論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	Algorithms for Finite Field Arithmetics and Their Applications to Public-key Cryptosystems
著者(和文)	伊東利哉
Author(English)	Toshiya Itoh
出典(和文)	学位:工学博士, 学位授与機関:東京工業大学, 報告番号:乙第1881号, 授与年月日:1988年12月31日, 学位の種別:論文博士, 審査員:
Citation(English)	Degree:Doctor of Engineering, Conferring organization: , Report number:乙第1881号, Conferred date:1988/12/31, Degree Type:Thesis doctor, Examiner:
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Algorithms for Finite Field Arithmetics

and Their Applications to Public-Key Cryptosystems

Toshiya ITOH

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING,
FACULTY OF ENGINEERING,
TOKYO INSTITUTE OF TECHNOLOGY

1988

Dedicated to my lovely Colleagues

and research activities.

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

TABLE OF CONTENTS (CONTINUED)

ABSTRACT OF THE DISSERTATION

Algorithms for Finite Field Arithmetics and Their applications to Public-Key Cryptosystems

by
Toshiya ITOH
Doctor of Philosophy in Engineering,
Tokyo Institute of Technology, 1988

Professor Shigeo TSUJII, Chair

In this dissertation we study several algorithms, i.e., the configuration of parallel type multiplier of $GF(2^m)$, the fast (sequential and recursive) algorithm for computing multiplicative inverses in $GF(2^m)$, the efficient probabilistic algorithm for solving quadratic equations over $GF(p)$ and $GF(2^m)$ and the efficient algorithm for deciding quadratic residuosity in $GF(p^m)$ (p:odd prime, $m \geqq 2$), and consider their applications to public-key cryptosystems.

In 1981, Massey and Omura have developed a parallel type multiplier of $GF(2^m)$ which uses the normal basis representation. Massey-Omura Multiplier has elegant properties such as regularity and modularity in the structure which are suitable for VLSI implementation. In this dissertation we propose a parallel type multiplier in $GF(2^m)$ which uses the canonical basis representation and has the similar properties with Massey-Omura Multiplier, i.e., regularity and modularity in the structure. To construct a newly developed parallel type multiplier, we define *Equally Spaced Polynomial* and present the necessary and sufficient condition that *Equally Spaced Polynomials* are irreducible.

Two types of fast algorithms, sequential and recursive ones, for computing multiplicative inverses in $GF(2^m)$ are developed. The sequential type fast algorithm for $GF(2^m)$ iterates multiplications in $GF(2^m)$ to compute multiplicative inverses in $GF(2^m)$, and reduces the number of multiplication to $O(\log m)$. The recursive type fast algorithm for $GF(2^m)$ $(m=2^k)$ also iterates multiplications in $GF(2^m)$ and in the subfields, $GF(2^{m/2})$, $GF(2^{m/4})$, ..., $GF(2^4)$ to compute multiplicative inverses in $GF(2^m)$, and reduces the number of multiplications to two multiplications in $GF(2^m)$ and in each subfield of $GF(2^m)$, i.e., $GF(2^{m/2})$, $GF(2^{m/4})$, ..., $GF(2^4)$.

Efficient probabilistic algorithms for solving quadratic equations over $GF(p)$ and $GF(2^m)$ are presented. The efficient probabilistic algorithm for $GF(p)$ is based on the efficient computation of Legendre's symbol and that for $GF(2^m)$ is based on the efficient computation of Trace relative to $GF(2)$.

Furthermore an efficient algorithm for deciding quadratic residuosity in $GF(p^m)$ (p:odd prime, $m \geqq 2$) is developed. This algorithm efficiently maps the decision problems in $GF(p^m)$ to those in $GF(p)$ by matrix manipulation over $GF(p)$ and outputs the result by the computation of Legendre's symbol. The algorithm can be applied to derive an efficient probabilistic algorithm for solving quadratic equations over $GF(p^m)$.

We propose several public-key cryptosystems and study their security and the applications of the above algorithms to them. Furthermore we develop an ID-Based cryptosystem, which is the one of its earliest concrete examples in a strict sense, and consider the security against the conspiracy of some entities.

CHAPTER I :

INTRODUCTION

The theory of finite field is a branch of modern algebra. In the last 40 years the significance and the diverse applications have been found in coding theory [Ber68,MS77,McE87], cryptography [BKS79,Den82,Kra86,IM85,TKIFM86,TKIFM87], digital signal processing [RT75] and digital communication [McE87].

A finite field is a field with finite elements in which it is possible to add, subtract, multiply and divide, except that division by 0 is not allowed. A finite field $GF(p)$, where p is prime, is a finite field with p elements. It is well-known that every finite field with p elements is isomorphic to Z_p, i.e., a set of integers {0,1,2,...,p-1}. Furthermore for a composite number N, Z_N does not construct a finite field, i.e., for composite number N, there does not exists a finite field with N elements. An extension field $GF(p^m)$ can be regarded as a vector space of dimension m over $GF(p)$. Finite field $GF(2^m)$ especially has the diverse applications to error-correcting codes [Ber68,MS77], cryptography [BKS79,IM85,TKIFM86,TKIFM87], etc., because of its tractability by Boolean circuits, e.g., AND-gates and EOR-gates.

The finite field $GF(2^m)$ is a number system containing 2^m elements. The attractiveness and the availability in practical applications stem from the fact that each element can be repre-

sented by m-dimensional vector over $GF(2)$. The practical application to error-correcting codes makes considerable use of the arithmetics in $GF(2^m)$. Both the encoding and decoding procedures for *Reed-Solomon codes* [Ber68,MS77] are carried out by the arithmetics in $GF(2^m)$ and the decoding procedure for *BCH* codes is also performed by the arithmetics in $GF(2^m)$. Further applications of finite field arithmetics have been found in data and communication security, such as encryption and decryption of digital messages [BSK79,IM85,TKIFM86,TKIFM87] and key-distribution between two parties [DH76] or among multiparties.

The arithmetics in $GF(2^m)$ consist of fundamental operations, i.e., addition, subtraction, multiplication and division. Finite field $GF(2^m)$ can be defined by m-dimensional vector space over $GF(2)$, hence additions and subtractions in $GF(2^m)$ can be easily implemented in hardware, i.e., digital circuits, by componentwise addition and subtraction in $GF(2)$, respectively. (This implies that the implementation of addition and subtraction in hardware require small circuit size and running cost.) On the other hand, multiplications and divisions in $GF(2^m)$ are somewhat complicated operations, hence in general the hardware implementation for multiplications and divisions require large circuit size or large computation time.

The multiplications in $GF(2^m)$, in the usual case, are carried out by the shift-register type multiplier [McE87], thus they can be realized in small circuit size but with long delay.

Especially in the case of $GF(2^m)$ of large order, the computation time for multiplications is much larger than that for additions or subtractions, thus the computation time for instruments or algorithms processing the elements in $GF(2^m)$ is mainly dominated by the computation of multiplications. The divisions in $GF(2^m)$, a/b ($a,b \in GF(2^m)$), can be performed by computing the multiplicative inverse b^{-1} and multiplying a and b^{-1}, therefore the divisions in $GF(2^m)$ are reducible to the computation of multiplicative inverses in $GF(2^m)$. *Euclidean Algorithm* [Ber68] is widely known to be the algorithm for computing multiplicative inverses in finite fields. The *Euclidean Algorithm* runs efficiently, however it is not suitable for the hardware implementation. Thus the development of efficient algorithms for computing multiplicative inverses, which are suitable for the hardware implementation, is considerably useful in a practical sense.

Exponentiations in finite fields can be regarded as iterative multiplications in finite fields, however the techniques of an efficient algorithm for exponentiations in finite fields derives an efficient algorithm for powering polynomials in polynomial ring, which are applicable to polynomial factorization over finite fields and the other practical problems. Thus the exponentiations in finite fields must be studied as one of the basic operations, not as the iterative multiplications in finite fields.

The other operations in finite fields such as quadratic

residuosity in $GF(p)$ (or $GF(p^m)$) are of somewhat theoretical interest. However, Rabin has pointed out the close relation between polynomial factorization over $GF(p)$ (or $GF(p^m)$) and quadratic residuosity in $GF(p)$ (or $GF(p^m)$), and has proposed a probabilistic algorithm for factoring polynomials over finite fields [Ra80a]. The probabilistic algorithm can be applied to the decryption of public-key cryptosystems [Ra79,Wil80,Wil85,KIT87, KIT88a,KIT88b], thus quadratic residuosity in finite fields is not only of theoretical interest but also of practical importance.

In this dissertation, we study the several algorithms for finite fields and their applications to public-key cryptosystems. The organization of this dissertation is as follows:

In *CHAPTER II* , we describe the historical perspective for public-key cryptosystems and mention the relation between finite field arithmetics and public-key cryptosystems.

CHAPTER III presents a new configuration of parallel type multiplier in $GF(2^m)$ [IMT87,IT87b]. The proposed parallel type multiplier also has elegant features such as regularity and modularity, which are suitable for VLSI implementation. We define a specialized polynomial, called *Equally Spaced Polynomial (ESP)*, and present the necessary and sufficient condition for the *ESP* to be irreducible.

CHAPTER IV provides two types of fast algorithms, sequential and recursive ones, for computing multiplicative inverses

in $GF(2^m)$. The sequential type fast algorithm for $GF(2^m)$ [IT88a] iterates multiplications in $GF(2^m)$ to compute multiplicative inverses in $GF(2^m)$, and reduces the number of multiplications in $GF(2^m)$ to $O(\log m)$. The recursive type fast algorithm for $GF(2^m)$ $(m=2^k)$ [IT88b,IT88d] also iterates multiplications in $GF(2^m)$ and in the subfields of $GF(2^m)$, $GF(2^{m/2})$, $GF(2^{m/4}),\ldots,GF(2^4)$ to compute multiplicative inverses in $GF(2^m)$, and reduces the number of multiplications to two multiplications in $GF(2^m)$ and in the subfields of $GF)2^m)$, i.e., $GF(2^{m/2})$, $GF(2^{m/4}),\ldots,GF(2^4)$.

In *CHAPTER V*, we propose efficient probabilistic algorithm for solving quadratic equations over $GF(p)$ and $GF(2^m)$ [Ito87,Ito88a]. The efficient probabilistic algorithm for $GF(p)$ is based on the efficient computation of Legendre's symbol [Sch86] and that for $GF(2^m)$ is based on the efficient computation of Trace [McE87] relative to $GF(2)$.

CHAPTER VI develops an efficient algorithm for deciding quadratic residuosity in $GF(p^m)$ (m:odd prime, $m \geqq 2$) [IT88c]. This algorithm efficiently maps the quadratic residuosity problems in $GF(p^m)$ to those in $GF(p)$ by matrix manipulation over $GF(p)$ and outputs the result by the computation of Legendre's symbol.

In *CHAPTER VII*, we present several public-key cryptosystems and consider their security and the applications of the algorithms proposed in *CHAPTER III -VI* to the cryptosystems. *Section* 7.2 analyzes the security of *Knapsack Type Public-Key Cryptosystem* [IKT84,CR84,KIST87]. *Section* 7.3 proposes a public-key

cryptosystem based on the difficulty of solving a system of non-linear equations [TKIFM86,TKIFM87] and considers the security. In *Section* 7.4, we present a public-key cryptosystem [KIT87, KIT88a,KIT88b], for which it is proven that inversion of the ciphertexts is equivalent to factorization of a large composite number. Furthermore *Section* 7.5 proposes an *ID-Based Cryptosystem* [Sha84] and analyzes the security against the conspiracy of some entities. The proposed *ID-Based Cryptosystem* [TIK87,TI88] is supposed to be one of the earliest concrete examples in a strict sense.

CHAPTER Ⅷ finally describes the conclusions and furthermore mentions the perspective of the related works and fields.

CHAPTER II :

MODERN CRYPTOLOGIES AND FINITE FIELD ARITHMETICS

To transmit messages, most of the people usually make use of some media, e.g., *post cards, letters, telephones, electronic mails,* etc. In these media, the messages are not strictly protected against the third parties, i.e., anyone can read the sentences on post cards or in letters and anyone can also hear the conversation on telephone lines by wiretapping. (The security of the transmitted messages essentially depends on *conscience* and *morality* of many people of good sense.) If the messages to be transmitted are of no importance, e.g., *"Hello."*, *"How are you?"*, *"Good morning."*, *"Good-bye."*, etc., the protection of the messages is not so crucial, hence such media are useful and available for those messages. On the other hand, if the messages to be transmitted are of serious importance, e.g., *a large amount of transactions or contracts*, *diplomatic top secrets*, *military top secrets*, *the newest information for stock prices or exchange rate*, etc., the protection of the messages is considerably crucial, because for *diplomats*, *presidents* or *stockbrokers*, those messages are as valuable as their lives and the disclosure of such information might make their countries, their companies or themselves *seriously disadvantageous!* One method for the transmission of such valuable information is that a professional courier has the information with him and carries it from the senders (e.g., *stockbrokers*) to the receivers (e.g., *clients*) accompanying well-disciplined guards. However, such a

method costs much money and time, thus the other methods are required with low costs and high security. For those requirements, *CRYPTOGRAPHIES* are expected to be one of the most powerful countermeasures to protect the valuable messages.

Cryptographies have long history from ancient times. In ancient times, cryptographies are very simple; Each character of the messages are permuted or rewritten by some rules which are held in common between the sender and the receiver. In 1976, *Data Encryption Standard* (denoted DES) [DES77] was standardized by the Department of Commerce, and it has been widely used for the data protection. The structure of DES is essentially the same with the cryptographies in ancient times, i.e., it is composed of the combinations of simple transforms such as permutation or rewriting of input data. On the other hand, in 1976 a completely different and epoch-making concept for the data protection was proposed, which is widely known nowadays to be *PUBLIC-KEY CRYPTOSYSTEM* (denoted *PKC*) [DH76].

SECRET-KEY CRYPTOSYSTEMS (denoted *SKC*), e.g., DES, must share the same information between the sender and the receiver, called *key*, to encrypt the messages and to decrypt the encrypted messages. (The idea and the technique for SKC are *extremely natural* for the data protection.) On the other hand, PKC *surprisingly* need not to share the same information between the sender and the receiver, i.e., the encryption-key is not equal to the decryption-key. In PKC's, the receiver can publishes his

encryption-key keeping his decryption-key secret, and any senders can *safely* transmit their messages to the receiver by the receiver's encryption algorithm even when the receiver's encryption-key is made public. To materialize the above scheme, the existence of the function f, which satisfies the following conditions, is required:

C1. For $^{\forall}x \in$ *dom* f, f(x) is easily computable;

C2. For almost all $y \in$ *rang* f, to compute $f^{-1}(y)$ is practically infeasible;

C3. There exists some information *s*, called *trapdoor information*, such that for $^{\forall}y \in$ *rang* f, $f^{-1}(y;s)$ can be easily computed.

The function satisfying the above conditions C1. and C2. is referred to as *one way function* [DH76], and furthermore the one way function satisfying the condition C3. is called *trapdoor one way function* [DH76]. The first concrete example for PKC, the celebrated *RSA* [RSA78], was found in 1978. The trapdoor one way function of RSA is based on the formidability of factorization of a large composite number. RSA is known to be one of the most secure PKC's proposed so far, and is believed that to find the decryption algorithm of RSA is equivalent to factor a large composite number, however it is not proven yet. The other PKC's [MH78,Ra79,Wil80,CR84,GM84,Wil85,El85a,BBS86,KIT87,KIT88a] have been found so far, and for some of them, it is proven that to find the decryption algorithm is equivalent to solve the intractable problems, e.g., factorization [Ra79,Wil80,WI185,KIT87, KIT88a,KIT88b], quadratic residuosity [BBS86], etc. Hardware

implementation for RSA can be found in [Riv80,Mi83,TO88].

Most of PKC's are constructed over finite fields, thus the encryptions and decryptions are performed by finite field arithmetics, i.e., additions, subtractions, multiplications, divisions, etc. Since PKC's, in general, make use of finite field of large order, e.g., $GF(2^{100})$, the running time of additions or subtractions is much larger than that of multiplications or divisions. Hence for PKC's, the computation time of encryptions and decryptions are mainly dominated by the running time of multiplications and divisions. These facts motivate the studies on finite field arithmetics especially for multiplications and divisions. In the field of error-correcting codes, the multiplications are usually carried out by *shift-register type multipliers* (denoted SRM) [McE87] or *bit-serial type multiplier* (denoted BSM) [Ber82,McE87], and the divisions are performed by table look-up, i.e., access to *Read Only Memory (ROM)*. For finite field of small order, e.g., $GF(2^8)$, SRM and BSM require small circuit size and low running cost, however for finite field of large order, e.g., $GF(2^{100})$, the running cost of multiplications by SRM or BSM considerably increases, *not negligible!* The divisions by table look-up can be implemented with low running cost and comparatively large circuit size for finite field of small order, e.g., $GF(2^8)$. On the other hand, the implementation of divisions by table look-up is practically impossible for finite field of somewhat larger order, e.g., $GF(2^{16})$. Furthermore divisions can be computed by *Euclidean Algorithm*

[Ber68] with small running cost, however the divisions by *Euclidean Algorithm* are not necessarily suitable for hardware implementation. Hence for finite field of large order, the development of the fast (*sequential and recursive*) algorithm for multiplications and divisions is of theoretical and practical use. In addition the design of the (*parallel type*) multipliers and dividers is of considerable importance for high performance cryptographical instruments. The related works for the design of multipliers can be found in [MO81,WW84,Be85,WTSDOR85,BCG86, STP86,IMT87,IT87b,MK88] and those for divisions can be found in [WTSDOR85,IT88a,IT88b,IT88d,MK88].

The further relations between PKC's and finite field arithmetics are given in [Ra79,Wil80,CR84,Wil85,El85a,KIT87, TIK87,KIT88a,KIT88b,TI88]. Especially for the PKC's [Ra79, Wil80,Wil85,KIT87,KIT88a,KIT88b], they are *provable secure* under the assumption that the factorization is hard. Furthermore the PKC's [El85a,TIK87,TI88] and the *public-key distribution system* [DH76] are based on *Discrete Logarithm Problem*, and the related works for Discrete Logarithm Problem can be found in [PH78,Ad79, HR82,BFMV84,EL85b].

CHAPTER III :

CONFIGURATION OF PARALLEL TYPE MULTIPLIERS IN GF(2^m)

In this Chapter, we develop a new configuration of parallel type multipliers in *GF*(2^m). The proposed multiplier has elegant and interesting features as follows:

F1. The multiplier has regularity and modularity in the structure;

F2. The multiplier can be easily implemented only by AND-gates and EOR-gates;

F3. The number of gates, AND-gates and EOR-gates, in the multiplier are proportional to m^2, and the delay time is proportional to *log* m.

Furthermore we define a polynomial over *GF*(2) of special form, *Equally Spaced Polynomial* (denoted *ESP*), and prove a necessary and sufficient condition for ESP's to be irreducible over *GF*(2).

3.1 INTRODUCTION

This Chapter studies only the multipliers in *GF*(2^m). (The main reason why we concentrate on not general finite field *GF*(q^m) but *GF*(2^m) is of practical use, e.g., error-correcting codes, cryptographies, maximum-length sequences [McE87], etc. The extension to general finite fields is *trivial!*)

Multiplications in *GF*(2^m) have been usually carried out by *Shift Register Type Multipliers* (denoted *SRM*). SRM is a multiplier in *GF*(2^m) with an m-bit shift register, AND-gates and EOR-gates. The circuit size of SRM for *GF*(2^m) is proportional to m

and the computation time of SRM for $GF(2^m)$ is proportional to m. In 1982, Berlekamp has developed a new type of multiplier [Ber82], called *Bit-Serial Type Multiplier* (denoted *BSM*), for *Reed-Solomon encoders* by introducing dual bases [MS77,McE87] in $GF(2^m)$. BSM is a multiplier in $GF(2^m)$ with an m-bit shift register, AND-gates and EOR-gates. The circuit size of BSM for $GF(2^m)$ is proportional to m, but somewhat larger than that of SRM. In addition the computation time of BSM for $GF(2^m)$ is proportional to m, but somewhat less than that of SRM. Thus both SRM and BSM for $GF(2^m)$ require $O(m)$ circuit size and $O(m)$ computation time.

In 1981, Massey and Omura, on the other hand, have developed a completely different type of multiplier [MO81, WTSDOR85], called *Massey-Omura Multiplier* (denoted *MOM*), by introducing *normal bases* [MS77] in $GF(2^m)$. MOM can be implemented in both *parallel* and *serial* type configuration. (Both parallel and serial type) MOM are multipliers in $GF(2^m)$ with the following features:

F1. MOM has regularity and modularity in the structure;

F2. MOM is implemented only by AND-gates and EOR-gates;

F3. For $^{\forall} x \in GF(2^m)$, squaring x can be carried out by cyclic shift of vector representation of x.

Furthermore the circuit size of parallel type (serial type) MOM for $GF(2^m)$, in the general case, is $O(m^3)$ ($O(m^2)$), but in the special case $O(m^2)$ ($O(m)$), and the computation time of parallel type (serial type) MOM for $GF(2^m)$ is $O(log\ m)$ ($O(m log\ m)$) in any case. Thus the parallel type MOM for $GF(2^m)$ has high perfor-

mance from a standpoint of the computation time.

One of the main purposes of this Chapter is to develop a new parallel type multiplier in $GF(2^m)$ with regularity and modularity and furthermore with the small circuit size (i.e., $O(m^2)$) and the low computation time (i.e., $O(log\ m)$).

The organization of this Chapter is as follows:

Section 3.2 is the preliminaries for this Chapter and includes two subsections. (Subsection 3.2.1 provides several mathematical definitions and lemmas and subsection 3.2.2 presents a skeleton of *Massey-Omura Multiplier* (denoted *MOM*) and the related lemmas.) Section 3.3 proposes a new configuration of multipliers for $GF(2^m)$ based on canonical bases, called *Canonical Bases Multiplier* (denoted *CBM*). In Section 3.4, we introduce a new concept, *Equally Spaced Polynomial* (denoted *ESP*), and give a necessary and sufficient condition for ESP's to be irreducible over $GF(2)$. Section 3.4, in addition, presents a new parallel type multiplier in $GF(2^m)$ applying ESP. Section 3.5 finally summarizes the results in this Chapter and gives conclusions, some remarks and open problems.

3.2 PRELIMINARIES

In this Section, we give some mathematical definitions and lemmas for the subsequent discussions. Furthermore we show the configuration of (parallel and serial type) Massey-Omura Multiplier and give some lemmas for the multipliers.

3.2.1 MATHEMATICAL PRELIMINARIES

This subsection gives some mathematical definitions and and lemmas for the subsequent discussions.

DEFINITION 3.1 [MS77] *A basis in $GF(2^m)$ over $GF(2)$ of the form, $\{a, a^2, a^{2\hat{}2}, \cdots, a^{2\hat{}(m-1)}\}$, is called a **normal basis** in $GF(2^m)$ over $GF(2)$.* □

Remark 3.2 The above notation, 2^2 and 2^(m-1), implies 2^2 and 2^{m-1}, respectively. More generally, a^b implies a^b, and the notation will be used in the sequel. □

DEFINITION 3.3 [WW84] *A polynomial $p(x)$ over $GF(2)$ of the form, $p(x)=x^m+x^{m-1}+\cdots+x+1$, is called **All One Polynomial** (denoted AOP) of degree m.* □

Remark 3.4 Throughout this Chapter, we restrict AOP's only over $GF(2)$. □

LEMMA 3.5 [WW84] *The following two statements,*

S1. An AOP of degree m is irreducible over $GF(2)$;

S2. $(m+1)$ is a prime and 2 is the generator of $GF^*(m+1)$, where $GF^*(m+1)$ is the multiplicative group in $GF(m+1)$,

are equivalent. □

The following table illustrates the examples of m's satisfying S2. in Lemma 3.5.

TABLE 3.6 Example of m's

2	28	66	138
4	36	82	148
10	52	100	162
12	58	106	172
18	60	130	178

Irreducible AOP's are irreducible polynomials of specified form, hence they have an interesting property as in the following lemma:

LEMMA 3.7 [WW84] *Let $p(x)$ be an irreducible AOP of degree m. Then all the roots of $p(x)=0$, $\{a, a^2, a^{2^2}, \cdots, a^{2^{(m-1)}}\}$, where $p(a^{2^i})=0$ $(0 \leqq i \leqq m-1)$, are linearly independent over $GF(2)$.* □

The above Lemma (LEMMA 3.7) implies that the set of the roots of p(x)=0 constructs a normal basis [MS77] over $GF(2)$, where p(x) is an irreducible AOP.

3.2.2 MASSEY-OMURA MULTIPLIER

This subsection presents concisely the configuration of (parallel and serial type) *Massey-Omura Multiplier* (denoted *MOM*) and gives some properties of MOM, e.g., the number of gates in MOM and its lower bound, etc.

Assume that $\{a, a^2, a^{2^2}, \cdots, a^{2^{(m-1)}}\}$ is a normal basis in $GF(2^m)$. Then $\forall x, y \in GF(2^m)$ can be represented by the *normal basis* as follows:

$$x = x_0 a + x_1 a^2 + x_2 a^{2^2} + \cdots + x_{m-1} a^{2^{(m-1)}}, \quad x_i \in GF(2) \ (0 \leqq i \leqq m-1),$$

$$y = y_0 a + y_1 a^2 + y_2 a^{2^2} + \cdots + y_{m-1} a^{2^{(m-1)}}, \quad y_j \in GF(2) \ (0 \leqq j \leqq m-1).$$

The product of x and y, i.e., z=xy, can be also represented by the normal basis such that

$$z = xy = z_0 a + z_1 a^2 + z_2 a^{2^2} + \cdots + z_{m-1} a^{2^{(m-1)}}, \quad z_k \in GF(2) \ (0 \leqq k \leqq m-1).$$

Then z_k $(0 \leqq k \leqq m-1)$ is given by the Boolean function f of x_i's

and y_j's ($0 \leqq i,j \leqq m-1$) [WTSDOR85] as follows:

$$z_{m-1}=f(x_0,x_1,\cdots,x_{m-1};y_0,y_1,\cdots,y_{m-1}),$$

$$z_{m-2}=f(x_{m-1},x_0,x_1,\cdots,x_{m-2};y_{m-1},y_0,y_1,\cdots,y_{m-2}),$$

$$\vdots$$

$$z_0=f(x_1,x_2,\cdots,x_{m-1},x_0;y_1,y_2,\cdots,y_{m-1},y_0).$$

The following figures, FIGURE 3.8 and 3.9, show the *parallel* and *serial* type configuration of MOM.

Here each z_k ($0 \leqq k \leqq m-1$) can be implemented only by AND-gates and EOR-gates, hence the number of gates of MOM depends on the number of terms in $f(\cdots;\cdots)$ defined above. The following lemma gives the non-trivial lower bound for the number of terms in $f(\cdots;\cdots)$.

