Investigation of the Efficiency of the Elliptic Curve Cryptosystem for Multi-application Smart Card

HUANG MENG YUAN
B.Eng.Sc. (Electronic and Communication)

A thesis submitted for the degree of
Master Engineering Science

Faculty of Electrical and Electronic Engineering
University of Adelaide

August 1998

To my wife Qin Mei-zhen,
my parents and my loving family.

Abstract

Public key cryptography algorithms offer the potential to be the most secure algorithms for smart card. Currently, several attempts have been made to find practical public key systems [1, 2, 3] based on the difficulty of factoring large integers, such as the Rivest-Shamir-Adleman (RSA) cryptosystem, or based on the difficulty of solving discrete logarithms over finite fields, such as the ElGamal cryptosystem.

Recently Odlyzko [4] has forecast that a 512 bit module will be vulnerable to factorization in a couple of years and is therefore not suitable on a long-term basis for security protection. It is likely that a 1024 bit RSA will become common in the near future. Though it will probably remain secure for many years, it requires too much memory for the smaller size chip required for the Multi-application Smart Card (**MSC**). There has also been recent progress in computing discrete logarithms over finite fields, but the requirement of the ElGamal cryptosystem in terms of memory capacity is the same as the RSA cryptosystem, which can provide security for longer periods, but then requires larger memory space.

For the development of the multi-application smart card in future, the memory capacity of the chip is not big enough to store larger and more complex programs needed for the multi-application operating system and the protocol codes of the cryptography algorithm involving large integers. This is because the chip on the smart card is restricted in size. However, the advantage of the Elliptic Curve Cryptosystem (**ECCs**) is that it provides equivalent security to existing public key schemes but with much shorter key lengths. A small memory requirement is a crucial factor in the design of Smart Cards [5] and will be significant in the design of the multi-application smart card.

This thesis considers the efficiency of the Elliptic Curve Cryptography (**ECC**) in the design of the MSC in future and describes the problems of the smaller memory

requirement in public key cryptosystems for the MSC. The ElGamal and Elliptic curve algorithms will be compared, where the modeling of these Cryptosystems will clearly show the source of the efficiency of the Elliptic Curve Cryptography Algorithm (**ECCA**) as a basis for achieving the required processing efficiency for future application.

This thesis also shows the lengthy time required for the operation of the Elliptic Curve algorithms, which is the main cause of resistance to the ECCs, and which prevents it from being put into practice. The thesis also compares the running time of the three kinds of public key cryptosystem. Therefore, it also indicates the direction for research and further development of the ECCs.

Declaration

This work described in this thesis neither contains any material, which has been accepted by any other degree or diploma in any University or Institution nor contains any material previously published by other professional persons, except those references that have been made in the text.

I will give my permission to the copy of this thesis, when lodged into the Library in University of Adelaide, and also available for loading and photocopying.

Huang Meng yuan

University of Adelaide, Australia
August 15, 1998

Signature:

Data: August 20, 1998

Conference Publication

M. Y. Huang, ***"Investigation of the Efficiency of the Elliptic Curve Cryptosystem For Multi-applications Smart Card"***, Second International Conference on Knowledge-Based Intelligent Electronic Systems, Adelaide, Australia. April 21 ~ 23, 1998.

Acknowledgments

I am very grateful to my family for their help, especially to my wife Mrs. Qin Mei Zhen for her support and moral encouragement during the whole period of my study. She undertakes all of the family work, which ensured enough time for the completion of my study successfully. It would be difficult to imagine, it being possible without her support.

I would like to especially express my true thanks to my supervisor: Prof. Reginald Coutts for his financial help and moral encouragement during my two years studying at the Centre for Telecommunication Information Network (CTIN) in the University of Adelaide. His timely comments and rich experience in telecommunication and research is a major source of inspiration for the direction of my study.

I would like to thank all staff working and students for CTIN, who provide valuable references for my research, and assistance during my study. I would especially like to thank Ms. Collette Snowden for her valuable help with the use of English language.

I am very grateful to my parents who have deferred a visit to Australia to avoid any disturbance to my study. Their keen interest and devoted guidance have always inspired me to pursue my postgraduate course.

I would like to thank all of the persons and the institutions that have provided help and support to my study, and any other individuals who have enabled me to bring this thesis to completion.

List of Abbreviations

Afnor --- Association Franchise de Normalization

ATR --- Answer To Reset

CLK --- Clock

CMOS --- Complementary Metal-Oxide-Semiconductor

CPU --- Central Processing Unit

DF --- Dedicated File

DIR --- Directory

DOS --- Date Operating System

DS --- Directory Sturcture

ECC --- Elliptic Curve Cryptography

ECCA --- Elliptic Curve Cryptography Algorithm

ECCs --- Elliptic Curve Cryptosystem

ECSS --- Elliptic Curve Signature Scheme

EDC --- Error Detection Code

EEPROM --- Electrically Erasable Programmable Read-Only-Memory

EF --- Elementary File

EPROM --- Erasable Programmable Read-Only-Memory

etu --- Elementary Time Unit

GND --- Ground

INF --- Information Field

I/O --- Input / Output

ISO --- International Organization for Standardization

LEN --- Length

MF --- Master File

MSC --- Multi-application Smart Card

NAD --- Node Address Detection

NIST --- National Institute of Standards and technology

PCB --- Protocol Control Block

PrCA --- Privacy Key Cryptography Algorithm

PuCA --- Public Key Cryptography Algorithm

RAM --- Random Access Memory

RFU --- reserved for Future Use

ROM --- Read Only Memory

RSA --- Rives-Shamir-Adleman cryptosystem

RST --- Reset

SCOS --- Smart Card Operating System

SSC --- Single-application Smart Card

TPDU --- Transaction Protocol Data Unit

UV --- Ultraviolet

VCC --- Supply Voltage

VPP --- Programming Voltage

List of Symbols and Notation

E --- The Elliptic Curve

#E --- Number of points on the elliptic curve

exp --- exponentiation

F --- Finite field,

$\mathbf{F}_q$ --- Finite field with q elements,

$\mathbf{F}_p$ --- Finite field in order p,

#**F** --- Number of points over the finite field

gcd(...) --- Greatest common division

GF --- the Galois Field

K --- a finite prime field

ln --- Logarithm on base e

log --- Logarithm on base 10

mod --- module

o(1) --- zero function

O(...) --- Number of operation

P+Q --- Addition law on a cubic curve,

P*Q (P×Q) --- Multiplication law on a cubic curve,

T_r --- Trace function,

Z --- The field of integers,

$\mathbf{Z}_p$ --- The field of integers modulo a prime number p,

$\phi(...)$ --- Euler's totient function,

$\mathcal{O}$ --- Infinitive point (or zero element),

Π --- Production,

Σ --- Summation,

{...} --- Group,

$\lceil x \rceil$ --- The smallest integer greater than or equal to x,

$\lfloor x \rfloor$ --- The largest integer less than or equal to x,

$|$ --- Divisible,

$\nmid$ --- Indivisible,

$!$ --- Factorial,

$\| \|$ --- Size of a vector,

Contents

Preface

Smart card developments all over the world can be correlated with a rapid increase in both the scale and scope of smart card technologies. Although several multi-application smart cards (**MSC**) are emerging in France, Germany, Japan and USA, they are likely still in their infancy. The aim of this thesis is to explore the development and the achievements of the MSC in the future, and discuss the many technological problems that need to analyzed and solved. In particular the important issue of the limited memory capacity in the chip is a key problem to be overcome for the successful use of MSC.

The MSC in future has few specifications with many services in one card and the highest level of security which contain not only all of the controller security features, but a coprocessor that processes asymmetrically on chip security algorithms. These functions will be supported by a complex software system and will need a large memory storage capability. In order to ensure the performance of these software functions, the Smart Card Operating System will become more complex and larger, and the program code of the Operating System will also need a large memory storage capacity.

However, the chip on the smart card is restricted in size and although the industry can already produce 16 kbit ROM and 16 kbit EEPROM at present, and has already enhanced data storage and programmed service in optional code. The memory capacity of the chip is not big enough to store the larger and more complex programs for the multi-application operating system and the protocol codes necessary for the cryptography algorithm for factoring large integers. With this limited memory space, cryptographic keys will be stored in EEPROM, the ROM mask normally storing the operating system and higher level instructions, which execute cryptographic algorithms. It is therefore necessary to look for an efficient cryptographic algorithm which satisfies both the security level based on the factoring of large integers and the memory space of the restricted size chip in order to successfully develop the MSC.

An important fact is that the public key cryptosystems based on factoring algorithm or classical discrete logarithm algorithms would provide the most security for MSC, but their disadvantage is that the secure-term provided by the cryptosystems will be longer, and then the memory space required will be larger. However, the advantage of the Elliptic Curve Cryptosystem (**ECCs**) is that it potentially provides equivalent security to these two existing public key schemes but with much shorter key lengths. Smaller memory requirements for the ECCs will be necessary in order to achieve the maximum potential of the MSC in future.

Another important fact is that a lengthy running time will be required for the computational operation of the ECCs. This disadvantage is created by the complex architecture of computation of the ECCs and is a main barrier to prevent the ECCs becoming a practical alternative. One of the results in this thesis is that it clearly shows the efficiency of the smaller memory requirement trade off against the more lengthy running time.

Therefore, the investigation of the efficiency and running time of the ECCs in this thesis will be significant in determining the direction of ECCs research. It is worth noting that the choices and decisions made in relation to smart card technology development will determine the future of MSC.

Smart card development also relates to many other social and technological areas, such as communication systems, microelectronics, international standards, and privacy protection. However, this thesis only considers the efficiency and running time related to the use of smart card, and is divided into four parts. Chapter 1 is a description of the physical and logical architecture of future smart cards, which will affect the production and development of MSC in the future, and which specifically reflects the smaller memory capacity in the card. Chapter 2 discusses the three kinds of public key cryptosystems, and shows the three different types of cryptography algorithm and security functions. Chapter 3 introduces the mathematical principles corresponding to

three cryptosystems, the RSA, ElGamal and Elliptic Curve systems. Here only a minimum of knowledge of number theory is required to understand how the efficiency and the running times for the three different public key cryptosystems are determined. Chapter 4 compares the efficiency and running time for the three public key cryptosystems. The results show the advanced efficiency combining a smaller memory requirement with a high degree of security compared with other public key cryptosystems, but with the disadvantage of longer running times than 450 ms limitation. Finally the conclusion of this thesis outlines the direction of research involving the ECC, and predicts the result of future development of the MSC.

Huang Meng Yuan
University of Adelaide
August 19, 1998

Introduction

The higher degree security feature in the smart card will be a vital ingredient in creating and developing a Multi-applications Smart Card (MSC). People have recognized that information security will become a key issue for the smart card. Smart cards are already in wide public use and will affect many areas of our lives. To ensure confidentiality, privacy and security, cryptology is pervading our everyday life. It has a large influence on security in various fields of applications, not only in banking, but also in the areas of health, Pay TV, personal computer, education, employment, communication, ...etc.

There are a number of new security features in response to market demands. In most cases the new features will be equally useful to general cards and on-line systems. However, they are going to enhance smart card functions and capabilities. Some of the new techniques that are being developed are as follows:

(1) Personal identification verification

Three of the coming changes are Signature verification by signature dynamics, identification by retinal scanning and verification by hand geometry. These techniques and others will be tested and tried. An advantage biometrics technique will eventually emerge after several years of trial and use. The PIN number will be continuously used in existing environments and will be useful for many years as a supplement to the emerging technique.

(2) Encryption algorithm

As an introduction to security of the smart cards, it will firstly describe the status development of the smart cards. Then the status of cryptology for smart card, mainly the Public key Cryptography Algorithm (**PuCA**) will be discussed. The Privacy key Cryptography Algorithm (**PrCA**) will be neglected for terseness in this report. Finally it will briefly describe the status and applications of the Elliptic Curve Cryptography Algorithm (**ECCA**).

1.1 The status development of smart card

The smart card is a single chip microcomputer, which is a miniature computer system on a single piece of silicon. There are different devices and procedures compared to a personal computer. There are no keyboards, displays, disk drives, etc. outside the card. There is the built-in capability to prevent, by various means, unauthorized access to the CPU, the memories, the buses, and any data being stored or processed within the card at any time.

The smart card development all over the world can be correlated with a rapid increase in both the scale and scope. Currently, a large number of smart cards are produced and issued with a number of different applications in many countries, mostly applications involving electronic money transfer and identification of individuals. Several new kinds of services have been developed like Pay TV, Access key, financial services, transportation, Medicare,...etc., and all of these new kinds of services will have very different specifications, and require specific approaches to fulfill the arising security demands. This will provide the impetus for creation of the MSC.

However most smart card are still in their infant stage with single application (means specification), Many existing applications do not fit with their initial definition. However, for many service providers and application designers, the smart card domain is still not perfectly well identified in technology and capabilities.

The developed cryptography on the smart card will be a key technology for secure electronic commerce and electronic payment applications. The card offers the unique advantage to keep cryptographic mechanisms securely in tamper-proof equipment. Smart cards will be used for access control instead of passwords, for the generation of digital signatures, for encryption or decryption, as an electronic purse and as a repository of any confidential information.

For the last four years, there has been an increasing demand for a public-key smart card from national administrations and large companies such as telephone operators, bank, and insurance corporations. The security degree of smart card has caught a high point. In 1993 [6] although the first MSC were emerging in Germany for PCs appeared on the market, but the development of the MSC is quite slow. Two special conditions of the MSC: the higher degree security and the smaller memory space will obstruct quick development to MSC. Because the chip on the smart card is restricted in size, the memory capacity of the chip is not big enough to store the larger and complex program for the Multi-application Operating System and the protocol codes of the cryptography algorithm with the factoring of large integers. The detail of constrictions of MSC will be shown in chapter 1.

So looking for an efficient cryptography algorithm which balances the security level based on the factoring of large integers with the required memory space of the restricted size chip is importance.

1.2 The status of cryptology for smart card

Public key cryptography is a powerful security tool in the field of information technology. The DES private key cryptosystem conforms to the U.S. standard determined by the National Bureau of Standards (now called NIST) in 1977 [5]. It uses the same key to encrypt and to decrypt a piece of data, and creates an implementation need: how to distribute and protect the key. As a possible solution to these problems, Diffie and Hellmam introduced the concept of Public Key Cryptography, based on the difficulty of solving a “trapdoor” problem in 1976 [7]. Several studies have been done to find a practical Public Key Cryptosystem [8, 9]. In a public key cryptosystem, each individual in the cryptosystem is assigned a unique pair of keys, one for encryption and the other for decryption. This simplifies the key management requirement. The holders of public keys can not "see" (or decrypt) previously encrypted data using other pairs of public keys. But

this algorithm is slower than private key algorithm, and required key values are very long. However, the feature of the one-way key is very attractive.

The Rivest-Shamir-Adleman (RSA) cryptosystem that was invented in 1977 [10] is the current standard for Public key encryption today, and is applied widely. As RSA compatible chip cards become available, the use of this standard will increase rapidly. The security of RSA depends on the size of the modulus N. Odlyzko [4] has recently forecast that a 512 bits module will be vulnerable to factorization in a couple of years and is not suitable for the long-term protection of secrets. Perhaps 768 bits module will be vulnerable to factorization by the year 2004. These estimates are based on projections of computing power, algorithmic advances and continuing ability to organize disparate resources over the Internet. Barring major advances in algorithms, 1024 bit RSA will probably be secure for many years to come and seems likely to become commonly used soon.

Since the first useable public key cryptosystem RSA was introduced, a variant having a common property based on the problem of factoring large integers was created and developed rapidly. For getting a cryptography algorithm more closely to the feature of the one-way function, another type of public key cryptography -- based on the discrete analogue of the logarithm function -- gave rise to a second current of research in computational number theory. It is called classical discrete logarithm defined over a finite field. The security of this public key cryptosystem is based on the difficulty of the discrete logarithm problem.

Diffie-Hellman Key Exchange and ElGamal Cryptosystem are two cryptosystems, of which the securities are based on the difficulty of the discrete logarithm problem. The details will be described in chapter 2.

1.3 The status of Elliptic Curve Cryptosystem for smart card

In 1985 [5] a variant of discrete log cryptography was proposed, based on the discrete logarithm problem in the group of points of an elliptic curve defined over a finite field. The cryptosystems using discrete logarithms in this group of points have two potential advantages over systems based on the multiplicative group of a finite field (also over systems based on RSA):

(1). There are a huge number of different elliptic curves available in the groups,
(2). The absence of sub-exponential time algorithms that could find discrete logs in the groups.

Moreover, the discrete logarithm problem in this group is believed to be very difficult, in particular, harder than the discrete logarithm problem using finite fields of the same size as the key. It was for these reasons that elliptic curves were first suggested in 1985 by N. Koblitz[11] and V. Miller [12] for implementing public key cryptosystems.

In developing elliptic curve cryptography, the most dramatic was the demonstration by Menezes, Okamoto and Vanstone in 1990 that the discrete log problem on a 'supersingular' elliptic curve can be reduced to the discrete log problem in a finite field. This result means that one should avoid the set of supersingular curves if one wants to have a cryptosystem whose cracking problem is, to the best of our current knowledge, of fully exponential complexity.

Elliptic curve cryptosystems potentially provide the equivalent security compared with the existing public key schemes, but with shorter key lengths, which means smaller memory space is required. The advantage of this feature can be a crucial factor in the design of the MSC. It is also the reason, for MSC specially, why this report explores the feasibility of implementing secure and efficient public key cryptosystems using elliptic curves in a chip of restricted size.

Elliptic curves over finite fields can be used to implement the Diffie-Hellman key exchange scheme, and the ElGamal and NIST signature schemes. These systems potentially provide equivalent security to the existing public key schemes, but with shorter key lengths. So it is most suitable for the design of MSC, where both memory and processing power are limited.

For applications that cannot operate efficiently with a large key length N, cryptosystems based on elliptic curves using discrete logarithm algorithm have been available for several years. At present, no satisfactory method for determining the parameters for cryptographic use has been developed. The Siemens Corporate Research and Development Department [13], as part of a joint project with the University of Essen, has developed a method for constructing elliptic curves with given parameters. The advantage of cryptosystems constructed in this way is their compatibility with normal congruent arithmetic. A large number of elliptic curves, including all required parameters, can be explicitly determined with this method. These curves can be directly used for new, efficient cryptosystems, with each curve defining its own cryptosystem. For example, the coprocessor of the SLE44C200 produced by Siemens Co. can perform these calculations. Key lengths from 128 or 256 bits are sufficient.

At present, expansion of the memory to be used for program and data is important. However, it is expected that the development of silicon industry technology for the expansion of memory capacity will be quite slow in the near feature. A large number of technological improvements are still conceivable in hardware (e.g. random number generator for key creation, authentication) and software (e.g. new cryptology, such as elliptic curves cryptosystem). So that the next smart card generation will be suitable for memory intensive applications, such as in MSC.

This report will be presented in four chapters. Beginning with chapter 1, there is an introduction to Development of the Multi-application Smart Card that is proposed for next generation smart card. Chapter 2 provides an introduction of three public key cryptosystems: RSA, ElGamal Scheme and Elliptic Curve Cryptosystem that will be

needed for the investigation in the next two chapters. In chapter 3, the main task is a description of the mathematical principles corresponding to three different kinds of cryptography algorithm: RSA, ElGamal Scheme and The Elliptic Curve Cryptosystem. Here the description of mathematical principles in number theory required understanding the different characteristics in the different cryptography algorithms. In chapter 4, the comparison of the efficiency and running time between three cryptosystems will be considered. It will be shown that the Elliptic Curve Cryptosystem has the desirable features of security and smaller memory capacity required for the development of MSC. On the other hand, there is a disadvantage of a lengthy running time in the implementation of the algorithm, which is one of the important reasons that the Elliptic Curve Cryptosystem has been proposed for several years, but has still stayed on paper at present. With improvement in the cryptographic algorithms and increased computational power, it may be possible to improve the computation time of the algorithm to meet the ISO standard, which would lead to the MSC becoming practical.

In conclusion, there are some suggestions required for the development and implementation of three technological tasks:

(1) Development of the silicon technology to expand the memory capacity,
(2) Improvement of designing hardware (e.g. OS, Data Base, instructions) and software (e.g. algorithm, protocols) in the MSC,
(3) Improvement of computing architecture and the rate of the ECC.

Implementation of all above technology tasks will have a great of effect on the creation of the MSC. So that it is likely to have a rapid development both in quality and quantity on a mass scale worldwide in the next ten years, and will affect many areas of our lives. The security, convenience and integrity of MSC technology will benefit applications as varied as commercial, telecommunication, transportation, medical, identification, education, employment, ... etc., and will change people's lives.

1. Development of the Multi-applications Smart Card

From the aspect of applications, the smart card can be divided into two types: Single application and Multiple application. A single-application smart card (**SSC**) is defined as one, which has only one specification in which there are multiple services or functions, such as a phone card issued by Telecom. A Multi-application Smart Card (**MSC**) is defined as one, which will probably support different types of applications with different specifications, such as financial services, identification, transportation, Medicare and access key, ... etc..

Currently, the most common type of smart card with surface contacts to reach the market place is the single-application smart card with few services or functions. This chapter, describes the possible future development of MSC, restricted by kind of contact, and will be in three sectors: the physical architecture of the MSC, the logical architecture of the MSC and the Security of the MSC.

1.1 Physical Architecture of the MSC

1.1.1 Dimensions and Location

The dimensions and location of each of the contacts shall comply with Figure 2 of ISO 7816-2, with the contacts on the front of the card. The location of the contacts [14] relative to embossing and/or magnetic stripe shall be as shown in Figure 1-1.

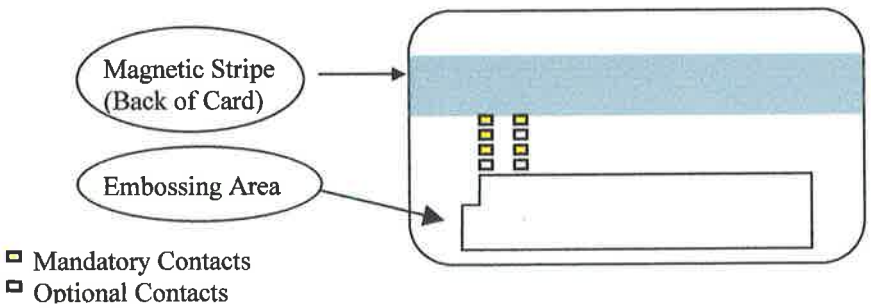

Fig.1-1. Location of Contacts

Smart cards rely on chip technology not only for information storage, but for information processing as well. A microcircuit is embedded in the plastic base of an existing smart card. The microcircuit consists of an electronic chip bonded to a circuit board and connected to electrical contacts on the board. The production of smart card consists of two steps [15], as shown in Fig. 1-2.

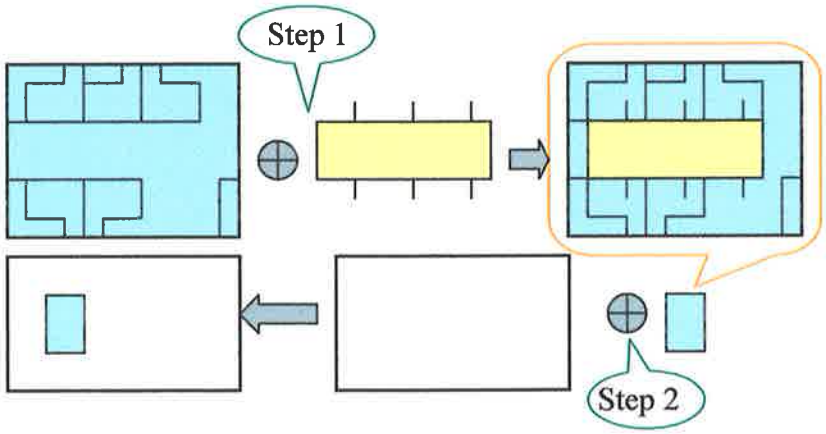

Fig.1-2. Smart Card Production

Step 1. Wire bonding (chip + circuit board),

Step 2. Potting (chip + circuit board + plastic card).

1.1.2 Standard of the Card

A Smart card must resist mechanical stresses like falls, torsion, and bending. Smart cards must also be resistant to static electricity and to exposure to various types of radiation such as x-rays, ultraviolet (UV) light, and electromagnetic fields. These physical characteristics are very precisely specified in the existing standards.

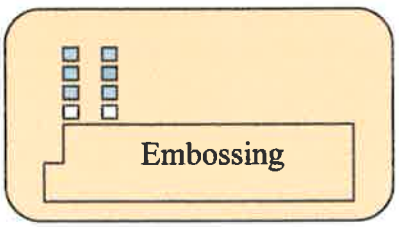

Fig. 1-3 Lower location on front

In 1981 [16], the French standards institution (Association Franchise de Normalization (Afnor)) proposed a location and an assignment of six operational contacts plus two contacts reserved for future use. These are located on the front of the card, near the upper left corner, as shown in Fig. 1-3. This location corresponds to the minimum mechanical constraints for the microcircuit when the card is under torsion and bending stress.

The international standard (ISO 7816/2) was published in 1988. The changeover to the new standard will occur in the early 1990s. After that time a lower location has been adopted [16], the standard refers to a corner, on any side of the card.

There are in fact two final lower locations: in rear as shown in Fig. 1-1 and on the front as shown in Fig.1-3. Because of consideration of technology aspects, the most probable ultimate location may be the lower one on the rear of the card.

1.1.3 Standard of the Chip

The international standard of the chip size would still be well within the 25 mm^2 required by ISO 7816.

In the 1980s [16], a breakthrough occurred with the development of CMOS technology which consumes much less power, and which also provides this capability at an acceptable cost. No doubt the next step in this development, high speed CMOS technology, will be an important part of the integrated circuit and memory capacity development, along with the development of semiconductor technology, because at present the memory capacity of the chip has large spaces for expansion. The chips made by a few large semiconductor companies will lead the market, as shown in Fig. 1-4.

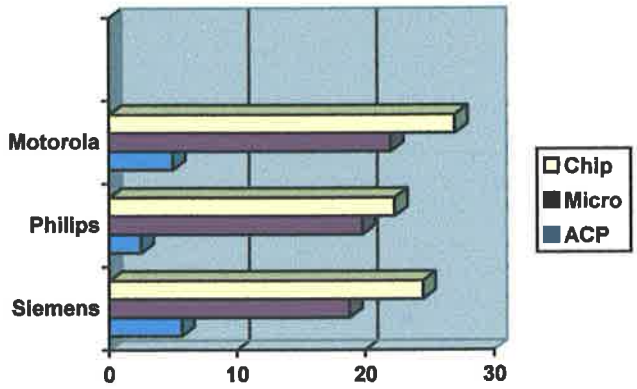

Fig. 1-4 Microprocessor area (mm^2)

Technological advances will place microprocessors that are more powerful on smart cards. These improvements will certainly lead to further decreases in computation times. More importantly, memory capacities will increase further. For example, from Hitachi [17], we can expect increases as shown in table 1-1, [18].

	Current	Future
RAM	256 byes	2 Kbyte
ROM	16 Kbyte	80 Kbyte
EEPROM	16 Kbyte	40 Kbyte

Table 1-1 Memory size of the chip

The interface to the outside would consist of eight contacts. The assignment of the contacts shall be as defined in ISO 7816-2, as shown in table 1-2.

C_1	Supply voltage (VCC)	C_5	Ground (GND)
C_2	Reset (RST)	C_6	Not Used[1]
C_3	Clock (CLK)	C_7	Input/Output (I/O)
C_4	Not Used	C_8	Not Used

Table 1-2 Contact Assignment of the Smart Card

----- C_4 and C_8 are not used and need not be physically present.

[1] Defined by ISO as VPP, the ICC shall not require VPP.

----- C_6 is not used and need not be physically present; If present, it shall be Electrically Isolated* from the integrated circuit itself and other contacts on the MSC.

*1) Defined in ISO/IEC 7816 as programming voltage (VPP).

*2) Electrically Isolated means that the resistance measured between C_6 and any other contact shall be > 10MΩ with an applied voltage of 5V DC.

1.2 Logical Architecture of the MSC

1.2.1 The Basic Architecture of a MSC

Initially smart cards were produced and issued with a number of different applications in many countries. Most of these smart cards with surface contacts were to reach the market place with only a single application, so called Single-application Smart Card (**SSC**). These cards are pre-paid, and are not debit or credit, But now most cards are becoming multi-function and should soon be able to provide normal debit and credit facilities, authentication and loyalty recording functions as well.

In the near future, smart cards will become MSC rather than just multi-function. Different services such as security, financial, medical information storage and transportation require quite separate applications, and these applications will have very different specifications. The different basic Architectures of a SSC and a MSC are shown in Fig. 1-5 and Fig. 1-6.

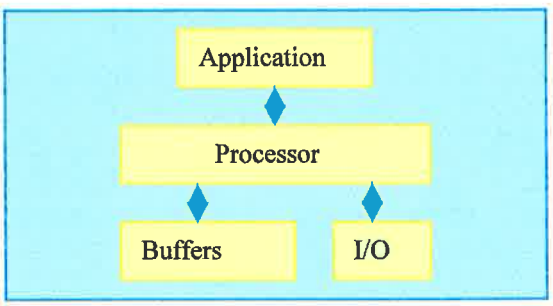

Fig. 1-5 Architecture of the SSC

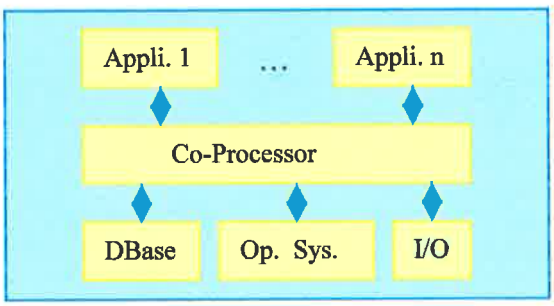

Fig. 1-6 Architecture of the MSC

1.2.2 The Architecture of a MSC Memory

We will review here the logical structure of memory in the chip of the MSC. The memory comprises three parts: RAM, ROM and EPROM/EEROM as shown in Fig. 1-7.

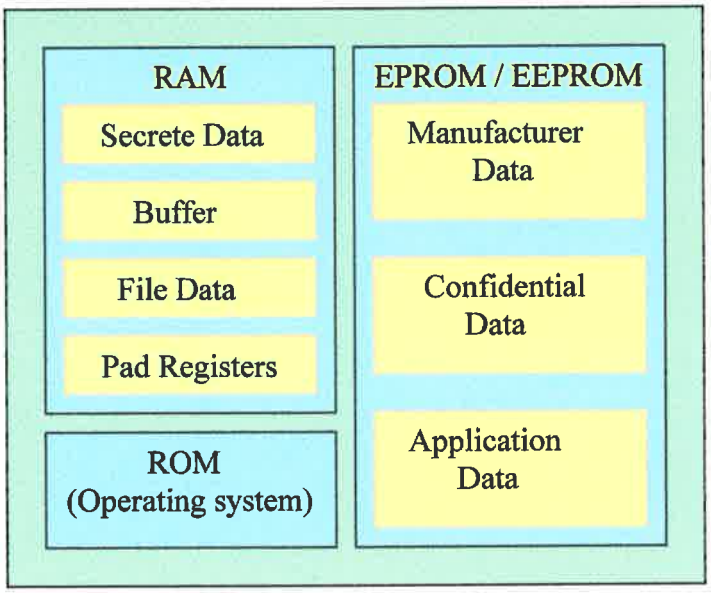

Fig. 1-7 Memory structure in a Card

1) The ROM memory containing the Smart Card Operating System (SCOS) and several manufacturer codes cannot be accessed by issuers or by user.

2) The RAM memory is employed chiefly to store intermediate results of the microprocessor. Also it is used to handle ready-to-use information required by the microprocessor and it usually has several predefined zones: Scratch Pad Registers, Built-in Data Encryption, Buffer Area and File Data Zone,

   --- Scratch Pad Registers have the same function as the usual microprocessor registers, such as holding numerical results, addresses, and pointers. Its length may be 1 byte or longer.

   --- Built-in Data Encryption usually has a dedicated RAM zone where data such as intermediate data, cryptographic parameters, and random numbers are stored and handled.

--- Buffer Area may be used to store command parameters. Additionally, data transmission and reception may use separate buffers, the transmission buffer may include a counter, and the reception buffer may store sequentially the last byte received through the I/O port.

--- File Data Zone stores the name and parameters of the working file when using files in the MSC. The microprocessor uses this zone to keep the address of the file within the user/application memory, as well as its size, record number, record lengths, and most importantly, the addresses in the secret zone where the protection rules of that specific file are stored. These protection rules may in turn be temporarily stored in RAM.

3) The EPROM/EEPROM is nonvolatile memory (i.e., non-rewritable or rewritable). The most important part of this memory is the Directory Structure (DS). The DS is structured according to one of the following formats:

--- Memory is divided into several zones, whose size may be constant or variable. Every zone may hold one or several data files. Zones are separated according to their functionality, data type and protection level.

--- ISO 7816/4 defined a hierarchical file structure, though zones are also accepted for historical reasons (most current cards use zone structure) and for technological reasons (many cards cannot be hierarchically structured). Three file categories are considered: master files, dedicated files and elementary files. Each card has a single master file; paths are defined to access other files. More details will be discussed in next subchapter 1.2.3.

In either case, the card must be formatted to define suitable areas. Some recent card models allow reformatting, other classic models are preformed by the manufacturer.

1.2.3 The Architecture of a MSC Operating System

The Hierarchical Memory Structure is the most suitable structure for a MSC. ISO 7816/4 proposes a hierarchical file structure as shown in Fig. 1-8.

