CAPITAL UNIVERSITY OF SCIENCE AND TECHNOLOGY, ISLAMABAD

A Computational Package for Working with Elliptic Curve Groups

by

Sulman Liaquat

A thesis submitted in partial fulfillment for the degree of Master of Philosphy

in the

Faculty of Computing
Department of Mathematics

2020

To my Mother, late Father, Brothers, Sisters and Someone very special for their support and love.

CERTIFICATE OF APPROVAL

A Computational Package for Working with Elliptic Curve Groups

by

Sulman Liaquat

(MMT161012)

THESIS EXAMINING COMMITTEE

S. No.	Examiner	Name	Organization
(a)	External Examiner	Dr. Waqas Mehmood	QAU, Islamabad
(b)	Internal Examiner	Dr. Qamar MEhmood	CUST, Islamabad
(c)	Supervisor	Dr. Rashid Ali	CUST, Islamabad

Dr. Rashid Ali
Thesis Supervisor
June, 2020

Dr. Muhammad Sagheer	Dr. Muhammad Abdul Qadir
Head	Dean
Dept. of Mathematics	Faculty of Computing
June, 2020	June, 2020

Author's Declaration

I, **Sulman Liaquat** hereby state that my M. Phil thesis titled "**A Computational Package for Working with Elliptic Curve Groups**" is my own work and has not been submitted previously by me for taking any degree from Capital University of Science and Technology, Islamabad or anywhere else in the country/abroad.

At any time if my statement is found to be incorrect even after my graduation, the University has the right to withdraw my M. Phil Degree.

(Sulman Liaquat)

Registration No: MMT161012

Plagiarism Undertaking

(Sulman Liaquat)

Registration No: MMT161012

Acknowledgements

All praise to Almighty Allah, the most Benevolent and Merciful, the Creator of Universe and man, who gave me the vision and courage to accomplish this work successfully. A research project at any level is very difficult to be accomplished alone. The contributions of many people have made it possible for me to complete this work. I would like to extend my appreciation especially to the following.
First and foremost I extend my sincerest gratitude to my thesis supervisor **Dr. Rashid Ali** who has supported me throughout my thesis work in every possible way. He presented the things in simplest way. His efforts and encouragement is really appreciable. He gave me self-belief, and confidence and provided me his support and guidance all way along. Under the supervision of him I never felt anything difficult in my thesis. **Dr. Rashid Ali** provided me the friendly and comfortable environment to work in, without his guidance and support I would never be able to put this topic together.
I am also thankful to all the faculty of CUST in particular the faculty of Mathematics department. Their ability to simplify the most difficult things helped me to understand the things that I would never be able to understand.
The research oriented environment provided to students by **Mian Amer Mehmood**, Chancellor at CUST, **Prof. Dr. Muhammad Mansoor Ahmed**, Vice Chancellor at CUST and **Dr. Muhammad Sagheer**, Head of Department of Mathematics made the research work easier and more pleasant. It is their positive attitude that enables such excellent research work from staff and students.
My friends, class mates and fellow research workers **Tahir Bhai**, **Syed Burhan** , **M.Usman Farooqi** , **Mujtaba Azim**, **Sohail Abid**, **Bilal Ahmed** and **Sultan Mehmood** also helped me a lot and guided me whenever I needed it.
Last but not least, I would like to pay high regards to my parents and all family members for their sincere encouragement and inspiration throughout my research work and lifting me uphill this phase of life. I owe everything to them.

(Sulman Liaquat)

Registration No: MMT161012

Abstract

In modern world, security of confidential information has become one of the fundamental requirement. Elliptic curve turn out to be an important tool for secure communication in cryptography it provides the same level of security with smaller key as compare to RSA. For working with the elliptic curves over the finite fields of very large orders we need a mathematical tool like Matlab, ApCoCoA, Mathematica etc. Matlab and Mathematica are not freely available softwares. In this thesis we have developed a package in ApCoCoA, which is a freely available mathematical software for computation in Algebra, that helps us to perform some complex computations in an easy and efficient way. We used CoCoL programming language of ApCoCoA to make a computational package for working with the elliptic curve groups. This package is made to perform different operations on elliptic curve that are necessary to run schemes based on ECC. Also, use of ApCoCoA is secure as compared to other online available softwares as it needs not to be connected with a third party like desmos, over a public network. Using online available softwares can compromise the security because one must share the EC points over internet to perform different calculations. As the security of many schemes are based on EC points, therefore, if a hacker, pretending to be the third party, can take EC points from the user, can weaken the security. Our package works off line and hence it is better for security perspective. Once the message is encrypted it can be sent to the receiver over a public network without compromising the security because it can only be understandable to a authorized receiver as it is in encrypted form. We also efficiently executed using the package made in ApCoCoA.

Contents

List of Figures

List of Tables

Abbreviations

AES	Advanced Encryption Standard
ApCoCoA	Applied Computations in Commutative Algebra
DES	Data Encryption Standard
DH	Diffie-Hellman Key Exchange Protocol
DLPl	Discrete Logarithm Problem
DSA	Digital Signature Algorithm
ECDLP	Elliptic Curve Discrete Logarithm Problem
GCD	Greatest Common Divisor
RSA	Rivest-Shamir-Adleman

Symbols

P	Plaintext or Message
C	Ciphertext
E	Encryption Algorithm
D	Decryption Algorithm
K	Key
E_k	Encryption key
D_k	Decryption key
$\mathbb{Z}$	Set of Integers
$\mathbb{R}$	Set of Real Numbers
$\mathbb{Z}_p$	Finite Field Of Order Prime p

Chapter 1

Introduction

The security of communication remained a big problem from very beginning. Roman knew some cryptographic methods and used the Shift Cipher or Caesar Cipher [1] while communicating with each other. As the time passed, new methods in cryptography were developed that provided more security. Cryptography is the study of transmitting a message in a form so that no third party can read or gain a control it. It is the technique that uses mathematical functions for securing the data or information from adversaries. The original message known as plaintext is converted into coded message (ciphertext) via the encryption algorithm and with the help of key (a secret information) for transmitting it to the public network. The ciphertext is then converted back to plaintext by the receiver or an authorized person via the decryption algorithm again with the help of a key. Cryptographic techniques are categorized into two types; technique that uses same key for encryption/decryption is called symmetric key [2] cryptography whereas a technique that uses two different keys, but there must be some mathematical relation between the two keys, is called asymmetric key cryptography. In asymmetric key [3] cryptography one key is recognized as public key of the user and the other one is called the private key.

1.1 Background

In this thesis, our entire focus will be on asymmetric key cryptography. The famous mathematical techniques that have been used in the history for asymmetric key cryptography [3] are integer factorization problem and discrete logarithm problem. In integer factorization two large primes are used which when multiplied gives an integer. The security of the system mainly relies on the difficulty of factorizing that integer. Note that all the work in integer factorization is done in some finite field. On the other hand, in discrete logarithm problem. A multiplicative cyclic group is selected to define a discrete logarithm problem using the generator of the group. We will see these techniques in detail in the next chapter.
There is another branch that tries to reveal the secret information in a secure communication without having the idea of key or to obtain the key illegally. This branch is known as cryptanalysis. Discrete logarithm problem provides a fully secured cryptosystem when a finite field of 1024 bits is used. Since computations in 1024 bit finite field take a lot of time and are also a bit complicated so there had been a need of some improved method that could provide the same security with less computational cost.

1.2 Role of Elliptic Curve

As described in the previous section that to obtain high security a relatively large field must be used. In large fields computational cost of an algorithm also gets bigger which in some cases become inefficient and obtaining desired results become an issue for the user. To attain the same security with less computational cost, Neil Koblitz and Victor Miller [4], in 1985, defined the use of elliptic curve [5] in cryptography which was major breakthrough in cryptography. It was completely a new idea. An elliptic curve is an equation in two variables one of which has degree two and the other is of degree 3. The general form is:

$$y^2 = ax^3 + bx + c$$

One can say that it was an advancement of discrete logarithm problem. Because this time discrete logarithm problem was defined over an elliptic curve which gave the same level of security only by working 128 bit finite field. In Chapter 3, we will look into more details of elliptic curves and their relation with cryptography.

1.3 Tools for Elliptic Curve

To make use of these techniques, one must use some mathematical software that does all the computations and execute the algorithm. Because manual calculations cannot be done when working in bigger fields. There are many tools like MATLAB[6], Mathematica [7] and other programs that are working perfectly fine to implement different cryptographic schemes in our daily life. There are calculators which are available online like DESMOS [8],Cryptomath [9], Christomath [10] etc, which have been in use for years. The objective of these calculators is to enable the user to perform operations on elliptic curves like point addition (same or distinct), n times addition, order of a point, sketching a graph of elliptic curve etc. In this thesis, we will introduce the use of ApCoCoA to implement cryptographic techniques. A package in ApCoCoA is made that is helpful to find the points lying on elliptic curve, addition of these points to obtain resultant points etc. There was no uch package available in ApCoCoA before.

1.4 Summary

In this thesis, the purpose of making a package in ApCoCoA is all about algebraic calculations, it has different built-in functions of algebraic computations that can be used to make an advance package which is helpful for performing operations on elliptic curve. We used this package to implement elliptic curve Diffie Hellman key exchange protocol. The scheme runs smoothly and the operations on elliptic curve are computed using the package that is made in ApCoCoA. Since ECC is related to basic algebra so it sets perfect platform for us to make such a package. ApCoCoA is also user-friendly and simple to use that is why we chose it.

1.5 Distribution of Thesis

In chapter 2, we will only be dealing with some basic stuff related to basic algebra and cryptography to facilitate the reader understanding the terms in a best way. Chapter 3 is regarding the use of elliptic curve in cryptography together with elliptic curve discrete logarithm problem and some elliptic curve cryptosystems for a program to run efficiently. All the basics about ApCoCoA and different programs which are compiled in a single package will be discussed in Chapter 4.

Chapter 2

Preliminaries

This chapter deals with some basic definitions and notations which are going to help us accomplishing this thesis. In the first section we will recall terms like group, ring, field together with some examples. Then in the next section, we will define cryptography and look into the types of it. The possible attacks on a cryptosystem will also be covered in this chapter. Last section is devoted on the discussion on elliptic curve cryptography.

2.1 Mathematical Background

In this section we review some mathematical background from algebra which are necessary for a good understanding of the chapters in this thesis.

Definition 2.1.1. (Groups)
A group [11] G is a finite or infinite set of elements together with a binary operation (called the group operation) that satisfy the four fundamental properties: The operation with respect to which a group is defined is often called the “group operation”, and a set is said to be a group “under” this operation. Elements $x, y, z, \ldots$ with binary operation between x and y denoted $x * y$ form a group if

1. **Closure law**: If x and y are two elements in G, then the product xy is also in G or

   $x * y \in G$

2. **Associativity**: The defined multiplication is associative, i.e., for all x, y, z in G,

   $(x * y) * z = x * (y * z)$.

3. **Identity Element**: There is an identity element I (such as $1, E, ore$) such that

   $I * x = x * I = x$ for every element x in G.

4. **Inverse Element**: There must be an inverse (reciprocal) of each element. Therefore, for each element x of G, the set contains an element $y = x^{-1}$ such that $x * x^{-1} = x^{-1} * x = I$

Definition 2.1.2. (Abelian Groups)

If one more property holds with four properties of group G,

$$x * y = y * x$$

for all $x, y \in G$ then it is known as **abelian group** [11]. Example of abelian group are the set $\mathbb{R}$ of $\mathbb{R}$ real number and $\mathbb{Z}$ the set of integers with respect to addition.

Example 2.1.3. Real numbers $\mathbb{R} \setminus \{0\}$ with respect to multiplication is an example of abelian group.

1. Closure property holds since $a \cdot b \in \mathbb{R} \setminus \{0\}$, $\forall a, b \in \mathbb{R} \setminus \{0\}$.

2. Associativity also holds as $(a \cdot b) \cdot c = a \cdot (b \cdot c)$, $\forall a, b, c \in \mathbb{R} \setminus \{0\}$.

3. 1 the is identity element.

4. Inverse $\frac{1}{a}$ of each nonzero a element also exists.

5. Furhter, $a \cdot b = b \cdot a$, $\forall a, b \in \mathbb{R} \setminus \{0\}$. Hence, it is an abelian group.

Definition 2.1.4. (Cyclic Groups)

Group G is known as cyclic group [11], if there exists that $x \in G$ can be generate every element of G. This x is known as the generator of of the group G and we write $G = \langle x \rangle$. Thus for some element x every $u \in G$ has the form x^n. Note that every cyclic group is an abelian group.
Suppose $a, b \in G$ and G be a cyclic group.
Then $u = x^n$ and $v = x^m$
$\Rightarrow uv = x^n x^m = x^{n+m} = x^{m+n} = x^m x^n = vu.$

Example 2.1.5. The set of integers $\mathbb{Z}$ holds all the properties of being a group. Moreover, with respect to addition, all the elements of $\mathbb{Z}$ can be generated by -1 and 1. So, it is a cyclic group with -1 and 1 as generators.

Definition 2.1.6. (Order of an Element in G)

For a cyclic group [11] G with "a'' as its generator, the smallest whole number n that gives the identity "e' in G, is termed as the order of an element "a'' in G. Mathematically it is written as,

$$a^n = e \quad a \in G, \quad n \in \mathbb{W}$$

Definition 2.1.7. (Ring)

A non-empty set R together with the pair of algebric operations '$\cdot$' and '$+$' is said to be a ring [11] of. For all $x, y, z \in R$, the following properties hold:

1. **Abelian**: Ring R is an abelian group under $+$.
2. **Associativity**: The operation . is associative and it is of course closed also

$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

3. **Distributivity**: The set R also holds both distributive laws:

$$(y + z) \cdot x = y \cdot x + z \cdot x$$

$$x \cdot (y + z) = x \cdot y + x \cdot z$$

Definition 2.1.8. (Commutative Ring)

If a ring R further holds that

$$x \cdot y = y \cdot x$$

then it is called commutative ring [11].

Example 2.1.9. Set of integers $\mathbb{Z}$ is a commutative ring.

1. Since $a + b = b + a$, $\forall a, b \in \mathbb{Z}$, so it is an abelian group under addition.

2. The operation
$$(a \cdot b) \cdot c = a \cdot (b \cdot c)$$
also exists $\forall a, b, c \in \mathbb{Z}$. So, it is associative.

3. Distributive property also exists $\forall a, b, c \in \mathbb{Z}$, i.e.,
$$(b + c) \cdot a = b \cdot a + c \cdot a$$
$$a \cdot (b + c) = a \cdot b + a \cdot c$$

4. Further, $\forall a, b \in \mathbb{Z}$
$$a \cdot b = b \cdot a$$
Hence, it is a commutative ring.

Definition 2.1.10. (Field)

Consider a ring $(F, +, \cdot)$. If the non-zero elements of $\mathbb{F}$ form an ablian group under multiplication, then the ring $\mathbb{F}$ is called a field [11].

Examples of field includes $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{C}$ wit respect to usual operations of addition and multiplication .

Further, a field with finite number of elements is called a finite field. The Galois fields are example of finite fields. These are denoted by $GF(p^n)$ where p is prime number.

When $n = 1$, the elements of $GF(p)$ are given in the set $0, 1, 2, 3, ..., p - 1$. The set of nonzero elements of $GF(p)$ form a group under multiplications modulo p.

Definition 2.1.11. Order of a Field

By order of a field, we mean that how many elements a field has. For example, if a filed has 4 elements then its order will be 4.

Definition 2.1.12. Galois or Finite Field

If a field has finite number of elements then it is recognized as finite or Galois field. The notation for Galois field is $GF(q)$, where $q = p^n$ and p is a prime number.
In cryptography, we further categorized it in two types. In the first type, we take $n = 1$ which gives us a field of prime numbers.
In the second type, we set $p = 2$ which gives us polynomials of maximum $n - 1$ degree, whose coefficients are only 0 and 1.

Next, our aim is to show multiplication and addition in finite fields. To find the multiplicative inverse of a number in a given finite field, we will use an algorithm called **Extended Euclidean Algorithm**. Here it is given:

Algorithm 2.1.13. (Extended Euclidean Algorithm)

To find the inverse of b under modulo m, below mentioned steps are to be followed:

Input: b and m
Output: $b^{-1} \mod m$

```
1. Set $(X, Y, Z) = (1, 0, m)$ and $(L, M, N) = (0, 1, b)$
2. **If** $N = 0$, **return** that the inverse does not exist and $Z$ is the *gcd* of $(b, m)$.
3. **If** $N = 1$, **return** that the inverse is $M$ and $N$ is the *gcd* of $(b, m)$
4. Store $T = \lfloor Z/N \rfloor$, where $\lfloor . \rfloor$ represents the floor value.
5. $(P, Q, R) = (X - TL, Y - TM, Z - TN)$
6. $(X, Y, Z) = (L, M, N)$
7. $(L, M, N) = (P, Q, R)$
8. Go back to step no.2
```

For the fields $GF(13)$, the operations are shown in the addition and multiplication are shown in the Table 2.1 and Table 2.2, respectively.

$+$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	12	0
2	3	4	5	6	7	8	9	10	11	12	0	1	2
3	4	5	6	7	8	9	10	11	12	0	1	2	3
4	5	6	7	8	9	10	11	12	0	1	2	3	4
5	6	7	8	9	10	11	12	0	1	2	3	4	5
6	7	8	9	10	11	12	0	1	2	3	4	5	6
7	8	9	10	11	12	0	1	2	3	4	5	6	7
8	9	10	11	12	0	1	2	3	4	5	6	7	8
9	10	11	12	0	1	2	3	4	5	6	7	8	9
10	11	12	0	1	2	3	4	5	6	7	8	9	10
11	12	0	1	2	3	4	5	6	7	8	9	10	11
12	0	1	2	3	4	5	6	7	8	9	10	11	12

TABLE 2.1: Addition in $GF(13)$

$\times$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	10	11	12
2	0	2	4	6	8	10	12	1	3	5	7	9	11
3	0	3	6	9	12	2	5	8	11	1	4	7	10
4	0	4	8	12	3	7	11	2	6	10	1	5	9
5	0	5	10	2	7	12	4	9	1	6	11	3	8
6	0	6	12	5	11	4	10	3	9	2	8	1	7
7	0	7	1	8	2	9	3	10	4	11	3	10	4
8	0	8	3	11	6	1	9	4	12	7	2	10	5
9	0	9	5	1	10	6	2	11	7	3	12	8	4
10	0	10	7	4	1	11	8	5	2	12	9	6	3
11	0	11	9	7	5	3	1	12	10	8	6	4	2
12	0	12	11	10	9	8	7	6	5	4	3	2	1

TABLE 2.2: multiplication in $GF(13)$

In Table 2.1 the elements which give 0 are additive inverses of each other and in Table 2.2 the elements which give 1 are multiplicative inverses of each other.

Definition 2.1.14. Trapdoor Function

Trapdoor function is a function that is easy to compute in one direction but difficult to compute in the reverse direction if some special information known as "Trapdoor" is not known.