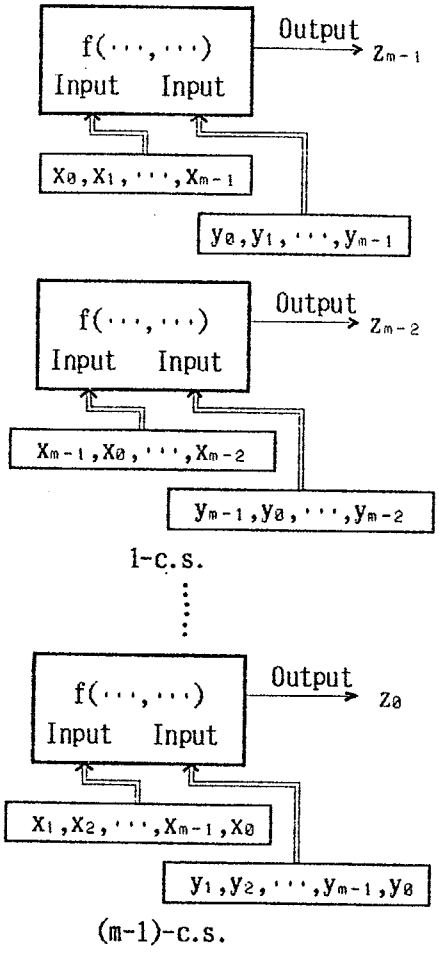
LEMMA 3.10 [Va87] *Let $f(\cdots;\cdots)$ be a Boolean function defined above. Then the number of terms $TERM_{MOM}$ in $f(\cdots;\cdots)$ satisfies $2m-1 \leqq TERM_{MOM} \leqq m^2$.* □

Proof (Sketch): Let $\{a, a^2, a^{2^2}, \cdots, a^{2^{(m-1)}}\}$ be a normal basis in $GF(2^m)$. Here define $a'=Ba$, where $a'=\{a_{mi+j}\}^T$ ($a_{mi+j+1}=a^{(2^i)+(2^j)}$ ($0 \leqq i \leqq m-1$, $0 \leqq j \leqq m-1$)), B is an $m^2 \times m$ matrix over $GF(2)$ and $a=\{a, a^2, a^{2^2}, \cdots, a^{2^{(m-1)}}\}^T$. It is clear that each column vector in B has the same *Hamming weight* s and the Hamming weight s is equal to $TERM_{MOM}(m)$. Here the weight of the matrix B is definitely ms.

Let b_k ($1 \leqq k \leqq m^2$) be row vectors in the matrix B. For $a^{(2^i)+(2^j)}=b_{mi+j+1} \cdot a$ ($0 \leqq i \leqq m-1$, $0 \leqq j \leqq m-1$),

$$\{a^{(2^i)+(2^j)}\}^2=a^{\{2^{(i+1)}\}+\{2^{(j+1)}\}}=b_{m(i+1)+j+2} \cdot a,$$

thus b_{mi+j+1} and $b_{m(i+1)+j+2}$ have the same weight. Define the



k-c.s. : k cyclic shifts

FIGURE 3.8 Parallel type Configuration of MOM in $GF(2^m)$

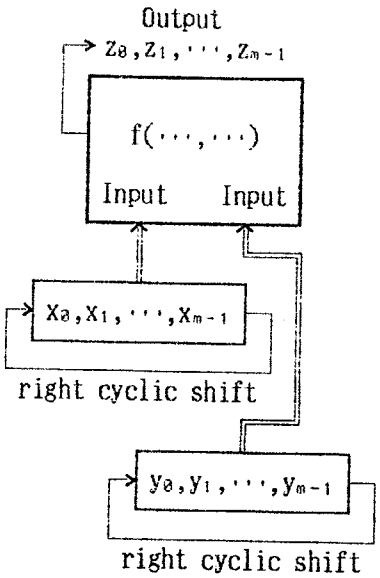


FIGURE 3.9 Serial Type Configuration of MOM in $GF(2^m)$

matrix C, $\mathbf{a}''=C\mathbf{a}$, where $\mathbf{a}''=\{a_i\}^T$ ($a_i=a^{1+\{2\hat{}(i-1)\}}$ ($1\leqq i\leqq m$)), and let the weight of the matrix C, $w(C)$, be t. Then $mt=ms(=w(B))$, so we have $t=s$. This implies that $w(C)$ is equal to $\mathrm{TERM}_{MOM}(m)$. Here $\{a, a^2, a^{2\hat{}2}, \cdots, a^{2\hat{}(m-1)}\}$ constructs a normal basis in $GF(2^m)$, thus $\{a^{1+1}, a^{1+2}, a^{1+2\hat{}2}, \cdots, a^{1+2\hat{}(m-1)}\}$ also constructs a basis in $GF(2^m)$. Hence the matrix C is *invertible* over $GF(2)$. Multiplying C^{-1} to both hand sides of $\mathbf{a}''=C\mathbf{a}$, then $C^{-1}\mathbf{a}''=\mathbf{a}$. Let the first row of C^{-1} be $\mathbf{d}=(d_1, d_2, \cdots, d_m)$, then we have $\mathbf{d}\cdot\mathbf{a}''=a$. Multiplying a^{-1} to both hand sides of $\mathbf{d}\cdot\mathbf{a}''=a$, then $\mathbf{d}\cdot\mathbf{a}=1$ is derived. Eventually $d_i=1$ ($1\leqq i\leqq m$) by the property of normal bases. Here assume that $\mathbf{c}_i$ ($1\leqq i\leqq m$) are column vectors of C. Noting that $C^{-1}C=1$ and $d_i=1$ ($1\leqq i\leqq m$), we have

$$(1,1,\cdots,1)\mathbf{c}_1=1,$$
$$(1,1,\cdots,1)\mathbf{c}_2=0,$$
$$\vdots$$
$$(1,1,\cdots,1)\mathbf{c}_m=0.$$

The above equations imply that $w(\mathbf{c}_1)$ is odd and $w(\mathbf{c}_i)$ ($2\leqq i\leqq m$) are even. Recalling that C is invertible, then we have $w(\mathbf{c}_1)\geqq 1$ and $w(\mathbf{c}_i)\geqq 2$ ($2\leqq i\leqq m$). Hence

$$w(C)=\sum_{i=1}^{m} w(\mathbf{c}_i)\geqq 1+2(m-1)=2m-1.$$

Thus we have $\mathrm{TERM}_{MOM}(m)=w(C)\geqq 2m-1$.

The upper bound for TERM_{MOM} is *trivial*. □

Remark 3.11 Lemma 3.10 was proven by S.A. Vanstone of the University of Waterloo. He explained it in *Special Lecture at NTT Basic Research Lab.* in May 1987. The above proof is due to author, however, the priority and the

originality of the proof, needless to say, belong to S.A. Vanstone. □

For the irreducible AOP p(x) of degree m, the set of roots of p(x)=0 constructs a normal basis (Lemma 3.7). Here we have the following lemma:

LEMMA 3.12 [WW84] *Let p(x) be an irreducible AOP of degree m. Then for the normal basis given by the set of roots of p(x)=0, TERM$_{MOM}$(m)=2m-1.* □

Lemma 3.10 suggests that TERM$_{MOM}$(m) is lowerly bounded by (2m-1) as long as the multipliers in $GF(2^m)$ are constructed by normal bases. Lemma 3.12, on the other hand, shows that there exist normal bases which satisfy the lower bound for TERM$_{MOM}$(m). In the next Section, we will develop an new configuration of parallel type multipliers in $GF(2^m)$, for which the circuit size exceeds the lower bound for MOM, by introducing canonical bases.

3.3 PARALLEL TYPE MULTIPLIERS IN $GF(2^m)$ BASED ON CANONICAL BASES

In this Section we propose a parallel type multiplier in $GF(2^m)$ based on canonical bases, called *Canonical Bases Multiplier* (denoted *CBM*). CBM has the following features:

F1. CBM has regularity and modularity in the structure;

F2. The number of gates in CBM is less than that in parallel type MOM;

F3. For $\forall x \in GF(2^m)$, x^2 can be computed by permutation of (*extended*) vector representation of x.

3.3.1 CONFIGURATION OF CBM IN $GF(2^m)$

The following simple theorems play an important role in the configuration of parallel type CBM.

THEOREM 3.13 *Let $p(x)$ be an AOP of degree m. Then $p(x)$ divides $x^{m+1}+1$, i.e., $p(x) \mid x^{m+1}+1$.* □

THEOREM 3.14 *Let $D(x)$ and $d(x)$ be polynomials over $GF(2)$ of degree m and $(m-1)$, respectively.*

$$D(x)=D_0+D_1x+D_2x^2+\cdots+D_mx^m,$$

$$d(x)=d_0+d_1x+d_2x^2+\cdots+d_{m-1}x^{m-1}.$$

Assume that $D(x)$ and $d(x)$ have the relation such that

$$D(x)=d(x) \pmod{p(x)},$$

where $p(x)$ is an AOP of degree m. Then the coefficients of $d(x)$ are given by those of $D(x)$, i.e., $d_i=D_i+D_m \pmod 2$ $(0 \leqq i \leqq m-1)$. □

The following example illustrates the parallel type configuration of CBM in $GF(2^m)$.

EXAMPLE 3.15:

Assume that m=4, then 5(=m+1) is a prime and 2 is the generator in GF*(5), i.e., m(=4) satisfies S2. in Lemma 3.5. Hence $p(x)=x^4+x^3+x^2+x+1$ is irreducible. For $\forall a,b \in GF(2^4)$,

$$a=a_0+a_1x+a_2x^2+a_3x^3,\ a_i \in GF(2)\ (0 \leqq i \leqq 3),$$

$$b=b_0+b_1x+b_2x^2+b_3x^3,\ b_j \in GF(2)\ (0 \leqq j \leqq 3),$$

where $\{1,x,x^2,x^3\}$ is a canonical basis, the product of a and b, i.e., d=ab, is given by

$$d=d_0+d_1x+d_2x^2+d_3x^3,\ d_k \in GF(2)\ (0 \leqq k \leqq 3).$$

Here define A and B such that

$$A=(A_0,A_1,A_2,A_3,A_4)=(a_0,a_1,a_2,a_3,0),$$

$$B=(B_0,B_1,B_2,B_3,B_4)=(b_0,b_1,b_2,b_3,0).$$

(We call A an *extended* vector representation of a.) Furthermore define D (=AB (mod x^5+1)) such that

$$\begin{aligned}D&=AB \pmod{x^5+1}\\&=(A_0+A_1x+A_2x^2+A_3x^3+A_4x^4)(B_0+B_1x+B_2x^2+B_3x^3+B_4x^4)\pmod{x^5+1}\\&=D_0+D_1x+D_2x^2+D_3x^3+D_4x^4,\end{aligned}$$

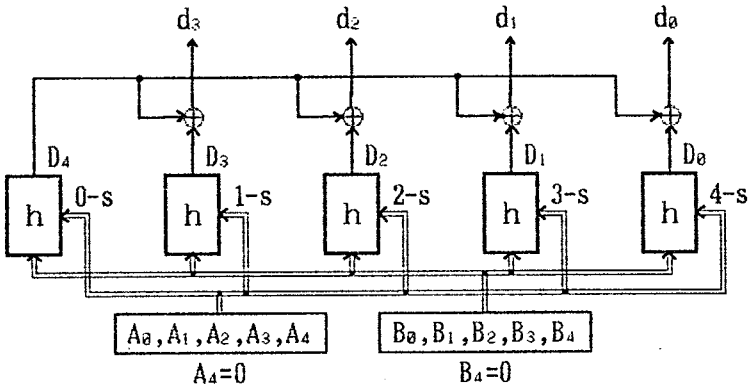
where $A=A_0+A_1x+A_2x^2+A_3x^3+A_4x^4$ and $B=B_0+B_1x+B_2x^2+B_3x^3+B_4x^4$. Here D_i $(0\leqq i\leqq 4)$ are given by $D_i=\sum_{j+k=i\,(\mathrm{mod}\ 5)} A_jB_k \pmod 2$, thus

$$\begin{aligned}D_4&=h(A_0,A_1,A_2,A_3,A_4;B_0,B_1,B_2,B_3,B_4)\\&=A_4B_0+A_3B_1+A_2B_2+A_1B_3+A_0B_4,\\D_3&=h(A_4,A_0,A_1,A_2,A_3;B_0,B_1,B_2,B_3,B_4)\\&=A_3B_0+A_2B_1+A_1B_2+A_0B_3+A_4B_4,\\D_2&=h(A_3,A_4,A_0,A_1,A_2;B_0,B_1,B_2,B_3,B_4)\\&=A_2B_0+A_1B_1+A_0B_2+A_4B_3+A_3B_4,\\D_1&=h(A_2,A_3,A_4,A_0,A_1;B_0,B_1,B_2,B_3,B_4)\\&=A_1B_0+A_0B_1+A_4B_2+A_3B_3+A_2B_4,\\D_0&=h(A_1,A_2,A_3,A_4,A_0;B_0,B_1,B_2,B_3,B_4)\\&=A_0B_0+A_4B_1+A_3B_2+A_2B_3+A_1B_4.\end{aligned}$$

Recalling Theorem 3.13, d(=ab (mod p(x))) and D have the relation such that d=D (mod p(x)), where $p(x)=x^4+x^3+x^2+x+1$. Furthermore noting Theorem 3.14, $d_k=D_k+D_4 \pmod 2$ $(0\leqq k\leqq 3)$, and thus we have the configuration of CBM in $GF(2^4)$ in FIGURE 3.16.

In this example, squaring A (an extended vector representation of a) can be carried out by

$$\begin{aligned}A^2&=(A_0+A_1x+A_2x^2+A_3x^3+A_4x^4)^2 \pmod{x^5+1}\\&=A_0+A_1x^2+A_2x^4+A_3x^6+A_4x^8 \pmod{x^5+1}\\&=A_0+A_3x+A_1x^2+A_4x^3+A_2x^4.\end{aligned}$$



k-s : k cyclic shifts

FIGURE 3.16 Configuration of CBM in $GF(2^4)$

Thus squaring A is given by the following permutation:

$$(A_0, A_1, A_2, A_3, A_4)^2 \rightarrow (A_0, A_3, A_1, A_4, A_2),$$

and this can be easily implemented by hardwiring. □

3.3.2 NUMBER OF GATES IN MOM AND CBM

The configuration of CBM in $GF(2^m)$ requires an irreducible AOP of degree m, hence m must satisfy S2. in Lemma 3.5. On the other hand, m satisfying S2. in Lemma 3.5 realizes the lower bound for $TERM_{MOM}(m)$. In this subsection, we compare the number of gates (AND-gates and EOR-gates) of CBM with those of parallel type MOM for m satisfying S2. in Lemma 3.5.

The function $f(\cdots;\cdots)$ (See subsection 3.2.2.) of MOM in $GF(2^m)$ can be materialized by m AND-gates and (2m-2) EOR-gates for m satisfying S2. in Lemma 3.5 [WW84]. Hence the total number of AND-gates $AND_{MOM}(m)$ and that of EOR-gates $EOR_{MOM}(m)$ in parallel type configuration of MOM are:

$$AND_{MOM}(m)=m^2, \tag{3.1}$$

$$EOR_{MOM}(m)=2m^2-2m. \tag{3.2}$$

On the other hand, the function $h(\cdots;\cdots)$ (See Example 3.15.) of CBM in $GF(2^m)$ can be materialized by (m+1) AND-gates and m EOR-gates for m satisfying S2. in Lemma 3.5. Observing ***EXAMPLE 3.15***, the total number of AND-gates $AND_{CBM}(m)$ and that of EOR-gates $EOR_{CBM}(m)$ in parallel type configuration are:

$$AND_{CBM}(m)=(m+1)^2=m^2+2m+1, \tag{3.3}$$

$$EOR_{CBM}(m)=m(m+1)+m=m^2+2m. \tag{3.4}$$

Thus we can conclude that the total number of AND-gates of CBM is almost the same with that of MOM, (See Eq.(3.1) and (3.3).)

and the total number of EOR-gates of CBM is about half of that of parallel type MOM. (See Eq.(3.2) and (3.4).)

3.4. PARALLEL TYPE MULTIPLIERS IN $GF(2^m)$ BASED ON EQUALLY SPACED POLYNOMIAL

In Section 3.3, we presented a new configuration of parallel type multipliers in $GF(2^m)$ with small circuit size. This Section studies the further extension of the multipliers in Section 3.3, and develops parallel type multipliers in $GF(2^m)$ based on *Equally Spaced Polynomial*.

3.4.1 EQUALLY SPACED POLYNOMIAL

In this subsection, we define polynomials of special form, *Equally Spaced Polynomial* (denoted *ESP*), and give a necessary and sufficient condition for ESP's to be irreducible.

DEFINITION 3.17 [IMT87,IT87b] *A polynomial $g(x)$ over $GF(2)$ of the form, $g(x)=x^{sn}+x^{s(n-1)}+x^{s(n-2)}+\cdots+x^s+1=p(x^s)$, where $p(x)$ is an AOP of degree n, is called* ***s-Equally Spaced Polynomial (denoted s-ESP)*** *of degree sn.* □

Remark 3.18 Throughout this Chapter, we restrict ESP's only over $GF(2)$. □

Here we show the following theorems to derive a necessary and sufficient condition for ESP's to be irreducible.

THEOREM 3.19 [IMT87,IT87b] *Let p is a prime number and let k be a positive integer. If $x=y \pmod{p^k}$ $(k \geq 1)$, then $x^p=y^p \pmod{p^{k+1}}$.* □

Proof: Since $x=y \pmod{p^k}$, x can be represented by some in-

teger n as $x=y+np^k$. Multiplying both hand sides of $x=y+np^k$ by p times, then we have the following equation:

$$x^p = y^p + {}_pC_1\, y^{p-1} np^k + {}_pC_2\, y^{p-2} n^2 p^{2k} + \cdots + n^p p^{kp}$$

$$= y^p + y^{p-1} np^{k+1} + {}_pC_2\, y^{p-2} n^2 p^{2k} + \cdots + n^p p^{kp}$$

$$= y^p + Ap^{k+1},$$

where $A = y^{p-1} n + {}_pC_2\, y^{p-2} n^2 p^{k-1} + \cdots + n^p p^{(k-1)p-1}$. The above equation implies that $x^p = y^p \pmod{p^{k+1}}$. □

THEOREM 3.20 [IMT87,IT87b] *If an AOP of degree m is irreducible and $2^{m(m+1)^{(k-2)}} \neq 1 \pmod{(m+1)^k}$, then we have $ord(2;(m+1)^k) = m(m+1)^{k-1}$, where $ord(a;n)$ denotes the least positive integer i satisfying $a^i = 1 \pmod{n}$.* □

Proof: Let an AOP of degree m be irreducible, then (m+1) is a prime and 2 is the generator in $GF^*(m+1)$. (See Lemma 3.5.) By Euler's Theorem [Sch86], we have $2^{m(m+1)^{(k-1)}} = 1 \pmod{(m+1)^k}$. Here we define S, the set of divisors of $m(m+1)^{k-1}$ except $m(m+1)^{k-1}$ itself, by $S=\{d : d \mid m(m+1)^{k-1}$, where $d < m(m+1)^{k-1}\}$. The set S can be partitioned into two disjoint sets S1 and S2:

$S1=\{d \mid d=m(m+1)^i \ (0 \leqq i \leqq k-2)\}$,

$S2=\{d \mid d=s(m+1)^j \ (0 \leqq j \leqq k-1)$, where $s \mid m \ (s<m)\}$,

because (m+1) is a prime number. In order to show that $ord(2;(m+1)^k) = m(m+1)^{k-1}$, it is sufficient to show that every d in S1 satisfies $2^d \neq 1 \pmod{(m+1)^k}$ and every d in S2 satisfies $2^d \neq 1 \pmod{(m+1)^k}$.

We prove that every d in S1 satisfies $2^d \neq 1 \pmod{(m+1)^k}$ by contradiction. Suppose that $2^d = 1 \pmod{(m+1)^k}$ for some d in S1. This implies that there exists some i $(0 \leqq i \leqq k-2)$ such that $2^{m(m+1)^i} = 1 \pmod{(m+1)^k}$. Multiplying both hand sides of

$2^{m(m+1)^i}=1 \pmod{(m+1)^k}$ by $(m+1)^{k-2-i}$ times, then we have

$$\{2^{m(m+1)^i}\}^{(m+1)^{(k-2-i)}}=2^{m(m+1)^{(k-2)}} \pmod{(m+1)^k}$$
$$=1^{(m+1)^{(k-2-i)}} \pmod{(m+1)^k}$$
$$=1 \pmod{(m+1)^k}.$$

This contradicts that $2^{m(m+1)^{(k-2)}} \neq 1 \pmod{(m+1)^k}$, hence every d in S1 must satisfy $2^d \neq 1 \pmod{(m+1)^k}$.

In the similar way we prove that every d in S2 satisfies $2^d \neq 1 \pmod{(m+1)^k}$ by contradiction. Assume that for some d in S2, $2^d=1 \pmod{(m+1)^k}$. This implies that for some j ($0 \leq j \leq k-1$) and s ($s \mid m$, $s<m$), $2^{s(m+1)^j}=1 \pmod{(m+1)^k}$. By this congruence and Fermat's Theorem [Sch86], we have

$$2^{s(m+1)^j}=2^s \pmod{(m+1)}$$
$$=1 \pmod{(m+1)}.$$

The above congruence contradicts that 2 is the generator in $GF^*(m+1)$, hence every d in S2 must satisfy $2^d \neq 1 \pmod{(m+1)^k}$. Thus $\mathrm{ord}(2;(m+1)^k)=m(m+1)^{k-1}$. □

THEOREM 3.21 [IMT87,IT87b] *Let $g(x)$ be a $(m+1)^{k-1}$-ESP of degree $m(m+1)^{k-1}$. Then $g(x)$ is irreducible if and only if an AOP of degree m is irreducible and $2^{m(m+1)^{(k-2)}} \neq 1 \pmod{(m+1)^k}$.* □

Proof:

Proof of if-part: Consider the following identity:

$$x^{(m+1)^k}+1=(x^{(m+1)^{(k-1)}}+1)(x^{m(m+1)^{(k-1)}}+$$
$$x^{(m-1)(m+1)^{(k-1)}}+\cdots+x^{(m+1)^{(k-1)}}+1)$$
$$=(x^{(m+1)^{(k-1)}}+1)g(x), \qquad (3.5)$$

where $g(x)$ is a $(m+1)^{k-1}$-ESP of degree $m(m+1)^{k-1}$. Assume that u is one of the roots in $x^{(m+1)^k}+1=0$ and satisfies $\mathrm{ord}(u)=(m+1)^k$,

where ord(a) denotes the least positive integer i satisfying $a^i=1$. It is clear from the above identity, Eq.(3.5), that u satisfies g(u)=0 and every conjugate root of g(x)=0 is given by $u, u^2, u^{2^2}, \cdots, u^{2^{\{m(m+1)^{(k-1)}-1\}}}$. Here the assumption, an AOP of degree m is irreducible and $2^{m(m+1)^{(k-2)}} \neq 1 \pmod{(m+1)^k}$, gives $ord(2;(m+1)^k)=m(m+1)^{k-1}$. (See Theorem 3.20.) Recalling the facts that $ord(u)=(m+1)^k$ and $ord(2;(m+1)^k)=m(m+1)^{k-1}$, we can conclude that every conjugate root of g(x)=0 is distinct from each other. Hence g(x) is an irreducible $(m+1)^k$-ESP of degree $m(m+1)^{k-1}$.

Proof of only if-part: *By contradiction.*

Assume that g(x) is an irreducible $(m+1)^k$-ESP of degree $m(m+1)^{k-1}$, then we have $g(x)=p(x^{(m+1)^k})$, where p(x) is an AOP of degree m. Furthermore assume that p(x) is reducible, i.e., $p(x)=\{q_1(x)\}^{e_1}\{q_2(x)\}^{e_2}\cdots\{q_t(x)\}^{e_t}$, where $q_i(x)$ $(1 \leqq i \leqq t)$ are irreducible and $e_j \geqq 1$ $(1 \leqq j \leqq t)$. Then we have

$$g(x)=p(x^{(m+1)^k})$$
$$=\{q_1(x^{(m+1)^k})\}^{e_1}\{q_2(x^{(m+1)^k})\}^{e_2}\cdots\{q_t(x^{(m+1)^k})\}^{e_t}.$$

The above equation contradicts the assumption that g(x) is irreducible, hence p(x), an AOP of degree m, must be irreducible.

Noting Eq.(3.5), we have $ord(u)=(m+1)^k$, where u is a root of g(x)=0. Assume that $2^{m(m+1)^{(k-2)}}=1 \pmod{(m+1)^k}$, then for the conjugate roots of g(x)=0, we have $u^{2^{\{m(m+1)^{(k-2)}\}}}=u$.

This equation contradicts the irreducibility of g(x), hence $2^{m(m+1)^{(k-2)}} \neq 1 \pmod{(m+1)^k}$. □

COROLLARY 3.22 [IMT87,IT87b] *Define the ESP's such that* $g_i(x)=p(x^{(m+1)^i})$ $(0 \leqq i \leqq r-1)$, *where p(x) is an AOP of*

degree m. Then every $g_i(x)$ $(0 \leqq i \leqq r-1)$ is irreducible if and only if an AOP of degree m is irreducible and $2^{m(m+1)^{(r-2)}} \neq 1 \pmod{(m+1)^r}$. □

Proof:

Proof of if-part: If an AOP of degree m is irreducible, then $g_0(x)(=p(x))$ is clearly irreducible. Thus we concentrate on the case for $1 \leqq i \leqq r-1$. By the assumption that an AOP of degree m is irreducible, we have (m+1) is a prime. (See Lemma 3.5.) By Theorem 3.19, the assumption, $2^{m(m+1)^{(k-2)}} \neq 1 \pmod{(m+1)^r}$, gives $2^{m(m+1)^{(k-3)}} \neq 1 \pmod{(m+1)^{r-1}}$. In the similar way, we have $2^{m(m+1)^{(j-1)}} \neq 1 \pmod{(m+1)^{j+1}}$ $(1 \leqq j \leqq r-1)$ are recursively derived. By this equation and Theorem 3.21, we can conclude that every $g_i(x)$ $(0 \leqq i \leqq r-1)$ is irreducible.

Proof of only if-part: Every $g_i(x)$ $(0 \leqq i \leqq r-1)$ is irreducible, hence $g_{r-1}(x)$ is an irreducible $(m+1)^{r-1}$-ESP of degree $m(m+1)^{r-1}$. This result gives that an AOP of degree m is irreducible and $2^{m(m+1)^{(r-2)}} \neq 1 \pmod{(m+1)^r}$. (See Theorem 3.21). This completes the proof. □

EXAMPLE OF COROLLARY 3.22:

Let p(x) be an AOP of degree 2, i.e., $p(x)=x^2+x+1$. Here we can easily verify that $2^{2(2+1)^4} \neq 1 \pmod{(2+1)^6}$, then we have irreducible ESP'S, $g_i(x)$ $(0 \leqq i \leqq 5)$, as follows:

$g_0(x)=p(x^{(2+1)^0})=x^2+x+1$,

$g_1(x)=p(x^{(2+1)^1})=x^6+x^3+1$,

$g_2(x)=p(x^{(2+1)^2})=x^{18}+x^9+1$,

$g_3(x)=p(x^{(2+1)^3})=x^{54}+x^{27}+1$,

$g_4(x)=p(x^{(2+1)^4})=x^{162}+x^{81}+1$,

$g_5(x)=p(x^{(2+1)^5})=x^{486}+x^{243}+1$.

