Signcryption Schemes with Forward Secrecy Based on Elliptic Curve Cryptography

Ramesh Kumar Mohapatra

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India

Signcryption Schemes with Forward Secrecy Based on Elliptic Curve Cryptography

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

by

Ramesh Kumar Mohapatra

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Orissa, 769 008, India

May 2010

Signcryption Schemes with Forward Secrecy Based on Elliptic Curve Cryptography

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

by

Ramesh Kumar Mohapatra

(Roll- 208CS214)

Under the guidance of

Prof.(Dr.) Banshidhar Majhi

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Orissa, 769 008, India

May 2010

Dedicated to my parents...

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, Orissa, India.

Dr. Banshidhar Majhi
Professor

May 25, 2010

Certificate

This is to certify that the work in the thesis entitled ***Signcryption Schemes with Forward Secrecy based on Elliptic Curve Cryptography*** by ***Ramesh Kumar Mohapatra*** bearing ***Roll No-208CS214*** is a bonafide record of research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of Master of Technology in Computer Science and Engineering in the department of Computer Science and Engineering, National Institute of Technology Rourkela. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Banshidhar Majhi

Acknowledgement

"Any fool can count the seeds in an apple. Only God can count all the apples in one seed."

First of all I would like to thanks, the Almighty God, without whose blessings I wouldn't have been writing this "thesis".

I owe deep gratitude to the ones who have contributed greatly in completion of this thesis.

Foremost, I would like to express my sincere gratitude to my advisor, Prof. Banshidhar Majhi for providing me with a platform to work on challenging areas of security. I express my heartfelt thanks to him, for his guidance, support, and encouragement during the course of my master study at the National Institute of Technology, Rourkela. I am especially indebted to him for teaching me both research and writing skills, which have been proven beneficial for my current research and future career. The experimental methods and results presented in this thesis have been influenced by him in one way or the other. It has been a great honor and pleasure for me to do research under Prof.(Dr.) Banshidhar Majhi's supervision.

I am also thankful to Prof.(Dr.) S. K. Rath, Prof.(Dr.) S. K. Jena, Prof.(Dr.) D. P. Mohapatra, Prof.(Dr.) A. K. Turuk, Dr. R. Baliarsingh, Prof.(Dr.) P. M. Khilar, Asst. Prof. Sujata Mohanty for giving encouragement during my thesis work.

I thank all the members of the Department of Computer Science and Engineering, and the Institute, who helped me by providing the necessary resources, and in various other ways, in the completion of my work.

Finally, I thank my parents and all my family members for their unlimited support and strength. Without their dedication and dependability, I could not have pursued my M.Tech. degree at the National Institute of Technology, Rourkela.

Ramesh Kumar Mohapatra

Abstract

In this thesis two efficient signcryption schemes based on elliptic curve cryptosystem are proposed which can effectively combine the functionalities of digital signature and encryption and also take a comparable amount of computational cost and communication overhead. They provide confidentiality, authentication, integrity, unforgeability and nonrepudiation, along with forward secrecy of message confidentiality and public verification. By forward secrecy of message confidentiality function we mean, although the private key of the sender is divulged inattentively, it does not affect the confidentiality of the previously stored messages. By the public verification function we mean, any third party can verify directly the signature of the sender of the original message without the sender's private key when dispute occurs. It enhances the justice of judge. In addition, proposed schemes save great amount of computational cost. The proposed scheme II gives a better result as compare to the proposed scheme I, but it requires a zero-knowledge interactive protocol to exchange recipient's private key to a third party or judge for verification. The proposed schemes can be applied to the lower computational power devices, like mobile devices, smart card based applications, e-voting and many more, due to their lower computational cost.

Keywords: *Signcryption, Public key cryptography, Elliptic curve cryptography, Digital signature, Forward secrecy.*

Acronyms

AES	Advance Encryption Standard
CA	Certificate Authority
CRHF	Collision Resistance Hash Function
DES	Data Encryption Standard
DLP	Discrete Logarithmic Problem
DSS	Digital Signature Standard
ECC	Elliptic Curve Cryptosystem
ECDLP	Elliptic Curve Discrete Logarithmic Problem
ECPA	Elliptic Curve Point Addition
ECPM	Elliptic Curve Point Multiplication
EEPROM	Electrically Erasable Programmable Read Only Memory
GF	Galois Field
HECC	Hyper Elliptic Curve cryptosystem
IFP	Integer Factorization Problem
KH	Keyed Hash Function
MAC	Message Authentication Code
MD	Message Digest
MDC	Modification Detection Code
MIPS	Million of Instructions per Second
OWHF	One Way Hash Function
RAM	Random Access Memory
ROM	Read Only Memory
RSA	Rivest Shamir Adleman
SDSS	Shortened Digital Signature Scheme
SECDSS	Shortened Elliptic Curve Digital Signature Scheme
SHA	Secure Hash Algorithm

Symbols and Notations

$\vert$	Divides
$\vert\vert$	Concatenation
$=$	Equality
$\equiv$	Congruence
$D_k(c)$	Symmetric key decryption
$E_k(m)$	Symmetric key encryption
$GF(p)$	The finite field of order p
mod	modulo operator
G	Group
E	Elliptic curve
$\forall$	For all
$\approx$	Approximately
$\times$	Multiplication
$\vert p\vert$	Number of bits in p
$hash()$	One way hash function
$\leq$	Less than equal to
Z_p	set of non-negative integers less than p

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 1

Introduction

Information is an asset that has a value like any other asset. As an asset, information needs to be secured from attacks. Now-a-days security becomes an essential feature in almost all area of communication. While sending a message to a person over an insecure channel such as internet we must provide confidentiality, integrity, authenticity and non-repudiation [1]. These are the four major security aspects [2] or goals. Before the modern era, cryptography was concerned solely with message confidentiality (i.e., encryption)-conversion of messages from a comprehensible form into an incomprehensible one and back again at the other end, rendering it unreadable by interceptors or eavesdroppers without secret knowledge (namely the key needed for decryption of that message). Encryption was used to (attempt to) ensure secrecy in communications, such as those of spies, military leaders, and diplomats. In recent decades, the field has expanded beyond confidentiality concerns to include techniques for message integrity checking, sender/receiver identity authentication, digital signatures, and interactive proofs and secure computation, among others. In ancient times, the use of cryptography was restricted to a small community essentially forms by the military and secret services. The keys were distributed secretly by a courier and the same key is used to encipher and decipher the message. We have a number of encryption algorithms those can be broadly classified into two categories: *Symmetric/Private key encipherment* and *Asymmetric/Public key encipherment* [3,4]. The difference between these two is that to communicate with n people *private key cryptography* requires $(n \times (n-1))/2$ number of keys as shown in the Figure 1.1 whereas; *public key cryptography* requires only n number of key pairs (one private and one public key) as shown in the Figure 1.2.

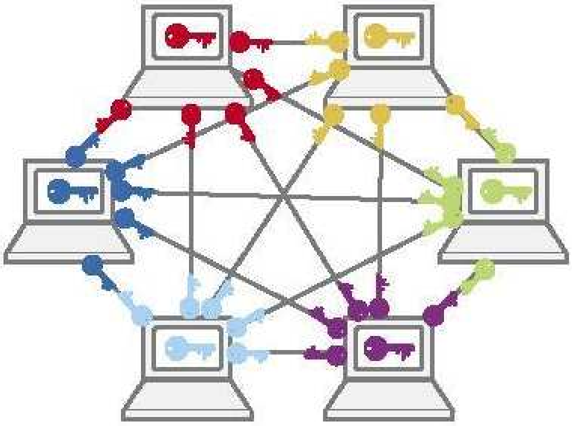

Figure 1.1: Symmetric Key Encryption requires $(n \times (n-1))/2$ number of keys.

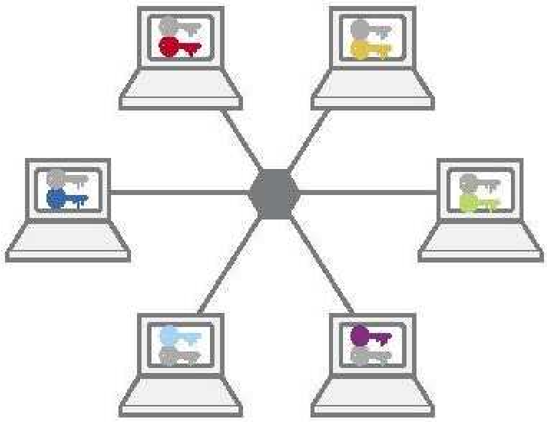

Figure 1.2: Asymmetric/public Key Encryption requires n number of key pairs.

Public key cryptography discovered nearly two decades ago has revolutionized the way for the people to communicate securely and in an authenticated way [1]. Any message (text, binary files, or documents) that are encrypted by using the public key can only be decrypted by applying the same algorithm, but by using the matching private key. Any message that is encrypted by using the private key can only be decrypted by using the matching public key. This means that you do not have to worry about passing public keys over the Internet (the keys are supposed to be public). A problem with asymmetric encryption, however, is that it is slower than symmetric encryption. It requires far more processing power to both encrypt and decrypt the content of the message. To check the authenticity of the message, i.e. the proof of originator, the sender has to sign the message before it gets delivered to the recipient. To achieve authenticity of the message the sender uses any one of the digital signature scheme [3, 4] depending upon the level of security. Message security and sender's authentication for communication in the open channel is a basic and important technology of Internet. Until the before decade, message encryption and digital signature have been viewed as important but distinct building blocks [5, 6] of various cryptographic systems. In the public key schemes, the traditional method is to digitally sign the message then encrypt it and send it to the recipient. Then the recipient will decrypt the message and check the authenticity of the message. We call this two step approach as *"Signature-Then-Encryption "*. The main disadvantage of this approach is that any arbitrary scheme can't guarantee the security. Signature generation and encryption consumes machine cycles, and also introduce "expanded" bits to an original message. Now it is possible to combine both the operations logically in a single step. This process is called *Signcryption* [1]. A signcryption scheme simultaneously fulfills the security attributes of an encryption and those of a digital signature. Such properties mainly include: confidentiality, integrity, unforgeability and non-repudiation. Some signcryption scheme provides further attributes such as public verifiability and forward secrecy of the message.

1.1 Message Encryption

Encryption means conversion of messages from a comprehensible form into an incomprehensible one and back again at the other end, rendering it unreadable by interceptors or eavesdroppers without secret knowledge (namely the key needed for decryption of that message). The sequence of data processing steps required for the transformation of the plaintext into cipher text is called message encryption. Various parameters used by an encryption algorithm, are derived from a secret key. As discussed in the previous Chapter we have a number of encryption algorithms. *DES* or *AES* can be used for message encryption. We can also use the RSA encryption algorithm for simplicity.

1.2 Digital Signature

Figure 1.3 shows the digital signature [7, 8] process. Here *Alice* is the sender and *Bob* is the receiver. *Alice* uses a signing algorithm to sign the message. The message and the signature are sent to the receiver. Then the receiver receives both and applies the verifying algorithm whether to accept the message or not. Several digital signature

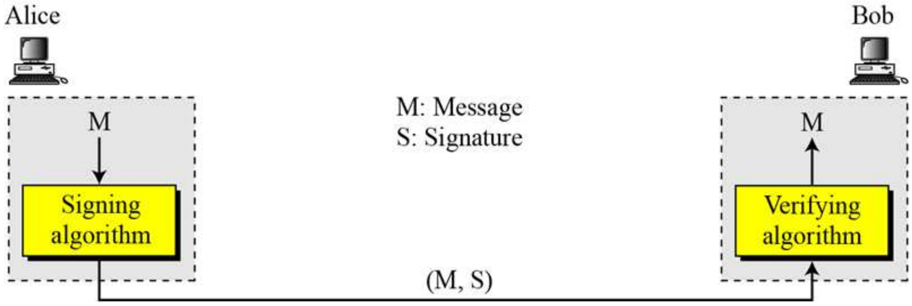

Figure 1.3: *Digital Signature Process.*

schemes have been evolved during the last few decades. Some of them have been implemented [9]. They are: *RSA Digital Signature Scheme*, *ElGamal Digital Signature Scheme*, *Schnoor Digital Signature Scheme*, *Digital Signature standard (DSS)*, *Elliptic Curve Digital signature Scheme.*
Now we can apply both the operations one after other to provide message confidentiality and authenticity. This is known as *"Signature-Then-Encryption"* [1]. In the next section, *"Signature-Then-Encryption"* is discussed more in detail.

1.3 Signature-Then-Encryption

In order to send a confidential letter in a way that it can't be forged, it has been a common practice for the sender of the letter to be sign it, put it in an envelope and then seal it before handing it over to be delivered.
Discovering public key cryptography has made communication between people who have never met before over an open and insecure network such as Internet [10], in a secure and authenticated way possible. Before sending a message the sender has to do the following:

1. Sign it using a digital signature scheme (DSS)
2. Encrypt the message and the signature using a private key encryption algorithm under randomly chosen encryption key
3. Encrypt the random message encryption key using receiver's public key
4. Send the message following steps 1 to 3

This approach is known as *"Signature-Then-Encryption "*. It can be shown in the following Figure 1.4. This figure has been taken from [10].

