attacks. The main cause for this is the selection of RSA as the default signature algorithm for DNSSEC. Signature schemes based on elliptic curve cryptography (ECC) address these issues [30].

ECC is more suitable than RSA in providing security for IoT systems and devices. To achieve end-to-end authentication in IoT devices, current authentication schemes and protocols
require a two-factor authentication mechanism. Two factor authentication mechanisms can be implemented using One-Time Password (OTP) schemes. OTP generation in Identity-Based Encryption ECC or IBE-ECC Lamport OTP scheme is quicker than the same in RSA OTP scheme when both provide the same security level. Also, IBE-ECC OTP scheme requires less resources than RSA OTP scheme [31]. So, the IBE-ECC OTP scheme is more suitable for IoT systems.

Elliptic Curve Diffie-Hellman (ECDH) algorithm was found superior to RSA algorithm in terms of power and area when IC layouts for both were realized in Synopsys using 90nm UMC Faraday library [32]. Results are shown in the following tables. This makes ECC suitable for IoT devices which have power and area constraints.

<table>
<thead>
<tr>
<th>CAD Tool</th>
<th>RSA</th>
<th>ECDH</th>
</tr>
</thead>
<tbody>
<tr>
<td>PrimeTime(Synopsys)</td>
<td>1.512 mW</td>
<td>0.570 mW</td>
</tr>
</tbody>
</table>

**Table 4: Comparison of Power Required for Implementing ECDH and RSA Algorithms in Synopsys [32]**

<table>
<thead>
<tr>
<th>Design Size</th>
<th>RSA</th>
<th>ECDH</th>
</tr>
</thead>
<tbody>
<tr>
<td>90nm</td>
<td>0.386 mm²</td>
<td>0.032 mm²</td>
</tr>
</tbody>
</table>

**Table 5: Comparison of Area Required for Implementing ECDH and RSA Algorithms in Synopsys [32]**
3. Elliptic Curve Arithmetic

In this chapter, properties of a group, Abelian group, field, and finite field are explained with examples. Following this, finite field arithmetic operations such as modular addition, modular subtraction, modular multiplication, and modular inversion are described along with examples. The later part of the chapter focuses on elliptic curves over real and prime fields and operations on points that lie on those curves. An example curve over a prime field is given and results of operations on points of the curve are presented. The focus is on creating a clear understanding of elliptic curve arithmetic. Then, the ElGamal cryptosystem is presented.

3.1 Group

A group is a non-empty set $G$, which can be finite or infinite, with binary operator $\cdot$ such that the following four properties are satisfied.

(i) Closure: if $a$ and $b$ both belong to $G$, then $a \cdot b$ also belongs to $G$.

(ii) Associativity: $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ for all $a$, $b$, $c$ in $G$.

(iii) Identity element: There is an element $i$ in $G$ such that $a \cdot i = i \cdot a = a$ for every element $a$ in $G$.

(iv) Inverse element: For every element $a$ in $G$, there is an element $a'$ such that $a \cdot a' = i$, where $i$ is the identity element.

If a group satisfies the commutativity property, that is, $a \cdot b = b \cdot a$ for every $a$, $b$ in $G$, the group is called an Abelian group [15].
Example:

Let us take the set of integers $\mathbb{Z}$ and the operators addition and multiplication.

<table>
<thead>
<tr>
<th></th>
<th>Addition</th>
<th>Multiplication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Closure</strong></td>
<td>$a+b$ is an integer</td>
<td>$a\cdot b$ is an integer</td>
</tr>
<tr>
<td><strong>Associativity</strong></td>
<td>$a+(b+c) = (a+b)+c$</td>
<td>$a\cdot (b\cdot c) = (a\cdot b)\cdot c$</td>
</tr>
<tr>
<td><strong>Existence of an identity element</strong></td>
<td>$a+0 = a$</td>
<td>$a\cdot 1 = a$</td>
</tr>
<tr>
<td><strong>Existence of inverse elements</strong></td>
<td>$a+(-a) = 0$</td>
<td>Only $1$ and $-1$ have inverses. $1\cdot 1 = 1$, $-1\cdot (-1) = 1$</td>
</tr>
<tr>
<td><strong>Commutativity</strong></td>
<td>$a+b = b+a$</td>
<td>$a\cdot b = b\cdot a$</td>
</tr>
</tbody>
</table>

Table 6: Checking for Group Properties with Set of Integers, $\mathbb{Z}$, along with $+$ and $\cdot$ Operators

It is clear from Table 6 that $(\mathbb{Z},+)$ is a group, but $(\mathbb{Z},\cdot)$ is not a group. According to the table, $(\mathbb{Z},+)$ is not only a group, but an Abelian group because in addition to the four group properties, it also satisfies the commutative property. Lastly, $(\mathbb{Z},\cdot)$ is not a group because most of the elements in the set don’t have multiplicative inverses.

3.2 Field

A field is a non-empty set $F$ with two binary operators, which are usually denoted by $+$ and $\cdot$, that satisfy the following:

(i) $(F,+)$ is an Abelian group with (additive) identity denoted by $0$. 
(ii) \((F\setminus\{0\},\ast)\) is an Abelian group with (multiplicative) identity element 1.

(iii) The distributive law holds: \((a+b)\ast c = a\ast c + b\ast c\) for all \(a, b, c \in F\). [8]

If the set \(F\) is finite, then the field is called a finite field. A finite field is also called a Galois field. An example of a finite field is \(\mathbb{Z}_p\), the integers modulo the prime \(p\). Elements are the set of integers \(\{0, 1, 2, \ldots, p-1\}\). This set along with modulo \(p\) forms a finite field. To understand how, it is important to first understand addition and multiplication modular arithmetic.

### 3.2.1 Modular Arithmetic

The modulo operation, denoted by \(\text{mod}\), calculates the remainder of the division of a positive integer by another positive integer. \(a \ (\text{mod} \ n) = r\), where \(a\) is the dividend, \(n\) is the divisor, and \(r\) is the remainder. Examples: \(5 \ (\text{mod} \ 2) = 1\), \(7 \ (\text{mod} \ 4) = 3\). Moduli are restricted to positive integers in this paper, although negative integers can also be used.

The congruence symbol \(\equiv\) is used to indicate modular equality. The statement \(a \equiv c \ (\text{mod} \ n)\) means \(a\) and \(c\) differ by a multiple of \(n\). We can also say \(a\) and \(c\) are congruent mod \(n\). The following examples give an idea of how to use \(\equiv\).

Examples:

If \(n = 26\) and set of numbers \(\{0, 1, 2, 3, \ldots, 25\}\) is being operated on, then

\[
26 + 3 \equiv 3 \ (\text{mod} \ 26)
\]

\[
15 + 16 \equiv 5 \ (\text{mod} \ 26)
\]
3.2.1.1 Modular Addition, Modular Subtraction, and Modular Multiplication

Modular addition, modular subtraction, and modular multiplication are three important modular arithmetic operations.

Properties of Modular Arithmetic

Assume a, b, c, d, and n are integers with n > 0. If \( a \equiv c \pmod{n} \) and \( b \equiv d \pmod{n} \), then

(i) \( a+b \equiv c+d \pmod{n} \)

(ii) \( a-b \equiv c-d \pmod{n} \)

(iii) \( ab \equiv cd \pmod{n} \) [8]

3.2.1.2 Modular Inverse

Modular inverse is another important modular arithmetic operation. If a is an integer, its modular inverse or inverse \( a^{-1} \) is an integer that satisfies the relation \( a \cdot a^{-1} \equiv 1 \pmod{p} \). Finding the inverse of an integer modulo a prime p is simplified by Fermat’s Little Theorem.

Fermat’s Little Theorem:

Fermat’s Little Theorem states that if p is prime and the greatest common divisor of p and an integer a is 1, then \( a^{p-1} \equiv 1 \pmod{p} \) [15].

By rewriting the relation given by Fermat’s Theorem, a method is obtained to find the inverse of an integer modulo prime p. \( a^{p-1} \equiv 1 \pmod{p} \Rightarrow a \cdot a^{p-2} \pmod{p} \equiv 1 \Rightarrow a^{p-2} \pmod{p} \) is inverse of a, i.e. \( a^{-1} = a^{p-2} \pmod{p} \).
3.2.2 Field $\mathbb{Z}_p (= \text{GF}(p))$

Now, it can be seen that if $p$ is prime, then $\mathbb{Z}_p$ forms a finite field with operation modulo $p$ since every non-zero element has a multiplicative inverse.