The irreducibility of the above ESP's can be confirmed by the list of irreducible trinomials [Zie68]. □

In the next Section, we will show the configuration of parallel type multipliers based on ESP.

3.4.2 PARALLEL TYPE MULTIPLIERS BASED ON ESP

In this subsection, we consider the application of ESP to the configuration of parallel type multipliers. The configuration of parallel type multipliers based on ESP, called *Equally Spaced Polynomial Multipliers* (denoted *ESPM*), is very similar with that of CBM. Here we have the following two theorems, which correspond to Theorem 3.13 and 3.14, for the configuration of parallel type ESPM.

THEOREM 3.23 [IMT87,IT87b] *Let $g(x)$ be an s-ESP of degree sn. Then $g(x)$ divides $x^{(n+1)s}+1$, i.e., $p(x)\mid x^{(n+1)}+1$.* □

THEOREM 3.24 [IMT87,IT87b] *Let $D(x)$ and $d(x)$ be polynomials over $GF(2)$ of degree $s(n+1)-1$ and $sn-1$, respectively.*

$$D(x)=D_0+D_1x+\cdots+D_{(n+1)s-1}x^{(n+1)s-1},$$

$$d(x)=d_0+d_1+\cdots+d_{ns-1}x^{ns-1}.$$

Assume that $D(x)$ and $d(x)$ have the relation such that

$$D(x)=d(x) \pmod{g(x)},$$

where $g(x)$ is an s-ESP of degree sn. Then the coefficients of $d(x)$ are given by those of $D(x)$, i.e.,

$d_{i+js}=D_{i+js}+D_{i+sn} \pmod 2$ $(0\leqq i\leqq s-1, 0\leqq j\leqq n-1)$. □

The following example illustrates the parallel type configuration of ESPM.

EXAMPLE 3.25:

Assume that m=2 and k=2, then 3(=m+1) is a prime, 2 is the generator in $GF^*(3)$ and $2^2 \not\equiv 1 \pmod{3^2}$. Hence $g(x)=x^6+x^3+1$, 3-ESP of degree 6, is irreducible over $GF(2)$. (See Theorem 3.21.) For $\forall a,b \in GF(2^6)$,

$$a=a_0+a_1x+a_2x^2+\cdots+a_5x^5, \quad a_i \in GF(2) \ (0 \leqq i \leqq 5),$$

$$b=b_0+b_1x+b_2x^2+\cdots+b_5x^5, \quad b_j \in GF(2) \ (0 \leqq j \leqq 5),$$

where $\{1,x,x^2,x^3,x^4,x^5\}$ is a canonical basis, the product of a and b, i.e., d=ab, is given by

$$d=d_0+d_1x+d_2x^2+\cdots+d_5x^5, \quad d_k \in GF(2) \ (0 \leqq k \leqq 5).$$

Here define A and B such that

$$A=(A_0,A_1,\cdots,A_8)=(a_0,a_1,\cdots,a_5,0,0,0),$$

$$B=(B_0,B_1,\cdots,B_8)=(b_0,b_1,\cdots,b_5,0,0,0).$$

(We also call A an *extended vector representation* of a as in *EXAMPLE 3.15.*) Furthermore define $D(=AB \pmod{x^9+1})$ such that

$$D=AB \pmod{x^9+1}$$

$$=(A_0+A_1x+\cdots+A_8x^8)(B_0+B_1x+\cdots+B_8x^8) \pmod{x^9+1},$$

where $A=A_0+A_1x+\cdots+A_8x^8$ and $B=B_0+B_1x+\cdots+B_8x^8$. Here D_i are given by $D_i=\sum_{j+k=i \pmod 9} A_jB_k \pmod 2$ $(0 \leqq i \leqq 8)$, thus we have

$$D_8=t(A_0,A_1,\cdots,A_8;B_0,B_1,\cdots,B_8)$$

$$=A_8B_0+A_7B_1+A_6B_2+A_5B_3+A_4B_4+A_3B_5+A_2B_6+A_1B_7+A_0B_8,$$

$$D_7=t(A_8,A_0,A_1,\cdots,A_7;B_0,B_1,\cdots,B_8),$$

$$\vdots$$

$$D_0=t(A_1,A_2,\cdots,A_8,A_0;B_0,B_1,\cdots,B_8).$$

Recalling Theorem 3.23, d (=ab mod g(x)) and D (defined above) have the relation such that $d=D \pmod{g(x)}$, where $g(x)=x^6+x^3+1$.

Furthermore noting Theorem 3.24 (s=2 and n=3), d_k ($0 \leqq i \leqq 5$) are given by $d_{i+2j}=D_{i+2j}+D_{i+6}$ (mod 2) ($0 \leqq i \leqq 1$, $0 \leqq j \leqq 2$). Thus we have the configuration of ESPM in $GF(2^6)$ as in FIGURE 3.26.

In this example, squaring A can be carried out by

$$A^2=(A_0+A_1x+A_2x^2+\cdots+A_8x^8)^2 \pmod{x^9+1}$$
$$= A_0+A_1x^2+A_2x^4+\cdots+A_8x^{16} \pmod{x^9+1}$$
$$= A_0+A_5x+A_1x^2+A_6x^3+A_2x^4+A_7x^5+A_3x^6+A_8x^7+A_4x^8.$$

Thus squaring A is given by the following permutation:

$$(A_0, A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8) \rightarrow (A_0, A_5, A_1, A_6, A_2, A_7, A_3, A_8, A_4)$$

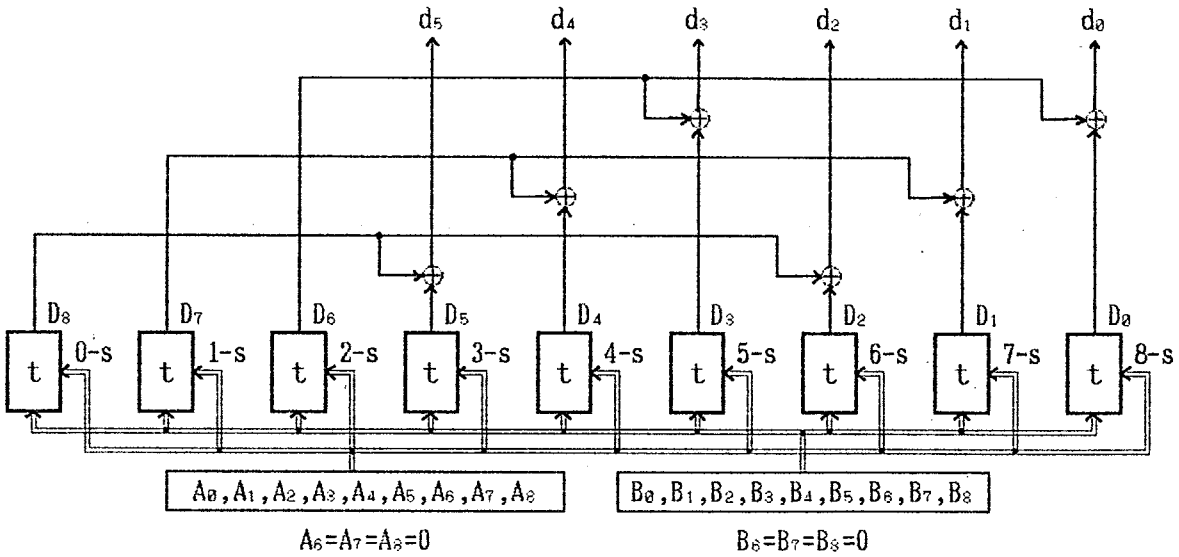
and this can be also easily implemented by hardwiring. □

3.4.3 NUMBER OF GATES IN ESPM

ESPM in $GF(2^M)$, where $M=m(m+1)^{k-1}$, is implemented by a $(m+1)^{k-1}$-ESP of degree $m(m+1)^{k-1}$. Observing ***EXAMPLE 3.25***, the total number of AND-gates $AND_{ESPM}(M)$ and EOR-gates $EOR_{ESPM}(M)$ in parallel type configuration of ESPM in $GF(2^M)$ are:

$$AND_{ESPM}(M)=(m+1)^k(m+1)^k=(m+1)^{2k},$$
$$EOR_{ESPM}(M)=\{(m+1)^k-1\}(m+1)^k+m(m+1)^{k-1}$$
$$=(m+1)^{2k}-(m+1)^{k-1},$$

where $M=m(m+1)^{k-1}$. The above equations imply that parallel type ESPM in $GF(2^M)$ can be implemented by $O(M^2)$ AND-gates and $O(M^2)$ EOR-gates, where M is the dimension of $GF(2^M)$ ($M=m(m+1)^{k-1}$). The parallel type MOM for $GF(2^m)$, in general, has $O(m^3)$ AND-gates and $O(m^3)$ EOR-gates, however, no condition that parallel type MOM for $GF(2^m)$ has $O(m^2)$ AND-gates and $O(m^2)$ EOR-gates has been known yet except for Lemma 3.12. Thus we can conclude that ESPM for $GF(2^M)$ has small circuit size, because $AND_{ESPM}(M)$



k-s : k cyclic shifts

FIGURE 3.26 Configuration of ESPM in $GF(2^6)$

and EOR_{ESPM} are $O(M^2)$, where $M=m(m+1)^{k-1}$.

3.5. CONCLUSION

In this Chapter, we presented two kinds of parallel type multipliers in $GF(2^m)$, i.e., one is CBM based on canonical bases and the other is ESPM by introducing a new concept, *Equally Spaced Polynomial*. The features of the proposed multipliers can be summarized as follows:

F1. CBM and ESPM have regularity and modularity in the structure;

F2. The number of gates, AND-gates and EOR-gates, in CBM and ESPM for $GF(2^m)$ is $O(m^2)$;

F3. The computation time of CBM and ESPM for $GF(2^m)$ is $O(log\ m)$;

F4. For CBM and ESPM, squaring $\forall a \in GF(2^m)$ can be carried out by permutation of the *extended* vector representation of a.

Here we restate the main results obtained in this Chapter.

CBM, *Canonical Bases Multipliers*, for $GF(2^m)$ is defined. The total number of AND-gates of CBM is almost the same with that of parallel type MOM and the total number of EOR-gates of CBM is about half of that of parallel type MOM. (See Eq.(3.1)~(3.4).) A new concept, *Equally Spaced Polynomial* (denoted *ESP*), is introduced and Theorem 3.21 provides a necessary and sufficient condition for ESP's to be irreducible. Corollary 3.22, in addition, gives a simple *criterion* for producing a sequence of irreducible ESP's. Corollary 3.22 is of considerable and prac-

tical use in designing multipliers of various finite field $GF(2^m)$. Furthermore ESPM, *Equally Spaced Polynomial Multipliers*, for $GF(2^m)$ is presented as the application of ESP. ESPM for $GF(2^m)$ can be implemented with $O(m^2)$ AND-gates and $O(m^2)$ EOR-gates and furthermore with $O(log\ m)$ delay time.

CBM and ESPM can not be *necessarily* constructed for $GF(2^m)$ of arbitrary m, because some restrictions are imposed on m, e.g., (m+1) is a prime, 2 is the generator in $GF^*(m+1)$, etc. In general, however, the number of gates for parallel type MOM is proportional to m^3, thus CBM's or ESPM's are advantageous to MOM in parallel type configuration for m satisfying S2. in Lemma 3.5 or Theorem 3.21. CBM and ESPM have the similar property with parallel type MOM, (See F4. in the above.) hence they are applicable to the fast algorithm for computing multiplicative inverses in $GF(2^m)$ [Va87,IT88a], which requires $O(log\ m)$ multiplications in $GF(2^m)$ and (m-1) permutations.

Further studies are required to give a constructive method to design parallel type multipliers in $GF(2^m)$, which have regularity and modularity in the structure and small circuit size, i.e., proportional to m^2.

CHAPTER IV :

FAST ALGORITHM FOR MULTIPLICATIVE INVERSES IN FINITE FIELDS

This Chapter develops (*sequential* and *recursive* type) fast algorithms for computing multiplicative inverses in finite fields. The fast algorithm for $GF(2^m)$ developed in this Chapter iterates multiplications in $GF(2^m)$ to compute multiplicative inverses in $GF(2^m)$ and requires $O(log\ m)$ multiplications in $GF(2^m)$ and $O(m)$ cyclic shifts.

4.1 INTRODUCTION

In this Chapter, we study a fast algorithm for computing multiplicative inverses in finite fields, especially in $GF(2^m)$. (The extension of the fast algorithm to general finite fields $GF(q^m)$ is *not difficult!* Thus we concentrate on the studies of the fast algorithm for $GF(2^m)$.)

Several digital audio instruments, e.g., *CD* (*Compact Disc*) *player* and *DAT* (*Digital Audio Tape-recorder*), make use of *Reed-Solomon codes* over $GF(2^8)$. In those instruments, table look-up is applied for deciding multiplicative inverses in the field. Such a method, however, is not *realistic* for somewhat larger finite fields, e.g., $GF(2^{16})$, because the table size *enormously* grows. *Euclidean Algorithm* [Ber68], on the other hand, is well-known as an algorithm for computing multiplicative inverses in finite fields. Though this algorithm can be efficiently carried out, it is unfortunately *not suitable* for hardware implementation.

In 1985, Wang *et.al* developed an *elegant* algorithm for computing multiplicative inverses in $GF(2^m)$, employing normal bases [WTSDOR85]. The algorithm is suitable for *VLSI implementation* and requires (m-2) multiplications in $GF(2^m)$ and (m-1) cyclic shifts to compute multiplicative inverses in $GF(2^m)$. However the running cost of *Wang's Algorithm* is somewhat large, i.e., the number of multiplications in $GF(2^m)$ is proportional to m, thus the reduction of the number of multiplications is necessary especially for $GF(2^m)$ of large order. The studies in this Chapter are motivated by such a requirement.

This Chapter proposes two kinds of fast algorithms, i.e., *sequential type* and *recursive type*, for computing multiplicative inverses in $GF(2^m)$. The sequential type fast algorithm for $GF(2^m)$ [IT87a,Va87,IT88a] requires $O(log\ m)$ multiplications in $GF(2^m)$, thus the running cost of the sequential type algorithm is $O((log\ m)^2)$ because that of the multiplications in $GF(2^m)$ is $O(log\ m)$ by parallel type MOM. The recursive type fast algorithm for $GF(2^m)$ $(m=2^k)$ [IT88b,IT88d], on the other hand, requires two multiplications in $GF(2^m)$ and in each subfield of $GF(2^m)$, i.e., $GF(2^{m/2})$, $GF(2^{m/4}),\ldots,GF(2^8)$ and $GF(2^4)$. Furthermore the recursive type fast algorithm is suitable for hardware implementation because of its *hierarchical* structure.

The organization of this Chapter is as follows:

Section 4.2 is the preliminaries for this Chapter and provides some mathematical preliminaries and the related works.

In Section 4.3, we develop a sequential type fast algorithm for computing multiplicative inverses in $GF(2^m)$ [IT87a,Va87,IT88a], and mention the extension of the algorithm to general fields $GF(q^m)$. Furthermore Section 4.4 proposes a recursive type fast algorithm for multiplicative inverses in $GF(2^m)$ ($m=2^k$) [IT88b, IT88d] and considers hardware implementation of the recursive algorithm. Section 4.5 finally summarizes the results in this Chapter and gives conclusions, some remarks and open problems.

4.2 PRELIMINARIES

This Section provides some mathematical definitions and lemmas, and furthermore gives the related works for the subsequent discussions.

4.2.1 MATHEMATICAL PRELIMINARIES

In this subsection, we give some mathematical definitions and lemmas for the subsequent discussions.

DEFINITION 4.1 [MS77] *A basis in $GF(2^m)$ over $GF(2)$ of the form, $\{a, a^2, a^{2^2}, \cdots, a^{2^{(m-1)}}\}$, is called a **normal basis** in $GF(2^m)$ over $GF(2)$.* □

Remark 4.2 The above definition is completely the same with Definition 3.1 in Chapter 3. □

Remark 4.3 The above notation, 2^2 and 2^(m-1), similarly implies 2^2 and $2^{(m-1)}$, respectively. (See Remark 3.2 in Chapter 3.) □

LEMMA 4.4 [WTSDOR85] *Let $x \in GF(2^m)$. If x is represented by a normal basis such that*

$$x=x_0 a+x_1 a^2+x_2 a^{2\hat{}2}+\cdots+x_{m-1} a^{2\hat{}(m-1)}$$

$$=[x_0, x_1, x_2, , \cdots, x_{m-1}],$$

then we have $x^2=[x_{m-1}, x_0, x_1, \cdots, x_{m-2}]$, *i.e.,* x^2 *can be computed by a cyclic shift of the above vector representation of* x. □

Remark 4.5 We call the cyclic shift in Lemma 4.4 "cyclic shift over $GF(2)$" throughout this Chapter. □

LEMMA 4.6 [MS77] *For* $\forall x \in GF(2^m)$, *we have the identity such that* $e^{2\hat{}m}=e$. □

Lemma 4.6 is the celebrated *Fermat's Theorem* [Sch86]. Here we have the following lemma for $\forall x \in GF(2^m)$ $(x \neq 0)$, which is one of the most essential lemmas in this Chapter.

LEMMA 4.7 [WTSDOR85] *For* $\forall x \in GF(2^m)$ $(x \neq 0)$, *we have the identity such that* $e^{-1}=e^{(2\hat{}m)-2}$. □

4.2.2 THE WANG'S ALGORITHM

This subsection presents the algorithm proposed by Wang *et.al.* [WTSDOR85], which computes multiplicative inverses in $GF(2^m)$ and is suitable for *VLSI implementation*. For $\forall x \in GF(2^m)$ $(x \neq 0)$, we have the identity such that $x^{-1}=x^{(2\hat{}m)-2}$. (See Lemma 4.7.) Here $2\hat{}m-2=(2^m-2)=2+2^2+\cdots+2^{m-1}$, thus we have

$$x^{-1}=(x^2)(x^{2\hat{}2})\cdots(x^{2\hat{}(m-1)}). \qquad (4.1)$$

Assume that multiplications in $GF(2^m)$ are performed by parallel type MOM. (See Chapter 3.) Note that for $\forall x \in GF(2^m)$ x^2 can be carried out by a cyclic shift over $GF(2)$ of vector representation of x. (See Lemma 4.4.) The running cost of multiplications in $GF(2^m)$ is considerably larger than that of

cyclic shifts over $GF(2)$, thus we have a *systematical* procedure for multiplicative inverses in $GF(2^m)$ employing normal bases with comparatively low running cost [WTSDOR85]. The algorithm described below computes multiplicative inverses in $GF(2^m)$, following Eq.(4.1).

ALGORITHM 4.8:

```
S1.  y := x
S2.    for k := 1 to m-2 do
S3.      begin
S4.        z := y²   (a cyclic shift over GF(2))
S5.        y := zx   (a multiplication in GF(2^m))
S6.      end
S7.    y := y²       (a cyclic shift over GF(2))
S8.  write y
```

□

Recalling Lemma 4.4, *ALGORITHM 4.8* requires (m-2) multiplications in $GF(2^m)$ and (m-1) cyclic shifts over $GF(2)$ to compute multiplicative inverses in $GF(2^m)$.

4.3 SEQUENTIAL TYPE FAST ALGORITHM FOR MULTIPLICATIVE INVERSES

This Section provides a *sequential* type fast algorithm for computing multiplicative inverses in both $GF(2^m)$ and $GF(q^m)$, employing normal bases.

4.3.1 SEQUENTIAL TYPE FAST ALGORITHM FOR $GF(2^m)$

In this subsection, we develop a sequential type fast algorithm for computing multiplicative inverses in $GF(2^m)$, employing *normal bases*. The central idea to derive the fast algorithm is very simple and is regarded as an extension of Lemma 4.4.

LEMMA 4.9 [IT87a,Va87,IT88a] *Let* $x \in GF(2^m)$. *If* x *is represented by a normal basis such that*

$$x=x_0 a+x_1 a^2+x_2 a^{2^2}+\cdots+x_{m-1} a^{2^{(m-1)}}$$

$$=[x_0, x_1, x_2, , \cdots, x_{m-1}],$$

then we have $x^{2^k}=[x_{m-k}, x_{m-k+1}, \cdots, x_{m-1}, x_0, \cdots, x_{m-k-1}]$, *i.e.,* x^{2^k} $(1 \leqq k \leqq m-1)$ *can be computed by* k *cyclic shifts over* $GF(2)$ *of the above vector representation of* x. □

Here we present a sequential type fast algorithm for multiplicative inverses in $GF(2^m)$ in the special case that $m=2^r+1$. (The algorithm for the special case the $m=2^r+1$ can be regarded as a *sub-algorithm* to design a *general* sequential fast algorithm for computing multiplicative inverses in $GF(2^m)$, which will be presented in Theorem 4.13.)

THEOREM 4.10 [IT87a,Va87,IT88a] *Let* $x \in GF(2^m)$ $(m=2^r+1, x \neq 0)$. *Then there exists a sequential type algorithm for computing* x^{-1}, *which requires* $\{\log (m-1)\}=r$ *multiplications in* $GF(2^m)$ *and* $(m-1)=2^r$ *cyclic shifts over* $GF(2)$. □

Remark 4.11 The above Theorem is suppose to have been found *independently* at *almost the same time* by author [IT87a,IT88a] and S.A. Vanstone [Va87] of the University of Waterloo. □

Proof: For simplicity of the notations, the following symbols are defined such that

$$\#t=1+2+2^2+\cdots+2^{(t-1)},$$

$$\%t=1+2+2^2+\cdots+2^{(2^t)-1},$$

$$\&t=2^{2^t},$$

and they will be used throughout this Chapter. Let M(t) and

S(t) be the number of multiplications in $GF(2^m)$ $(m=2^r+1)$ and that of cyclic shifts over $GF(2)$, respectively, to compute $x^{\%t}$ $(1 \leqq t \leqq r)$. Since $x^{\%t}=\{x^{\%(t-1)}\}^{\&(t-1)} \cdot x^{\%(t-1)}$ $(1 \leqq t \leqq r)$, we have $M(t)=M(t-1)+1$ and $S(t)=S(t-1)+2^{t-1}$. (See Lemma 4.9.) Here it is clear that $M(0)=S(0)=0$, thus $M(r)=r$ and $S(r)=2^r-1$. Note that

$$2^m-2=2\#(m-1)=2\#2^r$$
$$=2\%r=2\{\%(r-1)\cdot\&(r-1)+\%(r-1)\}.$$

Hence we have $x^{-1}=x^{(2^m)-2}=(x^{\%r})^2$, and thus the algorithm proposed above requires $r=(log\ m)$ multiplications in $GF(2^m)$ and $S(r)+1=2^r(=m-1)$ cyclic shifts over $GF(2)$. □

The fast algorithm for $GF(2^m)$ $(m=2^r+1)$ proposed in Theorem 4.11 can be described as follows:

ALGORITHM 4.12:

```
S1.  y := x
S2.    for k := 0 to r-1 do
S3.      begin
S4.        z := y^&k   (2^k cyclic shifts over GF(2))
S5.        y := yz     (a multiplication in GF(2^m))
S6.      end
S7.    y := y^2        (a cyclic shift over GF(2))
S8.  write y
```

□

The following Theorem gives a general sequential type algorithm for computing multiplicative inverses in $GF(2^m)$, which has no restrictions such that $m=2^r+1$.

THEOREM 4.13 [IT87a,Va87,IT88a] *Let $x \in GF(2^m)$ $(x \neq 0)$. Then there exists a general sequential type algorithm for computing x^{-1}, which requires $\{[log\ (m-1)]+H_w(m-1)-1\}$ multiplications in $GF(2^m)$ and $(m-1)$ cyclic shifts over $GF(2)$,*

where [x] denotes maximum a integer $\leqq x$ *and* $H_w(i)$ *denotes the Hamming weight of binary representation of i.* □

Remark 4.14 The above Theorem is also suppose to have been found *independently* at *almost the same time* by author [IT87a,IT88a] and S.A. Vanstone [Va87] of the University of Waterloo. □

Proof: Note that $x^{-1}=\{x^{\#(m-1)}\}^2$. Assume that

$$m-1=2^{k_1}+2^{k_2}+\cdots+2^{k_n},$$

where $k_1>k_2>\cdots>k_n$, thus we have

$$x^{-1}=\{(x^{\%k_1})^{2^{e_1}}(x^{\%k_2})^{2^{e_2}}\cdots(x^{\%k_n})^{2^{e_n}}\}^2,$$

where $e_j=2^{k_{j+1}}+2^{k_{j+2}}+\cdots+2^{k_n}$ and $e_n=0$. Reordering the terms in the above equation, we have

$$x^{-1}=[x^{\%k_n}\{x^{\%k_{n-1}}\cdots(x^{\%k_2}(x^{\%k_1})^{2^{k_2}})^{2^{k_3}}\cdots\}^{2^{k_n}}]^2.$$

Let M(t) and S(t) be the number of multiplications in $GF(2^m)$ and cyclic shifts over $GF(2)$, respectively, to compute $x^{\%t}$. By Theorem 4.10, we have $M(k_1)=k_1$ and $S(k_1)=(2^{k_1})-1$. Since every term $x^{\%k_j}$ $(2\leqq j\leqq n)$ is already computed in the process of computing $x^{\%k_1}$ (See the *Proof* of Theorem 4.10.), the algorithm proposed above requires (k_1+n-1) multiplications in $GF(2^m)$ and $\{2^{k_1}-1+(2^{k_2}+2^{k_3}+\cdots+2^{k_n}+1)\}=m-1$ cyclic shifts over $GF(2)$.

Noting the facts that $[log\ (m-1)]=k_1$ and $H_w=n$, we can finally conclude that the algorithm presented in this Theorem requires $\{[log\ (m-1)]+H_w(m-1)-1\}$ multiplications in $GF(2^m)$ and $(m-1)$ cyclic shifts over $GF(2)$. □

EXAMPLE 4.15:

Let $x\in GF(2^{11})$ $(x\neq 0)$. Notice that $x^{-1}=x^{(2^{11})-2}=x^{2046}$, thus we have the following procedure:

S1. $(x)^2=x^2$: 1 cyclic shift over $GF(2)$

S2. $x^2x=x^3$: 1 multiplication in $GF(2^{11})$

S3. $(x^3)^{2^2}=x^{12}$: 2 cyclic shifts over $GF(2)$

S4. $x^{12}x^3=x^{15}$: 1 multiplication in $GF(2^{11})$

S5. $(x^{15})^{2^4}=x^{240}$: 4 cyclic shifts over $GF(2)$

S6. $x^{240}x^{15}=x^{255}$: 1 multiplication in $GF(2^{11})$

S7. $(x^{255})^{2^2}=x^{1020}$: 2 cyclic shifts over $GF(2)$

S8. $x^{1020}x^3=x^{1023}$: 1 multiplication in $GF(2^{11})$

S9. $(x^{1023})^2=x^{2046}$: 1 cyclic shift over $GF(2)$

$=x^{-1}$

□

Observing the above procedure, the number of multiplications in $GF(2^{11})$ is 4 (in S2, S4, S6 and S8) and that of cyclic shifts over $GF(2)$ is 10 (in S1, S3, S5, S7 and S9). Here we have $[log\ (11-1)]+H_w(11-1)-1=4$ and $11-1=10$, thus the above Example confirms Theorem 4.13.