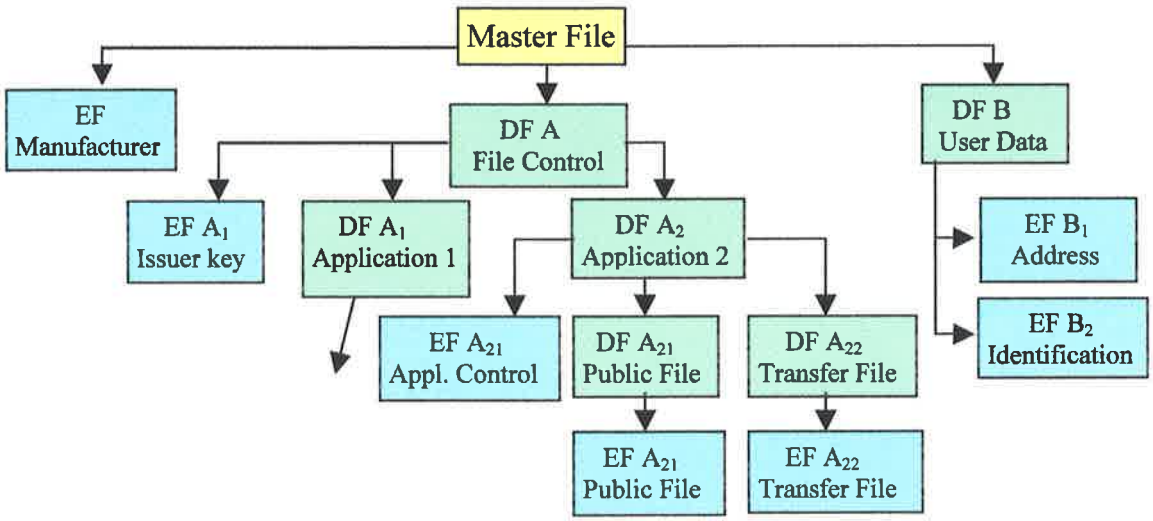

Fig. 1-8 The allocation of application in MSC with hierarchical file structure

The mandatory root is called Master File (MF). The role of subdirectories is carried out by optional Dedicated File (DF). Data are stored mainly in Elementary Files (EF). EFs may not be parents of DFs or other EFs. File control information is stored in the files or in the file's parents. A 2-byte identifier references files. Linking the identifiers of their parents, grandparents, and so forth, down to MF makes paths to specific files. Four EF types are defined:

* Public EF: free access,
* Application control EF: read protected, stores control information of the application,
* Internal secret EF: external access is always avoided,
* Working EF: to store application data.

Strict access control mechanisms aside, the file hierarchical structure of a MSC described in ISO 7816/4 is intended to solve most of the file managing pitfalls in the MSC (Fig. 1-9).

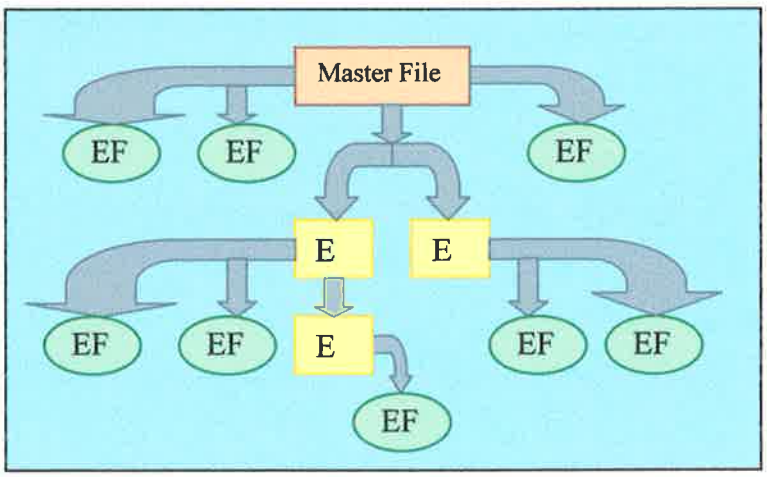

Fig. 1-9 The security architecture in O.S. of MSC

Each application can be logically placed in a different branch of the directory tree, owning a dedicated file that watches data flow to and from its branch. The master file at the root may look after general matters of the card, including data flow to and from the outside world.

In a MSC, the card issuer, the service managers, and the service providers are clearly identified and associated with levels of files. The card issuer controls the MF and the creation of DFs, each service manager controls the creation and the development of EFs in its own dedicated (sub) files, each user either controls access rights. Therefore the MSC can support several independent "application" files. A new DF can be created at any time under control of the MF. The MF and each DF contain a bunch of ID keys: management keys (for managing access rights and updating keys) and control keys (for deciphering control words). The first management key in a DF is mandatory written under control of a management key of the MF. The entitlements, along with various names and addresses, are stored in EF. Moreover, in the MF, EFs may hold parameters for a general-purpose device. Such parameters are security information, software to be downloaded, or connection information to access a remote management center.

The security policy of the MF and DF is based on a set of independent diversified cryptographic keys: one issuer key, few secondary keys used for authenticating the service provider and for securing operations, and more secondary keys for signature and

authentication of the card. The MF, as well as each DF, also contains a set of other EFs storing various data. In each file, there are a few erase keys for the erasure of the EEPROM and a few dedicated keys for digital signatures.

A hierarchical structure of the files in the MSC operating system has been briefly described in this subchapter. The remains are the inter-structure of each file and the transmission protocols of these files in this chapter, and will be described in next subchapter.

1.2.4 The Architecture of a MSC File

In a card [19], every file is made of an 8 byte header and an arbitrary number of identical records, as shown in Fig. 1-10.

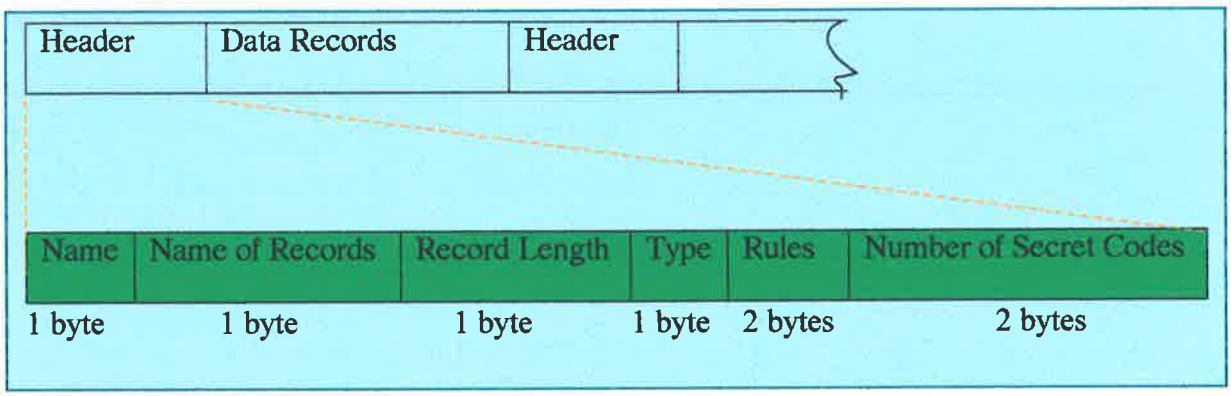

Fig. 1-10 Architecture of a file

The header includes six parts:

Byte_1: file name,

Byte_2: number of records contained in the file,

Byte_3: record length; the file size excluding the header may be calculated as the byte_2 times the byte_3,

Byte_4: file type,

Byte_5 & 6: access rules,

Byte_7 & 8: secret codes; the nibbles of bytes 7 and 8 contain the numbers of the secret codes employed in the access rules of bytes 5 and 6 respectively.

	B_8	B_7	B_6	B_5	B_4	B_3	B_2	B_1
T = 1 only	1	0	0	0	0	0	0	1

Table 1-3 Basic response coding of character TD1

Two types of transmission protocols [14] are defined: character protocol (T=0) and block protocol (T=1). The MSC shall support either protocol T=0 or T=1. Terminals shall support both protocol T=0 and T=1. The protocol to be used for subsequent communication between the card and terminal shall be T=0 or T=1 indicated in TD1[(1)], as shown in table 1-3.

Character Frame Data is passed over the I/O line in a character frame. Prior to transmission of a character, the I/O line shall be in state H. The structure of a character frame shows in Fig. 1-11.

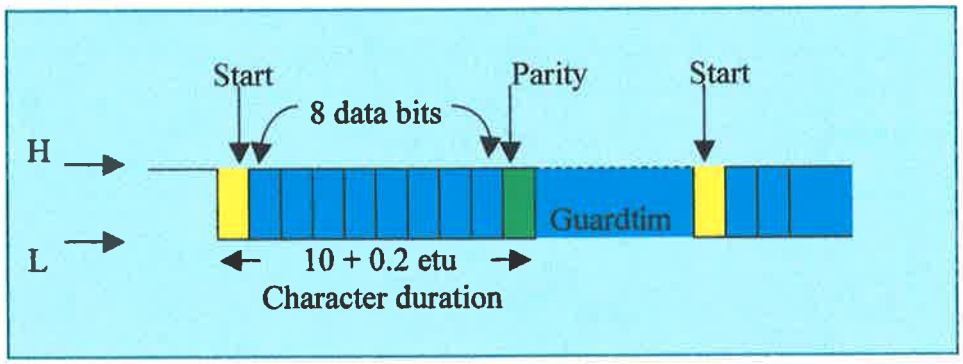

Fig. 1-11, Architecture of a Character Frame

A character consists of 10 consecutive bits as following:

* 1 start bit in state L,
* 8 bits which comprise the data byte,

(1) TD1 . Interface Character, convey information that shall be used during exchanges between the terminal and the card to the answer to reset. It indicates whether any further interface bytes are to be transmitted.

* 1 even parity checking bit.

The receiving end detects the start bit by periodically sampling the I/O line. The 8 bits of data and the parity bit itself are included in this check but not the start bit.

* Block Frame

The protocol consists of blocks transmitted between the terminal and the card to convey command and response Transaction Protocol Data Unit (TPDU) and transmission control information. The data link layer block frame structure is shown in table 1-4.

Prologue Field			Information Field	Epilogue
Node Address Detection (NAD)	Protocol Control (PCB)	Length (LEN)	Control Information (INF)	Error (EDC)

Table 1-4. Architecture of a block Frame

The block consists of three parts:

1) Mandatory prologue field, which consists of three mandatory bytes:
   a. Node address to identify source and intended destination of the block and to provide VPP state control,
   b. Protocol control byte to control data transmission,
   c. Length of the optional information field.

2) Optional information field, which is conditional. When present in an I_block, it conveys application data. When present in a S_block, it conveys control information. R_block shall not contain an information file.

3) Mandatory epilogue field, which contains the EDC of the transmitted block. A block is invalid when a parity error and/or an EDC error occurs. This specification only supports the LRC as EDC. The LRC is one byte in length and is

calculated as the exclusive-OR of all the bytes starting with the NAD and including the last byte of an information file, if present.

The interested reader can refer to [14] for more details.

1.2.5 Coding of Transmission

In ISO standard 7816/3, only the physical procedure of 'answer to reset' (**ATR**) has so far been defined for the "synchronous transmission" mode to which smart cards are assigned. This states that after a reset signal, the chip must output 32 bits of information synchronously with the clock pulse. The bit duration used on the I/O line is defined as an elementary time unit (**etu**) [14]. A linear relationship exists between the etu on the I/O line and CLOCK frequency (f), which shall be in the range 1 MHz to 5 MHz. Current etu = F/Df seconds

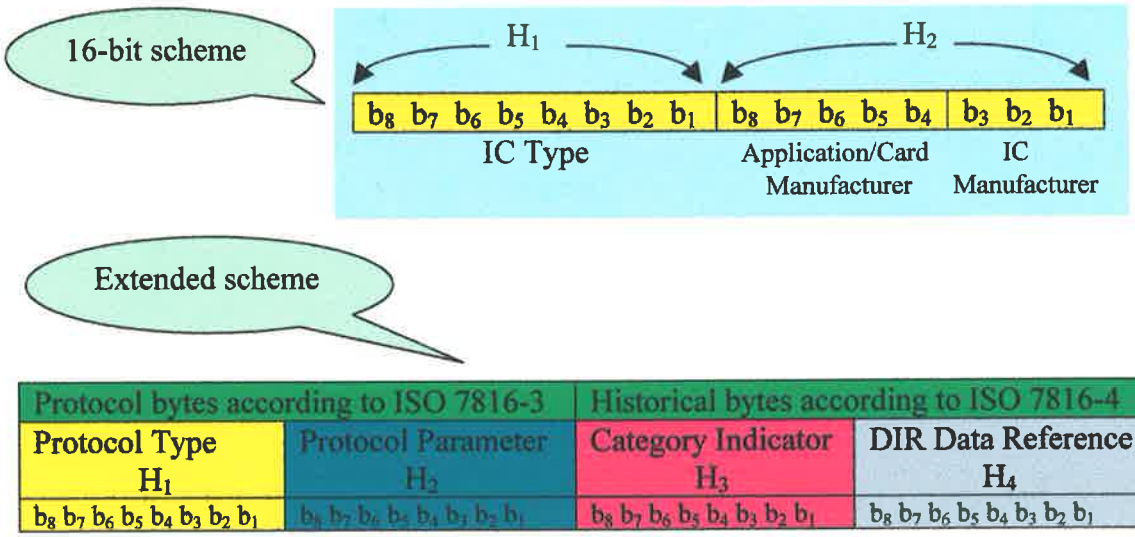

Fig. 1-12. 16-bit scheme (ISO 7816) and 32-bit scheme (DIN NI-17.4) for ATR in Transmission

Note: For the basic ATR, only a F = 372 and D = 1 are supported. Thus the current etu is the same as the initial etu given by 372/f seconds. If the card has an internal clock, the initial etu is 1/9,600 second [19]. If not, the initial etu is 372/f second, where f is provided by the interface device on the clock. During the ATR, the maximum interval between the leading edges of the start bits of two consecutive characters shall be 9,600 initial etus. The current etu is meant unless otherwise

specified. As smart cards become more widespread, 16 bits code is coming up against its limits, shown in Fig. 1-12. They can be distinguished by the first two bits, known as the structure identifier [20].

The 16_bit of the ISO 7816 scheme contains the chip type in the first byte and the chip manufacturer and application in the second. The 32_bit of the ISO scheme is divided into the four bytes from H_1 to H_4:

- H_1 --- specifying the protocol type:

  $b_8 \sim b_5$ Protocol tupe s,

  $b_4 \sim b_3$ Reserved for future use (RFU),

  $b_2 \sim b_1$ Structure Identifier: 00 = defined by ISO,
  10 = Structure 1,
  01 = Structure 2,
  11 = Structure 3.

- H_2 --- specifying the protocol parameters:

  B_8: RFU,
  B_7: 0 = Read to end, 1 = Read with defined length,
  $B_6 \sim b_4$: Number of data units,
  $B_3 \sim b_1$: Length of data units in bits.

- H_3 --- specifying the category indicator:

  $B_8 \sim b_7$: Category indicator according to ISO 7816-4.

- H_4 --- specifying the DIR data reference:

  $B_8 = 1$: DIR data reference specified, $b_7 \sim b_1$ = reference of DIR data,

  $B_8 = 0$: DIR data reference not specified, $b_2 \sim b_1$=outside the scope of ISO 7816-4.

A scheme to be adopted by the ISO will then contain the combination 00. Four bits are available for the protocol type. The associated parameters are held in the second byte with the number and length of the data units. The next two bytes are assigned under ISO 7816/4, to the category indicator and data reference to the presence of further data.

Currently, the 16_bit scheme has been successfully retained for the smart card (e.g. Siemens chip SLE 4404/4406). The extended 32_bit scheme is to be introduced for all other types of chips after a transitional period.

1.3 Security of the MSC

In the course of a transaction involving a smart card, the card delivers information (stored data, computation results) and modifies its contents (data storage, event memorization): the built-in electronic circuits both process data and store information in internal memory. These semiconductor technology trends definitely enhance both the physical and logical security of MSC:

* Better integration enhances physical security by making it more difficult to physically probe and recover information from the chips dedicated to MSC.

* Additions in processing power (CPU, RAM) and in operating systems (ROM) enhance logical security by allowing the implementation of more complex cryptographic algorithms and protocols in MSC.

A number of assault scenarios are conceivable, for which there is a range of countermeasures based on semiconductor technology. Attacking on the security of a MSC can be divided into three categories:

1) Data spying,
2) Data alteration,
3) Forging.

However, there are a range of countermeasures against the manipulation and forging of a smart card to ensure effective protection as the following two sectors:

1) Physical security,
2) Logical security.

The physical security means the hardware security such as security of chip and terminal. Here the description of the physical security is restricted to the chip and card. The logical security means the software and system security such as security of operating system and message transmission among card, terminal and systems. The descriptions are restricted to the operating system and the applications of security system. They will be described individually in the following.

1.3.1 Physical Security of the MSC

Smart cards are very difficult to reproduce without the right facilities and expertise. Manufacture of the chips requires very complex and expensive equipment. Even if stolen chips are used, both their bonding to the substrates and their encapsulation into the card require specialized equipment. The chips for smart cards are not publicly available and would not be easy to obtain.

As we know the MSC has both the memory and the microprocessor on the same chip. Where they are separate devices it could be possible to X-ray the card and ascertain the position of the communication lines between the two parts. By careful probing through the outer layers of the card, it could be possible to read out the information flowing between the microprocessor and memory. However, it is nearly impossible to extract this information from the smart card which has the microprocessor and memory combined on the same chip.

Another form of security [16], we should consider in chip production, is the security of chip testing:

1) Each chip supports about twenty additional test contacts, and tests are conducted under control of the outside world,

2) Each chip supports one or two test contacts, and an internal self-testing program written in a small extra ROM conducts tests.

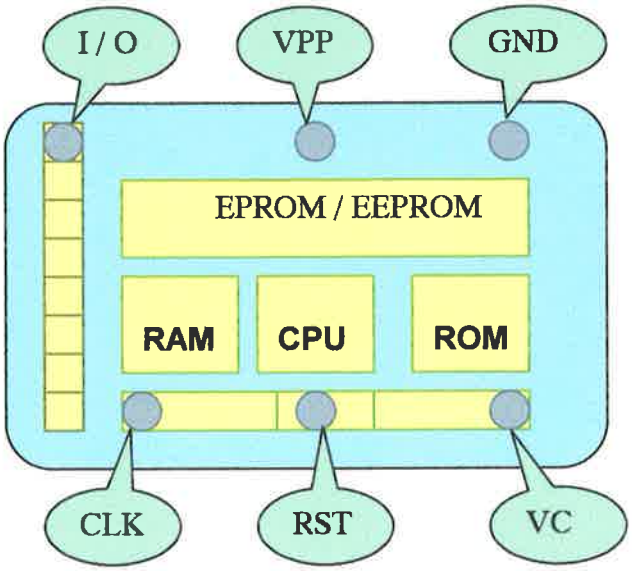

Fig. 1-13 The model of Self-program micro-computer

Before cutting wafers on production lines, a 512-byte internal routine is activated through two specific test contacts, shown in Fig. 1-13 near RST under the CPU. The EEPROM of each validated component receives various information: locks, codes, erasure indicators, chip serial number, while nothing is written in rejected components. Breaking fuse links buried in the silicon then systematically destroys the two test contacts. This operation, which eliminates non-user modes on valid chips, also positively disables invalid chips where nothing has been written.

As a matter of fact, the self-testing routine may write these erasure indicators to be tested by the card before executing any command in user mode during any transaction. If such an erasure indicator is erased, either by accident or by violation, then the chip is definitely disabled. Such EEPROM cells are constructed so as to be the most sensitive ones to erasing radiation. This is an example of the current reliability philosophy of using weak-link/strong-link designs to enhance reliability, since the weak-link is designed to disable the device before the operational strong-links can be subverted. Valid chips are then inserted into cards during the process of card manufacture. A manufacturing code or key is used for protecting chips from the time of chip manufacturing to card issuing.

Throughout the operation, several testers in the chip determine readiness: voltages, clock frequency, light, temperatures are all measured. These indications may also be used by the operating system to increase security. The mapping of memory addresses should be controlled by the internal program itself, and not be accessible to outside control. Whatever the physical security systems, system designers must carefully consider the potential consequences of chip violations. Secret keys must be as diversified as possible, tied to user identification number and chip serial number. A successful violation then compromises only one user and does not endanger the whole system, thus reducing the risk of widespread fraud. These aspects of logical security are strongly related to cryptology.

1.3.2 Logical Security of the MSC

The logical security of the MSC is based on its operating system. The smart card operating system deals with different commands and with the general security of the whole system. The most important part of a smart card system is software. A poor software design can induce weak security, inefficient functions, erroneous data, deadlocks, and many other potential problems. On the other hand, a good software design provides the user with qualified operations and additional functions. The efficiency of an operating system is not only related to ROM size, but also to the virtuosity of the software designer who finally specifies the technical configuration of the card. In this subchapter, the logical security of the MSC will be described individually in two sections: the security of operating system and the security of the MSC system.

1.3.2.1 The security of Operating System

The logical security is fundamental in the logical architecture of the operating system. The descriptions of the security of operating system consist of two parts: logical security and functional security.

For logical security, the independence between data files associated with flexible file management is the basis of any high security MSC operating system. This evolution clarifies the security architecture that is presently mature enough to be standardized. The security cannot be granted on an existing data file organization as in the existing operating system: DOS and UNIX. The difference takes place mainly in the security management that has to be taken into account in the model from the beginning of the design.

Because a file is fathered by another file, the essential creation process has to be protected. In other words, the right to create or to access a file has to be transmitted by heredity to enforce the independence between applications. This does not compel a son to have the same rights as its father, because it has the freedom to choose its way except for the creation procedure. The transmission of hereditary rights is managed by specific attributes that are transmitted to the son by the father.

The commands affect the objects and the entities specified by the security architecture. The set of commands should be defined afterward to permit compatibility and interchange between cards supporting different applications. The MSC may introduce the notions of Master File (MF), Dedicated File (DF) and Elementary File (EF) with the following set of definitions:

--- Master file: The DF at the highest level of the card is unique and mandatory file containing all the other files.

--- Dedicated file: Containing control information and other files, and giving access to EFs and DFs.

---Elementary file: Containing a set of records, and having a security policy under which an EF is never used as entries point to another file.

This structure can be interpreted in terms of security, and is standardizing the security architecture in the Operating System of a MSC, as shown in Fig. 1-9.

For functional security, the operating system has the security functions with multiple specifications accessed by secret keys for different companies, organizations and government departments. Every specification is separate from each other, so that one company or organization can only access its own specification by cardholder's secret keys. It also has several levels of security that can be controlled by secret keys in every specification for multiple functions. The company or organization can only get a part of services within their own functions, but for some services on higher level security within the same specification, it will need cardholder's secret keys. All of these personal secret keys will be produced through the personalization and electronic signature by the manufacturer and card's issuer. Of course, the operating system must be designed with these features, and must present much stronger security ability.

1.3.2.2 The Security of MSC System

A security system consists of the access control related to the hardware and software procedures and the application of intelligent cryptographic methods to the message communications between card, terminal and background systems. Different applications call for different levels of security. Therefore, chip manufacturers must provide a differentiated range of components to match the various security levels. The system security would be classified into five levels based on the different applications of hardware and software as shown in Fig. 1-14, and is described as follows.

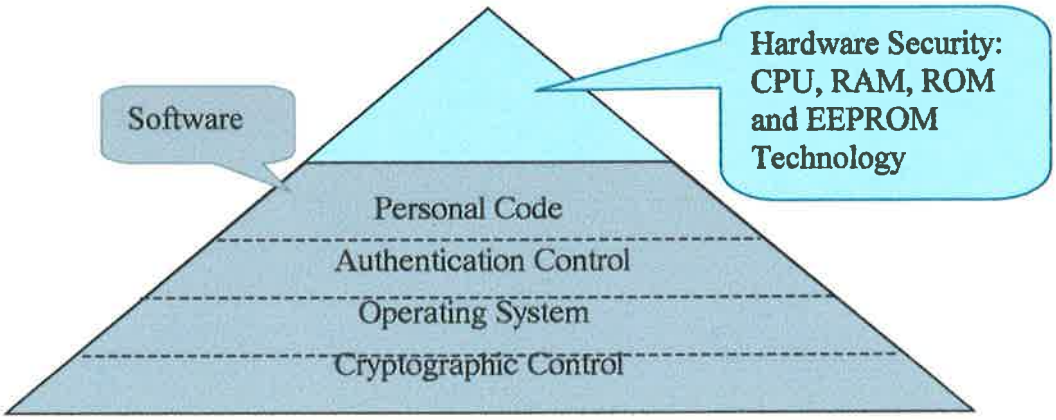

Fig. 1-14 Security Level for Smart Card Environment

1) The contents of EEPROM cells cannot be read by optical means. Any attempt to break into the cell by analyzing its contents will lead to immediate destruction of the data. A change can occur only in one direction from the charged state (valuable) to the uncharged state (value-less). If an intruder tries to determine the charged state by etching away the layers of the semiconductor chip, the contents of the EEPROM cell are destroyed by themselves as the gate is approached. Intrusion in a zero voltage state is impossible. The EEPROM also offers a facility for setting irreversible byte by byte or block by block flags to support the security features.

2) When the personal security code is used as a password, a comparison is made between the value entered by the user and the code stored on the chip. However, this is not done in the system, because the personal security code would have to be transferred from the chip to the terminal and thus exposed to probing, but by the chip's comparator logic. So the personal security code stays in a secure environment.

3) At the high security stage, symmetrical authentication by a hardware algorithm is inexpensive and can be implemented very effectively with an intelligent memory chip without a processor. In a challenge-and-response process between the terminal and the card, a hardware function on the chip responds to a query by the terminal. A secret key that is stored securely in the chip controls the hardware function. The parallel operation running in the terminal's security module allows comparison of both results for authentication between card and terminal.

4) The next higher security level is that of the Operating System. Its security performance is based on the following characteristics:

----- Programmability of complex internal sequences and intelligent analysis of information and signals,

----- Availability of application specific features,

----- Self-checking capability without the need for additional signals from outside,

----- Software implementation of symmetrical on-chip security algorithms.

5) The highest level of security is obtained with crypt-controllers. These contain not only all the controller security features of the previous level, but also an arithmetic hardware unit, with which the highest security level can be implemented by means of various asymmetrically on-chip security algorithms. The principal characteristic of these algorithms is separation of the encryption and decryption processes. So anyone gaining control of a procedure would be unable to master the complementary process as well.

Summary

As described above, the development of the MSC seems to be very viable. Along with further enhancement of the security requirements on the market, the complex and perfect software of the operating system has to be created. It is demonstrated that a cryptography algorithm will determine the high degree security of the MSC system. All of these require more memory space that is restricted with the maximum 25mm^2 required by ISO 7816. Currently the semiconductor manufacturer uses CMOS 0.3μm--1.0μm technology to increase the memory space of the RAM, ROM and EEPROM corresponding with 256-byte, 16-kbyte and 16-kbyte. But the memory space in the chip is still not enough for the huge and complex software, and for the high degree security based on the factoring of large integers provided by the various public key cryptography algorithm, although semiconductor technology has had enormous development.

Therefore, investigating and developing an efficient cryptography algorithm will not only be suitable for the smaller memory space of the chip, but will reach the same high degree security as other public key cryptography algorithm based on the problem of factoring large integers. These will be proposed and be described in more detail in the next chapters.

2. Public Key Cryptosystem for the Multi-application Smart Card

Abstract: Since 1976, when Diffie and Hellman [7] introduced the concept of public key cryptography, there have been twenty years of evolution in the procedure, and many different types of cryptography algorithms with various functions and characters have been proposed. The most widely used algorithms are the Rivest_Shamir_Adleman (**RSA**) public key cryptosystem which is based on the problem of factoring large integers, and the ElGamal cryptosystem based on the difficulty of computing discrete logarithm. Their problems are used to implement a fully functional public key cryptosystem, including digital signatures.

Elliptic Curve Cryptosystem (**ECCs**) is an another type of public key cryptosystem, the difference is that the elliptic curve analogue of this cryptosystem has more difficulty undertaking computations of group operation on an elliptic curve over finite fields than the computations of ElGamal algorithm over finite fields [11].

In fact, the evolutionary procedure of the public key cryptosystem is similar according to the model of Trapdoor one-way function that will be introduced in Appendix A. Therefore, at present the Elliptic Curve algorithm has the most efficient feature of one way function compared to any other algorithm.

In this chapter, the description of the public key cryptosystem applied to multi-application smart card consists of three subchapters which are the RSA cryptosystem, ElGamal cryptosystem and Elliptic Curve Cryptosystem, and will be introduced as following.

2.1 The Rivest_Shamir_Adleman (RSA) Cryptosystem

The RSA cryptosystem is the most widely used public key cryptosystem at present. The RSA cryptosystem not only provides the solution to problems in the Private Key

Cryptosystem, but also can be used to obtain Digital Signature, and hence achieves higher degree security as well. The description below will include RSA cryptography algorithm and RSA digital signature.

2.1.1 RSA Cryptography Algorithm

The RSA algorithm is a public key algorithm that provides a block cipher. The RSA algorithm is based on the fact that it is computationally simple to find large prime numbers, but believed to be computationally infeasible to factor the product of two such numbers [21]. The RSA cryptography algorithm is shown in Fig 2-1.

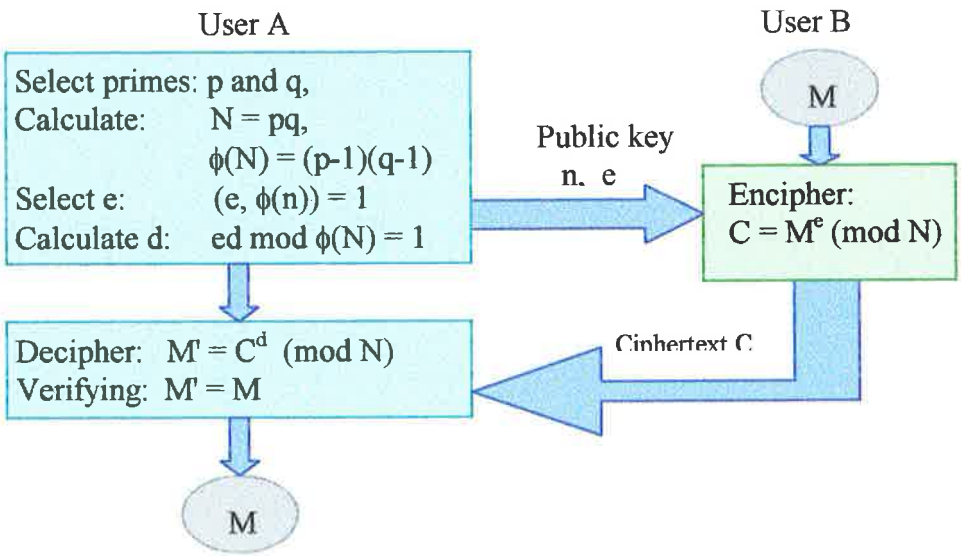

Fig. 2-1 The RSA Cryptography Algorithm

User A randomly selects two very large prime numbers p and q, then calculates the product N

$$N = pq \tag{2.1-1}$$

and Euler's totient function (see Appendix A):

$$\phi(N) = (p-1)(q-1) \tag{2.1-2}$$

User A then randomly selects another number e from the interval $1< e <\phi(N)$ such that

$$(e,\ \phi(N)) = 1 \tag{2.1-3}$$

which can be verified by using the Euclidean algorithm. The numbers N and e are the public keys which are openly transmitted to another user upon request.

If user B wishes to communicate with User A, then user B represents the message as a sequence of blocks, each of which has a value in (0, N-1). A plaintext block M is transformed into a ciphertext block C by using the public key and calculating the RSA one-way function

$$C = M^e \bmod N \tag{2.1-4}$$

This computation can be done rapidly by the discrete exponentiation method illustrated in example (2) of [21]. Since $e < N$ and $M < N$, all quantities in (2.1-4) are representable by b bits if N is slightly less than 2b, and discrete exponentiation requires at most 2b multiplications. The restriction $M < N$ ensures that (2.1-4) is a one-to-one transformation with an identical domain and range. User A calculates the number d such that

$$de = 1 \bmod \phi(N). \tag{2.1-5}$$

The integers d, p and q are secret keys. Euler's theorem and (2.1-3) ensure that d exists, and d can be computed by using the Euclidean algorithm. Equation (2.1-5) means that

$$de = k\phi(N) + 1 \tag{2.1-6}$$

where k is an integer. When ciphertext C is received from user B, user A deciphers C by computing

$$M' = C^d \pmod N \tag{2.1-7}$$

Substituting (2.1-4) into (2.1-7) and using (2.1-6), we obtain

$$\begin{aligned} M' &= (M^e)^d \pmod N \\ &= M^{ed} \pmod N \\ &= M^{\phi(N)k}M \pmod N \end{aligned} \tag{2.1-8}$$

If (M, N) = 1, then Euler's theorem yields M' = M. Thus, the deciphering successfully recovers the message. If M = 0, then M' = M trivially.

If $M > 0$ and $(M, N) \neq 1$, then either p or q divides M. It is not possible for both p and q to divide M because $M < N = pq$. Suppose that p divides M, then $M = ip$ for some positive integer i and $(M, q) = 1$. Since p and q are primes, Euler's theorem (see Appendix A) yields

$$M^{k\phi(N)} = M^{\phi(q)(p-1)k} = 1 \bmod q \tag{2.1-9}$$

This result implies that

$$1 = M^{k\phi(N)} - jq \tag{2.1-10}$$

when j is an integer. Multiplying both sides of this equation by M and using $M = ip$ and $N = pq$, we obtain

$$M = M^{k\phi(N)}M - ijN \tag{2.1-11}$$

Substituting (2.1-11) into (2.1-8) yields $M' = M$. Therefore (2.1-7) restores the message for all M in (0, N-1).

Since only d and N are required to decipher C, these two numbers constitute the trapdoor of the one-way exponentiation function. Since N is part of the public key, the number d can be considered the private key. However, d is generated from e and $\phi(N)$, hence, from e, p and q. e is a part of the public key. An example of the RSA algorithm is as follows:

Example 2.1 (Example 3 in [21]), Suppose that p=11 and q=13. Then $N = pq = 143$ and $\phi(N) = (11-1)(13-1) = 120$. Let e=11, which is a legitimate choice because $(e,\phi(N)) = (11,120) = 1$. According to (2.1-5), d is the solution to the equation $d \times 11 \pmod{120} = 1$. From (A.1-2) and $\phi(120) = (3 - 1)(40 - 1) = 78$, thus, the equation is obtained $d = 11^{77} \pmod{120} = 11$.