Example: $\mathbb{Z}_5$

$(\mathbb{Z}_5,+)$ forms an Abelian group. Each integer in $\mathbb{Z}_5$ has an additive inverse. Table 7 shows the results obtained after addition of two of the integers in $\mathbb{Z}_5$. The results are also in the set $\mathbb{Z}_5$. This indicates closure. Entries represent $x+y \pmod{5}$. The additive identity is 0.

Associativity and commutativity properties can be simply verified with the help of Table 7.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 7: Modular Addition Table for $\mathbb{Z}_5$
It can be shown \((\mathbb{Z}_5 - \{0\}, \cdot)\) is an Abelian group. Each integer in \(\mathbb{Z}_5 - \{0\}\) has a multiplicative inverse. Table 8 shows the results obtained after multiplying two integers in \(\mathbb{Z}_5 - \{0\}\). All the results lie in the set \(\mathbb{Z}_5 - \{0\}\). This indicates closure. Entries represent \(x \cdot y \pmod{5}\). The multiplicative identity is 1.

Associativity and commutativity properties can be simply verified with the help of Table 8.

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 8: Modular Multiplication Table for \(\mathbb{Z}_5\)

### 3.3 Elliptic Curves over Real Numbers and GF(p)

In this section, elliptic curves over real and prime fields are introduced. All the properties that points on these curves have are explained. Elliptic curve over a prime field is the main focus. In addition, an ElGamal cryptosystem is described.
3.3.1 Elliptic Curves over Real Numbers

ECC was discovered by Victor Miller and Neil Koblitz in 1985 [16]. An elliptic curve over real numbers is a set of points \((x,y)\) on the curve defined by the equation \(y^2 = x^3 + ax + b\), where \(x\), \(y\), \(a\), and \(b\) are all real numbers. The points \((x,y)\) along with point at infinity \(O\) form an Abelian group with point addition operator \(+_e\) if \(4a^3 + 27b^2 \neq 0\). This is shown in Table 9.

For the table, four points on the curve \(A\), \(B\), \(C\), and \(A'\) are taken. \(A\) is \((x,y)\) and \(A'\) is \((x,-y)\). This means \(A' = -A\) on the elliptic curve.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closure</td>
<td>(A +_e B) is another point on the curve</td>
</tr>
<tr>
<td>Associativity</td>
<td>(A +_e (B +_e C) = (A +_e B) +_e C)</td>
</tr>
<tr>
<td>Existence of an identity element</td>
<td>(A +_e O = A) so (O) is the identity element</td>
</tr>
<tr>
<td>Existence of inverse elements</td>
<td>(A +_e A' = O)</td>
</tr>
<tr>
<td>Commutativity</td>
<td>(A +_e B = B +_e A)</td>
</tr>
</tbody>
</table>

Table 9: ECC Abelian Group Properties

Operations commonly performed on elliptic curve points are point addition, point doubling, and scalar multiplication. They are thoroughly explained next. These operations can be done geometrically.
3.3.1.1 Point Addition

Suppose that P and Q are two distinct points on a real elliptic curve, P is not −Q, and P + e Q = R. To add the points P and Q, a line is drawn through the two points. This line will intersect the elliptic curve in exactly one more point, call it -R. The point -R is reflected with respect to the x-axis to the point R [17]. This is illustrated in Figure 3 [17] by taking points P(-2.35,-1.86) and Q(-0.1,0.836) on the curve y^2 = x^3 - 7x and adding them. P + e Q = R = (3.89,-5.62)

![Figure 3: Point Addition on the Curve y^2 = x^3-7x][17]

3.3.1.2 Adding a Point to Its Negative

Assume there are two points P(x_p,y_p) and another point Q(x_q,y_q) such that y_q = -y_p. This means Q = -P or point Q is the negative of point P. Line through points P and -P is vertical and doesn’t intersect the elliptic curve at a third point. This means addition of a point and its negative cannot be done using the previous method. Point at infinity O is introduced for this reason in
elliptic curve arithmetic. Adding a point and its negative gives the point at infinity. \( P + (-P) = O \).

This is illustrated in the Figure 4 [17] by taking the curve \( y^2 = x^3-6x+6 \).

![Figure 4: Adding a Point to Its Negative on the Curve \( y^2 = x^3-6x+6 \) [17]](image)

**3.3.1.3 Adding a Point to the Point at Infinity**

Adding a point \( P(x_p,y_p) \) to \( O \) will result in point \( P \), i.e. \( P + O = P \).

**3.3.1.4 Point Doubling**

To add a point \( P \) to itself, a tangent line to the curve is drawn at the point \( P \). If \( y \)-coordinate of point \( P \) is not 0, then the tangent line intersects the elliptic curve at exactly one other point, \(-R\). \(-R\) is reflected with respect to the x-axis to get \( R \), which is \( 2P \) [17]. This is illustrated in Figure 5 [17] by taking the point \( P(2, 2.65) \) and doubling it. \( 2P = R = (-1.11, 2.64) \).
If y-coordinate of point P is 0, then doubling point P gives point at infinity, i.e. $2P = O$.

Tangent drawn at such a point will not intersect the elliptic curve at any other point.

### 3.3.1.5 Scalar Multiplication

Calculating $aP$ where $a$ is an integer greater than or equal to 2 is called scalar multiplication. Scalar multiplication is calculated using a combination of point doubling and point addition. For example, $3P = 2P +_e P$ and $5P = 2^2(2P) +_e P$.

### 3.3.2 Elliptic Curves over GF(p)

An elliptic curve over a prime field is a set of points $(x,y)$ on the curve defined by the equation $y^2 = x^3 + ax + b \pmod{p}$, where $x$, $y$, $a$, and $b$ are elements of GF(p) for some prime $p \neq 3$. 
The points \((x,y)\) along with point at infinity \(O\) form an Abelian group with point addition operator \(+_e\) if \(4a^3+27b^2 \neq 0\).

Just like elliptic curves over real numbers, three operations can be done with the points on elliptic curves over \(GF(p)\). They are point addition, point doubling, and scalar multiplication. The difference is that these operations can’t be done geometrically. Formulae have to be utilized.

### 3.3.2.1 Point Addition

If there are two distinct points \(P(x_p,y_p)\) and \(Q(x_q,y_q)\) on the curve such that \(P\) is not \(-Q\), then

\[
R = (x_R,y_R), \quad \text{where } s = \frac{(y_p-y_q)}{(x_p-x_q)} \pmod{p}, \quad x_R = s^2 - x_p x_q \pmod{p}, \quad \text{and } y_R = -y_p + s(x_p-x_R) \pmod{p}.
\]

\(s = (y_p-y_q)/(x_p-x_q)\) is equivalent to \(s = (y_p-y_q)* (x_p-x_q)^{-1}\).

Example:

Elliptic curve is represented by the equation \(y^2 \equiv x^3+2x+4 \pmod{5}\). Consider the points \(P(2,4)\) and \(Q(0,3)\) on the curve. \((2,4) +_e (0,3)\) can be calculated as follows.

\[
x_p = 2, \quad y_p = 4, \quad x_q = 0, \quad \text{and } y_q = 3
\]

\[
y_p - y_q = 4 - 3 = 1
\]

\[
x_p - x_q = 2 - 0 = 2
\]

\[
(x_p-x_q)^{-1} \equiv 3 \pmod{5}
\]

\[
s = \frac{(y_p-y_q)}{(x_p-x_q)} = \frac{(4-3)}{2} = \frac{1}{2} \equiv \frac{3}{2} = \frac{3}{2} \cdot \frac{2}{2} = \frac{3}{1} \equiv 3 \pmod{5}
\]

\[
x_R = s^2 - x_p x_q \pmod{p}
\]

\[
s^2 = 3^2 = 9 \equiv 4 \pmod{5}
\]
\[ \begin{align*}
  s^2 - x_p & = 4 - 2 = 2 \\
  s^2 - x_p - x_q & = 2 - 0 = 2 \\
  \text{So, } x_R & = 2.
\end{align*} \]

\[ y_R = -y_p + s(x_p - x_R) \pmod{p} \]

\[-y_p = -4 \equiv 1 \pmod{5} \]

\[ x_p - x_R = 2 - 2 = 0 \]

\[ s(x_p - x_R) = 3*0 = 0 \]

\[ y_R = -y_p + s(x_p - x_R) = 1 \]

So, adding P and Q, point (2,1) on the curve is obtained.

### 3.3.2.2 Adding a Point to Its Negative

Adding a point and its negative gives the point at infinity. \( P + (-P) = O \). Addition formulae don’t apply to this case of addition.

### 3.3.2.3 Adding a Point to the Point at Infinity

Adding a point \( P(x_p, y_p) \) to \( O \) will result in point \( P \), i.e. \( P + O = P \).