4.3.2 SEQUENTIAL TYPE FAST ALGORITHM IN $GF(q^m)$ $(q=2^n)$

This subsection presents a sequential type fast algorithm for computing multiplicative inverses in $GF(q^m)$ $(q=2^n)$, employing normal bases. The central idea to derive the fast algorithm is also very simple. The following Lemma is a natural extension of Lemma 4.9.

LEMMA 4.16 [IT87a,IT88a] *Let $x\in GF(q^m)$ $(q=2^n)$. If x is represented by a normal basis such that*

$$x=x_0\alpha+x_1\alpha^q+x_2\alpha^{q^2}+\cdots+x_{m-1}\alpha^{q^{(m-1)}}$$

$$=[x_0, x_1, x_2, , \cdots, x_{m-1}],$$

then we have $x^{q^k}=[x_{m-k}, x_{m-k+1}, \cdots, x_{m-1}, x_0, \cdots, x_{m-k-1}]$, i.e., x^{q^k} $(1\leqq k\leqq m-1)$ can be computed by k cyclic shifts

of the above vector representation of x. □

Remark 4.17 We call the cyclic shifts in Lemma 4.16 "cyclic shifts over $GF(q)$ $(q=2^n)$" in the rest of this Chapter. □

Applying the above Lemma, we have a sequential type fast algorithm for computing multiplicative inverses in $GF(q^m)(q=2^n)$.

THEOREM 4.18 [IT87a,IT88a] *Let $x \in GF(q^m)$ $(q=2^n, x \neq 0)$. Then there exists an algorithm for computing x^{-1}, which requires $\{[\log (m-1)]+H_w(m-1)\}$ multiplications in $GF(q^m)$, $\{[\log (n-1)]+H_w(n-1)-1\}$ multiplications in $GF(q)$ $(q=2^n)$, $(m-1)$ cyclic shifts over $GF(q)$ and $(n-1)$ cyclic shifts over $GF(2)$, where $[x]$ and $H_w(i)$ denote the same symbols with Theorem 4.13, respectively.* □

Proof: Let $x \in GF(q^m)$ $(q=2^n, x \neq 0)$. Notice that $x^{-1}=x^{(q^m)-2}$. Here q^m-2 can be decomposed by

$$q^m-2=(q-2)(q^{m-1}+q^{m-2}+\cdots+q+1)+(q^{m-1}+q^{m-2}+\cdots+q)$$
$$=(q-2)a+b,$$

where $a=q^{m-1}+q^{m-2}+\cdots+q+1$ and $b=q^{m-1}+q^{m-2}+\cdots+q$. Thus we have $x^{-1}=y^{q-2}z$, where $y=x^a$ and $z=x^b$. Note that $y=zx$. Applying the similar procedure in Theorem 4.13 (See *Proof* of Theorem 4.13.), $\{[\log (m-1)]+H_w(m-1)\}$ multiplications in $GF(q^m)$ and $(m-1)$ cyclic shifts over $GF(q)$ are required to compute y and z. Since y is the *Norm* of x [LN83], $y \in GF(q)=GF(2^n)$. Hence $y^{q-2}=y^{-1}$, and by Theorem 4.13, $\{[\log (n-1)]+H_w(n-1)-1\}$ multiplications in $GF(q)=GF(2^n)$ and $(n-1)$ cyclic shifts over $GF(2)$ are required to compute y^{-1}. This completes the proof. □

This Section has presented the sequential type fast algorithms for computing multiplicative inverses in $GF(2^m)$ and

$GF(q^m)(q=2^n)$. (See Theorem 4.13 and Theorem 4.18.) Theorem 4.13 is of practical use for error-correcting codes, cryptographies, etc., while Theorem 4.18 is somewhat of theoretical interest. (The generalization of the algorithm in Theorem 4.18 to $GF(q^m)$ ($q=p^n$, p;odd prime) is, needless to say, possible in the similar way with Theorem 4.18.) However, Theorem 4.18 inspires the clue of the studies on a *recursive* type fast algorithm for computing multiplicative inverses in $GF(2^m)(m=2^k)$, which will be presented in the next Section.

4.4 RECURSIVE TYPE FAST ALGORITHM FOR MULTIPLICATIVE INVERSES

In this Section, we develop a *recursive* type fast algorithm for computing multiplicative inverses in $GF(2^m)$ ($m=2^k$). The recursive algorithm for $GF(2^m)$ ($m=2^k$) requires two multiplications in $GF(2^m)$ and in each subfield of $GF(2^m)$, i.e., $GF(2^{m/2}), GF(2^{m/4}), \cdots, GF(2^8)$ and $GF(2^4)$, and (m-1) cyclic shifts over $GF(2)$ to compute multiplicative inverses in $GF(2^m)$ ($m=2^k$). Furthermore the recursive algorithm has an additional advantageous feature that every multiplication in each subfield of $GF(2^m)$ can be carried out by the same multiplier in $GF(2^m)$, and is suitable for hardware implementation because of its *hierarchical* structure.

4.4.1 RECURSIVE TYPE FAST ALGORITHM FOR MULTIPLICATIVE INVERSES

This subsection proposes a recursive type fast algorithm for computing multiplicative inverses in $GF(2^m)$ ($m=2^k$). The core idea to derive the recursive algorithm is very simple and

is materialized by the following lemma:

LEMMA 4.19 [Ber68,Sch86] *Let x be a non-zero element in a finite field of characteristic 2. Then $x \in GF(2^m)$ if and only if $x^{2^m-1}=1$.* □

For $x \in GF(2^m)$ $(m=2^k, x \neq 0)$, the multiplicative inverse x^{-1} can be computed by $x^{-1}=x^{2^m-2}$. (See Lemma 4.7.) Here 2^m-2 can be decomposed by $2^m-2=(2^{m/2}+1)(2^{m/2}-2)+2^{m/2}$, thus

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

Squaring $x(\in GF(2^m))$ can be carried out by a cyclic shift over $GF(2)$ (See Lemma 4.4.), employing normal bases, hence the computation of x^{-1}, i.e., $x^{-1}=(x^{2^{(m/2)}} \times x)^{2^{(m/2)-2}} \times x^{2^{(m/2)}}$, requires m/2 cyclic shifts over $GF(2)$ and two multiplications in $GF(2^m)$, i.e., the computation of $y_1 (=x^{2^{(m/2)}} \times x)$ and that of $y_1^{2^{(m/2)-2}} \times x^{2^{(m/2)}}$. Note that

$$y_1^{2^{(m/2)-1}}=(x^{2^{(m/2)+1}})^{2^{(m/2)-1}}=x^{2^m-1}=1,$$

thus $y_1 \in GF(2^{m/2})$. (See Lemma 4.19.) Hence $y_1^{2^{(m/2)-2}}=y_1^{-1}$. This implies that the algorithm for multiplicative inverses in $GF(2^m)$ includes that in $GF(2^{m/2})$ as a *sub-algorithm*. Similarly the computation of $y_1^{-1}=y_1^{2^{(m/2)-2}}$ can be carried out by

$$y_1^{-1}=y_1^{2^{(m/2)-2}}$$
$$=(y_1^{2^{(m/4)+1}})^{2^{(m/4)-2}} \times y_1^{2^{(m/4)}},$$

hence it requires m/4 cyclic shifts over $GF(2)$, two multiplications in $GF(2^{m/2})$ and the computation of $y_2^{-1}=y_2^{2^{(m/4)-2}}$, where $y_2=y_1^{2^{(m/4)+1}} \in GF(2^{m/4})$. (See Lemma 4.19.) This also implies that the algorithm for multiplicative inverses in $GF(2^{m/2})$ includes that in $GF(2^{m/4})$ as a *sub-algorithm*. Noting that the

hierarchical structure of the above algorithm, the recursive algorithm for computing multiplicative inverses in $GF(2^m)$ $(m=2^k)$ can be defined as follows:

$$y_i=(y_{i-1})^{2^{\{m/(2^i)\}+1}} \quad (1\leqq i\leqq k),$$

$$(y_{i-1})^{-1}=\{(y_{i-1})^{2^{\{m/(2^i)\}+1}}\}^{2^{\{m/(2^i)\}-2}} \times (y_{i-1})^{2^{\{m/(2^i)\}}}$$

$$=y_i^{-1}\times (y_{i-1})^{2^{\{m/(2^i)\}}} \quad (1\leqq i\leqq k),$$

where k is an integer satisfying $m=2^k$ and $y_0=x$. It requires $m/2^i$ cyclic shifts over $GF(2)$ and two multiplications in $GF(2^{m/\{2^{(i-1)}\}})$, where $1\leqq i\leqq k$, for computing y_i and $(y_{i-1})^{-1}$. The above procedure terminates at the step of $(y_{k-1})^{-1}=(y_{k-1})^2$, in which requires one cyclic shift over $GF(2)$ but no multiplications because $y_k=1$. Hence the total number of cyclic shifts over $GF(2)$ in the recursive algorithm is $m/2+m/4+\cdots+4+2+1=m-1$, where $m=2^k$. Here we have the following Theorem:

THEOREM 4.20 [IT88b,IT88d] *There exists a recursive algorithm for computing multiplicative inverses in $GF(2^m)$, where $m=2^k$, which requires $(m-1)$ cyclic shifts over $GF(2)$ and two multiplications in $GF(2^m)$ and in each subfield of $GF(2^m)$, i.e., $GF(2^{m/2}), GF(2^{m/4}), \cdots, GF(2^8)$ and $GF(2^4)$.* □

Remark 4.21 Consider the case for computing multiplicative inverses in $GF(2^{16})$. By the sequential type fast algorithm in Theorem 4.13, 6 multiplications in $GF(2^{16})$ and 15 cyclic shifts over $GF(2)$ are required to compute multiplicative inverses in $GF(2^{16})$ (See Theorem 4.13.) On the other hand, by the recursive type fast algorithm in Theorem 4.20, 2 multiplications in each of $GF(2^{16})$, $GF(2^8)$ and $GF(2^4)$ and 15 cyclic shifts over $GF(2)$

are required to compute multiplicative inverses in $GF(2^{16})$. It is clear that the running cost of multiplications in $GF(2^{16})$ is larger than that in each of $GF(2^8)$ and $GF(2^4)$, thus the total running cost of the recursive type fast algorithm is less than that of the sequential type fast algorithm. In the case of $GF(2^m)$ of larger order, e.g., $GF(2^{32})$, $GF(2^{64})$, etc., the difference of the running cost between the sequential and recursive type algorithm *enormously* grows! □

The above recursive algorithm can be described as follows:

ALGORITHM 4.22:

```
S1.  function inv(x,m);
S2.    begin
S3.      if m=1 or m=2
S4.        then
S5.          if m=1
S6.            then inv:=x
S7.            else inv:=one cyclic shift over GF(2) of x
S8.        else
S10.         begin
S11.           a:=m/2 cyclic shifts over GF(2) of x;
S12.           b:=a× x;
S13.           if m=4
S14.             then c:=one cyclic shift over GF(2) of b
S15.             else c:=inv(b,m/2);
S16.           inv:=a× c;
S17.         end;
S18.   end;
```

□

4.4.2 HARDWARE IMPLEMENTATION OF THE RECURSIVE ALGORITHM

The recursive algorithm proposed in Subsection 4.4.1 is

suitable for hardware implementation because of its *hierarchical* structure. In this subsection, we demonstrate the *availability* of the recursive algorithm in hardware implementation.

LEMMA 4.23 [IT88b,IT88d] *Let* $\{a, a^2, a^{2^2}, \cdots, a^{2^{(m-1)}}\}$ *be a normal basis in* $GF(2^m)$, *where* $m=2^k$. *For* $\forall x \in GF(2^{m/(2^i)})$ $(1 \leqq i \leqq k)$, x *can be represented by the normal basis as*

$$x=[x_1, x_2, \cdots, x_{2^i}],$$

where $x_j=[x_0, x_1, \cdots, x_{m/(2^i)-1}]$ $(1 \leqq j \leqq 2^i, 1 \leqq i \leqq k)$. □

Proof: For simplicity we prove the special case that i=1. The general case that $1 \leqq i \leqq k$ can be proven in the similar way. Let $x \in GF(2^{m/2})$. Here x has a vector representation by the normal basis in $GF(2^m)$ such that $x=[x_0, x_1, \cdots, x_{m-1}]$. By the assumption that $x \in GF(2^{m/2})$, we have $x^{2^{(m/2)}}=x$. (See Lemma 4.19.) Notice that $x^{2^{(m/2)}}$ can be computed by m/2 cyclic shifts over $GF(2)$ of x. (See Lemma 4.9.) Hence we have

$$[x_{m/2}, x_{m/2+1}, \cdots, x_{m-1}, x_0, x_1, \cdots, x_{m/2-1}]=[x_0, x_1, \cdots, x_{m-1}].$$

This implies that $x_i=x_{i+m/2}$ $(0 \leqq i \leqq m/2-1)$. □

Lemma 4.23 suggests that multiplications in each subfield of $GF(2^m)$ $(m=2^k)$, i.e., $GF(2^{m/2}), GF(2^{m/4}), \cdots, GF(2^8)$ and $GF(2^4)$ can be carried out by the same multiplier for $GF(2^m)$. Thus we have two types of configurations of the recursive algorithm in hardware as follows:

CONFIGURATION OF TYPE 1:

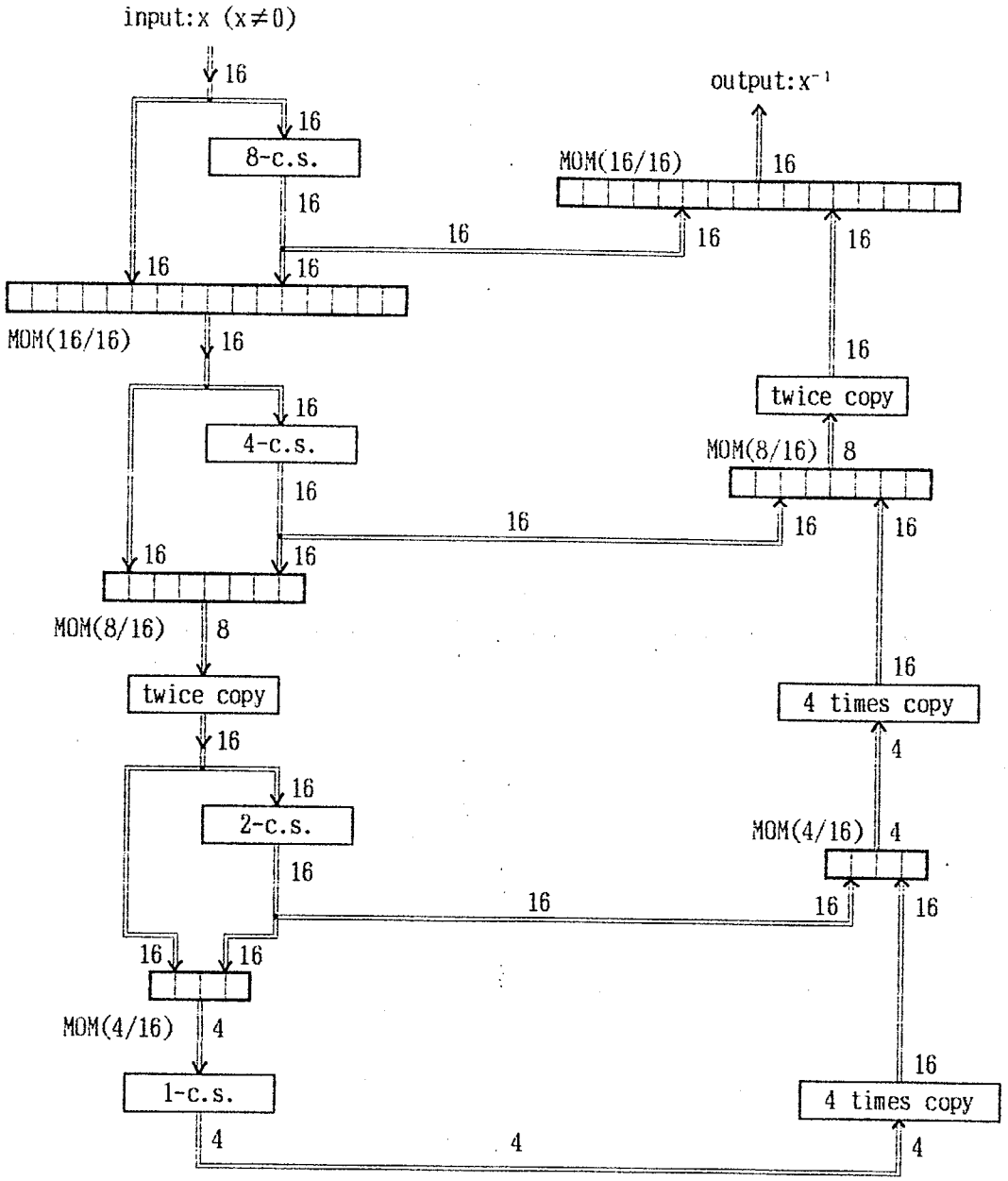
Prepare a parallel type MOM for $GF(2^m)$ $(m=2^k)$. Notice that a parallel type MOM for $GF(2^m)$ is constructed by m basic circuits of the same structure, for each of which has 2m inputs and one output. (See Subsection 3.2.2.) Multiplications in

$GF(2^{m/(2^i)})$ $(1 \leqq i \leqq k)$ can be carried out by only $m/2^i$ basic circuits of the parallel type MOM. (See Lemma 4.23.) The outputs of the multiplications in $GF(2^{m/(2^i)})$ $(1 \leqq i \leqq k)$ provides only $m/2^i$ bits, thus the outputs must be *transcribed* 2^i times and those transcripts must be *concatenated* to match the basic circuits of the parallel type MOM.

Let *SIZE* be the *circuit size* of the parallel type MOM for $GF(2^m)$ and let *DEPTH* be the *depth* of the parallel type MOM for $GF(2^m)$, i.e., maximum path of gates from the inputs to the outputs. Then the multiplications in $GF(2^{m/(2^i)})$ $(1 \leqq i \leqq k)$ can be carried out by the circuit of *size* SIZE/2^i and of *depth* DEPTH. On the other hand, transcriptions, concatenations and cyclic shifts have simple circuits of *size* m and of *depth* 1. The recursive algorithm for $GF(2^m)$ requires two multiplications in $GF(2^m)$ and in each of subfield of $GF(2^m)$, i.e., $GF(2^{m/2})$, $GF(2^{m/4}), \cdots, GF(2^8)$ and $GF(2^4)$ and (m-1) cyclic shifts over $GF(2)$. Thus the recursive algorithm for $GF(2^m)$ can be implemented by the circuit of *size* about 4·SIZE and of *depth* about k·DEPTH, where $m=2^k$. It is known, in general, that SIZE=$O(m^3)$ and DEPTH=$O(log\ m)$, thus the recursive algorithm for $GF(2^m)$ has *simultaneous size* $O(m^3)$ and *depth* $O((log\ m)^2)$). □

The following figure, FIGURE 4.24, illustrates the *Configuration of Type 1* for the recursive algorithm in $GF(2^{16})$.

Here we present two technical lemmas [Ito88b] on the generation of normal bases in $GF(2^m)$ (m:even) to derive the configuration of type 2 for the recursive algorithm for $GF(2^m)$.



k-c.s. : k cyclic shifts

MOM(n/m) : n basic circuits of parallel type MOM in $GF(2^m)$

FIGURE 4.24 Configuration of Type 1 for $GF(2^{16})$

LEMMA 4.25 [Ito88b] *Let* $\{a, a^2, a^{2^2}, \cdots, a^{2^{(m-1)}}\}$ *be a normal basis in* $GF(2^m)$ *(m:even) over* $GF(2)$. *Define* $b=a+a^{2^{(m/2)}}$, *then* $\{b, b^2, b^{2^2}, \cdots, b^{2^{(m/2-1)}}\}$ *constructs a normal basis in* $GF(2^{m/2})$ *over* $GF(2)$. □

Proof: For simplicity we prove the special case that m=8. The general case that m is even can be similarly proven. Let $\{a, a^2, a^{2^2}, \cdots, a^{2^7}\}$ be a normal basis in $GF(2^8)$ over $GF(2)$. Define $b=a+a^{16}$, then we have $b^{16}=(a+a^{16})^{16}=a^{16}+a^{256}=a^{16}+a=b$. This implies that $b \in GF(2^4)$. Furthermore define $b=a+a^{16}$, $b^2=a^2+a^{32}$, $b^4=a^4+a^{64}$ and $b^8=a^8+a^{128}$. It is easy to show that $\{b, b^2, b^4, b^6\}$ is *linearly independent* over $GF(2)$ by noting the linear independency of $[a, a^2, a^{2^2}, \cdots, a^{2^7}\}$ over $GF(2)$. □

LEMMA 4.26 [Ito88b] *Let* $\{b, b^2, b^{2^2}, \cdots, b^{2^{(m/2-1)}}\}$ *be a normal basis in* $GF(2^{m/2})$, *where m is even, over* $GF(2)$. *Then* $\{a, a^2, a^{2^2}, \cdots a^{2^{(m-1)}}\}$ *constructs a normal basis in* $GF(2^m)$ *over* $GF(2)$, *where "a" is one of the roots of* $x^{2^{(m/2)}}+x=b$. □

Proof: For simplicity we prove the special case that m=8. The general case that m is even can be similarly proven. Let $\{b, b^2, b^4, b^8\}$ be a normal basis in $GF(2^4)$. Define $x^{16}+x=b$ over $GF(2^4)$. Let "a" be one of the roots of $x^{16}+x=b$, then we have

$$a^{16}+a=b. \qquad (4.2)$$

Noting that $b \in GF(2^4)$, $a^{256}=(a^{16})^{16}=(a+b)^{16}=a^{16}+b^{16}=a^{16}+b=a$. This implies that $a \in GF(2^8)$. (See Lemma 4.19.) Furthermore we can easily prove that $a \notin GF(2^4)$, because if $a \in GF(2^4)$ then we have $a^{16}+a=a+a=0=b$, and this contradicts the assumption that $\{b, b^2, b^4, b^8\}$ constructs a normal basis in $GF(2^4)$.

Here we will show that $\{a, a^2, a^{2^2}, \cdots, a^{2^7}\}$ constructs a normal basis in $GF(2^8)$ *by contradiction*. Assume that there exists a vector $c=(c_0, c_1, \cdots, c_7) \neq 0$ over $GF(2)$ such that

$$c_0 a + c_1 a^2 + c_2 a^{2^2} + \cdots + c_7 a^{2^7} = 0. \qquad (4.3)$$

Substituting Eq.(4.2) to Eq.(4.3),

$$\begin{aligned} & c_0 a + c_1 a^2 + + c_2 a^{2^2} + \cdots + c_7 a^{2^7} \\ & = c_0 a + c_1 a^2 + c_2 a^4 + c_3 a^8 \\ & \quad + c_4 (a+b) + c_5 (a^2 + b^2) + c_6 (a^4 + b^4) + c_7 (a^8 + b^8) \\ & = (c_0 + c_4) a + (c_1 + c_5) a^2 + (c_2 + c_6) a^4 + (c_3 + c_7) a^8 \\ & \quad + c_4 b + c_5 b^2 + c_6 b^4 + c_7 b^8 \\ & = d_0 a + d_1 a^2 + d_2 a^4 + d_3 a^8 + e = 0, \qquad (4.4) \end{aligned}$$

where $e = c_4 b + c_5 b^2 + c_6 b^4 + c_7 b^8$, $d_0 = c_0 + c_4$, $d_1 = c_1 + c_5$, $d_2 = c_2 + c_6$ and $d_3 = c_3 + c_7$. By Eq.(4.4), we have $d_0 a + d_1 a^2 + d_2 a^4 + d_3 a^8 = e \in GF(2^4)$, and thus we can show that

$$\begin{aligned} e^{16} & = (d_0 a + d_1 a^2 + d_2 a^4 + d_3 a^8)^{16} \\ & = d_0 a^{16} + d_1 a^{32} + d_2 a^{64} + d_3 a^{128} \\ & = e = d_0 a + d_1 a^2 + d_2 a^4 + d_3 a^8. \end{aligned}$$

This equation implies that

$$\begin{aligned} & d_0 (a + a^{16}) + d_1 (a^2 + a^{32}) + d_2 (a^4 + a^{64}) + d_3 (a^8 + a^{128}) \\ & = 0 \\ & = d_0 b + d_1 b^2 + d_2 b^4 + d_3 b^8. \end{aligned}$$

(See Eq.(4.2).) Noting that $\{b, b^2, b^4, b^8\}$ is linearly independent over $GF(2)$, we have $d_0 = d_1 = d_2 = d_3 = 0$, thus $c_0 = c_4$, $c_1 = c_5$, $c_2 = c_6$ and $c_3 = c_7$. Recalling Eq.(4.4), we have $c_4 b + c_5 b^2 + c_6 b^4 + c_7 b^8 = 0$. By the assumption that $\{b, b^2, b^4, b^5\}$ constructs a normal basis in $GF(2^4)$, hence $c_4 = c_5 = c_6 = c_7 = 0$ and it follows that $c=0$. (Note that $c_0 = c_4$, $c_1 = c_5$, $c_2 = c_6$ and $c_3 = c_7$.) This contradicts the assumption that $c \neq 0$. Hence we can conclude that $\{a, a^2, a^{2^2}, \cdots, a^{2^7}\}$ constructs a normal basis in $GF(2^8)$ over $GF(2)$. The *general* proof can be shown in the similar way. □

CONFIGURATION OF TYPE 2:

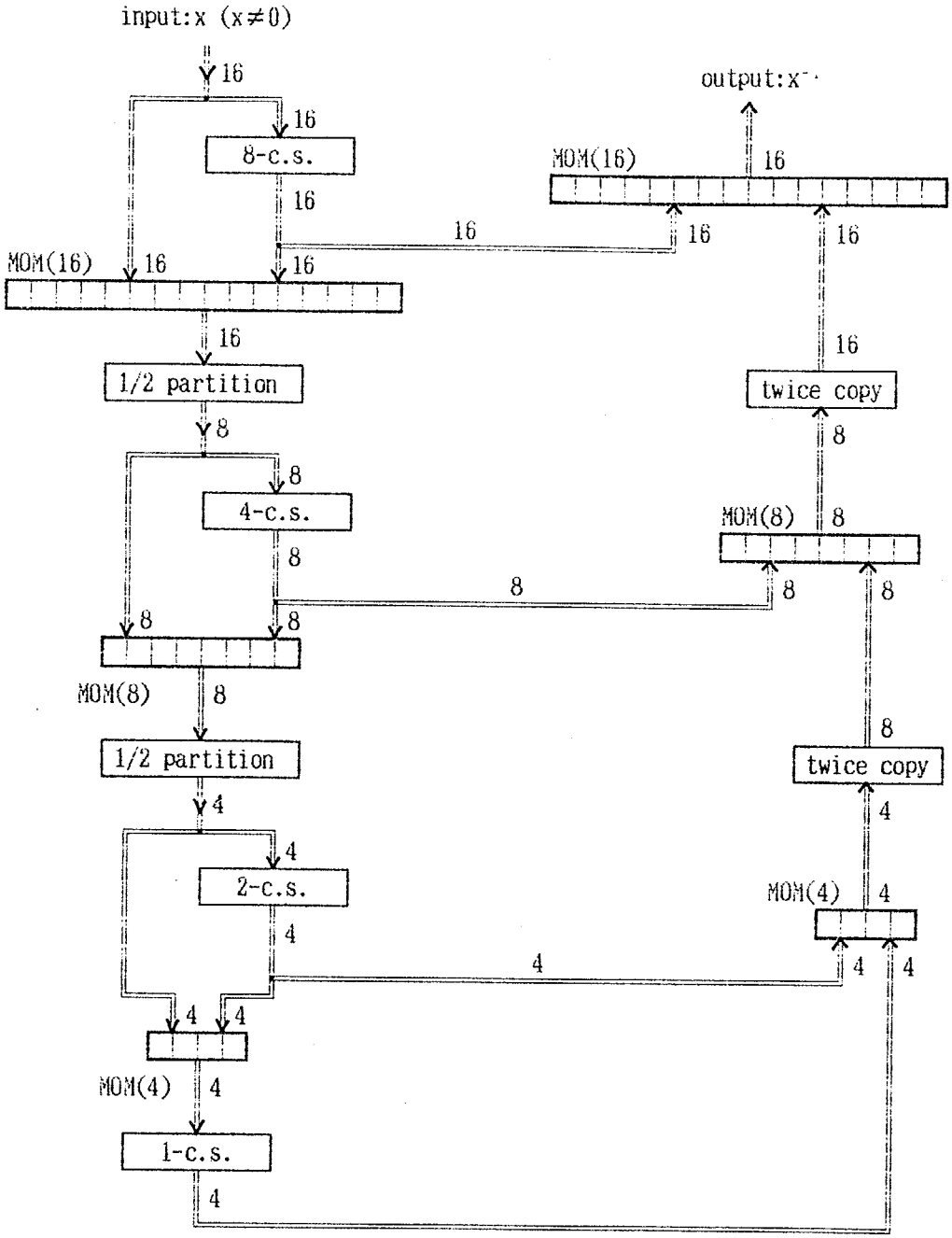
Prepare a parallel type MOM for $GF(2^m)$ $(m=2^k)$, which is designed by a normal basis $\{a, a^2, a^{2^2}, \cdots, a^{2^{(m-1)}}\}$ in $GF(2^m)$. (See Subsection 3.2.2.) Recalling Lemma 4.25, we can define a normal basis $\{b, b^2, b^{2^2}, \cdots, b^{2^{(m/2-1)}}\}$ in $GF(2^{m/2})$, where $b=a+a^{2^{(m/2)}}$, and can design a parallel type MOM for $GF(2^{m/2})$ based on the normal bases in $GF(2^{m/2})$. Furthermore we can analogously design parallel type MOM's for $GF(2^{m/4})$, $GF(2^{m/8})$, $\cdots, GF(2^4)$, applying the similar methodology. For multiplications of $x \in GF(2^{m/\{2^{(i-1)}\}})$ and $y \in GF(2^{m/(2^i)})$ $(1 \leqq i \leqq k)$, the same techniques with the *Configuration of Type 1*, i.e., the *transcriptions* of outputs and the *concatenations* of the transcribed outputs, etc., are required. Let SIZE(m) be the *circuit size* of a parallel type MOM for $GF(2^m)$ and let DEPTH(m) be the *depth* of the parallel type MOM. Note that $SIZE(m) \leqq s_m$ for some constant s_m (See Lemma 3.10.) and $DEPTH(m) \leqq d_m(log\ m)$ for some constant d_m. Furthermore transcriptions, concatenations and cyclic shifts have simple circuits of *size* $O(m)$ and of *depth* 1. Thus the recursive algorithm for $GF(2^m)$ can be implemented by the circuit of *size*

$$s_m m^3 + s_{m/2}(m/2)^3 + s_{m/4}(m/4)^3 + \cdots + s_4 4^3 \leqq sm^3$$

for some constant s and of *depth*

$$d_m(log\ m) + d_{m/2}\{log\ (m/2)\} + d_{m/4}\{log\ (m/4)\} + \cdots$$
$$\cdots + d_4(log\ 4) \leqq d(log\ m)^2$$

for some constant d. Hence we can conclude that the recursive algorithm for $GF(2^m)(m=2^k)$ has *simultaneous size* $O(m^3)$ and *depth* $O((log\ m))^2)$. □



k-c.s. : k cyclic shifts

MOM(m) : parallel type MOM in $GF(2^m)$

FIGURE 4.27 Configuration of Type 2 for $GF(2^{16})$

FIGURE 4.27 illustrates the *Configuration of Type 2* for the recursive algorithm in $GF(2^{16})$. Note that the configuration of each multiplier in Type 2 is different from that in Type 1.

4.5. CONCLUSION

In this Chapter, we have developed *sequential* and *recursive* type fast algorithm for computing multiplicative inverses in finite fields, employing normal bases.

The sequential type fast algorithm in finite fields, especially in $GF(2^m)$, iterates multiplications in $GF(2^m)$ and cyclic shifts over $GF(2)$ to compute multiplicative inverses in $GF(2^m)$. Here we restate the sequential type fast algorithm for $GF(2^m)$, which is formulated in Theorem 4.13.

THEOREM 4.13 [IT87a,Va87,IT88a] *Let $x \in GF(2^m)$ $(x \neq 0)$. Then there exists a sequential type fast algorithm for computing x^{-1}, which requires $\{[log\ (m-1)]+Hw(m-1)-1\}$ multiplications in $GF(2^m)$ and $(m-1)$ cyclic shifts over $GF(2)$, where $[x]$ denotes a maximum integer $\leq x$ and $Hw(i)$ denotes the Hamming weight of binary representation of i.* □

Note that $\{[log\ (m-1)]+Hw(m-1)-1\} \leq 2[log\ (m-1)]$, thus the sequential type fast algorithm for $GF(2^m)$ requires considerably less multiplications than the *Wang's* algorithm [WTSDOR85], which requires (m-2) multiplications in $GF(2^m)$. The Wang's algorithm, in addition, *alternatively* iterates a multiplication in $GF(2^m)$ and a cyclic shift over $GF(2)$, while the sequential type fast algorithm for $GF(2^m)$ *alternatively* iterates a multiplications in

$GF(2^m)$ and 2^i cyclic shifts over $GF(2)$. Thus, if the special machines are available to perform 2^i cyclic shifts over $GF(2)$ in *one machine cycle*, then (m-1) cyclic shifts over $GF(2)$ will be reduced to $\{[log\ (m-1)+H_w(m-1)\}$ cyclic shifts over $GF(2)$.

We extended the sequential type fast algorithm for $GF(2^m)$ to that for $GF(q^m)$ $(q=2^n)$. Here we also restate the sequential type fast algorithm for $GF(q^m)$ $(q=2^n)$, which is formulated in Theorem 4.18.

THEOREM 4.18 [IT87a,IT88a] *Let $x \in GF(q^m)$ $(q=2^n, x \neq 0)$. Then there exists an algorithm for computing x^{-1}, which requires $\{[log\ (m-1)]+H_w(m-1)\}$ multiplications in $GF(q^m)$, $\{[log\ (n-1)]+H_w(n-1)-1\}$ multiplications in $GF(q)$ $(q=2^n)$, $(m-1)$ cyclic shifts over $GF(q)$ and $(n-1)$ cyclic shifts over $GF(2)$, where $[x]$ and $H_w(i)$ denote the same symbols with Theorem 4.13, respectively.* □

The sequential type fast algorithm in Theorem 4.18 also *alternatively* iterates a multiplications in $GF(q^m)$ and 2^i cyclic shifts over $GF(q)$ or a multiplications in $GF(q)=GF(2^n)$ and 2^j cyclic shifts over $GF(2)$. Thus, if the special machines are available to perform 2^i cyclic shifts over $GF(q)$ and 2^j cyclic shifts over $GF(2)$ in one machine cycle, then (m-1) cyclic shifts over $GF(q)$ will be reduced to $\{[log\ (m-1)]+H_w(m-1)\}$ cyclic shifts over $GF(q)$ and (n-1) cyclic shifts over $GF(2)$ will also be reduced to $\{[log\ (n-1)]+H_w(n-1)\}$ cyclic shifts over $GF(2)$. It is, needless to say, possible to develop a sequential type fast algorithm for computing multiplicative inverses in $GF(q^m)$ $(q=p^n$, p:odd prime) applying the similar idea described above.

We have also shown the recursive type fast algorithm for computing multiplicative inverses in $GF(2^m)$ $(m=2^k)$. The studies on the recursive type fast algorithm is motivated by the reduction of the running cost for multiplications in $GF(2^m)$ $(m=2^k)$. (The significance of $GF(2^m)$ $(m=2^k)$ can be found in several digital audio instruments such as *CD* (*Compact Disc*) *player* and *DAT* (*Digital Audio Tape-recorder*). The recursive type fast algorithm for $GF(2^m)$ $(m=2^k)$ also iterates multiplications in $GF(2^m)$ and cyclic shifts over $GF(2)$, however, it reduces the running cost of multiplications in $GF(2^m)$ by introducing the subfields of $GF(2^m)$ $(m=2^k)$, i.e., $GF(2^{m/2}), GF(2^{m/4}), \cdots, GF(2^8)$ and $GF(2^4)$.

Here we restate the recursive type fast algorithm for computing multiplicative inverses in $GF(2^m)$ $(m=2^k)$, which is formulated in Theorem 4.20.

THEOREM 4.20 [IT88b,IT88d] *There exists a recursive algorithm for computing multiplicative inverses in $GF(2^m)$ $(m=2^k)$, which requires $(m-1)$ cyclic shifts over $GF(2)$ and two multiplications in $GF(2^m)$ and in each of subfield of $GF(2^m)$, i.e., $GF(2^{m/2}), GF(2^{m/4}), \cdots, GF(2^8)$ and $GF(2^4)$.* □

The recursive type fast algorithm for multiplicative inverses in $GF(2^m)$ $(m=2^k)$ can be regarded as an algorithm defined *constructively* as follows: The algorithm for multiplicative inverses in $GF(2^m)$ $(m=2^k)$ includes that in $GF(2^{m/2})$, furthermore the algorithm for multiplicative inverses in $GF(2^{m/2})$ includes that in $GF(2^{m/4}), \cdots\cdots$, the algorithm for multiplicative inverses in $GF(2^8)$ finally includes that in $GF(2^4)$. Hence the above hierarchical structure of the recursive type algorithm provides

simple circuit configurations of the algorithm. We have presented two kinds of configurations of the recursive type algorithm. (See *Configuration of Type 1* and *Type 2*.) Note that both configurations are essentially the same except that the configuration of multipliers for the subfields of $GF(2^m)$ $(m=2^k)$ in those configurations. (See FIGURE 4.24 and 4.27.) Thus both configurations of the recursive type fast algorithm for $GF(2^m)$ $(m=2^k)$, *Configuration of Type 1* and *Type 2*, have *simultaneous size* $O(m^3)$ and *depth* $O((log\ m)^2)$. The related works for multiplicative inverses in finite fields can be found in [MK88].

Further studies are required to give the maximum lower bound for the number of multiplications for computing multiplicative inverses in $GF(2^m)$. In addition, for the circuit configurations of the recursive type fast algorithm in $GF(2^m)$ $(m=2^k)$, the constructive methods must be developed to provide simple circuits of *simultaneous* $O(m^2)$ and *depth* $O((log\ m)^2)$.

CHAPTER V :

EFFICIENT PROBABILISTIC ALGORITHMS FOR QUADRATIC EQUATIONS OVER FINITE FIELDS

This Chapter presents efficient probabilistic algorithms for solving quadratic equations over finite fields, $GF(2^m)$ and $GF(p)$ (p:odd prime) [Ito87,Ito88a]. The efficient probabilistic algorithms developed in this Chapter are applicable to the decryption procedures of some public-key cryptosystems [Ra79,Wil80,Wil85,KIT87,KIT88a,KIT88b]. The application of the probabilistic algorithm to those public-key cryptosystems will be considered in *CHAPTER VII*.

5.1 INTRODUCTION

In this Chapter, we study efficient probabilistic algorithms for solving quadratic equations over $GF(p)$ (p:odd prime) and $GF(2^m)$ [Ito87,Ito88a]. (The probabilistic algorithm for $GF(p^m)$ ($m \geq 2$, p:odd prime) will be studied in *CHAPTER VI* from technical reasons.)

A probabilistic algorithm [Ra76] is an algorithm including procedures of random choice, e.g., *coin tosses*. More formally, the probabilistic algorithm can be regarded as a *deterministic* algorithm $A[\cdot;\cdot]$ which outputs $y=A[x;r]$, where x is an *input* to be solved, r is a *randomly chosen input* and y is an *output* for x and r. Because of its randomness, the probabilistic algorithm $A[\cdot;\cdot]$ does not necessarily output correct answers, i.e., the

algorithm A[·;·] outputs correct and incorrect answers depending on the choice of random inputs r's. If Pa, the probability that the probabilistic algorithm A[·;·] outputs correct answers, is *non-negligible*, e.g., Pa ≧ 1/2, then the probabilistic algorithm gives a correct answer with overwhelming probability by performing A[·;·] *independently* for many different random inputs r's and taking majority of the answers.

A class of problems, which are accepted by such probabilistic algorithms in *polynomial time*, is defined by *RP* or *BPP* [Gil77,GJ79] in the *Structural Computational Complexity Theory*. Several instances, e.g., primality test [SS78,Ra80b], factorization of polynomials over finite fields [Ra80a], etc., are known to be in *R*. More theoretical related works for *R* or *BPP* can be found in [La83].

In 1980, Rabin proposed probabilistic algorithms for factoring polynomials over *GF*(p) (p:odd prime) and *GF*(2^m) [Ra80a]. The remarkable idea of the probabilistic algorithm for *GF*(p) is to notice the uniform distribution of quadratic residuous element c∈ *GF*(p) (p:odd prime) with probability 1/2. In addition, the central idea of the probabilistic algorithm for *GF*(2^m) is to notice the uniform distribution of element c∈ *GF*(2^m) satisfying Tr(c)=1 with probability 1/2. By utilizing the above inherent feature for *GF*(p) (*GF*(2^m)), a probabilistic polynomial factorization algorithm over *GF*(p) (*GF*(2^m)) can be derived, in which the complete factorization of the given polynomial over *GF*(p) (*GF*(2^m)) is obtained probabilistically in polynomial time. More precisely, the probabilistic polynomial factorization algo-

rithm over $GF(p)$ ($GF(2^m)$) iterates the procedure for randomly chosen inputs r's, until one of the factors of the given polynomial over $GF(p)$ ($GF(2^m)$) can be obtained. The probabilistic polynomial factorization algorithm over $GF(p)$ ($GF(2^m)$) succeeds with probability at least 1/2 for each trial, thus the average number of trials is 2 to output one of the factors of the given polynomial over $GF(p)$ ($GF(2^m)$). Hence the probabilistic polynomial factorization algorithm over $GF(p)$ ($GF(2^m)$) terminates probabilistically in polynomial time to output the complete factorization of the given polynomial over $GF(p)$ ($GF(2^m)$). The above probabilistic polynomial factorization algorithms are definitely applicable to quadratic equations over $GF(p)$ (p:odd prime) and $GF(2^m)$.

On the other hand, some researchers have developed public-key cryptosystems, in which inverting ciphertexts is equivalent to factoring the product of two large prime numbers N, where N=pq (p,q:odd primes) [Ra79,Wil80,Wil85,KIT87,KIT88a,KIT88b]. (Note that Rabin was the first person who had pointed out the equivalence between factoring a composite number N and solving quadratic equations over Z_N. Those public-key cryptosystems are not only of theoretical interest but also of practical importance, because they are *provably secure* under the assumption that factoring is *hard*.) In those public-key cryptosystems, the encryption procedures are defined by quadratic function over Z_N, where N=pq (p,q:odd primes), and the decryption procedures are carried out by solving quadratic equations over $GF(p)$ and $GF(q)$.

Hence it is important in a practical sense to develop ef-

ficient algorithms for solving quadratic equations over $GF(p)$ (p:odd prime). (The development of such algorithms enables us to establish *secure* and *efficient* public-key cryptosystems.) The studies in this Chapter are motivated by such a requirement.

This Chapter develops efficient probabilistic algorithm for solving quadratic equations over $GF(p)$ (p:odd prime) and $GF(2^m)$ [Ito87,Ito88a]. The core idea to derive efficient probabilistic algorithm for $GF(p)$ and $GF(2^m)$ is simple, however, it presents an uniformly efficient probabilistic algorithm .

The organization of this Chapter is as follows:

Section 5.2 includes two subsections; Subsection 5.2.1 presents a probabilistic algorithm for quadratic equations over $GF(p)$ applying the Rabin's probabilistic polynomial factorization algorithm, and Subsection 5.2.2 develops an efficient version for it. Section 5.3 also includes two Subsections; Subsection 5.3.1 shows a probabilistic algorithm for quadratic equations over $GF(2^m)$ applying the Rabin's probabilistic polynomial factorization algorithm, and Subsection 5.3.2 proposes an efficient version for it. Section 5.4 finally summarizes the results in this Chapter and gives conclusions, remarks and open problems.

5.2 EFFICIENT PROBABILISTIC ALGORITHM OVER $GF(p)$

This Section develops an efficient probabilistic algorithm for solving quadratic equations over $GF(p)$ (p:odd prime) [Ito87, Ito88a]. The central idea to derive the efficient algorithm is

very simple, however, it enables us to establish uniformly efficient algorithm embedding a verification step to the Rabin's probabilistic polynomial factorization algorithm [Ra80a].

5.2.1 RABIN'S POLYNOMIAL FACTORIZATION ALGORITHM OVER *GF*(p)

The primary interest in this Section is to solve quadratic equation over *GF*(p) (p:odd prime). Thus we concentrate on the factorization of polynomials of degree 2 over *GF*(p) by Rabin's probabilistic algorithm [Ra80a]. To show the Rabin's probabilistic algorithm, we present some mathematical definitions and lemmas as preliminaries.

DEFINITION 5.1 [Kra86] *Call an x quadratic residue (mod m), if* $x=y^2$ *(mod m) for some y; otherwise x is called quadratic non-residue (mod m).* □

Remark 5.2 Throughout this Section, the set of all *quadratic (non) residues* (mod m) is denoted by QR_m (QNR_m). □

DEFINITION 5.3 [Kra86] *Let p be an odd prime number and let* $x \not\equiv 0$ *(mod p). Define the following symbol such that*

$(x/p)=1$, $x \in QR_p$,

$(x/p)=-1$, $x \in QNR_p$.

Here (x/p) is called ***Legendre symbol*** *of x mod p.* □

DEFINITION 5.4 [Kra86] *Let* $m=p_1 p_2 \cdots p_r$, *where* p_i $(i \leqq i \leqq r)$ *are primes, not necessarily distinct. Then* ***Jacobi symbol*** *(x/m) is defined by* $(x/m)=(x/p_1)(x/p_2)\cdots(x/p_r)$, *where* (x/p_i) $(1 \leqq i \leqq r)$ *denote Legendre symbol.* □

Here we show the following two fundamental lemmas for the evaluation of Jacobi symbol.

LEMMA 5.5 [Kra86] *Let x and y be coprime to m.*

F1. If $x=y \pmod m$, then $(x/m)=(y/m)$;

F2. $(xy/p)=(x/p)(y/p)$.

□

LEMMA 5.6 [Kra86] *Let m and n be positive odd integers.*

F1. $(m/n)=(-1)^{(m-1)(n-1)/4}(n/m)$;

F2. $(-1/m)=(-1)^{(m-1)/2}$;

F3. $(2/m)=(-1)^{(m^2-1)/8}$.

□

Remark 5.7 Notice that Lemma 5.6, especially F1, provides an efficient algorithm for evaluating (x/p). □

The following lemma categorizes quadratic equations over *GF*(p) by simple criteria.

LEMMA 5.8 [Ito87,Ito88a] *Let $f(x)=x^2+ax+b$ be a quadratic polynomial over $GF(p)$, i.e., $a,b\in GF(p)$. Then we have*

T1. $D=0 \pmod p$ iff $f(x)=(x+a/2)^2 \pmod p$;

T2. $D\in QNR_p$ iff $f(x)$ is irreducible over $GF(p)$;

T3. $D\in QR_p$ iff $f(x)=(x-u)(x-v)$, $u,v\in GF(p)$ $(u\neq v)$,

where $D=a^2-4b \pmod p$. □

It is very easy to decide whether $D\in QR_p$ or $D\in QNR_p$ by Lemma 5.5 and 5.6 Hence we concentrate on the case that $D\in QR_p$, i.e., f(x)=0 has two distinct rootse $\in GF(p)$. Here we show two essential lemmas to derive probabilistic algorithm for solving quadratic equations over *GF*(p) (p:odd prime).

LEMMA 5.9 [Sch86] *For $^\forall s\in GF(p)$,*

$s\in QR_p$ iff $s^{(p-1)/2}-1=0 \pmod p$.

This implies that $x^{(p-1)/2}-1=\prod_{s\in QR_p}(x-s) \pmod p$.

On the other hand, for $\forall t \in GF(p)$,

$t \in QNR_p$ *iff* $t^{(p-1)/2}+1=0 \pmod p$.

This implies that $x^{(p-1)/2}+1=\prod_{t \in QNR_p}(x-t) \pmod p$. □

Remark 5.10 The above Lemma is well-known as ***Euler's Criterion*** in *GF*(p) [Sch86,Kra86]. □

LEMMA 5.11 [Ra80a] *For* $\forall a \in GF(p)$, *define the following two sets such that*

$R_a=\{i \mid (a-i) \in QR_p,\ i \in GF(p)-\{a\}\}$,

$NR_a=\{j \mid (a-j) \in QNR_p,\ j \in GF(p)-\{a\}\}$.

Then we have $|R_a|=|NR_a|=(p-1)/2$, *where* $|A|$ *denotes the number of elements in a set A.* □

Assume that $f(x)=x^2+ax+b$ and $D \in QR_p$, where $a,b \in GF(p)$ and $D=a^2-4b \pmod p$. Then by noting Lemma 5.8, f(x) can be factored into the product of two distinct linear factors such that $f(x)=(x-u)(x-v)$, where $u,v \in GF(p)$ $(u \neq v)$. Hence for $\forall c \in GF(p)$, we have $f(x+c)=\{x-(u-c)\}\{x-(v-c)\}$. TABLE 5.12 classifies *quadratic residuosity* of (u-c) and (v-c).

TABLE 5.12 Classification of (u-c) and (v-c)

	u-c	v-c	probability
C1	QR_p	QR_p	1/4
C2	QR_p	QNR_p	1/4
C3	QNR_p	QR_p	1/4
C4	QNR_p	QNR_p	1/4

It is easy to see that the probability for each class-

ification is 1/4, respectively. (See Lemma 5.11.) Recalling Lemma 5.9, for each classification, GCD{f(x+c),$x^{(p-1)/2}-1$} gives the factorization of f(x+c) as in TABLE 5.13.

TABLE 5.13 Factorization of f(x+c)

	GCD{f(x+c),$x^{(p-1)/2}-1$}	probability
C1	f(x+c)	1/4
C2	x-(u-c)	1/4
C3	x-(v-c)	1/4
C4	1	1/4

Observing TABLE 5.13, one of the roots of f(x)=0 can be found by iterating g(x){=GCD{f(x+c),$x^{(p-1)/2}-1$}-c} for different c's$\in$ *GF*(p) until *deg* g(x)=1. Furthermore the probability that *deg* g(x)=1 is 1/2 for each trial, thus AT will be

$$AT=\sum_{k>0} k\cdot(1/2)^k=2,$$

where AT is the average number of trials to be required until *deg* g(x)=1. FIGURE 5.14 describes the Rabin's probabilistic algorithm (denoted *RPA*) over *GF*(p).

5.2.2 EFFICIENT PROBABILISTIC ALGORITHM OVER *GF*(p)

In this subsection, we develop an efficient probabilistic algorithm for solving quadratic equations over *GF*(p) (p:odd prime). The following Theorem provides a necessary and sufficient condition for c$\in$ *GF*(p) to output g(x) of degree 1, where g(x)=GCD{f(x+c),$x^{(p-1)/2}-1$}-c.

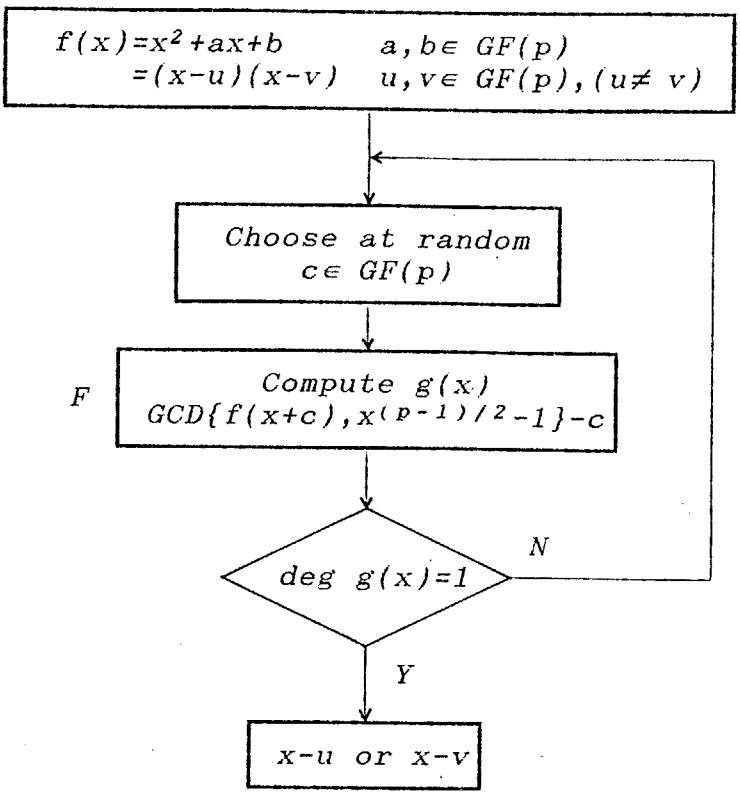


FIGURE 5.14 Rabin's Probabilistic Algorithm *RPA* OVER *GF*(p)

THEOREM 5.15 [Ito87,Ito88a] *Let f(x) be quadratic polynomial over GF(p) such that* $f(x)=x^2+ax+b=(x-u)(x-v)$, $u,v\in GF(p)$ $(u\neq v)$. *For* $^{\exists}c\in GF(p)$, $g(x)=GCD\{f(x+c),x^{(p-1)/2}-1\}-c=x-u$ *or* $x-v$ *iff* $(f(c)/p)=-1$. □

Proof: Observing TABLE 5.13, *deg* g(x)=1 iff C2 or C3. Thus it follows that ((u-c)/p)((v-c)/p)=((u-c)(v-c)/p)=((c-u)(c-v)/p) =(f(c)/p)=-1. (See TABLE 5.12 and F2 in Lemma 5.5.) □

It is easy to see that the probability for $^{\forall}c\in GF(p)$ to be (f(c)/p)=-1 is 1/2. Furthermore the evaluation of (f(c)/p) can be efficiently carried out (See Lemma 5.6.), thus we have the following efficient probabilistic algorithm (denoted *EPA*)

for solving quadratic equations over *GF*(p) as in FIGURE 5.16.