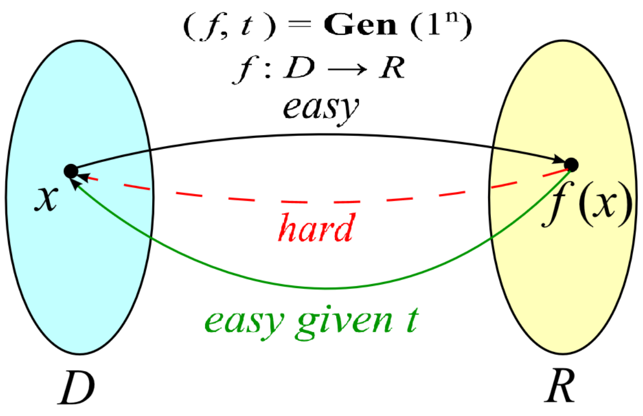


FIGURE 2.1: Trapdoor Funtion[12]

Which Functions are not "Trapdoor"?

Suppose we define a function $x + y = a$, in this function if given x and y then a can be computed easily but is it also easy to compute x and y when any of the other two are given. That is why it is not a trapdoor function as the definition says that there must be some special knowledge which must be taken into account while computing the function in the reverse direction.

2.2 Cryptographic Background

Cryptology is the study of cryptography and cryptanalysis [13]. Cryptography is used for secure communication. When two parties communicate with each other in the presence of an attacker, they try to make mechanism which allows them to have secure communication. Cryptography is basically the technique which

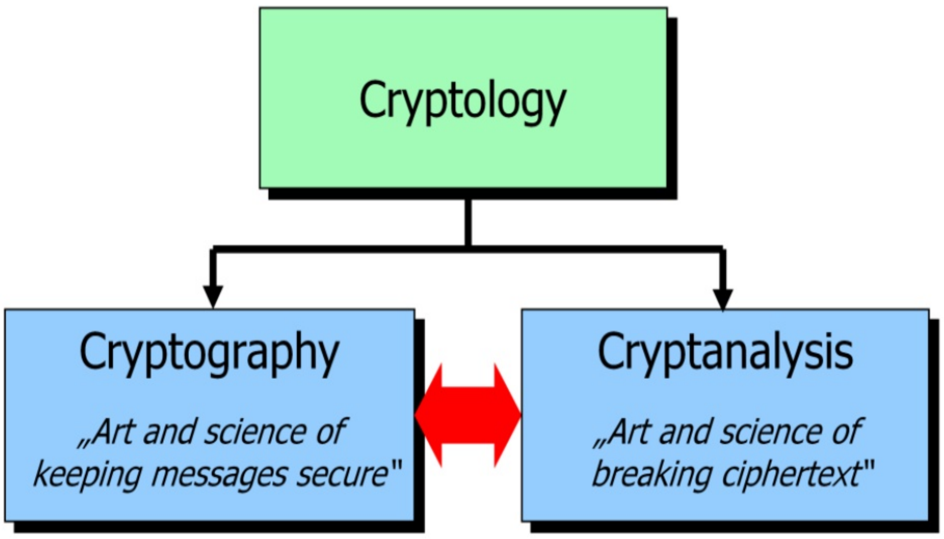

Cryptology
Cryptography
„Art and science of keeping messages secure"
Cryptanalysis
„Art and science of breaking ciphertext"

FIGURE 2.2: Cryptology

sets up the mechanism for them. Cryptography sets this mechanism by taking the original message into some coded message so that only the concerned reader is able to understand/read it. In cryptography, our soul purpose is to design a system that converts the original message into some unreadable/coded message. This system is known as cryptosystem. There are some cryptographic terms that are used for making a cryptosystem.

In any communication, the two parties who are engaged in conversation are usually known as **Alice** and **Bob**. In this conversation, the original message is called cleartext or **plaintext** denoted as P. There are two ways to encrypt the plaintext one is block cipher [14] and the other is stream cipher [15]. The **block cipher** takes one block of elements as an input and produces an output block for each input block. Whereas **stream cipher** takes input elements continously and produces output for one element at a time.

The plaintext is not directly sent to the receiver rather it is transformed into coded form which is not understandable. The coded form in cryptography is known as **ciphertext**. The plaintext is passed through some process via an algorithm to convert it into a ciphertext. This algorithm is called **encryption algorithm**. To make the ciphertext readable, again an algorithm is used to convert it into plaintext. This algorithm is known as **decryption algorithm**. To encrypt and deycrypt the message, both algorithms require a secret information/function which is known as **key**. Guarantee of a secure communication is highly dependent on the key. If a third person gets the knowledge of the key then the security of the conversation might be compromised.

Cryptography Basic Working

The basic working of any cryptographic technique is explained in the following algorithm:

Algorithm 2.2.1.
To make a system that enables Alice (sender) and Bob (receiver) to communicate securely. Encryption and decryption have to be taken place in the following fashion:

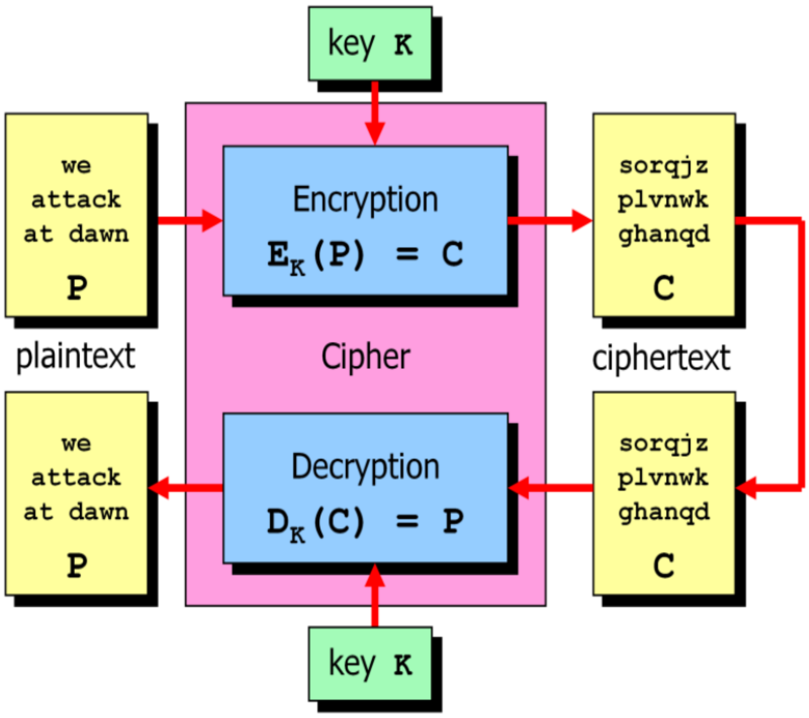


FIGURE 2.3: Block Diagram

Encryption

Suppose user Alice wants to send some confidential information to Bob. He has to go through from the following steps to encrypt the original message P.

Input: original message or plaintext P, encryption algorithm E, key K
Output: Ciphertext C

1. User Alice has plaintext P.
2. Using any encryption algorithm E and key K (must be secret), plaintext is encrypted as $E_K(P)$ and ciphertext C is generated.
3. Alice sends C to Bob through a public channel.

Decryption

Now Bob has to apply the following steps on C to make it readable.
Input: Ciphertext C, decryption algorithm D, key K
Output: original message or plaintext P

1. Bob has ciphertext C.
2. ciphertext is decrypted by applying decryption algorithm D and K on C as $D_K(P)$ and plaintext P is obtained.

This procedure can be further illustrated by using any encryption algorithm, for example, shift cipher. We are using shift cipher because it is simple and helpful to understand the basic working of cryptography. It works as follows:

Example 2.2.2.

Shift cipher is a technique of encrypting a plaintext P by applying a shift on each alphabet for example by a shift of 5. In this way each letter is replaced by the letter that is on the fifth place in the alphabetical order from current letter.
Encryption takes place in the following manner:

Encryption

Input: Plaintext P, encryption algorithm E i.e. shift cipher, key K(the shift size)
Output: Ciphertext C

1. User Alice has plaintext $P =$"shift cipher is a branch of symmetric crptography".
2. By applying a key of right shift of 5 on P.
3. Ciphertext $C =$ "xmnky xdkczm dn v wmvixc ja nthhzomdxxmtkojbmvkct"

Decryption

Now Bob has to apply the following steps on C to make it readable.
Input: Ciphertext C, decryption algorithm D, key K (the shift size)
Output: original message or plaintext P

1. Bob has ciphertext $C =$ "xmnky xdkczm dn v wmvixc ja nthhzomdxxmtkojbmvkct".
2. By applying a key of left shift of 5 on C.
3. Plaintext $P =$ "shift cipher is a branch of symmetric crptography".

2.2.1 Applications/Objectives of Cryptography

The objectives/applications of a secure communication consist of the following elements:

Confidentiality

Confidentiality of information or data refers to protection of the information from disclosure to third parties. In todays world, information has value especially in government documents, personal information, credit card numbers, trade secrets, bank account statements etc. Every one wants privacy of information that they wish to be kept secret. Confidentiality makes sure that no third party can understand the information unless they are authorized to do so.

Intergrity

Integrity guarantees the information being received in its original form. Integrity of information refers to protecting it from being modified by unauthorized parties. Information that has been changed or modified could prove to be costly. Information only has value if it is correct. For example, suppose a case where we are going to send Rs.50 but the information is changed or modified and shows that Rs.50,000 had actually been sent. Integrity is very useful in these kind of circumstances for making sure that information is not altered.

Authentication

Authentication helps us to authenticate a person whenever needed. The person who is claiming that the information is sent by me is actually is sent by him. It reveals the identity of the sender and helps the receiver to believe in the sender. Suppose that the participants Alice, Bob and John are engaged in exchange of information with one another. Now participant Alice sends a message to participants John. Later, Participants Bob claims that the message is sent by him. Here authentication plays its role to confirm who actually sent the message.

Non-Repudiation

Non-repudiation refers to a state where people cannot be deceived. Suppose participant Alice sends a message to participant Bob. Here, because non-repudiation participant Alice can never deny that the message is not sent by him.
Cryptographic schems are further classified into two categories.

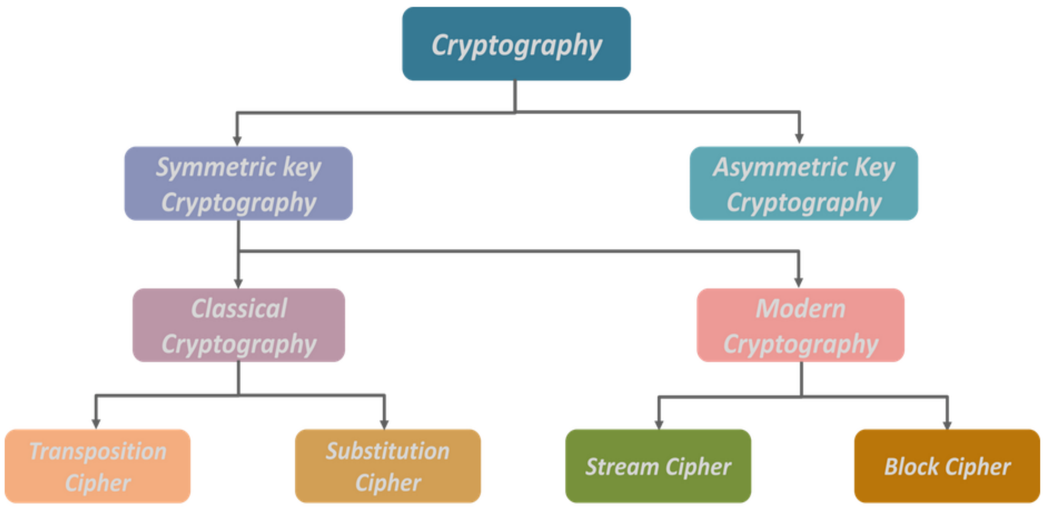


FIGURE 2.4: Cryptographic protocols [16]

Cryptographic Protocols There are two basic protocols in cryptography which are differentiated on the bases of keys used in them. They are given below:

A. Symmetric key cryptography

B. Asymmetric key cryptography

2.2.2 Symmetric Key Cryptography

Symmetric key cryptography is a method that ensures secrecy and security of the communication. An adversary who does not know the key, gets the encrypted message should not be able to understand the message. When two parties, say, Alice and Bob communicate over an insecure channel using some symmetric key protocol. Then the key used for encryption and decryption is same. There might be cases where decryption key can be obtained easily from the encryption key.

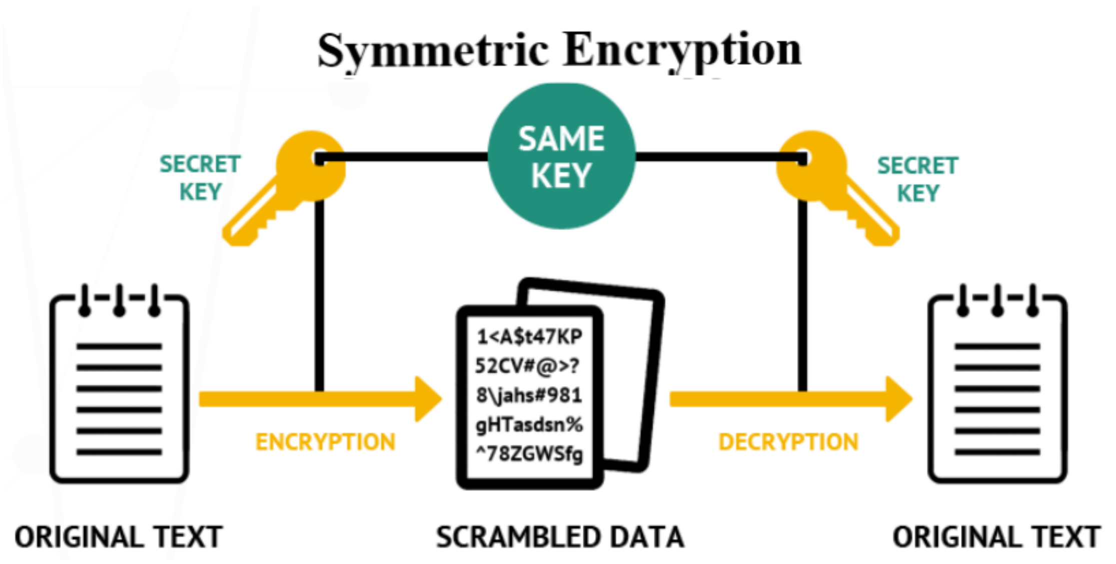


FIGURE 2.5: Symmetric key protocol [17]

In this protocol use of secure channel is required for the exchange of key. Because, Bob cannot be able to decrypt the message unless and until he has the key. So, Alice after encrypting the message sends the key through some secure channel which then enables him to decrypt the message. For example popular symmetric schemes includes AES [18], DES [19], RC4 [11] etc.

2.2.3 Asymmetric Key Cryptography

Asymmetric key cryptography was invented in 1976 by Whitfield Diffie and Martin Hellman. Symmetric key cryptography has a drawback of sharing a key and therefore asymmetric key cryptography was introduced. It is a method that is used for secure communication. It uses two keys; one for encryption and the other for decryption. The key used for encryption is known as **public key** of the owner(receiver) and decryption key is known as **private key** of the owner. It does not require a secure channel/way for the exchange of key. Everybody has access to public key and private key is only known to its owner(receiver).

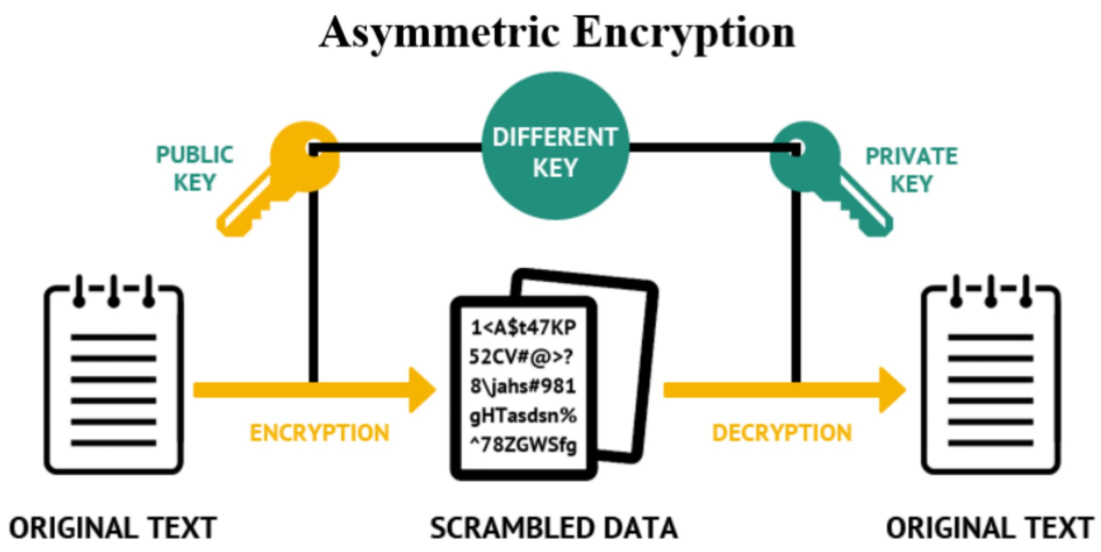


FIGURE 2.6: Asymmetric protocol[17]

Suppose Alice and Bob want to communicate over a insecure channel using asymmetric protocol. Then Alice first obtains the public key of Bob and use it to encrypt the message. The ciphertext is sent to Bob. Bob then uses his private key to decrypt the ciphertext into plaintext. Here we can observe that there is no need of secure channel for the exchange of key. Hence, it solves the issue of using a secure channel for key transfer. For example some famous asymmetric key protocols include DiffieHellman key exchange protocol, ElGamal [20], RSA [21] encryption algorithm , Elliptic curve [22] etc.

2.3 Cryptanalysis

Cryptanalysis [2] is a method that is used to break the cryptosystem in order to obtain the plaintext or some useful information like key. It is also studied to check that how strong and secure a cryptosystem is. The person who does cryptanlysis is called cryptanalyst. Cryptanalysis is always possible if a cryptosystem lacks any of the following properties:

1. Confidentialty

2. Integrity

3. Authentication

4. Non-Repudiation

There are many types of attacks [2], some of them are described below:

1. **Brute Force Attacks**

   In this attack [23], ciphertext and the decryption algorithm are known to an attacker. On the basis of which attacker attempts every possible key to obtain the plaintext. This attack requires a lot of time to accomplish the goal as the attacker has to look for every key available in the key space. If it is not feasible to attempt all possible keys in a reasonable time frame then this attack is not possible.

2. **Ciphertexts Only Attacks**

   In this attack [24], the attacker only knows ciphertexts. The corresponding plaintexts are usually not known. He uses this known ciphertexts to obtain the corresponding plaintexts.

3. **Chosen Ciphertext Attacks**

   In this attack [25], the attacker has access to decrypted plaintexts of some ciphertexts. On the basis of this known information he can try to obtain the key or the plaintexts of other ciphertexts.

4. **Chosen Plaintext Attacks**

   In this attack [24], the attacker knows the plaintext and the corresponding ciphertext through which he tries to guess the key or obtains as much information as possible.