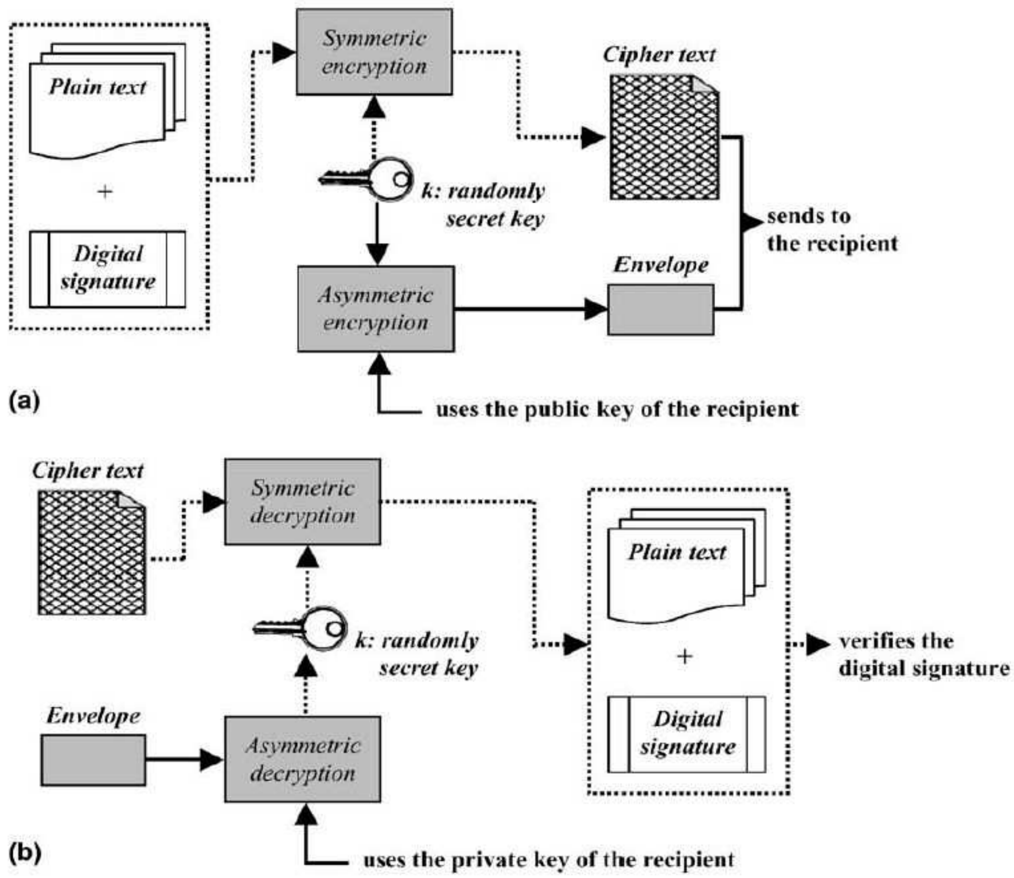

Figure 1.4: (a) Signature-Then-Encryption (b) Decryption-Then-Verification

1.4 Disadvantages of Signature-Then-Encryption Approach

The main disadvantage of this approach is that, digitally signing a message and then encrypting it, consumes more machine cycles and bloats the message by introducing extended bits to it [1]. Hence, decrypting and verifying the message at the receivers end, a lot of computational power is used up. Thus you can say that the cost of delivering a message using signing-then-encryption is in effect the sum of the costs of both digital signatures and public key encryption.

Is it possible to send a message of arbitrary length with cost less than that required by signature-then-encryption?

The answer is **yes**; according to Yuliang Zheng [1] it is possible to combine both the operations logically in a single step. It is discussed in the section 1.5.

1.5 Signcryption

1.5.1 What is signcryption?

The word *signcryption* was first coined by *Yuliang Zheng* in the year 1997 at Monash University, Australia. According to him signcryption [1] is a cryptographic primitive which combines both the functions of digital signature and public key encryption logically in a single step, and with a computational cost significantly less than that needed by the traditional signature-then-encryption approach.

1.5.2 Preliminaries on signcryption

A signcryption scheme typically consists of three algorithms: *Key Generation, signcryption, Unsigncryption.* The key generation algorithm generates all the public keys required for signcryption and unsigncryption. It also generates the key pair of Alice and Bob. The signcryption scheme will generate signcrypted message (c, r, s) and send it to Bob. Bob, the verifier decrypts the message and checks the authenticity of the message in the unsigncryption phase.

Any signcryption scheme should have the following properties:

1. *Correctness:* There exist an unsigncryption schemes from which the plain text can be recovered from the signcrypted message.

2. *Efficiency:* A signcryption scheme is said to be efficient if the computational cost and the communication cost should be smaller than that of signature-than-encryption standard.

3. Security: It should fulfill the security properties of both digital signature and encryption standard. Some of the security issues are discussed hereunder:

   - *Confidentiality:* It should be infeasible for an eavesdropper to get any information from the signcrypted message without knowing the sender′s and receiver′s private key.

- *Integrity:* The intended or authenticated user can only modify the content of the message.
- *Unforgeability:* There should not be two signcrypted messages which give the same plain text. Otherwise an adaptive attacker can create an authentic signcrypted text that can be accepted by the unsigncryption algorithm.
- *Forward Secrecy* If the long term private key of the sender is compromised, no one should be able to extract any information of the past messages.
- *Non-repudiation* After sending the message later Alice should not deny that she has sent the message or after receiving the message Bob can not deny that he has received the message.
- Public Verifiability Any third party or judge can verify whether the message has been sent by the intended user.

1.5.3 How it works?

The sender *Alice* uses her private key for signing the message and also uses the public key of the recipient for generating the secret key in the signcryption phase. After the

recipient receives the cipher text and the digital signature, he uses his private key to derive the same secret key and using that key he decrypts the message and avows the authenticity of the message originator [10].

1.5.4 Key Generation

The different phases of signcryption are discussed here after. These include: key generation, signcryption phase, unsigncryption phase, and finally verification phase.

Public key parameters

The public parameters used in the process of signcryption and unsigncryption are given below:

- p — a large prime.
- q — a large prime factor of p-1.
- g — an integer with order q modulo p chosen randomly from [1,...,p-1].
- hash — a one way hash function.
- KH — a keyed one way hash function.
- $E_k(\cdot)/D_k(\cdot)$ — symmetric encryption/decryption algorithm with private key k such as *DES* or *AES*.

Alice's keys:

1. x_a — *Alice's* private key chosen at random from [1,...,q-1].
2. y_a — *Alice's* public key $y_a = g^{x_a}$ mod p.

Bob's keys:

1. x_b — Bob's private key chosen at random from [1,...,q-1].
2. y_b — Bob's public key ($y_b = g^{x_b}$ mod p)

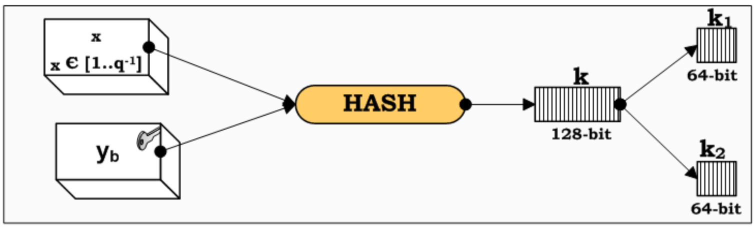

Figure 1.5: Key generation phase

1.5.5 Signcryption Phase

In this phase *Alice*, sends the signcrypted message to the recipient *Bob*. First she digitally signs the message then encrypts it and sends it to *Bob*.

Signcryption of a message by Alice the sender

- Choose a number x at random from the set [1, ..., q-1].

  And compute k = hash(y_b^x mod p).

  Split k into k_1 and k_2 of equal length.

- Calculate $r = KH_{k_2}(m)$.

- c = $E_{k_1}(m)$.

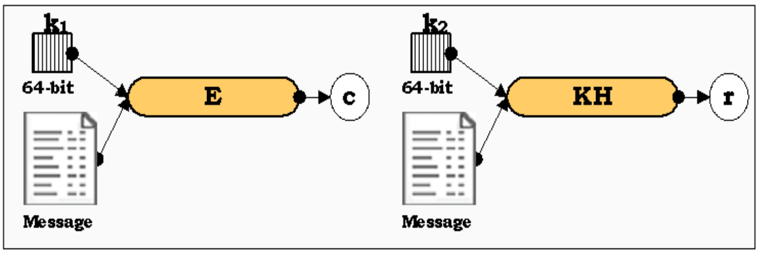

Figure 1.6: Signcryption phase

- $s = x/(r + x_a)$ mod q if SDSS1 is used. or

  $s = x/(1 + x_a \cdot r)$ mod q if SDSS2 is used instead.

- Send (c, r, s) to *Bob* the recipient.

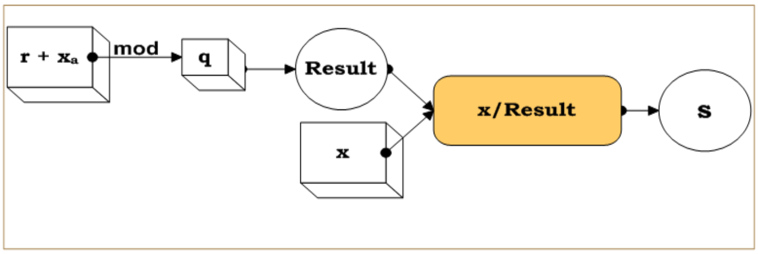

Figure 1.7: Signature

1.5.6 Unsigncryption Phase

In this phase *Bob* decrypts the message sent by *Alice* and verifies the authenticity of the message.

Unsigncryption Phase

After receiving the signcrypted message *Bob* does the following steps:

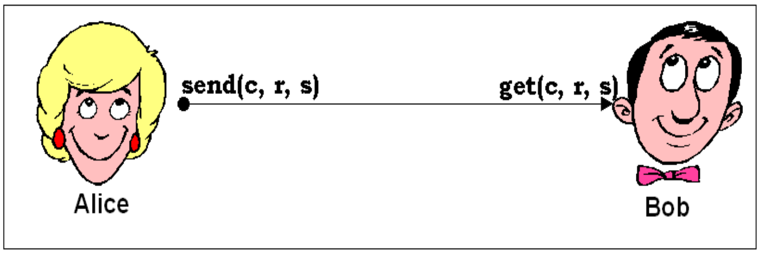

Figure 1.8: Alice sends c, r, s to Bob

- Compute k from r, s, g, p, y_a, x_b.

  $k = hash((y_a \cdot g^r)^{s \cdot x_b} \bmod p)$ if SDSS1 is used, or

  $k = hash((g \cdot y_a^r)^{s \cdot x_b} \bmod p)$ if SDSS2 is used.

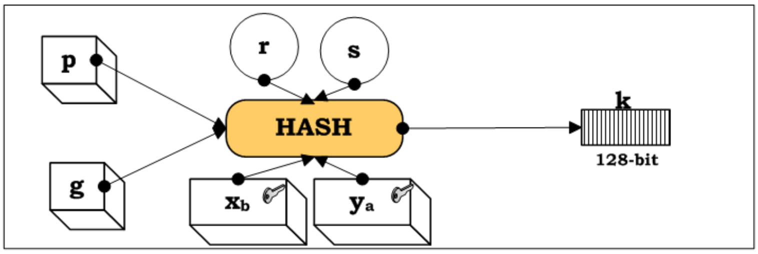

Figure 1.9: Derive the key K

- Split k into k_1 and k_2 of equal length.
- Calculate m = $D_{k_1}(c)$.

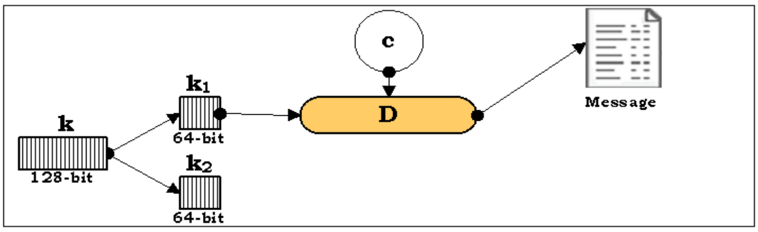

Figure 1.10: Decryption of the message

- Accept m if $KH_{k_2}(m) = r$. It ensures that the message has come from Alice. Otherwise he rejects.