The public key is $(N, e) = (143, 11)$, the private key is $d = 11$. The primes are $\{p, q\} = \{11,13\}$. Suppose the message block is $M = 10$, the transmitted ciphertext is,

$$C = 10^{11} \pmod{143} = 43,$$

and the deciphering is $M' = 43^{11} \pmod{143} = 10 = M$.

To prevent a factoring of N by an exhaustive search and to render existing factorization algorithms computational infeasible, the primes p and q should have approximately larger than 100 decimal digits. The primes in this size also ensure that the probability that (M, N) $\neq 1$ is on the order of 10^{-100} [22]. The enciphering exponent e should be chosen so that $2^e > N$, which makes it impossible for M to be recovered from C simply by calculating $C^{1/e}$ because a modulo-N reduction occurs during the enciphering except when M = 0 or 1. The security of the RSA algorithm depends not only on the intractability of factoring N, but also on the unproved assumption. So that any way of inverting the RSA one-way function is approximately as difficult as factoring N.

Finding the value $\phi(N)$ directly would breaks the RSA cipher by allowing the computation of d in equation (2.1-5) without knowing p and q. If the $\phi(N)$ is known, then the equations (2.1-1) and (2.1-2) can be solved simultaneously for p and q. Therefore, the calculation of $\phi(N)$ provides a method of factoring N, and consequently is at least as difficult as factoring N.

Finding the integer d directly would break the RSA cipher by computation of M' in equation (2.1-7) without knowing $\phi(N)$ or factoring N. But if the integer d is known, then the equation (2.1-6) indicates that de - 1 is a multiple of $\phi(N)$. Therefore, the calculation of d is also at least as difficult as factoring N.

To use the RSA system, a user must first choose the primes p and q that should differ in length by a few digits, (p-1, q-1) should be small, but both p-1 and q-1 should contain large prime factors. To ensure that p - 1 has a large prime factor, a large prime p_1 can be first randomly selected and then the prime $p = ip_1 + 1$ is generated for as many even values of the integer i as is required. The RSA algorithm can also be used to obtain digital signatures, which is described below.

2.1.2 The RSA Signature Scheme

The function of digital signatures is the authentication and validation of electronic messages exchanged over the channels of a communication network. This kind of authentication is seriously deficient because both the sender and receiver must know a secret key. The sender uses the key to generate an authenticator, and the receiver uses it to check the authenticator. With this key, the receiver can also generate authenticators and can therefore forge messages appearing to come from the sender. In other words, authentication can protect both sender and receiver against third party enemies. If Alice sends a message to Bob, for example, Bob might fraudulently claim to have received a different message. Supposing Bob takes some action in response to a genuine received message, Alice can still claims that Bob in fact forged the message. Alice's a solution to the dispute problem arising from the dishonesty of sender or receiver, Diffie and Hellman [7] proposed the use of a digital signature based on certain public key cryptosystems. The signature procedure for the RSA cryptosystem is shown as Fig. 2-2.

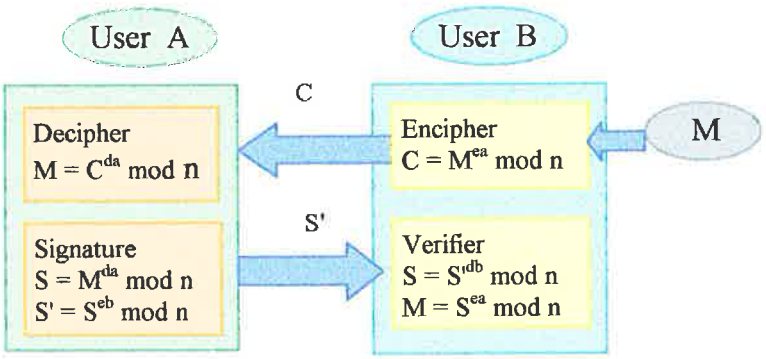

Fig. 2-2 The Principle of RSA Cryptography and Signature Scheme

To begin the signature process, A user chooses primes p and q, and computes $N = pq$, $\phi(N) = (p-1)(q-1)$, and chooses e to be an integer in $[1, \phi(N)]$ with $\gcd(e, \phi(N)) = 1$, and chooses d to be an integer in $[1, \phi(N)]$ with $ed = 1 \bmod \phi(N)$. The N and e are public, and d, p and q are secret.

* Signature Generation:

If A user wants to send a signed message M to B user, A user first computes a "signature" S for the message M using secret key d_a:

$$S = M^{d_a} \pmod N$$

Then A user encrypts S using B user's e_b that is the public key,

$$S' = S^{e_b} \pmod N$$

and sends the S' to B user.

** Signature verification:

B user first decrypts the S' with d_b to obtain S.

$$S = S'^{d_b} \pmod N$$

B user knows who is the presumed sender of the signature. This can be given if necessary in plain text attached to S. B user then encrypts the S with A user's public key e_a, and verifies M.

$$M = S^{e_a} \pmod N.$$

A user cannot later deny having sent B user this message, since no one else could have created $S = M^{d_a}$. Also B user can neither modify M to a different version M', nor forge A user's signature for any other message.

Summary: The RSA cryptosystem has not resisted all attacks. Some of the protocols for using RSA have been broken [23, 24]. Developing another type of public key cryptosystem with more security features is significant. Although the ElGamal algorithm based on the difficulty of discrete logarithm problems over finite fields has the ability to overcome some attack to the RSA system, but would not provide a higher degree of security than RSA system. ElGamal scheme gives an explicit methodology for using discrete logarithm problem to implement a fully functional public key cryptosystem, including digital signatures, and will be introduced in the next subchapter.

2.2 The ElGamal Cryptosystem

Abstract: In 1985, T. ElGamal [25] proposed the following public key scheme based on discrete exponentiation that has the properties of one-way function similar to RSA system. The one-way function is the discrete exponential function [21], as equation (2.2-1).

$$Y = \alpha^X \pmod p. \qquad (2.2\text{-}1)$$

where p is a large prime, x is an integer with $1 \le x \le p-1$, and α is an integer with $1<\alpha< p$ and $\alpha^1, \alpha^2, ..., \alpha^{p-1}$, in some order, congruent modulo p to 1, 2, ..., p-1. Such α is called a "primitive root modulo p". Every prime p has a primitive root modulo p [22]. For example, if p=7, α=3 is a primitive root modulo 7:

$$\alpha_1 = 3^1 \bmod 7 = 3, \qquad \alpha_2 = 3^2 \bmod 7 = 2,$$
$$\alpha_3 = 3^3 \bmod 7 = 6, \qquad \alpha_4 = 3^4 \bmod 7 = 4,$$
$$\alpha_5 = 3^5 \bmod 7 = 5, \qquad \alpha_6 = 3^6 \bmod 7 = 1.$$

The inverse of the discrete exponential function is the discrete logarithm and is denoted by

$$X = \log_\alpha Y, \quad 1 \le Y \le p-1 \qquad (2.2\text{-}2)$$

If p is a large prime and p-1 has a large prime factor q then evaluating the discrete logarithm requires far more computation than the approximately $2\log_2 p$ multiplications required for discrete exponentiation.

2.2.1 The ElGamal Cryptography Algorithm

This algorithm uses a very large prime p, a large prime factor q of p-1 and an element α of the finite field $\mathbf{F}_p = \{0, ..., p-1\}$ whose order (see Appendix A-2) is q. These system parameters p, q and α are known to all users. The plaintext message units are given numerical equivalent m in $\mathbf{F}_p$. Each user A randomly chooses an integer x_a, say in the range $0<x_a<q-1$, and each user B randomly chooses an integer x_b, in the same range

$0<x_b<q-1$. Both x_a and x_b are secret keys. A user computes $Y_a = \alpha^{x_a} \pmod p$, and sends Y_a to B user. Similarly, B user computes $Y_b = \alpha^{x_b} \pmod p$ and sends Y_b to A, and both Y_a and Y_b are public keys. Hence both A user and B user are able to compute K_{ab} [25].

$$K_{ab} = Y_a^{x_b} \pmod p$$
$$= Y_b^{x_a} \pmod p$$
$$= \alpha^{x_a x_b} \pmod p.$$

Notice that an intruder who solves the discrete logarithm problem in $\mathbf{F}_p$ breaks the cryptosystem by finding the secret key x_a from the public key α^{x_a}. In theory, there could be a way to use knowledge of α^{x_a} and α^{x_b} to find $\alpha^{x_a x_b}$, and hence break the cipher, without solving the discrete logarithm problem. However, it is conjectured that there is no way to go from α^{x_a} and α^{x_b} to $\alpha^{x_a x_b}$ without essentially solving the discrete logarithm problem.

In any of the public key cryptosystem based on discrete logarithms, p must be chosen such that p-1 has at least one large prime factor. If p-1 has only small prime factors, then computing discrete logarithms is relatively easy [26].

Suppose that A user wants to send a message m to B user, where $0 \le m \le p-1$.

1) A user chooses a random number x_a and computes the public key $Y_a = \alpha^{x_a} \bmod p$,

2) A user looks up B user's public key $Y_b = \alpha^{x_b} \bmod p$, and computes

$$K_{ab} = Y_b^{x_a} \bmod p, \qquad (2.2\text{-}3)$$

where $Y_b = \alpha^{x_b} \bmod p$ which is sent by B user.

3) A user computes $C = mK_{ab} \bmod p$, (2.2-4)
and sends to B user the pair of group elements (Y_a, mK_{ab}).

4) B user computes $Y_a^{x_b}$ and uses this to recover m.

It can easily be seen that the ElGamal cryptosystem is the application of Diffie-Hellman key exchange. So each case security is equivalent and is the based on the difficulty of the discrete logarithm problem.

Note: 1) The size of the message encrypted is double the size of the plaintext.

2) The multiplication operation in (2.2-4) can be replaced by any other invertible operation such as addition mod n. The decryption splits into two steps:

Step 1. B recovers the K_{ab} by computation, $K_{ab} = Y_a^{x_b} \pmod p = \alpha^{x_a x_b} \bmod p$, the x_b is known to B only.

Step 2. B recovers the message m by computation, $m=C/K_{ab} \bmod p$.

Breaking the system is very difficult [27]. Firstly, if m can be computed from (Y_a, mK_{ab}) and Y_i, then K_{ab} can also be computed from (Y_a, mK_{ab}) and Y_i, but (Y_a, mK_{ab}) and Y_i appear like random numbers since x_i and m are unknown. Secondly, even if m is known, computing x_i from (Y_a, mK_{ab}) and Y_i is equivalent to computing discrete logarithms. Because the x_i appears in the exponent in Y_i and Y_a.

2.2.2 The ElGamal Signature Scheme

ElGamal [25] also designed a signature scheme which makes use of the finite group GF_p, p is a large prime, with $2^{512} < p < 2^{1024}$, and $p - 1$ has a large prime factor q. It is known that GF_p is cyclic and is generated by α. Suppose the signature scheme is defined for $m \in \mathbf{F}_p = \{0,...,p-1\}$, where m is typically the hashed value[1] of a message to be signed.

* Signature Generation, A user does the following:

1) A user chooses a random integer $k \in \mathbf{F}_p$, such that $\gcd(k, p-1) = 1$.

2) A computes the group element $r_1 = \alpha^k \pmod p$.

3) A computes $s \in \mathbf{F}_p$ from

1 A *hash function* is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called *hash-values*.

$$sk = (m + r_1 x_a) \pmod{p-1} \qquad (2.2\text{-}5)$$

Here if s = 0, then A user chooses the random number k again. Of course such a probability is small. The signature for m is the pair (r_1, s).

** Signature Verification,

Given m and the signature (r_1, s), the verification is as follows:

1) B computes $r_1^s = \alpha^{ks} \pmod{p}$, $\alpha^m \pmod{p}$, and $Y_a^{r_1} = \left(\alpha^{x_a}\right)^{r_1} \pmod{p}$,

2) B iterates the equation

$$r_1^s = \alpha^m Y_a^{r_1}, \pmod{p} \qquad (2.2\text{-}6)$$

3) B verifies that they are the same from (2.2-5),

$$\alpha^{ks} \pmod{p} = (\alpha^m)(\alpha^{x_a r_1}) \pmod{p},$$

To forge A user's signature for a message m, a forger would have to solve the equation for r_1 and s.

$$\alpha^{ks} \pmod{p} = (\alpha^m)\left(\alpha^{x_a}\right)^{r_1} \pmod{p}, \qquad (2.2\text{-}7)$$

Fixing r_1 first and then attempting to solve for s is a discrete logarithm problem in GF_p. Fixing s first and then attempting to solve for r_1 gives a mixed exponential congruence in r_1, for which no efficient algorithm is known. Hence the security of the ElGamal signature scheme is based on the difficulty of the discrete logarithm problem in GF_p.

In practice, the message to be signed is a long sequence of entries from m. It is inefficient to sign each element of the sequence. So instead a hash function is first applied to the message to reduce a much smaller message digest, and it is this message digest which is then signed. The hash Function is public knowledge. To prevent forgery and impersonation, it must be infeasible to find two distinct inputs that hash to the same output value, and it must be infeasible to find an input that hashes to a given value.

Summary: Although the ElGamal scheme based on the difficulty of discrete logarithm algorithm over a finite field has the same high security features as RSA algorithm, the complex computation of this method is still based on the large prime p. It means that the ElGamal scheme also requires the same large memory space as the RSA required, if the ElGamal scheme would drive to the same high degree security as the RSA scheme.

In the application to MSC with small memory capacity, the chip on the smart card is restricted in size. Therefore the ElGamal scheme meets the same problem as RSA met, in which the memory capacity of the chip is not enough to store the large and complex program and the protocol codes of the cryptography algorithm with the factoring of large integers.

Elliptic Curve cryptosystems potentially provide equivalent security to the existing public key schemes, but with shorter key lengths which means smaller memory requirement, and will be introduced in the next chapter.

2.3 Elliptic Curve Cryptosystem

Abstract: Elliptic Curve Cryptosystems (**ECCs**), first suggested by Victor Miller [12] and Neal Koblitz [11], were a natural choice because they are immune to the index calculus attack (refer to §3.2.2). This means that smaller numbers can be used to achieve the same degree of security for the ElGamal algorithm as the 512-bit version described above.

The elliptic curve method has stronger one-way function, because the elliptic curve method uses a different discrete logarithm algorithm with group operation over the group of points on an elliptic curve compared to the discrete logarithm with integers mod N over finite fields, and the group operation is arithmetically more complicated [28].

The points on an elliptic curve E over a finite field form an Abelian group, hence the group E can be used to implement an analog of the Diffie-Hellman key exchange scheme, ElGamal scheme and the other public key cryptosystems. In this chapter, the Elliptic Curve analogue of the ElGamal cryptosystem is only described to compare with the multiplication of integers mod p as described in chapter 2.2.

The description of Elliptic Curve Algorithm requires the introduction of elaborate and complex mathematics. In order to keep the descriptions concise these mathematical issues are explained in more detail in chapter 3. The remainder of this chapter is organized as two sectors: Elliptic Curve Cryptography Algorithm and Elliptic Curve Signature Scheme, which are described as follows.

2.3.1 Elliptic Curve Cryptography Algorithm

The Elliptic Curve Cryptography Algorithm (**ECCA**) means the elliptic curve analog of the ElGamal algorithm based on an elliptic curve E, and a point P(x, y) to generate the whole addition group of E. The base field, curve equation, and starting point are all public parameters [28].

Let E be the non-supersingular curve (this term will be described in chapter 3):

$$y^2+a_3y = x^3 + a_2x^2 + a_6 \qquad (2.3\text{-}1)$$

defined over $\mathbf{F}_p$, where p is a large prime, and let P be a publicly known point on the elliptic curve E, preferably a generator of E. The elements of $\mathbf{F}_p$ are assumed to be represented by a normal basis (see Appendices B). User A randomly chooses an integer 'a' and makes a public key aP. The integer a is user A's secret key. Suppose the messages $M = (m_1, m_2, ..., m_{n-1})$ are ordered pairs of elements in $\mathbf{F}_p$.

To transmit the message (m_1, m_2) to A, sender B selects a random integer k which is B's secret key, and computes the points kP and akP = (x', y'). Assuming that $x' \neq 0$ and $y' \neq 0$

(the event x' = 0 or y' = 0 occurs with negligible probability for random k), then B sends the point (kP, MakP) to A, and MakP is the ordered pair of field elements (m_1x', m_2y').

To read the message, A calculates the point akP by A's secret key 'a' to obtain (x', y'), from which A can recover m_1 and m_2 by two divisions.

In the ECCs, four field elements are transmitted in order to convey a message consisting of two field elements. It is called "message expansion" by a factor of 2. The Message Expansion can be reduced to 3/2 by only sending x_1 and a single bit of y_1/x_1 (if $x_1 \neq 0$), instead of sending the point $P=(x_1, y_1)$. The following method can then be used to recover y_1.

Firstly, if $x_1=0$, then $y_1=a_6^{1/2}$. If $x_1 \neq 0$, then the change of variables $(x, y) \rightarrow (x, xz)$ transforms the equation of the curve (2.3-1) to $z^2 + z = x + a_2 + a_6x^{-2}$. Compute $f = x_1 + a_2 + a_6x_1^{-2}$. To solve the quadratic equation $z^2 + z = f$, let

$$z = (z_0, z_1, \ldots, z_{n-1}) \tag{2.3-2}$$

and

$$f = (f_0, f_1, \ldots, f_{n-1}) \tag{2.3-3}$$

be the vector representations of z and f respectively over $\mathbf{F}_p$. Then

$$z^2 + z = (z_{n-1}+z_0, z_0+z_1, \ldots, z_{n-2}+z_{n-1}). \tag{2.3-4}$$

Each choice $z_0=0$ or $z_0=1$ uniquely determines a solution z' to $z^2 + z = f$, by comparing the components of equation (2.3-3) and (2.3-4). The correct solution z' is selected by comparison with the corresponding bit of y_1/x_1 that was transmitted. Finally, y_1 is recovered as $y_1 = x_1z'$. The drawback of the method is that if an intruder happens to know m_1 (or m_2), he can then easily obtain m_2 (or m_1). This attack can be prevented by only sending (kP, m_1x'), or by embedding m_1 on the curve [29].

If the user wishes to embed messages on the elliptic curve, the following deterministic scheme may be used. Suppose that messages are (n-1) bit strings $M = (m_1, m_2, \ldots, m_{n-2})$, where M is an element of $\mathbf{F}_p$, and $m_{n-1}=0$. To embed M on the curve, M^3 is first computed

and then the Trace of M^3 (see Appendix H) is evaluated. If $Tr(M^3)=0$, then set $x_M = M$, otherwise set $x_M = M+1$. In either case, $Tr(x_M^3) = 0$. As in the preceding paragraph, one can easily find y_M such that $P_M = (x_M, y_M)$ is a point on E. Sender B can now send the pair of points $(kP, akP + P_M)$ to A. With this scheme the message expansion is by a factor of 4. The message expansion can be reduced to 2 by sending only the x-coordinate and a single bit of the y-coordinate of each point.

Note that after user A recovers x_M, A can decide whether the message sent is x_M or x_{M+1}, by checking whether the last bit of x_M is 0 or 1 respectively.

If every user uses the same elliptic curve and base point P, then the public key, namely the point aP, is n+1 bits in length. Otherwise the public key consists of a_6 (a_2 can be fixed to be 0), the points P and aP, for a total size of 3n + 2 bits.

2.3.2 Elliptic Curve Signature Scheme

The Elliptic Curve Signature Scheme (**ECSS**) means that the ElGamal type signature is constructed on an elliptic curve, which can be implemented in smaller sizes than finite fields, since the most serious attacks defined on finite fields cannot be applied to elliptic curves [30]. Therefore such characteristics might be suitably used on signature schemes.

Suppose that an elliptic curve E is over finite the field $\mathbf{F}_p$ ($p \neq 2^n$ is a large prime), and a base point $P \in E(\mathbf{F}_p)$ with a large prime order q. The user A has a secret key x_a and publishes the corresponding public key $Y_a = x_aP \pmod p$.

- Signature Generation:

  A user does the following:

  1) A user chooses a random number $k \in \mathbf{F}_q$, and computes $R = kP \bmod p$ on E,

  2) A sets $r_1 = x(R) \bmod q$ and computes $s \in \mathbf{F}_q$ from equation (2.2-5),

$$sk = (m + r_1 x_a) \pmod q \qquad (2.3\text{-}4a)$$

where x(R) denotes the x-coordinate of R. Here if either x(R) = 0 or s = 0, then A chooses the random number k again. The signature for m is the pair $(r_1, s) \in \mathbf{F}_p \times \mathbf{F}_q$.

•• Signature Verification,

Given m and the signature (r_1, s), the verification is as follows,

1) B computes $Rs = skP \pmod p$, (2.3-5)

and $r_1 Y_a = r_1 x_a P \pmod p$, (2.3-6)

2) B iterates the following equation by (2.3-5) and (2.3-6),

$$Rs = (mP + r_1 Y_a) \pmod p, \qquad (2.3\text{-}7)$$

3) B verifies that it is the same from (2.3-4a),

$$skP = (mP + r_1 x_a P) \pmod p, \qquad (2.3\text{-}8)$$

to forge A user's signature for a message m, a forger would have to solve the equation (2.3-8) for r_1 and s.

Summary: From the description of the Elliptic Curve Cryptosystem (**ECCs**) above, we have seen that the elliptic curve analogue of the ElGamal scheme has a different method of encryption to the ElGamal cryptosystem based on discrete exponentiation in a finite field. The former has the public key Y = xP with smaller bits size, while the latter has the public key $Y=\alpha^x$ with the same size as the field, namely at least 512 bits in length. Neal Koblitz [11] said that using the analogous procedure (i.e., doublings and additions), people can compute a multiple xP of a given point P in the same order of time as it takes to exponentiate α^x. Although there is no strict proof in mathematics proposed currently, it is believed to be the case that the computation of the group operation based on a group of points on an elliptic curve is more complex than the computation based on discrete exponentiation in a finite field [31]. That is the reason why the Elliptic Curve Cryptography (**ECC**) is likely to provides equivalent security but with shorter key length.

The advantages of the ECC providing equivalent security with shorter key length will be a crucial factor in designing the MSC. The investigation of the efficiency and practicality of the ECC will be also significant in developing the MSC, and will be introduced in the next two chapters.

Here, the ECCA describes a very efficient implementation of the group operation in the Galois field GF(2^{155}). It is easily to see that a properly chosen elliptic curve over GF(2^{155}) means using smaller numbers, but offers the same degree of security as working modulo a 512 bit prime p in the RSA or the ElGamal cryptosystem.

3. The Principal of Mathematics in the Public Key Cryptosystem

Abstract: All of the properties and efficiencies in a cryptosystem source from the methods of computation and the architecture of algorithms in cryptography. Both of these refer to the mathematical principals of the cryptography algorithm. It is necessary to know the mathematics of principals for exploring the efficiencies of the public key cryptography algorithm. In this paper, the description will be limited to the public key cryptography algorithms corresponding to the three cryptosystems discussed in the previous chapter. It begins with the factoring algorithm used for RSA cryptosystem, then the discrete logarithm algorithm used for ElGamal cryptosystem will be described, and finally the elliptic curve algorithm used for the Elliptic Curve cryptosystem (**ECCs**) will be introduced.

3.1 The Principle of the Factoring Algorithm

This section presents some basic facts concerning primality and factoring, especially from the point of view of RSA. The RSA cryptosystem was invented in 1977 by Rivest-Shamir-Adleman [10], and was the first realization of Diffie-Hellman's abstract model for public key cryptography that we introduced in Section 2.1. The security of the RSA cryptosystem is based on the assumption that is much easier for someone to find two extremely large prime numbers p and q than it is to recover them, if only their product n is known. In generating the RSA algorithm, it is necessary to generate large random primes. In practice large random numbers will be generated, and then be tested for primality. A primality test is a criterion for a number n not to be a prime. If n "passes" a primality test, then it may be a prime. On the other hand, if n fails any primality test, then it is definitely not a prime. Therefore, it leaves a difficult problem: find the prime factors of n.

It is necessary to say that the primality testing of algorithms must run in polynomial time in order for the receiver to find primes p and q effectively. In this section the probabilistic polynomial time algorithm such as the Miller-Rabin algorithm will be presented initially,

then a method of factorization such as the Quadratic Sieve algorithm will be introduced.

3.1.1 Primality Test

A few definitions from number theory have to be introduced, and these are useful in the primality test to be introduced later.

Fermat's Theorem: Let p be a prime and suppose the greatest common division gcd (a, p) = 1, then $a^{p-1} = 1 \pmod p$.

According to Fermat's theorem, if n is prime, then for any b such that gcd.(b, n) = 1,

$$b^{n-1} = 1 \pmod n. \qquad (3.1\text{-}1)$$

If n is not prime, it is still possible (but not very likely) that equation (3.1-1) holds for a number of choices of b.

Pseudoprime: If n is an odd composite number and b is an integer such that gcd(n, b) = 1 and $b^{n-1} = 1 \pmod n$ holds, then n is called a *pseudoprime* to the base b.

Example: The number n = 91 is a *pseudoprime* to the base b = 3, because $3^{90} = 1 \pmod{91}$. However, 91 is not a *pseudoprime* to the base 2, because $2^{90} = 64 \pmod{91}$. If it is had not already known that 91 is composite, the fact that $2^{90} \neq 1 \pmod{91}$ would shown that it is.

The Legendre symbols: Let a is an integer and p > 2 a prime. The *Legendre symbol* (a/p) is defined to equal 0, 1 or –1, as follows:

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{if } p \mid a; \\ 1, & \text{if a is a quadratic residue mod p;} \\ -1, & \text{if a is a non-residue mod p.} \end{cases}$$

Thus, the *Legendre symbol* is simply a way of identifying whether or not an integer is a quadratic residue modulo p.

$$\left(\frac{a}{p}\right) = a^{(p-1)/2} \pmod{p}.$$

Proof. If a is divisible by p, then both sides are = 0 (mod p). Suppose p □ a. By Fermat's Theorem, in $\mathbf{F}_p$ the square of $a^{(p-1)/2}$ is 1, so $a^{(p-1)/2}$ itself is ±1. Let g be a generator of $\mathbf{F}_p^*$ (the multiplication group of $\mathbf{F}_p$), and let $a = g^j$. The a is a residue if and only if j is even. And $a^{(p-1)/2} = g^{j(p-1)/2}$ is 1 if and only if j(p-1)/2 is divisible by p-1, i.e., if and only if j is even. Thus, both sides of the congruence in the proposition are ±1 in $\mathbf{F}_p$, and each side is +1 if and only if j is even.

The Jacobi symbol, general to composite number, is as Legendre symbol.

The Jacobi symbols: Let a is an integer, and let n be any positive odd number. Let $n = p_1^{\alpha_1} \ldots p_r^{\alpha_r}$ be the prime factorization of n. Then the *Jacobi symbol* (a/n) is defined as the product of the *Legendre symbols* for the prime factors of n:

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{\alpha_1} \ldots \left(\frac{a}{p_r}\right)^{\alpha_r}.$$

A word of warning is in order here. If (a/n) = 1 for n composite, it is not necessarily true that a is a square modulo n. For example, (2/15) = (2/3)(2/5) = (-1)(-1) = 1 (mod 15). Now the Proposition to the *Jacobi symbol* [32] is generalized.

Proposition: For any positive odd n, the congruence $\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}$ is holding.

(The reader interested in the proof can refer to Appendix A-3)

Euler Pseudoprimes: Odd composite numbers n satisfying (3.1-2), for some b with (b, n) = 1, is called an *Euler Pseudoprimes* to the base b. Because (3.1-2) implies (3.1-1), an *Euler pseudoprime* to the base b is also a pseudoprime to the base b. The converse is not true, i.e. 91 is a pseudoprime but not an Euler pseudoprime to the base 3. Because (3.1-1) is satisfied but $3^{45} = 27 \pmod{91}$, implying that (3.1-2) is not satisfied. The number 91 is an Euler pseudoprime to base 10,

since $10^{45} = 10^3 = -1 \pmod{91}$. From the above two results, we can obtain an efficient probabilistic test for whether or not a large odd integer n is a prime, which starting with the following definition.

Lemma 3.1 (Lemma 4.4 in [72]): If n is an odd composite number, then at most half of the integers b with $1 \le b \le n$ and $(b, n) = 1$ satisfy (3.1-2).

Proof: Firstly suppose that a b' does not satisfy (3.1-2) (for b = b'). If assume the square p^2 of a prime p divides n, then it results $b' = 1 + n/p$. Thus, $(b'/n) = 1$, but the left side of equation (3.1-2) is not congruent to 1 (mod n), since p does not divide (n-1)/2.

Secondly, assume that n is a product of distinct primes and p is one of them. Choosing any quadratic non-residue (refer to Appendix I) s modulo p and let b' with $1 \le b' < n$, satisfy the congruence

$$b' = s \pmod{p},$$
$$b' = 1 \pmod{(n/p)}.$$

This b' can found by the Chinese Remainder Theorem (see Appendix C). So $(b'/n) = -1$ but

$$b'^{(n-1)/2} = 1 \pmod{n/p},$$

Bring about $$b'^{(n-1)/2} \neq -1 \pmod{n}.$$

To set up a b' which does not satisfy the equation (3.1-2), let b_i with $1 \le i \le t$ be all of the integers satisfying (3.1-2) (as well as the condition $1 \le b' < t$ where $(b', n) = 1$). Where t is threshold with $n^s < t < n^e$, s and e is signature key and encryption key respectively [72]. The numbers

$$u_i = b'b_i \pmod{n} \qquad 1 \le i \le t,$$

are all distinct and satisfy $1 \le u_i < n$ and $(u_i, n) = 1$. If u_i satisfies (3.1-2), then

$$(b'^{(n-1)/2})(b_i^{(n-1)/2}) = (b'/n)(b_i/n) \pmod{m}.$$

Since b_i satisfies (3.1-2), it can be deduced further that

$$b'^{(n-1)/2} = (b'/n) \pmod n$$

This is a contradiction with the facts that b' does not satisfy (3.1-2). Hence, none of the numbers u_i satisfies (3.1-2). There are as many of them as there are numbers b_i.

The Miller-Rabin primality test

The Miller-Rabin test is based on the notion of a "strong pseudoprime", which will be introduced below. But some number theoretic facts will be given without proofs. If the reader is interested more information, please refer to [32].

Suppose that n is a pseudoprime to the base b, i.e., $b^{n-1} = 1 \pmod n$. The idea of the strong pseudoprime criterion is to extract successive square roots of the congruence (3.1-1) and check whether the first number $\neq 1$ on the right side of the congruencies thus obtained is actually equal to -1. For example if b is raised to the ((n-1)/2)th, ((n-1)/2)th,..., ((n-1)/2^s)th powers (where $(n-1)/2^s$ is odd), then the first residue class is either 1 or -1if n is prime, because ± 1 are the only square roots of 1 modulo n. If n fails this test, the first number different from 1 equals -1, but n is composite, then n is referred to as a strong pseudoprime to the base b. In practice, one can set $n-1 = 2^s t$ with t odd, then computing $b^t \pmod n$, if that is not equal to -1, squaring to get $b^{2t} \pmod n$, then squaring again to get $b^{2^2 t} \pmod n$, etc., until one get the residue 1. Before getting 1 the n has be known as composite, or the -1 must be got.

Definition: Assume n is an odd composite number, and write $n-1 = 2^s t$ with t odd. Choose a number b with $1 \le b < n$ and $(b, n) = 1$. If n and b satisfy the condition either

$$b^t = 1 \pmod n \text{ or } b^{2^r t} = -1 \pmod n \qquad (3.1\text{-}3)$$

for some r with $0 \le r < s$, then n is called a strong pseudoprime to the base b.

Lemma 3.2 (Lemma 4.5 in [72]): If n is an odd composite integer, then n is a strong pseudoprime to the base b for at most 25% of all b's satisfying $1 \leq b < n$. (Proof can be found in [32]).

In the Miller-Rabin primality test, assume $n-1 = 2^s t$ with odd t, and choose a random integer b with $0 < b < n$. To determine whether a large positive odd integer n is prime or composite, first compute $b^t \pmod n$. If the result is ± 1, then n passes the test (3.1-3) for the particular b; repeat the procedure for another b. Otherwise square $b^t \pmod n$, then square that modulo n, and so on, until we get -1. If the -1 is obtained, then n passes the test. However if the -1 is never obtained, i.e., the $b^{2^{r+1}} = 1 \pmod n$ while $b^{2^r} \neq -1 \pmod n$, then n fails the test and n is composite. If n passes the test (3.1-3) for all random choices of b (suppose we try k different bases b), then by Lemma 3.2 the n has at most a 1 out of 4k chance of being composite. Because if n is composite, then at most ¼ of the bases with $0 < b < n$ satisfy (3.1-3).

In practice one does not have to check through a large number of bases b in order to be almost sure that n is a prime. If it is a strong pseudoprime to each of these bases, i.e., the four bases 2, 3, 5, 7, only one composite number $n < 2.5 \cdot 10^{10}$, namely $n = 3215031751$, is a strong pseudoprime to each of these four bases.

3.1.2 The Quadratic Sieve algorithm

The quadratic sieve algorithm for factoring large integers, developed by Pomerance in the early 1980's [32], was more successful than any other method for factoring integers n of general type which have no prime factor of order of magnitude significantly less than $\sqrt{n}$. An important difference between algorithms for primality testing and ones for factorization is that primality testing algorithms can tell if a given integer is composite or prime. Factoring algorithms, however, give the actual factors when the integer is composite.

There are a huge number of papers on factoring algorithms. Three algorithms that are most effective on very large number N as follows:

(1) The Quadratic Sieve algorithm,

(2) The Elliptic Curve algorithm,

(3) The Continued Fraction algorithm.

All of these methods can factor N with the running time:

$$L(N) = \exp\{(1 + o(1)\sqrt{\ln n \ln\ln n}\}. \qquad (3.1\text{-}3A)$$

Although no one seems to have any idea how to obtain even heuristic lower bounds for factoring, most of the papers believe the conjecture that L(N) is in fact the true complexity of factoring. Here it is not necessary to introduce all of the three algorithms, since all of them having similar features. Firstly the quadratic sieve algorithm only will be introduced, then an example given to illustrate the method.