5. **Known Plaintext Attacks**

   In this attack [26], for a given ciphertext there is some part of plaintext known to attacker which is further analyzed to get the complete plaintext or the decyption key.

6. **Man-in-the-middle Attacks**

   In this attack, when two parties try to agree on a key for secure communication. The attacker places himself between them in order to agree on a key without knowing them. Two keys are selected by attacker to deceive both parties. He uses one of the key to make first party agree on exchange of information by pretending to be the second party. The other key is used to deceive the second party. The two parties actually thinks that they are communicating with each other, but it is the attacker who is obtaining the information from both ends and hence attacks the communication.

2.4 Popular Asymmteric Techniques

Asymmetric key cryptography is based on the idea of one way trapdoor function. Now, we take a close into some of the most popular techniques of public key cryptography. We will see their working and highlight their strong and effective points which made them popular.

2.4.1 RSA

RSA was named on the names of its inventors i.e. Rivest, Shamir and Adleman. The idea of RSA was given in 1977. It falls in the category of public key cryptography as the algorithm uses two keys and also uses trapdoor function in its working. The technique guarantees the existence of confidentiality, integrity, non-repudiation and authentication in the communication between two parties. The scheme works on the principle of integer factorization problem. The complete method of RSA is mentioned below:
Suppose Alice and Bob wants to setup a secure system through RSA to communicate over an insecure channel. They must follow the following points:

Key Generation Phase:

1. Alice chooses two large primes p and q. For the confirmation of p, q being primes primality test is used.
2. The modulus n is computed as $n = p \times q$.
3. Then Euler totient function is computed by $\phi(n) = (p-1) \times (q-1)$.
4. With the help of Euler totient function public key k_1 and private key k_2 are generated.
5. Public key k_1 may be taken any number that lies between 1 and $\phi(n)$ but $\gcd(k_1, \phi(n)) = 1$.
6. Private key k_2 is the inverse of k_1 mod $\phi(n)$, which is computed using Algorithm 2.1.13.

Message Encryption/Decryption Phase:

1. A message m is split in blocks $m_1, m_2, ..., m_\ell$ where m_l is the last block in the message.
2. Each block is then encrypted as $c_i = m_i^{k_1} \mod n$ for $i = 1, 2, ...\ell$.
3. The receiver decrypts the ciphertext by $m_i = c_i^{k_2} \mod n$ for $i = 1, 2, ..., l$.

Example 2.4.1. This toy example shows how RSA is used to encrypt and decrypt a message. Note that this example is made for generating public and private keys of Alice. So here Bob will use Alice's public key to send him an encrypted message and then Alice will use his private key to get it back in its original form. Same will be done by Bob to generate his private and public keys and then Alice will use Bob's public key to continue the communication in a secure fashion.

Key Generation Phase:

1. Alice chooses two primes i.e., $p = 37$ and $q = 97$.
2. The modulus n is evaluated as $n = 37 \times 97 = 3589$.
3. Then Euler totient function is computed by $\phi(3589) = (37-1) \times (97-1) = 3456$.
4. Public key $k_1 = 31$ is chosen, as $\gcd(31, 3456) = 1$.
5. Using Algorithm 2.1.13, the inverse of k_1 in mod 3456 is found to be 223, which is the private key of the user. So, $k_2 = 223$.

Alice will make $(31, 3589)$ public so if Bob wants to communicate with him. He can do so. But $\phi(n)$ and k_2 cannot be made public as the complete security is dependent on it. The above procedure is also shown in the Figure 2.7.

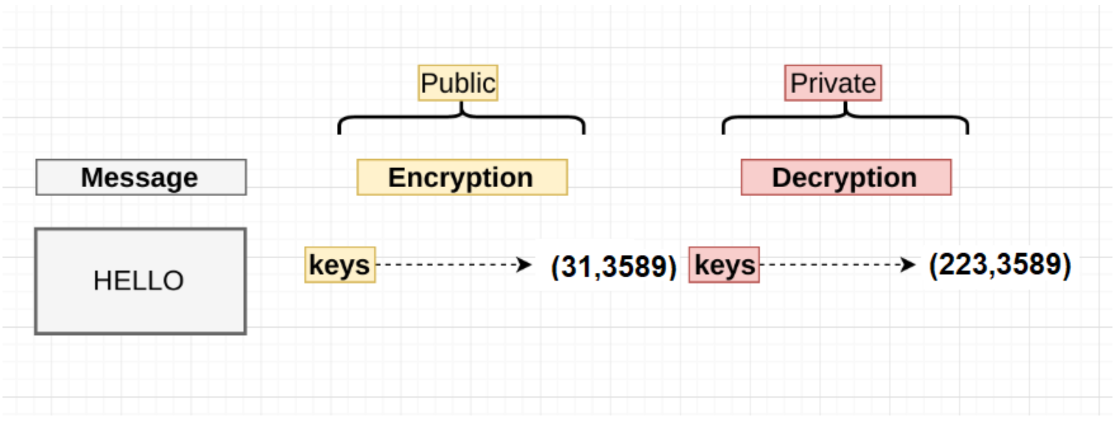


FIGURE 2.7: RSA key generation block diagram[27]

Message Encryption/Decryption Phase:

1. Suppose the original message m is "Hello"= 2, which Bob is going to send to Alice.
2. To obtian ciphetext c. Encryption is done as

$$c = 2^{31} \mod 3589$$
$$c = 1909 \mod 3589$$

3. Now Alice applies his private key k_2 on ciphertext c to decrypt it. By decrypting the ciphertext c, original message m is computed in the following way:

$$m = 1909^{223} \mod 3589$$
$$m = 2 \mod 3589$$

4. Alice gets the plaintext "Hello".
NOTE: All the above calculations are done by RSA calculator available online at https://www.cs.drexel.edu/ jpopyack/IntroCS/HW/RSAWorksheet.html.

The above procedure is also shown in the Figure 2.8.

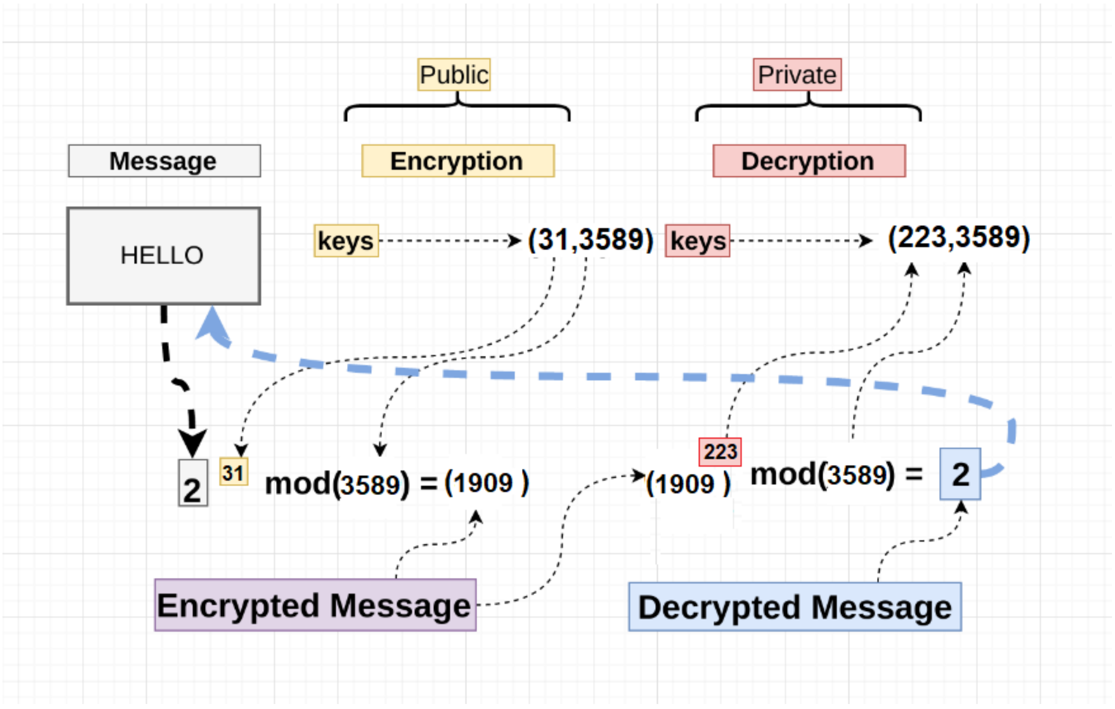


FIGURE 2.8: RSA encryption/decryption block diagram[27]

2.4.2 Discrete Logarithm Problem

Recall that in the start of this chapter we defined some mathematical terms which we need in this thesis. In that section, multiplicative cyclic groups were defined.

So, in order to define discrete logarithm problem [28] we need to take a help from cyclic group, which is

$$g^a = b$$

where $g \in G$ and G is a cyclic group and g is its generator. Since G is cyclic so $b \in G$. But for discrete logarithm problem, one must take a finite field i.e. $\mathbb{Z}_p^*$, where p must be prime. Then the discrete logarithm problem is, for a generator g of $\mathbb{Z}_p^*$, it is easy to compute

$$g^a = b \mod p$$

for $a \in \mathbb{Z}$. But knowing only g and b, it is hard to find a. This is known as discrete logarithm problem. Usually, a is the private key of the user. Discrete logarithm works on the principle of one-way trapdoor function. Remember that p must be a large prime number in order to protect a cryptosystem being attacked. For smaller p, sub-exponential algorithm might break the cryptosystem efficiently. So, p must be of 1024 bit large to avoid being attacked. Both key exchange protocol Diffie-Hellman and ElGamal encryption scheme are based on DLP.

2.5 Cryptography and Elliptic Curves

In the world of science and technology different ways of communication are being utilized. To communicate in a secure way is one of the fundamental objectives now a days. Because there are confidential informations that one will never want to be placed in the wrong hands. So to protect the information cryptographic tools are used so that it is ensured that security of the secret message is not compromised. A recent new approach of using public key cryptography is based on elliptic curve finite groups. The next chapter is devoted to elliptic curve cryptography.

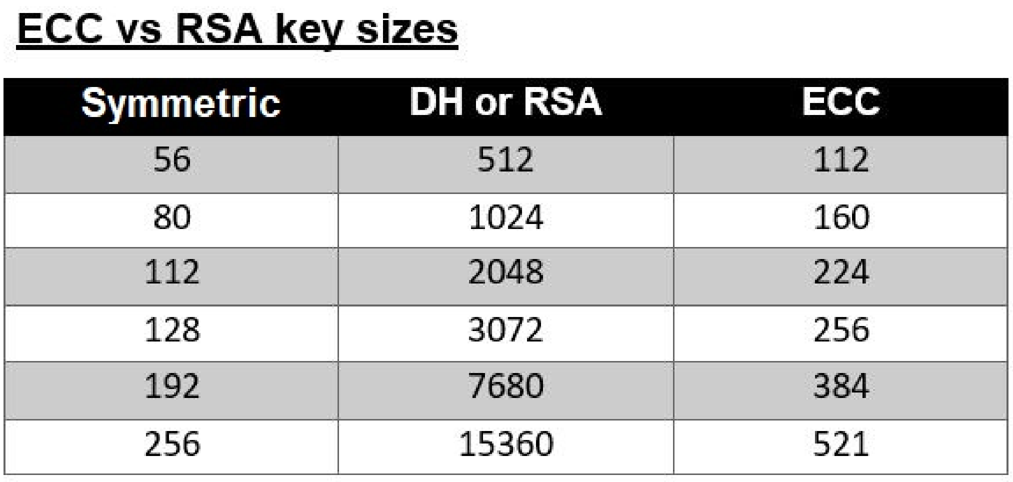
2.6 Summary

In this chapter an overview of mathematical background was presented which favored us to define the basic techniques of cryptography in an effective way. Most of the cryptosystems are based on the idea of finite fields. How computations and different operation like additon multiplication work in finite fields have also been discussed to define algorithms such as RSA, DLP etc. At the end of the chapter a strong connection between cryptography has been shown to set a platform for next chapter which will give us insights into elliptic curve cryptography.

Chapter 3

Elliptic Curve Cryptography

Initially the algorithms were based on integer factorization problem or Discrete Logarithm Problem (DLP). Some important algorithms include RSA and Diffie-Hellman key exchange protocol. RSA was proposed by R.Rivest, A.Shamir, and L.Adleman in 1976 and it is based on integer factorization problem. Whereas Diffie-Hellman is based on Discrete Logarithm problem proposed in 2002.
In integer factorization problem two large primes are chosen and then they are multiplied to obtain an integer. The algorithms based on integer factorization problem depend on the difficulty of the product being factorized. Discrete Logarithm Problem is defined using elements of cyclic group with modular arithmetic. Let g be a generator of multiplicative cyclic group $\mathbb{Z}_p$ where p is prime. We know that $g^a = b \in \mathbb{Z}_p$. Then discrete logarithm problem is to find "a" when only the knowledge of"g" and "b" is known.
The key size [] that National Institute of Standards and Technology (NIST) has suggested for DLP is 1024 bits meaning that one need to work with the field of at least 1024 bits to ensure secure communication. And because of a very large key size [29] the computations take a long time to be executed.



ECC vs RSA key sizes

Symmetric	DH or RSA	ECC
56	512	112
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

FIGURE 3.1: Nist recommended key sizes [30]

So, a need of improved approach was required in cryptography. Then the use of elliptic curve was introduced and observed that the discrete logarithm problem can be made more harder if it is defined over elliptic curve.The major advantage of the use of elliptic curves is that the same security level can be achieved by only working in a field of 160 bits and hence elliptic curve solves the problem of computational complexity to achieve the desired security extent. In the next section, we will see elliptic curve in detail and talk about how are they generated.

3.1 Elliptic Curve Cryptography

The equation of the form:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_5 \tag{3.1}$$

defined over a finite field $\mathbb{F}$ is a plane curve and known as Weierstrass equation [31]. Where a_1, a_2, a_3, a_4, a_5 and a_6 are called Weierstrass coefficients and they are selected from finite field $\mathbb{F}$. This curve is called a smooth curve if the discriminant

$$-b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6 \neq 0$$

where

$b_2 = a_1^2 + 4a_2$

$b_4 = 2a_4 + a_1a_3$

$b_6 = a_3^2 + 4a_6$

$b_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$

3.1.1 Elliptic Curves Over $\mathbb{R}$

In cryptography generally our focus is on simplified form of Weierstrass equation which is

$$y^2 = x^3 + Ax + B \tag{3.2}$$

Where A and B are Weierstrass coefficients and they are selected from a finite field $\mathbb{F}$. The discriminant for simplified form of (3.1) is $(4A^3 - 27B^2)$. This curve is said to be smooth if the discriminant $(4A^3 - 27B^2)$ is nonzero. The smooth Weierstrass curve is called elliptic curve. If the field $\mathbb{F} = \mathbb{R}$ the field of Now,let us consider a curve:

$$y^2 = x^3 - 6x + 4 \tag{3.3}$$

over the field of $\mathbb{R}$ the graphical representation is shown in Figure 3.2 is an elliptic curve because it does not have edges or self intersection and hence it is smooth Weierstrass curve. Smoothness of the curve can also be verified by its discriminant that is $4A^3 - 27B^2 \neq 0$, which can be seen as follows:

$$4A^3 - 27B^2 = -1296 \neq 0$$

Next, we see the group structure on elliptic curve. Suppose, we have two points P and Q which are on an elliptic curve E.

a) To add these points, the following steps must be followed:

1. A straight line is passed from points P and Q.
2. The straight line intersects the curve on some point say S of E.

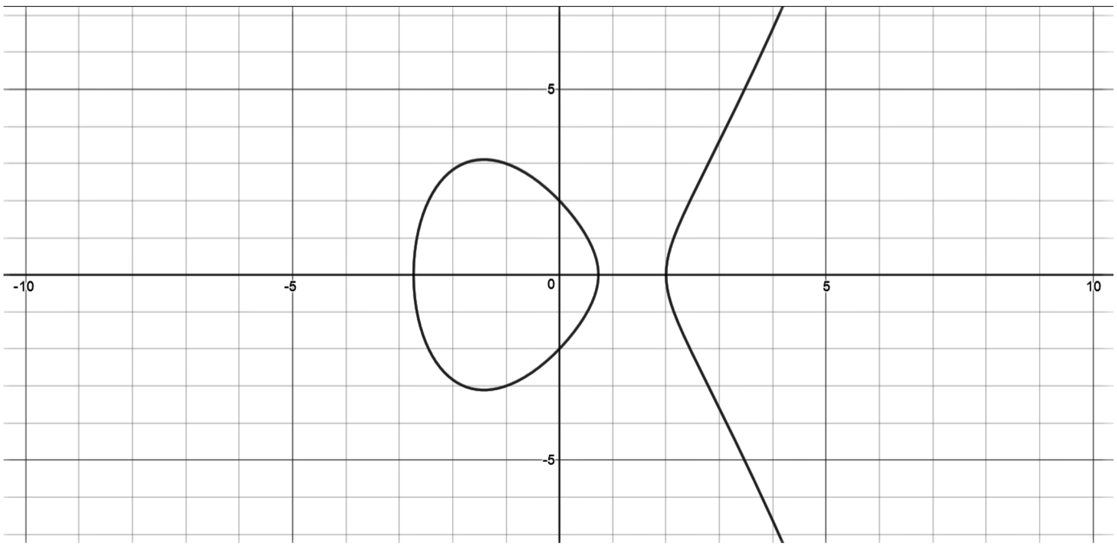


FIGURE 3.2: Graph of $y^2 = x^3 - 6x + 4$ over $\mathbb{R}$

3. Next to get the point R as the resultant of addition of P and Q, we just need to take the negative of S which is $-S = (x, -y)$.

This procedure can be best illustrated by showing points addition [32] graphically. Consider elliptic curve in (3.3) $E : y^2 = x^3 - 6x + 4$. The points $P = (0, 2)$ and $Q = (2, 0)$ are displayed by purple and orange color respectively lie on E. Using above mentioned procedure the resultant point R is shown by green color in Figure 3.3.

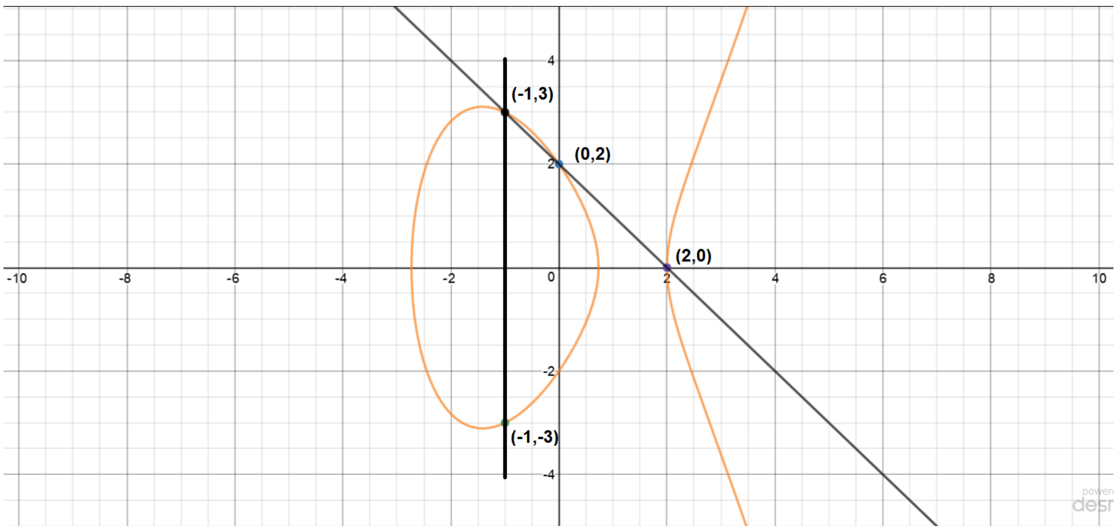


FIGURE 3.3: Graph of points addition $y^2 = x^3 - 6x + 4$ over $\mathbb{R}$

b) To add a point P to itself, following steps are applied.