### 3.3.2.4 Point Doubling

If there is a point \( P = (x_p, y_p) \) with \( y_p \neq 0 \) of an elliptic curve modulo the prime \( p \), then point \( R \) on the elliptic curve, i.e. \( R = 2P \) has the following coordinates \( x_R = s^2 - 2x_p \pmod{p} \) and \( y_R = -y_p \).
+s(x_p- x_R) (mod p) where s = (3x_p^2+a)/(2y_p) (mod p). s = (3x_p^2+a)/(2y_p) (mod p) is equivalent to s = (3x_p^2+a)*(2y_p)^{-1} (mod p) [8].

Example:

Elliptic curve is represented by the equation $y^2 \equiv x^3 + 2x + 4 \pmod{5}$. Consider the point (4,4) on the curve. $2 \cdot (4,4)$ can be calculated as follows.

First, s has to be calculated.

$s = (3x_p^2+a)*(2y_p)^{-1} \pmod{p}$

$3x_p^2 = 3 \cdot 16 = 48 \equiv 3 \pmod{5}$

$a = 2$

$3x_p^2+a = 3+2 = 0 \pmod{5}$

$2 \cdot y_p = 8 \equiv 3 \pmod{5}$

$(2y_p)^{-1} = 3^{-1} \pmod{5} = 2$

$s = 0 \cdot 2 = 0$

Next, $x_R$ and $y_R$ have to be calculated.

$x_R = s^2 - 2x_p \pmod{p}$

$s^2 = 0^2 = 0$

$2 \cdot x_p = 8 \equiv 3 \pmod{5}$

$s^2 - 2 \cdot x_p = 0 - 3 = -3 \equiv 2 \pmod{5} \Rightarrow x_R = 2$

$y_R = -y_p + s(x_p - x_R) \pmod{p}$
\[-y_p = -4 \equiv 1 \pmod{5}\]
\[s(x_p - x_R) = 0*(4-2) = 0\]

So, \(y_R = 1\). The resultant point obtained after doubling the point \(P\) is \((2,1)\), which is also on the curve.

If \(y_p = 0\), then \(2P = O\).

### 3.3.2.5 Scalar Multiplication

Calculating \(aP\) where \(a\) is an integer greater than or equal to 2 is called scalar multiplication.

Scalar multiplication is calculated using a combination of point doubling and point addition. For example, \(3P = 2P + _P\) and \(5P = 2*(2P) + _P\).

### 3.3.3 Example of Elliptic Curve over \(GF(p)\)

The operations covered in previous section are explained here. We take an elliptic curve over \(GF(5)\) to explain. The equation of the elliptic curve is \(y^2 \equiv x^3 + ax + b \pmod{p}\). Here, \(p = 5\). \(a\) and \(b\) are chosen such that \(4a^3 + 27b^2 \neq 0\). Let \(a = 2\) and \(b = 4\). Then,

\[4a^3 + 27b^2 \equiv 4*8 + 27*16 \equiv 4*3 + 2*1 \equiv 14 \equiv 4 \pmod{5}\]

\[\Rightarrow 4a^3 + 27b^2 \neq 0.\]

So, the curve is \(y^2 \equiv x^3 + 2x + 4 \pmod{5}\). Points on the curve are: \((0,2)\), \((0,3)\), \((2,1)\), \((2,4)\), \((4,1)\), and \((4,4)\).

Table 10 is a point addition table for these points. \((0,2) + _e (2,4) = (4,4)\), \((2,1) + _e (4,4) = (0,2)\), etc. Results that are obtained by doubling a point on the given curve are automatically
Scalar multiplication can be performing by repeated point additions and point doubling, making use of Table 10.

<table>
<thead>
<tr>
<th></th>
<th>(0,2)</th>
<th>(0,3)</th>
<th>(2,1)</th>
<th>(2,4)</th>
<th>(4,1)</th>
<th>(4,4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,2)</td>
<td>(4,1)</td>
<td>O</td>
<td>(2,4)</td>
<td>(4,4)</td>
<td>(2,1)</td>
<td>(0,3)</td>
</tr>
<tr>
<td>(0,3)</td>
<td>O</td>
<td>(4,4)</td>
<td>(4,1)</td>
<td>(2,1)</td>
<td>(0,2)</td>
<td>(2,4)</td>
</tr>
<tr>
<td>(2,1)</td>
<td>(2,4)</td>
<td>(4,1)</td>
<td>(0,3)</td>
<td>O</td>
<td>(4,4)</td>
<td>(0,2)</td>
</tr>
<tr>
<td>(2,4)</td>
<td>(4,4)</td>
<td>(2,1)</td>
<td>O</td>
<td>(0,2)</td>
<td>(0,3)</td>
<td>(4,1)</td>
</tr>
<tr>
<td>(4,1)</td>
<td>(2,1)</td>
<td>(0,2)</td>
<td>(4,4)</td>
<td>(0,3)</td>
<td>(2,4)</td>
<td>O</td>
</tr>
<tr>
<td>(4,4)</td>
<td>(0,3)</td>
<td>(2,4)</td>
<td>(0,2)</td>
<td>(4,1)</td>
<td>0</td>
<td>(2,1)</td>
</tr>
</tbody>
</table>

Table 10: Point Addition Table for Curve $y^2 = x^3 + 2x + 4 \pmod{5}$

### 3.3.4 NIST-Recommended Elliptic Curves over Prime Fields

There are 5 NIST-recommended prime fields, which are $p_{192} = 2^{192} - 2^{64} - 1$, $p_{224} = 2^{224} - 2^{96} + 1$, $p_{256} = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$, $p_{384} = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$, and $p_{521} = 2^{521} - 1$. For each of those prime fields, one randomly selected elliptic curve of the form $y^2 = x^3 - 3x + b \pmod{p}$ was recommended, denoted by P-192, P-224, P-256, P-384, and P-521 [18], [28]. NIST specifies points chosen on the curves have to be of order $n$, which is a large prime [28]. A point $P$ on an elliptic curve of the form $y^2 = x^3 + ax + b \pmod{p}$ is of order $n$ if $n$ is the small positive integer such that $nP = O$. Elliptic curves of the form $y^2 = x^3 - 3x + b \pmod{p}$ have a cofactor of 1 [28]. Cofactor of an elliptic curve of the form $y^2 = x^3 + ax + b \pmod{p}$ is the ratio of actual number of points on the curve divided by the order $n$. 
3.3.5 Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA is a Federal Information Processing Standard (FIPS) for digital signature generation and verification using elliptic curves. It is an analog of the Digital Signature Algorithm (DSA) [28].

3.4 ElGamal Cryptosystem

Up to this point, elliptic curve arithmetic was explained. Elliptic curve arithmetic can actually be used to understand a real-world elliptic curve cryptosystem like the ElGamal cryptosystem. In this section, we explain the ElGamal cryptosystem [8].

The system consists of three important processes (i) key generation, (ii) encryption, and (iii) decryption. To use the system, each user has to generate their public key and private key. So, each user selects an elliptic curve modulo some large prime $p$ and a point on the curve, say $S$. Each user selects an integer $b$ in the interval greater than or equal to 2 and computes $bS = T$. The public key of the user is $(p,S,T)$. The private key is $b$.

When a user (sender) wants to send a message to another user (receiver), both users decide on a curve and then encryption and decryption can be performed. Encryption is performed by the sender and decryption is performed by the receiver.

Encryption and Decryption Scheme for the ElGamal Cryptosystem

Assume the sender is Alice and the receiver is Bob. Assume Bob’s public and private keys are (i) $(p,S,T)$ and (ii) $b$ respectively. Assume Alice’s private key is $a$. 
Encryption

Alice chooses a message, which is point A on the curve chosen by both her and Bob. She computes aS and A +c aT obtaining Bob’s public key. Then, Alice sends (aS, A +c aT) to Bob. This is the encrypted message.

Decryption

Bob computes baS. Then, he computes (A +c aT)-baS.

\[(A +c aT)-baS = (A +c aT) - abS \Rightarrow (b(aS) = baS = a(bS) = abS)\]

\[= A +c abS +c (-abS)\]

\[= A +c O \Rightarrow (P +c (-P) = O \text{ if } P \text{ is a point on the elliptic curve over finite field } Z_p)\]

\[= A \Rightarrow (P +c O = P)\]

So Bob recovers A.