The Quadratic Sieve algorithm:

The Quadratic sieve algorithm is a variant of the Factor Base approach (see Appendix A-2). As the factor base F someone takes the set of all primes $p \leq B$ (where B is some bound to be chosen in some optimal way) such that n is a quadratic residue mod p, i.e., $(n/p) = 1$ for p odd, and $p = 2$ is always included in F. The set of integers S in which one look for F-numbers (recall that a F-number is an integer divisible only by primes in F) will be the same set that someone used in *Fermat factorization*[1], namely:

$$S = \{t2 - n \mid [\sqrt{n}] + 1 \leq t \leq [\sqrt{n}] + A\}$$

For some suitably chosen bound A.

The main idea of the method is that, instead of taking each $s \in S$ one by one and dividing it by the prime $p \in F$ to see if it is an F-number. One takes each $p \in F$ one by one and examines divisibility by p (and powers of p) simultaneously for all of the $s \in S$. The word

[1] It is a way to factor a composite number n that is efficient if n is a product of two integers which are close to one another. This method, called "*Fermat factorization*", is based on the fact that n is then equal to a

"sieve" refers to this idea, which one can use to make a list of all primes $p \leq A$. For example, to list the primes ≤ 1000 one takes the list of all integers ≤ 1000. Then for each p = 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, one discards all multiples of p greater than p — one "lets them fall through a sieve which has holes spaced a distance p apart" — after which the numbers that remain are the primes. The basic steps are summarized as follows (the interested reader can refer to [32, 33, 78] for details), then give an example.

Step 1. Suppose n is an odd composite integer, A and B are bounds, both of order of magnitude roughly are

$$\exp((\log n \log\log n)^{1/2}).$$

Generally, A should be larger than B, but not larger than a small power of B, e.g., $B < A < B^2$. This function $\exp((\log n \log\log n)^{1/2})$ has an order of magnitude intermediate between polynomial in logn and polynomial in n. If $n \approx 10^6$, then $L(n) \approx 400$.

Step 2. For $t = [\sqrt{n}] + 1, [\sqrt{n}] + 2, \ldots, [\sqrt{n}] + A$, make a column listing the integers $t^2 - n$. For each odd prime $p \leq B$, first check that $(n/p) = 1$; if not, then throw that p out of the factor base. Assuming that p is an odd prime such that n is a quadratic residue mod p, solve the equation $t^2 = n \pmod{p^\beta}$ for $\beta = 1, 2, \ldots$, (the details of this method refer to Exercise 20 of II.2 in [32]). One takes the increasing values of β until to find that there is no solution t which is congruent modulo p^β to any integer in the range $[\sqrt{n}]+1 \leq t \leq [\sqrt{n}]+A$. Let β be the largest integer such that there is some t in this range for which $t^2 = n \pmod{p^\beta}$. Let t_1 and t_2 be two solutions of $t^2 = n \pmod{p^\beta}$ with $t_2 = -t_1 \pmod{p^\beta}$, t_1 and t_2 are not necessarily in the range from $[\sqrt{n}] + 1$ to $[\sqrt{n}] + A$.

Step 3. Still with the same value of p, run down the list of $t^2 - n$ from step 2. In a column under p, one puts a 1 next to all values of $t^2 - n$ for which t differs from t_1 by a multiple of p, and changes the 1 to a 2 next to all values of $t^2 - n$ for which t differs

difference of two squares, one of which is very small.

from t_1 by a multiple of p^2, and changes the 2 to a 3 next to all values of $t^2 - n$ for which t differs from t_1 by a multiple of p^3, and so on until p^β. Then do the same with t_1 replaced by t_2. The largest integer that appears in this column will be β.

Step 4. As someone goes through the procedure in step 3, each time one puts down a 1 or changes a 1 to a 2, a 2 to a 3, etc., divides the corresponding $t^2 - n$ by p and keeps a record of what's left. In the column $p = 2$, if $n \neq 1 \pmod 8$, then simply put a 1 next to the $t^2 - n$ for t odd and divide the corresponding $t^2 - n$ by 2. If $n = 1 \pmod 8$, then solve the equation $t^2 = n \pmod{2^\beta}$ and proceed exactly as in the case of odd p (except that there will be 4 different solutions t_1, t_2, t_3, t_4, modulo 2^β if $\beta \geq 3$).

Step 5. When you finish with all primes $\leq B$, throw out all of the $t^2 - n$ except for those which have become 1 after division by all the powers of $p \leq B$. One will has a table of the form in Example 9 of §V.3 in [32], in which the column labeled b_i will have the values of t, $[\sqrt{n}] + 1 \leq t \leq [\sqrt{n}] + A$, for which $t^2 - n$ is a F-number, and the other columns will correspond to all values of $p \leq B$ for which n is a quadratic residue.

The interested reader can refer to §V.3 in [32] for more details.

Example 3.1 (Example in [32]): Let us try to factor n = 1042387, taking the bound P = 50 (the set of all primes $p \leq P$) and bound A = 500 (a list of all primes $p \leq A$) with $P < A < P^2$. Here $[\sqrt{n}] = 1020$. The factor base consists of the 8 primes {2, 3, 11, 17, 19, 23, 43, and 47} for which 1042387 is a quadratic residue. Since $n \neq 1 \pmod 8$, the column corresponding to p = 2 alternates between 1 and 0, with a 1 beside all odd t, $1021 \leq t < 1520$.

Now it is starting to describe in detail how to form the column under p = 3. One wants a solution

$$t_1 = t_{1,0} + t_{1,1}\cdot 3 + t_{1,2}\cdot 3^2 + \cdots + t_{1,\beta-1}\cdot 3^{\beta-1} \text{ to } t_1^2 = 1042387 \pmod{3^\beta},$$

where $t_{1,j} \in \{0, 1, 2\}$ (for other solution t_2, one can take $t_2 = 3^{\beta} - t_1$). It is clear one can take $t_{1,0} = 1$. (For each of the 8 primes the first step: solving $t_1^2 = 1043287 \pmod p$, it can be done quickly by trial and error).

Next, it works on modulo 9: $(1+ 3t_{1,1})^2 = 1043287 = 7 \pmod 9$,

$6t_{1,1} = 6 \pmod 9$,

$2t_{1,1} = 2 \pmod 3$,

thus $t_{1,1} = 1$,

And modulo 27: $(1 + 3 + 9\, t_{1,2})^2 = 1043287 = 25 \pmod{27}$,

$16 + 18\, t_{1,2} = 25 \pmod{27}$,

$2t_{1,2} = 1 \pmod 3$,

so $t_{1,2} = 2$.

Then modulo 81: $(1 + 3 + 18 + 27\, t_{1,3})^2 = 1043287 = 79 \pmod{81}$,

which leads to the $t_{1,3} = 0$.

Continuing until 3^7, one find the solutions: $t_1 = (210211)^3 \pmod{3^7}$,

$t_2 = (2012012)^3 \pmod{3^7}$.

However there is no t between 1021 and 1520, which equals to t_1 or $t_2 \pmod{3^7}$. Thus, one has $\beta = 6$, and one takes $t_1 = (210211)^3 = 589 = 1318 \pmod{3^6}$ and $t_2 = 3^6 - t_1 = 140 = 1112 \pmod{3^5}$ (note that there is no number in the range from 1021 to 1520 which is $= t_2 \pmod{36}$).

Now one can construct "sieve" for the prime 3. Starting from 1318, it takes jumps of 3 down until 1021 is reached and up until 1519 is reached. Each time it can put a 1 in the column. It can divide the corresponding $t^2 - n$ by 3, and can record the result of the division (Actually, for t odd, the number divided by 3 is half of $t^2 - n$, since 2 has already divided $t^2 - n$ when the column of alternating 0's and 1's under 2 is formed.). Then one can do the same with jumps of 9, each time changing the 1 to 2 in the column under 3, dividing the quotient of $t^2 - n$ by another 3, and recording the result. One goes through the analogous procedure with jumps of 27, 81, 243, and 729 (there is no jump possible for 729, one nearly does not change the 5 to 6 next to 1318 and divide the quotient of 13182

(1043287) by another 3). Finally, one goes through the same steps with $t_2 = 1112$ instead of $t_1 = 1318$, this time stopping with jumps of 243.

After passing through this procedure for the remaining 6 primes in factoring base, there is a 500 × 8 array of exponents, each row corresponding to a value of t between 1021 and 1520. Now one throw out all rows for which $t^2 - n$ has not been reduced to 1 by repeated division by powers of p as the table (3-1) formed. I.e., the rows for which $t^2 - n$ is a B-number is only taken. The following table 3-1 is left for the n = 1043287 (the blank spaces denote zero exponents):

t	t^2 - n	2	3	11	17	19	23	43	47
1021	54	1	3	-	-	-	-	-	-
1027	12342	1	1	2	1	-	-	-	-
1030	18513	-	2	2	1	-	-	-	-
1061	83334	1	1	-	1	1	-	1	-
1112	194157	-	5	-	1	-	-	-	1
1129	232254	1	3	1	1	-	1	-	-
1148	275517	-	2	3	-	-	1	-	-
1175	338238	1	2	-	-	1	1	1	-
1217	438702	1	1	1	2	-	1	-	-
1390	889713	-	2	2	-	1	-	1	-
1520	1268013	-	1	-	1	-	2	-	1

Table 3-1 the array of exponents

Now it can look for relations modulo 2 between the rows of this matrix. Moving down from the first row, one looks for a subset of the rows, which sums to an even number in each column. The first such subset found here is the first three rows, the sum of which is twice the row 1 3 2 1 - - -. Therefore, one can obtain the congruence:

$$(1021 \cdot 1027 \cdot 1030)^2 = (2 \cdot 3^3 \cdot 11^2 \cdot 17)^2 \pmod{1043287}.$$

But this is not so lucky: the two numbers being equal in the above congruence are both = 111078 (mod 1043287). So it is necessary to continue down the matrix. Finally, one notices that the last row, corresponding to the last value of t, is dependent on the earlier rows. More precisely, it is equal modulo 2 to the fifth row. It gives that

$$(1112 \cdot 1520)^2 = (33 \cdot 17 \cdot 23 \cdot 47)^2 \pmod{1043287},$$

thus $647853^2 = 496179^2 \pmod{1043287}$,

the factor is that $\gcd(647853 - 496179, 1043287) = 1487$.

3.2 The Principal of the Discrete Logarithm Algorithm

The RSA cryptography based on the factoring algorithm that is infeasible to factor the product of two large primes has been introduced above. The ElGamal cryptography based on the discrete logarithm algorithm has similar powerful "one way" function with the RSA has it. When calculating with the real numbers, exponentiation (calculating b^x to a prescribed value) is not significantly easier than the inverse operation (calculating $\log_b x$ to a prescribed value). When the work is done in a finite field $\mathbf{F}_q$ (with the group operation of multiplication). One can compute b^x for large x rather rapidly (in polynomial time in log x), because of the repeated-squaring method. But if an element which is of the form b^x (suppose that the "base" b is fixed) is given, how can $x = \log_b y$ be quickly computed? This is called the "discrete logarithm problem". Thus, finding discrete logarithm is in fact much more difficult than raising to a power in a large finite field.

Coppersmith's algorithm to fine discrete logarithms, namely the Index Calculus algorithm, works in two steps. The first step (a preprocessing step) is obtaining a system of linear equations that are satisfied by the discrete logarithms of certain group elements belonging to a set called a factor base and solving the set of equations to determine the discrete logarithm of the elements of the factor base. The second step is computing any desired logarithm from the pre-computed library of logarithms for the factor base. Finally an example is given to illustrate the algorithm.

3.2.1. Algorithms for Finding Discrete Logarithms in Finite Fields

This method for computing logarithms in a cyclic group by factoring the order of the group is called Pohlig-Hellman method [26]. It supposes that all of the prime factors of q–1 is small. In this case it is said that q-1 is "smooth". There is a fast algorithm for finding the discrete logarithm of an element $y \in \mathbf{F}_q$ to the base b under this assumption. For simplicity, it can also be supposed that b is a generator of $\mathbf{F}_q$.

First, for each prime p dividing q-1, one can compute the p-th roots of unity $r_{p,j} = b^{j(q-1)/p}$ for j = 0, 1, …, p-1 by the repeated squaring method to raise b to a large power. With the table of $\{r_{p,j}\}$ determined it is ready to compute the discrete logarithm of any $y \in \mathbf{F}_q$ (note: if b is fixed, this first computation needs only be done once, after which the same table is used for any y). The goal is to find x ($0 \le x < q - 1$), such that $b^x = y$. If $q - 1 = \prod_p p^{\alpha}$ is the prime factorisation of $q - 1$, then it suffices to find x (mod p^{α}) for each p dividing $q - 1$. This x is uniquely determined by using the algorithm in the proof of the Chinese Remainder Theorem. Now it can start to fix a prime p dividing $q - 1$, and to show how to determine x (mod p^{α}).

Suppose that $x = x_0 + x_1 p + \cdots + x_{\alpha-1} p^{\alpha-1} \pmod{p^{\alpha}}$ with $0 \le x_i < p$. To find x_0, one computes $y^{(q-1)/p}$, and gets a p-th root of 1, since $y^{q-1} = 1$. Since $y = b^x$, it follows that

$$y^{(q-1)/p} = b^{x(q-1)/p} = b^{x0(q-1)/p} = r_{p,x0}.$$

Thus, compare $y^{(q-1)/p}$ with the $\{r_{p,j}\}_{0 \le j < p}$ and to set x_0 equal to the value of j for which

$$y^{(q-1)/p} = r_{p,j}.$$

Next to find x_1, one replaces y by $y_1 = y/b^{x0}$. Then y_1 has discrete logarithm:

$$x - x_0 = x_1 p + \cdots + x_{\alpha-1} p^{\alpha-1} \pmod{p^{\alpha}}.$$

Since y_1 is a p-th power, it has $y_1^{(q-1)/p} = 1$ and

$$y_1^{(q-1)/p^2} = b^{(x-x_0)(q-1)/p^2} = b^{(x_1+x_2p+\cdots)(q-1)/p} = b^{x_1(q-1)/p} = r_{p,x_1}.$$

So one compares $y_1^{(q-1)/p^2}$ with $\{r_{p,j}\}$ and set x_1 equal to the value of j for which $y_1^{(q-1)/p^2} = r_{p,j}$. Continue in the same way to find all $x_0, x_1, \ldots, x_{\alpha-1}$. Set

$$y_i = y/b^{x_0+x_1p+\cdots+x_{i-1}p^{i-1}} \quad \text{(where } i = 1, 2, \ldots, \alpha\text{-}1),$$

which has discrete logarithm congruent mod $p\alpha$ to $x_ip^i + \cdots + x_{\alpha-1}p^{\alpha-1}$. Since yi is a pi-th power, thus

$$y_i^{(q-1)/pi} = 1$$

and
$$y_i^{(q-1)/pi+1} = b^{(x_i+x_{i+1}p+\cdots)(q-1)/p} = b^{x_i(q-1)/p} = r_{p,j}.$$

So one can set x_i equal to the value of j for which $y_i^{(q-1)/pi+1} = r_{p,j}$, and can get x mod p^α. After doing this for each p|q-1, One can find x by using the Chinese Remainder Theorem.

This algorithm is efficient, when all of the primes dividing q-1 are small. But clearly the computation of the table of $\{r_{p,j}\}$ and the comparison of the $y_i^{(q-1)/pi+1}$ with this table will take a long time if q-1 is divisible by a large prime.

Example 3.2 (Example 4 in [32] page103): To find the discrete logarithm of 28 to the base 2 in $\mathbf{F}_{37}$ (2 is a generator of $\mathbf{F}_{37}$) using the Silver-Pohlig-Hellman algorithm.

Solution: First, write $37-1 = 2^2 \cdot 3^2$. Calculating $2^{18} = 1 \pmod{37}$, so $r_{2,0} = 1$, $r_{2,1} = -1$ (for p = 2, always $\{r_{2,j}\} = \{\pm 1\}$). Then $2^{36/3} = 26$, $2^{2\cdot 36/3} = 10 \pmod{37}$, so $\{r_{3,j}\} = \{1, 26, 10\}$. Let $2^x = 28 \pmod{37}$. Firstly one takes p = 2 and finds x (mod 4), which can write as $x_0 + 2x_1$. One calculates $28^{36/2} = 1 \pmod{37}$, hence gets $x_0 = 0$. Then one computes $28^{36/4} = -1 \pmod{37}$, hence $x_1 = 1$. Next one takes p = 3 and finds x (mod 9), which one write as $x_0 + 3x_1$ (for each p, the x_i are defined differently). To find x_0, one calculates $28^{36/3} = 26 \pmod{37}$, so $x_0 = 1$. Then

one computes $(28/2)^{36/9} = 14^4 = 10 \pmod{37}$, thus, $x_1 = 2$, so $x = 1 + 2 \cdot 3 = 7 \pmod 9$. The remainder is to find the unique x (mod 36) such that $x = 2 \pmod 4$ and $x = 7 \pmod 9$. It is $x = 34$. Thus, $2^{34} = 28 \pmod{37}$.

3.2.2. The Index-calculus algorithm

Index-calculus algorithm [35] has three basic stages. In the first stage, a set of linear equations giving relations for the elements of the factor base B is generated. In the second stage, the set of linear equations is solved, to determine the discrete logarithms of the elements of factor base B. Both stages comprise a precomputation phase. In the third stage, the calculation of individual logarithms is done using the previously computed values of the discrete logarithms for the factor base, from which $x = \log_a b$ can be recovered.

First stage: The Index Calculus algorithm works well in finite fields GF(q). The reason is that the factoring is easy for the integers, if all of the prime factors of an integer I are less than a given bound u, and one can completely factor I with at most u + log I divisions.

Let $q_1, \ldots, q_m$ to be the first m primitive element of GF(q), and let GF be generated by g, of order n. One randomly chooses an integer $\beta \in [1, n]$ and computes the least positive integer e with $e = g^{\beta} \pmod q$. Then one can try to factor e as a product of the first m primes, using division by these primes. If e factors by this method, a relation of the form

$$\prod_{j=1}^{m} q_j^{\alpha_{ij}} = g^{\beta_i} \tag{3.2-1}$$

is obtained. It is clearly shown that if the bigger m is, the chance that e will factor as a product of the first m primes is greater. But as the m increases, the work to factor residues and to solve the equations in the second stage increases. Now one introduces an argument to balance these constraints in order to optimize the running time of the algorithm as follows:

Suppose β is chosen from a uniform distribution, then the probability that e will factor as a product of the first m primes is $\Gamma(q,q_m)/q$, where $\Gamma(x, y)$ is defined to be the number of positive integers $\leq x$ that have no prime factors exceeding y. The asymptotic behavior of the function Γ has been extensively studied, and in particular it is known that [36]

$$\Gamma(x, y) = x \exp((-1 + o(1)) v \log v),$$

where $v = \log x/\log y$, for $v \to \infty$ and $y \geq \log^2 x$. If $q_m \approx L(q)^c$ is chosen, where c is a constant and

$$L(q) = \exp(\sqrt{\log q \log\log q}),$$

then the probability that e has all prime factors among the first m primes is $L(q)^{-1/(2c)+o(1)}$. Therefore one can expected to generate the equation (3.2-1) after trying $L(q)^{1/(2c)+o(1)}$ values of β, and to generate 2m such equations of (3.2-1) should take about

$$2mL(q)^{1/(2c)+o(1)} = L(q)^{c+1/(2c)+o(1)}$$

values of β. If one uses trial division to do the factoring, then for each β it will take at most m + log q divisions to decide whether an equation (3.2-1) is given. Therefore one has a total running time for the generation of relations in the first stage of $L(q)^{2c+1/(2c)+o(1)}$. Once 2m equations are generated (or any number slightly larger than m), it is reasonable to expect that the corresponding set of 2m equations in m unknowns should have full rank, thus the $\log_g q_j$ (which denotes the discrete log of $q_j \in GF(q)$) is solved (here the full rank means full column rank modulo p for every prime p dividing q-1).

Second Stage: The problem of solving a system of linear congruencies does not present any great difficulty, but there are a few points that deserve comment. For example, consider the problem of solving the equations:

$$3x_1 + 2x_2 = 0 \pmod{30},$$
$$5x_1 - 3x_2 = 2 \pmod{30}. \qquad (3.2\text{-}2)$$

Note that none of the coefficients are invertible modulo 34, and it is impossible to add a multiple of one row to another in such a way as to introduce a zero entry (to use the

Gaussian elimination). However, there is a unique solution $x_1 = 16 \pmod{30}$, $x_2 = 6 \pmod{30}$.

Here it starts to describe a method for solving a system of equations of linear congruencies. Suppose one want to solve $\mathbf{B}x = \beta \pmod{q-1}$, where $\mathbf{B}$ is $I \times m$ with rank ($\mathbf{B} \bmod p$) = m for every prime p dividing q-1, there are a few steps as follows. First factor q-1 as a product of prime powers. Then the equations modulo p (for every prime p dividing q-1) are solved by Gaussian elimination, then the solution modulo p is lifted to a solution modulo the power of p dividing q-1 by Hensel's method. Finally using the Chinese Remainder Theorem combines all the solutions for a solution modulo q-1. However, the factorization of q-1 is generally not needed to solve the equations and should probably be avoided. One choice is to perform as if $\mathbf{Z}/(q-1)\mathbf{Z}$ is a field and attempt to perform Gaussian elimination in the usual manner. However, this can break down if there is a column in which no entry is invertible modulo q-1 (as in equation (3.2-2)). The Euclidean algorithm or Hensel's method can recover from this by using the Chinese Remainder Theorem, but the details are somewhat tedious so another strategy is used.

In standard Gaussian elimination, the jth column of the matrix would be searched to find an entry that is invertible modulo q-1. The rows are exchanged to bring it into the jj location, and a multiple of the jth row be added to the rows below it to introduce zero entries. An alternative procedure is searching the jth column to find an entry that is nonzero and exchanging rows to bring it into the jjth location. Then to introduce a zero into the ijth location, one uses the extended Euclidean algorithm ([37] §4.5.2) to find integers g, e, and k for which

$$g = \gcd(\alpha_{ij}, \alpha_{jj}) = e\alpha_{ij} + k\alpha_{jj}.$$

Then the row i is replaced by $(\alpha_{ij}/g)\cdot(\text{row j})-(\alpha_{jj}/g)\cdot(\text{row i})$, and the row j of the matrix B is replaced by $e\cdot(\text{row i}) + k\cdot(\text{row j})$. These operations on the system preserve the solution set and have the effect of replacing the ijth and jjth entries by 0 and g respectively.

If this method would be used, the solution of the system of equations will need $O(m^3)$ operations modulo q-1, and $O(m^2)$ applications of the Euclidean algorithm. Therefore the

total expected time for stages one and two to find $\log_g p_j$ is

$$L(q)^{(2c+1/(2c))+o(1)} + L(q)^{3c+o(1)} \qquad (3.2\text{-}3)$$

operations on integers of size q. Due to the fact each such operation can be done in $L(p)^{o(1)}$ bit operations, the same number of bit operations can will be used. If the c=1/2, the running time for the first two stages is $L(p)^{2+o(1)}$ bit operations.

Stage three: This stage starts to compute individual Discrete Logarithms. In order to compute $\log_g \alpha$, suppose one choices a random integer r, to compute $e = \alpha g^r \pmod q$, and to see if e factors as a product of the first m primes. If $e = \prod_{j=1}^{m} q_j^{r_j}$ is obtained, then this implies

$$\log_g \alpha = \sum_{j=1}^{m} r_j \log_g q_j - r \pmod{(q-1)}.$$

This stage can be analyzed using the same way as before, giving an expected running time of $L(q)^{(c+1/2c)+o(1)}$, or $L(q)^{3/2+o(1)}$ (if c=1/2).

There are variations of the Index Calculus method due to Coppersmith, Odlyzko, and Schroeppel [35], which are conjectured to be faster. They describe three such algorithms that are called the linear sieve, the Residue List sieve, and the Gaussian Integer method. In each of these methods there is a heuristic analysis that suggests a running time of $L(q)^{1+o(1)}$ for the first two stages, followed by a running time of $L(q)^{1/2+o(1)}$ to compute individual logarithms in the third stage. Here details for the Gaussian Integer method will not be described. The interested reader can refer to [35].

Example (Example 5.4 in [41]): Here is a small, very artificial, example to illustrate the two steps in the algorithm.

Suppose q = 10007 and a = 5 is the primitive element used as the base of logarithms module q. Suppose B = {2,3,5,7} is taken as the factor base. Due to $\log_5 5 = 1$, therefore there are three logs of factor base elements to be determined. Suppose the "lucky" exponents that might be chosen are 4063, 5136 and 9865. If x = 4063, then the

computation is that

$$5^{4063} \pmod{10007} = 42 = 2 \times 3 \times 7.$$

This leads to the congruence

$$\log_5 2 + \log_5 3 + \log_5 7 = 4063 \pmod{10006}.$$

Similarly, since

$$5^{5136} \pmod{10007} = 54 = 2 \times 3^3,$$

$$5^{9865} \pmod{10007} = 189 = 3^3 \times 7,$$

two more congruencies can be obtained as follows,

$$\log_5 2 + 3\log_5 3 = 5136 \pmod{10006},$$

$$3\log_5 3 + \log_5 7 = 9865 \pmod{10006}.$$

Now there are three unknowns in three congruencies, and there are three unique solutions (mod 10006), namely $\log_5 2 = 6578$, $\log_5 3 = 6190$ and $\log_5 7 = 1301$. Now suppose that one wishes to find $\log_5 9451$. Let's choose the "random" exponent $r = 7736$, and compute

$$9451 \times 5^{7736} \pmod{10007} = 8400.$$

Since $8400 = 2^4 \times 3^1 \times 5^2 \times 7^1$ factors over B, therefore

$$\begin{aligned}\log_5 9451 &= 4\log_5 2 + \log_5 3 + 2\log_5 5 + \log_5 7 - r \pmod{10006} \\ &= 4 \times 6578 + 6190 + 2 \times 1 + 1301 - 7736 \pmod{10006} \\ &= 6057.\end{aligned}$$

To verify, it can be checked by computing $5^{6057} = 9451 \pmod{10007}$. The heuristic analyses of various versions of the algorithm have been done. Under reasonable assumptions, the asymptotic running time of the precomputation phase is

$$o\left(e^{(1+o(1))\sqrt{\ln q \ln\ln q}}\right), \tag{3.2-4}$$

and the time to find an individual discrete log is

$$o\left(e^{(1/2+o(1))\sqrt{\ln q \ln\ln q}}\right). \tag{3.2-5}$$

3.3 The Principal of the Elliptic Curve Algorithm

In recent years the elliptic curves in number theory, more precisely the theory of elliptic curves defined over finite fields, has found application in cryptography. The basic reason is that the elliptic curve over finite fields provides an example of Abelian group, which has advanced functions because of their special structure. For the Factoring and Logarithm algorithms used with the multiplicative groups of fields, elliptic curve provides a natural analog of these algorithms, but has the advantage that one has more flexibility in choosing an elliptic curve than a finite field.

Here the description includes only the minimal amount of background necessary to understand the applications to cryptography. It starts by presenting the basic definitions and facts about the Geometry of Elliptic Curve including the Group Law on the Elliptic Curve. Then the Arithmetic operations of Elliptic Curve in field $\mathbf{F}_q$ will be introduced. Finally it gives an example and concretizes descriptions at the Practice of the Elliptic Curve Cryptography Algorithm as follows.

3.3.1 The Geometry of Elliptic Curves

Elliptic Curve Cryptography Algorithm (**ECCA**) is a principal object of study in this report. The Group law underlies the general principle in the ECCA. It is necessary to have a thorough understanding of the geometry before making progress on introducing the properties and algorithm of Group Law over arithmetically interesting fields, such as finite fields.

In the first section, it starts with the Group Law on elliptic curves, given by explicit polynomial equations, called Weierstrass equations (see Appendix D). Using these explicit equations, it is shown that the set of points of an elliptic curve forms an Abelian group, and the group law is given by rational functions. Then the description of Group Law on singular and non-singular curve shows that the Group Law satisfies the

associative law by using the intersection theory. Finally, an example is given to illustrate the Group Law on an elliptic curve.

3.3.1.1 The Group Law on Elliptic Curves

In this section let **K** be a field, which will be either be the field of real, rational or complex numbers, or the finite field $\mathbf{F}_q$, with $q = p^m$ elements, p is a prime. Let E be an elliptic curve given by a Weierstrass equation (see appendix D):

$$Y^2Z+a_1XYZ+a_3YZ^2=X^3+a_2X^2Z+a_4XZ^2+a_6Z^3$$

where $E \in \mathbf{K}^3$ field consists of the points P = (x, y, 0) satisfying the equation together with the point $\mathcal{O}$ = [0, 1, 0] at infinity. Let $L \in \mathbf{K}^3$ field is a line. Then since the equation has degree three, L intersects E at exactly 3 points, said P, Q, and R.

Note: If L is tangent to E, then there are only two intersections, and P, Q, and R may not be distinct.

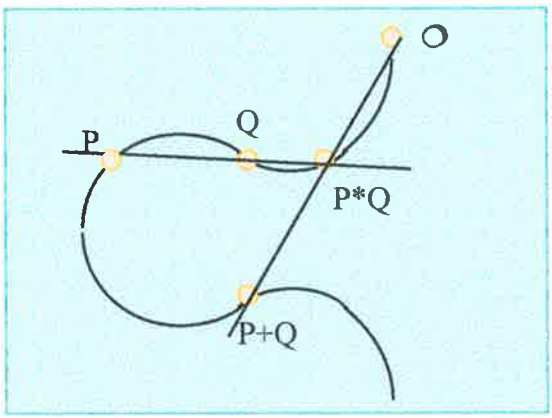

Fig. 3-1. The Group Law on a Elliptic Curve

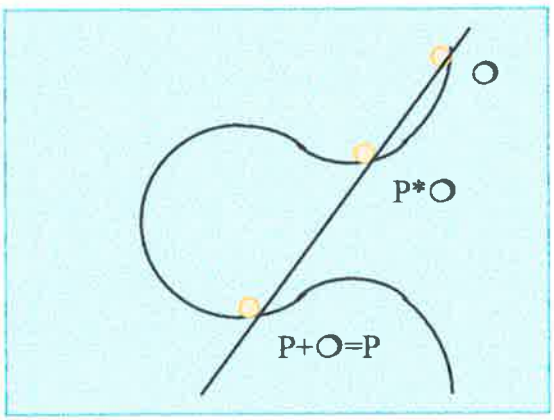

Fig. 3-2. The Verification of the Zero Element

Group Law: Let P, Q ∈ E, L is the line connecting P and Q (if P = Q, L is the tangent line to E), and R is the third point of intersection of L with E. Let L' is the line connecting R and $\mathcal{O}$. Then P + Q is the point such that L' intersects E at R, $\mathcal{O}$, and P + Q, thus $P + Q = \mathcal{O} \times (P \times Q)$.

The Group law [38] is illustrated in figure 3-1, and $\mathcal{O}$ acts as the zero element, as shown in figure 3-2. (The interested reader can refer to [38] for the detail proof of the Group Law).

If the Group Law was associative, then the group would existed [39]. The proof of the Associative Law is following:

Proof: Let P, Q, R be three points on the elliptic curve. To get P + Q, it can form P × Q and take the third intersection of the line connecting it to $\mathcal{O}$. To add P + Q to R point, it can draw the line from R through P + Q, and that meets the curve at (P + Q) × R. To get (P + Q) + R, one has to join (P + Q) × R to $\mathcal{O}$ and takes the third intersection. Now the picture is to show that

$$(P + Q) + R = P + (Q + R),$$

it will be enough to show that

$$(P + Q) \times R = P \times (Q + R)$$

as in figure 3-3.

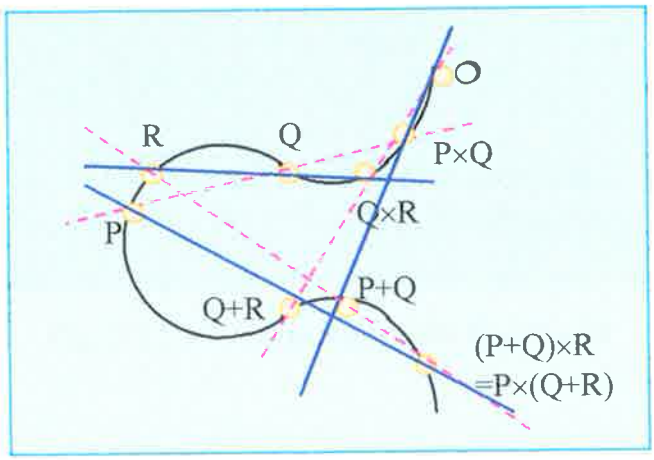

Fig. 3-3 The Verification of the Associative Law

To form P × (Q + R), first one has to find Q × R which joins to $\mathcal{O}$, and takes the third intersection which is Q + R, then one must join Q + R to P, which gives the point P × (Q + R). That should be the same as (P + Q) × R. Now each of the points $\mathcal{O}$, P, Q, R, P + Q, Q + R, Q × R lies on one of the dotted lines and one of the solid lines. It is clear to see that the dotted line through P + Q and R points and the solid line through P and Q + R points are the same. So the equation P × (Q + R) = (P + Q) × R has been proved [39].

The Group Law has the following properties:

1) If a line L intersects E at the points P, Q, and R, then $P + Q + R = O$.

2) $P + O = P \quad \forall \quad P \in E$.

3) $P + Q = Q + P \quad \forall \quad P, Q \in E$.

4) Let $P \in E$, if there is an opposition point (called –P) of E, such that $P + (-P) = O$.

5) Let P, Q, and $R \in E$, then $(P + Q) + R = P + (Q + R)$.

The above properties of the Group Law show that all points on E from an Abelian group with zero element O (called identity element). The proof of the remaining properties is omitted, the interested reader can refer to [39].

3.3.1.2 The Group Law on Singular and Non-singular Curve

A Weierstrass equation is a homogeneous equation of degree 3 of the form

$$X^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3,$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbf{K}$. The Weierstrass equation is called to be non-singular if for all projective points $P = (X:Y:Z) \in P^2(\mathbf{K})$ satisfying

$$F(X, Y, Z) = X^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3 = 0$$

at least one of the three partial derivatives $\partial F/\partial X$, $\partial F/\partial Y$, $\partial F/\partial Z$ is non-zero at point P. If all three partial derivatives vanish at some point P, then P is called a singular point, and the equation is said to be singular.