1. Draw a tangent on P.
2. It intersects the curve on some point again considered as S of E.
3. Next to get the point $R = 2P$ as the resultant of addition of P to itself, taking negative of S is only required.

Next, we plot the graph for doubling a point means that we are adding a point to itself. The point $P = (0, 2)$ is displayed by purple color in Figure 3.4. This time we need to draw a tangent on the given point as mentioned earlier and the tangent then intersects the curve at point S shown by black color in Figure 3.4. The doubling of point is shown by green color which is just the reflection of point S.

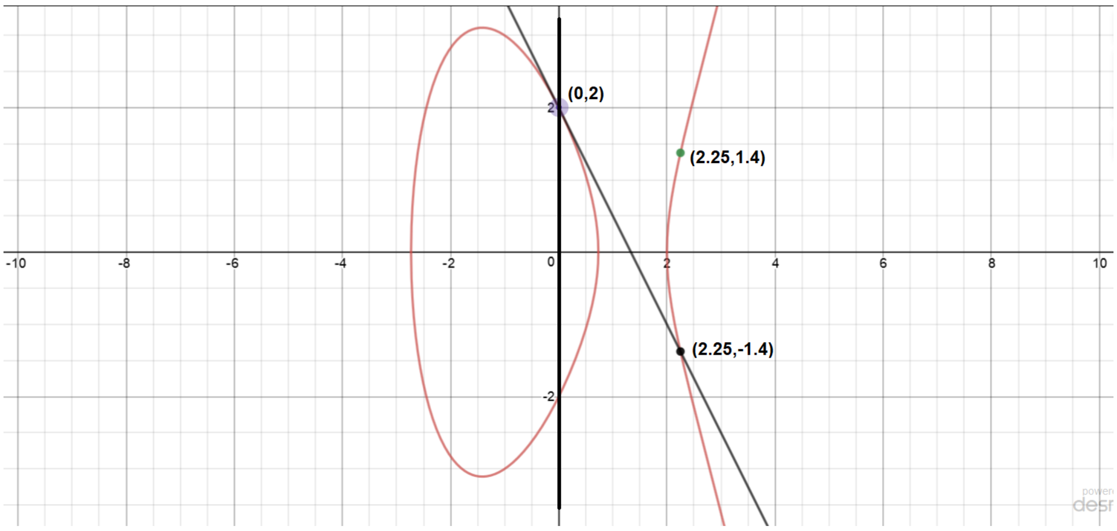


FIGURE 3.4: Graph of doubling a point on $y^2 = x^3 - 6x + 4$ over $\mathbb{R}$

c) The same procedure can be used for the addition of P to $-P$. We know that $-P$ is just the reflection of P. So, when we pass a straight line from them it reaches to infinity. We also know that when we add a point to its additive inverse we get the additive identity and thus in the definition of elliptic curve we define a special point which is located at infinity and recognized as point at infinity. The point at infinity is denoted as $\mathcal{O}$.

In Figure 3.5 points $P = (0, 2)$ and $-P = (0, -2)$ are displayed by orange and blue colors respectively. It is clear from the figure that if we pass a line from them it approaches to infinity and there we imagine a zero point or point at infinity $\mathcal{O}$ which serves as an additive identity in elliptic curve.

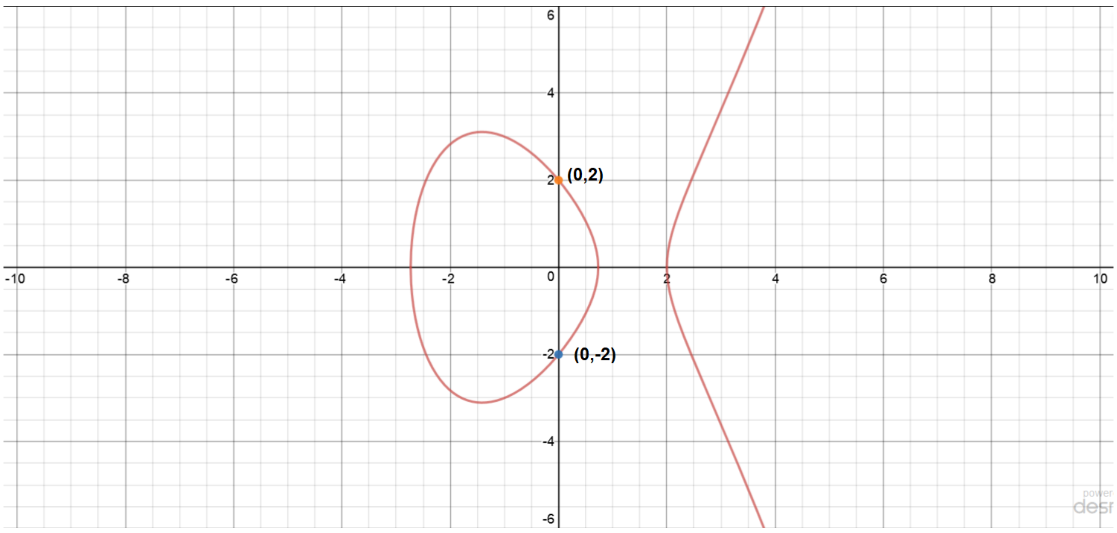


FIGURE 3.5: Elliptic curve point at infinity $\mathcal{O}$

This is now clear that how do the points on elliptic curve behave when they are added.
Keeping in mind the above graphical representation of point addition, we next see the mathematical representation of it.

3.1.1.1 Mathemtical Formulas for Point Addition on Elliptic Curve

In order to add a point $P(x_1, y_1)$ into $Q(x_2, y_2)$ on the elliptic curve in (3.2), i.e.

$$y^2 = x^3 + Ax + B$$

A line must be drawn through them as following the graphical structure of point addition. Let the the line ℓ passes through P and Q, the point slope form of ℓ is:

$$\ell : y = sx + c$$

We are only left to define the slope s. It takes the following cases:

Case 1: If P and Q are two different points, then

$$s = \frac{y_2 - y_1}{x_2 - x_1} \tag{3.4}$$

Case 2: If P and Q coincide, then

$$s = \frac{3x_1^2 + A}{2y_1} \tag{3.5}$$

the new point say $R(x_3, y_3)$ obtained by adding P and Q has following co-ordinates:

$$x_3 = s^2 - x_1 - x_2 \tag{3.6}$$

$$y_3 = s(x_1 - x_3) - y_1 \tag{3.7}$$

How the Coordinates of the New Point are Computed?

As we know that the line that connects P and Q is:

$$y = sx + c$$

using it in (3.2), we get

$$(sx + c)^2 = x^3 + Ax + B$$

$$\implies s^2x^2 + c^2 + 2csx = x^3 + Ax + B \tag{3.8}$$

As (3.2) is a cubic equation, it has three roots x_1, x_2 and x_3 can expressed in the following form:

$$(x - x_1)(x - x_2)(x - x_3) = 0$$

That is,

$$x^3 + x^2(-x_1 - x_2 - x_3) + x(x_1x_2 - x_2x_3 + x_1x_3) - x_1x_2x_3 = 0 \tag{3.9}$$

On comparing the coefficients of x^2 in 3.8 and 3.9, we have:

$$x_1 + x_2 + x_3 = s^2$$

So, the first coordinate of the resultant R is found to be

$$x_3 = s - x_1 - x_2$$

For the second coordinate, we have

$$y_3 = sx_3 + c$$

substituting $c = y_1 - sx_1$ in the above equation

$$y_3 = sx_3 + y_1 - sx_1$$

Taking negative of y_3 and with some simplification, the second coordinate is:

$$-y_3 = s(x_1 - x_3) - y_1$$

x_3 and $-y_3$ are the required coordinates of the new point.

Example 3.1.1. Let us again take (3.3) which is used to show addition of points graphically. It can be shown that the same points are obtained using the mathematical formulas given in the previous section. For elliptic curve:

$$y^2 = x^3 - 6x + 4$$

For points $P(0, 2)$ and $Q(2, 0)$, slope s is evaluated by (3.4):

$$s = \frac{0-2}{2-0}$$
$$s = -1$$

By adding P and Q, coordinates (x_3, y_3) of R are obtained using (3.6) and (3.7):

$$x_3 = (-1)^2 - 0 - 2$$
$$x_3 = -1$$
$$y_3 = (-1)(0 - (-1)) - 2$$
$$y_3 = -3$$

That is $P + Q = R = (-1, -3)$

Next we add P into itself. s is calculated by (3.5).

$$s = \frac{3(0)^2 - 6}{2(2)}$$
$$s = \frac{-6}{4}$$
$$s = -6(-4^{-1})$$

coordinates (x_4, y_4) of W is obtained using (3.6) and (3.7):

$$x_4 = (\frac{-6}{4})^2 - 2(0)$$
$$x_4 = \frac{36}{16}$$
$$x_4 = \frac{9}{4}$$
$$y_4 = (-1)(0 - (-1)) - 2$$
$$y_4 = -3$$

Thus P= Q= R= $(\frac{9}{4}, -3)$

So far we have only considered the field of real numbers. Now, we extend the idea to finite fields because our next aim is to introduce the use of elliptic curve in cryptography.

3.1.2 Elliptic Curve over Finite Fields

When we were dealing elliptic curve over real numbers, the graph shows a smooth curve. To define a curve over finite field, we have to use modular arithmetic. Now, the curve in Equation 3.2 takes the following form:

$$()y^2 = x^3 + Ax + B \mod p \tag{3.10}$$

which is defined over a finite field $\mathbb{F}_p$ and $p > 3$ is prime number. The coefficients A, B and variables x, y are from finite field $\mathbb{F}_p$. The ordered pair (x, y) which fulfills Equation (3.10) lie on the elliptic curve. If we talk the geometric behavior of elliptic curve over finite field then this time we do not see any smooth curve rather the discrete points appear on the graph.

To strengthen this claim we plot a graph and see its behavior. Let us consider the same curve over F_{13}.

$$y^2 = x^3 + 7x + 4 \mod 13 \tag{3.11}$$

The points that lie on the given curve are shown in Table 3.1. Elliptic curve points

x	y^2	$y_{1,2}$	$P(x,y)$	$P'(x,y)$
0	4	2,11	(0,2)	(0,11)
1	12	5,8	(1,5)	(1,8)
2	0	0	(2,0)	-
3	0	0	(3,0)	-
4	5	-	-	-
5	8	-	-	-
6	2	-	-	-
7	6	-	-	-
8	0	0	(8,0)	-
9	3	4,9	(9,4)	(9,9)
10	8	-	-	-
11	8	-	-	-
12	9	3,10	(12,3)	(12,10)

TABLE 3.1: Points of $E_{\mathbb{F}_{13}}(7, 4)$

addition for $E_{\mathbb{F}_{13}}(7, 4)$ is given in Table 3.2.

The graph of it is shown in Figure 3.6.

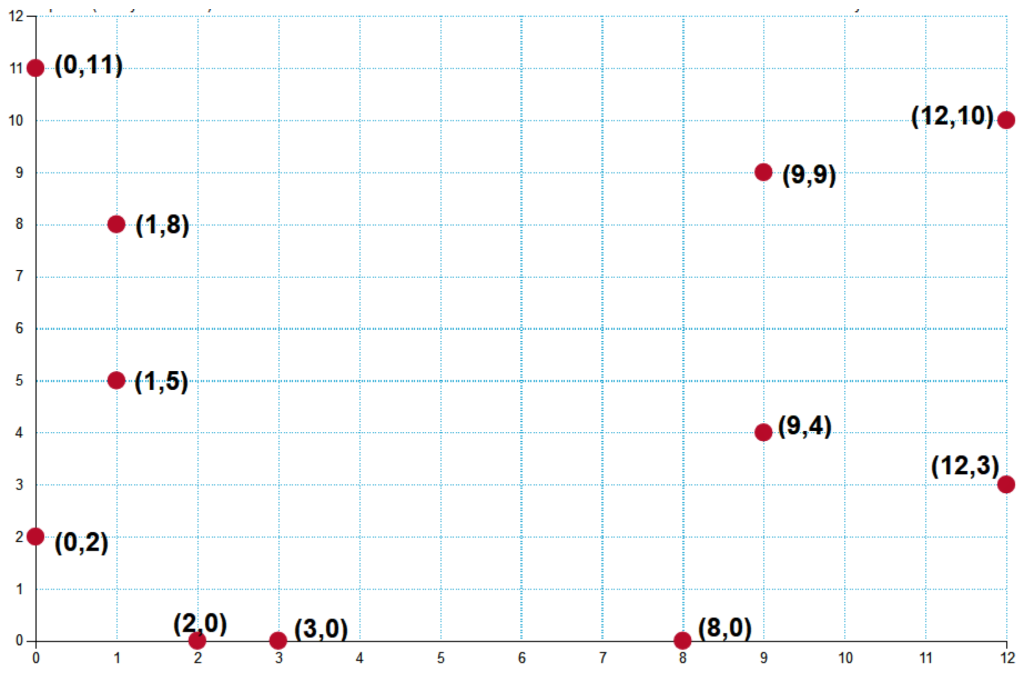


FIGURE 3.6: Elliptic curve $y^2 = x^3 + 7x + 4$ over $\mathbb{F}_{13}$

The same curve that displayed a smooth curve over real numbers is now showing discrete points which is only because of finite field $\mathbb{F}_{13}$. This is the change of behavior that we mentioned previously. The points in red are the ordered pairs (x, y) that satisfy the given equation (3.11).

Suppose we want to add points $P(9, 4)$ and $Q(12, 10)$ on the elliptic curve (3.11). To use the formula for point addition, one has to use the modular arithmetic. Then using the formulas in (3.6) and (3.7) give us the co-ordinates of new point $R(x_3, y_3)$. First we calculate the slope s by (3.4).

$$\begin{aligned} s &= \frac{10-4}{12-9} \mod 13 \\ s &= \frac{6}{3} \mod 13 \\ s &= 6(3^{-1}) \mod 13 \end{aligned}$$

By using extended Euclidean algorithm 2.1.13, we get

$$3^{-1} = 9 \mod 13$$

There

$$\begin{aligned} s &= (6)(9) \mod 13 \\ s &= 2 \mod 13 \end{aligned}$$

Now placing the value of s in (3.6) and (3.7), gives us:

$$\begin{aligned} x_3 &= (2)^2 - 9 - 12 \mod 13 \\ x_3 &= 9 \mod 13 \\ y_3 &= 2(9-9) - 4 \mod 13 \\ y_3 &= 9 \mod 13 \end{aligned}$$

Thus $R = (x_3, y_3) = (9, 9)$

So the addition of $P(9, 4)$ and $Q(12, 10)$ gives us $R(9, 9)$.

Now, let us add a point $P(9, 4)$ into itself. To compute $2P = P + P$, we again first find s by using formula (3.5) as:

$$\begin{aligned} s &= \frac{3(9)^2 + 7}{2(4)} \mod 13 \\ s &= 250 \times 8^{-1} \mod 13 \\ s &= 250 \times 5 \mod 13 \\ s &= 2 \end{aligned}$$

The co-ordinates of the new point are:

$$\begin{aligned} x_3 &= (2)^2 - 2(9) \mod 13 \\ x_3 &= 12 \\ y_3 &= 2(9-12) - 4 \mod 13 \\ y_3 &= 3 \end{aligned}$$

Which means adding P into itself gives us a point, $R(12, 3)$. Similarly, any point on the elliptic curve can be added to another point of the elliptic curve and also, a point can be added to itself as many times as we want. Table 3.2 shows the addition of all the points on (3.11).

$+$	∞	**(0,2)**	**(0,11)**	**(1,5)**	**(1,8)**	**(2,0)**	**(3,0)**	**(8,0)**	**(9,4)**	**(9,9)**	**(12,3)**	**(12,10)**
∞	∞	(0,2)	(0,11)	(1,5)	(1,8)	(2,0)	(3,0)	(8,0)	(9,4)	(9,9)	(12,3)	(12,10)
(0,2)	(0,2)	(12,3)	∞	(8,0)	(9,9)	(12,10)	(9,4)	(1,8)	(1,5)	(3,0)	(2,0)	(0,11)
(0,11)	(0,11)	∞	(12,10)	(9,4)	(8,0)	(12,3)	(9,9)	(1,5)	(3,0)	(1,8)	(0,2)	(2,0)
(1,5)	(1,5)	(8,0)	(9,4)	(12,10)	∞	(9,9)	(12,3)	(0,11)	(2,0)	(0,2)	(1,8)	(3,0)
(1,8)	(1,8)	(9,9)	(8,0)	∞	(12,3)	(9,4)	(12,10)	(0,2)	(0,11)	(2,0)	(3,0)	(1,5)
(2,0)	(2,0)	(12,10)	(12,3)	(9,9)	(9,4)	∞	(8,0)	(3,0)	(1,8)	(1,5)	(0,11)	(0,2)
(3,0)	(3,0)	(9,4)	(9,9)	(12,3)	(12,10)	(8,0)	∞	(2,0)	(0,2)	(0,11)	(1,5)	(1,8)
8,0	(8,0)	(1,8)	(1,5)	(0,11)	(0,2)	(3,0)	(2,0)	∞	(12,10)	(12,3)	(9,9)	(9,4)
(9,4)	(9,4)	(1,5)	(3,0)	(2,0)	(0,11)	(1,8)	(0,2)	(12,10)	(12,3)	∞	(8,0)	(9,9)
(9,9)	(9,9)	(3,0)	(1,8)	(0,2)	(2,0)	(1,5)	(0,11)	(12,3)	∞	(12,10)	(9,4)	(8,0)
(12,3)	(12,3)	(2,0)	(0,2)	(1,8)	(3,0)	(0,11)	(1,5)	(9,9)	(8,0)	(9,4)	(12,10)	∞
(12,10)	(12,10)	(0,11)	(2,0)	(3,0)	(1,5)	(0,2)	(1,8)	(9,4)	(9,9)	(8,0)	∞	(12,3)

TABLE 3.2: Addition of points of $E_{\mathbb{F}_{13}}(7,4)$

3.1.3 Elliptic Curve Discrete Logarithm Problem (ECDLP)

From the previous discussion we know that elliptic curves over finite fields form a cyclic group. It can be taken as a good motivation to define discrete logarithm problem over elliptic curves. As DLP [33] is based on cyclic group so for an elliptic defined over $\mathbb{F}_p$, we can obatin a point Q on the elliptic curve by adding P, n number of times, to itself. Mathematically it can be written as

$$\underbrace{P + P + P + ... + P}_{\text{n times}} = nP = Q$$

It is easy to find Q, knowing n and P but relatively difficult to obtain n when only P and Q are known. The difficulty of finding n by only having the knowledge of P and Q is known as ECDLP [22]. In DLP section, it was discussed that at least 1024 bits p is required to make a secure system but in case of ECDLP the same level of security can be achieved by just having a curve over p of 128 bits. This is why elliptic curves have been in a greatly investigated by the researcher and being used widely.