Example:

Assume both Bob and Alice agree on a curve \(y^2 \equiv x^3 + 2x + 4 \pmod{5}\). Bob chooses a point \(S = (2,1)\) on the curve to be a part of his public key. Bob chooses his private key, \(b\), to be 3. So, his public key is \((5,(2,1),3*(2,1))\).

\[3*(2,1) = 2*(2,1) +c (2,1) = (0,3) +c (2,1) = (4,1)\]

So, his public key is \((5,(2,1),(4,1))\). Alice selects the point (0,2) on the curve as her message and encrypts it using her private key \(a = 2\). She sends \((2*(2,1),(0,2) +c 2*(4,1)) \Rightarrow ((0,3),(0,2) +c (2,4)) \Rightarrow ((0,3),(4,4))\) to Bob.
Bob decrypts by calculating $3^* (0,3) = 2^* (0,3) +_e (0,3) = (4,4) +_e (0,3) = (2,4)$. He then subtracts this from (4,4).

$$(4,4) +_e (2,4) = (4,4) +_e (2,1) = (0,2)$$

So, Bob gets the message sent by Alice, which is (0,2).

In this system, if an attacker were to crack the system, the discrete logarithm problem of finding an integer $c$ such that $Q = cP$ has to be solved. If $c$ is a large number, that is highly difficult. An example is finding what $a$ is if the point $aS$ is given. Say Alice and Bob go back and forth sending messages. Then, if someone other than Alice and Bob figures out $a$ and $b$ from the messages sent to each other, future messages being sent between Alice and Bob can be decrypted.
4. Efficient Methods of Implementing Elliptic Curve Arithmetic in Hardware

In this chapter, we focus on how ECC can be efficiently implemented in hardware. The efficiency of ECC-implementing hardware can be improved using three methods. One method involves using Montgomery multiplication. The second method involves using projective coordinates instead of affine coordinates. The third method is using both Montgomery multiplication and projective coordinates together.

In the Montgomery multiplication method, the modulo operation is performed fewer times. The modulo operation involves repeated division by N, which is very costly in hardware. The modulo operation is replaced by division with power of 2, which comes at very little cost in hardware because it can be implemented by a right shift [19]. Coordinates of points are converted into the Montgomery domain before performing point addition, point doubling, and scalar multiplication to reduce the number of times the modulo operation is performed.

In the projective coordinates method, the inversion is performed a fewer number of times [20], [21]. Inversion is also time-consuming [21]. The affine coordinates of points are converted to projective coordinates before performing scalar multiplication.

In the third method, the previous two methods are combined so that inversion and modulo operations are both performed a fewer number of times.

4.1 Montgomery Multiplication Method

When elliptic curve arithmetic is implemented in hardware, usually numbers are converted into Montgomery domain, basically another type of number system, so that the time-consuming operation of reduction modulo m, where m is prime, is avoided [19]. The following explains how
the conversion of numbers into the Montgomery domain is done and how Montgomery multiplication is done to avoid reduction modulo m in elliptic curve arithmetic.

To convert numbers into the Montgomery domain, the m-residue of the numbers is calculated. The m-residue of an integer \(a < m\) is defined as \(\bar{a} = ar \pmod{m}\). Modulus \(m\) is an n-bit integer such that \(2^{n-1} < m < 2^n\) and \(r = 2^n\). The set \(\{ar \pmod{m} | 0 < a < m-1\}\) contains all the numbers between 0 and m-1. This means there is a one-to-one relation between the numbers 0 to m-1 and the set.

Given two m-residues \(\bar{a}\) and \(\bar{b}\), Montgomery multiplication is the method of obtaining \(\bar{c} = \bar{a} \bar{b} r^{-1} \pmod{m}\) or \(\bar{c} = cr \pmod{m}\), which is also known as the Montgomery product. Both expressions for \(\bar{c}\) are the same as

\[
\bar{c} = \bar{a} \bar{b} r^{-1} \pmod{m},
\]

\[
\bar{c} = ar*br*r^{-1} \pmod{m},
\]

\[
\bar{c} = abr \pmod{m},
\]

\[
\bar{c} = cr \pmod{m}.
\]

Here, \(r^{-1}\) is the inverse of \(r \pmod{m}\). It satisfies the property \(rr^{-1} \pmod{m} = 1\). Montgomery multiplication can also be defined as the method of obtaining the m-residue of the product \(c = ab \pmod{m}\). For performing Montgomery multiplication, \(r\) and \(m\) should be relatively prime. This means \(\gcd(r,m)\) should be 1. So, \(m\) should be odd. \(\bar{c}\) can be efficiently obtained in hardware using the following method.
Method

Let \( x = \bar{a} \bar{b} \)
\[
u = (x + (x \cdot m' \mod r))m/r
\]
If \( u > m \), then \( u = u - m \). Otherwise, \( u \) does not change.
\[
\bar{c} = u
\]
\( m' \) can be calculated from the relation \( rr^{-1}mm' = 1 \) [22].

The method is explained further with the following example.

Example:

Let \( a = 7 \), \( b = 5 \), \( m = 11 \), and \( r = 16 \). \( \bar{c} \) is to be computed using the mentioned method.

\[
\bar{a} = 7 \cdot 16 \mod 11 = 112 \mod 11 = 2
\]
\[
\bar{b} = 5 \cdot 16 \mod 11 = 80 \mod 11 = 3
\]
So, \( x = 2 \cdot 3 = 6 \)
\[
u = (6 + (6 \cdot m' \mod 16)) \cdot 11)/16
\]

\( m' \) can be calculated using Extended Euclidean Algorithm. The algorithm is a method of finding the solution \( (x,y) \) to \( ax + by = \gcd(a,b) \) and also helps to find \( \gcd(a,b) \). Here, \( a, b, x, \) and \( y \) are all integers.

Example:

Let \( a = m = 11 \) and \( b = r = 16 \).

\( 16 \) is expressed as \( 16 = 11 \cdot 1 + 5 \cdot (i) \)
11 is expressed as $11 = 5*2 + 1$ -(ii)

$5 = 1*5 + 0$ -(iii)

(i), (ii), and (iii) are all of the form $j = k+o$. The last non-zero value of $o$ is 1. This means $\gcd(11, 16)$ is 1. So, with Extended Euclidean Algorithm, the equation $11x + 16y = 1$ is solved.

Then, (ii) and (i) are rewritten as follows.

$$1 = 11 - 5*2$$ -(iv)

$$5 = 16 - 11*1$$ -(v)

Then, (v) is substituted in (iv).

$$1 = 11 - (16 - 11*1)*2$$

$$=> 1 = 11 + 2*11 - 2*16$$

$$=> 1 = 3*11 - 2*16$$ -(vi)

So, integers $x, y$ are $x = 3$ and $y = -2$.

Using this last relation (vi), $r^{-1}$ is calculated. The portion not divisible by $m = 11$ is seen in (vi), which is $-2*16$. Then, coefficient of $r = 16$ is taken, which is $-2$. $r^{-1} = -2 \equiv 9 \pmod{11}$. That can be verified as $r*r^{-1} = 16*9 = 144$. $144 \pmod{11} = 1$. Now, $m'$ can be solved from the relation $rr^{-1} - mm' = 1$. Every value is known except for $m'$.

$$rr^{-1} - mm' = 1$$

$$16*9 - 11*m' = 1$$

$$=> 11*m' = 16*9 - 1$$

$$=> m' = 143/11 = 13$$
Since $m'$ has now been computed, $u$ can be calculated as follows.

\[
u = (6 + (6 \cdot m' \mod 16) \cdot 11) / 16
\]
\[
u = (6 + (6 \cdot 13 \mod 16) \cdot 11) / 16
\]
\[
u = (6 + (78 \mod 16) \cdot 11) / 16
\]
\[
u = (6 + 14 \cdot 11) / 16
\]
\[
u = 160 / 16 = 10
\]

Since $u$ is not greater than $m$, $u-m$ doesn’t have to be computed. So $\bar{c} = 10$.

78 (mod 16) can be calculated as 78 (mod $2^4$), which can be found in hardware more quickly than for example, 78 (mod 11). 11 is a prime number, whereas 16 is a power of 2. a (mod $2^n$) where a is an integer can be directly found by grouping the rightmost n bits of a in binary form. This is very simple in hardware. 78 = 0100 1110 .This means the bits 1110 are grouped and 1110 = 14. In other words, 78 (mod 16) = 14.

Division by $2^n$ can be calculated in similar manner. a/$2^n$ is same as shifting the integer right by n bits. This calculation can also be done easily in hardware.

It can be verified that by calculating $c = ab \mod m$ and $\bar{c}$ from c that $\bar{c} = 10$. This is a direct method of calculating $\bar{c}$.

\[
c = ab \mod m
\]
\[
c = 35 \mod 11
\]
\[
c = 2
\]
\[ \bar{c} = cr \pmod{m} \]
\[ \bar{c} = 2*16 \pmod{11} \]
\[ \bar{c} = 32 \pmod{11} \]
\[ \bar{c} = 10 \]

The efficient method works just like the direct method. This example shows that multiplying two numbers in Montgomery domain or Montgomery multiplication doesn’t require a modulo m operation as multiplying two numbers in non-Montgomery form or normal multiplication requires. \( abr \pmod{m} \) can be calculated without mod m operation, unlike \( ab \pmod{m} \).