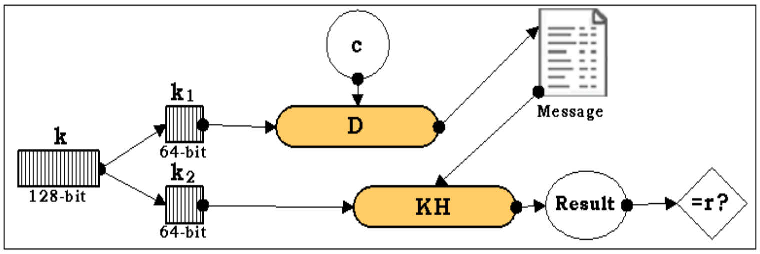

Figure 1.11: Verification

1.6 Cost of Signcryption vs. Cost of Signature-Then-Encryption

To compare the efficiency of two different methods for secure and authenticated message delivery, two types of cost involved [1, 10, 11]: computational cost and communication overhead (storage overhead). The *computational cost* indicates how much computational effort has to be invested both by the sender and recipient of a message. Generally, the computational cost is estimated by counting the number of dominant operations involved. Typically these operations include private key encryption and decryption, hashing, modular addition, multiplication, division and exponentiation. In addition to computational cost, digital signature and encryption based on public key cryptography also require extra bits to be appended to a message. This is known as communicational overhead. On the basis of these two we can say one algorithm is better than another if these costs are less in the former algorithm as compare to the later.
The advantage of signcryption over signature-then-encryption [1] lies in the dramatic reduction of computational cost and communication overhead which can be symbolized by the following inequality,

$$Cost(signcryption) \ll Cost(signature) + Cost(encryption)$$

where, [EXP = *the number of modular exponentiation*, MUL = *the number of modular multiplication*, DIV = *the number of modular division*, ADD = *the number of modular addition or subtraction*, HASH= *the number of one way or keyed hash function*, ENC = *the number of encryption using a private key cipher*, DEC = *the number of decryption using a private key cipher*. Parameters in the brackets indicate the number of operations involved in the unsigncryption process].
Table 1.1 gives the comparative analysis of signcryption scheme proposed by Yuliang

Table 1.1: Cost of signature-Then-Encryption vs. Cost of Signcryption

Various schemes	Computational cost	Communication overhead
signature-then-encryption based on RSA	EXP=2,HASH=1,ENC=1 (EXP=2,HASH=1,DEC=1)	$\|n_a\| + \|n_b\|$
signature-then-encryption based on "DSS + ElGamal encryption"	EXP=3,MUL=1,DIV=1 ADD=1,HASH=1,ENC=1 (EXP=2.17,MUL=1,DIV=2 ADD=0,HASH=1,DEC=1)	$2\|q\| + \|p\|$
signature-then-encryption based on "Schnoor signature + ElGamal encryption"	EXP=3,MUL=1,DIV=0 ADD=1,HASH=1,ENC=1 (EXP=2.17,MUL=1,DIV=0 ADD=0,HASH=1,DEC=1)	$\|hash(.)\| + \|q\| + \|p\|$
signcryption SCS1	EXP=1,MUL=0,DIV=1 ADD=1,HASH=2,ENC=1 (EXP=1.17,MUL=2,DIV=0 ADD=0,HASH=2,DEC=1)	$\|KH(.)\| + \|p\|$
signcryption SCS2	EXP=1,MUL=1,DIV=1 ADD=1,HASH=2,ENC=1 (EXP=1.17,MUL=2,DIV=0 ADD=0,HASH=2,DEC=1)	$\|KH(.)\| + \|p\|$

Zheng, which was based on DLP, with signature-then-encryption schemes. From this it is clear that signcryption scheme saves less computational time as well as communication overhead as compare to other schemes.

With the signature-then-encryption based on Schnoor digital signature and ElGamal encryption [4], the number of modular exponentiation is three, for both the processes. Out of three, two of them are used to verify the signature. That′s why they spent more time in computing $g^s \cdot y_a^r$ mod p. Shamir [10] has suggested a technique (see Appendix B) for fast computation of the product of multiple exponentials with the same with the same modulo such as $g^s \cdot y_a^r$ mod p can be computed, on an average, in (1+3/4)$|q|$ modular multiplication.

Since a modular exponentiation can be completed, on average, in about 1.5$|q|$ modular multiplications using the well known square-and-multiply method [4], (1+3/4)$|q|$ modular multiplication is computationally equivalent to 1.17 modular exponentiation. Thus, the number of modular exponentiations involved in the decryption-then-verification phase of Schnoor Digital signature scheme, can be reduced from 3 to 2.17. Combined computational cost for both sender and receiver will be 5.17 as compare to 2.17 for the signcryption scheme shown in the Table 1.1.

$$\frac{5.17 - 2.17}{5.17} = 58\%$$

So this represents a reduction of 58% in average computational cost.

Saving in computational overhead

From the Table 1.1 it may conclude that the saving in communication overhead will be,

$$\frac{|hash()| + |q| + |p| - (|KH()| + |q|)}{|hash()| + |q| + |p|}$$

Yuliang Zheng assumed that $|hash()| = |KH()| \cong |q|/2$, $|p|$=512 bits and $|q|$=144 bits. Hence the saving in communication overhead will be 70.3%.

1.7 Application of Signcryption

As discussed in the introduction, a major motivation of this work is to identify for a more efficient method for secure and authenticated message transfer. So it can be applied in smart card based applications, e-voting, personal health card, digital card payment systems and many more. If we could apply digital signcryption in this area, then we could save 50% computational cost and 85% communication overhead.

1.8 Motivation

Public key cryptography has drawn considerable attention over the last two decades. From there onwards a lot of public key cryptosystems have been stated. It includes RSA cryptosystem, Rabin Cryptosystem, ElGamal cryptosystem, Elliptic Curve cryptosystem. Currently, the elliptic curve cryptography is being used in a wide variety of applications. The elliptic curve based cryptosystem (ECC) [12, 13] can attend to a desire security level with significantly smaller keys than those of required for their counterparts. It is suitable for smart card based application, e-voting and also in other areas like digital cash payment system. There are lots of constraints like memory, bandwidth, and computational speed that must be considered while developing smart cards. ECC's unique properties make it especially well suited for smart card [14] based applications and in any such area of applications. It provides the highest strength per bit of any cryptosystem known today. Now it is better to use ECC to generate the keys for signcryption as well as unsigncryption phase. This could save the computational cost and communication overhead.

1.9 Thesis Organization

The rest of the thesis is organized as follows:

Chapter 2 discusses various literature surveyed related to the work. The Zheng and Imai scheme as well as Hwang et. al signcryption scheme are discussed here in detail which are based on elliptic curve cryptography.

The mathematical preliminaries required for the implementation of the proposed schemes have been depicted in **Chapter 3**. SHA-1 hash function is used to produce a message digest of size 160 bits. A comparison has been made between ECDLP and DLP. The elliptic curve cryptosystem is discussed here more in detail. Finally it ends with the application of ECC.

In **Chapter 4**, two forward secure signcryption schemes have been proposed which are based on ECC. The security features along with the cost of computation and communication are analyzed. It has been observed that the proposed approaches performs better in terms of computational cost and communication overhead as compare to other existing signcryption schemes.

Finally, **Chapter 5** discusses the concluding remarks with the scope of further research work.

Chapter 2

Literature Review

Related Work
Observation
Problem Definition
Contribution
Summary

Chapter 2

Literature Review

2.1 Related Work

Many of the proposed signcryption scheme include modular exponentiation while some of them are based on elliptic curves. Y.Zheng [1] proposed signcryption scheme which saves about 58% computational cost and saving about 40% communication cost than the traditional signature-then-encryption scheme . This scheme was based on discrete logarithmic problem. It involves modular exponentiation and RSA takes a large key size of about 1024 bits. After that, Jung et al. showed that Zheng's scheme does not provide forward secrecy of message confidentiality when the sender's private key disclosed. They also proposed a new signcryption based on discrete logarithm problem (DLP) with forward secrecy. In Jung's scheme [15], even attacker obtains the sender's private key, he cannot get the corresponding original message yet that sender had sent. However, in those research results, when dispute occurs, the judge cannot directly verify the signature because of not knowing the recipient's private key. Bao and Deng [16] enhanced Zheng's signcryption scheme such that the judge can verify signature without the recipient's private key. But a key exchange protocol is required in the process of verification. Gamage et al. [17] modified Zheng's signcryption scheme so that anyone can verify the signature of cipher text. Their scheme only verifies the cipher text to protect confidentiality of message in firewall application. Then Zheng and Imai [18] suggested an ECC based signcryption scheme thus providing all the basic security features, with cost less than as required by *"signature-then-encryption "*. They choose ECC because elliptic curve based solutions are usually based on the difficulty of ECDLP which is discussed in the next Chapter. As it is based on elliptic curve cryptosystem the key size used is smaller as compare to the other schemes, which is one of the advantages of this scheme but still it needs forward secrecy.

2.1.1 Zheng and Imai Signcryption Scheme:

Task: Alice has to send a message m to Bob. The *key generation phase* has the following steps:

Public parameters:

- C — Consider C as an elliptic curve over a finite field GF(p^m) ,either with p $\geq 2^{160}$ and m=1 or p=2 and m$\geq$160.
- q — a large prime whose size is approximately of order p^m-1.
- G — a point with order q. Chosen randomly from the points on C.
- hash(.) — a one way hash function whose output has say at least 160 bits.
- KH(.) — a keyed one-way hash function.
- (E,D) — the encryption and decryption algorithms of a private key cipher.

Alice's keys:

- v_a — Alice's private key chosen uniformly at random from [1,...,q-1].
- P_a — Alice's public key. ($P_a = v_a$G , a point on C).

Bob's keys:

- v_b — Bob's private key chosen uniformly at random from [1,...,q-1].
- P_b — Bob's public key. ($P_b = v_b$G , a point on C).

Signcryption scheme by Zheng and Imai

v ϵ [1,...,q-1].A random number chosen by Alice.
(k_1, k_2)=hash(vP_b).
c=E_{k_1}(m).
r=$KH_{k_2}(m, blind\_info)$.
s= $\frac{v}{r+v_a}$ mod q.

send c,r,s to Bob.

Unsigncryption scheme by Zheng and Imai

u=sv_b mod q.

(k_1, k_2)=hash(uP_a + urG).

if SECDSS1 is used, or

(k_1, k_2)=hash(uG + urP_a).

if SECDSS2 is used.

m=D_{k_1}(c).

Accept m only if

$KH_{k_2}(m, blind\_info)$=r.

The disadvantage of the above scheme is that it doesn't support forward secrecy and encrypted message authentication. From the above Zheng and Imai scheme we can see that if *Alice* divulged his private key v_a inattentively then an adversary can get the information about the past messages.

Now let′s discuss Hwang et al. [10] signcryption scheme based on elliptic curve cryptosystem, which provides forward secrecy.

2.1.2 Hwang et al. Signcryption Scheme:

Task: Alice has to send a message m to Bob. The *key generation phase* has the following steps:

Signcryption Phase:

Step 1: Choose r ϵ [1,...,q-1]

Step 2: K = rP_b = (x_k, y_k)

Step 3: Calculate R = rG = (x_R, y_R)

Step 4: c=E_{x_k}(m).

Step 5: e = hash(m$||x_R$).

Step 6: s = v_a-er mod n.

sends c, R, s to Bob.

Unsigncryption Phase:

Step 1: K = v_bR = (x_k, y_k)

Step 2: m=D_{x_k}(c).

Step 3: e = hash(m||x_R).

Step 4: He verifies the signature by comparing sG + eR = P_a ?

The Hwang et al scheme satisfies all the security attributes. The signcryption phase involve with 2 elliptic curve point multiplication in the signcryption phase and 3 elliptic curve point multiplication in the unsigncryption phase. In **chapter 4** this scheme has been compared with the proposed scheme.

2.2 Observation

So finally it has been observed that out of all signcryption schemes stated above the signcryption scheme given by Zheng and Imai [18] supports all the four major security goals, discussed in the introduction Chapter, takes a considerable amount of computational and communication overhead. But it does not support forward secrecy. So our objective is to propose a new scheme such that it will take a comparable computational and communication cost and should provide forward secrecy.

2.3 Problem Definition

From the above scheme of Zheng and Imai [18] we may conclude that if the private key of Alice divulged inattentively then anyone can get the message. But it generally doesn't happen; if it happens then the system is no more secure. Hence it is necessary to provide forward secrecy to the past messages. Elliptic curve cryptosystem [19, 20] is one of the best public key cryptosystem which gives same level of security with smaller key size as required by its counterparts. This can enhance the speed. The power, bandwidth, and storage space which are basic limitations of resource restricted devices can be efficiently utilized. So we need to formulate signcryption schemes based on ECC thus providing the forward secrecy and public verification.

2.4 Contribution

In this thesis, two signcryption schemes based on elliptic curve have been proposed and implemented. They provide all the basic security goals such as confidentiality, authentication, integrity, unforgeability and non-repudiation, along with forward secrecy of message confidentiality and public verification. By forward secrecy of message confidentiality function, although the private key of the sender is disclosed, it does not affect the confidentiality of previous messages. Any third party should be able to verify directly the signature of original message without the sender's private key when dispute occurs [10]. In addition, they save great amount of computational cost. The proposed schemes can be applied to the lower computational power devices like mobile devices and many more, efficiently due to their lower computation cost.