3.1.4 Elliptic Curve Diffie-Hellman Key Exchange Protocol

In order to communicate in a secure fashion Alice and Bob need to exchange their keys so that they can encrypt and decrypt the messages. In 1976 Whitefield Diffie and Martin Hellman [34] gave the idea of exchanging keys over a public network without compromising the security. The scheme is designed using a cyclic group of points of elliptic curve and security [35] relis on the difficulty of solving ECDLP. The following procedure tells all the story that ho do Alice and Bob exchange keys using Diffie-Hellman key exchange Protocol.

1. An elliptic curve E is mutually selected by Alice and Bob over a finite field $\mathbb{F}_q$together with G as base point of the group of n that is generated by E.
2. Alice selects a random number $K_A \in 1, 2, 3, ..., n-1$ as his secret key and computes $P_A = P_A G$ that again turns out to be a point of E.
3. Bob does the same with $K_A \in 1, 2, 3, ..., n-1$ as his secret key and computes $P_B = P_B G$.
4. Both P_A and P_B are exchanged with each other.
5. Alice then computes P_{AB}: $P_{AB} = K_A P_B$
6. Alice then computes P_{AB}: $P_{AB} = K_B P_A$

   P_{AB} is used as a session key security on finding K_A and K_B.

3.2 Summary

This chapter is all about use of elliptic curve in cryptography. In the start of the chapter, we discussed why previously existing techniques were not much efficient to run a secure cryptosystem. The reason was, those cryptosystems demands a user to work in bigger fields which eventually make the computations more complicated and increases the computational cost of the algorithm as well. But in elliptic curve cryptography same level of security can be achieved by working in smaller fields as proved secured by NIST. After defining all the basics of elliptic curve we moved towards the elliptic curve discrete logarithm problem which is currently the heart of many cryptosystems. One of the applications of ECDLP is Diffie-Hellman key exchange protocol that is also taken into consideration to show how ECDLP works.

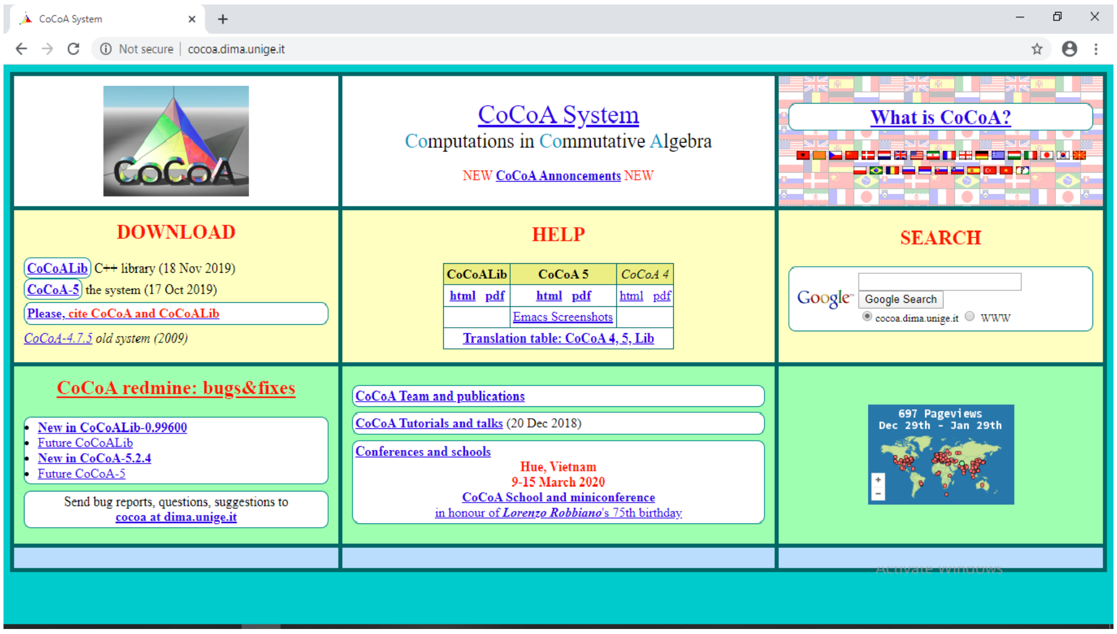
Chapter 4

The Package Elliptic Curve

In this chapter, we will introduce ApCoCoA, a software that is used to compute with algebra. In previous chapters we learned about different operations that can be performed on elliptic curve but once the field gets bigger manual or by hand calculations become more difficult. So, we need to have some application in computer that does all the things for us by just giving a command to computer. Currently, there are many tools available that deals with the elliptic curve such as MATLAB, MATHEMATICA etc. But, we are going to make a package in ApCoCoA for elliptic curve as it is highly efficient for algebraic computations.

ApCoCoA is a computer algebra system that enables the scientists and mathematicians to transform the manual computations of mathematical expressions in a program that a machine can understand. The term ApCoCoA stands for“Applied computations in commutative algebra” and it is originated in 1987 by University of Genova, Italy. It is a free software that is developed for computations related to numbers and polynomials. It can be easily accessed on modern operating systems such as windows, LINUX etc. It is useful for the researchers to execute an algorithm but it can also be used for simple calculations. The software is available on its official website http://cocoa.dima.unige.it. [36]

ApCoCoA is very useful for dealing with polynomials rings in more than one variables that are defined over rationals or modular integers. Its high-level language and easy access make it more suitable and reliable tool for working with algebraic computation. Its usage is simple and the application is easy to learn. The application is also being used for education purpose like teaching and learning purposes. It is highly active for computations with multivariate polynomials which are defined over rationals or integers. Implementations of most mathematical techniques is based on Grobner basis.



CoCoA System
Computations in Commutative Algebra
NEW CoCoA Annoncements NEW

What is CoCoA?

DOWNLOAD
CoCoALib C++ library (18 Nov 2019)
CoCoA-5 the system (17 Oct 2019)
Please, cite CoCoA and CoCoALib
CoCoA-4.7.5 old system (2009)

HELP

CoCoALib	CoCoA 5	*CoCoA 4*
html pdf	html pdf	html pdf
	Emacs Screenshots	
Translation table: CoCoA 4, 5, Lib		

SEARCH
Google Search
cocoa.dima.unige.it WWW

CoCoA redmine: bugs&fixes
- New in CoCoALib-0.99600
- Future CoCoALib
- New in CoCoA-5.2.4
- Future CoCoA-5

Send bug reports, questions, suggestions to cocoa at dima.unige.it

CoCoA Team and publications
CoCoA Tutorials and talks (20 Dec 2018)
Conferences and schools
Hue, Vietnam
9-15 March 2020
CoCoA School and miniconference
in honour of *Lorenzo Robbiano*'s 75th birthday

697 Pageviews
Dec 29th - Jan 29th

FIGURE 4.1: ApCoCoA basic features[36]

4.1 Mathematical Functionality

Basic arithmetic operations as well as some advanced operations like GCD, operations on polynomials such as composition and factorization naturally exists in the application.ApCoCoA is highly efficient for algebraic computation[37]. The software is designed to deal with very big integers as 2^{300000} which was difficult before. ApCoCoA has a tool that does not approximate the fractions in order to get the accurate answers. For instance, $1/9$ is approximately equal to 0.11111111 and 9×0.11111111 gives 0.99999999 but $(1/9) \times 9 = 1$. The software is well suited for performing different operations on polynomials such as multiplication, division, factorization etc. ApCoCoA can also be used to solve system of linear equations. The other features of the software include dealing with logical problems like truth tables, coloring different portions of a figure like maps and it can also calculate the area of a triangle using Heron's formula by using three sides of the triangle. The key tool for effective computations in commutative algebra is the concept of

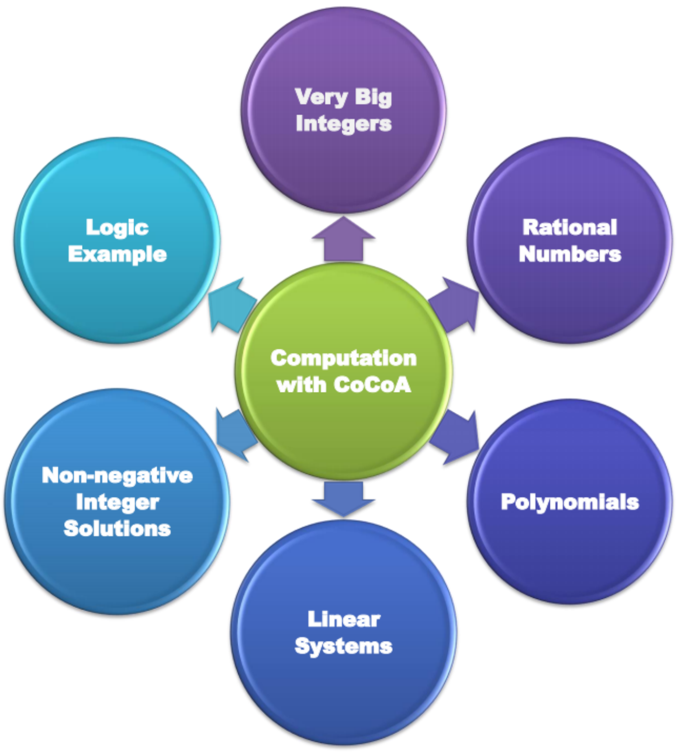


FIGURE 4.2: ApCoCoA features

Grobner basis. The default ring for commutative algebra has been set as $Q[x, y, z]$, but it can be changed by defining any other polynomial ring. The complete on-line help along with application can be obtained by http://cocoa.dima.unige.it [36]

4.2 How to use ApCoCoA?

For complete help guide, we refer the user to on-line help available at:

http://cocoa.dima.unige.it [36]

Here we give a brief overview that how a user can start the application in his system so that he may not face any problem in initializing the application.

4.2.1 Requirements

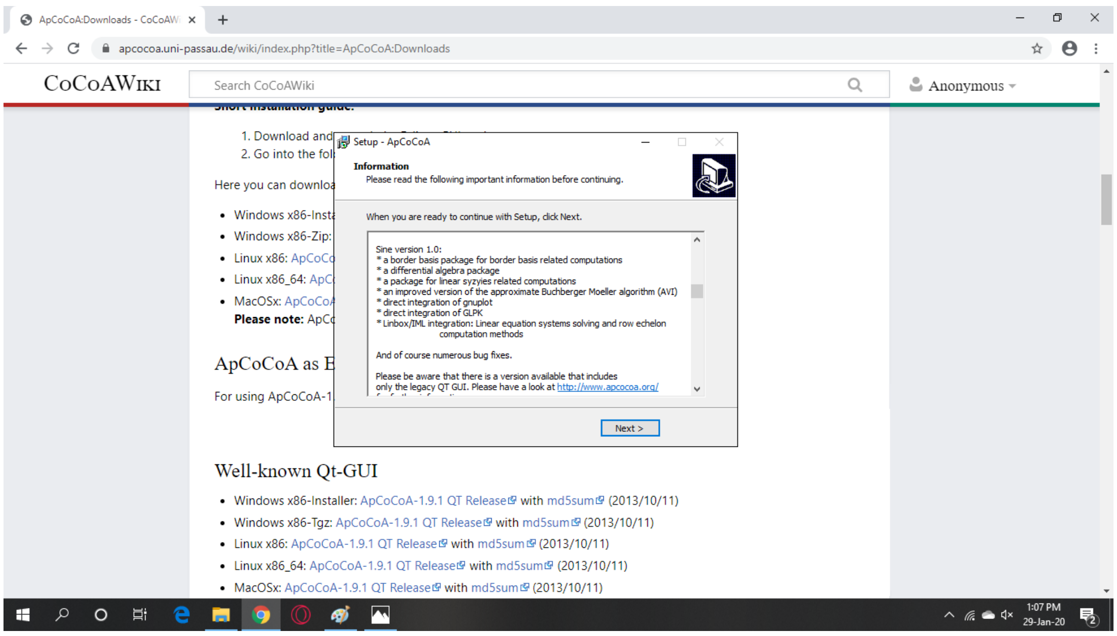
Here are some basic things that must be kept in mind before setting up the application in the system:

1. A user must have installed any operating system like windows 7,8,10, Linux, MAC in his machine.

2. Must have installed ApCoCoA in his machine.

4.2.2 Installation

1. Download the application's setup from https://apcocoa.uni-passau.de/ [38]. ApCoCoA's official page looks like this



ApCoCoA:Downloads - CoCoAWi
apcocoa.uni-passau.de/wiki/index.php?title=ApCoCoA:Downloads
CoCoAWiki
Search CoCoAWiki
Anonymous

1. Download and
2. Go into the fol

Here you can downloa

- Windows x86-Insta
- Windows x86-Zip:
- Linux x86: ApCoCo
- Linux x86_64: ApC
- MacOSx: ApCoCoA
  Please note: ApC

ApCoCoA as E

For using ApCoCoA-1

Setup - ApCoCoA

Information
Please read the following important information before continuing.

When you are ready to continue with Setup, click Next.

Sine version 1.0:
* a border basis package for border basis related computations
* a differential algebra package
* a package for linear syzyies related computations
* an improved version of the approximate Buchberger Moeller algorithm (AVI)
* direct integration of gnuplot
* direct integration of GLPK
* Linbox/IML integration: Linear equation systems solving and row echelon computation methods

And of course numerous bug fixes.

Please be aware that there is a version available that includes only the legacy QT GUI. Please have a look at http://www.apcocoa.org/

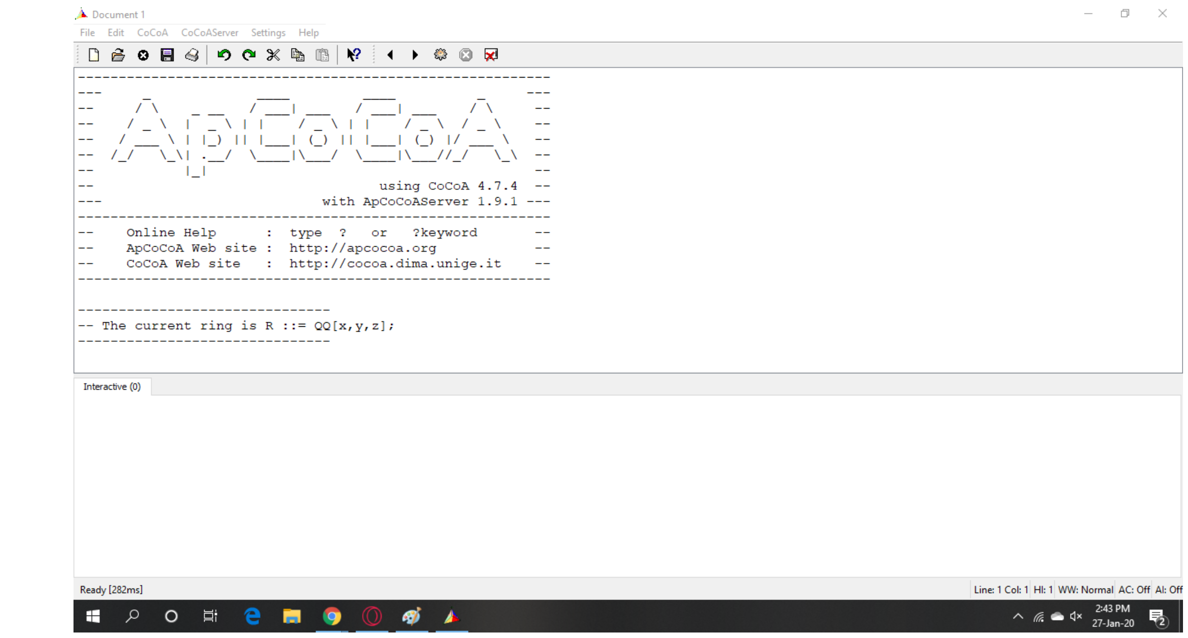
Well-known Qt-GUI

- Windows x86-Installer: ApCoCoA-1.9.1 QT Release with md5sum (2013/10/11)
- Windows x86-Tgz: ApCoCoA-1.9.1 QT Release with md5sum (2013/10/11)
- Linux x86: ApCoCoA-1.9.1 QT Release with md5sum (2013/10/11)
- Linux x86_64: ApCoCoA-1.9.1 QT Release with md5sum (2013/10/11)
- MacOSx: ApCoCoA-1.9.1 QT Release with md5sum (2013/10/11)

FIGURE 4.3: Official website of ApCoCoA

2. Run the setup from where it is saved.

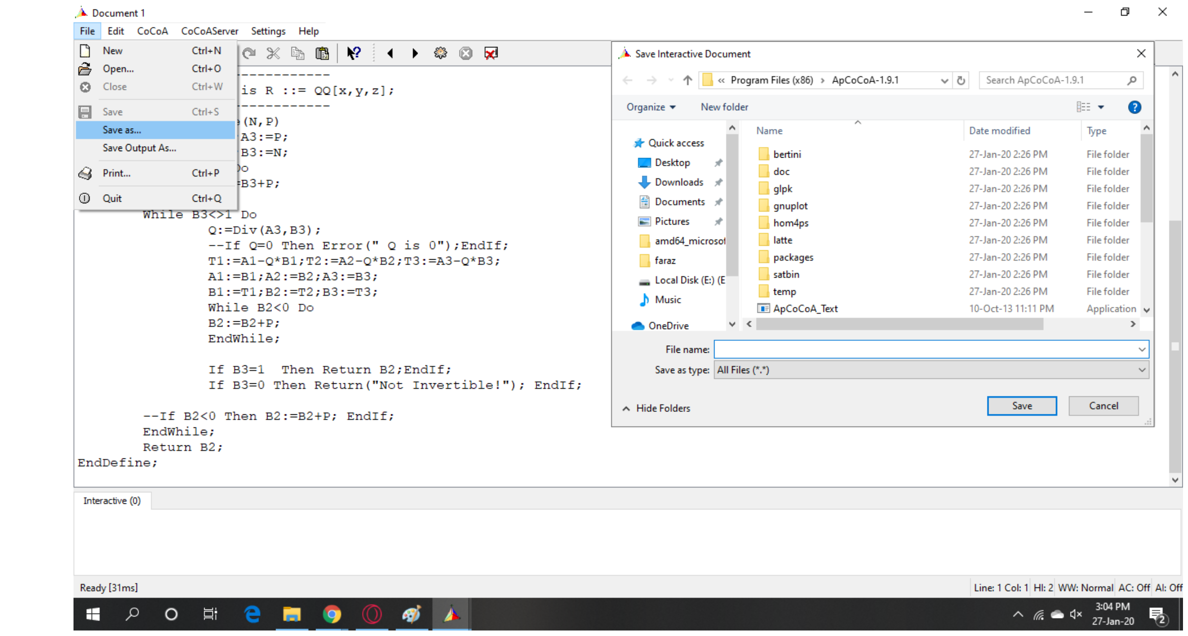
3. After successful installation of the application, following window will be appeared:


Document 1
File Edit CoCoA CoCoAServer Settings Help
Interactive (0)
Ready [282ms]
Line: 1 Col: 1 Hl: 1 WW: Normal AC: Off AI: Off
2:43 PM
27-Jan-20

```
-----------------------------------------------------------------
---            _               ____           ____              _        ---
--            / \    _ __     / ___| ___     / ___| ___      / \        --
--           / _ \  | '_ \  | |    / _ \  | |    / _ \    / _ \       --
--          / ___ \ | |_) | | |___| (_) | | |___| (_) |/ ___ \      --
--         /_/   \_\| .__/  \____|\___/  \____|\___//_/   \_\     --
--                    |_|                                              --
--                                                 using CoCoA 4.7.4  --
---                                      with ApCoCoAServer 1.9.1 ---
-----------------------------------------------------------------
--    Online Help       :  type  ?   or   ?keyword       --
--    ApCoCoA Web site :  http://apcocoa.org              --
--    CoCoA Web site   :  http://cocoa.dima.unige.it     --
-----------------------------------------------------------------


------------------------------
-- The current ring is R ::= QQ[x,y,z];
------------------------------
```

FIGURE 4.4: ApCoCoA Installation setup


Document 1
File Edit CoCoA CoCoAServer Settings Help
New Ctrl+N
Open... Ctrl+O
Close Ctrl+W
Save Ctrl+S
Save as...
Save Output As...
Print... Ctrl+P
Quit Ctrl+Q
Save Interactive Document
« Program Files (x86) › ApCoCoA-1.9.1
Search ApCoCoA-1.9.1
Organize
New folder
Quick access
Desktop
Downloads
Documents
Pictures
amd64_microsof
faraz
Local Disk (E:) (E
Music
OneDrive
Name
Date modified
Type
bertini 27-Jan-20 2:26 PM File folder
doc 27-Jan-20 2:26 PM File folder
glpk 27-Jan-20 2:26 PM File folder
gnuplot 27-Jan-20 2:26 PM File folder
hom4ps 27-Jan-20 2:26 PM File folder
latte 27-Jan-20 2:26 PM File folder
packages 27-Jan-20 2:26 PM File folder
satbin 27-Jan-20 2:26 PM File folder
temp 27-Jan-20 2:26 PM File folder
ApCoCoA_Text 10-Oct-13 11:11 PM Application
File name:
Save as type: All Files (*.*)
Hide Folders
Save
Cancel
Interactive (0)
Ready [31ms]
Line: 1 Col: 1 Hl: 2 WW: Normal AC: Off AI: Off
3:04 PM
27-Jan-20

```
                    -----------
                    is R ::= QQ[x,y,z];
                    -----------
                    (N,P)
                    A3:=P;
                    B3:=N;
                    o
                    B3+P;

        While B3<>1 Do
                Q:=Div(A3,B3);
                --If Q=0 Then Error(" Q is 0");EndIf;
                T1:=A1-Q*B1;T2:=A2-Q*B2;T3:=A3-Q*B3;
                A1:=B1;A2:=B2;A3:=B3;
                B1:=T1;B2:=T2;B3:=T3;
                While B2<0 Do
                B2:=B2+P;
                EndWhile;

                If B3=1  Then Return B2;EndIf;
                If B3=0 Then Return("Not Invertible!"); EndIf;

        --If B2<0 Then B2:=B2+P; EndIf;
        EndWhile;
        Return B2;
EndDefine;
```

FIGURE 4.5: ApCoCoA input and output window

It can be seen that the main window is split into two parts; upper is output window where the output is displayed and the lower is input window where the program is written.

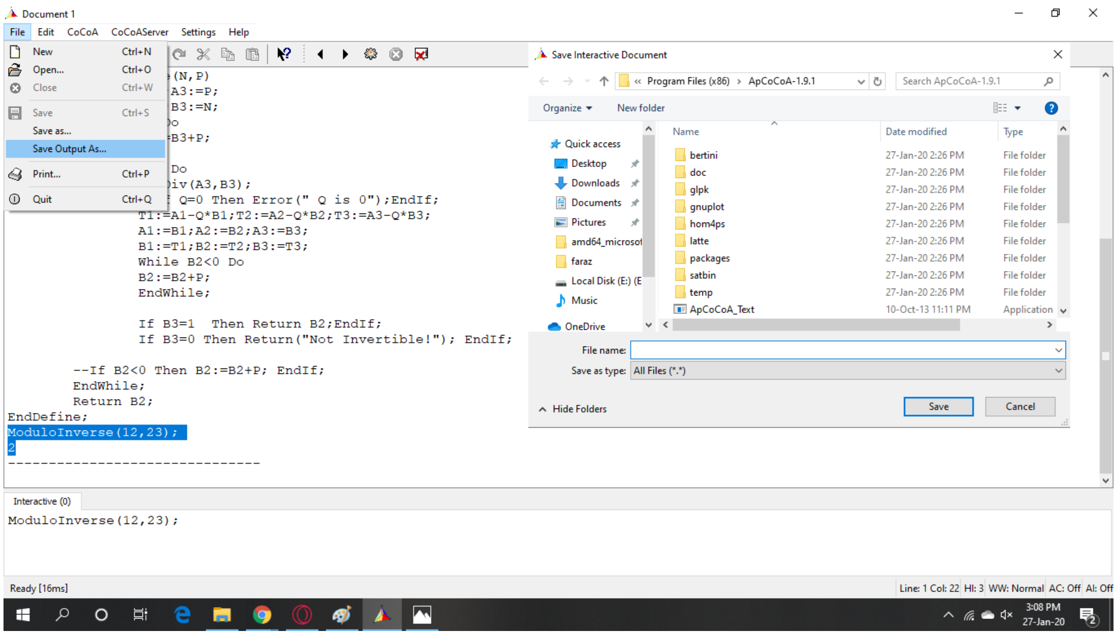
Note: User must have installed java editor to run ApCoCoA properly.

4.2.3 How to Save and Run a Programe in ApCoCoA

1. To **save a program** in ApCoCoA, user writes a program in the input window by setting up a name and all the inputs that are to be used in the program. Then the folloing steps are to be followed:

   - Go to file.
   - Click on "save as" to save the program.

   In Figure, we called the input function **ModuloInverse** written along with its inputs **N,P**. Save the program as displayed in Figure.



```
(N,P)
A3:=P;
B3:=N;
Do
B3+P;

Do
Div(A3,B3);
 Q=0 Then Error(" Q is 0");EndIf;
            T1:=A1-Q*B1;T2:=A2-Q*B2;T3:=A3-Q*B3;
            A1:=B1;A2:=B2;A3:=B3;
            B1:=T1;B2:=T2;B3:=T3;
            While B2<0 Do
            B2:=B2+P;
            EndWhile;

            If B3=1  Then Return B2;EndIf;
            If B3=0 Then Return("Not Invertible!"); EndIf;

        --If B2<0 Then B2:=B2+P; EndIf;
        EndWhile;
        Return B2;
EndDefine;
ModuloInverse(12,23);
2
-------------------------------
```

```
ModuloInverse(12,23);
```

FIGURE 4.6: How to save a program

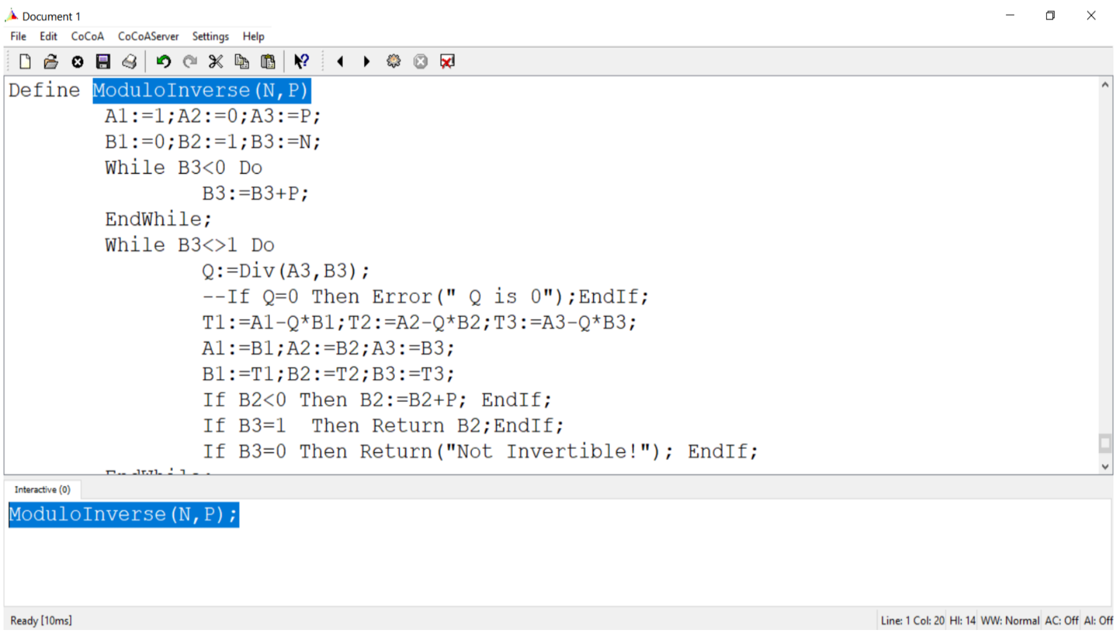
2. To **run the saved program**, user calls the function in the output window and then define the inputs for which the program is to be run. For instance if we are to find the multiplicative inverse of 12 in $\mathbb{Z}_{23}$, it is to be written like this **ModuloInverse{12,23}**; in the input window. To run the program user presses **ctrl+enter**. Moreover, the output displayed can also be saved.

4.3 Computations on Elliptic Curve Groups

The objective of this work is to develop a package in ApCoCoA for the execution of several programs such as performing operations on the points of elliptic curves. The package also provides full protocol to find multiplicative inverse of a number in a finite field. The package consists on the following programs:

4.3.1 Modulo Inverse

The idea of extended euclidean algorithm 2.1.13 is used to make a program to compute the inverse of a given number in a finite field. All the computations in this program are done in modulo arithmetic. This program is further used in other program in the package to find the inverse. The algorithm is implemented for computation of inverse no only in $\mathbb{F}_p$ but also in te Galois fields $\mathbb{F}_p{}^q$. The user just need to call the function in the input window for example in our case it is **ModuloInverse(N,P)**. The algorithm is implemented in the function **ModuloInverse(N,P)**; with inputs N and P. Here N is an element of $\mathbb{GF}p^q$ and P is the prime numbers.
Output: The inverse of N.
In ApCoCoA, to find the inverse of a number, a code is prepared for Algorithm 2.1.13. Figure 4.7 shows the screen of the code that a user see on his system.

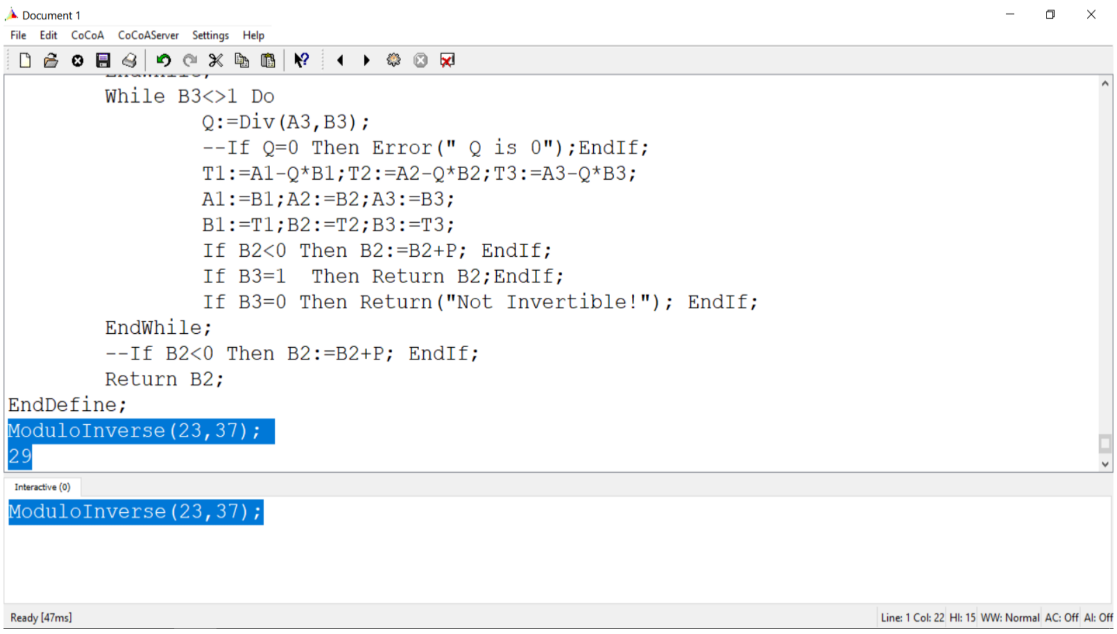

Document 1
File Edit CoCoA CoCoAServer Settings Help
Interactive (0)
Ready [10ms]
Line: 1 Col: 20 Hl: 14 WW: Normal AC: Off AI: Off

```
Define ModuloInverse(N,P)
        A1:=1;A2:=0;A3:=P;
        B1:=0;B2:=1;B3:=N;
        While B3<0 Do
                B3:=B3+P;
        EndWhile;
        While B3<>1 Do
                Q:=Div(A3,B3);
                --If Q=0 Then Error(" Q is 0");EndIf;
                T1:=A1-Q*B1;T2:=A2-Q*B2;T3:=A3-Q*B3;
                A1:=B1;A2:=B2;A3:=B3;
                B1:=T1;B2:=T2;B3:=T3;
                If B2<0 Then B2:=B2+P; EndIf;
                If B3=1  Then Return B2;EndIf;
                If B3=0 Then Return("Not Invertible!"); EndIf;
```

```
ModuloInverse(N,P);
```

FIGURE 4.7: Use of ApCoCoA for finding inverse in finite field

Suppose, if a user wants to find the inverse of 23 under modulo 37. He will call the function **ModuloInverse(N,P)**; in input window and replace N by 23 and P by 37 i.e. **ModuloInverse(23,37)**; and executing the function will display the answer 29 in the output window as shown in Figure 4.8.


Document 1
File Edit CoCoA CoCoAServer Settings Help
Interactive (0)
Ready [47ms]
Line: 1 Col: 22 Hl: 15 WW: Normal AC: Off AI: Off

```
        While B3<>1 Do
                Q:=Div(A3,B3);
                --If Q=0 Then Error(" Q is 0");EndIf;
                T1:=A1-Q*B1;T2:=A2-Q*B2;T3:=A3-Q*B3;
                A1:=B1;A2:=B2;A3:=B3;
                B1:=T1;B2:=T2;B3:=T3;
                If B2<0 Then B2:=B2+P; EndIf;
                If B3=1  Then Return B2;EndIf;
                If B3=0 Then Return("Not Invertible!"); EndIf;
        EndWhile;
        --If B2<0 Then B2:=B2+P; EndIf;
        Return B2;
EndDefine;
ModuloInverse(23,37);
29
```

```
ModuloInverse(23,37);
```

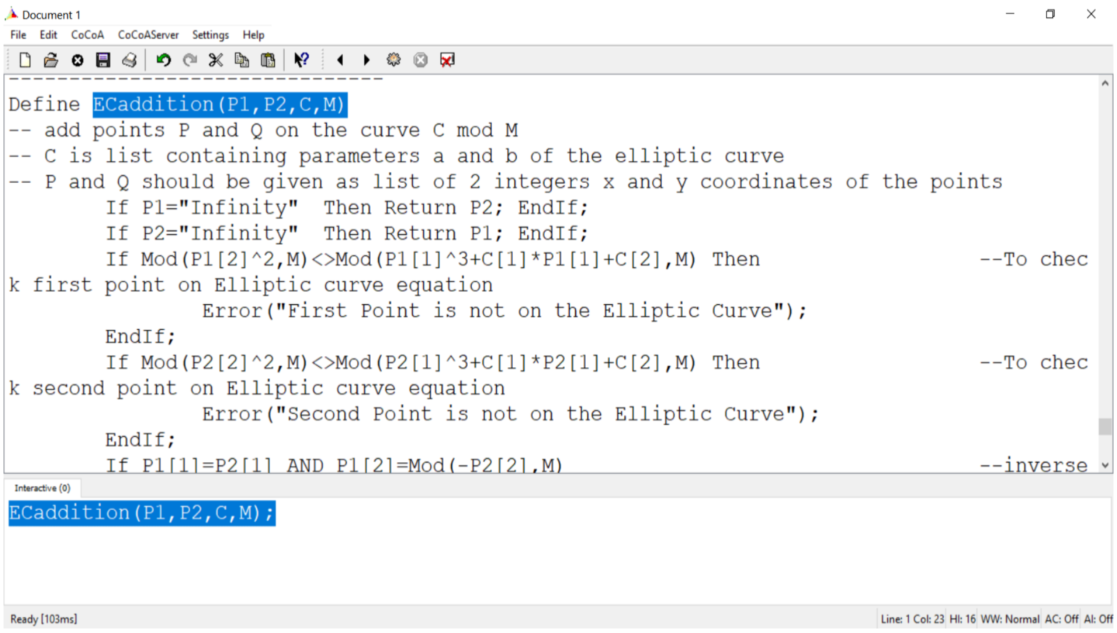
FIGURE 4.8: Toy Example for user

4.3.2 EC Addition

The package contains a program for the addition of points on elliptic curve. Again the basic idea, that is given in Chapter 3, is transformed in ApCoCoA to enable the system perform the operations on elliptic curve. It can also be used to add same points. To add two different elliptic curve points using ApCoCoA. Following steps are required:

a) The user will first run the complete package.

b) In input window, he calls the function **ECaddition{P1,P2,C,M}**; as shown in Figure 4.9.