Montgomery multiplication is very important for time efficiency in ECC because the coordinates of points can be converted into Montgomery domain before doing point addition, point doubling, and scalar multiplication using the algorithms shown in section 3.3.2. Then, Montgomery multiplication can be done instead of normal multiplication when implementing the algorithms. As the previous example illustrated, normal multiplication takes more time than Montgomery multiplication because modulo m operation is totally avoided.

### 4.2 Projective Coordinates

Projective coordinates are used when performing scalar multiplication to reduce the number of inversions done as mentioned in [20], [21], thereby reducing time required for scalar multiplication as mentioned in [21]. Jacobian coordinates are the type of projective coordinates usually used in case of elliptic curves over \( \text{GF}(p) \), where p is prime.
So before performing scalar multiplication, first coordinates of the points or affine coordinates as they are called have to be converted to projective coordinates. After performing scalar multiplication, projective coordinates are converted back to affine coordinates. The conversions can be done as shown [20].

Conversion of affine coordinates \((x,y)\) to projective coordinates \((X,Y,Z)\):

\[
X \leftarrow x, 
Y \leftarrow y, 
Z \leftarrow 1
\]

Conversion of projective coordinates \((X,Y,Z)\) to affine coordinates \((x,y)\):

\[
x = X/Z^2, 
y = Y/Z^3
\]

### 4.2.1 Elliptic Curve Point Addition and Doubling Using Projective Coordinates

Once affine coordinates are converted to projective coordinates, the following algorithms for point addition and point doubling are applied [20].

Algorithm for Point Addition

\[
(X_0,Y_0,Z_0)+(X_1,Y_1,Z_1) = (X_2,Y_2,Z_2)
\]

\[
U_0 = X_0Z_1^2
\]

\[
S_0 = Y_0Z_1^3
\]

\[
U_1 = X_1Z_0^2
\]

\[
S_1 = Y_1Z_0^3
\]

\[
W = U_0 - U_1
\]

\[
R = S_0 - S_1
\]
\[
T = U_0 + U_1 \\
M = S_0 + S_1 \\
Z_2 = Z_0 Z_1 W \\
X_2 = R^2 - TW^2 \\
V = TW^2 - 2X_2 \\
2Y_2 = VR - MW^3
\]

Example for Point Addition

Two points on the curve \( y^2 \equiv x^3 + 2x + 4 \) (mod 5), \((2,4)\) and \((0,3)\), are taken.

They are converted to projective coordinates.

\((2,4)\) and \((0,3)\) are \((2,4,1)\) and \((0,3,1)\) in the projective coordinate system.

\[
X_0 = 2 \\
Y_0 = 4 \\
Z_0 = 1 \\
X_1 = 0 \\
Y_1 = 3 \\
Z_1 = 1 \\
U_0 = 2 \cdot 1^2 = 2 \\
S_0 = 4 \cdot 1^3 = 4 \\
U_1 = 0 \cdot 1^2 = 0 \\
S_1 = 3 \cdot 1^3 = 3 \\
W = 2 - 0 = 2 \\
R = 4 - 3 = 1
\]
T = 2+0 = 2
M = 4+3 = 7
Z_2 = 1*1*2 = 2
X_2 = 1^2 - (2*2^2) = 1-8 \equiv -7 \equiv 3 \pmod{5}
V = 2*2^2 - 2*3 = 2
2Y_2 = 2*1 - 7*2^3 = 2-56 = -54 \equiv 1 \pmod{5}
Y_2 = 1*(2)^{-1} = 1*3 = 3

So, point addition of (2,4,1) and (0,3,1) using projective coordinates gives (3,3,2).

These coordinates are converted to affine coordinates(x,y) by calculating \( x = X_2/Z_2^2 \), \( y = Y_2/Z_2^3 \).

\[
x = 3*3^2 = 27 \equiv 2 \pmod{5}
\]
\[
y = 3*3^3 = 81 \equiv 1 \pmod{5}
\]

The point (2,1) on the curve \( y^2 \equiv x^3+2x+4 \pmod{5} \) is obtained. This means point addition of (2,4) and (0,3) gives (2,1). This can be verified from Table 10.

**Algorithm for Point Doubling**

\[
2(X_1,Y_1,Z_1) = (X_2,Y_2,Z_2)
\]

\[
M = 3X_1^2+aZ_1^4
\]

\[
Z_2 = 2Y_1Z_1
\]

\[
S = 4X_1Y_1^2
\]

\[
X_2 = M^2-2S
\]

\[
T = 8Y_1^4
\]

\[
Y_2 = M(S-X_2)-T \ [20]
\]

\( a \) is the value from elliptic curve equation \( y^2 \equiv x^3+ax+b \pmod{p} \).
Example for Point Doubling

The point (4,4) on the curve \( y^2 \equiv x^3 + 2x + 4 \) (mod 5) is taken.

In the projective coordinate system (4,4) is (4,4,1).

\[
\begin{align*}
X_1 &= 4 \\
Y_1 &= 4 \\
Z_1 &= 1 \\
M &= 3 \times 4^2 + 2 \times 1^4 = 48 + 2 = 50 \equiv 0 \pmod{5} \\
Z_2 &= 2 \times 4 \times 1 = 8 \equiv 3 \pmod{5} \\
S &= 4 \times 4 \times 4^2 \equiv 1 \pmod{5} \\
X_2 &= 0 - 2 \times 1 = -2 \equiv 3 \pmod{5} \\
T &= 8 \times 4^4 \equiv 3 \pmod{5} \\
Y_2 &= 0 - 3 = -3 \equiv 2 \pmod{5}
\end{align*}
\]

So, doubling (4,4,1) in the projective coordinate system gives (3,2,3).

These coordinates are converted to affine coordinates (x,y) by calculating 
\( x = X_2/Z_2^2 \), \( y = Y_2/Z_2^3 \).

\[
\begin{align*}
x &= 3 \times 2^2 = 12 \equiv 2 \pmod{5} \\
y &= 2 \times 2^3 = 16 \equiv 1 \pmod{5}
\end{align*}
\]

The point (2,1) on the curve \( y^2 \equiv x^3 + 2x + 4 \) (mod 5) is obtained. This means doubling the point (4,4) gives (2,1). This can be verified from the Table 10.

As one can observe, no inversions are required in the algorithms. In point addition using affine coordinates, one inversion has to be performed. That is also the case with point doubling using affine coordinates. So, if point addition and point doubling were carried out repeatedly using
projective coordinates, as is the case for scalar multiplication, the number of inversions required will be much smaller than what would be the case if affine coordinates were used.

4.3 Combining Montgomery Multiplication and Projective Coordinates

If a combination of Montgomery residues and projective coordinates were used, the most time efficient ECC system would be obtained. In that case, to calculate aP, where P is a point on the elliptic curve, the coordinates of P can be converted to projective coordinates and then to the Montgomery domain. After repeated point addition and point doubling, where the minimum number of inversions and modulo m operations are performed, the point aP is obtained. Then, the coordinates of aP are converted back to non-Montgomery form. After this, the coordinates are converted into affine coordinate system. This time efficiency is further explained with the following example.

Example:

Point P (2,1) on the curve \( y^2 \equiv x^3+2x+4 \pmod{5} \) is taken.

In the projective coordinate system, (2,1) is (2,1,1).

In Montgomery domain, this point is (1,3,3) if \( r = 8 \).