2.5 Summary

Signcryption is a new cryptographic primitive which can fulfill both message encryption and signature logically in a single step thus reducing the computational cost and communication overhead. For the implementation of signcryption and unsigncryption algorithms, based on ECC, we must know about the elliptic curve cryptosystem and hash functions. These are discussed in the next Chapter in detail.

Chapter 3

Mathematical Background

Mathematics of Cryptography
Secure hash algorithm (SHA-1)
Elliptic curve Cryptosystem
Elliptic curves over Finite Fields
Application of ECC
Summary

Chapter 3

Mathematical Background

3.1 Mathematics of Cryptography

The basic properties of groups, rings, fields [2, 4, 8] along with the fundamentals of elliptic curve cryptography are discussed in this Chapter along with the secure hash algorithm (SHA-1) [4].

3.1.1 Elementary algebraic structures

Groups

Definition 3.1: A **Group (G)** is a set of elements with a binary operator "•"that satisfies the following four properties

1. *Closure*: If x and y are the elements of **G**, then z = x •y is also an element of **G**.
2. *Associativity*: If x, y and z are elements of **G**, then (x •y) •z = x •(y •z).
3. *Existence of identity*: For all x in **G**, there exist an element e, called the identity element, such that e •a = a •e = a.
4. *Existence of inverse*: For each x in **G**, there exists an element x', called the inverse of x, such that x •x' =x' •x = e.

Along with those properties if it also satisfies the commutative property then it is called as *commutative group* or *abelian group*. Commutative property means for all x and y in **G**, we have x •y = y •x.

Ring

A **Ring** is an abelian [4] structure with two operations. It is denoted as $R =< \{...\}, \bullet, \square >$ The first operation must satisfy all the five properties that is required for an abelian group. The second operation must satisfy only the first two. In addition, it should also satisfy the *Distributive* property which states that for all x, y and z elements of **R**, we have $x\square(y \bullet z) = (x\square y) \bullet (x\square z)$ and $(x \bullet y)\square z = (x\square z) \bullet (y\square z)$.
A Ring is said to be commutative ring if the second operation also satisfy the commutative property.

Field

A **Field**, denoted by $F =< \{...\}, \bullet, \square >$, is a commutative ring in which the second operation satisfies all the five properties defined for the first operation except that the identity of the first operation.

Finite Fields

Only finite fields are extensively used in cryptography. Galois showed that for a field to be finite, the number of elements should be p^n, where p is a prime and n is a positive integer. The finite fields are usually called as **Galois fields** and denoted as **GF(**P^n**)**.

GF (P)

When n=1, we have **GF(P)** field [2,4]. This field consists of the elements 0, 1, ..., P-1, with two arithmetic operations addition and multiplication.

3.2 Secure hash algorithm (SHA-1)

The Merkle-Damgard scheme [4] is the basic for many cryptographic hash functions today. We should use a compression function that is collision resistant. There are two different approaches in designing a hash function: it can be made from scratch like MD, MD2, MD4, MD5, SHA, SHA1. Second approach is that it can also be designed by using symmetric key block cipher. SHA-1 [21] hash function is being used in our schemes. A hash function is a function *hash()* which should satisfy the following properties:

- *Compression – hash()* takes input m of arbitrary length and produce a fixed length string output *hash(m)*.
- *Non-invertible*– Given *hash(m)* and hash() it is difficult to get m.

Two types of hash function are discussed: keyed or non-keyed hash function.
Modification detection code (MDC) [8] is a non-keyed hash function which is further divided into one way hash function(OWHF) and collision resistance hash function (CRHF). Both of them supports random oracle model as described in [4].

In our thesis work we have used the SHA-1, one way hash function, for the message digest. This compression function will give a fixed length output of 160 bits. Maximum message size that it takes is 2^{64}-1 and the block size is 512 bits.. Total 80 numbers of rounds has been used with a word size of 32 bits. For the implementation of SHA-1 you may refer [4].

3.3 Elliptic curve Cryptosystem

3.3.1 Introduction

Since the invention of public key cryptography in 1976 by Whitefield Diffie and Martin Hellman numerous public key cryptographic systems have been proposed. All of these systems are based on the difficulty of solving a mathematical problem. Over the years, many of the public key cryptography systems have been broken and some are proved to be impractical. Today only three types of system are considered to be safe, secure and efficient. They are,

1. Integer factorization problem (IFP)
2. Discrete Logarithm Problem (DLP)
3. Elliptic Curve Discrete Logarithm Problem (ECDLP)

3.3.2 Integer factorization problem

The integer factorization problem (IFP) is the following: given a composite number n that is the product of two large prime numbers p and q, find p and q. While finding large prime numbers is a relatively easy task, the problem of factoring the product of two such numbers is considered computationally intractable if the primes are carefully selected. Based on the difficulty of this problem, Rivest, Shamir and Adleman [2] developed the RSA public-key cryptosystem.

3.3.3 Discrete Logarithm Problem

If p is a prime number, then Zp denotes the set of integers 0, 1, 2, . . . , p - 1, where addition and multiplication are performed modulo p. It is well-known that there exists a non-zero element $\alpha \in Z_p$ such that each non-zero element in Z_p can be written as a power of α such an element α is called a generator of Z_p. The discrete logarithm problem (DLP) [5] is the following: given a prime p, a generator α of Z_p, and a non-zero element $\beta \in Z_p$, find the unique integer k, $0 \leq$ k $\leq$ p - 2, such that $\beta \equiv \alpha^k$(mod p).

The integer k is called the discrete logarithm of β to the base α.

3.3.4 Elliptic Curve Discrete Logarithm Problem

If q is a prime power, then F_q denotes the finite field containing q elements. In applications, q is typically a power of 2 (2^m) or an odd prime number (p). The elliptic curve discrete logarithm problem (ECDLP) [13, 15, 20, 22] is the following: given an elliptic curve E defined over F_q, a point P$\in$(F_q) of order n, and a point Q$\in$(Fq), determine the integer k, $0 \leq$ k $\leq$ p - 1, such that Q = kP, provided that such an integer exists.

3.3.5 Comparison

Figure 3.1 compares, the time required to solve an instance of a problem based on ECC with the time required to solve the problem based on IFP or DLP. Here the time is measured in MIPS. As a benchmark, it is generally accepted that 10^{12} MIPS years represents reasonable security at this time. In the Figure 3.1 the times of RSA and DSA are grouped together because the asymptotic running time for both is same. As we can see that to achieve reasonable security, RSA and DSA should employ 1024-bit modulo, while a 160-bit modulus should be sufficient for ECC [19,20,23]. Moreover, the security gap between the systems increases dramatically as the modulo sizes increases.

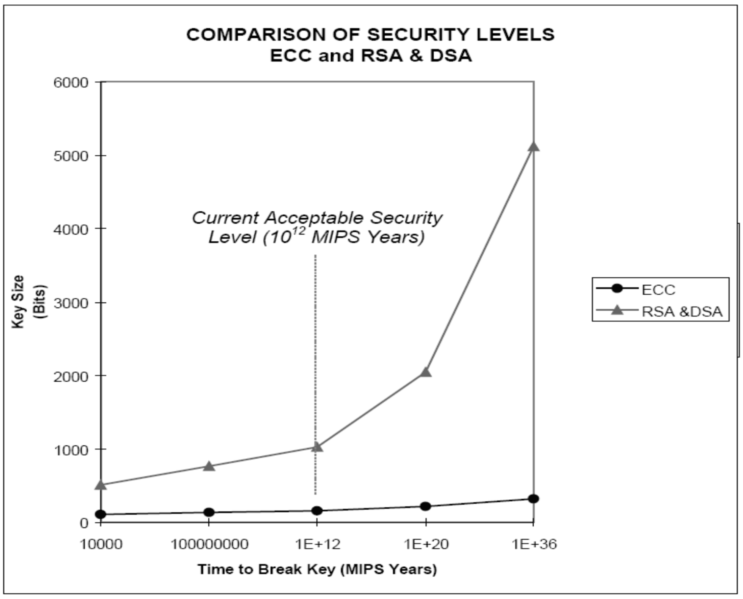

Figure 3.1: Comparison of security levels

3.4 Elliptic curves over Finite fields

Elliptic curve cryptosystems(ECCs) are becoming more popular because of the reduced number of key bits required in comparison to other cryptosystems(e.g. a 160 bit ECC has roughly the same security strength as 1024 bits RSA). The original ElGamal public key encryption and digital signature schemes are defined on finite fields. In 1985 Niel Koblitz and Victor Miller from the University of Washington proposed the elliptic curve cryptosystem (ECC). Elliptic curves over finite fields appeared to be intractable and hence ElGamal encryption and signature schemes have natural counterparts on these curves. Elliptic curve can be defined over F_q and F_{2^m}. For simplicity we will discuss only Elliptic curves over F_q which is discussed in the next section.

3.4.1 Elliptic curves over F_q

A finite field is a set of elements that have a finite order (number of elements).The order of *Galois field (GF)* [4] is normally a prime number or a power of a prime number.
Assume first that F_q has characteristic greater than 3. An elliptic curve E over F_q is the set of all solutions (x, y) $\in \overline{F}_q \times \overline{F}_q$ to an equation called *Weierstrass equation* (see Appendix A).

$$y^2 = x^3 + ax + b$$

where a, b $\in F_q$ and $4a^3 + 27b^2 \neq 0$, together with a special point ∞ called the point at infinity [8].

It is well known that E is an (additively written) abelian group [4] with the point ∞ serving as its identity element. The rules for group addition are summarized below.

Addition Formulas for the Curve (1). There is a rule, called the chord-and-tangent rule, for adding two points on an elliptic curve $E(F_q)$ to give a third elliptic curve point. Together with this addition operation, the set of points $E(F_q)$ forms a group with ∞ serving as its identity. It is this group which is used in the construction of elliptic curve cryptosystems. The addition rule is best explained geometrically.

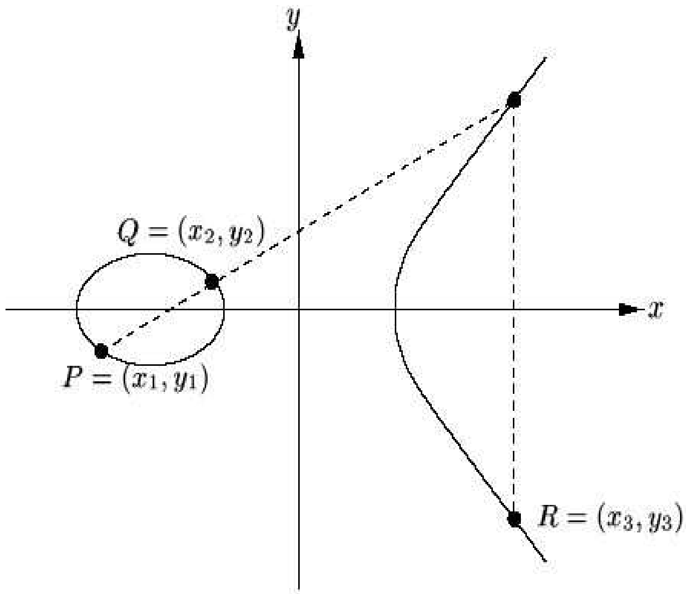

Figure 3.2: Addition of two points P and Q : R=P+Q

Let P(x_1, y_1) and Q(x_2, y_2) are the two points on the elliptic curve. Then the sum of P and Q is denoted by another point say R=(x_3, y_3) as it is shown in Figure3.2 Let P =(x_1,y_1) $\in$ E ; then -P = $(x_1,-y_1)$. If Q = (x_2,y_2) $\in$ E , $Q \neq -P$, then P + Q = (x_3,y_3),where

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1,$$

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

The above elliptic curve is said to be non singular if it satisfy the following condition,

$$4a^3 + 27b^2 \neq 0$$

else, the curve is singular.

Adding and doubling points on an elliptic curve C over GF(2^m) are defined in a similar way as shown in the Fig. 3.

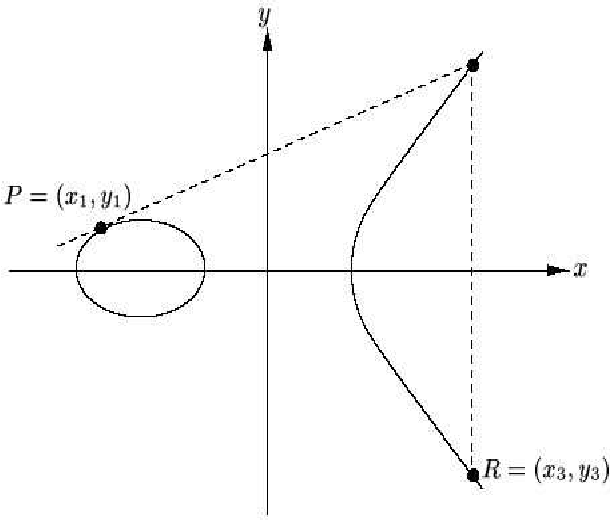

Figure 3.3: Doubling a point P : R=P+P=2P

Excluding the point at infinity ∞ every point P =(x,y) on an elliptic curve C over GF(p^m) can be represented as (or "compressed" to) P = (x,$\tilde{y}$),where $\tilde{y}$ is a single bit:
1. if x =0 then $\tilde{y}$ =0.
2. if x $\neq$ 0, then $\tilde{y}$ is the parity of y when it is viewed as an integer.