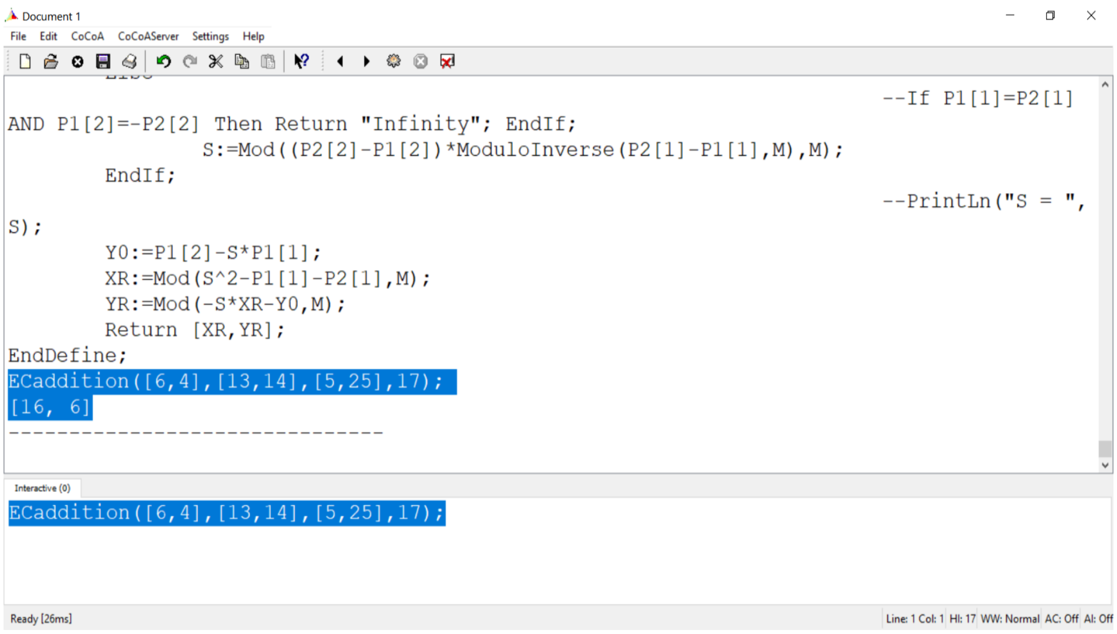

Document 1
File Edit CoCoA CoCoAServer Settings Help
Interactive (0)
Ready [103ms]
Line: 1 Col: 23 Hl: 16 WW: Normal AC: Off AI: Off

```
---------------------------------
Define ECaddition(P1,P2,C,M)
-- add points P and Q on the curve C mod M
-- C is list containing parameters a and b of the elliptic curve
-- P and Q should be given as list of 2 integers x and y coordinates of the points
        If P1="Infinity"  Then Return P2; EndIf;
        If P2="Infinity"  Then Return P1; EndIf;
        If Mod(P1[2]^2,M)<>Mod(P1[1]^3+C[1]*P1[1]+C[2],M) Then                     --To chec
k first point on Elliptic curve equation
                Error("First Point is not on the Elliptic Curve");
        EndIf;
        If Mod(P2[2]^2,M)<>Mod(P2[1]^3+C[1]*P2[1]+C[2],M) Then                     --To chec
k second point on Elliptic curve equation
                Error("Second Point is not on the Elliptic Curve");
        EndIf;
        If P1[1]=P2[1] AND P1[2]=Mod(-P2[2],M)                                     --inverse
```

```
ECaddition(P1,P2,C,M);
```

FIGURE 4.9: Use of ApCoCoA for the addition of points on elliptic curve

c) The user now types the values of all the inputs $P1, P2, C, M$, where M is a prime number and the values of $P1, P2$ and C are to be given in the form of a list because $P1$ and $P2$ are the points of elliptic curve in the form of ordered pairs and C contains the coefficients a and b of elliptic curve. In particular, user writes it in the following fashion **ECaddition{[6,4],[13,14],[5,25],17**; } and the output is $[16, 6]$ as shown in Figure 4.10.

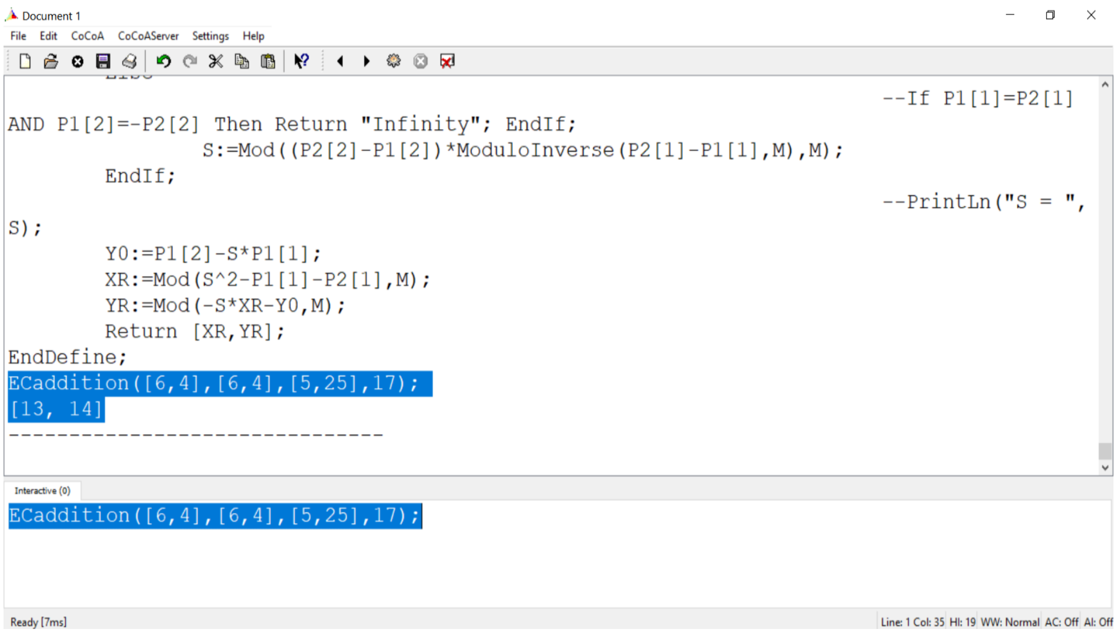


```
                                                    --If P1[1]=P2[1]
AND P1[2]=-P2[2] Then Return "Infinity"; EndIf;
                S:=Mod((P2[2]-P1[2])*ModuloInverse(P2[1]-P1[1],M),M);
        EndIf;
                                                    --PrintLn("S = ",
S);
        Y0:=P1[2]-S*P1[1];
        XR:=Mod(S^2-P1[1]-P2[1],M);
        YR:=Mod(-S*XR-Y0,M);
        Return [XR,YR];
EndDefine;
ECaddition([6,4],[13,14],[5,25],17);
[16, 6]
-------------------------------

ECaddition([6,4],[13,14],[5,25],17);
```

FIGURE 4.10: Toy Example for user

To add a point into itself, the user will again call the function in the input window as **ECaddition{P1,P2,C,M;}**. The user now writes the same point at $P1$ and $P2$. For example, if we wish to add $[6,4]$ into itself the the user writes **ECaddition{[6,4],[6,4],[5,25],17;}**, the output of it is $[13,14]$ which can be seen in Figure 4.11.



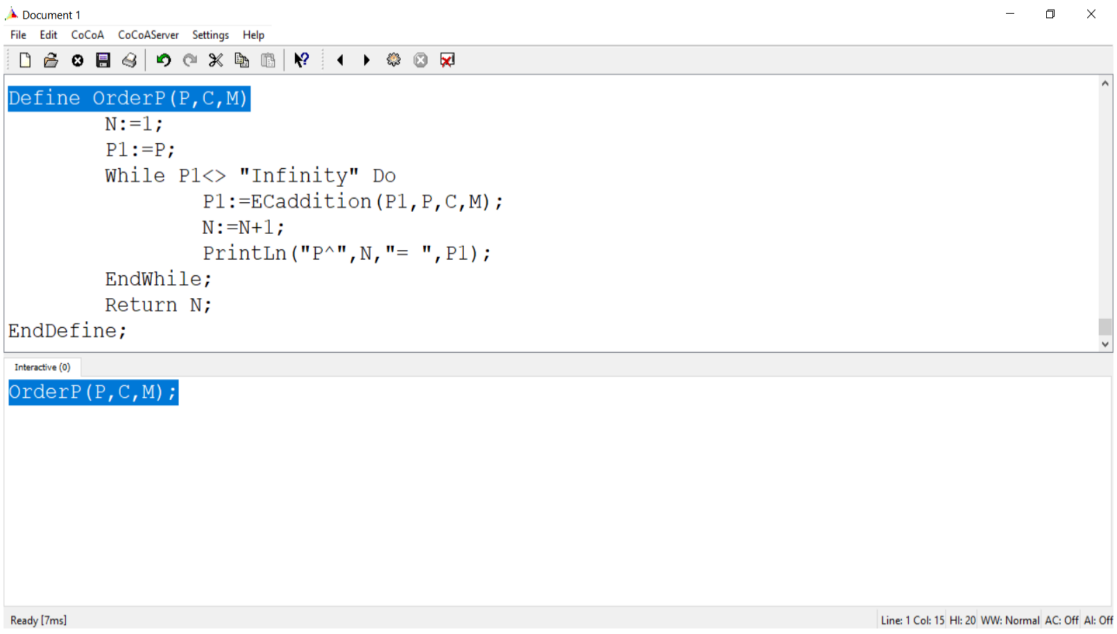
```
                                                    --If P1[1]=P2[1]
AND P1[2]=-P2[2] Then Return "Infinity"; EndIf;
                S:=Mod((P2[2]-P1[2])*ModuloInverse(P2[1]-P1[1],M),M);
        EndIf;
                                                    --PrintLn("S = ",
S);
        Y0:=P1[2]-S*P1[1];
        XR:=Mod(S^2-P1[1]-P2[1],M);
        YR:=Mod(-S*XR-Y0,M);
        Return [XR,YR];
EndDefine;
ECaddition([6,4],[6,4],[5,25],17);
[13, 14]
-------------------------------

ECaddition([6,4],[6,4],[5,25],17);
```

FIGURE 4.11: ApCoCoA's example to add a point into itself

4.3.3 Order of the Point of Elliptic Curve

In cryptography it is very important to construct a cyclic group to define ECDLP on it as it is the base for many currently available algorithms in the world of security and cryptography. This program evaluates the order of a point on a given elliptic curve very efficiently. To find the order [39] of a point of an elliptic curve. The user will call the function **OrderP(P,C,M);** from the —- package as in Figure 4.12.

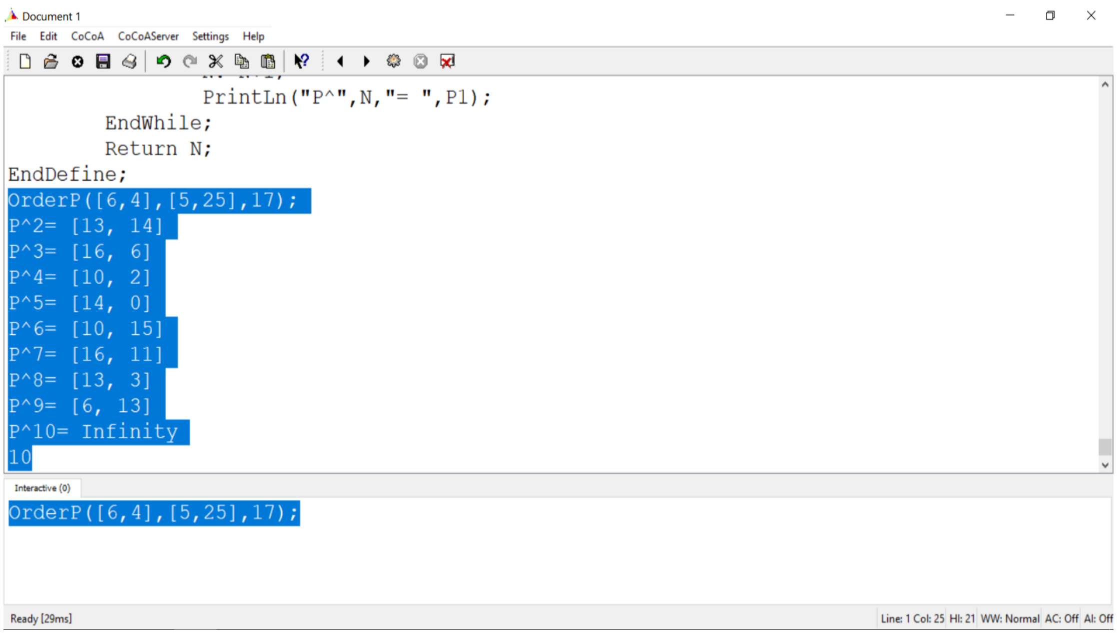

Document 1
File Edit CoCoA CoCoAServer Settings Help
Interactive (0)
Ready [7ms]
Line: 1 Col: 15 Hl: 20 WW: Normal AC: Off AI: Off

```
Define OrderP(P,C,M)
        N:=1;
        P1:=P;
        While P1<> "Infinity" Do
                P1:=ECaddition(P1,P,C,M);
                N:=N+1;
                PrintLn("P^",N,"= ",P1);
        EndWhile;
        Return N;
EndDefine;
```

```
OrderP(P,C,M);
```

FIGURE 4.12: Use of ApCoCoA for finding the order of a point on elliptic curve

To find the order of $(6, 4)$ on elliptic curve $y^2 = ax^3 + 5x = 25 \mod 17$, the user types **OrderP([6,4],[5,25],17);** and the order of $(6, 4)$ along with all the points that can be obtained by adding $(6, 4)$ into itself multiple times will be displayed in the output window. This task is illustrated in Figure 4.13.


Document 1
File Edit CoCoA CoCoAServer Settings Help
Interactive (0)
Ready [29ms]
Line: 1 Col: 25 Hl: 21 WW: Normal AC: Off AI: Off

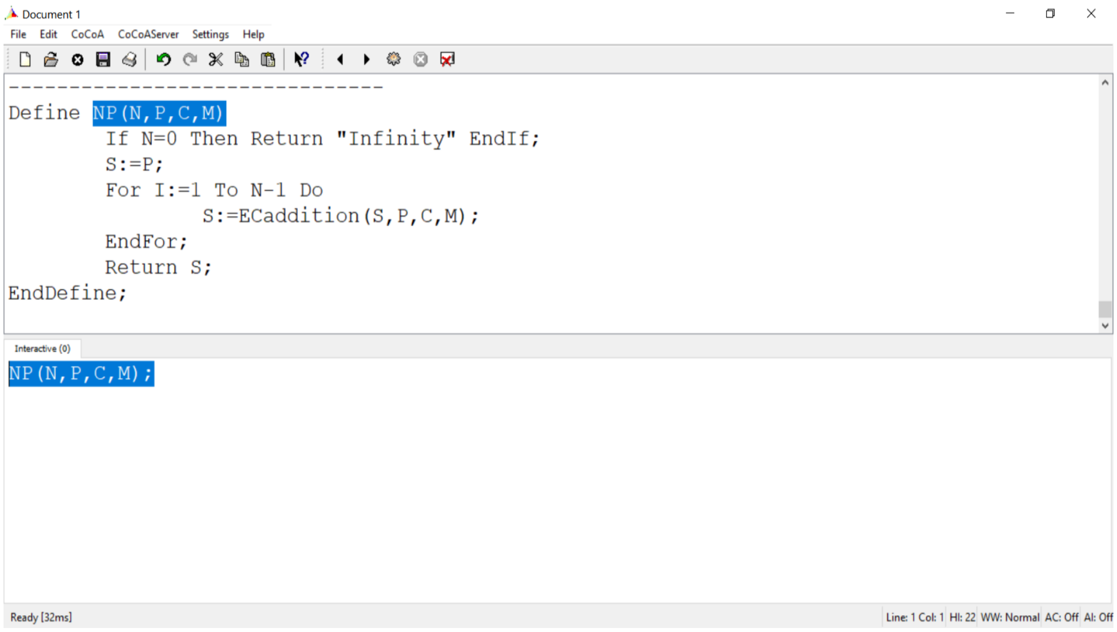
```
                PrintLn("P^",N,"= ",P1);
        EndWhile;
        Return N;
EndDefine;
OrderP([6,4],[5,25],17);
P^2= [13, 14]
P^3= [16, 6]
P^4= [10, 2]
P^5= [14, 0]
P^6= [10, 15]
P^7= [16, 11]
P^8= [13, 3]
P^9= [6, 13]
P^10= Infinity
10
```

```
OrderP([6,4],[5,25],17);
```

FIGURE 4.13: Toy Example for user

4.3.4 Adding a Point to itself Multiple Times

This package is well suited to add points multiple time or to a fixed number of times say n. User just needs to give the command to the system and it can give the resultant point in no time after adding it to a desired number of times. If user wants to add an elliptic curve point to a desired number of times, he can do so using the code that we are going to introduce next. Figure — shows that apoint can be added N times. Here, user just needs to call **NP{N,P,C,M}**; in the input window as illustrated in Figure 4.14.

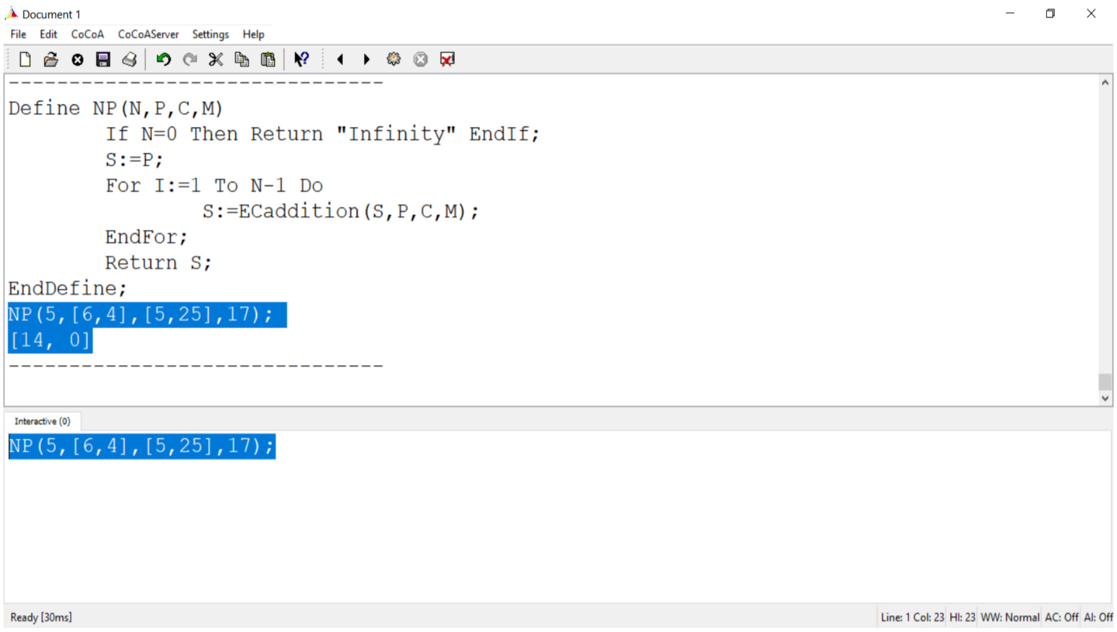

Document 1
File Edit CoCoA CoCoAServer Settings Help
Interactive (0)
Ready [32ms]
Line: 1 Col: 1 Hl: 22 WW: Normal AC: Off AI: Off

```
-------------------------------
Define NP(N,P,C,M)
        If N=0 Then Return "Infinity" EndIf;
        S:=P;
        For I:=1 To N-1 Do
                S:=ECaddition(S,P,C,M);
        EndFor;
        Return S;
EndDefine;
```

```
NP(N,P,C,M);
```

FIGURE 4.14: Use of ApCoCoA to add a point N times

Now, the user will tell the function **NP{N,P,C,M}**; that how many times he wants to add point P. For example, he wants to add $P(6,4)$ 5 times. So he will replace N by 5, P and C are in the form of list as $[6,4]$ and $[5,25]$, respectively. We have used a field of $\mathbb{F}_{17}$ so modulo used here is $M = 17$. Therefore, the function becomes **NP{5,[6,4],[5,25],17}**; and by executing it gives $[14,0]$ which means by adding $(6,4)$ 5 times we get $(14,0)$ as it can be seen in Figure 4.15.