For calculating 8P, point doubling is performed thrice using the mentioned formulae for point doubling using projective coordinates. Integer coefficients in the formulae have to be converted to Montgomery domain before applying the formulae.
\[ M = 3X_1^2 + aZ_1^4 \] becomes \[ M = 4X_1^2 + aZ_1^4. \]
\[ Z_2 = 2Y_1Z_1 \] becomes \[ Z_2 = 1Y_1Z_1. \]
\[ S = 4X_1Y_1^2 \] becomes \[ S = 2X_1Y_1^2. \]
\[ X_2 = M^2 - 2S \] becomes \[ X_2 = M^2 - 1S. \]
\[ T = 8Y_1^4 \] becomes \[ T = 4Y_1^4. \]

When performing addition and subtraction of numbers in Montgomery domain, the formula \( xr \pmod{m} + yr \pmod{m} = (x+y)r \pmod{m} \) has to be used. Whenever multiplication of two numbers in Montgomery domain is done, Montgomery product is to be calculated. \( Z_1^4 \) should be calculated as a series of 4 multiplications \( Z_1 \times Z_1 \times Z_1 \times Z_1 \), multiplying two numbers at a time. Other variables raised to a power greater than 1 should be calculated in the same way. To explain this, the value of \( M \) when calculating \( 2 \times (1,3,3) \) is shown.

\[ M = 4X_1^2 + aZ_1^4 \]
\[ a = 2 \] in Montgomery domain, \( a = 2 \times 8 \pmod{5} = 1 \)
\[ X_1 = 1 \]
\[ Y_1 = 3 \]
\[ Z_1 = 3 \]
\[ M = 4X_1^2 + 1Z_1^4 \]
\[ M = 4 \times 1 \times 1 + 1 \times 3 \times 3 \times 3 \times 3 \]
\[ M = 4 \times 2 + 1 \times 3 \times 3 \times 3 \times 3 \]
\[ M = 1 + 1 \times 3 \times 3 \times 3 \times 3 \]
\[ M = 1 + 1 \times 3 \times 3 \]
\[
M = 1 + 1 \times 3
\]

\[
M = 1 + 1 = (2 + 2) \times 8 \pmod{5} = 2 \text{ because here } x_r \pmod{m} = 1 \text{ and } y_r \pmod{m} = 1, \text{ which means } x = y = 1 \times r^{-1} \pmod{5} = 1 \times 2 \pmod{5} = 2.
\]

\[
2 \times (1, 3, 3) = (0, 2, 1)
\]

\[
4 \times (1, 3, 3) = (2, 2, 3)
\]

\[
8 \times (1, 3, 3) = (1, 3, 3)
\]

The point \(8 \times (1, 3, 3)\) or \(8P\) in non-Montgomery form is \((2, 1, 1)\). \(8P\) in the affine coordinate system is \((2, 1)\). This means \(8 \times (2, 1)\) in the affine coordinate system is \((2, 1)\). This can be verified from the Table 10.

In calculating \(8P\) using this method, only one inversion is required. That is when converting the coordinates of the point from projective coordinates to affine coordinates at the end. If \(8P\) was calculated simply using just affine coordinates and no Montgomery conversions, 3 inversions have to be performed, once when calculating \(2P\), once when calculating \(4P\), and once when calculating \(8P\). This can be verified from point addition and point doubling formulae for points in affine coordinates. Also, in this method, modulo \(m\) operation is never performed when multiplying two numbers because they are in Montgomery domain. If \(8P\) was calculated simply using just affine coordinates and no Montgomery conversions and it is assumed every multiplication of two numbers requires a modulo \(m\) operation, many modulo \(m\) operations would be required.
5. GUI Application for Elliptic Curve Arithmetic and ElGamal Cryptosystem

A Java GUI application (.jar) was designed in the Eclipse IDE environment for ECC. BigInteger class and associated cryptographic functions as well as associated general functions were used. BigInteger class was used so the application can accept very large numbers. It is part of the java.math package that comes with the JDK installation. There is no range specified for BigIntegers. They are made as large as necessary to accommodate results of an operation [23]. So, BigIntegers can be large as the computer memory available.

The application can be used to understand an ElGamal cryptosystem, perform point addition, perform point doubling, and perform scalar multiplication. In this section, how it can be used for various purposes will be explained. If users are dealing with large input and output values that don’t fit in the input text boxes and output text boxes respectively, they may simply run the Java code for the application without the GUI.

5.1 ElGamal Cryptosystem Mode

An ElGamal cryptosystem can be simulated by using the application in ElGamal cryptosystem mode. In this mode, the application takes integer inputs and gives integer outputs. Integer inputs can be as large as the computer memory available.

To show how the application in this mode can be used to simulate an ElGamal cryptosystem, we take an example elliptic curve over $\mathbb{Z}_{23}$. The particular curve we have taken is $y^2 \equiv x^3+4x+9$ (mod 23).

The points on the curve are: $(0,3)$, $(0,20)$, $(2,5)$, $(2,18)$, $(3,5)$, $(3,18)$, $(5,4)$, $(5,19)$, $(7,9)$, $(7,14)$, $(8,1)$, $(8,22)$, $(11,2)$, $(11,21)$, $(13,2)$, $(13,21)$, $(14,7)$, $(14,16)$, $(16,11)$, $(16,12)$, $(18,5)$, $(18,18)$, $(20,4)$, $(20,19)$, $(21,4)$, $(21,19)$, $(22,2)$, and $(22,21)$.
Assume Alice sends a message to Bob. She uses Bob’s public key to encrypt her message. Bob decrypts the encrypted message using his private key. The user enters the following: (i) Bob’s point on the curve which is part of his public key, (ii) Bob’s private key, (iii) Alice’s message or point on the curve, (iv) Alice’s private key, (v) the prime \( p \) from the equation \( y^2 \equiv x^3 + ax + b \) (mod \( p \)), (vi) the value of \( a \) from the equation \( y^2 \equiv x^3 + ax + b \) (mod \( p \)), and (vii) the value of \( b \) from equation \( y^2 \equiv x^3 + ax + b \) (mod \( p \)). The inputs to be entered are shown in Figure 6. One main reason values of \( a \) and \( b \) are entered is check if the condition \( 4a^3 + 27b^2 \neq 0 \) is met.

Then, when the user clicks simulate, Bob’s public key is generated. The encrypted message sent by Alice is also generated. Finally, Alice’s original message that Bob obtains after decrypting the encrypted message is generated. This is shown in Figure 7.

![Figure 6: Setup for ElGamal Cryptosystem Mode](image-url)
5.2 Arithmetic Mode

The application can be used in arithmetic mode to perform point addition, point doubling, or scalar multiplication using any point on any elliptic curve over a prime field. In this mode, the application takes integer inputs and gives integer outputs. Integer inputs can be as large as the computer memory available. For point addition, (i) point P on the elliptic curve, (ii) another point Q on the elliptic curve, (iii) the value of a from the equation $y^2 \equiv x^3+ax+b \pmod{p}$, (iv) the value of b from the equation $y^2 \equiv x^3+ax+b \pmod{p}$, and (v) prime p from the equation $y^2 \equiv x^3+ax+b \pmod{p}$ are entered. This is shown in Figure 8. One of the main reasons values of a and b of the elliptic curve equation are entered is to check if the condition $4a^3+27b^2 \neq 0$ is met. Results obtained are shown in Figure 9.
Figure 8: Setup for Point Addition

Figure 9: Output of Application for Point Addition
For point doubling, (i) point P on the elliptic curve, (ii) the value of a from the equation $y^2 \equiv x^3 + ax + b \pmod{p}$, (iii) the value of b from the equation $y^2 \equiv x^3 + ax + b \pmod{p}$, (iv) prime p from the equation $y^2 \equiv x^3 + ax + b \pmod{p}$, and (v) the scalar multiplier $c = 2$ are entered. This is shown in Figure 10. One of the main reasons values of a and b of the elliptic curve equation are entered is to check if the condition $4a^3 + 27b^2 \neq 0$ is met. Results are shown in Figure 11.

![Figure 10: Setup for Point Doubling](image)
For scalar multiplication, (i) point \( P \) on the elliptic curve, (ii) the value of \( a \) from the equation \( y^2 \equiv x^3 + ax + b \pmod{p} \), (iii) the value of \( b \) from the equation \( y^2 \equiv x^3 + ax + b \pmod{p} \), (iv) prime \( p \) from the equation \( y^2 \equiv x^3 + ax + b \pmod{p} \), and (v) scalar multiplier \( c \) are entered. \( c \) is assigned a value greater than or equal to 2 here. This is shown in Figure 12. One of the main reasons values of a and b of the elliptic curve equation are entered is to check if the condition \( 4a^3 + 27b^2 \neq 0 \) is met. Results are shown in Figure 13.
Figure 12: Setup for Scalar Multiplication

Figure 13: Output of Application for Scalar Multiplication
6. Conclusions and Future Work

As we have shown, ECC has its own unique arithmetic for doing point addition, point doubling, and scalar multiplication with points on an elliptic curve. Elliptic curve arithmetic is important for understanding ECC-based cryptosystems.

ECC-based cryptosystems are important to understand as they are improving browser and mobile security [6], [30], [33], [34]. This is because ECC is superior in key strength to the most popular public key cryptography scheme, which is RSA. A smaller key size in ECC provides the same security level as a much bigger key size in RSA [1], [2], [3], [4], [25]. ECC is also superior in performance and in suitability for resource-constrained devices [5], [24], [26]. It is a better option than RSA for providing security for DNSSEC and IoT systems and devices [30], [31], [32]. Devices like phones and tablets are all resource-constrained devices. So, ECC is likely to replace competing cryptography schemes like RSA in many applications.
References


Appendix A-User Manual for GUI Application

A.1 Instructions for Opening the Application

1. Download the application ECC.jar.

2. Install the java runtime environment (JRE) because the application is an executable jar.

3. Double click on ECC.jar executable jar file.

A.2 Instructions for Using the Application

The application can be used in two modes. Separate instructions apply to each mode. In case the text fields are small for the sizes of input or output values, please contact Dr. Carla Purdy at purdycc@ucmail.uc.edu to obtain the Java code for the application without the GUI.