If the E is non-singular curves over $\mathbf{F}_{2m}$, then the admissible change of variables transforms E to a curve given by a set of the form

$$\{y^2 + xy = x^3 + a_2x^2 - a_6 \mid a_6 \in \mathbf{F}_{2m}, a_2 \in (0, 1)\}. \qquad (3.3\text{-}1)$$

If E is a singular curve over $\mathbf{F}_{2m}$, then the admissible change of variables transforms E to a curve given by equations of the form

$$\begin{aligned} &(1) \quad y^2 + y = x^3, \\ &(2) \quad y^2 + y = x^3 + x, \\ &(3) \quad y^2 + y = x^3 + x + 1, \end{aligned} \tag{3.3-1a}$$

where m is odd, or given by a set of the form

$$\begin{aligned} &(1) \quad y^2 + \mu y = x^3 + \nu, \\ &(2) \quad y^2 + \ \ y = x^3 + \alpha x, \\ &(3) \quad y^2 + \ \ y = x^3 + \beta, \end{aligned} \tag{3.3-1b}$$

where m is even [5].

Let E be a singular elliptic curve defined over a field **K**, such that E is a singular Weierstrass equation of the form

$$f(x, y) = y^2 + a_1xy + a_3y - x^3 - a_2x^2 - a_4x - a_6 = 0. \tag{3.3-2}$$

Then E has precisely one singular point, and suppose that this point is $P = (x_0, y_0) \in E(\mathbf{K})$. If the variables are changed, such that $x \rightarrow x' + x_0$ and $y \rightarrow y' + y_0$, then the singular point can be assumed to be P = (0, 0). Since f(P) = 0, $\frac{\partial}{\partial x} f(P) = 0$ and $\frac{\partial}{\partial y} f(P) = 0$, it leads to $a_6 = a_4 = a_3 = 0$, and so the Weierstrass equation for E simplifies to

$$y^2 + a_1xy - a_2x^2 - x^3 = 0, \qquad a_1, a_2 \in \mathbf{K}. \tag{3.3-3}$$

If $y^2 + a_1xy - a_2x^2 = (y - \alpha x)(y - \beta x)$, where α, β are in **K** or in $\mathbf{K}_1$ ($\mathbf{K}_1$ is the quadratic extension of **K**), then P is called a node if $\alpha \neq \beta$, or a cusp if $\alpha = \beta$. Let $E_{ns}(\mathbf{K})$ denote the set of solutions $(x, y) \in \mathbf{K} \times \mathbf{K}$ to equation (3.3-3), excluding the point P, and including $\mathcal{O}$ at infinity, thus $E_{ns}(\mathbf{K})$ is called the non-singular part of $E(\mathbf{K})$. A Group law on $E_{ns}(\mathbf{K})$ is defined by the chord-tangent law, as described in the last section for $E(\mathbf{K})$. The next result states that $E_{ns}(\mathbf{K})$ is a group, and determines the structure of this group. $\mathbf{K}^+$ and $\mathbf{K}^*$ denote the addition group of **K** and Multiplicative group of non-zero elements of **K** respectively ([5] §4.2.1).

Theorem 3.1 (Theorem 7.2 in [40]): Let E is a singular elliptic curve defined over the finite field **K** with singular point P.

(i) If P is a node, and $\alpha, \beta \in \mathbf{K}$, then the map $\phi : E_{ns}(\mathbf{K}) \rightarrow \mathbf{K}^*$ defined by

$$\phi : O \mapsto 1$$

$$\phi : (x, y) \mapsto (y - \beta x)/(y - \alpha x)$$

is a group isomorphism.

(ii) If P is a node, and $\alpha, \beta \notin \mathbf{K}$, $\alpha, \beta \in \mathbf{K}_1$, then let L be the subgroup of $\mathbf{K}_1^*$ consisting of the elements of norm 1. The map $\Phi : E_{ns}(\mathbf{K}) \rightarrow L$ defined by

$$\Phi : O \mapsto 1$$

$$\Phi : (x, y) \mapsto (y - \beta x)/(y - \alpha x)$$

is a group isomorphism.

(iii) If P is a cusp, then the map $\varphi : E_{ns}(\mathbf{K}) \rightarrow \mathbf{K}^+$ defined by

$$\varphi : O \mapsto 0$$

$$\varphi : (x, y) \mapsto x/(y - \alpha x)$$

is a group isomorphism.

Using the result above, the following results can be derived:

Corollary 3.2 (Theorem 4.2 in [5]) Let E is a singular elliptic curve defined over the finite field $\mathbf{F}_q$ with singular point P.

(i) If P is a node, then the logarithm problem in $E_{ns}(\mathbf{F}_q)$ is reducible in polynomial time to the logarithm problem in $\mathbf{F}_q$ or $\mathbf{F}_q^2$, depending on whether $\alpha \in \mathbf{F}_q$ or $\alpha \notin \mathbf{F}_q$, respectively.

(ii) If P is a cusp, then the logarithm problem in $E_{ns}(\mathbf{F}_q)$ is reducible in polynomial time to the logarithm problem in $\mathbf{F}_q^+$.

If $q = p^m$, where p (**which is a small prime**) is the characteristic of $\mathbf{F}_q$, then

$$\mathbf{F}_q^+ \cong \underbrace{F_p^+ + \cdots + F_p^+}_{m}.$$

It is shown that the logarithm problem in $\mathbf{F}_p^+$ can be efficiently solved in polynomial time by the extended Euclidean algorithm. Thus if a basis of $\mathbf{F}_q$ over $\mathbf{F}_p$ is given, then one can also compute logarithms in $\mathbf{F}_q^+$ in polynomial time.

3.3.1.3 The Algorithm of Group Law on Elliptic Curve

Let elliptic curve E over $\mathbf{Z}_p$ is the set of solutions $(x, y) \in \mathbf{Z}_p \times \mathbf{Z}_p$ to the equation

$$y^2 = x^3 + ax + b \pmod p, \tag{3.3-4}$$

where $p > 3$ is prime, and $a, b \in \mathbf{Z}_p$ are constants such that $4a^3 + 27b^2 \neq 0 \pmod p$, together with a special point $\mathcal{O}$ called the point at infinity.

(1) If $x_1 = x_2$, $y_1 = -y_2$, then $P_1 + P_2 = \mathcal{O}$,

(2) If $x_1 \neq x_2$, then $P_1 + P_2 = P_3$ is given by

$$x_3 = k^2 - x_1 - x_2,$$
$$y_3 = k(x_1 - x_3) - y_1, \tag{3.3-5}$$

With
$$k = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1}, & \text{if } P = Q \end{cases} \tag{3.3-6}$$

Then $y = kx + c$ is the line through P_1 and P_2, or tangent to E if $P_1 = P_2$.

Note that inverses are very easy to compute. The inverse of (x, y) is $(x, -y)$, for all $(x, y) \in E$.

Example (Example 5.7 in [41]): Let E be the elliptic curve $y^2 = x^3 + x + 6$ over $\mathbf{Z}_{11}$. The first task is to determine the points on E. This can be done by looking at each possible $x \in \mathbf{Z}_{11}$, computing and trying to solve the equation:

$$y^2 = x^3 + x + 6 \pmod{11}. \tag{3.3-7}$$

For a given x, it can be tested to see whether $r = x^3 + x + 6 \pmod{11}$ is a quadratic residue by applying Euler's criterion. There is an explicit formula to compute square roots of quadratic residues (mod p) for primes $p = 3 \pmod 4$. Applying this formula, the square roots of a quadratic residue r can be obtained as follows:

$$\pm r^{(11+1)/4} \pmod{11} = \pm r^3 \pmod{11}.$$

There are 13 points on E. Since any group of prime order being cyclic, it follows that E is isomorphic to $\mathbf{Z}_{13}$, and any point other than the point at infinity is a generator of E. Suppose the generator is P = (2, 7); then the "powers" of P can be computed by writing as multiples of P, since the group operation is addition.

To compute 2P = (2, 7) + (2, 7), first one can use (3.3-6) to compute

$$k = (3 \times 2^2 + 1)(2 \times 7)^{-1} \pmod{11}$$
$$= 8 \pmod{11}$$

Then one can use (3.3-4) and (3.3-5) to calculate

$$x_3 = 8^2 - 2 - 2 \pmod{11} = 5$$
$$y_3 = 8(2 - 5) - 7 \pmod{11} = 2,$$

so 2P = (5, 2).

Using the same method as above, the remaining multiples can be computed. The results are shown in Table 3-2. Note that the 13P = O that is called the point at infinity. Therefore, P = (2, 7) is a primitive element.

P = (2, 7)	4P = (10, 2)	7P = (7, 2)	10P = (8, 8)
2P = (5, 2)	5P = (3, 6)	8P = (3, 5)	11P = (5, 9)
3P = (8, 3)	6P = (7, 9)	9P = (10, 9)	12P = (2, 4)

Table 3-2. The Points on the Elliptic Curve over $\mathbf{Z}_{11}$

3.3.2 The Arithmetic Operations of an Elliptic Curve over Field F_q ($q = 2^m$)

For one purpose, the most interested elliptic curve is over finite fields of characteristic two. Here discuss the most efficient techniques for performing the arithmetic operations in such fields. The field $\mathbf{F}_{2m}$ can be viewed as a vector space of dimension m over $\mathbf{F}_2$. In other words, there exists a set of m elements $\beta_0, \beta_1, \ldots, \beta_{m-1}$ in $\mathbf{F}_{2m}$ such that each $\beta \in \mathbf{F}_{2m}$ can be written uniquely in the form

$$\beta = \sum_{i=0}^{m-1} u_i\beta_i, \qquad \text{where } u_i \in \{0, 1\}.$$

Then β can be represented as the 0 or 1 vector $(u_0, u_1, \ldots, u_{m-1})$. There are many different bases of $\mathbf{F}_{2m}$ over $\mathbf{F}_2$. A normal basis of $\mathbf{F}_{2m}$ over $\mathbf{F}_2$ is a basis of the form

$$\left\{\alpha, \alpha^2, \alpha^{2^2}, \cdots, \alpha^{2^{m-1}}\right\},$$

where $\alpha \in \mathbf{F}_{2m}$; it is well known [42] that such a basis always exists. The β can be written as

$$\beta = \sum_{i=0}^{m-1} u_i\alpha^{2^i} \qquad \forall\, \beta \in \mathbf{F}_{2m}, \quad \text{where } u_i \in \{0, 1\}.$$

Since squaring is a linear operator in $\mathbf{F}_{2m}$, hence

$$\beta^2 = \sum_{i=0}^{m-1} u_i\alpha^{2^{i+1}} = \sum_{i=0}^{m-1} u_{i-1}\alpha^{2^i} = (u_{m-1}, u_0, \ldots, u_{m-2}),$$

with indices reduced modulo m.

Therefore a normal basis representation of $\mathbf{F}_{2m}$ is advantageous due to squaring a field element can be accomplished by a simple rotation of the vector representation, which is

easily completed with one clock cycle for squaring an element. Multiplication in a normal basis representation is more complicated. Let

$$U = (u_0, u_1, \ldots, u_{m-1})$$

$$V = (v_0, v_1, \ldots, v_{m-1})$$

are arbitrary elements in $\mathbf{F}_{2^m}$, let

$$W = UV = (w_0, w_1, \ldots, w_{m-1}),$$

then

$$W = \sum_{k=0}^{m-1} w_k \alpha^{2^k} = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} u_i v_j \alpha^{2^i}\alpha^{2^j}. \tag{3.3-8}$$

Suppose that

$$\alpha^{2^i}\alpha^{2^j} = \sum_{k=0}^{m-1} \lambda_{ij}^{(k)} \alpha^{2^k}, \qquad \lambda_{ij}^{(k)} \in \{0, 1\}, \tag{3.3-9}$$

comparing coefficients of α^{2^k} in (3.3-8), then generates the formulae

$$w_k = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} u_i v_j \lambda_{ij}^{(k)}, \qquad 0 \le k \le m-1. \tag{3.3-10}$$

If the power grows up to 2^{-h} th on both sides of equation (3.3-9), then it can find that

$$\alpha^{2^{i-h}}\alpha^{2^{j-h}} = \sum_{k=0}^{m-1} \lambda_{i-h,j-h}^{(k)} \alpha^{2^k} = \sum_{k=0}^{m-1} \lambda_{ij}^{(k)} \alpha^{2^{k-h}}, \tag{3.3-11}$$

Equating the coefficients of α^{2^0} in (3.3-11), it can show that $\lambda_{ij}^{(h)} = \lambda_{i-h,j-h}^{(0)}$, $\forall\ 0 \le (i, j, h) \le m-1$. Then equation (3.3-10) can be rewritten as

$$w_k = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} u_i v_j \lambda_{i-k,j-k}^{(0)} = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} u_{i+k} v_{j+k} \lambda_{ij}^{(0)}.$$

Therefore if a logic circuit with inputs U and V is built to compute the product digit w_0, then the same circuit with inputs U^{2-k} and V^{2-k} will output the product digit w_k. Here the U^{2-k} and V^{2-k} are simply cyclic shifts of the vector representations of U and V. In the same way W can be computed in m clock cycles [43]. The complexity of such a circuit is determined by W_N which is the number of non-zero terms $\lambda_{ij}^{(0)}$, because this amount

measures the number of interconnections between the registers containing U, V and W. Since $W_N \leq m^2$, a lower bound on W_N is that $W_N \geq 2m - 1$. If $W_N = 2m - 1$, then the normal basis is called optimal [44].

From the point of view of minimizing the number of multiplications, the most efficient technique to compute an inverse of an element in $\mathbf{F}_{2m}$ was introduced in reference [45] as follows:

If $\beta \in \mathbf{F}_{2m}, \beta \neq 0$, then $$\beta^{-1} = \beta^{2^m-2} = \left(\beta^{2^{m-1}-1}\right)^2. \qquad (3.3\text{-}12)$$

If m is odd, then since $2^{m-1} - 1 = (2^{(m-1)/2} - 1)(2^{(m-1)/2} + 1)$, The term in (3.3-12) can been written as

$$\beta^{2^{m-1}-1} = \left(\beta^{2^{(m-1)/2}-1}\right)^{2^{(m-1)/2}+1}.$$

Therefore it requires only one multiplication to evaluate $\beta^{2^{m-1}-1}$ once the value of $\beta^{2^{(m-1)/2}-1}$ has been computed.

If m is even, then the term in (3.3-12) can be written as

$$\beta^{2^{m-1}-1} = \beta^{2(2^{(m-2)/2}-1)(2^{(m-2)/2}+1)+1},$$

it requires two multiplication to evaluate $\beta^{2^{m-1}-1}$ once $\beta^{2^{(m-1)/2}-1}$ has been computed. The procedure is then repeated recursively. By induction this method requires

$$f(m) = \log_2(m-1) + g(m-1) - 1$$

field multiplication, where g(m-1) denotes the number of 1's in the binary representation of m-1.

3.3.3 The Practice of Elliptic Curve algorithm in Cryptosystems

In this subchapter, the principle and complexity of arithmetic for the elliptic curve cryptosystem will be clearly shown by introducing a practice of implementation of Elliptic Curve cryptosystem explored by A. J. Menezes [5]. For simplicity, the proofs and inferences will be abbreviated.

This subchapter consists of three sections. In the first two sections, a description of the selection of an elliptic curve and field **F** and counting the points on Elliptic Curves, the non-singular curves and singular curves over field $\mathbf{F}_{2}$m will be considered. Finally, the Completion of the ECCA will be discussed in the third section.

3.3.3.1 Selection of an Elliptic Curve and Field $\mathbf{F}_q$

For simplicity, the elliptic curve can be recalled from (3.3-4) as follows:

$$y^2 = x^3 + ax + b \pmod{p}.$$

If $P \in E$, then the inverse point is $-P \in E$. If $Q_{(x2, y2)} \in E$, and $Q \neq -P$, then $P + Q = R_{(x3, y3)}$, where $x_3 = k^2 - x_1 - x_2$, $y_3 = k(x_1 - x_3) - y_1$, and

$$k = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq Q \\ \dfrac{3x_1^2 + a}{2y_1}, & \text{if } P = Q \end{cases}$$

For the discussion above, it was shown that the addition operations of two distinct points on an elliptic curve will require three multiplications and one inversion of field elements in the underlying field $\mathbf{F}_{2m}$, and doubling a point will require one inversion and four multiplications in $\mathbf{F}_{2m}$. Here the additions and subtractions are not considered, because these operations are not expensive. For minimizing the number of field operations, the selection of a curve and field $\mathbf{F}_{2m}$ has to be a consideration to satisfy certain conditions

(§6.2 in [5]) as follows:

(1) The arithmetic operation in field $\mathbf{F}_{2m}$ has to be easier to complete than those in finite fields of characteristic ≥ 3.

(2) Using a normal basis representation for the elements of $\mathbf{F}_{2m}$, so that doubling a point becomes a simple cyclic shift operation of the vector representation, and therefore the squaring operation is reduced to a simple operation.

(3) It is easier to recover the y-coordinate of a point given its x-coordinate with a single bit of extra information. It is helpful in reducing message expansion in the ElGamal cryptosystem, as shown in chapter 2.2.

3.3.3.1.1 Selection of an Non-singular Elliptic Curve and Field $\mathbf{F}_q$

First, consider the non-singular curves over $\mathbf{F}_{2m}$, recalling the equation (3.3-4),

$$y^2 = x^3 + ax + b \pmod{p},$$

where $a \in \mathbf{F}_q \backslash \{0\}$ ($q = 2^m$), $b \in \mathbf{F}_q$. If the method for reducing the elliptic curve logarithm problem to the discrete logarithm problem in a finite field is not feasible, then the best algorithm known for the logarithm problem in non-singular elliptic curves is the Baby-step Giant-step algorithm. A non-singular curve, which is suitable for cryptographic applications, is the one whose order (see Appendix A-2) is divisible by a large prime factor of at least 40 decimal digits. Therefore the underlying field should be the size at least 2^{130} (130 binary digits), and should have an optimal normal basis in order to perform efficiently the arithmetic in the field.

One method for the curve selection is to choose a curve E defined over $\mathbf{F}_q$, where q is small enough so that #E (where #E denotes the number of points on E) can be computed directly, then the group $E(\mathbf{F}_{q^n})$ can be used for suitable n. The $\#E(\mathbf{F}_{q^n})$ can easily be computed from $\#E(\mathbf{F}_q)$ by the Weil Theorem (see Appendix E). It is shown that if n is divisible by h, then $\#E(\mathbf{F}_{q^n})$ is divisible by $\#E(\mathbf{F}_{q^h})$. Therefore the n selected should be prime, or else a product of a small factor and a large prime.

Example (Example 6.2 in [5]) If one want to select a non-singular curve over $\mathbf{F}_{2^{155}}$, one can select a curve over $\mathbf{F}_{25}$. There are twelve possibilities for $\#E(\mathbf{F}_{25})$. Of these, there are five values for which $\#E(\mathbf{F}_{2^{155}})$ is divisible be a large prime. The size of the largest prime divisor of $\#E(\mathbf{F}_{2^{155}})$ for these five values is as shown on table 3-2. The curves with $\#E(\mathbf{F}_{2^{155}}) = 36$ or 42 would be best suited for security purposes in cryptography.

$\#E(\mathbf{F}_{25})$	The number of digits in the largest prime divisor of $\#E(\mathbf{F}_{2^{155}})$
22	37
28	36
36	46
38	36
42	41

Table 3-2. Five values of the number of points on non-singular curves E over $\mathbf{F}_{25}$.

Another method of selecting a random curve E was adapted by Neal Koblitz to curves over fields of characteristic 2 [77]. Using heuristic arguments, Koblitz [77] showed that if E over $\mathbf{F}_q$ is a randomly selected non-singular curve, then the probability that $N = \#E(\mathbf{F}_q)$ is divisible by a prime factor $\geq N/V$ (V should be an small factor integer) is $m^{-1}\log_2(V/2)$. So that the probability that the order of a randomly chosen non-singular curve over $\mathbf{F}_{2^{155}}$ is divisible by a 40 digits prime is approximately

$$\frac{1}{155}\log_2\left(\frac{2^{155}}{2\times 10^{40}}\right) \approx 0.136.$$

Therefore, it is clearly shown that one can expect to try about seven curves before a suitable one is found successfully.

The computation of kP and kaP takes 29 additions of points, 1 field inversion, 155 doublings and 2 field multiplications. Computing $M_1\bar{x}$ and $M_2\bar{y}$, where $kaP = (\bar{x}, \bar{y})$, takes a further 2 multiplications. Therefore, two field elements can be encrypted taking 2950 field multiplications. Since $\#E(\mathbf{F}_{2^{155}})$ is about a 47 digit prime, the square root attacks can be precluded (refer to [74]). Assuming that a clock rate is 5 MHz, and an inversion takes $f(155) = 10$ multiplication, the encryption rate is

$$\frac{155 \times 2 \times 5{,}000{,}000}{1000 \times 2950 \times 155} \approx 3.4 \text{ Kbits/sec.}$$

3.3.3.1.2 Selection of an Singular Curve and Field F_q

Now consider the singular curves over F_{2m}, where m is odd. A fourth condition which should be satisfied in the selection of a singular curve, in addition to except the three conditions described in the last subchapter, is

(4) For singular curves over F_{2m}, the inverse operation in doubling a point can be eliminated by choosing $a_3 = 1$, further reducing the operation count.

Recall from subchapter 3.3.1.2, here only consider the curve (3.3-1a), a representative curve is

$$y^2 + y = x^3 + x + 1.$$

If $P_{(x1, y1)} \in E$, then inverse point is $-P_{(x1, -y1)} \in E$. If $Q_{(x2, y2)} \in E$, and $Q \neq -P$, then $P + Q = R_{(x3, y3)}$, where

$$x_3 = \begin{cases} \left(\dfrac{y_1+y_2}{x_1+x_2}\right)^2 + x_1 + x_2, & P \neq Q \\ x_1^4 + 1, & P = Q \end{cases} \qquad y_3 = \begin{cases} \left(\dfrac{y_1+y_2}{x_1+x_2}\right)(x_1 + x_3) + y_1 + 1, & P \neq Q \\ x_1^4 + y_1^4 + 1, & P = Q \end{cases}$$

In the subchapter 3.3.2.1, it is shown that doubling a point on the curve (3) is "free", if a normal basis representation is chosen for the elements of F_{2m}, while adding two distinct points needs two multiplication and one inversion. The multiple kP of the point P is computed by the repeated square and multiply method. If g(k) = t + 1, then the exponentiation takes 2t multiplication and t inversions.

Resorting to projective coordinates can eliminate the inverse operation needed adding two points. For example, if $P = (x_1, y_1, 1)$, $Q = (x_2, y_2, z_2)$, and $R = (x_3, y_3, z_3)$, the addition formula $P + Q = R$ requires 9 multiplication of field elements. So that if the multiple kP (where P is the affine point $(x_1, y_1, 1)$) is computed by repeatedly using the square and multiply method, the result $kP = (x_3, y_3, z_3)$ can be converted back into affine coordinates by multiplying each coordinate by z_3^{-1}. If $g(k) = t + 1$, then the total operation count to compute kP is $9t + 2$ field multiplications and one inversion.

The estimation of the throughput rate of encryption using the elliptic curve analogue of the ElGamal cryptosystem is described as follows.

Suppose that a multiplication in $\bar{y}$, takes m clock cycles, while an inversion takes $f(m) = \log_2(m-1) + g(m-1) - 1$ multiplication's. For simplicity, the cost of field additions and squaring are ignored. The points on the curve E will be represented using projective coordinates. To increase the speed of the cryptosystem and to place an upper boundary on the time for encryption, the *Hamming weight*[1] of k would be limited to 30.

The computation of kP and kaP takes 58 additions of points, 2 field inversions, and 4 field multiplications. Computing $M_1\bar{x}$ and $M_2\bar{y}$, where $kaP = (\bar{x}, \bar{y})$, takes another 2 multiplications. Therefore two field elements can be encrypted taking $528 + 2f(m)$ field multiplications. For example: suppose that the curve (3.3-1a) is over $\mathbf{F}_{2m}$. Since $\#E(\mathbf{F}_{2^{239}})$ is a 72 digit prime, the square root attacks can be precluded (refer to [74]). Assuming that a clock rate is 5 MHz, and an inversion takes $f(239) = 12$ multiplication, the encryption rate is then

$$\frac{239 \times 2 \times 5{,}000{,}000}{1000 \times (528 + 2 \times 12) \times 239} \approx 18 \text{ K bits/sec.}$$

3.3.3.2 Counting the Points on Elliptic Curves

In this subchapter two for counting the points on elliptic curve are introduced. First Schoof's algorithm shows how to count the points on the non-singular curve, and later Waterhouse's algorithm counts the points on the singular curve. The description of the two algorithms follows.

3.3.3.2.1 Counting the Points on Non-singular Curves

Schoof's algorithm to compute the number of points on a non-singular elliptic curve is completed in three steps [5]. The first step consists of computing a number L for which the equation of the form

$$\prod_{\substack{l \le L \\ l \ne 2,p}} l > 4\sqrt{q}, \tag{3.3-13}$$

holds and of making a list of the *division polynomials* f_n (see Appendix A-2) for n = 1, 2, ..., L, where ℓ is prime. The second step consists of the computation of t (mod ℓ) in equation (3.3-16) for every prime $\ell \le$ L not equal 2 or p. The third step is the computation of t from the values of t (mod ℓ) obtained using the Chinese Remainder Theorem and the estimate $|t| \le 2\sqrt{q}$. An outline of Schoof's algorithm for computing the number of points on non-singular elliptic curve E over $\mathbf{F_q}$ ($q = 2^m$) will now be given.

Recall the equation (3.3-4) for E, which is given as follows:

$$y^2 = x^3 + ax + b \pmod p, \tag{3.3-14}$$

with a, b $\in \mathbf{F_q}$ and $4a^3 + 27b^2 \neq 0$. Let $\overline{F}$ denotes the algebraic closure of $\mathbf{F_q}$. Then $E(\overline{F})$ denote the set of points on E over $\overline{F}$, consisting of all the solutions (x, y) of (3.3-14). Let Φ be the *Frobenius endomorphism* (see Appendix A-2) on $E(\overline{F})$ defined by

$$\Phi: \quad (x, y) \rightarrow (x^q, y^q).$$

[1] The Hamming weight of an integer is the number of ones in its binary representation.

Let $\mathbf{R}_{Fq}$ denotes the ring of endomorphisms of E over $\overline{F}$. In $\mathbf{R}_{Fq}$, the Frobenius endomorphism satisfies a unique relation

$$\Phi^2 + q = t\Phi. \tag{3.3-15}$$

Let #E($\mathbf{F}_q$) denotes the number of points on E over $\mathbf{F}_q$ in the form

$$\#E(\mathbf{F}_q) = q + 1 - t, \tag{3.3-16}$$

where t is called the trace of the Frobenius endomorphism. It satisfies an inequality of the form:

$$|t| \le 2\sqrt{q}\,. \tag{3.3-17}$$

To start, choose a number L satisfying (3.3-13), where the product ranges over all primes ℓ ($3 \le \ell < L$). Since there is a bound on the size of t by (3.3-17), one can compute t (mod ℓ) for each odd prime $\ell < L$, such that ℓ = 3, 5, 7, ..., L, and one can determine t by applying the Chinese Remainder Theorem. The next step is to describe how to compute t (mod ℓ) for ℓ, a prime not equal to 2 or q.

Let P(x, y) $\in$ E[ℓ], let ℓ is an odd prime then E[ℓ] $\cong \mathbf{Z}_\ell \oplus \mathbf{Z}_\ell$, which can be viewed as a vector space over $\mathbf{F}_\ell$; the vector space has dimension 2. Let k = q (mod ℓ), $0 \le k \le \ell - 1$. One can search for an integer τ, where $0 \le \tau \le \ell - 1$, to satisfy

$$\Phi^2 P + kP = \tau\Phi P. \tag{3.3-15}$$

From (3.3-15) one deduces that $(t - \tau)\Phi P = \mathcal{O}$. Therefore, since ΦP is a point of order ℓ (see Appendix A-2), $t = \tau$ (mod ℓ). The problem with completing this idea is that the coordinates of P, which are in $\overline{F}$, may not lie in any small extension of $\mathbf{F}_q$. Thus, the t can not be found in general. The solution is by observing that $\bar{x}$ is a root of the *division polynomial* (see Appendix A-2) $f_\ell(x) \in \mathbf{F}_q$. The next Theorem 3.4 can be used to obtain an expression for kP and $\tau\Phi$P, where the coordinates of the expressions are rational functions

in x, y, then one can use the addition rules to sum Φ^2P and kP.

Theorem 3.4 (Theorem 7.2 in [5]): Let $n \geq 2$, $P(x, y) \in E$ with $nP \neq O$, then $nP(\bar{x}, \bar{y})$can be obtained by

$$\bar{x} = x + \frac{f_{n-1}f_{n+1}}{f_n^2},$$

$$\bar{y} = x + y + \frac{f_{n-1}f_{n+1}}{f_n^2} + \frac{f_{n-2}f_{n+1}^2}{\bar{x}f_n^3} + (x^2 + y) + \frac{f_{n-1}f_{n+1}}{\bar{x}f_n^2}.$$

(where $f_n = f_n(\bar{x})$).

To test the point P(x, y) in $E[\ell]$ satisfies (3.3-15), and to determine the correct sign, one can equate the x-coordinates and y-coordinates of $\Phi^2P + kP$ and $\tau\Phi P$, respectively, and eliminate the denominators and the variable y to result the equations $h_1(x) = 0$ and $h_2(x) = 0$ respectively. Thus one can compute $H_1(x) = \gcd(h_1(x), f_l(x))$ and $H_2(x) = \gcd(h_2(x), f_l(x))$ respectively. If $H_1(x) \neq 1$ or $H_2(x) \neq 1$, then the P satisfies (3.3-15). If $H_1(x) = 1$, then the P in $E[\ell]$ does not satisfy $\Phi^2P + kP = \pm\tau\Phi P$. If $H_2(x) = 1$, then P satisfies $\Phi^2P + kP = -\tau\Phi P$. For a detailed description refer to [46].

In the algorithm for pre-computing the polynomials f_n, it takes $O(\log^5 q)$ bits to store these polynomials. The amount of memory used in the rest of the algorithm is dominated by $O(\log^5 q)$. The computations of L and the Chinese Remainder Theorem are easily seen to be dominated by $O(\log^9 q)$ elementary operations.

The running time of $O(\log^8 q)$ elementary operations in the second step is obtained as follows: For each ℓ, the search for τ satisfying (3.3-15) is dominated by the computations of the residues of x^{q^2} and y^{q^2} modulo $f_\ell(x)$, where $\Phi^2P = (x^{q^2}, y^{q^2})$. Due to compute the degree of $f_\ell(x)$ is $O(\log^2 q)$ operations, these residues can be computed in $O(\log^5 q)$ field operations, or $O(\log^7 q)$ bit operations. If fast multiplication techniques are employed

for multiplication in K and in $\mathbf{F}_q$, then the total running time reduces to $O(\log^{5+e} q)$ bit operations, for any $e > 0$.

3.3.3.2.2 Counting the Points on Singular Curves

To determine the number of points on a singular elliptic curve E over $\mathbf{F}_{2m}$, the group type of the curves has to be determined by the Lemma 3.5 described as follows:

Lemma 3.5 [47]: Let $\#E(\mathbf{F}_q) = q + 1 - t$,

(1) If $t^2 = q$, $2q$, or $3q$, then $E(\mathbf{F}_q)$ is cyclic.

(2) If $t^2 = 4q$, then $E(\mathbf{F}_q) \cong Z_{\sqrt{q}\mp 1} \oplus Z_{\sqrt{q}\mp 1}$ depending on whether $t = \pm 2\sqrt{q}$ respectively.

(3) If $t = 0$ and $q \neq 3 \pmod 4$, then $E(\mathbf{F}_q)$ is cyclic. If $t = 0$ and $q = 3 \pmod 4$, then $E(\mathbf{F}_q)$ is cyclic, or $E(\mathbf{F}_q) \cong Z_{(q+1)/2} \oplus Z_2$.

where Z_n denotes the cyclic group on n elements.

When m is odd: Recall the equation (3.3-1a); each of the 3 isomorphism classes of singular curves over $\mathbf{F}_{2m}$ has a representative with coefficients in $\mathbf{F}_2$. One can easily determine the order of curves over $\mathbf{F}_{2m}$ by the Weil Theorem. The results are listed in Table 3-3.

	Elliptic Curves	Order of Curve
(1)	$y^2 + y = x^3$,	$q + 1$.
(2)	$y^2 + y = x^3 + x$,	$q + 1 \pm \sqrt{2q}$.
(3)	$y^2 + y = x^3 + x + 1$,	$q + 1 \pm \sqrt{2q}$.

Table 3-3. The Curves and Orders in 3 classes of supersingular curves over $\mathbf{F}_2$m (m is odd)

When m is even: Recall (3.3-1b); let $\#E(\mathbf{F}_{2^m}) = q + 1 - t_i$ for the three type of equations. The three term values of t_i are $\pm\sqrt{q}$, 0, and $\pm\sqrt{2q}$ respectively can be obtained by the Theorem 3.6 as following.

	Elliptic Curves	Orders of Curves
(1)	$y^2 + \mu y = x^3 + \nu$,	$q + 1 \pm \sqrt{q}$,
(2)	$y^2 + y = x^3 + \alpha x$,	$q + 1$,
(3)	$y^2 + y = x^3 + \beta$,	$q + 1 \pm \sqrt{2q}$.

Table 3-4. The Curves and Orders in 3 classes of supersingular curves over $\mathbf{F}_{2^m}$ (m is even)

Theorem 3.6 [48]: Let p be a prime and $q = p^m$. Let t be an integer with $|t| \le 2\sqrt{q}$. Then

$$N_q(t) = \begin{cases} H(t^2 - 4q), & \text{if } t^2 < 4q, \text{ and } p \square t. \\ H(-4p), & \text{if } t = 0 \text{ and m is odd.} \\ 1, & \text{if } t^2 = 2q, p = 2, \text{ m is odd.} \\ 1, & \text{if } t^2 = 3q, p = 3, \text{ m is odd.} \\ 1/12\{p + 6 + 4(3/p) + 3(4/p)\}, & \text{if } t^2 = 4q, \text{ and m is even.} \\ 1 + (3/p), & \text{if } t^2 = q \text{ and m is even.} \\ 1 + (4/p), & \text{if } t = 0 \text{ and m is even.} \\ 0, & \text{otherwise.} \end{cases}$$

where $N_q(t)$ denotes the number of isomorphism classes of elliptic curves over $\mathbf{F}_q$, $H(\Delta)$ denotes the Kronecker class number of Δ (see Proposition 7 in [75]), and "Δ" denotes a negative integer congruent to 0 or 1 (mod 4).