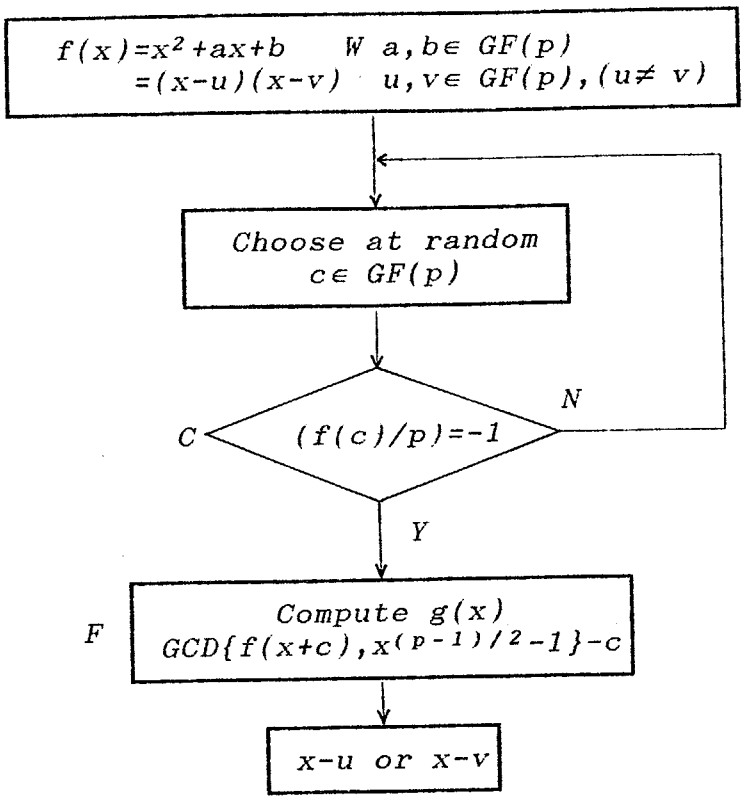


FIGURE 5.16 Efficient Probabilistic Algorithm *EPA* OVER *GF*(p)

Compare FIGURE 5.14 and 5.16. The Rabin's probabilistic algorithm *RPA* (FIGURE 5.14) carries out the procedure *F* twice *on the average* to factor f(x), while the efficient probabilistic algorithm *EPA* (FIGURE 5.16) carries out the procedure *C* twice *on the average* and *F* just once to factor f(x). The running time of the procedure *F* is much larger than that of the procedure *C*, thus the total running time of *EPA* will be considerably reduced. In order to show the *availability* of *EPA*, we quantatively analyzes the performance of *RPA* and *EPA* in the following.

Let TF and TC be the running time of the procedure F and C, respectively. In addition, let $T_{RPA}(k)$ and $T_{EPA}(k)$ be the running time of *RPA* and *EPA* for the number of trials k, respectively. Then we have

$$T_{RPA}(k)=TF\cdot k, \tag{5.1}$$

$$T_{EPA}(k)=TC\cdot k+TF. \tag{5.2}$$

On the other hand, the probability of success in both *RPA* and *EPA* is 1/2 for each trial, thus it follows that

$$AT_{RPA}=\Sigma_{k>0}\ T_{RPA}(k)\cdot(1/2)^k$$

$$=TF\cdot\Sigma_{k>0}\ k\cdot(1/2)^k=2\cdot TF, \tag{5.3}$$

$$AT_{EPA}=\Sigma_{k>0}\ T_{EPA}(k)\cdot(1/2)^k$$

$$=TF\cdot\Sigma_{k>0}\ (1/2)^k + TC\cdot\Sigma_{k>0}\ k\cdot(1/2)^k$$

$$=TF+2\cdot TC, \tag{5.4}$$

where AT_{RPA} and AT_{EPA} denote the *average* running time of *RPA* and *EPA*, respectively. By computer simulation, the relation that TC=TF/5 is verified. Substituting this relation to Eq.(5.4), we have $AT_{EPA}=1.4\cdot TF$. Thus we can conclude that the *average* running time of *EPA* is about 70% of that of *RPA*. In addition, substituting the relation that TC=TF/5 to Eq.(5.2), we have

$$T_{EPA}(k)=TF+(TF/5)\cdot k. \tag{5.5}$$

Observing Eq.(5.1) and (5.5), $T_{RPA}(k)$ grows proportionally to k, while $T_{EPA}(k)$ grows proportionally to k/5. Hence *EPA* provides uniform efficiency for the number of trials k.

Remark 5.17 Let k=5. Then we have $T_{RPA}(5)=5\cdot TF$ and $T_{EPA}(5)=2\cdot TF$. Thus $T_{RPA}(5)$ is about 40% of $T_{EPA}(5)$. For larger k, e.g., k=10, the difference between $T_{RPA}(k)$ and $T_{EPA}(k)$ *enormously* grows! Hence *EPA* provides uniform ef-

ficiency for the number of trials k. □

5.3 EFFICIENT PROBABILISTIC ALGORITHM OVER $GF(2^m)$

This Section proposes an efficient probabilistic algorithm for solving quadratic equations over $GF(2^m)$ [Ito87,Ito88a]. The core idea to derive the efficient algorithm is very similar with that for the case of $GF(p)$ (See Section 5.2.), and it also enables us to establish uniformly efficient algorithm.

5.3.1 RABIN'S POLYNOMIAL FACTORIZATION ALGORITHM OVER $GF(2^m)$

The primary interest in this Section is also to solve quadratic equations over $GF(2^m)$. Thus we concentrate on the factorization of polynomials of degree 2 over $GF(2^m)$ by Rabin's probabilistic algorithm [Ra80a]. In the case of $GF(p)$, quadratic residuosity presents an useful criterion to derive an efficient probabilistic algorithm. In the case of $GF(2^m)$, however, quadratic residuosity provides no available tools to derive an efficient probabilistic algorithm, because $\forall c \in GF(2^m)$ are quadratic residuous. Here we give some useful lemmas to show the Rabin's polynomial factorization algorithm over $GF(2^m)$.

LEMMA 5.18 [Ra80a] *Let $f(x)=x^2+ax+b$ be a quadratic polynomial over $GF(2^m)$, i.e., $a,b \in GF(2^m)$. Then we have*

T1. $a=0$ iff $f(x)=(x+b^{2^{(m-1)}})^2$;

T2. $Tr(a^{-2}b)=1$ iff $f(x)$ is irreducible over $GF(2^m)$;

T3. $Tr(a^{-2}b)=0$ iff $f(x)=(x+u)(x+v)$, $u,v \in GF(2^m)(u \neq v)$,

where $Tr(x)=x+x^2+x^{2^2}+\cdots+x^{2^{(m-1)}}$. □

By Lemma 5.18, it is clear that T1 and T2 are *trivial* cases. Hence we concentrate on the case that $Tr(a^{-2}b)=0$, i.e., $f(x)=0$ has two distinct rootse $\in GF(2^m)$. Here we show two essential lemmas to develop a probabilistic algorithm for solving quadratic equations over $GF(2^m)$.

LEMMA 5.19 [Ra80a] *Let* $Tr(x)=x+x^2+x^{2^2}+\cdots+x^{2^{(m-1)}}$. *Then* $Tr(x)=\prod_{Tr(s)=0}(x-s)$ *and* $Tr(x)+1=\prod_{Tr(t)=1}(x-t)$. □

LEMMA 5.20 [Ra80a] *For* $\forall a \in GF(2^m)$ $(a\neq 0)$, *define the sets such that* $S_0(a)=\{c \mid Tr(ca)=0\}$ *and* $S_1(a)=\{c \mid Tr(ca)=1\}$. *Then we have* $|S_0(a)| = |S_1(a)| = 2^{m-1}$, *where* $|A|$ *denotes the number of elements in a set* A. □

Assume that $f(x)=x^2+ax+b$ and $Tr(a^{-2}b)=0$, where $a,b \in GF(2^m)$ and $a\neq 0$. By noting Lemma 5.18, $f(x)$ can be factored into the product of two distinct linear factors by $f(x)=(x+u)(x+v)$, where $u,v \in GF(2^m)$ $(u\neq v)$. Hence for $\forall c \in GF(2^m)$ $(c\neq 0)$, it follows that $f(c^{-1}x)=(c^{-1}x+u)(c^{-1}x+v)=c^{-2}(c+cu)(x+cv)$. The following table, TABLE 5.21, classifies $Tr(cu)$ and $Tr(cv)$.

TABLE 5.21 Classification of Tr(cu) and Tr(cv)

	Tr(cu)	Tr(cv)	probability
C1	0	0	1/4
C2	0	1	1/4
C3	1	0	1/4
C4	1	1	1/4

Similarly with the case for $GF(p)$ (See Section 5.2.), it

is easy to see that the probability for each classification is 1/4, respectively. (See Lemma 5.20.) Recalling Lemma 5.19, for each classification GCD{f($c^{-1}x$),Tr(x)} gives the factorization of f($c^{-1}x$) as in TABLE 5.22.

TABLE 5.22 Factorization of f($c^{-1}x$)

	GCD{f($c^{-1}x$),Tr(x)}	probability
C1	f($c^{-1}x$)	1/4
C2	x+cu	1/4
C3	x+cv	1/4
C4	1	1/4

Observing TABLE 5.22, one of the roots of f(x)=0 can be found by iterating g(x)=GCD{f($c^{-1}x$),Tr(x)} for different $c \in GF(2^m)$ ($c \neq 0$) until *deg* g(x)=1. Furthermore the probability that *deg* g(x)=1 is 1/2 for each trial, thus AT will be $AT=\Sigma_{k>0}\ k\cdot(1/2)^k=2$, where AT is the *average* number of trials to be required until *deg* g(x)=1. FIGURE 5.23 describes the Rabin's probabilistic algorithm (denoted *RPA'*) over $GF(2^m)$.

5.3.2 EFFICIENT PROBABILISTIC ALGORITHM OVER $GF(2^m)$

In thus subsection, we present an efficient probabilistic algorithm for solving quadratic equations over $GF(2^m)$. The following Theorem provides a necessary and sufficient condition for $c \in GF(2^m)$ to output g(x)={GCD{f($c^{-1}x$),Tr(x)}} of degree 1.

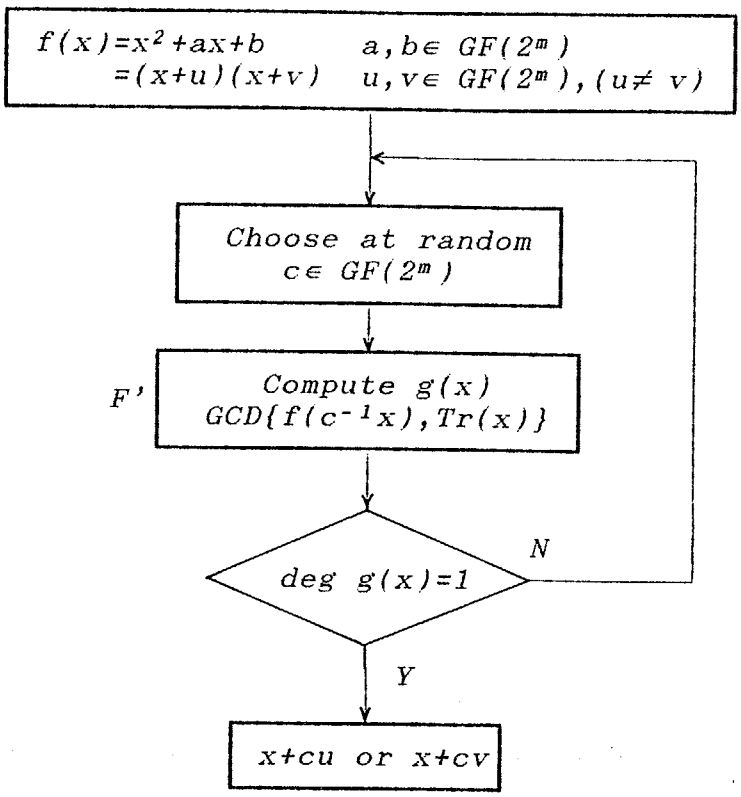


FIGURE 5.23 Rabin's Probabilistic Algorithm *RPA'* OVER $GF(2^m)$

THEOREM 5.24 [Ito87,Ito88a] *Let $f(x)$ be quadratic polynomial over $GF(2^m)$ such that $f(x)=x^2+ax+b=(x+u)(x+v)$, $u,v\in GF(2^m)$ $(u\neq v)$. For $\forall c\in GF(2^m)$ $(c\neq 0)$, $g(x)=GCD\{f(c^{-1}x),Tr(x)\}=x+cu$ or $x+cv$ iff $Tr(ca)=1$.* □

Proof: Observing TABLE 5.22, *deg* g(x)=1 iff C2 or C3. From the linearity of Tr(·) [Ber68,LN83,McE87] and the relation that a=u+v, we have Tr(ca)=Tr{c(u+v)}=Tr(cu+cv)=Tr(cu)+ Tr(cv)=1. □

It is not difficult tosee that the probability that for $\forall c\in GF(2^m)$ to be Tr(ca)=1 is 1/2. Furthermore the evaluation of Tr(ca) can be easily performed, thus we have the following efficient probabilistic algorithm (denoted *EPA'*) for solving

quadratic equations over $GF(2^m)$ as in FIGURE 5.25. (Note that *EPA'* are almost the same with *EPA* except the procedure *C'*.)

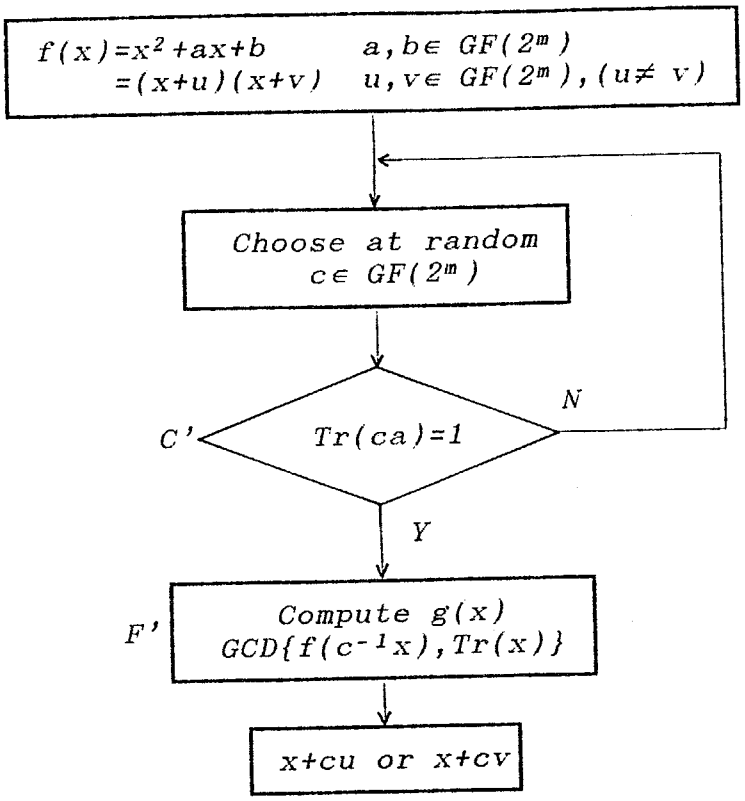


FIGURE 5.25 Efficient Probabilistic Algorithm *EPA'* OVER $GF(2^m)$

Compare FIGURE 5.23 and 5.25. The Rabin's probabilistic algorithm *RPA'* (FIGURE 5.23) carries out the procedure *F'* twice *on the average* to factor f(x), while the efficient probabilistic algorithm *EPA'* (FIGURE 5.25) carries out the procedure *C'* twice and *F'* just once to factor f(x). The running time of the procedure *F'* is larger than that of the procedure *C'*, thus the total running time of *EPA'* will be considerably reduced. In order to show the *availability* of *EPA'*, we also quantatively analyzes the performance of *RPA'* and *EPA'* in the following.

Let TF' and TC' be the running time of the procedure *F'* and *C'*, respectively. In addition, let $T_{RPA'}(k)$ and $T_{EPA'}(k)$ be the running time of *RPA'* and *EPA'* for the number of trials k, respectively. Then we have

$$T_{RPA'}(k)=TF'\cdot k, \tag{5.6}$$

$$T_{EPA'}(k)=TC'\cdot k+TF'. \tag{5.7}$$

On the other hand, the probability of success in both *RPA'* and *EPA'* is 1/2 for each trial, thus it follows that

$$AT_{RPA'}=\sum_{k>0} T_{RPA'}(k)\cdot(1/2)^k$$

$$=TF'\cdot\sum_{k>0} k\cdot(1/2)^k=2\cdot TF', \tag{5.8}$$

$$AT_{EPA'}=\sum_{k>0} T_{EPA'}(k)\cdot(1/2)^k$$

$$=TF'\cdot\sum_{k>0}(1/2)^k + TC'\cdot\sum_{k>0} k\cdot(1/2)^k$$

$$=TF'+2\cdot TC', \tag{5.9}$$

where $AT_{RPA'}$ and $AT_{EPA'}$ denote the *average* running time of *RPA'* and *EPA'*, respectively. By computer simulation, the relation that TC'=TF'/4 is verified. (Note that 4m multiplications in $GF(2^m)$ are required to compute $g(x)=GCD\{f(c^{-1}x),Tr(x)\}$, while m multiplications are required to compute Tr(ca). These facts theoretically support the result of the computer simulation that TC'=TF'/4.) Substituting this relation to Eq.(5.9), we have $AT_{EPA'}=1.5\cdot TF'$. Thus we can conclude that the *average* running time of *EPA'* is about 75% of that of *RPA'*. In addition, substituting the relation that TC'=TF'/4 to Eq.(5.7), we have

$$T_{EPA'}(k)=TF'+(TF'/4)\cdot k. \tag{5.10}$$

Observing Eq.(5.6) and (5.10), $T_{RPA'}(k)$ grows proportionally to k, while $T_{EPA'}(k)$ grows proportionally to k/4. Hence *EPA'* provides uniform efficiency for the number of trials k.

Remark 5.26 Let k=4. Then we have $T_{RPA'}(4)=4\cdot TF'$ and $T_{EPA'}(4)=2\cdot TF'$. Thus $T_{EPA'}(4)$ is about 45% of $T_{RPA'}(4)$. For larger k, e.g., k=10, the difference between $T_{RPA'}(k)$ and $T_{EPA'}(k)$ *enormously* grows! Hence *EPA'* also provides uniform efficiency for the number of trials k. □

5.4 CONCLUSION

This Chapter has proposed efficient probabilistic algorithms for solving quadratic equations over *GF*(p) (p:odd prime) and *GF*(2^m) [Ito87,Ito88a]. The efficient probabilistic algorithm over *GF*(p) (*GF*(2^m)) provides uniform efficiency embedding a simple verification procedure to the Rabin's probabilistic polynomial factorization algorithm over *GF*(p) (*GF*(2^m)) [Ra80a].

Here we restate the results obtained in this Chapter.

The following Theorem formulates the core idea to derive an efficient probabilistic algorithm for solving quadratic equations over *GF*(p) (p:odd prime).

THEOREM 5.15 [Ito87,Ito88a] *Let $f(x)$ be quadratic polynomial over $GF(p)$ such that $f(x)=x^2+ax+b=(x-u)(x-v)$, $u,v\in GF(p)$ $(u\neq v)$. For $^{\exists}c\in GF(p)$, $g(x)=GCD\{f(x+c),x^{(p-1)/2}-1\}-c=x-u$ or $x-v$ iff $(f(c)/p)=-1$.* □

The above Theorem enables us to establish the efficient probabilistic algorithm over *GF*(p) embedding the criterion that (f(c)/p)=-1 to the Rabin's probabilistic quadratic polynomial factorization algorithm over *GF*(p). Because of the efficient evaluation of (x/p), the criterion that (f(c)/p)=-1 provides

uniform efficiency for the number of trials.

The following Theorem similarly formulates the central idea to derive an efficient probabilistic algorithm for solving quadratic equations over $GF(2^m)$.

THEOREM 5.24 [Ito87,Ito88a] *Let $f(x)$ be quadratic polynomial over $GF(2^m)$ such that $f(x)=x^2+ax+b=(x+u)(x+v)$, $u,v\in GF(2^m)$ $(u\neq v)$. For $^\forall c\in GF(2^m)$ $(c\neq 0)$, $g(x)=GCD\{f(c^{-1}x),Tr(x)\}=x+cu$ or $x+cv$ iff $Tr(ca)=1$.* □

The above Theorem also enables us to establish the efficient probabilistic algorithm over $GF(2^m)$ embedding the criterion that Tr(ca)=1 to the Rabin's probabilistic quadratic polynomial factorization algorithm over $GF(2^m)$. Because the running time of Tr(ca) is less than that of GCD{f(c^{-1}x),Tr(x)}, the criterion that Tr(ca)=1 provides uniform efficiency for the number of trials.

Here the following natural question arises:

Is it possible in the similar way to develop an efficient probabilistic algorithm for solving quadratic equations over $GF(p^m)$ $(m\geqq 2$, p:odd prime)?

The question motivates us to study an efficient decision algorithm for quadratic residuosity in $GF(p^m)$ $(m\geqq 2$, p:odd prime). The problem will be considered in *CHAPTER VI*.

The algorithms to solve quadratic equations over $GF(p)$ are not only of theoretical interest but also of practical use, because those are applicable to the decryption procedures of

several *provably secure* public-key cryptosystems [Ra79,Wil80, Wil85,KIT87,KIT88a,KIT88b]. In addition, the algorithms to solve quadratic equations over $GF(2^m)$ are similarly not only of theoretical interest but also of practical importance, because those are supposed to be applicable to the decoding procedures of *Reed-Solomon Codes*, etc.

Further studies are necessary to realize those algorithms by hardwares or softwares with compact size. In addition, the extension of the algorithms to higher degree equations must be considered, however, this is somewhat of theoretical interest.

CHAPTER VI :

DECISION ALGORITHM FOR QUADRATIC RESIDUOSITY IN GF(p^m)

This Chapter develops two types of efficient decision algorithm for *quadratic residuosity* in *GF*(p^m) ($m \geqq 2$, p:odd prime) [IT87c,IT88c], i.e., one is based on canonical bases and the other is based on normal bases. Both algorithms, the algorithm based on canonical bases and the one based on normal bases, efficiently transform a quadratic residuosity problem in *GF*(p^m) to that in *GF*(p). Furthermore the transformed problem in *GF*(p) possesses the same quadratic residuosity with the original problem in *GF*(p^m), then those algorithms output the result by efficiently evaluating Legendre symbol. The algorithms are somewhat of theoretical interest, however, they are applicable to a probabilistic quadratic polynomial factorization over *GF*(p^m), where $m \geqq 2$ and p is an odd prime.

6.1 INTRODUCTION

In *CHAPTER V*, we have developed efficient probabilistic algorithms for solving quadratic equations over *GF*(p) (p:odd prime) and *GF*(2^m). The studies in *CHAPTER V* have provides us the following *natural* and *naive* question:

> ***Is it possible to develop an efficient probabilistic algorithm for solving quadratic equations over GF(p^m), where p is an odd prime and $m \geqq 2$?***

The above question is the primary motivation for the study on quadratic residuosity in *GF*(p^m). (See *CHAPTER V*.) Furthermore

the secondary motivation for the study on quadratic residuosity in $GF(p^m)$ is as follows:

Quadratic residuosity problems in $GF(p)$ (p:odd prime) can be efficiently evaluated by Legendre and Jacobi symbol, e.g., the law of quadratic recprocity [Sch86] (See Lemma 5.5 and 5.6 in *CHAPTER V*.), while those in $GF(p^m)$ ($m \geqq 2$, p:odd prime) has no useful algorithms except *Euler's Criterion* [Sch86]. However, Euler's Criterion is based on the exponentiation in $GF(p^m)$, thus the running time is somewhat large. From a theoretical point of view, it is a valuable work to develop a fast decision algorithm for quadratic residuosity in $GF(p^m)$.

The naive question and the theoretical interest *strongly* provoke us to develop new algorithms for quadratic residuosity in $GF(p^m)$ ($m \geqq 2$, p:odd prime).

In this Chapter, we present two types of efficient decision algorithm for quadratic residuosity in $GF(p^m)$ ($m \geqq 2$, p:odd prime), i.e., one is based on canonical bases and the other is based on normal bases. The algorithm based on canonical bases is composed of two parts, i.e., the first part efficiently transforms a quadratic residuosity problem in $GF(p^m)$ to that in $GF(p)$ and the second part decides the result of the original problem evaluating Legendre symbol. The algorithm based on normal bases also has the same structure with the above algorithm, however, it is inspired rather by the algorithm for multiplicative inverses in *CHAPTER IV*. Because of their efficiency, the decision algorithms for $GF(p^m)$ provide an efficient probabil-

istic algorithm for solving quadratic equations over $GF(p^m)$, where $m \geqq 2$ and p is an odd prime, and at the same time present uniform efficiency for the number of trials. Thus the proposed decision algorithms for $GF(p^m)$ are not only of theoretical interest but also of practical use.

The organization of this Chapter is as follows:

Section 6.2 provides some mathematical definitions and lemmas as preliminaries. Section 6.3 proposes an efficient decision algorithm for quadratic residuosity in $GF(p^m)$ ($m \geqq 2$, p:odd prime) based on canonical bases [IT87c,IT88c]. Furthermore Section 6.4 presents an efficient decision algorithm for quadratic residuosity in $GF(p^m)$ based on normal bases [IT87c, IT88c]. Section 6.5 finally summarizes the results in this Chapter and gives conclusions, remarks and open problems.

6.2 MATHEMATICAL PRELIMINARIES

This Section provides some mathematical definitions and lemmas for the subsequent discussions.

DEFINITION 6.1 [IT87c,IT88c] *Let $\forall c \in GF(p^m)$, where $m \geqq 2$, p is an odd prime and $c \neq 0$. If $x^2 = c \pmod{f(x)}$ has roots in $GF(p^m)$, c is called quadratic residue; otherwise quadratic non-residue, where $f(x)$ is an irreducible polynomial of degree m over $GF(p)$.* □

Remark 6.2 Throughout this Chapter, we assume that p is an odd prime and $m \geqq 2$. □

Remark 6.3 For notational simplicity, the set of all quad-

ratic (non) residues in $GF(p^m)$ is denoted by *QR* (*QNR*), respectively. □

The following definition is the generalization of *Legendre symbol* in $GF(p)$. (See Definition 5.3.)