Document 1
File Edit CoCoA CoCoAServer Settings Help
Interactive (0)
Ready [30ms]
Line: 1 Col: 23 Hl: 23 WW: Normal AC: Off AI: Off

```
-------------------------------
Define NP(N,P,C,M)
        If N=0 Then Return "Infinity" EndIf;
        S:=P;
        For I:=1 To N-1 Do
                S:=ECaddition(S,P,C,M);
        EndFor;
        Return S;
EndDefine;
NP(5,[6,4],[5,25],17);
[14, 0]
-------------------------------
```

```
NP(5,[6,4],[5,25],17);
```

FIGURE 4.15: Toy Example for user

In the next section, we will mention different commands along with their description and example.

4.4 CoCoA Command References

4.4.1 Inverse

Modulo inverse for integers.

Syntax

ModuloInverse($N : INT, P : INT) : INT$

where N is an integer: whose inverse to be found in the finite field $\mathbb{F}_P$

Description

This function returns the inverse of N in modulo P.

Example

ModuloInverse(5,7);

3

- -

4.4.2 Point addition

Addition for elliptic curve points.

Syntax

ECadd(P1:LIST,P2:LIST,C:LIST,M:INT):LIST
where $P1$, $P2$ and C are in the form of list: new point to be found in the finite field $\mathbb{F}_m$

Description

This function returns the point (in the form of a list) which is the sum of $P1$ and $P2$ in modulo m. By taking $P1$ and $P2$ this function adds the point into itself.

Example

ECaddition([16,6],[10,2],[5,25],17);

$[16, 11]$

- -

ECaddition([16,6],[16,6],[5,25],17);

$[10, 15]$

- -

4.4.3 Adding Point to itself N Number of Time

Addition of elliptic curve point to a desired number of time.

Syntax

NP(N:INT,P:LIST,C:LIST,M:INT):INT
where N is an integer, P and C are in the form of list.

Description

This function returns the point (in the form of list) by adding P to N number of time in modulo M.

Example

NP(3,[10,2],[5,25],17);
$[13, 14]$
- -

4.4.4 Point Order

Order of an elliptic curve point.

Syntax

OrderP(P:LIST,C:LIST,M:INT):INT
where P and C are in the form of list.

Description

This function returns the order of P in modulo M along with all te points which van be obtained by adding P to itself multiple times. Adding P to itself means $P + P = P^2$

Example

OrderP([10,2],[5,25],17);
$P^2 = [13, 3]$
$P^3 = [13, 14]$
$P^4 = [10, 15]$
$P^5 = Infinity$
5
- -

4.5 Implementation of ECDH Key Exchange Protocol using ApCoCoA Package

When Diffie Hellman key exchange protocol is defined over elliptic curve, it is known as ECDH key exchange protocol. For this purpose both parties agree up on elliptic curve domain parameters to share a secret. An overview of ECDH is given below.

4.5.1 Key Agreemenet Algorithm Example

Both parties agree on the following domain parameters to initialize the scheme.

1. Let us consider an elliptic curve $y^2 = x^3 + 30x + 24$ over a finite field $\mathbb{F}_199$.
2. Point $(2, 93)$ is taken as the generator G.

Step 1: Both users A and B choose their private keys as 150 and 123 respectively.

Step 2: Both users A and B compute their public keys Q_A and Q_B receptively as

$$Q_A = 150 \times (2, 93) = (79, 91)$$

$$Q_B = 123 \times (2, 93) = (12, 108)$$

Note: These points are computed using the package made in ApCoCoA.

Step 3 Both users exchange their public keys

Step 4 User A computes $150 \times (12, 108) = (196, 92)$ (using ApCoCoA package)

Step 5 User B computes $123 \times (79, 91) = (196, 92)$ (Using ApCoCoA Package) The shared secret is same and successfully delivered to both the users.

4.6 Message Encryption/Dycryption using the Scheme of Omar Reyad

In the scheme proposed by Omer Reyad [40], the following steps have been mentioned to encrypt/decrypt a message:

1. Suppose Alice and Bob are interested participants for the secure communication.
2. First, they agree on an elliptic curve over $\mathbb{F}_p$ along with a generator G.
3. Then Alice chooses a random number x as her private key and Bob chooses a random y as his private key.
4. They also compute their public keys as follows:
   Alice's public key $P_A = x \times G$
   Bob's public key $P_B = y \times G$

5. Now, if Alice wants to send an encrypted message to Bob. She works in the following fashion:

   a. Alice first transforms the message in the points of elliptic curve using ASCII code [41] and the mapping defined by Omer Rayed in [40].

   b. Alice chooses another random number k and uses Bob's public key to encrypt the message P_m.

   c. She then sends the pair of points P_C to Bob as an encrypted text.

$$P_C = [(([k] \times G)), (P_m + [k] \times P_B)]$$

6. In order to decrypt the ciphertext P_C, the following equation is used by BoB:

$$P_m = [(P_m + [k] \times P_B) - ([yk] \times G)]$$

4.6.1 Toy Example

The following examples shows how the scheme works.
Note that all the elliptic curve operations are performed using the package made in ApCoCoA.

1. Suppose Alice and Bob are interested participants for the secure communication.

2. They agree on an elliptic curve:

$$y = x^3 + 4x + 1$$

   defined over $\mathbb{F}_{503}$, with generator $G = (283, 315)$

3. Then Alice chooses a random number 25 as her private key and Bob chooses a random 101 as his private key.

4. They compute their public keys as follows:

   Alice's public key $P_A = 25 \times (283, 315) = (354, 146)$

   Bob's public key $P_B = 101 \times (283, 315) = (363, 109)$

5. Alice wants to send the word "ATTACK" as an original message to Bob. She works in the following fashion:

   a. Using ASCII code [41] and mapping defined by Omer Rayed in [40], she obtains the following corresponding elliptic curve points for the word "ATTACK":

   $$A = (407, 201)$$

   $$T = (308, 154)$$

   $$T = (308, 154)$$

   $$A = (407, 201)$$

   $$C = (486, 282)$$

   $$K = (477, 182)$$

   b. Alice chooses another random number 50 and uses Bob's public key $P_B = (363, 109)$ to encrypt the message

   $$(407, 201)(308, 154)(308, 154)(407, 201)(486, 282)(477, 182)$$

   .

   c. She then sends the pairs of points $P_{C_1}, P_{C_2}, ..., P_{C_6}$ to Bob as an encrypted text.

   $$P_{C_1} = [(50 \times (283, 315)), ((407, 201) + 50 \times (363, 109))] = (308, 154)$$

   $$P_{C_2} = [(50 \times (283, 315)), ((308, 154) + 50 \times (363, 109))] = (354, 357)$$

   $$P_{C_3} = [(50 \times (283, 315)), ((308, 154) + 50 \times (363, 109))] = (354, 357)$$

   $$P_{C_4} = [(50 \times (283, 315)), ((407, 201) + 50 \times (363, 109))] = (308, 154)$$

$$P_{C_5} = [(50 \times (283, 315)), ((486, 282) + 50 \times (363, 109))] = (306, 429)$$

$$P_{C_6} = [(50 \times (283, 315)), ((477, 182) + 50 \times (363, 109))] = (45, 303)$$

6. In order to decrypt the ciphertext P_{C_i}, the following equation are permomed

$$P_{m_1} = [(308, 154) - (422, 331)] = (407, 201)$$

$$P_{m_2} = [(354, 357) - (422, 331)] = (308, 154)$$

$$P_{m_3} = [(354, 357) - (422, 331)] = (308, 154)$$

$$P_{m_4} = [(308, 154) - (422, 331)] = (407, 201)$$

$$P_{m_5} = [(306, 429) - (422, 331)] = (486, 282)$$

$$P_{m_6} = [(45, 303) - (422, 331)] = (477, 182)$$

7. Now, seeing these pairs of points in ASCII code [41], the original text "ATTACK" is obtained.

Chapter 5

Conclusion

In this thesis we developed a package in ApCoCoA to perform different operations on elliptic curve and in modulo arithmetic such as finding the inverse of a number using extended euclidean algorithm.
In order to know about some basics terminologies or basic idea are always important that is why we first looked into the basics of mathematics like groups, rings and fields along with basics of cryptography to make a strong connection with the things like DLP, Elliptic curves and ECDLP. Moving on, we discussed asymmetric cryptographic techniques like RSA, Diffie-Hellman key exchange protocols.
To run any scheme of cryptography efficiently we are bound to rely on a machine that does all the computations. Previously, there are number of software available to perform the computation in modulo arithmetic as well as elliptic curve but in order to do that user must have access of internet since they do not work offline.Also, online available softwares are not trustworthy. Since the security cannot be compromised as the leakage of information to an unauthorized person can be disastrous. That is why using online available softwares can be risky and not entirely reliable source.
So, taking the idea of these online platforms we made a package in ApCoCoA that works equally and efficiently to perform the desired operations. ApCoCoA is reliable tool as the user does not need to share his personal information or secret data to a third party for execution of the algorithm. Its completely in the control of the

user and hence it is reliable source. Our package is simple and user friendly. Since ApCoCoA is freely available software and dose not have huge requirements to run at any system. Following are the points that elaborate what can be computed in this package.

1. Finding inverse of a number in modulo arithmetic.
2. Different point addition on elliptic curve.
3. Same point addition on elliptic curve.
4. Order of a point of elliptic curve.
5. n times addition of a point.
6. Number of points lying on elliptic curve

The above mentioned points are the major concerns with elliptic curve. But the ApCoCoA package that has been presented in this thesis can do the things efficiently and accurately over elliptic curve. So, one can use it to make his life easy without compromising on the security.

Bibliography

[1] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, 1st ed. USA: CRC Press, Inc., 1996.

[2] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 5th ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010.

[3] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theor.*, vol. 22, no. 6, pp. 644–654, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1109/TIT.1976.1055638

[4] V. S. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology — CRYPTO '85 Proceedings*, H. C. Williams, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 417–426.

[5] L. Washington, "Elliptic curves number theory and cryptography," 01 2008.

[6] "Matlab optimization toolbox," ¡The year of your version, you can find it out using ver¿.

[7] W. Research, "Mathematica 8.0," Champaign, Illinios, 2010.

[8] "Mathematica," Available at https://www.desmos.com/calculator/ialhd71we3.

[9] "Math," Available at http://extranet.cryptomathic.com/elliptic/index.

[10] "Mathematical ecc," Available at http://www.christelbach.com/ECCalculator.aspx.

[11] A. Mousa and A. Hamad, "Evaluation of the rc4 algorithm for data encryption."

[12] ““Trapdoor function,” https://en.wikipedia.org/wiki/Trapdoor_function.

[13] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*, 1st ed. Springer Publishing Company, Incorporated, 2009.

[14] P. M. Prasad, “Symmetric key generation algorithm in linear block cipher over lu decomposition method,” *International Journal of Trend in Scientific Research and Development*, vol. 1, no. 4, Jun. 2017. [Online].

[15] A. Klein, *Stream Ciphers*. Springer Publishing Company, Incorporated, 2013.

[16] “Cryptographic protocols,” https://www.edureka.co/blog/what-is-cryptography/.

[17] “Symmetric/Aymmetric key protocol,” https://www.cheapsslshop.com/blog/symmetric-vs-asymmetric-encryption-whats-the-difference/.

[18] N. I. of Standards and Technology, “Advanced encryption standard,” *NIST FIPS PUB 197*, 2001.

[19] D. Coppersmith, “The data encryption standard (des) and its strength against attacks,” *IBM J. Res. Dev.*, vol. 38, no. 3, pp. 243–250, May 1994. [Online]. Available: http://dx.doi.org/10.1147/rd.383.0243

[20] T. El Gamal, “A public key cryptosystem and a signature scheme based on discrete logarithms,” in *Proceedings of CRYPTO 84 on Advances in Cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1985, pp. 10–18. [Online]. Available: http://dl.acm.org/citation.cfm?id=19478.19480

[21] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978. [Online]. Available: http://doi.acm.org/10.1145/359340.359342

[22] N. Koblitz, “Elliptic curve cryptosystems,” *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, Jan. 1987.

[23] R. Hofstede, M. Jonker, A. Sperotto, and A. Pras, "Flow-based web application brute-force attack and compromise detection," *Journal of Network and Systems Management*, vol. 25, no. 4, pp. 735–758, Oct 2017. [Online]. Available: https://doi.org/10.1007/s10922-017-9421-4

[24] A. Biryukov, *Ciphertext-Only Attack*. Boston, MA: Springer US, 2011, pp. 207–207. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_560

[25] R. Canetti, S. Halevi, and J. Katz, "Chosen-ciphertext security from identity-based encryption," in *Advances in Cryptology - EUROCRYPT 2004*, C. Cachin and J. L. Camenisch, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 207–222.

[26] A. Biryukov, *Known plaintext attack*. Boston, MA: Springer US, 2005, pp. 342–343. [Online]. Available: https://doi.org/10.1007/0-387-23483-7_224

[27] "RSA encryption/decryption," https://hackernoon.com/how-does-rsa-work-f44918df914b.

[28] A. M. Odlyzko, "Discrete logarithms in finite fields and their cryptographic significance," in *Proc. Of the EUROCRYPT 84 Workshop on Advances in Cryptology: Theory and Application of Cryptographic Techniques*. New York, NY, USA: Springer-Verlag New York, Inc., 1985, pp. 224–314. [Online]. Available: http://dl.acm.org/citation.cfm?id=20177.20197

[29] C. Varma, "A study of the ecc, rsa and the diffie-hellman algorithms in network security," in *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, March 2018, pp. 1–4.

[30] D. Mahto and D. K. Yadav, "Performance analysis of rsa and elliptic curve cryptography." *IJ Network Security*, vol. 20, no. 4, pp. 625–635, 2018.

[31] J. Silverman, *The Arithmetic of Elliptic Curves*, 01 2009, vol. 106.

[32] I. Blake, G. Seroussi, N. Smart, and J. W. S. Cassels, *Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series)*. USA: Cambridge University Press, 2005.

[33] K. E. Lauter and K. E. Stange, "The elliptic curve discrete logarithm problem and equivalent hard problems for elliptic divisibility sequences," in *Selected Areas in Cryptography*, R. M. Avanzi, L. Keliher, and F. Sica, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 309–327.

[34] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, November 1976.

[35] D. J. Bernstein, "Curve25519: New diffie-hellman speed records," in *Public Key Cryptography - PKC 2006*, M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 207–228.

[36] J. Abbott, A. M. Bigatti, and L. Robbiano, "CoCoA: a system for doing Computations in Commutative Algebra," Available at `http://cocoa.dima.unige.it`.

[37] "Apcocoa help," http://cocoa.dima.unige.it/WhatIsCoCoA/WhatIsCoCoA-EnglishUS.html.

[38] J. Abbott, A. M. Bigatti, and L. Robbiano, "CoCoA: Application Computations in Commutative Algebra," Available at `https://apcocoa.uni-passau.de/`.

[39] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Selected Areas in Cryptography*, B. Preneel and S. Tavares, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 319–331.

[40] S. Verma and B. Ojha, "A discussion on elliptic curve cryptography and its applications," *International Journal of Computer Science Issues*, vol. 9, 01 2012.

[41] "ASCII codes," Available at https://www.ascii-code.com/.

Appendix

This section contain the ApCoCoA code for calculation of Elliptic Curve. It consists of **ModuloInverse**, **ECaddition**, **NP**, **OrderP**.

ModuloInverse calculate the inverse of a number under the mod. It require the input N, P where P is number and N is mod. This function uses the extended euclidean inverse algorithm.

Define ModuloInverse(N,P)

```
A1:=1;A2:=0;A3:=P;
B1:=0;B2:=1;B3:=N;
While B3 < 0 Do
B3:=B3+P;
EndWhile;
While B3 <> 1 Do
Q:=Div(A3,B3);
If Q=0 Then Error(" Q is 0");EndIf;
T1:=A1-Q*B1;T2:=A2-Q*B2;T3:=A3-Q*B3;
A1:=B1;A2:=B2;A3:=B3;
B1:=T1;B2:=T2;B3:=T3;
If B2 < 0 Then B2:=B2+P; EndIf;
If B3=1 Then Return B2;EndIf;
If B3=0 Then Return("Not Invertible!"); EndIf;
EndWhile;
Return B2;
EndDefine;
```

==

Dec2Bin(D) converts the decimal number into binary.

==

```
Define Dec2Bin(D); L:=[];Q:=1;Rem:=0;
While $Q <> 0$ Do
Q:=Div(D,2);Rem:=Mod(D,2);
Append(L,Rem);
D:=Q;
EndWhile;
Return Reversed(L);
EndDefine;
```

ECaddition is use to add the points on $P1$ and $P2$ on curve C mod M. It require $P1$, $P2$, C, M where $P1$, $P2$ should be given as the list of integers x and y coordinates of the points. C is list containing a and b of elliptic curve.

```
Define ECadd(P1,P2,C,M)
If P1="Infinity" Then Return P2; EndIf;
If P2="Infinity" Then Return P1; EndIf;
If
$Mod(P1[2]^2, M) <> Mod(P1[1]^3 + C[1] * P1[1] + C[2], M)$ Then
Error("First Point is not on the Elliptic Curve");
EndIf;
If
$Mod(P2[2]^2, M) <> Mod(P2[1]^3 + C[1] * P2[1] + C[2], M)$ Then
Error("Second Point is not on the Elliptic Curve");
EndIf;
If P1[1]=P2[1] AND P1[2]=Mod(-P2[2],M)
Then Return "Infinity";
EndIf;
If P1=P2 Then
$S := Mod((3P1[1]^2 + C[1]) * ModuloInverse(2P1[2], M), M)$; Else
$S := Mod((P2[2] - P1[2]) * ModuloInverse(P2[1] - P1[1], M), M)$;
EndIf;
$Y0 := P1[2] - S * P1[1]$;
$XR := Mod(S^2 - P1[1] - P2[1], M)$;
$YR := Mod(-S * XR - Y0, M)$;
Return [XR,YR];
EndDefine;
```

==

NP calculates the scalar product of point P on curve C *mod* M. It require the input N, P, C, M where N is integer P is the point. It calculate N times P.

```
Define NP(N,P,C,M))
S:=P;
For I:=1 To N-1 Do
    S:=ECaddition(S,P,C,M);
EndFor;
Return S;
EndDefine;
```

===

OrderP calculates the order of point P on curve $C\ mod\ M$.

```
Define OrderP(P,C,M))
N:=1;
P1:=P;
While P1<> "Infinity" Do
    P1:=ECaddition(P1,P,C,M);
    N:=N+1;
    PrintLn("P", N, " = ", P1);
    Appendix A 120
EndWhile;
Return N;
EndDefine;
```

===