It is known that given an arbitrary singular elliptic curve E over $\mathbf{F}_{2^m}$, the number of points $\#E(\mathbf{F}_{2^m})$ can be computed by first determining to which isomorphism class E belongs. Solving the appropriate root given by Theorem 2.2 in [5] can complete this. There are several efficient polynomial time algorithms for finding the roots of a polynomial over $\mathbf{F}_{2^m}$ in [49]. The interested reader can see [50] for more details.

3.3.3.3 The Completion of Elliptic Curve Cryptography Algorithm

The completion of Elliptic Curve cryptography algorithm is to describe the establishment of the cryptosystem. In chapter 2.3 the Elliptic Curve analog of ElGamal cryptosystem is well known example to the completion of Elliptic Curve cryptography algorithm. Here starts with a briefly to describe a implementation of a non-supersingular curve over fields of characteristic 2. In order to expound the programs of the completion of Elliptic curve cryptography algorithm, two examples of ElGamal cryptosystem and Menezes-Vanstone cryptosystem will be presented later.

3.3.3.3.1 Implementation of the ECCs

This subchapter describes how to implement the ECCs on non-supersingular curves over fields of characteristic 2. In comparison to supersingular curves the addition is slightly harder to compute, because doubling of a point is more complicated to calculate. The complexity of the basic arithmetic operations in finite fields differs considerably. The additions are negligible in comparison to multiplications. The inversions are the most time consuming operation, therefore the curve is represented in projective coordinates and the inversions can be eliminated. Only at the end of each calculation two inversions are needed to get a 'unique' representation [51, 29].

Suppose the non-supersingular curve over the field $\mathbf{F}_q$ (where $q= 2^m$) is $y^2 + xy = x^3 + ax^2 + b$, with a point $P = (x, y)$. From a base point $P = (x, y)$, it can calculate mP with a double point and add point algorithm. Starting with the highest bit of m, it needs only doublings of a point and additions of two different points of the form $(x, y) + (x_i, y_i)$. Assuming one point is given in affine coordinates, then $P = (x_1, y_2)$, $Q = (x_2, y_2)$, and $-P = (x_1, x_1 + y_1)$. If $Q \neq -P$, then the addition formulas is $P + Q = (x_3, y_3)$.

Note that two multiplications and an inversion are needed to add two distinct points, and a point can be doubled in 3 multiplications and an inversion. It also notes that inversions can be avoided by changing to projective coordinates at the expens of doing more

multiplications [52].

3.3.3.3.2 A Practical Example in ECCs

In completion of ECCs, there is a practical difficulty that is message expansion, since every ciphertext consists of four field elements. In ElGamal cryptosystem, four field elements are transmitted in order to convey a message consisting of two field elements, which is called message expansion by a factor of two, with the system implement in $GF(p^n)$ with $n > 1$. Here is an example of ElGamal cryptosystem using Elliptic Curve algorithm.

Example 3.6 (Example 5.8 in [41]): Suppose that $y^2 = x^3 + x + 6$, $\alpha = (2, 7)$ and Bob's secret exponent is $a = 7$, so that $\beta = 7\alpha = (7, 2)$ (recall table 3-2). The encryption is

$$e_K(x, k) = (k(2, 7), x + k(7, 2)),$$

where $x \in E$ and $0 \le k \le 12$, and the decryption is

$$d_K(y_1, y_2) = y_2 - 7y_1.$$

Suppose that Alice wishes to encrypt the message $x = (10, 9)$, which is a point on E. If Alice chooses the random value $k = 3$, then she will compute

$$y_1 = 3\alpha = 3(2, 7) = (8, 3),$$

and

$$\begin{aligned} y_2 &= (10, 9) + 3(7, 2) \\ &= (10, 9) + (3, 5) \\ &= (10, 2). \end{aligned}$$

so $y = ((8, 3), (10, 2))$. If Bob receives the ciphertext y, he decrypts it

$$\begin{aligned} x &= (10, 2) - 7(8, 3) \\ &= (10, 2) - (3, 5) \\ &= (10, 2) + (3, 6) \\ &= (10, 9). \end{aligned}$$

Therefore the decryption generates the correct plaintext.

Menezes and Vanstone [41] have found one more efficient variation in which the elliptic curve is used for "masking", and plaintexts and ciphertexts are allowed to be arbitrary ordered pairs of non-zero field elements (i.e., they are not required to be points on E). This will generate a message expansion factor of two, which is the same as in the ElGamal cryptosystem. An example of the Menezes-Vanstone cryptosystem is introduced as following.

Example 3.7 (Example 5.9 in [41]): as in example 3.6, suppose that $\alpha = (2, 7)$ and Bob's secret exponent is $a = 7$, so that $\beta = 7\alpha = (7, 2)$ (recall table 3-2). Suppose Alice want to encrypt the plaintext

$$x = (x_1, x_2) = (9, 1).$$

Note that x is not a point on E, and Alice chooses the random value $k = 6$. She computes

$$y_0 = k\alpha = 6(2, 7) = (7, 9),$$

$$k\beta = 6(7, 2) = (8, 3) = (c_1, c_2),$$

then she computes

$$y_1 = c_1x_1 \pmod P = 8x_1 \pmod P = 8 \times 9 \pmod{11} = 6,$$

$$y_2 = c_2x_2 \pmod P = 3x_2 \pmod P = 3 \times 1 \pmod{11} = 3.$$

The ciphertext she sends to Bob is

$$y = (y_0, y_1, y_2) = ((7, 9), 6, 3).$$

When Bob receives the ciphertext y, he computes

$$ay_0 = 7(7, 9) = (8, 3) = (c_1, c_2),$$

Then computes

$$x = (y_1 c_1^{-1} \pmod p, y_2 c_2^{-1} \pmod p)$$

$$= (6 \times 8\text{-}1 \pmod{11}, 3 \times 3\text{-}1 \pmod{11})$$

$$= (9, 1).$$

Therefore, the decryption generates the correct plaintext.

Summary:

The descriptions above give a brief introduction to three kinds of mathematical principles related to three different cryptosystems: the RSA, ElGamal and Elliptic Curve system respectively. It has show that the security of RSA depends on the difficulty of factoring large integers, the ElGamal system depends on the difficulty of soving the classical discrete logarithm problem and the Elliptic Curve system depends on the difficulty of discrete elliptic logarithm problem. Also it has been clearly shown how these public key cryptosystems have different degrees of security, length of keys, secure terms (periods) and running times. Therefore, the mathematical theory already discussed gives a basis for the computation and comparison of running time and efficiency that will be introduced in the next chapter.

4. The Efficiency of Elliptic Curve Cryptosystem

The purpose of this chapter is to discuss the efficiency and practicality of the Elliptic Curve Cryptosystem (**ECCs**). The efficiency of ECCs means the system has the same level of security performance as the classical Discrete Logarithm cryptosystem, but requires much smaller key length. However, the creation of efficiency in the ECCs is based on the complexity of computation that requires more time in processing the ECC operations, and that leads to less practicality. Therefore the efficiency discussed here contains two aspects, one is the high efficiency in ECCs, and another is the efficiency trade off for the time of computation.

4.1 The Advance Efficiency in Elliptic Curve Cryptosystems

The dramatic efficiency in ECCs will be determined by the structure of computation of the group operation in ECCs, which leads to a stronger One Way function and longer security term with shorter key length. This subchapter starts with security in public key cryptosystems, which describes security in cryptosystems by two characteristics of security feature: One Way function and Security Term in RSA, ElGamal and Elliptic Curve cryptosystems. Then the efficiency advantage will be shown by the comparison of efficiencies of the cryptosystems between the three cryptosystems.

4.1.1 Security in Public Key Cryptosystem

Currently there does not exist a standard method of measuring the security (including security level and security term) in a cryptosystem. The One-way function and Key length are usually considered as two characteristic terms to measure the security of a cryptosystem. Although no proof is known for the existence of a One-way function, it is widely believed that One-way function do exist. The following are candidates for One-way functions since they are easy to compute, but their inversion requires the solution of

the factoring problem, the ElGamal Discrete Logarithm problem or Elliptic Logarithm problem, respectively.

This section begins with a discussion of the security in the RSA system, including the analysis of attacking the system and the security bound on the size of the primes. Then a discussion follows about security in the ElGamal system. Finally there is a discussion of the security in the Elliptic Curve system.

4.1.1.1. The Security in the RSA System

Here is a discussion of various security issues related to RSA encryption, as well as various attacks. Appropriate measures to counteract these threats are presented.

(1) Relation to factoring

Recall in chapter 2, the task faced by a passive[1] adversary is that of recovering plaintext m from the corresponding ciphertext c, given the public key (n, e) of the intended receiver Alice. This is called the RSA problem. One possible approach that an adversary could employ to solving the RSA problem is to first factor n, and then computes $\phi(n)$ and d as Alice did in chapter 2.1. Once d is obtained, the adversary can decrypt any ciphertext intended for Alice.

On the other hand, if an adversary could somehow compute d, then it could subsequently factor n efficiently as next. First note that since $ed = 1 \pmod{\phi}$, there is an integer k such that $ed - 1 = k\phi$. Hence, by the fact of $a^{ed-1} = 1 \pmod{n}$ for all $a \in \mathbf{Z}_n$. Let $ed - 1 = 2^s t$, where t is an odd integer. Then it can be shown that $a^{2^{s-1}t} \neq \pm 1 \pmod{n}$ for at least half of all $a \in \mathbf{Z}_n$; if a is such an integer then $\gcd(a^{2^{s-1}t} - 1, n)$ is a non- trivial factor of n. Thus the

[1] A passive adversary is an adversary who is capable only of reading information from an unsecured channel.

adversary simply needs to repeatedly select random a $\in \mathbf{Z}_n$ and compute gcd($a^{2^{s-1}t}$-1, n); the expected number of a before a non-trivial factor of n is obtained is 2 [53].

The problem of computing the RSA decryption exponent d from the public key (n, e), and the problem of factoring n, is computationally equivalent. When generating RSA keys, it is imperative that the primes p and q should be selected so that factoring n = pq is computationally infeasible. The major restriction on p and q in order to avoid the elliptic curve factoring algorithm is that p and q should be about the same bit length, and sufficiently large. For example, if a 1024 bit modulus n is to be used, then each of p and q should be about 512 bits in length. A 512-bit modulus n provides only marginal security from concerted attack. In order to avoid the quadratic sieve and number field sieve factoring algorithms, a modulus n of at least 768 bits is recommended. For long term security, 1024-bit or larger module should be used.

(2) Small decryption exponent d

As was the case with the encryption exponent e, it may seem desirable to select a small decryption exponent d in order to improve the efficiency of decryption. However, if gcd(p-1, q-1) is small, as is typically the case, and if d has up to approximately one-quarter as many bits as the modulus n, then there is an efficient algorithm [53] for computing d from the public information (n, e). This algorithm cannot be extended to the case where d is approximately the same size as n. Therefore, to avoid this attack, the decryption exponent d should be roughly the same size as n.

(3) Multiplicative properties

Let m_1 and m_2 be two plaintext messages, and let c_1 and c_2 be their respective RSA encryption. Observe that

$$(m_1m_2)^e = m_1^e m_2^e = c_1c_2 \pmod{n}.$$

In other words, the ciphertext corresponding to the plaintext $m = m_1m_2 \pmod n$ is $c = c_1c_2 \pmod n$; this is sometimes referred to as the *homomorphic property* of RSA. This observation leads to the following *adaptive chosen-ciphertext attack* on RSA encryption.

Suppose that an active adversary wishes to decrypt a particular ciphertext $c = m^e \pmod n$ intended for Alice. Suppose also that Alice will decrypt arbitrary ciphertext for the adversary, other than c itself. The analyst can conceal c by selecting a random integer $x \in \mathbf{Z}_n$ and computing $\bar{c} = cx^e \pmod n$. Upon presentation of $\bar{c}$, Alice will compute for the analyst $\bar{m} = (\bar{c})^d \pmod n$. Since

$$\bar{m} = (\bar{c})^d = c^d(x^e)^d = mx \pmod n,$$

the analyst can then compute $m = \bar{m}\, x^{-1} \pmod n$.

This *adaptive chosen-ciphertext attack* can be circumvented in practice by imposing some structural constraints on plaintext messages. If a ciphertext c is decrypted to a message not possessing this structure, then the decryptor refects c as being fraudulent. Now if a plaintext message m has this (carefully chosen) structure, then with high probability mx (mod n) will not have this structure for $x \in \mathbf{Z}_n$. Therefore the *adaptive chosen-ciphertext attack* will fail because Alice will not decrypt $\bar{c}$ for the adversary [53].

(4) Cycling attacks

Let $c = m^e \pmod n$ be a ciphertext. Let k be a positive integer such that $c^{e^k} = c \pmod n$; since encryption is a permutation on the message space $\{0, 1, \ldots, n-1\}$ such an integer k must exist. For the same reason it must be the case that $c^{e^{k-1}} = m \pmod n$. This observation leads to the following *cycling attack* on RSA encryption. An adversary computes $c^e \pmod n$, $c^{e^2} \pmod n$, $c^{e^3} \pmod n$,... until c is obtained for the first time. If $c^{e^k} \pmod n = c$, then the previous number in the cycle, namely $c^{e^{k-1}} \pmod n$, is equal to

the plaintext m. A *generalized cycling attack* is to find the smallest positive integer u such that f = gcd(c^{e^u} -c, n) > 1. If

$$c^{e^u} = c \pmod{p} \text{ and } c^{e^u} \neq c \pmod{q}, \tag{4.1-1}$$

then f = p. Similarly, if

$$c^{e^u} \neq c \pmod{p} \text{ and } c^{e^u} = c \pmod{q}, \tag{4.1-2}$$

then f = q. In either case, n has been factored, and the adversary can recover d and then m. On the other hand, if both

$$c^{e^u} = c \pmod{p} \text{ and } c^{e^u} = c \pmod{q}, \tag{4.1-3}$$

then f = n and $c^{e^u} = c \pmod{n}$. In fact, u must be the smallest positive integer k for which $c^{e^k} = c \pmod{n}$. In this case, the basic *cycling attack* has succeed and so $m = c^{e^{u-1}} \pmod{n}$ can be computed efficiently. Since equation (4.1-3) is expected to occur much less frequently than (4.1-1) or (4.1-2), the *generalized cycling attack* usually terminates before the *cycling attack*. For this reason, the *generalized cycling attack* can be viewed as being essentially an algorithm for factoring n. Since factoring n is assumed to be intractable, these *cycling attacks* do not pose a threat to the security of RSA encryption.

4.1.1.2. The Security in the ElGamal System

The security in the ElGamal cryptosystem is based on the difficulty of the classical discrete logarithm problem in a finite group, which is the multiplicative group of GF(p). Here the discussion of security in the ElGamal cryptosystem includes two parts: security in encryption and signature.

(1) Security of ElGamal encryption

The security of many cryptographic techniques depends on the intractability of the discrete logarithm problem. These include the Diffie-Hellman and ElGamal systems. The problem of breaking the ElGamal scheme (i.e., recovering m given p, α, α^2, γ, and δ) is

equivalent to solving the Diffie-Hellman problem. In fact the ElGamal scheme can be viewed as simply comprising a Diffie-Hellman key exchange to determine a session key α^{ak}, and then encrypting the message by multiplication with that session key. For this reason, the security of the ElGamal scheme is to be based on the discrete logarithm problem in $\mathbf{Z}_p$, although such equivalence has not been proven.

It is important that different random integers k be used to encrypt different messages. Suppose the same k is used to encrypt two messages m_1 and m_2 and the resulting ciphertext pairs are (γ_1, δ_1) and (γ_2, δ_2). Then $\delta_1/\delta_2 = m_1/m_2$, and m_2 could be easily computed if m_1 were known.

(2) The parameter sizes

A 512-bit modulus p in ElGamal scheme provides only marginal security from concerted attack. In order to avoid the Index-calculus in $\mathbf{Z}_p$ algorithms, a modulus n of at least 768 bits is recommended. For long term security, 1024-bit or larger modulus should be used. For common system-wide parameters even larger key sizes may be warranted. This is because the dominant stage in the Index calculus algorithm for discrete logarithms in $\mathbf{Z}_p$ is the pre-computation of a database of factor base logarithms, following which individual logarithms can be computed relatively quickly. Therefore computing the database of logarithms for one particular modulus p will compromise the secrecy of all private keys derived using p.

(3) Security of ElGamal signatures

The prime p should be sufficiently large to prevent the Index calculus attack. The integer $p - 1$ should be divisible by a prime number q sufficiently large to prevent a Pohlig-Hellman discrete logarithm attack ([53] page 107). Suppose the generator α satisfies the following conditions:

(a) α divides $(p - 1)$,

(b) Computing logarithms in the subgroup S of order α in $\mathbb{Z}_p$ can be efficiently done (for example, if a Pohlig-Hellman attack can be mounted in S).

Then it is possible for a signature (without knowledge of Alice's private Key) which will be accepted by the verification algorithm (algorithm 11.64 in [53]). To see this, suppose that $p - 1 = \alpha q$. To sign a message m the adversary does the following:

(i) Computing $t = (p - 3)/2$ and setting $r = q$,

(ii) Determining z such that $\alpha^{qz} = y^q \pmod p$ where y is Alice's public key. This is possible since α^q and y^q are elements of S and α^q is a generator of S.

(iii) Computing $s = t \cdot \{h(m) - qz\} \pmod{(p - 1)}$, where h(m) is Hash function (see Appendix J).

(iv) The pair (r, s) is a signature on m that will be accepted by the verification algorithm.

This attack works because the verification equation $r^s y^r = \alpha^{h(m)} \pmod p$ is satisfied. To see this, first observe that

$$\alpha q = -1 \pmod p,$$

so that $$\alpha = -q^{-1} \pmod p,$$

and that $$q^{(p-1)/2} = -1 \pmod p.$$

The latter congruence follows from the fact that α is a generator of $\mathbf{Z}_p$ and $q = -\alpha^{-1} \pmod p$. From these, one deduces that $q^t = q^{(p-1)/2} q^{-1} = -q^{-1} = \alpha \pmod p$. Now $r^s y^r = (q^t)^{[h(m) - qz]} y^q = \alpha^{h(m)} \alpha^{-qz} y^q$ that the conditions specified in (iii) above are trivially satisfied. The Pohlig-Hellman attack can be avoided if α is selected as a generator for a subgroup of $\mathbf{Z}_p$ of prime order rather than a generator for $\mathbf{Z}_p$ itself.

4.1.1.3 The Security in the Elliptic Curve System

The security of elliptic curve cryptosystem is based on the difficulty of the Discrete Elliptic logarithm problem, in other words based on the difficulty of computing reversibly the structure of elliptic curve group which is an Abelian group. To get a better perspective on the discussion of security here, it is assumed that the curves are over the field of $F_{2^{155}}$.

(1) Security of the ECCs over finite fields

The discrete logarithm problem of elliptic curves refers to the elliptic logarithm problem as following. Let $E(\mathbf{F}_q)$ be an elliptic curve over $\mathbf{F}_q$ and P a point in $E(\mathbf{F}_q)$. For any point R $\in \langle P \rangle$ (which is the subgroup generated by P), determine an efficient method to find the integer k, $0 \leq k \leq \#P - 1$, where #P is the order of P (see Appendix A-2) such that kP = R.

The most powerful general algorithm known at present is the *baby-step giant-step* algorithm of Shanks. This method requires $O(\sqrt{P})$ in both time and space. Using a method due to Pollard [54] one can reduce the storage requirement. Pollard's method requires about $\sqrt{P}$ iterations on the elliptic curve where everyone iteration requires 3 elliptic curve additions. Since each addition on the curve requires 13 field multiplications and each field multiplication in $F_{2^{155}}$ takes 155 clock cycles, it follows that to determine one elliptic logarithm requires on average about $6045\sqrt{P}$ (where $6045 = 3 \times 13 \times 155$) clock cycles. If the order of the curve E contains a prime factor with at least 36 decimal digits, then the number of clock cycles to find a logarithm on the curve is about 6×10^{21}. Since the current computer runs at 300 MHz (3×10^8 cycles per second), and if one uses 10^3 devices in parallel, the time to find one logarithm is 2×10^{10} seconds or at least 200 years. Provided that the square root attacks are the best attacks on the elliptic logarithm problem, it is shown that an elliptic curve over F_{2^m} with m = 130 provides very secure systems. It is important to observe that the square root attacks require $O(\sqrt{P})$ iterations for each logarithm to be found.

The most successful attack on the elliptic logarithm problem is a method due to Menezes, Okamotom, and Vanstone [30]. In order to describe this method, first requires some terminology.

Let E be an elliptic curve over $\overline{\mathbf{F}}_q$, which is the algebraic closure of $\mathbf{F}_q$. $E(\mathbf{F}_q)$ is the set of all points in E with coordinates from $\mathbf{F}_q$. $E(\mathbf{F}_q)$ has finitely many points, whereas E has infinitely many. Define $E[n] = \{P \in E : nP = \mathcal{O}\}$. E[n] is called the set of n-torsion points of E. Now for each n, gcd(n, q) = 1, there exists a positive integer k such that $E[n] \subseteq$

$E(F_{2^k})$ and an isomorphism from E[n] to a subgroup of F_{2^k} can be computed using the Weil pairing (see Appendix E). Miller [55] has shown that there exists a random polynomial time algorithm for computing the Weil pairing. These results form the basis for the Menezes Okamoto Vanstone (MOV) attack (see Appendix G).

The MOV attack in the case of supersingular curves becomes a subexponential attack. This happens because it can be shown that all supersingular curves have very small values of k associated with them ([52] page 810). However, non-supersingular curves have large values of k associated with them. If $k > \log^2 q$, then the index calculus methods in F_{2^k} become fully exponential and the MOV attack is worse than the square root attacks. It notes that a necessary condition for $E[n] \subseteq E(F_{2^k})$ is that $n \mid q^k - 1$.

As an example of the above discussion, suppose one has a non-supersingular curve $E(\mathbf{F}_q)$ where $q = 2^{155}$. It is known [38] that $E(\mathbf{F}_q) \cong Z_{n1} \times Z_{n2}$ where $n_2 \mid n_1$. Suppose also that p is a prime dividing n_1 and that p has about 40 decimal digits (It is possible to find curves over $\mathbf{F}_q$ whose order is divisible by a prime factor with up to 46 decimal digits). Now if the smallest value of k for which $E[n_1] \subseteq E(F_{2^k})$ is at least 10, then the MOV attack requires an Index calculus attack in **F** field with more than 1500 bits. For this elliptic curve, the most efficient way to compute elliptic logarithms is by one of the Square root attacks, and these are infeasible for numbers of this size as pointed out above.

(2) The determination of the number of points,

Determining the number of points on an elliptic curve is important for two reasons. First is that the order must have a prime divisor large enough to give adequate security against one of the Square root attacks. Second is to permit signatures that are as small as possible. The supersingular cases can be determined theoretically as [56]. Computing the number of points is a more difficult problem. Here it will examine the case $q = 2^{155}$ in more detail.

The first case is the class of supersingular curves over $F_{2^{155}}$. Since m = 155 is odd, there are only 3 curves to examine (refer to chapter 3.3). For the curve $y^2 + y = x^3$, the k value is

2, so regardless of the size of the prime factor, dividing $2^{155} + 1$ and Index calculus attack in $F_{2^{310}}$ will compromise the system. The largest prime factor dividing $2^{155} + 1$ has 17 digits. For the other two supersingular curves over $\mathbf{F}_q$, the reduction algorithm requires computing logarithms in $F_{2^{620}}$, which is out of range for existent discrete logarithm methods. For the curve with order $2^{155} + 1 - 2^{78}$, the largest prime divisor has 20 digits; and for the one with order $2^{155} + 1 + 2^{78}$, there is a prime factor with 26 digits. Both of these curves do not provide enough high security from the Square root attack. If the supersingular curves are over $\mathbf{F}_q$ where $q = 2^{310}$ (i.e., a quadratic extension of $F_{2^{155}}$), the situation is much better. For the curves with order $q + 1 + \sqrt{q}$, the largest prime divisor has 65 decimal digits. For the curves with order $q + 1 - \sqrt{q}$, the largest prime divisor has 54 decimal digits. There are 4 supersingular curves of this type and each of these under the reduction attack requires the computation of discrete logarithms in $F_{2^{930}}$. These curves provide very high security.

Second case is the class of non-supersingular curves over $\mathbf{F}_q$ where $q = 2^{155}$. The number of this curves is $2(2^{155} - 1)$. This large choice of curves makes it possible to find large numbers of curves over this field for which the order of a curve is divisible by a large prime factor. In general, determining the order of an arbitrary non-supersingular curve over $\mathbf{F}_q$ is not trivial and requires some variant of Schoof's algorithm [57]. There is a relatively simple method for computing a fairly large number of non-supersingular curves over $\mathbf{F}_q$ and giving a high level of security. The method will use the Weil Conjecture for lifting curves [52].

Therefore, in order to avoid an easy solution to the discrete logarithm problem using the techniques that apply to any finite Abelian group, which takes approximately $\sqrt{p}$ operations (where p is the largest prime dividing the order of the group). It is important to choose E and q, so that $n = |E|$ is divisible by a large prime ([11] page 206).

4.1.2 The Comparison of Efficiency of the Cryptosystem

Here comparison of the efficiency of the cryptosystems is mainly concerned with two points: the security term and key length. From the description in chapter 3, it is shown that the security of cryptosystem will be determined by the difficulty of reversible computation of Abilene group. In fact, the different architecture of computations is existed in different group of finite field. For example, let GF be a finite group, and let a and b be elements of GF. Then the discrete logarithm problem for GF is to determine a value x (when it exists) such that $a^x = b$. The value for x is called a logarithm of b to the base a, and is denoted by $\log_a b$. It is clear that the difficulty of determining this quantity depends on the representation of GF. For example, if the abstract cyclic group of order m is represented in the form of the integers module m, then the discrete logarithm problem reduces to the extended Euclidean algorithm. However, if m + 1 is prime, and the group is represented in the form of the multiplicative group of the finite field $\mathbf{F}_{m+1}$, the problem is much more difficult to calculate for that representation of group (refer to [52]). If the group is represented as an elliptic curve group over a finite field, then the problem is again much more difficult.

In other words, it is convenient for the users of a public key cryptosystems that the key size be as small as possible. However, most of the known public key cryptosystems are insecure if the key size is smaller than 135 bits. In the RSA system [10] the public key consists of the integers e and modulus N. Since factoring 135 bits integers can readily be done on a microcomputer, the RSA system is insecure for keys of that size. So that although e can be small, N should be at least 512 bits in length. In the ElGamal system [25], the same holds true for the system whose security is based on discrete exponentiation in a finite field. The private key k can be restricted but the public key α^k (where α is a generator of the field) is the size of the field that should be at least 2^{512}.

An advanced candidate is the Elliptic Curve cryptosystem. The size of the group used in the ECC needs only 155 bits, but it offers the same of degree security as the RSA or ElGamal cryptosystem working with 512 bits. The remainder of this chapter will be

organized in two sections: a comparison between the RSA and the ElGamal cryptosystem, and a comparison between the ElGamal and the elliptic curve cryptosystem.

4.1.2.1 Comparison between the RSA and the ElGamal cryptosystem

From the discussion of chapter 3, the modulu N is a composite integer in the RSA system. The security of the RSA cryptosystem depends on the difficulty of factoring the published modulus N. If the number N can be factored, then the secret key (d, N) can be computed and all of Alice's private mail or digital signatures can be read. Therefore, if the RSA system is to be secure, it is certainly necessary that N = pq must be large enough that factoring it will be computationally infeasible. Recently Odlyzko [4] has forecast that a 512-bit module will be vulnerable to factorization in a couple of years, and is therefore not suitable as a long-term basis for security protection. It is likely that a 1024 bit RSA be become common in the near future. Though it will probably remain secure for many years [18], it requires too much memory for the multi-application smart card.

It is also been shown that the security of the ElGamal cryptosystem is based on the difficulty of the discrete logarithm problem in $\mathbf{Z}_N$. If the discrete logarithm problem in $\mathbf{Z}_N$ can be solved in polynomial time, then N can be factored in expected polynomial time ([53] page 114). In other words, the discrete logarithm problem in $\mathbf{Z}_N$ is no harder than the problems of factoring N and computing discrete logarithms in $\mathbf{Z}_p$ for each prime factor p of N. When utilizing finite fields GF(p), whether p is prime or $p = 2^k$, it is necessary to ensure that p – 1has a large prime factor, otherwise it is easy to find discrete logarithms in GF(p). This restriction is also similar to the need to choose the secret primes in the RSA system carefully [58, 59]. That is the reason that the key length in the ElGamal cryptosystem should be the same as the RSA cryptosystem. Currently p should be at least 512 bit modules which is secure from factorization for a only a few years, and which is only suitable for the short-term protection of secrets. If p were to increase to 1024-bit prime, it will provide over 100 years long-term protection of secrets [60].

So it can be said that the complexity for finding out the discrete logarithms in a field $\mathbf{Z}_p$ for a prime p is the same as the complexity for factoring an integer N with the same size, where N is the product of two approximately equal primes [35, 61]. That is the reason of requiring the same key length for the same length of security terms.

An important difference between the RSA and ElGamal cryptosystems is the computing architecture of cryptosystem. The RSA is a factoring algorithm, which is a "uniformly secure" system, in the sense that there can be no large sets of "weak messages": if a crypt-analyst can decrypt a fraction of messages encrypted with the RSA cryptosystem, then he could effectively decrypt all messages. Putting it another way, if the RSA cryptosystem offers security for the encrypted messages, then it offers uniformly high security for all messages. This follows from the multiplicative nature of the RSA scheme [62]. The ElGamal system relies on the discrete logarithm algorithm, which is much more difficult to use to decrypt a fraction of the messages, if the crypt-analyst does not know the key. In this sense that the ElGamal cryptosystem is based on a more powerful One way function than the RSA cryptosystem.

4.1.2.2 Comparison between the ElGamal and the Elliptic Curve cryptosystem

The security of the Elliptic Curve cryptosystem is based on the difficulty of the discrete Elliptic logarithm problem, which is analogous to the discrete logarithm problem on an elliptic curve over a finite field GF(q). The computation in the groups used in the ECCs and ElGamal system is the main difference between the two systems.

Recall from chapter 3.3, the points of the elliptic curve E over a finite field K = GF(q), where $q = p^m$, form an Abelian group. In some ways this group is similar to the multiplicative group K* of the field K. For example, Hasse proved that the order of the group $|E_K|$ is equal to $q + 1 - a_{E_K}$, where $|a_{E_K}| \leq 2\sqrt{q}$, and so it has the same asymptotic size as $|E_{K^*}| = q - 1$. Actually, one can obtain K* from the above construction of an additional law on E_K if one lets E_K "degenerate" by letting the cubic on the right in (3.3-4)

obtain a double root. Then if the two slopes at the singular point of E_K are in K, it proves that the set of nonsingular points of E_K (i.e., those whose x-coordinate is not the double root) form a group isomorphic to K*. But unlike K*, which is a cyclic group, the Abelian group E_K for K = GF(q) can either be cyclic or else a product of two cyclic groups. In practice, for a "random" elliptic curve, usually either this group is cyclic or else it can be written as a product with one of the cyclic factors much smaller than the other. For this reason, it seems that the elliptic curve cryptosystem based on the discrete elliptic logarithm are secure over much smaller fields than ElGamal cryptosystem based on the multiplicative group of the field. Also there is much more choice available when working with elliptic curves: for fixed q one has only one group $\mathbf{F}_q$, but one can obtain many groups of curves E by varying the coefficients of the defining equation of the elliptic curve (shown on Table 4-1).

The Addition Formula for E: $y^2 = x^3 + ax + b$,

	a	b	x	y
1)	-2	5	1,318	47,849
2)	4	-1	4,321	284,038
:	:	:	:	:
10)	-19	-51	2,955,980	5,082,205,677
:	:	:	:	:

Table 4-1. Isomorphism Classes of Curves over F_q ($q=2^n$)

The total number of elliptic curves E: $y^2 = x^3 + ax + b$ over $F_{2^{155}}$ is calculated by the equation of the form:

$$\binom{2^{155}}{2} = \frac{(2^{155})}{(2^{155}-2)\times 2!} \approx 2^{309}.$$

Note that here the coefficients of y^2 and x^3 are simplified to be 1. If the coefficient of y^2 or x^3 is selected from the field $F_{2^{155}}$, then the total number of curve E is that

$$\binom{2^{155}}{3} = \frac{(2^{155})}{(2^{155}-3)\times 3!} \approx 2^{462}.$$

Therefore, the size of the group used in the ECC is much smaller than that used in the ElGamal algorithm. The q needs only 155 bits for short-term protection or 201 bits for long-term protection. At these levels it offers the same degree of security as the ElGamal cryptosystem working with respectively 512 bits and 1024 bits prime, since the Elliptic Curve algorithm uses a different group operation in an Abelian group than multiplication of integers mod q.

Another target to be discussed is the ratio R of the term of security increased to the bit of key size increased, shown as Fig. 4-1.