An advantage of compressed representation of a point is that when a compressed point is stored internally in a computer or communicated over a network, it takes only one bit more than half of the bits required for storing or transmitting its uncompressed counterpart. This advantage, however, is not for free: recovering the y-coordinate from a compressed point involves a few arithmetic operations in the underlying finite field [11, 18].

If E is an elliptic curve over a finite field F_q , then let $E(F_q)$ denote the points in E having both coordinates in F_q , including the point ∞; the points in E(F_q) are also known as F_q - rational points. E(F_q) is an abelian group of rank 1 or 2. We have E(F_q) $\cong C_{n_1} \oplus C_{n_2}$, where C_n denotes the cyclic group of order n, n_2 divides n_1, and furthermore $n_2|q-1$. A well known theorem of Hasse states [18, 20, 24] that E(F_q) = q + 1 - t, where $|t| \leq 2\sqrt{q}$. The curve E is said to be super singular if t^2 = 0, q, 2q, 3q, or 4q; otherwise the curve is non-super singular. If q is a power of 2 and E is super singular, then E(F_q) is odd; if q is a power of 2 and E is non-super singular, then E(F_q) is even. If q is a prime, then for each t satisfying $|t| \leq 2\sqrt{q}$ there exists at least one elliptic curve E defined over F_q with E(F_q) = q + 1 - t; if q is a power of 2, then for each odd t satisfying $|t| \leq 2\sqrt{q}$ there exists at least one (non-super singular) elliptic curve E defined over F_q with E(F_q) = q + 1 - t [24].

3.4.2 ECC Domain Parameters

Elliptic curve cryptography (ECC) domain parameters over **GF(P)**, can be represented by a six tuple:
E = (q, a, b, G, n, h), where
q = P or q = 2^m, where m is a natural number.
a and b are the co-efficients of x^3 and x respectively used in the equation.

$$y^2 \equiv x^3 + ax + b \text{ (mod P) for q} = \text{P} \geq 3$$

$$y^2 + xy = x^3 + ax^2 + b \text{ for q} = 2^m \geq 1$$

G is a base point on the elliptic curve.
n is prime number which is of the order of G. The order of a point on an elliptic curve is the smallest positive integer r such that rP = ∞.
Finally h = $|E|$/n. where $|E|$ represents the total number of points on elliptic curve and it is called the curve order.

3.4.3 ECC key generation

A public key Q = (x_q, y_q) associated with a domain parameter (q, a, b, G, n, h) is generated for an entity say *Alice* using the following procedure:

- Select a random number or pseudo random integer d in the interval [1, ..., n-1].
- Compute Q=dG.
- *Alice's* public key is Q and her private key is d.

3.4.4 ECDSA

The elliptic curve digital signature algorithm (ECDSA) was proposed by Abdalla, Bellare and Rogaway in 1999 [4, 24]. Entity *Alice* has domain parameters D = (q, a, b, G, n, h) and public key Q_a and private key d_a. And entity *Bob* has authentic copies of D and Q_a. To sign a message m, *Alice* does the following:

- Select a random integer k from [1,n-1].
- Compute $kG = (x_1, y_1)$ and $r = x_1 modn$. If r = 0 then go to step 1.
- Compute k^{-1} mod n. Compute e = hash(m).
- Compute $s = k^{-1}e + d_a.r$ mod n. If s = 0 then go to step 1.
  Alice's signature for the message m is (r, s).

To verify *Alice's* signature (r, s) on m, *Bob* performs the following steps: Verify that r and s are integers in [1,n-1].

- Compute e = hash(m).
- Compute w = s^{-1} mod n.
- Compute u_1 = ew mod n and u_2 = rw mod n.
- Compute (x_1, y_1) = u_1G+ $u_2 Q_a$.
- Compute v = x_1 mod n. Accept the signature if and only if v = r. SHA-1 denotes the 160-bit hash function.

3.5 Application of ECC

Smart card [25] is a very good example where ECC can be applied. These are used in a wide variety of applications such as e-commerce, identification and many more. It is really advantageous if we use asymmetric key cryptography for short messages. The most important constraints from the programmer's point of view are the limited RAM, ROM and EEPROM available, and the requirement to limit the processing time. A typical inexpensive smart card has between 128 and 1024 bytes of RAM, 4 and 16 Kbytes of EEPROM, and 16 and 32 Kbytes of ROM. The CPU is typically 8-bit and clocked at 3.57 MHz [14, 25]. Any addition to memory or processing capacity increases the cost of the card. It is certainly not acceptable to use all resources available on the smart card to implement the cryptographic services. So it is better to use ECC on smart card which uses smaller key size. Table 3.1 shows a comparison of key sizes of different cryptosystems. From this it is clear that ECC gives same level of security with a smaller key size as compare to others. But we must choose the ECC parameters carefully.

Table 3.1: A comparison of key sizes needed to achieve equivalent level of security with three different methods.

Symmetric Encryption Key size in bits	RSA and Diffie-Hellman Key size in bits	Elliptic Curve Key size in bits
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	512

3.5.1 Summary

The use of public key cryptography received considerable attention. An important advantage of elliptic curve is the shorter key lengths. ECC's unique properties make it especially well suited for smart card base applications and in any such area of application. It provides the highest strength per bit of any cryptosystem known today. Based on the best known algorithms today, one can estimate that 160-bit elliptic curves correspond to 1024-bit RSA, and 224-bit elliptic curves correspond to 2048-bit RSA (Table 3.1).

Even though ECC was proposed in 1985 [8], the market was initially reluctant to move towards this new and more complex primitive. However, recently ECC has been adopted by the governments of Austria, Germany and the USA and is gaining more widespread acceptance [25]. The main attraction lies clearly in the shorter key lengths, this advantage over RSA will increase over time.

Chapter 4

The proposed signcryption schemes

The proposed signcryption schemes with forward secrecy
Algorithm of the Proposed Scheme I
Algorithm of the Proposed Scheme II
The security functions of the proposed schemes
Implementation of the proposed algorithms using JAVA
Cost Analysis

Chapter 4

The proposed signcryption schemes

4.1 The proposed signcryption schemes with forward secrecy

Two signcryption schemes are proposed here in this thesis, which provide the security functions such as message confidentiality, sender's authenticity, message integrity, non-repudiation, forward secrecy and public verification, with a cost less than or comparable with the existing schemes. Both of them are based on elliptic curve cryptosystem. The proposed schemes spend lower time in computation, especially for sender. It contains four phases: initialization phase, signcryption phase, unsigncryption phase and judge verification phase. In the initialization phase, system generates and publishes domain parameters of elliptic curve, and each user generates his own private key and the related public key. Each user should get the certification of his public key from the certificate authority (CA) [4, 8].

4.1.1 Algorithm of the proposed scheme I

In the signcryption phase, the sender Alice signs and encrypts a message. Then she sends the signcrypted text to the recipient Bob. In the unsigncryption phase, the recipient Bob derives secret key to decrypt plain text. He also verifies the signature. In the judge verification phase, a judge decides whether the sender Alice sent the signcrypted message or not, when dispute occurs. We describe these four phases in the following [6, 10].

Initialization phase:

In this phase, some public parameters are generated. The steps are as follows:

q — a large prime number, where q is greater than 2^{160} [15].

a, b are two integer elements which are smaller than q and satisfy the following condition

$$4a^3 + 27b^2 \text{ mod q } \neq 0$$

Let F the selected elliptic curve over finite field q: $y^2 = x^3$ + ax + b mod q,
G — a base point of elliptic curve F with order n,
O — a point of F at infinite,
n — the order of point G, where n is a prime, n×G = O and n $\geq 2^{160}$. (The symbol "× "denotes the elliptic curve point multiplication,)
H — a one-way hash function,
$E_k(\cdot)/D_k(\cdot)$ — symmetric encryption/decryption algorithm with private key k such as DES or AES.

The sender Alice randomly selects an integer d_a as her private key and $d_a \leq$ n-1. She computes her public key $P_a = d_a$·G. The recipient Bob also selects private key d_b and public key $P_b = d_b$·G by the same way as Alice. They need to get a certificate of their public key from the certificate authority (CA).

Signcryption phase:

Assume that Alice wants to send a message m to Bob. Alice generates digital signature (T, s) of message m and uses the symmetric encryption algorithm and secret key k to encrypt m. Let c be the cipher text. Alice generates the signcrypted text (c, T, s) in the following steps.

Step 1: Verifies Bob's public key P_b by using his certificate.
Step 2: Randomly selects an integer v, where v $\leq$ n - 1.
Step 3: Computes k_1 = hash(vG).
Step 4: Computes (k_2, k_3) = hash(vP_b).
Step 5: Uses the symmetric encryption algorithm to generate cipher text

$$c = E_{k_2}(m)$$

where the secret k_2 is generated in Step 4.
Step 6: Uses the one-way keyed hash function to generate

$$r = KH_{k_3}(c||k_1||ID_A||ID_B)$$

where ID_A and ID_B are the identifications given by the certificate authority(CA).
Step 7: Computes

$$s = \frac{v}{r + v_a} \text{ mod q}$$

Step 8: Compute T = rG.
Step 9: Sends the signcrypted text (c, T, s) to Bob.

Unsigncryption phase:

Bob receives the signcrypted text (c, T, s). He decrypts cipher text 'c' by performing symmetric decryption algorithm with secret key k. He also verifies the signature. Bob gets the plain text as follows.

Step 1: Verifies Alice's public key P_a by using her certificate.
Step 2: Computes k_1 = hash(sT + sP_a).
Step 3: Computes (k_2,k_3) = hash(v_bsT + v_bsP_a).
Step 4: Uses the one-way keyed hash function to generate

$$r = KH_{k_3}(c||k_1||ID_A||ID_B)$$

where ID_A and ID_B are the identifications given by the certification authority(CA).
Step 5: Uses a symmetric decryption algorithm to generate plain text

$$m = D_{k_2}(c)$$

where the secret key k_2 is computed in Step 3.
Step 6: Bob accepts the message 'm' only when rG = T .Otherwise he rejects.

Verification of the signcrypted message by a firewall or judge.

k_1=hash(sT+sP_a).
r=$KH_{k_2}(c||k_1||ID_a||ID_b)$.
Accept m if and only if

$$rG = T$$

Proof:

To proof the verification condition.

$$\begin{aligned}
v_b sT + sv_b P_a &= v_b\left(\frac{v}{r+v_a}\right)rG + \left(\frac{v}{r+v_a}\right)v_b P_a \\
&= \frac{v_b vrG}{r+v_a} + \frac{vv_b P_a}{r+v_a} \\
&= \frac{P_b vr}{r+v_a} + \frac{vv_b G v_a}{r+v_a} \\
&= \frac{vP_b r}{r+v_a} + \frac{vP_b v_a}{r+v_a} \\
&= vP_b\left(\frac{r+v_a}{r+v_a}\right) \\
&= vP_b
\end{aligned}$$

To proof the decryption stage

$$\begin{aligned}
sT + sP_a &= \frac{vT}{r+v_a} + \frac{vP_a}{r+v_a} \\
&= \frac{vrG}{r+v_a} + \frac{vv_a G}{r+v_a} \\
&= \text{vG}\left(\frac{r+v_a}{r+v_a}\right) \\
&= vG
\end{aligned}$$

Hence it is proved.

4.1.2 Algorithm of the proposed scheme II

The parameters, defined for scheme I, are also remaining same for this scheme.

Signcryption phase:

Assume that Alice wants to send a message m to *Bob*. *Alice* generates digital signature (T, s) of message m and uses the symmetric encryption algorithm and secret key k to encrypt m. Let c be the cipher text. *Alice* generates the signcrypted text (c, T, s) in the following steps.

Step 1: Verifies *Bob*'s public key P_b by using his certificate.

Step 2: Randomly selects an integer v, where v $\leq$ n - 1.

Step 3: Computes (k_1, k_2) = hash(vP_b).

Step 4: Uses the symmetric encryption algorithm to generate cipher text

$$c = E_{k_1}(m)$$

where the secret k_1 is generated in Step 3.

Step 5: Uses the one-way keyed hash function to generate

$$r = KH_{k_2}(c||ID_A||ID_B)$$

where ID_A and ID_B are the identifications given by the certificate authority(CA).

Step 6: Computes

$$s = \frac{v - r}{v_a} \text{ mod q}$$

Step 7: Compute T = rG.