A.2.1 Instructions for ElGamal Cryptosystem Mode

1. From the main menu, which is shown in Figure A.1, select ElGamal cryptosystem mode.

![Figure A.1: Main Menu Screen](image)
2. The next screen will be as shown in Figure A.2. It contains instructions for using the application in this mode. Read the instructions and to proceed, click next. Click back to return to the main menu.

Figure A.2: Instructions for ElGamal Cryptosystem Mode

3. The next screen is shown in Figure A.3. Enter the chosen values.
4. Click on Simulate. Values of all the text fields can be then reset by clicking the Reset button and steps 3 and 4 can be repeated. To return to the main menu, click on the Return to the Main Menu button.

A.2.2 Instructions for Arithmetic Mode

In this mode, three operations can be performed using the application. They are point addition, point doubling, and scalar multiplication. Depending on the operation to be performed, the instructions to use the application in this mode vary.
1. Click on Arithmetic Mode from the main menu. The next screen will be as shown in Figure A.4. It contains instructions for using the application in this mode. Read the instructions and to proceed, click next. The next screen is as shown in figure A.5. Click back to return to the main menu.

(a) For Performing Point Addition

2. Enter the chosen values.

3. Select only the P+Q checkbox.

4. Click calculate. Values of all the text fields can be then reset by clicking the Reset button and steps 2 and 4 can be repeated. To return to the main menu, click on the Return to the Main Menu button.
(b) For Performing Point Doubling

2. Enter the chosen values.

3. Select only cP checkbox.

4. Click calculate. Values of all the text fields can be then reset by clicking the Reset button and steps 2 and 4 can be repeated. To return to the main menu, click on the Return to the Main Menu button.

(c) For Performing Scalar Multiplication

2. Enter the chosen values.

3. Select only cP checkbox.
4. Click calculate. Values of all the text fields can be then reset by clicking the Reset button and steps 2 and 4 can be repeated. To return to the main menu, click on the Return to the Main Menu button.
Appendix B – Module

B.1 Questions

1. Do the points on the following elliptic curves over GF(p) and point at infinity O form an Abelian group with point addition operator $+_e$? Explain your answers.

(i) $y^2 \equiv x^3+3x+7 \pmod{17}$
(ii) $y^2 \equiv x^3+5x+4 \pmod{19}$
(iii) $y^2 \equiv x^3+10x+14 \pmod{23}$
(iv) $y^2 \equiv x^3+7x+6 \pmod{13}$

2. In an ElGamal cryptosystem, Bob’s public key is $(23, (2,18), (21,19))$. Alice sends her encrypted message, which is $((7,9),(18,18))$, to Bob. The elliptic curve chosen by Alice and Bob for the operation of their cryptosystem is $y^2 \equiv x^3+4x+9 \pmod{23}$. Decrypt Alice’s encrypted message to retrieve her original message if Bob’s private key is 17.

3. Perform the following operations on the elliptic curve $y^2 \equiv x^3+4x+9 \pmod{23}$ using algebraic formulae for elliptic curves over a prime field.

(i) $(16, 11) +_e (8, 22)$
(ii) $(3,5) +_e (18,18)$
(iii) $3*(5,19)$
(iv) $(21,4) +_e (0,20)$

4. Construct the point addition table for the elliptic curve $y^2 \equiv x^3+6x+12 \pmod{7}$. 
5. Bob and Alice choose the elliptic curve $y^2 \equiv x^3 + 4x + 9 \pmod{23}$ for the operation of their ElGamal cryptosystem. Bob’s public key is $(23, (11,21), (0,20))$. Bob and Alice agree to use a system where some points on the curve represent letters in the alphabet. This is their system.

P- (2,5)
A- (16,12)
O- (8,1)
B- (14,7)
S- (22,2)
R- (14,16)
K- (20,4)
L- (11,2)
W- (7,14)
D- (0,20)
M- (3,18)
T- (5,4)
K- (20,19)

Alice sends the following encrypted messages to Bob in sequence: (i)((5,19),(16,12)), (ii) ((5,19),(7,9)), (iii) ((5,19),(18,18)), (iv) ((5,19),(18,18)) (v) ((5,19),(16,11)), (vi) ((5,19),(20,19)), (vii) ((5,19),(13,21)) , and (viii) ((5,19),(21,4)). Decrypt the messages if Bob’s private key is 17.

6. If you were to create a secure mobile communication network, would it better to secure the network using Elliptic Curve Cryptography (ECC) or RSA? Why so?
7. Arrange the following public key cryptography schemes in decreasing order of security strength that they provide.

(i) 224-bit ECC
(ii) 3072-bit RSA
(iii) 384-bit ECC
(iv) 1024-bit RSA
(v) 256-bit ECC
(vi) 2048-bit RSA

8. What is the discrete logarithm problem in ECC? Does it have to be solved in order to break a RSA scheme too? If that is not the problem that has to be solved to break a RSA scheme, what problem has to be solved?

9. What are the steps to be performed to subtract P (x1,y1) from Q (x2,y2), which are two points on an elliptic curve over GF(p)? Can we do the subtraction directly?

10. Evaluate (i)-(iv) and check to see which will produce the same point as 16P, where P is a point on an elliptic curve over GF(p).

(i) 2*(8P)
(ii) 2*(4P) +e 2*(4P)
(iii) 2*(3P) +e 7P +e 4P - 2P
(iv) 2*(2*(4P)) - 3P +e 2P +e P
B.2 Solutions

1.

(i)
The points \((x,y)\) on the curve \(y^2 \equiv x^3 + ax + b \pmod{p}\) along with point at infinity \(O\) form an Abelian group with point addition operator \(+_e\) if \(4a^3 + 27b^2 \neq 0\).

\[ 4a^3 + 27b^2 = 4 \cdot (3)^3 + 27 \cdot (7)^2 = 108 + 1323 = 1431 \equiv 3 \pmod{17} \]

So, \(4a^3 + 27b^2 \neq 0\) and the points on the elliptic curve \(y^2 \equiv x^3 + 3x + 7 \pmod{17}\) along with point at infinity \(O\) form an Abelian group with point addition operator \(+_e\).

(ii)

\[ 4a^3 + 27b^2 = 4 \cdot (5)^3 + 27 \cdot (4)^2 = 500 + 432 = 932 \equiv 1 \pmod{19} \]

So, \(4a^3 + 27b^2 \neq 0\) and the points on the elliptic curve \(y^2 \equiv x^3 + 5x + 4 \pmod{19}\) along with point at infinity \(O\) form an Abelian group with point addition operator \(+_e\).

(iii)

\[ 4a^3 + 27b^2 = 4 \cdot (10)^3 + 27 \cdot (14)^2 = 4000 + 5292 = 9292 \equiv 0 \pmod{23} \]

So, \(4a^3 + 27b^2 = 0\). The points on the elliptic curve \(y^2 \equiv x^3 + 10x + 14 \pmod{23}\) along with point at infinity \(O\) don’t form an Abelian group with point addition operator \(+_e\).

(iv)

\[ 4a^3 + 27b^2 = 4 \cdot (7)^3 + 27 \cdot (6)^2 = 1372 + 972 = 2344 \equiv 4 \pmod{13} \]

So, \(4a^3 + 27b^2 \neq 0\) and the points on the elliptic curve \(y^2 \equiv x^3 + 7x + 6 \pmod{13}\) along with point at infinity \(O\) form an Abelian group with point addition operator \(+_e\).
2.

To decrypt Alice’s message, the following has to be done:

\[(18,18) - 17*(7,9) = (18,18) - (22,2) = (18,18) + e (22,21) = (8,1)\]

3.