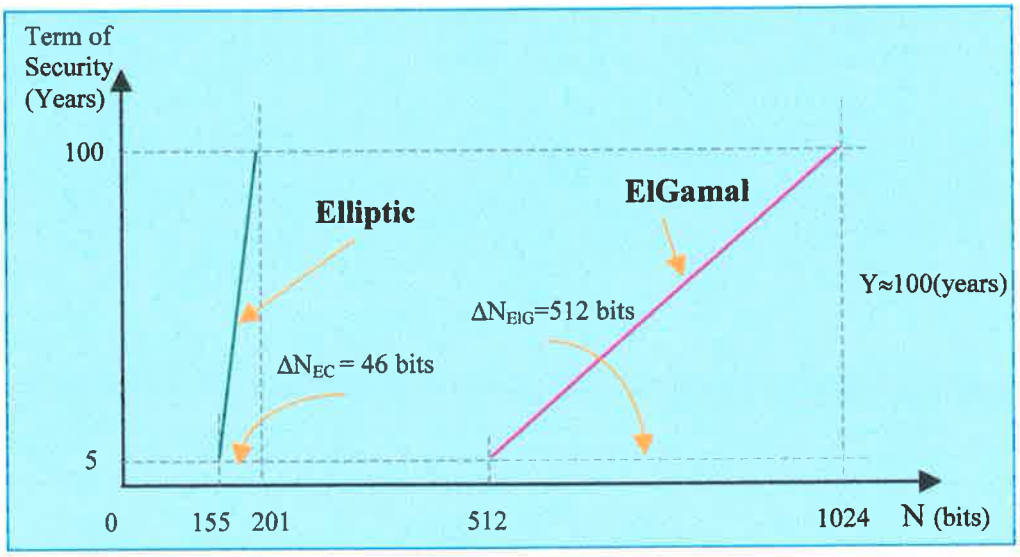

Fig. 4-1. Comparison of the Efficiency between Two Types of Cryptosystem.

A key advantage is that, if the security term increases quickly, the size of the numbers needed to achieve a particular level of security increases much more slowly for the Elliptic Curve system when compared to the ElGamal system. This increase in efficiency is shown as Fig. 4-1. Let $R=Y/\Delta N$, where Y and N denote the security terms in years and the number of bits individually. As previously described, where the size of field is over GF(q), where $q=2^m=2^{201}$, the security term is longer than 100 years[60]. It is reasonable to take $Y\cong 100$ years approximately as the difference between the term based on 155 bits and the term based on 201 bits, and similarly for ElGamal bit size. The Fig. 4-1 shows that the $R_{EIG} = Y/\Delta N_{EIG} = 100/(1024-512) = 0.195$ years/bit in the ElGamal algorithm, but $R_{EC} = Y/\Delta N_{EC} = 100/(201-155) = 2.174$ years/bit in the Elliptic Curve algorithm. It is

clearly shown that the efficiency of R_{EC} is as many as eleven times larger than R_{EIG}. The significant is that the potential problems to require long term security that restricted in the small memory capacity were being resulted. For example, to design a smart card system.

4.2 The Efficiency Trade Off Against the Time of Computation

From the discussion above, it is obvious that the security of the RSA system is dependent on the size of modulus N, which has to be a large prime of more than 512 bits for security purpose. The security of the ElGamal system depends on the same size of modulus N as the RSA system, although the discrete logarithm problem of the ElGamal system is different from the factoring integer problem of RSA system. The large size of modulus N has the benefits of long-term security, but it needs not only more memory for storage, but also more time for calculation.

In the case of the Elliptic Curve cryptosystem, it needs much smaller size bits over GF, but offers the same degree of security as the RSA and ElGamal cryptosystem working with large bits integer N. Therefore, it can efficiently save much memory space, but unfortunately it needs a longer time for computation, since the arithmetic of the group operations is quite complex. This is the main barrier to the practical implementation of the elliptic curve algorithm.

In a discussion of the computing time in cryptosystems, some basic knowledge of the measurement of time in cryptography will be abridged to simplify the topic of computing time (the interested reader should refer to [32]). It starts with the computation of time in public key cryptosystem and the comparison of time in cryptosystem as follows.

4.2.1 The Computation of Running Time in Cryptosystem

The computation of time in cryptosystem is a very complex topic, since it is related to the degree of security of each cryptosystem. The purpose here is not specifically that topic,

thus the introduction of computation of time will be simplified, so that it is simply shows that the running time in Elliptic Curve cryptosystem is harder to satisfy the standard adopted by the ICC.

4.2.1.1 The computation of running time in RSA system,

In the RSA cryptosystem, as N is very large, one has to use multiprecision arithmetic to perform computations in $\mathbf{Z}_n$, and the time required will depend on the number of bits in the binary representation of N.

Suppose N has k bits in binary representation, i.e., $k = \lfloor \log_2 N \rfloor + 1$. It is not difficult to see that an addition of two k-bit integers can be done in time O(k), and a multiplication can be done in time $O(k^2)$. Also a reduction modulo N of an integer having at most 2k bits can be performed in time $O(k^2)$.

Now suppose that $p, q \in \mathbf{Z}_n$. pq (mod N) can be computed by first calculating the produce pq (which is 2k-bit integer), and then reducing it modulo N. These two steps can be performed in time $O(k^2)$. This method is called *modular multiplication* [41].

The computation of a function of the form x^c (mod N) is called *modular exponentiation.* Thus, both of the encryption and the decryption operations in RSA are modular exponentiations. Computation of x^c (mod N) can be done using $c - 1$ modular multiplications. But this is very inefficient if c is large (c might be as big as $\phi(N) - 1$, which is exponentially large compared to k).

The *square-and-multiply* method [63] reduces the number of modular multiplications that require computing x^c (mod N) to at most 2l (where l is the maximum length of number of the bits in the binary representation of c). Since $l \leq k$, it follows that x^c (mod N) can be computed in time $O(k^3)$. Therefore, both encryption and decryption in RSA can be done in

polynomial time (as a function of k, which is the number of bits in one plaintext or ciphertext character).

There are many factoring algorithms at present. The three which are the most effective on large numbers, are the quadratic sieve, the elliptic curve algorithm and the number field sieve. First it is necessary to introduce a notation of "bound" B in Pollard's p – 1 algorithm, before briefly introducing the computations of time required.

The Pollard's p – 1 algorithm has two parameters: the (odd) integer N to be factored, and a "bound" B. Suppose p is a prime divisor of N, and $q \leq B$ for every prime power $q \mid (p - 1)$. Then it must be the case that $(p - 1) \mid B!$ [41]. Let

$$a = 2^{B!} \pmod{N},$$

so that $a = 2^{B!} \pmod{p}$.

since $p \mid N$, then $2^{p-1} = 1 \pmod{p}$.

Since $(p - 1) \mid B!$, then $a = 1 \pmod{p}$. It has that $p \mid (a - 1)$ and $p \mid N$, so that $p \mid d = \gcd(a-1, N)$.

In the p – 1 algorithm, there are B – 1 modular exponentiations, each requiring at most $2\log_2 B$ modular multiplications using square-and-multiply. The gcd computation can be done in time $O((\log N)^3)$ using the Euclidean algorithm. Hence, the complexity of the algorithm is $O(B \log B (\log N)^2 + (\log N)^3)$. If B is $O((\log N)^i)$ for some fixed integer i, then the algorithm is indeed a polynomial time algorithm. However, for such a choice of B, the probability of success will be very small. On the other hand, if the size of B is increased to $N^{1/2}$, then the algorithm will be successful, but it will be no faster than trial division.

The elliptic curve factoring algorithm (note that it is not the elliptic curve logarithm algorithm discussed above) by Lenstra in [64], is in fact a generalization of the p – 1 method. The elliptic curve method depends on the more likely situation that an integer "close to" p has only "small" prime factors. The p – 1 method depends on a relation that

holds in the group $\mathbf{Z}_p$, the elliptic curve method involves groups defined on elliptic curves modulo p.

Assuming the distribution of integers is not divisible by any prime > B in a small interval around p. Lenstra proves that the following probabilistic time estimate for the number of bit operations required produce a nontrivial divisor of N:

$$O(\exp((1+\varepsilon)\sqrt{2\log p \log\log p})),$$

where p is the smallest prime factor of N and ε approaches zero for large p. Since $p \leq \sqrt{N}$, it follows from the above is of the form:

$$O(\exp((1+\varepsilon)\sqrt{\log N \log\log N})).$$

The Quadratic sieve method, developed by Pomerance [65], uses a *factor base*, which is a set β of small primes. First it is necessary to obtain several integers x such that all the prime factors of x^2 (mod N) occur in the factor base β. The idea is to take the product of several of these x's in such a way that every prime in the factor base is an even number of times. This then gives a congruence of the desired type $x^2 = y^2$ (mod N), which will lead to a factorization of N.

Suppose $\beta = \{p_1, \ldots, p_B\}$ is the factor base. Let C be slightly larger than B (i.e., C = B + 10). Suppose one has obtained C congruences:

$$x_j^2 = p_1^{\alpha_{1j}} \times p_2^{\alpha_{2j}} \times \cdots \times p_B^{\alpha_{Bj}} \pmod{N}, \qquad (1 \leq j \leq C)$$

For each j, consider the vector $a_j = (\alpha_{1j} \pmod 2, \ldots, \alpha_{Bj} \pmod 2) \in \mathbf{Z}_2^B$. If a subset of the a_j's that sum modulo 2 to the vector (0, …, 0), can be found, then the product of the corresponding s_j's will use each factor in β an even number of times.

Pomerance [65] has proved that the expected running time of the quadratic sieve factoring method is asymptotically

$$O(\exp((1+\varepsilon)\sqrt{\log N \log\log N})),$$

for any $\varepsilon > 0$. There is a fairly large space requirement, also of the form $\exp(C\sqrt{\log N \log\log N})$.

The number field sieve method is a more recent factoring algorithm. It also factors N by constructing a congruence $x^2 = y^2 \pmod N$, but it does so by computation in rings of algebraic integers. It has a running time that is asymptotically

$$O(\exp(2+\varepsilon)(\log N)^{1/3}(\log\log N)^{2/3})).$$

In practice, it appears to be the fastest method for factoring numbers that are at or beyond the current (1994) upper limits of what can be factored, i.e., more than 150 digits. The running time of them is converged in Table 4-1.

Factoring Algorithm	Running Time
Quadratic sieve	$O(\exp((1+\varepsilon)\sqrt{\ln N \ln\ln N}))$.
Elliptic curve	$O(\exp((1+\varepsilon)\sqrt{2\ln p \ln\ln p}))$.
Number field sieve	$O(\exp(2+\varepsilon)(\ln N)^{1/3}(\ln\ln N)^{2/3}))$.

Table 4-1. The running time of three cryptography algorithms

The ε denotes a number that approaches 0 as $N \to \infty$, and p denotes the smallest prime factor of N. In the worst case, $p \approx \sqrt{N}$ and the asymptotic running times of the quadratic sieve and elliptic curve algorithms are the same. But in such a situation, the quadratic sieve generally is more advanced than the elliptic curve. The elliptic curve method is more useful if the prime factors of N are of different size [41].

The number field sieve is the most recent of the three algorithms. It seems to have great potential since its asymptotic running time is faster than either quadratic sieve or the elliptic curve. Although still in developmental stages, people have speculated that the number field sieve method might prove to be faster for numbers having more than 130-150 decimal digits [32].

Currently the industry of smart card crypt-processor has developed rapidly (refer to chapter 1). A comparison of the computing time of various chips produced by three main companies is shown in Table 4-2.

Chips Types	Company	RSA (512-bit) (ms)		RSA (1024) (ms)		Clock Rate	CMOS Technology
ST16CF54	Thomson	385	150(C)	---	---	5 MHz	1.2 μm
SLE44C200	Siemins	220	60(C)	---	450(C)	5 MHz	1.0 μm
MC68HC05	Motorola	500	125(C)	5600	1500(C)	5 MHz	1.2 μm

Table 4-2. Comparison of the Computing Time of Smart Card Chips

Note that C means Signature time with Chinese Remainder Theorem.

4.2.1.2 The computation of running time in ElGamal system

The Index calculus method for computing the discrete logarithm is considered to be one of the best factoring algorithms. Recalling chapter 3.2, this algorithm consists of three stages. The first two stages belong to pre-computation, and the third stage finds an individual discrete logarithm. There is a difference in the architecture of the ElGamal system compared to the RSA system. Consequently, there is also a difference between the method of computation in both cryptosystems. The computing time of the ElGamal system will consists of two parts, the running time of pre-computation and the time to find an individual discrete logarithm. The ElGamal system presents an asymptotic running time of

$$O(\exp((1+o(1))\sqrt{\ln p \ln\ln p})),$$

for the pre-computation phase, and the much shorter running time of

$$O(\exp((1/2+o(1))\sqrt{\ln p \ln\ln p})),$$

for finding an individual logarithm once the pre-computation is done. The running time of the pre-computation is roughly the same as that of the fastest known algorithms for factoring integers of size about p. The details are referred to in [35].

In practice, it is quite different from the situation of the finite field $GF(2^n)$ (or more generally, fields $GF(p^n)$ where p is held to be fixed and $n \to \infty$) where the first author [2] has shown that discrete logarithms can be computed in time

$$O(\exp(cn^{1/3}(\ln n)^{2/3}))$$

for some $c > 0$ depending on p. The Index calculus method can be modified to work in these fields. The pre-computation time is calculated to be

$$O(\exp((1.405 + o(1))\, n^{1/3}(\ln n)^{2/3})),$$

and the time to find an individual discrete logarithm is

$$O(\exp((1.098 + o(1))\, n^{1/3}(\ln n)^{2/3})).$$

Therefore, for large values of n (say $n > 800$), the discrete logarithm problem in $GF(2^n)$ is thought to be intractable provided 2^n has at least one large prime factor (in order to resist a Pohlig-Hellman attack).

The fastest known general algorithm for computing discrete logarithms modulo p is based on the Number-Field sieve and has asymptotic running time [35]

$$O(\exp((c(\ln p)^{1/3}(\ln\ln p)^{2/3}))$$

for some small constant c. At present the fastest implementations of discrete logarithm algorithms have larger asymptotic running time

$$O(\exp((c(\ln p)^{1/2}(\ln\ln p)^{1/2})).$$

Computing discrete logarithms modulo a prime p seems at present to be infeasible for primes of more than 120 digits [66].

4.2.1.3 The computation time in the Elliptic Curve system,

As described in chapter 4.1, the creation of efficiency in the ECC is based on the principle and complex computation on the Abelian group with the algebraic operation '+' of the Elliptic Curve algorithm. Thus, it leads to great complexity of computation and more time required for the pre-computation stage of the ECC. , Further details may be found in [28]. The brief description of the Elliptic Curve algorithm is listed below,

--- The working on the Elliptic Curve:

a. Adding and Doubling Points,

b. Choosing the curve,

c. Computing a Multiple of a Point.

--- The more complex computation of field operations:

a. Representation of the Field Elements,

b. Addition and Multiplication of the Field Elements,

c. Modular Reduction,

d. Computing Reciprocals.

These complex operations cause the time required in processing the ECC to become gradually longer. At present there is no algorithm faster than the Baby-step Giant-step algorithm (see Appendix F) for an ECC. The whole algorithm takes

$$O(q^{1/4}(\log q)^2/L^{1/2})$$

bits operations, and requires

$$O(q^{1/4}(\log q)/L^{1/2})$$

bits of storage [5]. Recall that $\#E(\mathbf{F}_q) = q + 1 - t$, the calculation of t modulo ℓ using Schoof's algorithm for small primes ℓ is very simple. However, since $\deg(f_\ell(x)) = (\ell^2 - 1)/2$, the calculation quickly becomes infeasible as the value of ℓ increases. J.Buchmann and V. Muller [5] combined Schoof's algorithm with Shanks' Baby-step Giant-step

method to count the points on a single randomly chosen elliptic curve over $\mathbf{F}(2^m)$. The algorithm was implemented in the C language on a SUN-2 SPARC-station with 64 Mbytes of memory [67], as shown in table 4-3.

$q = 2^m$ (m bits)	Size of space searched (bits)	Time required (Min.: sec.)
m = 100	2.8×10^8	1: 43
m = 135	1.2×10^{11}	51: 22
m = 148	3.6×10^{11}	100: 42
m = 155	6.7×10^{10}	44: 11

Table 4-3 Times for computing the points on Elliptic Curve over $\mathbf{F}(2^m)$

From a practical point of view, curves over fields of characteristic 2 are more attractive, since the arithmetic in $GF(2^m)$ is easier to implement in hardware than the arithmetic in GF(p) (p is odd). Schoof's algorithm for counting the points on an arbitrary curve over $GF(2^m)$ has improved the actual running time [68]. For the sake of comparison, the table here gives some total time (seconds) of computation using the field $GF(2^m)$ and GF(p), where p is the least prime greater than 2^m, as shown in Table 4-4. It considers the 50 random curves of the equation $y^2 = x^3 + x + b$ for $1 \le b \le 50$ for each of these primes, and has been done on a DecAlpha 3000/500 machine.

The dynamic strategy for $GF(2^m)$			Best running times for GF(p)		
$GF(2^{65})$	$GF(2^{89})$	$GF(2^{105})$	$GF(2^{65}+131)$	$GF(2^{89}+29)$	$GF(2^{105}+39)$
8.3 (s)	19.6 (s)	43.3 (s)	8.8 (s)	25.3 (s)	51.6 (s)

Table 4-4. Comparison of Running Time for Finite Field $GF(2^m)$ and GF(q)

The timing of the ECC system that calculated from making a few measurements on the SPARC (25 MHz) is shown in Table 4-5 [28]. Assuming the clock rate in the chip is 5 MHz.

Cryptography Algorithm	SPARC (25 MHz)	Chip (5 MHz)
Encryption (155 bits)	116 (ms)	580 (ms)
Decryption (155 bits)	94 (ms)	470 (ms)
Signature	24 (ms)	120 (ms)
Checking signature	220 (ms)	1100 (ms)

Table 4-5. Times for ECC operations

It is pronounced that the computing time for encryption and decryption on a randomly chosen Elliptic Curve will be longer than 450 ms of a standard operation at 5 MHz [5]. This is the main reason that the ECC system, with only 155-bit size of the group field, is unavailable in practice. However it is said that the efficiency of the ECC occurs at the expense of the time of computation.

4.2.2 The Comparison of Running Time in the Cryptosystems

In the comparison between the RSA and ElGamal system, recalled from chapter 4.1, it is shown that the complexity of finding discrete logarithms in a prime field GF(q) for a general prime q is essentially the same as the complexity of factoring an integer N of about the same size (where N is the product of two approximately equal primes). In other words, both the RSA and ElGamal cryptosystem have similar running times for computations, which also has been proved by the equations of computing time in the last subchapter.

Discrete Logarithm	Running Times	
	Pre-computing Time	Finding discrete logarithm
Index Calculate algorithm GF(q)	$O(\exp((1 + o(1))(\ln q \ln\ln q)^{1/2})$	$O(\exp((1/2 + o(1))(\ln q \ln\ln q)^{1/2})$
Index Calculate algorithm GF(2^m)	$O(\exp((1.41 + o(1))\, m^{1/3}(\ln m)^{2/3})$	$O(\exp((1.09+o(1))m^{1/3}(\ln m)^{2/3})$

Table 4-6. Comparison of Running Time between finite field GF(q) and GF(2^m)

In comparison between the factoring algorithm shown in Table 4-1 and the discrete logarithm algorithm shown in Table 4-6, both of the running times are nearly the same. Since the time of finding discrete logarithm is much less than the pre-computing time, which is reasonable to be abridged, as shown in Fig. 4-2 (using Index calculate algorithm in GF(q), where $q = 10^5$).

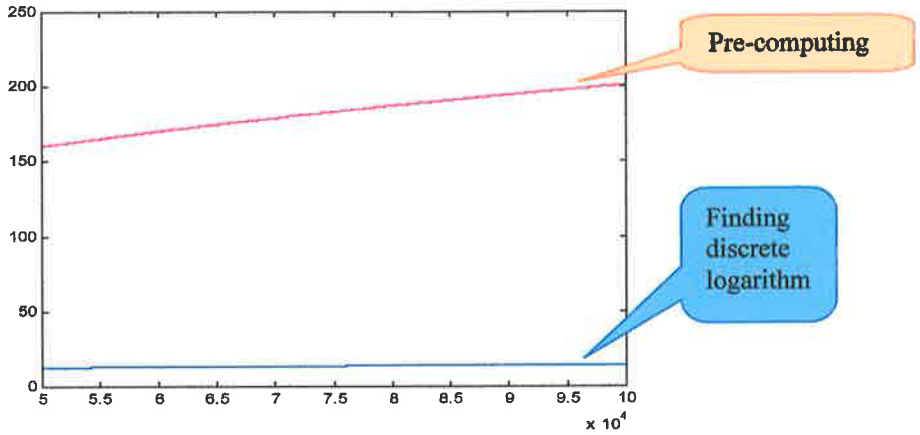

Fig. 4-2 Comparison of the Time of Two computing stages

In the comparison between the ElGamal and the Elliptic Curve system, it is well known that the "Baby-step giant-step" algorithm requires time to be fully exponential in the length of the largest prime factor of the order of the group. This situation is in contrast with that of the classical discrete logarithm problem in the multiplicative group of a finite field. There the "Index calculus" algorithm making the time to solve the discrete logarithm in GF(q) is bounded by the equation of $O(\exp((c(\ln q)^{1/3}(\ln\ln q)^{2/3}))$ for a fairly small constant c. Therefore, from a practical point of view, there are both positive and negative features using ECCs.

On the positive side, the ECCs potentially provides equivalent security to the existing public key schemes, but with shorter key lengths giving smaller bandwidth and memory requirements. This will be a crucial factor in the design of Smart Cards. On the negative side, the computation time of the ECCs is much more than that of the ElGamal system over a finite field. Therefore, the computation times on the ECCs are much longer than on the ElGamal cryptosystem. The running times of modulus and exponent operation between the three kinds of cryptosystems are listed in Table 4-7.

Algorithms	Running Times			
	135 (bits)	201 (bits)	512 (bits)	1024 (bits)
RSA (CRT)	---	---	60 ms	240 ms
RSA (standard)	---	---	220 ms	450 ms
Discrete Logarithm	---	---	270 ms	520 ms
Elliptic Curve	660 ms	1020 ms	--	--

Table 4-7. The Running Time of Modulus and Exponent Operation

Note that: (1) It is implementation on the SLE44C200 chip card processor at 5 MHz (for selected operation).

(2) CRT means running with Chinese Remainder Theorem.

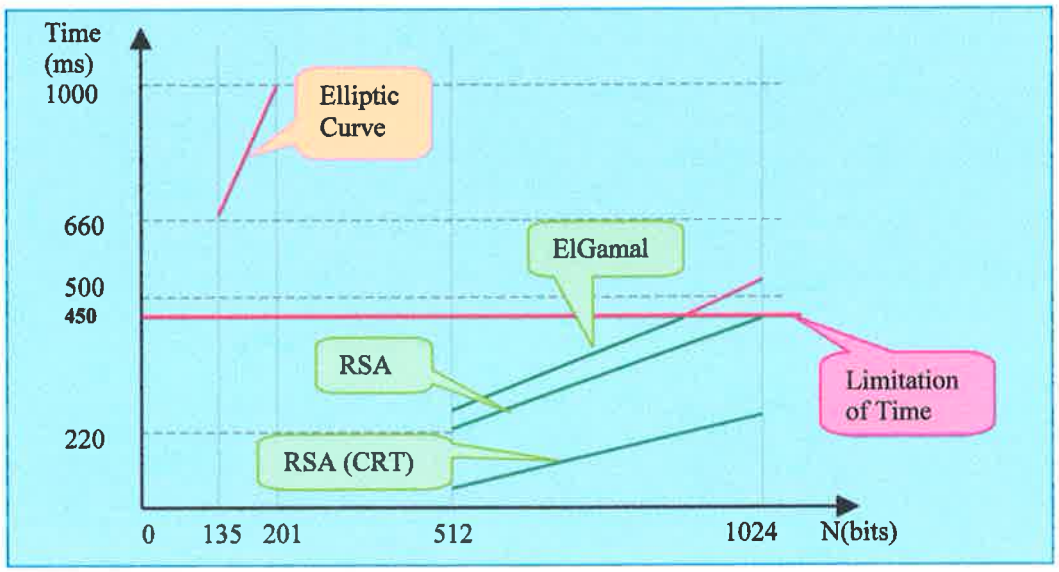

Fig. 4-3. Comparison of the Running Times between Three systems

In Fig.4-3, it is clearly shown that the running times for both the RSA system based on a factoring large integer and ElGamal system based on a classical discrete logarithm efficiently follow the 450ms running times of limitation made by ISO 7816 and satisfy its requirement in practice. However, it needs more memory to store the longer length of key for cryptography. The running times of the ECCs are much longer than the RSA and ElGamal systems, since its computing architecture is much more complex. This is the main reason that the ECC, with only 155-bit size of the group field, has been unavailable in practice, even though it has equivalent security to the other two public key schemes with the same length of the key. However it is said that the efficiency of the ECC occurs at the expense of the time of computation.

Summary:

This chapter has discussed efficiency and running times. From the computation and comparison of efficiency for the three kinds of public key cryptosystem, the advanced efficiency of the ECCs has clearly shown that the ECCs is a desirable cryptography algorithm. It potentially provides equivalent security compared with the existing public key schemes, but with shorter key lengths allowing smaller bandwidth and memory requirements and with the potential to be a crucial factor in the design of Multi-application smart cards. From the computation and comparison of running times, the disadvantages of a lengthy computation time in the ECCs has shown that the efficiency trades off with the running times, which is the main reason preventing use of the ECCs in practice.

5. Conclusion

The purpose of this research is to explore an efficient cryptographic algorithm to achieve a high degree of security for multi-application smart card (**MSC**) in the future. The aim of the MSC is ultimately to arrive at the day where people will only need to carry one smart card in their wallet. The security, convenience and integrity of a single smart card will have beneficial applications as varied as daily services. These advance performances will need a large number of memory spaces. But the chip on the smart card is restricted in size by the ISO 7816 standard, with the result that the memory capacity of the chip is not big enough to store the larger and complex programs for the operating system and to store protocol codes of the cryptography algorithms for factoring large integers. Within this limited memory space, cryptographic keys will be stored in EEPROM, the ROM mask normally store the operating system and higher level instructions which execute cryptographic algorithms. So investigation for an efficient cryptography algorithm which satisfies both the security levels based on factoring large integers and the memory space of the restricted size chip is of importance.

1. Outline of the thesis

The basic design and development of smart card technology for future use has been done. It is shown that the MSC is much more complex than SSC in both physical and logical constructions. More importantly, it is necessity to build an operating system and database inside the chip to achieve security. For these requirements, the MSC will need a large amount of memory, but the chip has only a small memory space since it is restricted in size by the ISO 7816 standard. This is the main barrier in the development of the MSC. Further more it is not certain how far the silicon industry technology will go towards that goal of having big enough memory size. The significant of this investigation is that the decision about the direction of smart card technology development will determine the achievement of MSC in future, or the changes introduced today will determine tomorrow's result.

The public key cryptosystem is the best secure system at present. Three kinds of public key cryptosystem have been discussed. It has be shown that the elliptic curve algorithm based on discrete elliptic logarithm has a different method than the ElGamal cryptosystem based on classical discrete logarithm in a finite field. Foremost it has the addition group operation, while later it has the multiple group operation. This is the main reason why the ECC potentially provides equivalent security to the existing public key schemes, but with shorter key lengths requiring smaller bandwidth and memory and which are a crucial factor in the design of the MSC.

For more understanding of the creation of the efficiency of elliptic curve cryptosystem, the minimum knowledge of the principle of mathematics related to three kinds of cryptosystems is introduced in this thesis, showing the different architecture of computation and comparing the efficiency and running time between three public key cryptosystem. In addition, it will give a source of basic notation showing that the efficiency of the ECC is created by the Abelian group operation, which is especially important for the MSC where the chip is restricted in size. The ECCA is a unique public key cryptography algorithm based on the mathematical principles of the Abelian group operation, and offers an efficient solution to achieve a high degree of security by using smaller numbers of key. Unfortunately, the lengthy running time is created by the complexity of the computing method of the ECC, which is the principal deterrent to developing the MSC.

The comparison of the efficiency and running time clearly shows that the advanced efficiency of ECCs is as many as eleven times larger than RSA and ElGamal systems. The disadvantage of the lengthy running time operated by ECCs is gradually longer than RSA and ElGamal systems, and exceeds the 450 ms standardized by ISO 7816. Therefore it gives a significant result that the advanced efficiency of the ECCs is trading off the redundant running time, which is the main reason that the ECC is presently unavailable for practice application.

2. The Elliptic Curve Cryptosystem in Future

In encryption and decryption of the MSC, performing the computations with limited memory in a reasonable time is a difficult task. A reasonable thread in the elliptic curve algorithm shown above is to increase the memory space on the chip and to reduce the lengthy running time.

The responsibility for the former belongs to the silicon industry technology, which expects that the memory size will increase to more than 120 Kbytes (refer to chapter 1) in a chip sized $25mm^2$ using 0.3 μm CMOS technology [17] in future. Reduction in running time requires increased speed in integer arithmetic by developing and improving the data structures and algorithm design. In particular, the arithmetic in characteristic 2 ($p = 2^m + r$) is the most efficient means to speed up the implementation of the ECCs.

Currently, the memory size in the chip is smaller than 30 Kbytes. It makes the research of data structures and algorithm design less attractive for the ECC, since these methods require more memory space. However, with the growth of memory capacity, the data structures and algorithm design of the ECC will be developed and improved enormously since the ECC requires smaller key length, and the additional memory space required for the improvement of data structures and algorithm design is also not large. Therefore, it will be possible to reduce lengthy running time to an effective time of 450ms that is suitable to ISO 7816 standard.

The ECC is a unique efficient algorithm for the achievement of MSC, but the key length of ECC stays at only 135 bits ~ 155 bits, since its running time is restricted by the complexity of the computing operation. These bits are only suitable for short-term security. Further study to expand to field $\mathbf{F}(2^{201})$ (201 bits) would provide a significant solution to the problem of the smaller memory space and the longer-term security. Under these conditions, the MSC with a high degree of security will indeed be practical.

Appendix A. Number Theorem

A-1 One-way Function

A one-way function is a function that is easy to compute, but difficult to invert by any known method as shown in Fig. A-1. A Polynomial function with many terms is one example. Other examples abound, but there is no rigorous proof that any function is truly one way in the sense that a simple inversion technique is impossible.

If extra information permits one to easily invert a one-way function, the extra information is called "trapdoor", then this function is called a trapdoor one-way function. In public key cryptography, the private key provides the extra information or "trapdoor", whereas the public key specifies the one-way function.

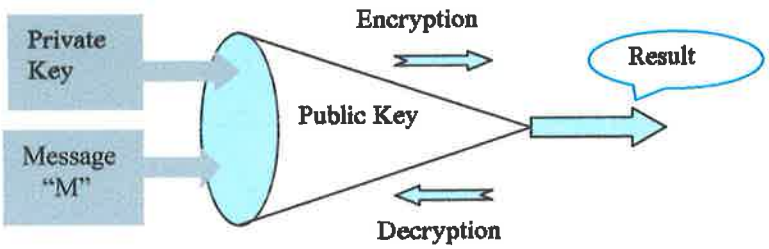

Fig. A-1. The Framework of One Way Function

A-2 Number Theory

Division Algorithm. If a is an integer and b is a positive integer, then there are unique integers k and l such that $a = kb + l$ with $0 < k < b$.

Euclidean Algorithm. Let r_0 and r_1 be non-negative integers with $r_1 \neq 0$. If the division algorithm is applied successively to obtain $r_j = k_{j+1}r_{j+1} + r_{j+2}$ with

$0 < r_{j+2} < r_{j+1}$ for $j = 0, 1, \ldots, n - 2$ and if $r_{n-1} = k_n r_n$, then $(r_0, r_1) = r_n$, the last non-zero remainder.

Proof: Successive applications of the division algorithm yield:

$$r_0 = k_1 r_1 + r_2, \qquad 0 < r_2 < r_1$$
$$r_1 = k_2 r_2 + r_3, \qquad 0 < r_3 < r_2$$
$$\vdots$$
$$r_{n-2} = k_{n-1} r_{n-1} + r_n, \qquad 0 < r_n < r_{n-1}$$
$$r_{n-1} = k_n r_n.$$

Eventually, a remainder of zero is obtained because the $r_1 > r_2 > \ldots > r_n > 0$, which ensures that the sequence of remainders has fewer than r_1 terms. Lemma 1 implies that $(r_0, r_1) = (r_1, r_2) = \ldots = (r_{n-1}, r_n) = (r_n, 0) = r_n$. Thus, $(r_0, r_1) = r_n$.

- **The Euler totient function:**

  $\varnothing(N)$ is the number of positive integers less than N that are relatively prime to N. For prime p, $\varnothing(p) = p - 1$.

- **Euler's Theorem:**

  If N is a positive integer and a is an integer with (a, N) = 1, then

$$a^{\varnothing(N)} = 1 \pmod N$$

or, equivalently

$$a^{\varnothing(N)} \pmod N = 1$$

So Euler's theorem can be used to find inverses modulo N:

$$ax \pmod N = 1$$

or

$$ax = 1 \pmod N \qquad \text{(A.2-1)}$$

where (a, N) = 1. If $1 \le x \le N$, the solution can be expressed as

$$x = a^{\varnothing(N)-1} \pmod N \qquad \text{(A.2-2)}$$

- **Theorem A-2.1** (Theorem 7 in [21]):

  If p and q are primes and N = pq, then $\varnothing(N) = (p-1)(q-1)$.

- **Definition A-2.2:** **The Order**

  --- **Order of a field** (in [73] page 32): A finite field is a field which has a finite number of elements, this number being called the order of the field.

  --- **Order of a point** (in [5] page 26): Let E is a torsion group, i.e., for each point P $\in$ E there is a positive integer k such that kP = $\mathcal{O}$. The smallest such integer is called the order of point P. An n-t0rsion point is a point P $\in (\overline{F}_q)$ satisfying nP = $\mathcal{O}$.

  --- **Order of a curve** (Lemma 2.9 in [5]): Let $\#E(\mathbf{F}_q)$ denotes the number of points on elliptic curve E defined over $\mathbf{F}_q$. By **Hasse's Theorem**: Let $\#E(\mathbf{F}_q) = q + 1 - t$, then $|t| \le 2\sqrt{q}$.

  There exists an elliptic curve E over $\mathbf{F}_q$ such that $E(\mathbf{F}_q)$ has order $q + 1 - t$ if and only if one of the following conditions holds:

  (I) $t \neq 0 \pmod p$ and $t^2 \le 4q$.

  (II) m is odd and one of the following holds:

  (1) $t = 0$,

  (2) $t^2 = 2q$ and $p = 2$,

  (3) $t^2 = 3q$ and $p = 3$.