DEFINITION 6.4 [IT87c,IT88c] *For* $\forall x \in GF(p^m)$ $(x \neq 0)$, *define the following symbol such that*

$$(x/GF(p^m)) = 1, \quad x \in QR,$$

$$(x/GF(p^m)) = -1, \quad x \in QNR.$$

Here $(x/GF(p^m))$ *is called generalized Legendre symbol (denoted GLS) in* $GF(p^m)$. □

The following lemma, in addition, is the generalization of *Euler's Criterion* in $GF(p)$. (See Lemma 5.9.)

LEMMA 6.5 [IT87c,IT88c] *For* $\forall x \in GF(p^m)$ $(x \neq 0)$, *we have the relation that* $x^{(p\hat{}m-1)/2} = (x/GF(p^m)) \pmod{f(x)}$, *where* $f(x)$ *is an irreducible polynomial of degree* m *over* $GF(p)$. *We refer to the above relation as generalized Euler's Criterion (denoted GEC).* □

Remark 6.6 The symbol p^m denotes p^m. (See Remark 3.2.) □

6.3 DECISION ALGORITHM FOR QUADRATIC RESIDUOSITY IN $GF(p^m)$ BASED ON CANONICAL BASES

This Section presents an efficient decision algorithm for quadratic residuosity in $GF(p^m)$ based on canonical bases. The algorithm transforms a quadratic residuosity problem in $GF(p^m)$ to that in $GF(p)$ by manipulating the matrix operations over $GF(p)$ and outputs the result of the original problem by evaluating Legendre symbol.

Let $f(x)=x^m+f_{m-1}x^{m-1}+\cdots+f_1x+f_0$ be an irreducible polynomial of degree m over $GF(p)$. Assume that r is one of the roots of $f(x)=0$, then the set $\{1,r,r^2,\cdots,r^{m-1}\}$ spans $GF(p^m)$, and is called a *canonical basis* over $GF(p)$. Then for $\forall x\in GF(p^m)$, x can be represented by $x_0+x_1r+x_2r^2+\cdots+x_{m-1}r^{m-1}$, where $x_i\in GF(p)$ $(0\leqq i\leqq m-1)$, employing the canonical basis. In addition, define the vector x over $GF(p)$ by $x=[x_0,x_1,x_2,\cdots,x_{m-1}]$. Here we present some significant lemmas to derive the decision algorithm for quadratic residuosity in $GF(p^m)$.

LEMMA 6.7 [IT87c,IT88c] *Let $f(x)=x^m+f_{m-1}x^{m-1}+\cdots+f_1x+f_0$ be an irreducible polynomial of degree m over GF(p), and define a canonical basis $\{1,r,r^2,\cdots,r^{m-1}\}$, where $f(r)=0$. If $\forall x\in GF(p^m)$ has the vector representation x by the canonical basis such that $x=[x_0,x_1,x_2,\cdots,x_{m-1}]^T$, then $y=xr \pmod{f(r)}$ has the vector representation y such that $y=[y_0,y_1,y_2,\cdots,y_{m-1}]^T=Fx$, where*

$$F=\begin{bmatrix} 0 & 0 & \cdots & 0 & -f_0 \\ & & & & -f_1 \\ & & & & -f_2 \\ & I_{m-1} & & & \vdots \\ & & & & \vdots \\ & & & & \vdots \\ & & & & -f_{m-1} \end{bmatrix},$$

I_k denotes $k\times k$ identity matrix and "T" denotes the transpose of matrices. □

Proof: Let $f(x)=x^m+f_{m-1}x^{m-1}+\cdots+f_1x+f_0$. For $\forall x\in GF(p^m)$, x is given by $x=x_0+x_1r+x_2r^2+\cdots+x_{m-1}r^{m-1}$, where $f(r)=0$. Then

$$y=y_0+y_1r+y_2r^2+\cdots+y_{m-1}r^{m-1}$$

$$
\begin{aligned}
&= xr \pmod{f(r)} \\
&= (x_0 + x_1 r + x_2 r^2 + \cdots + x_{m-1} r^{m-1}) r \pmod{f(r)} \\
&= x_0 r + x_1 r^2 + x_2 r^3 + \cdots + x_{m-1} r^m \pmod{f(r)} \\
&= -x_{m-1} f_0 + (x_0 - x_{m-1} f_1) r + (x_1 - x_{m-1} f_2) r^2 + \cdots \\
&\qquad \cdots + (x_{m-2} - x_{m-1} f_{m-1}) r^{m-1}.
\end{aligned}
$$

Observing the above equation, it follows that

$$y = [0, x_0, x_1, \cdots, x_{m-2}]^T + x_{m-1}[-f_0, -f_1, -f_2, \cdots, -f_{m-1}]^T.$$

Thus we have $y = Fx$, where $x = [x_0, x_1, x_2, \cdots, x_{m-1}]^T$. □

LEMMA 6.8 [IT87c, IT88c] *Let $f(x) = x^m + f_{m-1} x^{m-1} + \cdots + f_1 x + f_0$ be an irreducible polynomial of degree m over $GF(p)$ and let r be one of the roots of $f(x) = 0$. If r^n $(n \geqq 1)$ is given by*

$$r^n = c_0 + c_1 r + c_2 r^2 + \cdots + c_{m-1} r^{m-1},$$

then we have $c = [c_0, c_1, c_2, \cdots, c_{m-1}]^T = F^n [1, 0, 0, \cdots, 0]^T$. □

Proof: Definitely $r^n = \underbrace{r \times r \times \cdots \times r}_{n} \times 1$. Recalling Lemma 6.7, we have $c = F^n [1, 0, 0, \cdots, 0]^T$. □

Here we will show a *trivial* but *significant* property on the matrix F of Lemma 6.7. in the following lemma.

LEMMA 6.9 [IT87c, IT88c] *For the matrix F in LEMMA 6.7, we have $|F| = (-1)^m f_0$, where $|A|$ denotes the determinant of matrix A.* □

Proof: Notice the definition of the matrix F in Lemma 6.7. Thus we have $|F| = (-1)^{m+1}(-f_0) \cdot |I_{m-1}| = (-1)^{m+2} f_0 \cdot 1 = (-1)^m f_0$. □

The following lemma provides a simple property for *GLS*, however, it plays one of the most important roles to derive an efficient algorithm for quadratic residuosity in $GF(p^m)$.

LEMMA 6.10 [IT87c, IT88c] *Let g be a generator of the multiplicative group in $GF(p^m)$. For $\forall x \in GF(p^m)$ $(x \neq 0)$, there exists some integer n $(0 \leqq n \leqq p^m - 2)$ such that $x = g^n$. Then*

we have that if n is even $x \in QR$; *otherwise* $x \in QNR$. □

Proof: Assume that n is even. Then there exists an integer k such that n=2k. Recalling *GEC* (See Lemma 6.5.), we have

$$x^{(p^m-1)/2} = g^{n(p^m-1)/2}$$
$$= g^{2k(p^m-1)/2} = \{g^{(p^m-1)}\}^k$$
$$= 1^k = 1 = (x/GF(p^m)).$$

It follows that $x \in$ QR. The proof for odd n is quite similar. □

We have enough ingredients to develop an efficient decision algorithm for quadratic residuosity in $GF(p^m)$. Here we will show the algorithm in the case that f(x) is primitive.

CASE 1: f(x) is a primitive irreducible polynomial.

Let r be one of the roots of f(x)=0. Since f(x) is a primitive irreducible polynomial over *GF*(p), r is a *generator* of the multiplicative group in $GF(p^m)$. Hence f(x) can be factored over *GF*(p) into the form such that

$$f(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_1 x + f_0$$
$$= (x-r)(x-r^p)(x-r^{p^2})\cdots(x-r^{p^{(m-1)}}).$$

This equation definitely implies that

$$f_0 = (-1)^m r^{\{1+p+(p^2)+\ldots+p^{(m-1)}\}} = (-1)^m N(r),$$

where N(r) is *Norm* of r [LN83]. Here we have the following interesting lemma on the roots of f(x)=0.

LEMMA 6.11 [IT87c,IT88c] *Assume that f(x) is a primitive irreducible polynomial of degree m over GF(p) and r is one of the roots of f(x)=0. Then N(r) is a generator of the multiplicative group in GF(p).* □

Proof: Since f(x) is a primitive irreducible polynomial over

$GF(p)$ and r is one of the roots of $f(x)=0$, r is a generator of the multiplicative group in $GF(p^m)$. Notice that

$$N(r)=r^{\{1+p+p\hat{}2+\cdots+p\hat{}(m-1)\}}$$
$$=r^{(p\hat{}m-1)/(p-1)}\in GF(p).$$

Here we will prove *by contradiction* that N(r) is a generator of the multiplicative group in $GF(p)$. Assume that N(r) is not a generator of the multiplicative group in $GF(p)$, i.e., there exists an integer e $(1\leqq e\leqq p-2)$ such that $\{N(r)\}^e=1 \pmod p$. By the definition of N(r), we have $\{N(r)\}^e=r^{e(p\hat{}m-1)/(p-1)}=1$. This implies that the order of r is less than p^m-1, however, it contradicts the assumption that r is a generator of the multiplicative group in $GF(p^m)$. Hence N(r) must be a generator of the multiplicative group in $GF(p)$. □

Notice that r, one of the roots of $f(x)=0$, is a generator of the multiplicative group in $GF(p^m)$. For $\forall x\in GF(p^m)$ $(x\neq 0)$, there exists an integer n $(0\leqq n\leqq p^m-2)$ satisfying $x=r^n$. Thus x has a vector representation such that

$$x=r^n \pmod{f(r)}$$
$$=x_{00}+x_{10}r+x_{20}r^2+\cdots+x_{m-1,0}r^{m-1},$$

and we define the vector $x=[x_{00},x_{10},x_{20},\cdots,x_{m-1,0}]^T$ over $GF(p)$. Recalling Lemma 6.8, we have $x=F^n[1,0,0,\cdots,0]^T$. Furthermore consider $x_1=xr \pmod{f(x)}$. Then

$$x_1=x_{01}+x_{11}r+x_{21}r^2+\cdots+x_{m-1,1}r^{m-1}$$
$$=xr \pmod{f(x)}$$
$$=r^n\cdot r \pmod{f(x)}.$$

By Lemma 6.7 and $x=F^n[1,0,0,\cdots,0]^T$, it follows that

$$x_1=[x_{01},x_{11},x_{21},\cdots,x_{m-1,1}]^T=Fx$$
$$=F^n[0,1,0,\cdots,0]^T.$$

Define the following vectors x_i's over $GF(p)$ recursively.

$$x_i = F x_{i-1} \quad (1 \leqq i \leqq m-1),$$

where $x_i = [x_{0i}, x_{1i}, x_{2i}, \cdots, x_{m-1,i}]^T$ and $x_0 = x$. In addition define the matrix $S = [x_0, x_1, x_2, \cdots, x_{m-1}] = F^n \cdot I_m$. Recalling Lemma 6.9 and $f_0 = (-1)^m N(r)$, we have

$$\begin{aligned} |S| &= |F|^n |I_m| = \{(-1)^m f_0\}^n \cdot 1 \\ &= \{(-1)^m (-1)^m N(r)\}^n \\ &= \{N(r)\}^n, \end{aligned}$$

thus $(|S|/p) = (N(r)/p)^n$. Notice that $N(r)$ is a generator of the multiplicative group in $GF(p^m)$. (See Lemma 6.11.) It follows that $(N(r)/p) = -1$, and thus $(|S|/p) = (-1)^n$. In addition recalling Lemma 6.10, we have $(x/GF(p^m)) = (-1)^n$. Hence we obtain the relation that $(x/GF(p^m)) = (|S|/p)$.

Observing the relation that $(x/GF(p^m)) = (|S|/p)$, the quadratic residuosity problem in $GF(p^m)$ is reduced to that in $GF(p)$. Thus we have the following decision algorithm for quadratic residuosity in $GF(p^m)$.

ALGORITHM 6.12:

S1. *Input* $x \in GF(p^m)$ $(x \neq 0)$;
$$x = x_{00} + x_{10} r + x_{20} r^2 + \cdots + x_{m-1,0} r^{m-1}.$$

S2. *Define* $x_0 = [x_{00}, x_{10}, x_{20}, \cdots, x_{m-1,0}]^T$

S3. *Compute* $x_i = [x_{0i}, x_{1i}, x_{2i}, \cdots, x_{m-1,i}]^T$
$$= F x_{i-1} \quad (1 \leqq i \leqq m-1).$$

S4. *Define* $S = [x_0, x_1, x_2, \cdots, x_{m-1}]$.

S5. *Compute* $(|S|/p)$.

□

The above algorithm, *ALGORITHM 6.12*, is defined in the case that $f(x)$ is a primitive irreducible polynomial over $GF(p)$.

Then we will develop a more general algorithm, which decides quadratic residuosity in $GF(p^m)$ in the case that f(x) is a non-primitive irreducible polynomial over $GF(p)$.

CASE 2: f(x) is a non-primitive irreducible polynomial.

Assume that $f(x)=x^m+f_{m-1}x^{m-1}+\cdots+f_1x+f_0$ is a non-primitive irreducible polynomial over $GF(p)$. The following Theorem shows that under some condition *ALGORITHM 6.12* can be applied to decide quadratic residuosity in $GF(p^m)$ even in the case that f(x) is a non-primitive irreducible polynomial over $GF(p)$.

THEOREM 6.13 [IT87c,IT88c] *Let $f(x)=x^m+f_{m-1}x^{m-1}+\cdots+f_1x+f_0$ be a non-primitive irreducible polynomial of degree m over GF(p). If $((-1)^m f_0/p)=-1$, then ALGORITHM 6.12 decides quadratic residuosity in $GF(p^m)$.* □

Proof: Assume that s is one of the roots of f(x)=0 and g is a generator of the multiplicative group in $GF(p^m)$. Then g is given by $g=g_0+g_1s+g_2s^2+\cdots+g_{m-1}s^{m-1}$, where $g_i\in GF(p)(0\leqq i\leqq m-1)$. On the other hand, for $\forall x\in GF(p^m)$ $(x\neq 0)$ there exists an integer n $(0\leqq n\leqq p^m-2)$ such that $x=g^n$. Here x has a vector representation such that

$$x=g^n \pmod{f(s)}$$
$$=x_{00}+x_{10}s+x_{20}s^2+\cdots+x_{m-1,0}s^{m-1},$$

and we have the vector $x=[x_{00},x_{10},x_{20},\cdots,x_{m-1,0}]^T$ over $GF(p)$. Define the matrix G such that $G=g_0I_m+g_1F+g_2F^2+\cdots+g_{m-1}F^{m-1}$, then x is given by $x=G^n[1,0,0,\cdots,0]^T$. Notice that s, one of the roots of f(x)=0, can be represented by $s=g^k$ for some integer k. Recalling that $f_0=(-1)^mN(s)$, we have

$$
\begin{aligned}
f_0 &= (-1)^m N(s) \\
&= (-1)^m s^{(p^m-1)/(p-1)} \\
&= (-1)^m g^{k(p^m-1)/(p-1)} \\
&= (-1)^m \{N(g)\}^k .
\end{aligned}
$$

Observing the above equation, it follows that $(-1)^m f_0 = \{N(g)\}^k$. Noting the assumption that $((-1)^m f_0/p)=-1$, we have $((-1)^m f_0/p) = (N(g)/p)^k = -1$. This implies that k is odd, because $(N(g)/p)=-1$. (See Lemma 6.11.) Furthermore recalling that $s=g^k$, we obtain $F=G^k$, and thus we have $|F| = (-1)^m f_0 = |G|^k$. Hence it follows that $(|F|/p)=(|G|/p)^k=((-1)^m f_0/p)=-1$. Noting that k is odd, thus we have $(|G|/p)=-1$. Define the vectors x_i $(1 \leqq i \leqq m-1)$ over $GF(p)$ recursively by $x_i = Fx_{i-1} = [x_{0i}, x_{1i}, x_{2i}, \cdots, x_{m-1,i}]^T$, where $x_0 = x$. Recalling that $x=g^n$,

$$
\begin{aligned}
x_1 &= Fx_0 \\
&= FG^n [1,0,0,\cdots,0]^T \\
&= F\{g_0 I_m + g_1 F + g_2 F^2 + \cdots + g_{m-1} F^{m-1}\}^n [1,0,0,\cdots,0]^T \\
&= \{g_0 I_m + g_1 F + g_2 F^2 + \cdots + g_{m-1} F^{m-1}\}^n F[1,0,0,\cdots,0]^T \\
&= G^n [0,1,0,\cdots,0]^T .
\end{aligned}
$$

Here we have the matrix S such that $S=[x_0, x_1, x_2, \cdots, x_{m-1}] = G^n \cdot I_m$ and $|S| = |G|^n$. Recalling Lemma 6.10, it follows that $(x/GF(p^m))=(-1)^n$. In addition noting that $(|G|/p)=-1$, thus we have $(|S|/p)=(|G|/p)^n=(-1)^n$. Hence we obtain the relation that $(x/GF(p^m))=(|S|/p)$. □

Remark 6.14 If $f(x)=x^m+f_{m-1}x^{m-1}+\cdots+f_1x+f_0$ is primitive, then f_0 always satisfies that $((-1)^m f_0/p)=-1$. Hence the assumption of Theorem 6.13 includes the case that f(x) is a primitive irreducible polynomial over $GF(p)$. □

Here we give the following examples to demonstrate the *validity* and the *availability* of *ALGORITHM 6.12.*

EXAMPLE 6.15:

Let $f(x)=x^2+x+2$ be a polynomial over $GF(3)$. It is easy to verify that $f(x)$ is a primitive irreducible polynomial over $GF(3)$! Noting that $m=2$, $f_0=2$ and $p=3$, we have $((-1)^2 2/3)=-1$.

Here every quadratic residue$\in GF(3^2)$ is as follows:

$$\begin{aligned} 1 &= 1^2 = 2^2 \pmod{f(r)}, \\ 1+2r &= r^2 = (2r)^2 \pmod{f(r)}, \\ 2+r &= (1+r)^2 = (2+2r)^2 \pmod{f(r)}, \\ 2 &= (2+r)^2 = (1+2r)^2 \pmod{f(r)}, \end{aligned} \qquad (6.1)$$

where r satisfies $f(r)=r^2+r+2=0$. Consider $x=2+2r\in GF(3^2)$. Observing Eq.(6.1), $x=2+2r\in$ QNR. Define the vectors x_0 and x_1 over $GF(3)$ such that $x_0=[2,2]^T$ and $x_1=Fx_0=[2,0]^T$, where

$$F=\begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix}.$$

Here we define the matrix S such that

$$F=[x_0, x_1]=\begin{bmatrix} 2 & 2 \\ 2 & 0 \end{bmatrix}.$$

Recalling Theorem 6.13, $(2+2r/GF(3^2))=(|S|/3)=(2/3)=-1$. Hence *ALGORITHM 6.12* confirms that $x=2+2r\in$ QNR. Furthermore consider $y=2+r\in GF(3^2)$. Observing Eq.(6.1), it follows that $y=2+r\in$ QR. For $y=2+r\in GF(3^2)$, we similarly define the matrix S such that

$$S=\begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}.$$

By Theorem 6.13, we have $(2+r/GF(3^2))=(|S|/3)=(1/3)=1$. Thus *ALGORITHM 6.12* confirms that $y=2+r\in$ QR. □

EXAMPLE 6.16:

Let $f(x)=x^2+3$ be a polynomial over $GF(5)$. It is also easy to verify that $f(x)$ is a non-primitive irreducible polynomial over $GF(5)$! Noting that $m=2$, $f_0=3$ and $p=5$, the assumption in Theorem 6.13 that $((-1)^2 3/5)=-1$ is satisfied.

Here every quadratic residue$\in GF(5^2)$ is as follows:

$$\begin{aligned}
1 &= 1^2 = 4^2 \pmod{f(s)},\\
4 &= 2^2 = 3^2 \pmod{f(s)},\\
2 &= s^2 = (4s)^2 \pmod{f(s)},\\
3+2s &= (1+s)^2=(4+4s)^2 \pmod{f(s)},\\
1+4s &= (2+s)^2=(3+4s)^2 \pmod{f(s)},\\
1+s &= (3+s)^2=(2+4s)^2 \pmod{f(s)},\\
3+3s &= (4+s)^2=(1+4s)^2 \pmod{f(s)},\\
3 &= (2s)^2 = (3s)^2 \pmod{f(s)},\\
4+4s &= (1+2s)^2=(4+3s)^2 \pmod{f(s)},\\
2+3s &= (2+2s)^2=(3+3s)^2 \pmod{f(s)},\\
2+2s &= (3+2s)^2=(2+3s)^2 \pmod{f(s)},\\
4+s &= (4+2s)^2=(1+3s)^2 \pmod{f(s)},
\end{aligned} \tag{6.2}$$

where s satisfies $f(s)=s^2+3=0$. Consider $x=1+2s\in GF(5^2)$. Observing Eq.(6.2), $x=1+2s\in$ QNR. Define the vectors x_0 and x_1 over $GF(5)$ such that $x_0=[1,2]^T$ and $x_1=Fx_0=[4,1]^T$, where

$$F=\begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix}.$$

Here we define the matrix S such that

$$S=[x_0, x_1]=\begin{bmatrix} 1 & 4 \\ 2 & 1 \end{bmatrix}.$$

By Theorem 6.13, we have $(1+2s/GF(5^2))=(|S|/5)=(3/5)=-1$. Thus *ALGORITHM 6.12* confirms that $x=1+2s\in$ QNR. Furthermore consider $y=4+s\in GF(5^2)$. Observing the Eq.(6.2), we have $y=4+s\in$ QR. For $y=4+s\in GF(5^2)$, we similarly define the matrix S such that

$$S=\begin{bmatrix} 1 & 4 \\ 2 & 1 \end{bmatrix}.$$

By Theorem 6.13, we have $(4+s/GF(5^2))=(|S|/5)=(4/5)=1$. Thus *ALGORITHM 6.12* confirms that $y=4+s\in$ QR. □

6.4 DECISION ALGORITHM FOR QUADRATIC RESIDUOSITY IN $GF(p^m)$ BASED ON NORMAL BASES

In this Section, we propose an efficient decision algorithm for quadratic residuosity in $GF(p^m)$ based on normal bases. The primary idea to derive the algorithm is quite similar with that of a fast algorithm for multiplicative inverses in finite fields in *CHAPTER IV*.

DEFINITION 6.17 [MS77] *A basis in $GF(p^m)$ over $GF(p)$ of the form, $\{a, a^p, a^{p^2}, \cdots, a^{p^{(m-1)}}\}$, is called a normal basis in $GF(p^m)$ over $GF(p)$.* □

The following lemma provides a crucial clue to develop the efficient algorithm for quadratic residuosity in $GF(p^m)$.

LEMMA 6.18 [IT87c] *Let $x \in GF(p^m)$. If x is represented by a normal basis such that*

$$x = x_0 a + x_1 a^p + x_2 a^{p^2} + \cdots + x_{m-1} a^{p^{(m-1)}}$$

$$= [x_0, x_1, x_2, \cdots, x_{m-1}],$$

then we have $x^{p^k} = [x_{m-k}, x_{m-k+1}, \cdots, x_{m-1}, x_0, \cdots, x_{m-k-1}]$, i.e., x^{p^k} $(1 \leqq k \leqq m-1)$ can be computed by k cyclic shifts over $GF(p)$ of the above vector representation of x. □

Recall the *generalized Euler's Criterion* (denoted *GEC*), i.e., for $\forall x \in GF(p^m)$ $(x \neq 0)$, $x^{(p^m-1)/2} = (x/GF(p^m))$. Hence we can decide quadratic residuosity for $\forall x \in GF(p^m)$ $(x \neq 0)$, by computing $x^{(p^m-1)/2}$. Notice that

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

then it follows that

$$x^{(p^m-1)/2} = \{x^{\{p^{(m-1)}+p^{(m-2)}+\cdots+p+1\}}\}^{(p-1)/2}$$

$$= \{N(x)\}^{(p-1)/2}.$$

Recalling that *Euler's Criterion*, i.e., for $\forall x \in GF(p)$ $(x \neq 0)$, $x^{(p-1)/2} = (x/p)$ and $N(x) \in GF(p)$, thus we have

$$x^{(p^m-1)/2} = (N(x)/p).$$

Hence a quadratic residuosity problem in $GF(p^m)$ can be reduced to that in $GF(p)$. Here the following *naive* problem arise:

How can we efficiently compute $N(x)$ for $\forall x \in GF(p^m)$ $(x \neq 0)$?

The following Theorem presents an answer to the above problem.

THEOREM 6.19 [IT87c] *Assume that $\forall x \in GF(p^m)$ $(x \neq 0)$ and x is represented by a normal basis such that*

$$x = x_0 a + x_1 a^p + x_2 a^{p^2} + \cdots + x_{m-1} a^{p^{(m-1)}}$$

$=[x_0, x_1, x_2, \cdots, x_{m-1}]$.

Then there exists a fast algorithm for computing N(x), which requires [log m]+Hw(m)-1 multiplications in GF(p^m) and (m-1) cyclic shifts over GF(p), where [x] denotes a maximum integer $\leq x$ and Hw(i) denotes the Hamming weight of binary representation of i. □

Proof(Sketch): The proof of the above Theorem is quite similar with that of Theorem 4.13. Note that for $\forall x \in GF(p^m)$, N(x) is defined by $N(x)=x^{p^{(m-1)}+p^{(m-2)}+\cdots+p+1}$. Let M(m) and S(m) be the number of multiplications in GF(p^m) and that of cyclic shifts over GF(p), respectively to compute N(x). Recalling Lemma 6.18 and the proof of Theorem 4.13, we can easily show that there exists an algorithm, which requires M(m)=[*log* m]+Hw(m)-1 and S(m)=m-1 to compute N(x). (See the proof of Theorem 4.13 on details.) □

Remark 6.20 If the special machine are available to compute 2^k cyclic shifts over GF(p) by *one machine cycle*, then (m-1) cyclic shifts over GF(p) will be reduced to [*log* m]+Hw(m) cyclic shifts over GF(p). □

Here we demonstrate the *availability* of the above algorithm to compute N(x) in the following example.

EXAMPLE 6.21:

Let $x \in GF(3^{11})$. Then we have $N(x)=x^{88573}$. The following procedure computes $N(x)(=x^{88573})$ in [*log* 11]+Hw(11)-1=5 multiplications in GF(3^{11}) and (11-1)=10 cyclic shifts over GF(3). (See Theorem 6.19.)

S1. $x^3=x^3$: 1 cyclic shift over $GF(3)$

S2. $x^3 \cdot x=x^4$: 1 multiplication in $GF(3^{11})$

S3. $(x^4)^{3\hat{}2}=x^{36}$: 2 cyclic shifts over $GF(3)$

S4. $x^{36} \cdot x^4=x^{40}$: 1 multiplication in $GF(3^{11})$

S5. $(x^{40})^{3\hat{}4}=x^{3240}$: 4 cyclic shifts over $GF(3)$

S6. $x^{3240} \cdot x^{40}=x^{3280}$: 1 multiplication in $GF(3^{11})$

S7. $(x^{3280})^{3\hat{}2}=x^{29520}$: 2 cyclic shifts over $GF(3)$

S8. $x^{29520} \cdot x^4=x^{29524}$: 1 multiplication in $GF(3^{11})$

S9. $(x^{29524})^3=x^{88572}$: 1 cyclic shift over $GF(3)$

S10. $x^{88572} \cdot x=x^{88573}$: 1 multiplication in $GF(3^{11})$

$=N(x)$.