(i)

\[(x_p,y_p) + e (x_q,y_q) = (x_R,y_R)\]

\[x_p = 16, \quad y_p = 11, \quad x_q = 8 \quad \text{and} \quad y_q = 22\]

\[y_p - y_q = 11 - 22 \equiv -11 \equiv 12\]

\[x_p - x_q = 16 - 8 = 8\]

\[(x_p - x_q)^{-1} \equiv 3 \pmod{23}\]

\[s = (y_p - y_q)(x_p - x_q) = 12 \cdot 3 = 36 \equiv 13 \pmod{23}\]

\[x_R = s^2 - x_p - x_q \pmod{p}\]

\[s^2 = 13^2 = 169 \equiv 8 \pmod{23}\]

\[s^2 - x_p = 8 - 16 = -7 \equiv 16 \pmod{23}\]

\[s^2 - x_p - x_q = 16 - 8 \equiv 8 \pmod{23}\]

So, \(x_R = 8\).

\[y_R = -y_p + s(x_p - x_R) \pmod{p}\]

\[-y_p = -11 \equiv 12 \pmod{23}\]

\[x_p - x_R = 16 - 8 = 8\]

\[s(x_p - x_R) = 13 \cdot 8 = 104 \equiv 12 \pmod{23}\]
\[ y_R = -y_p + s(x_p - x_R) = 12 + 12 \equiv 1 \pmod{23} \]

So, \((16, 11) +_e (8, 22) = (8, 1)\).

(ii)
\[(x_p, y_p) +_e (x_q, y_q) = (x_R, y_R)\]
\[x_p = 18, \ y_p = 18, \ x_q = 3 \text{ and } y_q = 5\]
\[y_p - y_q = 18 - 5 = 13\]
\[x_p - x_q = 18 - 3 = 15\]
\[(x_p - x_q)^{-1} \pmod{23} = 20\]
\[s = (y_p - y_q)/(x_p - x_q) = (y_p - y_q)^* (x_p - x_q)^{-1} = 13*20 = 260 \equiv 7 \pmod{23}\]

\[x_R = s^2 - x_p - x_q \pmod{p}\]
\[s^2 = 7^2 = 49 \equiv 3 \pmod{23}\]
\[s^2 - x_p = 3 - 18 = -15 \equiv 8 \pmod{23}\]
\[s^2 - x_p - x_q = 8 - 3 \equiv 5 \pmod{23}\]

So, \(x_R = 5\).

\[y_R = -y_p + s(x_p - x_R) \pmod{p}\]
\[-y_p = -18 \equiv 5 \pmod{23}\]
\[x_p - x_R = 18 - 5 = 13\]
\[s(x_p - x_R) = 7*13 = 91 \equiv 22 \pmod{23}\]
\[y_R = -y_p + s(x_p - x_R) = 5 + 22 = 4 \pmod{23}\]
\((3,5) +_e (18,18) = (5,4)\)
(iii)

\[ 3 \cdot (5, 19) = 2 \cdot (5, 19) + e \cdot (5, 19) \]

First, \( 2 \cdot (5, 19) \) has to be calculated.

If, \( 2(x_p, y_p) = (x_R, y_R) \)

\( x_p = 5, \; y_p = 19 \)

\[ s = (3x_p^2 + a) \cdot (2y_p)^{-1} \pmod{p} \]

\[ 3 \cdot x_p^2 = 3 \cdot 25 \equiv 75 \equiv 6 \pmod{23} \]

\( a = 4 \)

\[ 3x_p^2 + a = 6 + 4 = 10 \]

\[ 2 \cdot y_p = 38 \equiv 15 \pmod{23} \]

\[ (2y_p)^{-1} = 15^{-1} \pmod{23} = 20 \]

\[ s = 10 \cdot 20 = 200 \equiv 16 \pmod{23} \]

Next, \( x_R \) and \( y_R \) have to be calculated.

\[ x_R = s^2 - 2x_p \pmod{p} \]

\[ s^2 = 16^2 = 256 \equiv 3 \pmod{23} \]

\[ 2 \cdot x_p = 10 \]

\[ s^2 - 2 \cdot x_p = 3 - 10 = -7 \equiv 16 \pmod{23} \implies x_R = 16 \]

\[ y_R = -y_p + s(x_p - x_R) \pmod{p} \]

\[ -y_p = -19 \equiv 4 \pmod{23} \]

\[ s(x_p - x_R) = 16 \cdot (5 - 16) = 16 \cdot (-11) = 16 \cdot 12 \equiv 8 \pmod{23} \]

\[ -y_p + s(x_p - x_R) = 4 + 8 = 12 \]
So, $y_R = 12$. So, $2^*(5,19) = (16,12)$.

$(16,12) +_e (5,19) = (14,16)$ if the algebraic formulae for point addition, which were applied in 1 and 2, are applied here.

(iv).

$(21,4) +_e (0,20) = (20,4)$ if the algebraic formulae for point addition, which were applied in (i) and (ii), are applied here.

4.

The points on the elliptic curve $y^2 \equiv x^3 + 6x + 12 \pmod{7}$ are: $(2,2)$, $(2,5)$, $(3,1)$, $(3,6)$, $(4,3)$, and $(4,4)$. Here is the point addition table.

<table>
<thead>
<tr>
<th></th>
<th>(2,2)</th>
<th>(2,5)</th>
<th>(3,1)</th>
<th>(3,6)</th>
<th>(4,3)</th>
<th>(4,4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,2)</td>
<td>(4,3)</td>
<td>O</td>
<td>(3,6)</td>
<td>(4,4)</td>
<td>(3,1)</td>
<td>(2,5)</td>
</tr>
<tr>
<td>(2,5)</td>
<td>O</td>
<td>(4,4)</td>
<td>(4,3)</td>
<td>(3,1)</td>
<td>(2,2)</td>
<td>(3,6)</td>
</tr>
<tr>
<td>(3,1)</td>
<td>(3,6)</td>
<td>(4,3)</td>
<td>(2,5)</td>
<td>O</td>
<td>(4,4)</td>
<td>(2,2)</td>
</tr>
<tr>
<td>(3,6)</td>
<td>(4,4)</td>
<td>(3,1)</td>
<td>O</td>
<td>(2,2)</td>
<td>(2,5)</td>
<td>(4,3)</td>
</tr>
<tr>
<td>(4,3)</td>
<td>(3,1)</td>
<td>(2,2)</td>
<td>(4,4)</td>
<td>(2,5)</td>
<td>(3,6)</td>
<td>O</td>
</tr>
<tr>
<td>(4,4)</td>
<td>(2,5)</td>
<td>(3,6)</td>
<td>(2,2)</td>
<td>(4,3)</td>
<td>O</td>
<td>(3,1)</td>
</tr>
</tbody>
</table>

5.

Decrypting the encrypted message sent by Alice, we get:

(i) $(16,12) - 17^*(5,19) = (16,12) - (8,1) = (16,12) +_e (8,22) = (2,5) \rightarrow P$
(ii) \((7,9) - 17*(5,19) = (7,9) - (8,1) = (7,9) + e (8,22) = (16,12) \rightarrow A\)

(iii) \((18,18) - 17*(5,19) = (18,18) + e (8,22) = (22,2) \rightarrow S\)

(iv) \((18,18) - 17*(5,19) = (18,18) + e (8,22) = (22,2) \rightarrow S\)

(v) \((16,11) - 17*(5,19) = (16,11) + e (8,22) = (7,14) \rightarrow W\)

(vi) \((20,19) - 17*(5,19) = (20,19) + e (8,22) = (8,1) \rightarrow O\)

(vii) \((13,21) - 17*(5,19) = (13,21) + e (8,22) = (14,16) \rightarrow R\)

(viii) \((21,4) - 17*(5,19) = (21,4) + e (8,22) = (0,20) \rightarrow D\)

So, Alice’s actually sent the text ‘PASSWORD’ using ECC.

6.
It would be better to secure the network using ECC. ECC is more suitable for resource-constrained devices like mobiles than RSA. ECC needs fewer resources and memory, which are mobile features. ECC also requires less computational overhead compared to RSA.

7.
The order of the public key cryptography schemes in decreasing order of security strength is 384-bit ECC > 3072-bit RSA, 256-bit ECC > 224-bit ECC, 2048-bit RSA > 1024-bit RSA.

8.
The discrete logarithm problem in ECC is finding an integer \(c\) such that \(Q = cP\) if \(Q\) and \(P\) are two points on an elliptic curve over a prime field.

No, the discrete logarithm problem doesn’t have to be solved to break a RSA scheme. Instead, the integer factorization problem of factoring the system modulus \(n\) has to be solved.
9.

To perform $Q - P$, the steps to be performed are as follows.

(i) Find the negative of the point $P$. $-P$ is $(x_1, -y_1)$.

(ii) Add $Q$ and $-P$, i.e. $Q + (-P)$.

No, we cannot do the subtraction of $P$ from $Q$ directly.

10.

(i) $2*(8P) = 16P$

(ii) $2*(4P) + 2*(4P) = 8P + 8P = 16P$

(iii) $2*(3P) + 7P + 4P - 2P = 6P + 7P + 2P = 15P$

(iv) $2*(2*(4P)) - 3P + 2P + P = 2*(8P) - 3P + 2P + P = 16P - 3P + 3P = 16P$

(i), (ii), and (iv) all produce the same point as $16P$.