Step 8: Sends the signcrypted text (c, T, s) to Bob.

Unsigncryption phase:

Bob receives the signcrypted text (c, T, s). He decrypts cipher text 'c' by performing symmetric decryption algorithm with secret key k. He also verifies the signature. *Bob* gets the plain text as follows.

Step 1: Verifies *Alice*'s public key P_a by using her certificate.

Step 2: Computes (k_1,k_2) = hash(v_bT + v_bsP_a).

Step 3: Uses the one-way keyed hash function to generate

$$r = KH_{k_2}(c||ID_A||ID_B)$$

where ID_A and ID_B are the identifications given by the certification authority(CA).

Step 4: Uses a symmetric decryption algorithm to generate plain text

$$m = D_{k_1}(c)$$

where the secret key k_1 is computed in Step 2.

Step 5: *Bob* accepts the message 'm' only when rG = T .Otherwise he rejects.

Verification of the signcrypted message by a firewall or judge.

Using Diffie-Hellman key exchange protocol *Bob* will send his private key to the verifier or judge.

The verifier computes (k_1,k_2) = hash(v_bT + v_bsP_a) and then

$$r = KH_{k_2}(c||ID_A||ID_B)$$

where ID_A and ID_B are the identifications given by the certification authority(CA). Finally the verifier accepts the message 'm' only when rG = T .Otherwise he rejects.

Proof:

To proof the decryption stage.

$$\begin{aligned} v_bT + v_bsP_a &= v_b(rG + sP_a) \\ &= v_bG(r + sv_a) \\ &= P_b(r + \tfrac{v-r}{v_a}v_a) \\ &= vP_b \end{aligned}$$

Hence proved.

4.2 The security functions of the proposed schemes

Table 4.1 indicates the security features supported by existing signcryption schemes along with the proposed schemes. The proof is based on the fact that it is almost intractable to solve the elliptic curve discrete logarithmic problem (ECDLP) [8, 20]. We should choose the parameters in such a way that it will become infeasible for an eavesdropper to solve ECDLP.

Table 4.1: Comparison based on security properties

	Confidentiality	Integrity	Unforgeability	Non-repudiation	Forward secrecy	Public Verification
Proposed scheme I	**Yes**	**Yes**	**Yes**	**Directly**	**Yes**	**Yes**
Proposed scheme II	**Yes**	**Yes**	**Yes**	**Another protocol**	**Yes**	**Yes**
Zheng	Yes	Yes	Yes	Another protocol	No	No
Zheng and imai	Yes	Yes	Yes	Another protocol	No	No
Bao and Deng	Yes	Yes	Yes	Directly	No	Yes
Gamage et al.	Yes	Yes	Yes	Directly	No	Yes
Jung et al.	Yes	Yes	Yes	Another protocol	Yes	No

Confidentiality

To be secure, the information needs to be hidden from unauthorized access. To achieve this we must make the data non-intelligible to the interceptor/eavesdropper. This is called confidentiality. In this discussion let us consider *Eve* as the attacker/eavesdropper. In both the schemes, if *Eve* wants to derive the key k then he has to solve ECDLP. Suppose he has got *hash (m)* and he knows the seed value of the curve i.e. G which is public to all. Then it is quite infeasible for *Eve* to solve it.

Authenticity

In the proposed schemes I, the recipient and the judge can use the sender's public key P_a with its certificate to authenticate the validity of the sender. But in case of scheme II, the recipient *Bob* should establish a zero-knowledge interactive protocol or Diffi-Hellman key exchange protocol with the judge to send his private key, for the authentication of the message sent by *Alice*. When the recipient decrypted the cipher text c to get the plain text m, he can use the formula given below to authenticate the correctness of the received message. If the equation holds, the recipient is sure that the received message does not modify in the transmission process. Therefore, the proposed scheme provides the authentication of the sender's identity and the transmitted message.
rG = T.

Integrity

In our proposed schemes, the recipient can verify whether the received message is the original one that was sent by the sender. In the signcryption phase, the sender computes and sends (c, T, s) to the recipient. If the attacker changes the cipher text c to c' then by the property of *Random Oracle Model* [4] it is infeasible to obtain two messages which give the same digest [4].

Unforgeability

Dishonest *Bob* is the most powerful attacker to forge a signcrypted message, because he is the only person who knows the private key v_b which is required to directly verify a signcryption from *Alice*. Given a signcrypted text(c, T, s) *Bob* can use his private key

v_b to decrypt the cipher text c and obtain (m, T, s). As we know ECDSA is unforgeable against adaptive attack. Hence it is unforgeable.

Non-repudiation

The target of non-repudiation is to prevent *Alice* from denying the signcryption she sent. Unforgeability implies non-repudiation if there is no duplication of the signcrypted message. If the signcryption text is forgeable, *Alice* will have opportunity to deny. When dispute occurs between sender and recipient, the recipient can send the signcrypted message to the judge for settling the original message M sent by the intended sender or not. Judge now run the verification algorithm and take the necessary action.

Forward Secrecy

An adversary that obtains v_a will not be able to decrypt past messages. Previously recorded values of (c, T, s) that were obtained before the compromise cannot be decrypted because the adversary that has v_a will need to calculate r to decrypt. Calculating r requires solving the ECDLP on T, which is computationally infeasible [8].

Public Verifiability

Verification requires knowing only Alice's public key. All public keys are assumed to be available to all system users through a certification authority or a public directly. For scheme I, the receiver of the message does not need to engage in a zero-knowledge proof communication with a judge or to provide to prove where as for scheme II, a zero knowledge key exchange protocol is needed.

4.3 Implementation of the proposed algorithms using JAVA

JAVA being an Object Oriented Language(OOL) includes the security packages. It could accept a prime number up to 1024 number of bits or even more. That could help us to generate large prime numbers. The following code is being used to generate the prime number of size depending upon the value of key size.

SecureRandom rndm1 = new SecureRandom();

BigInteger P_a *= BigInteger.probablePrime(keysize,rndm1);*

In our thesis work we have used the SHA-1, one way hash function, for the message digest. This compression function will give a fixed length output of 160 bits. Maximum message size that it takes is 2^{64}-1 and the block size is 512 bits.. Total 80 numbers of rounds has been used with a word size of 32 bits. The JAVA code used for the Secure Hash Function to make the digest of size 160 bits is given below:

```
public static String convertToHex(byte[] data) {
    StringBuffer buf = new StringBuffer();
    for (int i = 0; i < data.length; i++) {
            int halfbyte = (data[i] >>> 4) & 0x0F;
            int two_halfs = 0;
            do {
                if ((0 <= halfbyte) && (halfbyte <= 9))
                    buf.append((char) ('0' + halfbyte));
                else
                    buf.append((char) ('a' + (halfbyte - 10)));
                halfbyte = data[i] & 0x0F;
            } while(two_halfs++ < 1);
    }

    return buf.toString();
}

public static String SHA1(String text)
throws NoSuchAlgorithmException, UnsupportedEncodingException  {
    MessageDigest md;
    md = MessageDigest.getInstance("SHA-1");
    byte[] sha1hash = new byte[40];
    md.update(text.getBytes("iso-8859-1"), 0, text.length());
    sha1hash = md.digest();
    return convertToHex(sha1hash);
}
```

The program will generate the points on the elliptic curve depending upon the parameters chosen for the curve. It takes a string as input then signcrypt the message then generates (c, T, s) and send it to the recipient. The unsigncryption of the message is carried out by the receiver at the other end. The receiver accepts the string after checking the authenticity of the sender. A snapshot of the output is given below:

4.4 Cost Analysis

The cost which is being invested in the process of signcryption by the sender and in the process of unsigncryption by the receiver for both the proposed schemes are analyzed in this section. And also we will compare the cost with the existing schemes.

```
Output - JavaApplication2 (run-single)
init:
deps-jar:
Compiling 1 source file to C:\Documents and Settings\NIT User\JavaApplication2\build\classes
compile-single:
run-single:
Alice's p and q Pa =1074929665293619068134772007097375159214587207756586035447845919857922353680264112541437813991187782785504375094196236329984870758265091049989
Na= Pa*Qa = 13307273900128583708086618419520439562282022382936669954196602100086756592644597581880301895671517947611611113277480832639249708649643652075506569236 3
Alice's public key Ea = 8320083038031317991400866963613817319251287005687508475336339587188102922884292187887124219630872286895922862744713514938006793573153177 36
And private key Da= 7099905698141414143985664175365175077408317507314065828550474880821408352957958314236332095344504828414848737905864697786831060209145049639049
BOB's p and q Pb =1219547342903302365158985356997675133691654926095602657830298051880776005593805034167734763229842069548437537478018445102969296254563854838426 60
Nb= Pb*Qb = 14483744521483929618678848405674241150422475714548659159288897196706211069196526767762681426271907180750786446402106583724320646722569648561417458518 5
BOB's public key Eb = 11223937513255578441443106099224225329898259217486442776446656325752985804212347573675451838920994502778827073983153061799626355155220943337
And private key Db= 14208690462819882318875654644068353435789861863198438391002627010638796438353060783617998357366551262158275427802792458349965450899930387215 69
Enter the message
NIT rourkela
The msg u have entered = NIT rourkela
Generating r
value of r = 73657436179909948968665837005163582642235576193 9
R in binary 1000000100000101001000101111111110010001101110000010101111001001000111111010110011000000110001110010100010111110100100011010010010000100101111110100110
message in binary format100111001001001010101000010000001110010011011110111010101110010011010110110010101101100011 00001
 m||r in multiple of bytes 0100111001001001010101000010000001110010011011110111010101110010011010110110010101101100011000011000001100000010000010100100010111111110010001
Biginteger value of (m||r) = 354099626466864768370127888425656648246924283764048994352376775554746316623 55
No. of bits in the message (m||r) = 256
Hash of m||r = d9066023205ffdc5ee8990a9e132556964e9435e
biginteger value of hash(m||r) = 1238993176472330362912045326973852898017472496478
No. of bits in the above string is = 160
value of b=72372036919266393634441808797204967393039496971255363356969246249308563160732 and in binary10100000000000001000111110000110011001001100001011111110000100
M||R = 1001110010010010101010000100000011100100110111101110101011100100110101101100101011011000110 0001
[B@13e205f
m||r in unsign = 100111001001001010101000010000001110010011011110111010101110010011010110110010101101100011000011000001100000010000010100100010111111110010001101110000 01
Hash = d9066023205ffdc5ee8990a9e132556964e9435e
String is ACCEPTED!!!!!!
BUILD SUCCESSFUL (total time: 8 seconds)
```

Figure 4.1: Snapshot of the output of scheme I

Output - JavaApplication2 (run-single)

```
Enter the value of a for the equation (x)3+a(x)+b
3
Enter the value of b for the equation (x)3+a(x)+b
6
The values of p  = 967
The values of a  = 3
The values of b  = 6
not singular
the private key of Alice is va = 0
in key_generation For ALICE count = 1014 s[seed] = 21
The public key of alice pa =  164
the private key of Bob is db = 9
The public key of BOB pb =  238
in sign  v = 73 vpb =134
key = 4f330bdc75aa95b8f6e5cc765d24b52835fa6dff
 in sign k1 = 374008590781865969383141  and k2 = cc765d24b52835fa6dff
in key_generation_by_bob key k1  = 374008590781865969383141 e2 = 64
Enter the message now
123456789
in encrypt value of C1 = 324 value of C2 = 269
in KH sha of s = c2e454c0f8ef081f645a0ebd1cf50be3971085cf
in KH copy = 1112636162542212233108659558780927489128388396495
in sign  r = 1112636162542212233108659558780927489128388396495  ss = 40
k1k2 = 4f330bdc75aa95b8f6e5cc765d24b52835fa6dff
 un sign k1 = 4f330bdc75aa95b8f6e5  and k2 = cc765d24b52835fa6dff
 K1 in unsign = 374008590781865969383141
value of p in decrytpt 967 value of k1 in decrypt = 374008590781865969383141
in decrypt MSG = 287   msg =  287
String is ACCEPTED!!!!!!287    287
in KH sha of s = c2e454c0f8ef081f645a0ebd1cf50be3971085cf
in KH copy = 1112636162542212233108659558780927489128388396495
 tri = 1112636162542212233108659558780927489128388396495 and r = 11126361625422122331086595
!!!!! STRING IS ACCEPTED !!!!!
!!!!!! END OF PROCEDURE !!!!!!!
BUILD SUCCESSFUL (total time: 25 seconds)
```

Figure 4.2: Snapshot of the output of scheme II

Table 4.2: Elliptic curve DSS and its Variants

shortened schemes	Signature(r,s) on a message m	Verification of signature	Length of signature
ECDSS	r = vG mod q s = $\frac{hash(m)+v_a r}{v}$ mod q	K=$\acute{s}$(hash(m)G+rP_a) where $\acute{s}$= $\frac{1}{s}$ mod q, check whether K mod q = r	$2\|q\|$
SECDSS1	r = hash(vG,m) s = $\frac{v}{r+v_a}$ mod q	K = s(P_a + rG) check whether hash(K,m) = r	$\|hash(.)\| + \|q\|$
SECDSS2	r = hash(vG,m) s = $\frac{v}{1+rv_a}$ mod q	K = s(rP_a + G) check whether hash(K,m) = r	$\|hash(.)\| + \|q\|$

4.4.1 Saving in Computational cost

Table 4.2 taken from [1] shows the computational cost and the communication overhead of various elliptic curve DSS. With the signature-then-encryption based on SECDSS1 or SECDSS2 and elliptic curve ElGamal encryption [1], the number of computations of multiples of points is 3, both for the process of signature-then-encryption and that of decryption-then-verification.