  (III) m is even and one of the following holds:

  (1) $t^2 = 4q$,

  (2) $t^2 = q$ and $p \neq 1 \pmod 3$,

  (3) $t = 0$ and $p \neq 1 \pmod 4$.

- **Definition A-2.3: Factor Base**

A Factor Base (FB) is a set $B = \{p_1, p_2, \ldots, p_h\}$ of distinct primes, except that p_1 may be the integer –1. We say that the square of an integer b is a B-number (for a given n) if the least absolute residue $b^2 \pmod n$ can be written as a product of numbers from B.

Example: For n = 4633 and B = {-1, 2, 3}, the squares of the three integers 67, 68 and 69 are B-numbers, because $67^2 = -144 \pmod{4633}$, $68^2 = -9 \pmod{4633}$, and $69^2 = 128 \pmod{4633}$.

Let $\mathbf{F}_2^h$ denote the vector space over the field of two elements, which consists of h-tuples of zeros and ones. Given n and a factor base B containing h numbers, we show how to correspond a vector $\varepsilon \in \mathbf{F}_2^h$ to every B-number. Namely, we write $b^2 \pmod n$ in the form $\prod_{j=1}^{h} p_j^{\alpha_j}$ and set the j-th component εj equal to $\alpha_j \pmod 2$, i.e., $\varepsilon_j = 0$ if α_j is even, and $\varepsilon_j = 1$ if α_j is odd.

- **Definition A-2.4: Division Polynomials**

Let $q = 2^m$, and let E be a non-supersingular elliptic curve over $\mathbf{F}_q$, and has the form

$$y^2 + xy = x^3 + a_2x^2 + a_6, \qquad \text{(A2.4-1)}$$

where $a_2 \in \{0, \gamma\}$, $\gamma \in \mathbf{F}_q$ being a fixed element of trace 1, and $a_6 \in \mathbf{F}_q^*$. The *division polynomials* $f_n(x) \in \mathbf{F}_q[x]$ associated with the non-supsingular curve E given by the equation (A2.4-1) [77]:

$$f_0 = 0$$
$$f_1 = 1$$
$$f_2 = x$$
$$f_3 = x^4 + x^3 + a_6$$

$$f_4 = x^6 + a_6x^2$$

$$f_{2n+1} = f^3_n f^3_{n+2} + f_{n-1}f^3_{n+1}, \qquad n \geq 2$$

$$xf_{2n} = f^2_{n-1}f_n f_{n+2} + f_{n-2}\, f_n\, f^2_{n+1}, \qquad n \geq 3.$$

The polynomials f_n are monic in x, and if n is odd then the degree of f_n is $(n^2 - 1)/2$.

- **Definition A-2.5: Endomorphism**

Let $End_K E$ denotes the ring of *endomorphisms* of E that are defined over $\mathbf{F}_q$. For any integer m, the multiplication-by-m map $P \mapsto mP$ is an *endomorphism* of E, and hence $\mathbf{Z} \in End_K E$. The map $\Phi \in End_K E$ sending (x, y) to (x^q, y^q) and fixing $\mathcal{O}$ is called the *Frobenius endomorphism* of E. In $End_K E$, Φ satisfies the relation

$$\Phi^2 - t\Phi + q = 0$$

for a unique $t \in \mathbf{Z}$, called the trace of the *Frobenius endomorphism*. In fact, $t = q + 1 - \#E(\mathbf{F}_q)$. Recall that if ℓ is an odd prime then $E[\ell] \cong Z_\ell \oplus Z_\ell$. Consequently, $E[\ell]$ can be viewed as a vector space over $\mathbf{F}_\ell$; the vector space has dimension 2. The map Φ restricted to $E[\ell]$ is a linear transformation on $E[\ell]$ with characteristic equation $\Phi^2 - t\Phi + q = 0$.

Proposition A-3: For any positive odd n, the congruence $\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}$ is holding.

Proof: Let f(n) denote the function on the rigt side of the equality, as in the proof of Proposition II.2.4 in [32]. It is easy to see that $f(n_1n_2) = f(n_1)f(n_2)$ for any two odd numbers n_1 and n_2 (Just consider the different possibilities for n_1 and n_2 modulo 8). This means that the right side of the equality in the proposition equals.

$$f(p_1)^{\alpha_1} \cdots f(p_r)^{\alpha_r} = \left(\frac{2}{p_1}\right)^{\alpha_1} \cdots \left(\frac{2}{p_r}\right)^{\alpha_r}$$

But this is (2/n), by definition.

This proof is not tight, some parts of the argument are omitted. The interested reader can refer to reference [32] page 44 for more details.

Appendix B. Optimal Normal Bases

A normal basis for $\mathbf{F}(2^{105})$ over $\mathbf{F}_2$ is a basis of the form

$$N = \{\beta, \beta^2, \beta^{2^2}, ..., \beta^{2^{104}}\}.$$

The basis is optimal if its multiplication table is as "simple" as possible; see [69] for more details and for an easy way to construct such a basis. Given any $\alpha \in F_{2^{105}}$, then one can express $\alpha = \sum_{i=0}^{104} c_i \beta^{2^i}$, where $c_i \in \mathbf{F}_2$, and write $\alpha = (c_0, c_1, c_2, \ldots, c_{104})$. In software, α is represented by a bit vector of length 105, i.e. on a 32-bit machine, α is stored in an array of unsigned integers of length 4, the last 23 bits of which are unused.

Addition of elements is achieved by simply XOR the vector representations. Since

$$\alpha^2 = \sum_{i=0}^{104} c_i \beta^{2^{i+1}} = \sum_{i=0}^{104} c_{i-1} \beta^{2^i}$$

(with indices reduced modulo 105), squaring α is accomplished by a cyclic shift of its vector representation.

The most efficient way to compute the inverse of α is that first convert to a polynomial basis representation of $F_{2^{105}}$ using a precomputed change of basis matrix, to compute the inverse using an efficient implementation of the extended Euclidean algorithm as described in [70]. Then it can transform the result back to the normal basis representation.

Appendix C. Chinese Remainder Theorem

The Chinese Remainder Theorem is useful for purposes such as simplifying modular arithmetic. Suppose $m_1, \ldots, m_r$ are pairwise relatively prime (that is $\gcd(m_i, m_j) = 1$ if $i \neq j$). Suppose $a_1, \ldots, a_r$ are integers, and consider the following system of congruencies:

$$x = a_1 \pmod{m_1}$$
$$x = a_2 \pmod{m_2}$$
$$\vdots$$
$$x = a_r \pmod{m_r}.$$

The Chinese Remainder Theorem asserts that this system has a unique solution modulo

$$M = m_1 \times m_2 \times \ldots \times m_r.$$

This result will be proved in [41], and also describe an efficient algorithm for solving systems of congruencies of this type.

It is convenient to study the function ξ: $\mathbb{Z}_M \rightarrow \mathbb{Z}m_1 \times \ldots \times \mathbb{Z}m_r$, which be defined as follows:

$$\xi(x) = (x \bmod m_1, \ldots, x \bmod m_r).$$

Example: Suppose $r = 2$, $m_1 = 5$ and $m_2 = 3$, so $M = 15$. Then the function ξ has the following values:

$\xi(0) = (0, 0)$	$\xi(1) = (1, 1)$	$\xi(2) = (2, 2)$
$\xi(3) = (3, 0)$	$\xi(4) = (4, 1)$	$\xi(5) = (0, 2)$
$\xi(6) = (1, 0)$	$\xi(7) = (2, 1)$	$\xi(8) = (3, 2)$
$\xi(9) = (4, 0)$	$\xi(10) = (0, 1)$	$\xi(11) = (1, 2)$
$\xi(12) = (2, 0)$	$\xi(13) = (3, 1)$	$\xi(14) = (4, 2).$

Appendix D. Weierstrass Equations

Let $\mathbf{F}_q$ denote the finite field containing q elements, where q is a prime power. Let K is a field, let $\overline{K}$ denote its algebraic closure. The projective plane $P^2(K)$ over K is the set of equivalence classes of the relation "~" acting on $K^3\backslash\{(0, 0, 0)\}$, where $(x_1, y_1, z_1) \sim (x_2, y_2, z_2)$ if and only if there exists $u \in K^*$ such that $x_1 = ux_2$, $y_1 = uy_2$, $z_1 = uz_2$. A *Weierstrass Equation* is a homogeneous equation of degree 3 of the form

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3,$$

where $a_1, a_2, a_3, a_4, a_6 \in \overline{K}$. The *Weierstrass Equation* is called to be smooth or non-singular if for all projective points $P = (x, y, z) \in P^2(\overline{K})$ satisfying

$$F(X, Y, Z) = Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3 = 0,$$

at least one of the three partial derivatives $\partial F/\partial X$, $\partial F/\partial Y$, $\partial F/\partial Z$ is non-zero at P point. If all three partial derivatives vanish at some point P, then P is called a singular point, and the *Weierstrass Equation* is said to be singular.

For convenience, the *Weierstrass Equation* is usually be written for an elliptic curve using affine coordinates on Z plane (by mapping plane), such as let $x = X/Z$ and $y = Y/Z$ instead into equation (D – 1). Therefore, the form is that

$$E: \quad y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6,$$

always remembering that there is the extra point $\mathcal{O} = [0, 1, 0]$ out at infinity. If $a_1, a_2, a_3, a_4, a_6 \in K$, then E is said to be defined over K.

Appendix E. Weil Pairing & Weil Theorem

E-1. Weil Pairing,

Let $K = \mathbf{F}_q$ and let $\overline{K}$ denote its algebraic closure. Let E be an elliptic curve defined over K. If L is any field containing K, then E(L) denotes the set of points on the curve whose coordinates are both in L, and including the point at infinity. We will write E for $E(\overline{K})$.

A divisor D is a formal sum of points in E, $D = \sum_{P \in E} n_P(P)$, where $n_P \in \mathbb{Z}$, and $n_P = 0$ for all but finitely many $P \in E$. The degree of D is the integer $\sum n_P$. The divisors of degree 0 form an additive group, denoted D^0. The support of D is the set $\{P \in E \mid n_P \neq 0\}$.

If E is defined by the equation $r(x, y) = 0$, $r \in K[x, y]$, then the function field K(E) of E over K is the field of fractions of the domain $K[x, y]/I_r$, where I_r denotes the ideal generated by r. Similarly, $\overline{K}(E)$ is the field of fractions of $\overline{K}[x, y]/I_r$.

Let $f \in \overline{K}(E)^*$. For each $P \in E$, define $v_P(f)$ to be $n > 0$ or $-n < 0$ if f has a zero or a pole of order n at P, respectively. One can associate the divisor $\sum v_P(f)(P)$ to f, and denote it by (f), and can verity that $(f) \in D^0$. A divisor $D = \sum n_P(P)$ is said to be principal if D = (f) for some $f \in \overline{K}(E)^*$. One can also verify that D is principal, if and only if $\sum n_P = 0$ and $\sum n_P P = O$.

Let D_l denotes the set of all principal divisors; D_l forms a subgroup of D^0. If $D_1, D_2 \in D^0$, one can write $D_1 \sim D_2$ if $D_1, D_2 \in Dl$. For each $D \in D0$ there exists a unique point $P \in E$ such that $D \sim (P) - (O)$. If $D = \sum n_P(P)$ is a divisor and $f \in \overline{K}(E)^*$ such that D and (f) have disjoint supports, then one defines $f(D) = \Pi_{P \in E} f(P)^{nP}$.

Now let m be an integer co-prime to q and let P, Q ∈ E[m]. Let A, B ∈ D^0 such that A ~ (P) – (O) and B ~ (Q) – (O), and A and B have disjoint supports. Let f_A, $f_B \in \overline{K}(E)$ be such that $(f_A) = mA$ and $(f_B) = mB$. Then the Weil pairing $e_m(P, Q)$ is defined to be

$$e_m(P, Q) = f_A(B)/f_B(A).$$

E-2. Weil Theorem

The Weil theorem says in a much more general context (algebraic varieties of any dimension) that the *zeta function* has a very special form. In the case of an elliptic curve $E/\mathbf{F}_q$ Weil proved it as following.

Weil Theorem (Conjectures) for an elliptic curve:

The Zeta function is a rational function of T having the form

$$Z(T; E/\mathbf{F}_q) = (1 + aT = qT^2) / (1 - T)(1 - qT),$$

where only the integer 'a' depends on the particular elliptic curve E. The value 'a' is related to $N = N_1$ as follows:

$$N = q + 1 - a.$$

In addition, the discriminate of the quadratic polynomial in the numerator is negative (i.e., $a^2 < 4q$, which is Hasse's Theorem), and so the quadratic has two complex conjugate roots α, β both of absolute value $\sqrt{q}$ (more precisely, 1/α and 1/β are the roots, and α, β are the "reciprocal roots").

Appendix F. Baby-step giant-step Algorithm

Assume that a subset $X \subseteq \mathbf{Z}_{p-1}$ is known such that the required solution of $g^x = y \pmod p$ satisfies $s \in X$. Choose 'small' sets $A, B \subset \mathbf{Z}_{p-1}$ such that $X \subseteq A + B$ where the sum of the set is defined by $A + B = \{a + b \pmod{p-1} : a \in A, b \in B\}$. Rewrite $g^x = y \pmod p$ as $g^{a+b} = y \pmod p$ or as $g^a = yg^{-b} \pmod p$. Crete the lists $\{g^a \pmod p\}_{a \in A}$ and $\{yg^{-b} \pmod p\}_{b \in B}$, sort (or hash) them, and find a common member, $g^a = yg^{-b} \pmod p$. The corresponding 'a' and 'b' define the required solution, $x = a + b \pmod{p-1}$.

The method has time complexity O(s log(s)) (or O(s) if hashing is used) and space complexity O(s) where $s = \max\{|A|, |B|\}$. Clearly, $s \geq \sqrt{|X|}$ for any choice of A and B that satisfies $X \subseteq A + B$.

This example describe Pollard,s λ-method for catching kangaroos [71], in which X is some segment within $\mathbf{Z}_{p-1}$, or an arithmetic sequence in $\mathbf{Z}_{p-1}$. Let $n = \lceil \log p \rceil$ denotes the number of bits in p, t is some number between 0 and n, [n] denotes the set $\{0, 1, \ldots, n - 1\}$, and $\|x\|$ denotes the Hamming weight of a number x, that is the number of 1's in the binary representation of X. Also, $X_{=t} = \{x \in \mathbf{Z}_{p-1} : \|x\| = t\}$ denotes the set of logarithms with Hamming weight exactly t, and $X_{\leq t} = \{x \in \mathbf{Z}_{p-1} : \|x\| \ \forall \in \square \leq t\}$ denotes the set of logarithms with Hamming weight at most t.

- Step 1. For $X = X_{=t}$ with $t < n/2$ choose $A = B = X_{=t/2}$.

- Step 2. For $X_{\leq t}$ with $t < n/2$ guesses the Hamming weight of x and solves as above. Alternatively, choose $A = B = X_{\leq t/2}$.

- Step 3. For $X = X_{=t}$ with $t > n/2$ a cosmetic change in the method is convenient: choose $A = X_{x=(n+t)/2}$ and $B = X_{x=(n-t)/2}$, note that $X \subseteq A \backslash B$, and solve $g^{a-b} = y$, that is $g^a = yg^b \pmod p$.

- Step 4. When some subset $I \subseteq [n]$ is known to project every $x \in X$ in the same way. Namely, there are some fixed values $c_i \in \{0, 1\}$ for $i \in I$, such that every $x = \sum_{i=0}^{n-1} x_i 2^i \in X$ satisfies $x_i = c_i$ for all $i \in I$. For this case, one can pick $I_A \subseteq [n]$ and $I_B \subseteq [n]$ of (roughly) the same size, which are disjoint and satisfy $I_A \square I_B = [n] \setminus I$. Then one can choose $A = X \cap \{x: \forall i \in I_A, x_i = 0\}$ and $B = \{x: \forall i \in I \square I_B, x_i = 0\}$.

- Step 5. If X has restricted Hamming weight and in addition is restricted by some subset $I \subset [n]$ with fixed value c_i for $i \in I$ as above, the decomposition is easily obtained by 'merging' the two corresponding decompositions above.

Note that the complexity of Step 4 is exactly the square-root of the size of the structured set, X. The complexity of small Haming weight DL is worse than this, say $|X|^{\beta}$ for some $\beta > 1/2$.

Appendix G. Menezes-Okamoto-Vanstone attack

Let E be an elliptic curve over $\overline{F}_q$, which is the algebraic closure of $\mathbf{F}_q$. $E(\mathbf{F}_q)$ is the set of all points in E with coordinates from $\mathbf{F}_q$. $E(\mathbf{F}_q)$ has finitely many points, whereas E has infinitely many. Define $E[n] = \{P \in E : nP = O\}$. E[n] is called the set of n-torsion points of E. Now for each n, gcd(n, q) = 1, there exists a positive integer k such that $E[n] \subseteq E(F_{q^k})$ and an isomorphism from E[n] to a subgroup of F_{q^k} can be computed using the Weil pairing. A random polynomial time algorithm for computing the Weil pairing has been proved by Miller in [55]. These results form the basis for the Menezes-Okamoto-Vanstone (MOV) attack.

Let $E(\mathbf{F}_q)$ be an elliptic curve over $\mathbf{F}_q$ and let P be a point of order n (i.e., $\#\langle P\rangle = n$). To apply the MOV method if gcd(n, q) = 1, determine the smallest value of k such that $E[n] \subseteq E(F_{q^k})$. Now if R is a point of $E(\mathbf{F}_q)$ whose logarithm with respect to P is to be found, one proceeds as follows. Check that $R \in \langle P\rangle$ so that there will exist some integer s such that R = sP. Determine an element $Q \in E[n]$ such that the Weil pairing of P and Q in F_{q^k} generates the cyclic subgroup isomorphic to E[n]. Finally, determine the logarithm in F_{q^k} of the Weil pairing of Q and R. This logarithm is s. Note that this logarithm can be found by using the index calculus methods for $F_{q^k}^*$. Thus, even though the index calculus methods do not apply directly to $E(\mathbf{F}_q)$, one can map a subgroup of this group into an algebraic structure where the method does apply.

Appendix H. Trace Function

Let Trace function denotes the linear function $Tr : \mathbf{F}_{2m} \rightarrow \mathbf{F}_2$ defined by

$$Tr : \alpha \square \alpha + \alpha^{2^1} + \alpha^{2^2} + \cdots + \alpha^{2^{m-1}}.$$

If m is even, then let Te denotes the Trace function $Te : \mathbf{F}_{2m} \rightarrow \mathbf{F}_4$ defined by

$$Te : \alpha \square \alpha + \alpha^{2^2} + \alpha^{2^4} + \cdots + \alpha^{2^{m-2}}.$$

The elements of $\mathbf{F}_4$ are denoted by 0, 1, c_1, and c_2. Thus the identities are

$$c_1^2 + c_1 + 1 = 0,$$
$$c_2^2 + c_2 + 1 = 0,$$
$$c_1 c_2 = 1,$$
$$c_1 + c_2 = 1.$$

Note that $Te(c_1\alpha) = c_1 Te(\alpha)$, and $Te(c_2\alpha) = c_2 Te(\alpha)$.

The quadratic equation $x^2 + ax + b = 0$ $(a, b \in \mathbf{F}_{2m}, a \neq 0)$, has a solution in $\mathbf{F}_{2m}$ if and only if $Tr(a^{-2}b) = 0$. If x_1 is one solution, then the another solution is $x_1 + a$.

Using the general results in [ecc 98] concerning the number of roots of an affine polynomial over a finite field, one obtains the following results on the number of solutions in $\mathbf{F}_{2m}$ of the quartic equation

$$x^4 + ax + b = 0 \qquad (a, b \in \mathbf{F}_{2m}, a \neq 0). \qquad \text{(H-1)}$$

(1) If m is odd, then (H-1) has either no solution or exactly two solutions.

(2) If m is even and a is not a cube, then (H-1) has exactly one solution.

(3) If m is even and a is a cube, then (H-1) has four solutions if $Te(b/a^{4/3}) = 0$, and no sokution if $Te(b/a^{4/3}) \neq 0$.

Appendix I. Quadratic Residue

Suppose that p is an odd prime, i.e., $p > 2$. It is interested in knowing which of the nonzero elements $\{1, 2, \ldots, p-1\}$ of $\mathbf{F}_p$ are squares. If some $a \in \mathbf{F}_p^*$ is a square, say $b^2 = a$, then a has precisely two square roots $\pm b$ (since the equation $X^2 - a = 0$ has at most two solutions in a field). Thus, the squares in $\mathbf{F}_p^*$ can all be found by computing $b^2 \pmod p$ for $b = 1, 2, 3, \ldots, (p-1)/2$ (since the remaining integers up to $p - 1$ are all $= -b$ for one of these b), and precisely half of the elements in $\mathbf{F}_p^*$ are squares. For example, the squares in $\mathbf{F}_{11}$ are $1^2 = 1$, $2^2 = 4$, $3^2 = 9$, $4^2 = 5$, $5^2 = 3$. The squares in $\mathbf{F}_p$ are called quadratic residues modulo p. The remaining nonzero elements are called nonresidues. For $p = 11$ the nonresidues are 2, 6, 7, 8, 10.There are $(p-1)/2$ residues and $(p-1)/2$ nonresidues.

If g is a generator of $\mathbf{F}_p$, then any element can be written in the form g^j. Thus, the square of any element is of the form g^j with j even. Conversely, any element of the form g^j with j even is the square of some element, namely $\pm g^{j/2}$.

REFERENCE

[1] H.Cohn, Advanced Number Theory. New York: Dover, 1980.

[2] D.Coppersmith, "Fast evaluation of logarithms in fields of characteristic two," IEEE Trans. Inform. Theory, Vol. IT-30, PP. 587--594, July 1984.

[3] J.Dixon, "Asymptotically fast factorisation of integers," Math. Compute., Vol. 36, no. 153, Jan. 1981

[4] A.M.Odlyzko, Talk given at Hewlett-Packard Symposium on Information Security, Royal Holloway, University of London, 19 December 1994.

[5] Alfred J. Menezes, "Elliptic Curve Public Key Cryptosystems", Auburn University, Kluwer Academic Publishers, Dordrecht/London, 1993.

[6] Bernd Kowalski, . Use of smart cards for security applications by Deutsche Telekom. , Deutsche Telekom AG, PZ Telesec Untere Industriestr. 20, 57250 Netphen, Germany, IFIP World conference on Mobile Communications, 2~6 September, 1996, Canberra, Australia.

[7] W. Diffie and M. Hellman, "New directions in cryptography", IEEE Trans. Inform. Theory, vol. IT--22, pp. 472--492, 1976.

[8] H. Ong and C. Schnorr, "Signatures through approximate representations by quadratic forms", to appear.

[9] H. Ong. C. Schnorr and A. Shamir, "An efficient signature scheme based on quadratic forms", in Proc. 16th ACM Symp. Theoretical Computer Science, 1984, pp. 208--216.

[10] R. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM, 21 (1978), 120--126.

[11] N. Koblitz, . Elliptic curve cryptosystems. , Mathematics of computation, 48 (1987), pp 203-209.

[12] V. Miller, . Uses of elliptic curves in cryptography. , Advances in Cryptology . CRYPTO'85, Lecture Notes in Computer Science, 218 (1986), Springer-Verlag, pp 417 . 426.

[13] Peter Bauer and Heribert Peuckert, "Crypto Chipcards: Opening the Door to New Applications", Siemens Review, R & D Special, Spring 1994.

[14] 1995 Europay International S.A., Master Card International Incorporated, and Visa International Service Association. Part 1: "Electromechanical

characteristics, Logical Interface, and Transmission Protocols", ICC Specifications for Payment Systems, Version 2.0, June 30, 1995.

[15] David Naccache and David M'Raihi, "cryptographic Smart cards", IEEE Micro, IEEE computer Society, June 1996.

[16] Louis Claude Guillou, Michel Ugon, and Jean-Jacques Quisquater, "A Standardized Security Device dedicated to Public cryptology", Comtemporary Cryptology, chapter 12, Smart Card: A security Device, section 5, IEEE Press, New York, 1991.

[17] Hitachi Europe Ltd., em IC Card Devices --- Towards 2000, presented at Smart Card 95, London, 1995.

[18] Andreas Fuchsberger, Dieter Gollmann, Paul Lothian, Kenneth G. Paterson, and Abraham Sidiropoulos, . Public Key cryptogarphy on Smart Card. , Author Editor: Dawson Jovan Golic, Cryptography: Policy and Algorithms, International conference,Australia, July 1995.

[19] Jose Luis Zoreda and Jose Manuel Oton, . Smart Cards. , Artech House, Bostyon, London, 1994.

[20] Gert Krings, . Intelligent memory chips for smart cards. , Ics for Smart Cards, Applications, Components, No. 1, 1994.

[21] Don J. Torrieri, "Principles of Secure Communication Systems", Boston, 1992.

[22] K. H. Rosen, Elementary Number Theory and Its Applications, 2nd ed., Reading, MA: Addison Wesley, 1988.

[23] D. Kravitz and I. Reed, "Extension of RSA cryptostructure: A Galois approach", Electuon. Lett., vol. 18, pp. 255-256, 1982.

[24] J. H. Moore, "Protocol failures in cryptosystems", this volume.

[25] Taher ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", IEEE Transactions on Information Theory, Vol., IT-31, No. 4, pp. 469-472, July 1985.

[26] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over GF(p) and its cryptograhpic significance", IEEE Trans. Inform. Theory, vol. IT-24, pp. 106-110, 1978.

[27] L. Adleman, "A subexponential algorithm for the discrete logarithm problem with applications to cryptography", in Proc. 20th IEEE Symp. Foundations of computer Science 1979, pp. 55-60.

[28] R. Schroeppel, H. Orman, S. O'Malley and O. Spatscheck, "Fast key exchange with Elliptic Curve systems", Advances in Cryptology, CRYPTO'95, 15 annual international Cryptology, pp. 43-56.

[29] A. Menezes and S. Vanstone, "The implementation of elliptic curve cryptosystems", Advances in cryptology – AUSCRYPT'90, Lecture Notes in computer Science, 453 (1990), Springer-Verlag, 2-13.

[30] A. Menezes, T. Okamoto and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field", Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing, 80-89,1991.

[31] A. Bender and G. Castagnoli, "On the implementation of elliptic curve cryptosystems", Advances in cryptology – CRYPTO'89, Lecture Notes in Computer Science, 435 (1990), Springer-Verlag,417-426.

[32] Neal Koblitz, "A Course in Number Theory and Cryptography", 2nd Edition, Springer-Verlag, 1994.

[33] Robert D. Silverman, "The Multiple Polynomial Quadratic Sieve", American Mathematical Society, vol. 48, January 1987, pp 329-339.

[34] S. Wagon, "Primality testing", The Math. Intelligencer 8, No. 3 (1986), pp 58-61.

[35] Don Coppersmith, Andrew M. Odlzyko, and Richard Schroeppel, "Discrete Logarithms in GF(p)", Algorithmica, Springer-Verlag New York Inc., vol. 1, pp. 1-15, 1986.

[36] E. R. Canfield, P. Erdos, and C. Pomerance, "On a problem of Oppenheim concerning factorisatio Numerorum", J. Number Theory 17 (1983), pp. 1-28.

[37] D. E. Knuth, "The art of computer programming", vol. 2: Seminumerical algorithms, 2nd deition, Addison-Wesley, Reading, MA, 1981.

[38] Joseph H. Silverman, "The Arithmetic of Elliptic Curve", Springer-Verlag, New York, September, 1985.

[39] Joseph H. Silverman and John Tate, "Rational Points on Elliptic Curve", Springer-Verlag, New York, 1992.

[40] D. Husemoller, "Elliptic Curves", Springer-Verlag, New York, 1987.

[41] Douglas R. Stinson, "Cryptography Theory and Practice", The CRC Press series on *Discrete Mathematics and its Applications*, 1995.

[42] R. Lidl and H. Niederreiter, "Finite Fields", Cambridge University Press, 1987.

[43] J. Omura and J. Massey, "Computatioinal methosand apparatus for finite field arithmetic", U.S. patent number 4,587,627, May 1986.

[44] R. Mullin, I.Onyszchuk, S.Vanstone and R. Wilson, "Optimal normal bases in GF(p^n)",Discrete Applied Mathematics, 22 (1988/89), pp 149-161.

[45] T. Itoh, O. Teechai and S. Tsujii, " A fast algorithm for computing multiplicative inverses in GF(2^t) using normal bases", (in Japanese), J. Society forElectronic Communications (Japan), 44 (1986) pp 31-36.

[46] R. Schoof, "Elliptic curves over finite fields and the computation of square roots mod p", Mathematics of computation, 44 (1985), pp. 483-494.

[47] R. Schoof, "Nonsingular plane cubic curves over finite fields", Journal of Combinatorial Theory, A 46 (1987), pp 183-211.

[48] E. Waterhouse, "Abelian varieties over finite fields", Ann. Sci. Ecole Norm. Sup., 2 (1969), pp 521-560.

[49] M. Ben-Or, "Probabilistic algorithms in finite fields", 22nd Annual Symposium on Foundations of Computer Science, pp394-398, 1981.

[50] L. Charlap and D. Robbins, "An elementary introduction to elliptic curves", CRD Expository Report No. 31, Institute for Defense Analysis, Princeton, December 1988.

[51] T. Beth, W. Geiselmann, and F. Schaefer, "Arithmetics on Elliptic Curves", Algebraic and combinatorical Coding Theory, 2nd int. workshop, Leningrad, 1990, pp. 28-33.

[52] G. Agnew, R. Mullin and S. Vanstone, "An implementation of elliptic curve cryptosystems over F_{2155}, preprint, 1992.

[53] Alfred J. Menezes, Paul C. Van Oorschot and Scott A. Vanstone, "Handbook of Applied Cryptography", Discrete Mathematics and Its Applications, CRC press, 1997, pp. 287.

[54] J. Pollard, "Monte Carlo methods for index computation (mod p)", Math. Computat., vol. 32, pp. 918-924, 1978.

[55] V. Miller, "Short programs for functions on curves", unpublished manuscript, 1986.

[56] A. Menezes and S. Vanstone, "Elliptic curve cryptosystems and their implementation", submitted to J. Cryptol., 1991.

[57] A. Menezes, S. Vanstone and R. Zuccherato, "Counting points on elliptic curves over F_{2m}", to appear in Math. Computat., 1992.

[58] J. A. Gordon, "Strong primes are easy to find", in Lecture Notes in Computer Science 209; Advances in Cryptology: Proc. Eurocrypt'84, T. Beth, N. Cot, and I. Ingemarsson, Eds., Paris,France, April 9-11, 1984, pp. 216-233, Springer-Verlag, 1985.

[59] H. C. Williams and B. Schmid, "Some remarks concerning the MIT public key cryptosystem", BIT, vol. 19, pp. 525-538, 1979.

[60] Gustavus J. Simmons edited, "Comtemporary cryptology: The science of information integrity", chapter 4, Public key cryptography, section 1, Cryptography, IEEE press, 1992.

[61] B. A. LaMacchia and A. M. Odlyzko, "Computation of discrete logarithms in prime fields", *Designs, Codes, and Cryptograsphy*, vol. 1, pp. 46-62, 1991.

[62] Ronald L.Rivest, "RSA Chips (past/Present/Future)", Advances in cryptology: Proceedings of EUROCRYPT'84, Springer-Verlag, 1985.

[63] L. C. K. Hui and K. Y. Lam, "Fast square-and-multiply exponentiation for RSA", Electronics Letters vol. 30 No. 17, August 1994.

[64] H. W. Lenstra, Jr., "Factoring integers with elliptic curves", Annals of Math. (2) 126 (1987), pp. 649-673.

[65] C. Pomerance, "Analysis and comparison of some integer factoring algorithms", in computational Methods in Number Theory, ed.by H. W. Lenstra, Jr. and R. Tijdeman, Mathematisch Centrum, Amsterdam, 1982, pp. 89-139.

[66] Ueli M. Maurer, "Fast Generation of Prime Numbers and Secure Public Key Cryptographic Parameters", Journal of Cryptology, August 1995, pp. 123-155.

[67] Alfred J. Menezes, Scott A. Vanstone, and Robert J. Zuccherato, "counting points on elliptic curves over F_{2m}", Mathematics of Computation vol. 60, No.201, January 1993, pp. 407-420.

[68] Reynald Lercier and Francois Morain, "Counting the number o points on elliptic curves over finite fields: strategies and performances", Advance in cryptology – EUROCRYPT'95, International conference on the Theory and Application of Cryptographic Techniques, Springer-Verlag, 1995.

[69] R.C. Mullin, I.M. Onyszchuk and S.A. Vanstone, . Optimal Normal Bases in GF(p^n). , Discrete Applied Mathematics 22 (1988/89), pp. 149-161.

[70] E. Berlekamp, . Algebraic Coding Theory. , McGraw-Hill, New York, 1968.

[71] J.M. Pollard, . Monte Carlo methods for index computation (mod p). , Mathematics of computation, 32 (143): 918-924, July 1978.

[72] Arto Salomaa, "Public-key cryptography", The Academy of Finland, EATCS Monographs on Theoretical Computer Science, 1990.

[73] Raymond Hill, "A first course in coding theory", University of Salford, Oxford applied mathematics and computing science series, 1986.

[74] A. Odlyzko, "Personal communication", 1986.

[75] Neal Koblitz, "Introduction to Elliptic Curves and Modular Forms", Department of Mathematics, University of Washington, Library of Congress Cataloging-in-Publication Data, Springer-Verlag, New York, 1993.

[76] Ulrich Hamann, Susanne Hirsch, Stephan Ondrusch, "Cryptocards: Opening the Door to an Unprecedented Level of Security", Siemens Review, R&D Special, Spring 1995.

[77] N. Koblitz, "constructing elliptic curve cryptosystems in characteristic 2",Advances in Crytology – CRYPTO'90, Lecture Notes in Computer Science, 537 (1991), Springer-Verlag, pp 156-167.

[78] Carl Pomerance, J. W. Smith, and Randy Tuler, "A pipeline architecture for factoring large integers with the quadratic sieve algorithm", Society for Industrial and Applied Mathematics, vol. 17, 1-3, 1988.