Table 4.3: Comparative analysis of computational overhead

Signcryption scheme	Participants	EXP	DIV	ECPM	ECPA	MUL	ADD	KH(.)/ hash(.)
Zheng	Alice	1	1	-	-	-	1	2
	Bob	2	-	-	-	2	-	2
Jung et al.	Alice	2	1	-	-	-	1	2
	Bob	3	-	-	-	1	-	2
Bao and Deng	Alice	2	1	-	-	-	1	3
	Bob	3	-	-	-	1	-	3
Gamage et al.	Alice	2	1	-	-	-	1	2
	Bob	3	-	-	-	1	-	2
Zheng and Imai	Alice	-	1	1	-	1	1	2
	Bob	-	-	2	1	2	-	2
Han et al.	Alice	-	1	2	-	2	1	2
	Bob	-	1	3	1	2	-	2
Hwang	Alice	-	-	2	-	1	1	1
	Bob	-	-	3	1	-	-	1
Proposed scheme 1	**Alice**	**-**	**1**	**3**	**-**	**1**	**1**	**3**
	Bob	**-**	**-**	**2**	**1**	**-**	**-**	**3**
Proposed scheme 2	**Alice**	**-**	**1**	**2**	**-**	**1**	**1**	**2**
	Bob	**-**	**-**	**2**	**1**	**-**	**-**	**2**

We note that the "*square and multiply*" method for fast exponentiation (see Appendix B) can be adapted to a "*doubling and addition*" method for the fast computation of a multiple of a point on an elliptic curve. Namely a multiple can be obtained in about $1.5|q|$ point additions [12, 18].

Among the three multiples for decryption-then-verification, two are used in verifying a signature. More specifically these two multiples are spent in computing $e_1 G + e_2 P_a$ for two integer's e_1 and e_2. Shamir's technique [18] for fast computation of the product of multiple exponentials with the same modulo can be adapted to the fast computation of $e_1G + e_2P_a$. Thus on average the computational cost for $e_1G + e_2P_a$ is (1+3/4)$|q|$ point additions, or equivalently 1.17 point multiples. That is, the number of point multiples involved in decryption-then-verification, can be reduced from 3 to 2.17.

For scheme I and II:

In the proposed scheme I, we try to reduce the sender's computational cost. Table 4.3 shows the comparisons of computational cost of sender and recipient among our signcryption schemes and others [10]. The proposed scheme I require only 3 ECPM for

Table 4.4: comparative analysis of different schemes on the average computational time of major operations

Various schemes	Sender Average computational time in ms	Recipient Average computational time in ms
Scheme I	**3×83=249**	**2×83=166**
Scheme II	**2×83=166**	**2×83=166**
Zheng [1]	1×220=220	2×220=440
Zheng and Imai [18]	1×83=83	2×83=166
Bao and Deng [16]	2×220=440	3×220=660
Gamage et al. [17]	2×220=440	3×220=660
Jung et al.	2×220=440	3×220=660
Han et al. [11]	2×83=166	3×83=249
Hwang et al. [10]	2×83=166	3×83=249

signcryption and 2 ECPM for unsigncryption where as scheme II takes 2 ECPM for signcryption and 2 ECPM for unsigncryption. The elliptic curve point multiplication needs 83 ms and the modular exponentiation operation needs 220 ms for average computational time in the Infineon's SLE66CUX640P security controller [23]. Although our schemes are slower than Zheng and Imai scheme [18] as shown in the Table 4.4 but they provide added functionality such as forward secrecy, public verifiabilty. Figure 4.3 shows the comparative analysis of the proposed schemes with the existing schemes. From this we may conclude that the proposed schemes give better result than all other schemes except the schemes such as Hwang et al., Zheng and Imai. Scheme II gives better result as compare to Hwang et al. scheme.

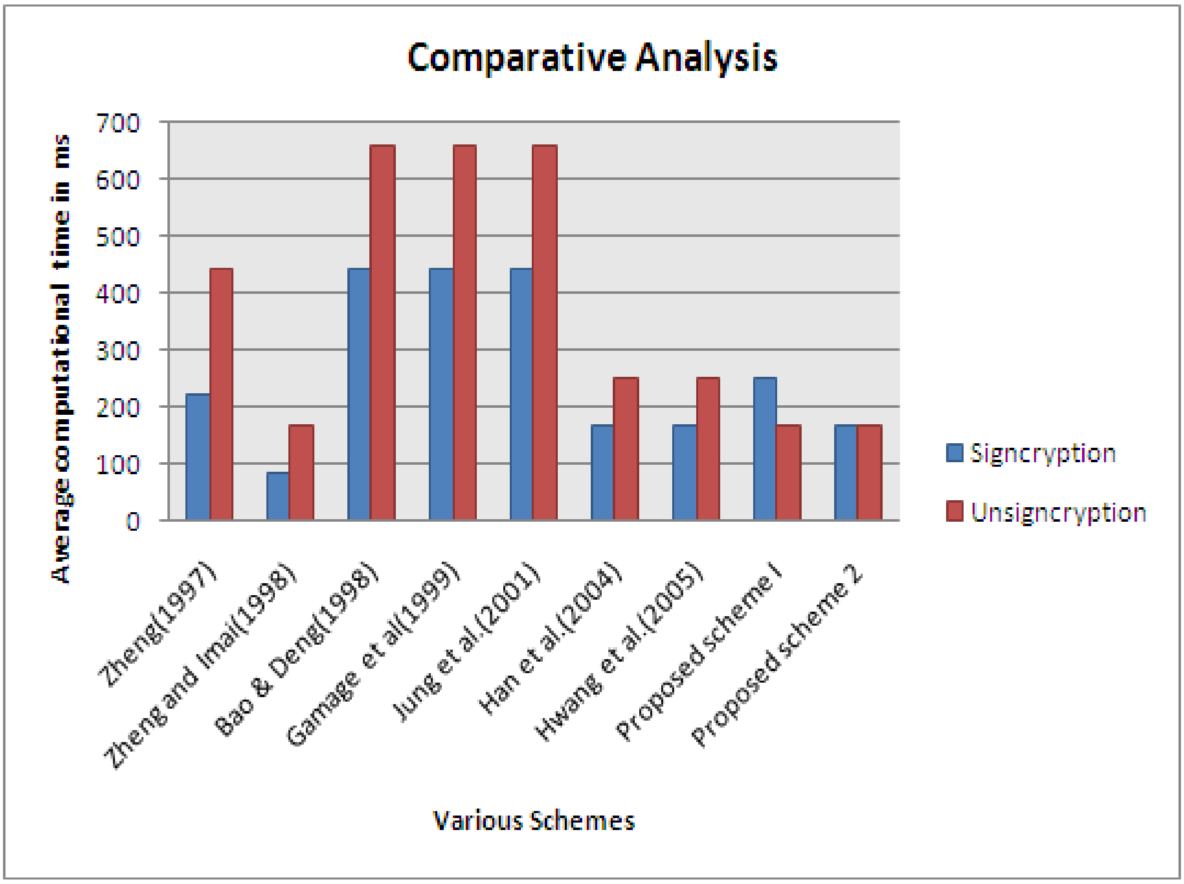

Figure 4.3: Performance

4.4.2 Analysis of Communication overhead

Communication overhead calculations are based on the following assumptions:

a)

$$|hash(.)| = |KH(.)| = |q| \div 2$$

b)

$$|q| \approx |p^m|$$

c) Point compression is used.

The communication overhead of SECDSS1 (see Table 4.2) followed by ElGamal elliptic curve encryption is [4],

$$(|hash(.)| + |q|) + 2(|q| + 1) = |hash(.)| + 3\,|q| = 3.5\,|q|$$

assuming that

$$|q| \gg 1$$

For Scheme I and II:

The communication overhead of the proposed schemes I and II are

$$|q| + (|q| + 1) = 2\,|q| + 1 \approx 2\,|q|$$

assuming that

$$|q| \gg 1$$

Thus bandwidth saving can be calculated as:

$$(3.5\,|q| - 2\,|q|)/3.5\,|q| = 42.86\%$$

This saving is higher than the one calculated in Zheng-Imai paper [18], which is 40%. However, it supports forward secrecy.

Chapter 5

Conclusion

Conclusion
Limitations of the work
Further development

Chapter 5

Conclusion

This thesis introduces two elliptic curve based signcryption schemes for secure and authenticated message delivery, which fulfills all the functions of digital signature and encryption with a cost less than that required by the current standard signature-then-encryption method. The Zheng and Imai scheme discussed in Chapter 2 is the most efficient signcryption scheme based on ECC. But the drawback of the above scheme is that it does not provide forward secrecy. So it is necessary to provide forward secrecy. There are few schemes which can provide forward secrecy but the computational cost and communication overhead is more. The cost of the proposed schemes are comparatively lower than other schemes in terms of computational and communication overhead. ECC has been used for the implementation of our algorithm because of its unique property of ECDLP which is significantly more difficult than either the IFP or DLP. Proposed schemes save more computational cost for the sender to suit the application of restricted computational devices like smart card based applications, mobile devices, etc.

5.1 Limitations of the work

The elliptic curve parameters must be chosen in such a way that it will be difficult for an adversary to solve the elliptic curve discrete logarithmic problem(ECDLP). Apart from it scheme II, proposed in this thesis, requires a zero-knowledge interactive protocol to verify the authenticity of the sender by a third party or judge.

5.2 Further development

In 1988 Koblitz [24] suggested to use the generalization of Elliptic Curves (EC) for cryptography, the so-called Hyperelliptic Curves (HEC). While ECC applications are highly developed in practice, the use of HEC is still of pure academic interest. However, one advantage of HECC [24] resides on the fact that the operand size for HECC is at least a factor of two smaller than the one of ECC. More precisely, while typical bit-lengths for ECC are at least 160 bits, for HECC this lower bound is around 80 bits (in the case of genus 2 curves). This fact makes HECC a very good choice for platforms with limited resources. Now we should look forward to develop schemes based on HECC which is an open challenge for us.

Bibliography

[1] Yuliang Zheng. Digital signcryption or how to achieve cost(signature encryption) $\ll$ cost(signature) + cost(encryption). In *CRYPTO '97: Proceedings of the 17th Annual International Cryptology Conference on Advances in Cryptology*, pages 165–179, London, UK, 1997. Springer-Verlag.

[2] William Stallings. *Cryptography and Network security: Principles and Practices.* Prentice Hall Inc., second edition, 1999.

[3] Paul C. van Oorschot Alfred J. Menezes and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[4] Behrouz A. Forouzan. *Cryptography and Network Security.* Tata McGraw-Hill, 2007.

[5] Varad Kirtane and C. Pandu Rangan. Rsa-tbos signcryption with proxy re-encryption. In *DRM '08: Proceedings of the 8th ACM workshop on Digital rights management*, pages 59–66, New York, NY, USA, 2008. ACM.

[6] Yuliang Zheng Joonsang Baek, Ron Steinfeld. Formal proofs for the security of signcryption. *Journal of Cryptology*, 20(2):203–235, 2007.

[7] William J Caelli, Edward P Dawson, and Scott A Rea. Pki, elliptic curve cryptography, and digital signatures. *Computers and Security*, 18(1):47 – 66, 1999.

[8] Lawrence C. Washington. *,Elliptic Curves: Number Theory and Cryptography.* CRC Press, 2003.

[9] Surya A. Effendi R. Sutikno, S. An implementation of elgamal elliptic curves cryptosystems. pages 483 –486, Nov 1998.

[10] Ren-Junn Hwang, Chih-Hua Lai, and Feng-Fu Su. An efficient signcryption scheme with forward secrecy based on elliptic curve. *Applied Mathematics and Computation*, 167(2):870 – 881, 2005.

[11] X. Yang Y. Han and Y. Hu. Signcryption based on elliptic curve and its multi-party schemes. *Proceedings of the 3rd ACM International Conference on Information Security (InfoSecu 04)*, pages 216–217, 2004.

[12] Ram Shanmugam. Elliptic curves and their applications to cryptography: An introduction : Andreas enge, kluwer academic press, norwell, ma, 1999, pp. 164. isbn 0-7923-8589-6. *Neurocomputing*, 41(1-4):193 – 193, 2001.

[13] Scott A. Vanstone. Elliptic curve cryptosystem – the answer to strong, fast public-key cryptography for securing constrained environments. *Information Security Technical Report*, 2(2):78 – 87, 1997.

[14] Johan Borst, Bart Preneel, and Vincent Rijmen. Cryptography on smart cards. *Computer Networks*, 36(4):423 – 435, 2001.

[15] Mohsen Toorani and Ali Asghar Beheshti Shirazi. Cryptanalysis of an efficient signcryption scheme with forward secrecy based on elliptic curve. *Computer and Electrical Engineering, International Conference on*, 0:428–432, 2008.

[16] R.H. Deng F. Bao. A signcryption scheme with signature directly verifiable by public key. *Proceedings of PKC'98 LNCS 1431*, pages 55–59, 1998.

[17] Ana Gamage, Jussipekka Leiwo, and Yuliang Zheng. Encrypted message authentication by firewalls. In *Proc. of PKC99, LNCS 1560*, pages 69–81. Springer-Verlag, 1999.

[18] Yuliang Zheng and Hideki Imai. How to construct efficient signcryption schemes on elliptic curves. *Inf. Process. Lett.*, 68(5):227–233, 1998.

[19] Mohsen Toorani and Ali Asghar Beheshti Shirazi. An elliptic curve-based signcryption scheme with forward secrecy. *Journal of Applied Sciences*, 9(6):1025–1035, 2009.

[20] G. Seroussi. Elliptic curve cryptography. page 41, 1999.

[21] Tom St Denis and Simon Johnson. Hash functions. In *Cryptography for Developers*, pages 203 – 250. Syngress, Burlington, 2006.

[22] Chik How Tan. Analysis of improved signcryption scheme with key privacy. *Information Processing Letters*, 99(4):135 – 138, 2006.

[23] Lejla Batina, SIddIka Berna rs, Bart Preneel, and Joos Vandewalle. Hardware architectures for public key cryptography. *Integration, the VLSI Journal*, 34(1-2):1 – 64, 2003.

[24] Henri Cohen and Gerhard Frey, editors. *Handbook of elliptic and hyperelliptic curve cryptography.* CRC Press, 2005.

[25] Bart Preneel. A survey of recent developments in cryptographic algorithms for smart cards. *Computer Networks*, 51(9):2223 – 2233, 2007. (1) Advances in Smart Cards and (2) Topics in Wireless Broadband Systems.

[26] Alain Trottier. *Java 2 Core Language Little Black Book.* Paraglyph Press, 2002.

[27] Behrouz A. Forouzan. *JAVA.* Tata McGraw-Hill, 2007.

[28] `http://www.signcryption.org/publications/`.

[29] Yuliang Zheng. Signcryption and its applications in efficient public key solutions. In *Information Security: Proceedings of 1997 Information Security Workshop(ISW'97)*, volume 1396, pages 291–312, Berlin, New York, Tokyo, 1998. Springer-Verlag.

[30] Guilin Wang, Feng Bao, Changshe Ma, and Kefei Chen. Efficient authenticated encryption schemes with public verifiability, 2005.

[31] Kiyomichi Araki, Takakazu Satoh, and Shinji Miura. Overview of elliptic curve cryptography. In *PKC '98: Proceedings of the First International Workshop on Practice and Theory in Public Key Cryptography*, pages 29–49, London, UK, 1998. Springer-Verlag.

[32] Julio Lpez, Ricardo Dahab, and Ricardo Dahab. An overview of elliptic curve cryptography. Technical report, 2000.

[33] Jorge Guajardo and Department Head. Efficient algorithms for elliptic curve cryptosystems, 1997.

[34] Penzhorn W.T. Botes J.J. Public-key cryptosystems based on elliptic curves. pages 1 –5, Aug 1993.

[35] Espinosa Garcia J. Hernandez Encinas L. Gayoso Martinez V., Sanchez Avila C. Elliptic curve cryptography: Java implementation issues. pages 238 – 241, Oct. 2005.

[36] Yu Fang Chung, Zhen Yu Wu, and Tzer Shyong Chen. Ring signature scheme for ecc-based anonymous signcryption. *Computer Standards and Interfaces*, 31(4):669 – 674, 2009.

[37] Herbert Schildt. *JAVA: The Complete Reference*. Tata McGraw-Hill, seventh edition, 2008.

[38] W. Diffie and M. Hellman. New directions in cryptography. In *IEEE Transactions on Information Theory*, volume IT-22, pages 472–492, 1976.

Dissemination of Work

[1] **Ramesh Kumar Mohapatra** and Banshidhar Majhi. A modified signcryption scheme with forward secrecy based on ECC. *In ICACC'2010: Proceedings of the first international conference on advance computing and communication,* pages 176–179, Kerala, India.

[2] **Ramesh Kumar Mohapatra** and Banshidhar Majhi. A New Forward-Secure Signcryption Scheme Based on ECC. In the *First National Conference on Networking and Information Security(NCNIS'10)*, Sikkim, India.

Appendix A

Weierstrass Equation

An elliptic curve is the graph of an equation of the form

$y^2 \quad = \quad x^3 + Ax + B$ Where A and B are constants. This will be reffered to as the *Weierstrass Equation* for an elliptic curve shown in the Figure 5.1. The set of points it includes is given by,

$$E(L) = \{\infty\} \cup \{(x, y) \in L \times L | y^2 = x^3 + Ax + B$$

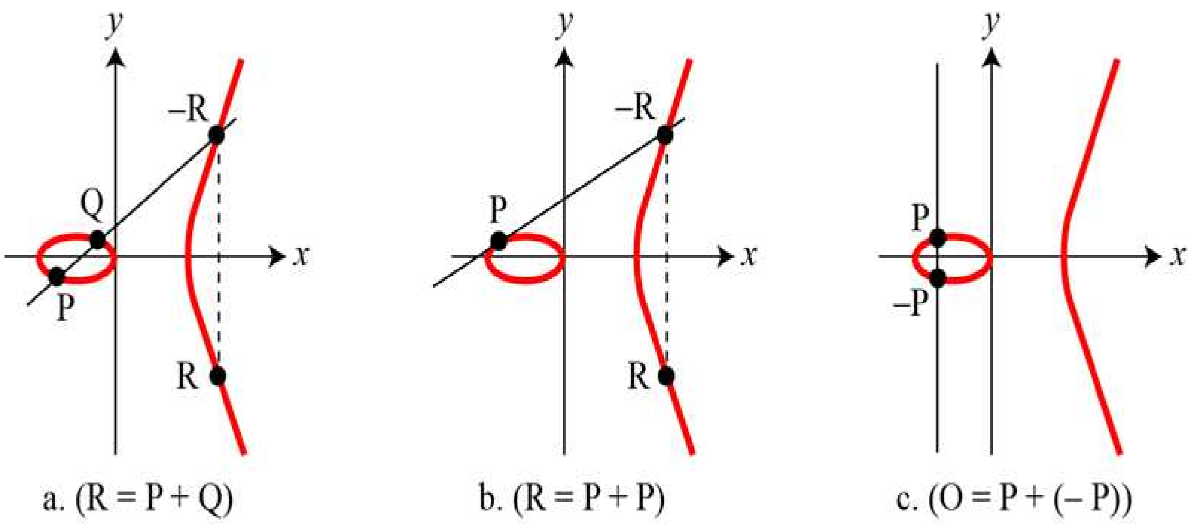

Figure 5.1: Three adding cases in an elliptic curve

Adding points on an elliptic curve:
Let P = (x_1, y_1) and Q = (x_2, y_2) are the two points on an elliptic curve as shown in the Figure 5.1. Now we need to find out a point R such that R = P + Q.

Case 1:

Assuming that P$\neq$Q and neither one is ∞, draw a line joining P and Q and it should cut the curve at some point let$'$s say R$'$= (x_3', y_3').

The slope of the line will be $\frac{y_2-y_1}{x_2-x_1}$. If $x_1 = x_2$ then the line will be a vertical line. Let us assume that $x_1 \neq x_2$.

The equation of the line is $y \;=\; m(x - x_1) + y_1$ To find the intersection point with the curve E, the must satisfy the curve condition. Substitute the above equation in the curve we will get

$$\begin{aligned} y^2 &= x^3 + Ax + B \\ (m\,(x - x_1) + y_1)^2 &= x^3 + Ax + B \end{aligned}$$

After rearranging the above equation we will get

$$0 = x^3 - m^2x^2 + ...$$

As we know that,

$(x - r)(x - s)(x - t) = x^3 - (r + s + t)x^2 + (rs + st + rt)x - rst$

where r, s, t are the roots of that equation.

Therefore we may conclude that r + s + t = m^2. It implies $x_3' + x_1 + x_2 = m^2$

x_3'= $m^2 - x_1 - x_2$

and y_3' = m$(x - x_1)$ + y_1

Now reflect the point about the x-axis we will get R(x_3, y_3)=R$'(x_3', -y_3')$.

$$\begin{aligned} x_3 &= x_3' = m^2 - x_1 - x_2 \\ y_3 &= -y_3' = -[m(x_3 - x_2) + y_1] \\ &= m(x_1 - x_3) - y_1 \end{aligned}$$

$$m = \frac{y_2-y_1}{x_2-x_1}$$

Therefore, $$\begin{aligned} x_3 &= m^2 - x_1 - x_2 \\ y_3 &= m(x_1 - x_3) - y_1 \end{aligned}$$

Case 2:

When the two points coincides as shown in the Figure 5.1(b), we take the line through them to be tangent line. Implicit differentiation allow us to find out the slope of that line,

$$\begin{aligned} y^2 &= x^3 + Ax + B \\ 2y\tfrac{dy}{dx} &= 3x^2 + A \\ \tfrac{dy}{dx} &= \tfrac{3x^2+A}{2y} \end{aligned}$$

Hence slope m $=(\frac{3x^2+A}{2y})$

$x_3 = m^2 - 2x_1$

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

Case 3:

If the two points are additive inverse of each other means P = -P as shown in the Figure 5.1(c) then the line connecting the two points does not intercept the curve at a third point. It will meet the curve at infinity that′s why P+(-P) = ∞.

Appendix B

Fast computation of the Product of Multiple exponentials with the same modulo

In unsigncryption, the most expensive part of computation by $g_0^{e_0} g_1^{e_1}$ mod p, where g_0, g_1 ,e_0, e_1 and p are all large integers. According to A. Shamir, the product involving the same modulo can be obtained with a smaller computational cost(see [3] page no. 618) by using the following algorithm.

Algorithm Simultaneous multiple exponentiation

INPUT: group elements $g_0, g_1, \ldots, g_{k-1}$ and non-negative t-bit integers $e_0, e_1, \ldots e_{k-1}$.
OUTPUT: $g_0^{e_0} g_1^{e_1} \cdots g_{k-1}^{e_{k-1}}$.

1. *Precomputation*. For i from 0 to $(2^k - 1)$: $G_i \leftarrow \prod_{j=0}^{k-1} g_j^{i_j}$ where $i = (i_{k-1} \cdots i_0)_2$.
2. $A \leftarrow 1$.
3. For i from 1 to t do the following: $A \leftarrow A \cdot A$, $A \leftarrow A \cdot G_{I_i}$.
4. Return(A).

Let us analyze the computational complexity of the above algorithm. For a small value of k, say k $\leq$ 4, the computational cost for pre-computing $G_1, G_2, ..., G_{2^k-1}$ is marginal when compared to the total cost for computing the product. The total computational cost is dominated by (t+v) modulo multiplication where v is the number of non-zero columns in the array. For $e_0, e_1, ..., e_{k-1}$ chosen randomly then $\left(\left(\frac{1}{2}\right)^k\right) t$ of the columns in the array are zeros. Thus, the expected number of modulo multiplications is $\left(2 - \left(\frac{1}{2}\right)^k\right) t$.

For k = 2, the expected computational cost is 1.75t modulo multiplication which is roughly equivalent to 1.17 modulo exponentiations when the standard *square and multiply* method is used.

Bio-Data

Ramesh Kumar Mohapatra

Date of Birth: 21st February 1981

Address for Correspondence

Department of Computer Science and Engineering,

National Institute of Technology Rourkela,

Rourkela, Orissa, India-769008.

Ph — +919438330083

e-mail — rkmohapatraiter@yahoo.co.in

Qualification

- M.Tech (CSE) [Continuing]
  NIT Rourkela, Orissa.
- B.E (CSE) [First division]
  ITER, Bhubaneswar, Orissa.
- +2 (Science) [First division]
  CHSE, Bhubaneswar, Orissa.
- 10th [First division]
  BSE, Bhubaneswar, Orissa.

Professional Experience

- Lecturer, ITER, Bhubaneswar, Orissa From May'2007 to July'2008.
- Lecturer DRIEMS, Cuttack, Orissa From August'2004 to May'2007.

Publications

- 02 Conference Articles

Permanent Address

Plot No. 388/389, Nayapalli,

Bhubaneswar, Orissa, India,

PIN-751012