It is clear that the above procedure requires 5 multiplications in $GF(3^{11})$ in S2, S4, S6, S8 and S10 and 10 cyclic shifts over $GF(3)$ in S1, S3, S5, S7, and S9. Thus the above example confirms *validity* and the *availability* of the algorithm in Theorem 6.19. □

Now we can derive an efficient decision algorithm for quadratic residuosity in $GF(p^m)$ applying Theorem 6.19. The algorithm also has two parts, i.e., for $\forall x \in GF(p^m)$ $(x \neq 0)$, the first part computes N(x) using Theorem 6.19 and the second part evaluates (N(x)/p).

Hence the decision algorithm proposed in this Section outputs $(x/GF(p^m))=(N(x)/p)$. (Note that $x^{(p\hat{}m-1)/2}=(N(x)/p)$.)

6.5 CONCLUSION

This Chapter has proposed two types of efficient decision algorithm for quadratic residuosity in $GF(p^m)$ ($m \geqq 2$, p:odd prime), i.e., one is based on canonical bases and the other is

based on normal bases.

The decision algorithm based on canonical bases (denoted *DACB*) is composed of two parts. The first part of DACB defines an m× m matrix S and transforms a quadratic residuosity problem in $GF(p^m)$ to that in $GF(p)$ by computing $|S|$, where $|S|$ denotes the determinant of the matrix S. The second part of DACB evaluates $(|S|/p)$ and outputs the quadratic residuosity of the original problem in $GF(p^m)$.

Here we will study the running cost of DACB. Observing *ALGORITHM 6.12*, m multiplications and (m-1) additions in $GF(p)$ are required to compute $x_i = Fx_{i-1}$. Thus the running cost to define the matrix S is $O_A(m^2)$, where $O_A(\cdot)$ denotes the order of the arithmetic operations in $GF(p)$. In addition, the running cost of a *naive* method to evaluate $|S|$, e.g., *Gaussian elimination method*, is $O_A(m^3)$, however, further studies have developed fast algorithms for evaluating the determinant of matrices [St69,Pan78,Pan81,CW82]. Hence the running cost of the evaluation of $|S|$ is $O_A(m^k)$, where $k< 2.495548$ [CW82]. Eventually the total running cost of DACB (denoted TRC_{DACB}) is

$$TRC_{DACB} = O_A(m^k) + LS,$$

where $k< 2.495548$ and LS denotes the running cost of the evaluation of Legendre symbol in $GF(p)$.

The decision algorithm based on normal bases (denoted *DANB*) is also composed of two parts. The first part of DANB transforms a quadratic residuosity problem in $GF(p^m)$ to that in

GF(p) by computing N(x), where N(x) is *Norm* of $x \in GF(p^m)$ [LN83]. The above transform can be efficiently carried out, because the fast algorithm given in Theorem 6.19 is applicable to compute N(x). The second part of DANB evaluates (N(x)/p) and outputs the quadratic residuosity of the original problem in $GF(p^m)$.

Here we will also study the running cost of DANB. Note that in general case the running cost of multiplications in $GF(p^m)$ is $O_A(m^3)$ employing normal bases.

Remark 6.22 It can be shown that f(x), *All One Polynomial* (denoted *AOP*) of degree m over *GF*(p), is irreducible over *GF*(p) iff (m+1) is a prime and p is a generator of the multiplicative group in *GF*(p). (The proof of the above statement is completely the same with that of Lemma 3.5. See [WW84] on details.) Furthermore the roots of f(x)=0 constructs a normal basis in $GF(p^m)$ over *GF*(p). In this case, the running cost of multiplications in $GF(p^m)$ is $O_A(m^2)$ employing *Massey-Omura Multiplier* in $GF(p^m)$. Unfortunately we have not yet found another conditions for multiplications in $GF(p^m)$ to be carried out in $O_A(m^2)$. □

Hence N(x) can be evaluated in $O_A(m^3 \log m)$ applying the fast algorithm given in Theorem 6.19. DANB is composed of two parts, i.e., the evaluation of N(x) and that of (N(x)/p), thus the total running cost of DANB (denoted TRC_{DANB}) is given by

$$TRC_{DANB} = O_A(m^3 \log m) + LS.$$

We finally consider the running cost of the generalized Euler's Criterion (denoted *GEC*). The GEC decides the quadratic residuosity in $GF(p^m)$ by computing $x^{(p^m-1)/2} = (x/GF(p^m))$. Note

that Schonhage's fast multiplication scheme [Sch77] carries out a multiplication of polynomials over $GF(p)$ in $O_A(m \log m)$.

Hence $(x/GF(p^m))$ can be evaluated in $O_A(m^2 \log m \cdot \log p)$ by applying Schonhage's fast multiplication scheme and the fast exponentiation scheme, e.g., binary method [Knu81], etc.

Now we will compare the performance of the above algorithms, i.e., DACB, DANB and GEC. The running cost of each algorithm can be summarized as follows:

$$TRC_{DACB} = O_A(m^k)+LS,\ k< 2.495548,$$

$$TRC_{DANB} = \begin{cases} O_A(m^3 \log m)+LS, & \text{for general case,} \\ O_A(m^2 \log m)+LS, & \text{for special case (Remark 6.21),} \end{cases}$$

$$TRC_{GEC} = O_A(m^2 \log m \cdot \log p).$$

Observing the above, GEC is the most efficient in three algorithms, i.e., DACB, DANB and GEC. However, GEC is not so suitable for hardware or software implementation, because of the complicated structure of Schonhage's fast multiplication scheme [Sch77]. If Schonhage's fast multiplication scheme is not applied to GEC, then TRC_{GEC} will be $TRC_{GEC}=O_A(m^3 \log m)$.

On the other hand, DACB by Gaussian elimination method is fortunately suitable especially for software implementation. In the above implementation, we have $TRC_{DACB}=O_A(m^3)+LS$. However, DACB's applied the other fast algorithms [St69,Pan78,Pan81,CW82] are not suitable for hardware or software implementation, because of their complicated structure. In addition, DANB is fortunately suitable for hardware implementation employing Massey-Omura Multiplier in $GF(p^m)$. Furthermore DANB has a remarkable

feature that it is possible to be partially parallelized. If DANB is implemented by the parallel type Massey-Omura Multiplier in $GF(p^m)$, then the computation time of DANB (denoted CT_{DANB}) will be $CT_{DANB}=O_A((log\ m)^2)$.

It is easy to see that if the above decision algorithms are applied to a probabilistic quadratic polynomial factorization algorithm over $GF(p^m)$, then the efficient version of the probabilistic algorithm will be derived in the similar way with those in *CHAPTER V*. Furthermore, because of the efficiency of the decision algorithms for $GF(p^m)$, the efficient version of the probabilistic algorithm also provides uniform efficiency for the number of trials. (See *CHAPTER V* on details.)

We finally list the open problems related to the above algorithms. Observing *ALGORITHM 6.12*, we can find that the matrix S has the specified form that $S=[x_0, Fx_0, F^2x_0, \cdots, F^{m-1}x_0]$. Hence we have the following open problem on DACB:

OP1. *Develop an efficient algorithm to evaluate the determinant of* $S=[x_0, Fx_0, F^2x_0, \cdots, F^{m-1}x_0]$, *where F is an* $m \times m$ *companion matrix.*

Furthermore we must point out the following significant open problem on DANB:

OP2. *Find the conditions that multiplications in* $GF(p^m)$ *can be carried out in* $O_A(m^2)$ *employing Massey-Omura Multiplier for* $GF(p^m)$.

The above open problem *strongly* related to the configuration of Massey-Omura Multiplier in *CHAPTER III*.

CHAPTER VII :

PUBLIC-KEY CRYPTOSYSTEMS AND FINITE FIELD ARITHMETICS

This Chapter proposes several public-key cryptosystems and analyzes their security. In addition, this Chapter applies the algorithms developed in *CHAPTER III , IV , V* and *VI* to the public-key cryptosystems proposed in this Chapter.

7.1 INTRODUCTION

This Chapter presents several public-key cryptosystems (denoted *PKC*), e.g., the PKC based on a system of non-linear equations and the PKC based on the factorization of a large composite number, and analyzes the security of the above PKC's and *Knapsack-Type* PKC's. Furthermore this Chapter shows the *availability* of the algorithms developed in *CHAPTER III , IV , V* and *VI* to materialize the PKC's.

In 1976, *Diffie* and *Hellman* pointed out the possibility to realize the public-key cryptosystems [DH76]. The earliest realization of PKC's was developed by *Rivest*, *Shamir* and *Adleman* [RSA78], which is widely known nowadays as *RSA*. The public-key cryptosystem has a *remarkable* feature that the encryption-key is not identical to the decryption-key. The feature enables the receivers to make public their encryption-keys while keeping their decryption-keys secret. This provides considerable advantages for *data protection* in large scale communication network, because the key management problem will be *drastically* reduced!

To materialize the above scheme, the existence of the function f satisfying the following conditions is required:

C1. For $\forall x \in dom\ f$, f(x) is polynomially computable;

C2. For almost all $y \in rang\ f$, $f^{-1}(y)$ is not polynomially computable;

C3. There exists some information s, called *trapdoor information*, such that for $\forall y \in rang\ f$, $f^{-1}(y;s)$ is polynomially computable.

The function, which satisfies the above conditions C1 and C2, is referred to as *one way function* [DH76], and furthermore the one way function satisfying the condition C3 is called *trapdoor one way function* [DH76].

The candidates of trapdoor one way function are factorization of a large composite number, discrete logarithm, quadratic residuosity in Z_N and the other *NP-statements*. The PKC's proposed so far can be classified regarding trapdoor one way function as follows:

T1. *Factorization of a Large Composite Number*:
[RSA78,Ra79,Wil80,Wil85,KIT87,KIT88a,KIT88b];

T2. *Discrete Logarithm*:
[El85a];

T3. *Quadratic Residuosity in* Z_N:
[GM84,GHY85a,GHY85b,BBS86];

T4. *Knapsack Problem*:
[MH78,CR84,Nie86,MKN86];

T5. *The Other NP-Statements*:
[McE78,IM85,TKIFM86,TKIFM87,TFH88a,TFH88b].

The security of the above PKC's has been widely studied. Nowadays RSA is known as one of the most secure PKC's proposed so far and is believed that inverting RSA is equivalent to factoring a composite moduli, however, it has not been proven yet. The *computer scientific approach* to the security of RSA can be found in [GMT82,BCS83,VV83,CG84,Gol84,SA84]. The Knapsack-Type PKC's are pessimistically supposed to be *not so strong*, because some breaking methods have been developed by [Sha82,BDS82,Ad83, BLO83,Lag83,LO83]. For some of the above PKC's, it is proven that inverting the system is equivalent to solving the intractable problem, e.g., factoring a large composite number [Ra79, Wil80,Wil85,KIT87, KIT88a,KIT88b], quadratic residuosity [GM84, GHY85a,GHY85b,BBS86], etc.

On the other hand, in 1984 *A.Shamir* suggested the significance of *ID-Based Systems* [Sha84], i.e., ID-Based Cryptosystems and ID-Based Signature Schemes. One of the primary motivations for ID-Based System is further reduction of key management problem, because in the conventional PKC's, the key management center must have the large public-key file of every subscriber by physically secure methods. The ID-Based System has been actively studied since Shamir's suggestion, while the other new concept, i.e., *ID-Based Key Distribution System* [Ta86,Ta87], has been defined. The concrete examples of ID-Based Key Distribution System can be found in [Ta86,KO87,MI87,Oka87,Ta87].

In this Chapter, we study the security of *Knapsack-Type* PKC's [IKT84,KIST87] and present two types of PKC, i.e., one is

based on a system of non-linear equations [TKIFM86,TKIFM87, TFH88a,TFH88b] and the other is based on the factorization of a large composite number [KIT87,KIT88a,KIT88b]. Furthermore we propose the ID-Based Cryptosystem based on discrete logarithm problem [TIK87,TI88], which is one of the earliest concrete examples *strictly* in Shamir's sense.

The organization of this Chapter in as follows:

Section 7.2 analyzes the security of Knapsack-Type PKC's [IKT84,KIST87]. Section 7.3 presents the PKC based on a system of non-linear equations and consider the security [TKIFM86, TKIFM87,TFH88a,TFH88b]. Section 7.4 proposes the PKC based on the factorization of a large composite number and proves that inverting the system is equivalent to factoring a composite moduli [KIT87,KIT88a,KIT88b]. Furthermore Section 7.5 provides the ID-Based Cryptosystem based on discrete logarithm problem and analyze the security against the conspiracy of the entities [TIK87,TI88]. Section 7.6 finally summarizes the results in this Chapter, considers the application of the algorithms in *CHAPTER III ,IV , V* and *VI* to the above PKC's and in addition gives conclusions, remarks and open problems.

7.2 ANALYSIS ON KNAPSACK TYPE PUBLIC-KEY CRYPTOSYSTEMS

In this Section, we analyze the security of *Knapsack Type PKC's*. This Section is composed of three subsections as follows: Subsection 7.2.1 considers the security of the *general* Knapsack Type PKC's [IKT84] and shows several sufficient condi-

tions for *Knapsack Problems* to be solved in *linear time*. On the other hand, Subsection 7.2.2 analyzes the *Multiplicative* Knapsack Type PKC [CR84] and shows that there exist many decryption-keys distinct from the original one designed by the key-maker [KIST87]. Finally Subsection 7.2.3 summarizes and discusses the results in this Chapter.

7.2.1 ANALYSIS ON THE GENERAL KNAPSACK TYPE PKC's

This subsection analyzes the security of the *general* Knapsack Type PKC's and shows the sufficient conditions for Knapsack Problems to be solved in *linear time*.

Here we define *Knapsack Problem* in the following:

DEFINITION 7.2.1 [GJ79] *Let a_i $(1 \leqq i \leqq k)$ be arbitrary fixed positive integers. Then Knapsack Problem is a problem to find the solution $x_i \in \{0,1\}$ $(1 \leqq i \leqq k)$ for M, where M is given by $M=a_1x_1+a_2x_2+\cdots+a_kx_k$.* □

Remark 7.2.2 In Knapsack Type PKC's, $\{a_i\}$ $(1 \leqq i \leqq k)$ corresponds to the public-key and furthermore $\{x_i\}$ $(1 \leqq i \leqq k)$ and M will be plaintext and ciphertext, respectively. □

Here we present a simple attacking method for the general Knapsack Type PKC's. Without loss of generality, we assume that $a_1 < a_2 < \cdots < a_k$. For each plaintext $x=(x_1, x_2, x_3, \cdots, x_k)$, define X such that $X=x_1+x_2 2+x_3 2^2+\cdots+x_k 2^{k-1}$. In addition, define the pairs such that $\{X_{i1}, L_i\}$ and $\{X_{i2}, U_i\}$, where $X_{i1}=2^{i-1}$, $L_i=a_i$, $X_{i2}=2^i-1$ and $U_i=a_1+a_2+a_3+\cdots a_i$ $(1 \leqq i \leqq k)$. Note that L_i and U_i correspond to ciphertexts of the plaintexts X_{i1} and X_{i2}, respectively. Consider the following two graphs:

$Y_1 = L_i$, $2^{i-1} \leqq X < 2^i$ $(1 \leqq i \leqq k)$,

$Y_2 = U_i$, $2^{i-1} - 1 < X \leqq 2^i - 1$ $(1 \leqq i \leqq k)$.

It is clear that every pair {plaintext, ciphertext} lies between two graphs Y_1 and Y_2. Here we have the following Theorem:

THEOREM 7.2.3 [IKT84] *Let M be a ciphertext. Then*

S1. If $U_{i-1} < M < L_{i+1}$, *then* $x_i = 1$ *and* $x_j = 0$ $(i < j \leqq k)$;

S2. If $M > U_{k-1}$, *then* $x_k = 1$.

□

The following example shows the *validity* of Theorem 7.2.3.

EXAMPLE 7.2.4:

For k=5, a=(11,19,29,38,85) and M=78, find the solution x such that $a_1x_1 + a_2x_2 + a_3x_3 + a_4x_4 + a_5x_5 = 78$. Noting that $U_3 = 59$, $L_3 = 85$ and $U_3 < M < L_3$, we have $x_4 = 1$ and $x_5 = 0$. (See S1 of Theorem 7.2.3.) Thus the original problem can be reduced as follows: For k=3, a=(11,19,29) and $M' = M - a_4 = 78 - 38 = 40$, find the solution x such that $a_1x_1 + a_2x_2 + a_3x_3 = 40$. Observing that $U_2 = 30$ and $M' > U_2$, we have $x_3 = 1$. (See S2 of Theorem 7.2.3.) Iterating the similar procedures, we finally obtain $x_1 = 1$ and $x_2 = 0$. Thus we can find the plaintext x=(1,0,1,1,0). □

The above attacking method can be described as follows:

ALGORITHM 7.2.5:

```
    begin
S1. U0 :=0; ak+1 =:∞ ; n:=k;
S2. for i:=1 to k do
S3.   begin xi :=0; Ui :=Ui-1 +ai ; end
S4. while n> 0 do
S5.   begin
S6.     if M=Un then begin
S7.       for j:=1 to n do xj :=1; n:=0:
```

```
S8.          end else
S9.        if M=a_n then begin
S10.         x_n:=1; n:=0; else end
S11.       if M> U_{i-1} then
S12.       if M< a_{n+1} then begin
S13.         x_n:=1; M:=M-a_n; n:=n-1;
S14.       end else failure; n:=0;
S15.     else n:=n-1;
S16.   end
     end
```

□

Remark 7.2.6 It is easy to see that *ALGORITHM 7.2.5* terminates in $O(k)$ steps. □

Here we consider the extension of Theorem 7.2.3.

THEOREM 7.2.7 [IKT84] *Assume that M be a ciphertext. If* $U_{i-1} < M < L_{t+1}$, *then* $x_{t+1}=x_{t+2}=\cdots=x_k=0$ *and there exist at least one non-zero elements in* $\{x_i, x_{i+1}, \cdots, x_t\}$. □

Proof: It is clear that $x_{t+1}=x_{t+2}=\cdots=x_k=0$, because the assumption that $M< L_{t+1}$ implies $M< a_{t+1}$. Furthermore noting the assumption that $M> U_{i-1}=a_1+a_2+\cdots+a_{i-1}$, there exist at least one non-zero elements in $\{x_i, x_{i+1}, \cdots, x_t\}$. □

Remark 7.2.8 If we assume that t=i in Theorem 7.2.7, then Theorem 7.2.7 is reduced to Theorem 7.2.3. Thus Theorem 7.2.7 is the *natural* extension of Theorem 7.2.3. □

To demonstrate the *availability* of Theorem 7.2.7, we show the following brief example.

EXAMPLE 7.2.9:

For k=5, **a**=(11,19,29,38,85) and M=48, find the solution x such that $a_1x_1+a_2x_2+a_3+x_3+a_4x_4+a_5x_5=48$. Noting that $U_2=30$,

$U_3=59$, $L_2=38$, $L_5=85$ and $U_2 < M < L_5$, we have $x_5=0$ and $x_3+x_4 \neq 0$. Assume that $x_4=1$. Then it follows that $M'=M-a_4=10<a_1$, thus this is clearly incorrect! On the other hand, assume that $x_3=1$. Then it follows that $M'=M-a_3=19=a_2$, thus we have $x_1=0$ and $x_2=1$. Hence we can find the plaintext $x=(0,1,1,0,0)$. □

The following Theorems provide sufficient conditions for Knapsack Problems to be solved in *linear time*. The conditions are weaker than the conventional one, which is generally known as *super-increasing* [MH78].

THEOREM 7.2.10 [IKT84] *If* $a_2+a_3+\cdots+a_{n-1}<a_n$ $(3 \leqq n \leqq k)$, *then there exists an algorithm which solves the Knapsack Problems in* $O(k)$ *steps.* □

THEOREM 7.2.11 [IKT84] *If* $a_1+a_3+a_4+\cdots+a_{n-1}<a_n$ $(3 \leqq n \leqq k)$, *then there exists an algorithm which solves the Knapsack Problems in* $O(k)$ *steps.* □

Remark 7.2.12 Theorem 7.2.10 and 7.2.11 can be easily proven observing the attacking method in *EXAMPLE 7.2.9.* □

7.2.2 ANALYSIS ON MULTIPLICATIVE KNAPSACK TYPE PKC

This subsection analyzes the security of the *Multiplicative Knapsack Type* PKC proposed by *Chor* and *Rivest* in 1984 [CR84] by demonstrating a concrete attacking method. The attacking method finds the secret-key if the public knapsack vector includes at least three elements whose values are close to each other. In addition, this subsection points out that there exist many decryption-keys distinct from the original one designed by the key-maker [KIST87].

The following describes the procedures for generating the Multiplicative Knapsack Type PKC based on the arithmetic in finite fields [CR84].

SYSTEM GENERATION:

SG1. Let $GF(p^h)$ be a finite field, where $p \geqq h$, and assume that discrete logarithm in $GF(p^h)$ is efficiently computable, i.e., (p^h-1) can be factored into the product of small primes [PH78];

SG2. Pick a multiplicative generator $g \in GF(p^h)$ at random. Let G(x) be the minimal polynomial of g, i.e., G(x) is s primitive irreducible polynomial of degree h over $GF(p)$:

SG3. Pick an irreducible polynomial $M_t(y)$ of degree h over $GF(p)$, where $M_t(y)=m_0+m_1y+\cdots+m_{h-1}y^{h-1}+y^h$. Let t be one of the roots of $M_t(y)=0$;

SG4. Compute $a_i=log_g(t+i)$ $(0 \leqq i \leqq p-1)$ and define the vector $a=(a_0,a_1,\cdots,a_{p-1})$;

SG5. Let $perm:\{0,1,\cdots,p-1\}\rightarrow\{0,1,\cdots,p-1\}$ be a randomly chosen permutation. Define $b_i=a_{perm(i)}$;

SG6. Define $c_i=b_i+d \pmod{p^h-1}$ for a randomly chosen d, where $(0 \leqq d \leqq p^h-2)$;

SG7. *Public-Key* will be $c=(c_0,c_1,\cdots,c_{p-1})$, p and h and *Secret-Key* will be t, g, *perm* and d. □

The encryption and the decryption are defined as follows:

ENCRYPTION:

Let $M=(x_0,x_1,\cdots,x_{p-1})$ be a p-bit binary message of weight h, i.e., $x_0+x_1+\cdots+x_{p-1}=h$. Then the ciphertext C is defined by

$$C=E(M)=x_0c_0+x_1c_1+\cdots+x_{p-1}c_{p-1}$$

$$=x_0 b_0 + x_1 b_2 + \cdots + x_{p-1} b_{p-1} + hd \pmod{p^h - 1}.$$

□

DECRYPTION:

D1. Compute

$$g^{E(M)-hd} \pmod{p^h-1} = g^{x_0 b_0 + x_1 b_1 + \ldots + x_{p-1} b_{p-1}}$$
$$= (t+i_1)(t+i_2)\cdots(t+i_h)$$
$$= e_0 + e_1 t + \cdots + e_{h-1} t^{h-1}.$$

Recalling that $M_t(t) = m_0 + m_1 t + \cdots + m_{h-1} t^{h-1} + t^h = 0$, thus

$$(t+i_1)(t+i_2)\cdots(t+i_h) = e_0 + e_1 t + \cdots + e_{h-1} t^{h-1} + m_0 + m_1 t + \cdots + m_{h-1} t^{h-1} + t^h.$$

D2. Define p(t) such that

$$p(t) = e_0 + e_1 t + \cdots + e_{h-1} t^{h-1} + m_0 + m_1 t + \cdots + m_{h-1} t^{h-1} + t^h.$$

Thus we have $p(t) = (t+i_1)(t+i_2)\cdots(t+i_h)$. Hence we can find the roots of $p(t)=0$, i_j's, by successive substitution. Applying the permutation $perm^{-1}$ to i_j's, we can finally recover the original message vector M having the bit "1". □

Here we show an *Attacking Method* for the above Knapsack Type PKC. The attacking method is composed of two parts; The first part finds the primitive polynomial G(x) if the public knapsack vector c includes at least three elements whose values are close to each other. Furthermore the second part derives all the rest of secret-key, i.e., t,d and *perm*, if the primitive polynomial G(x) is known.

THEOREM 7.2.13 [KIST87] *For* $\forall c_i, c_j, c_k \in c$ *and* m $(1 \leqq m \leqq p-1)$, *we have* $G(x) \mid f(x) = 1 - x^{(c_k - c_i)} - m\{1 - x^{(c_j - c_i)}\}$. □

Proof: Recall the construction of the PKC. (See ***SYSTEM GENERATION.***) Then we have

$$g^{ci}=g^{bi+d}=g^{d}(t+n_i),$$
$$g^{cj}=g^{bj+d}=g^{d}(t+n_j),$$

where $n_i, n_j \in GF(p)$ $(n_i \neq n_j)$. This yields $g^d=(n_i-n_j)^{-1}(g^{ci}-g^{nj})$. In the same way, we have $g^d=(n_i-n_k)^{-1}(g^{ci}-g^{ck})$. Hence we obtain $(g^{ci}-g^{ck})-m(g^{ci}-g^{cj})=0$, where $m=(n_i-n_j)^{-1}(n_i-n_k)$ (mod p). Thus $1-g^{(ck-ci)}-m\{1-g^{(cj-ci)}\}=0$ and this implies that $G(x) \mid f(x)$. □

Theorem 7.2.13 provides an useful method to find G(x), however, it may produce several primitive polynomials of degree h over *GF*(p). In such a case, the following lemmas are available to identify G(x).

LEMMA 7.2.14 [KIST87] *Define* $S_r=\sum_{i=0}^{p-1} i^r$. *If* $r \neq 0$ *(mod p-1) then* $S_r=0$ *(mod p); otherwise* $S_r=p-1$ *(mod p).* □

Proof: Note that the following two identity:

$$\sum_{k=0}^{p-1}\{(k+1)^2-k^2\}=p^2,$$
$$\sum_{k=0}^{p-1}\{(k+1)^2-k^2\}=\sum_{k=0}^{p-1}(2k+1)$$
$$=2\sum_{k=0}^{p-1}k+\sum_{k=0}^{p-1}1$$
$$=2S_1+p.$$

Then we have $2S_1+p=p^2$. This implies that $S_0=0$ (mod p). In the similar way with the above, we have

$$\sum_{k=0}^{p-1}\{(k+1)^3-k^3\}=3S_2+3S_1+p=p^3,$$

then, $S_2=0$ (mod p). Hence we obtain $S_1=S_2=\cdots=S_{p-2}=0$ (mod p). If r=0 (mod p), then $S_r=\sum_{i=0}^{p-1} i^r=\sum_{i=0}^{p-1} 1=p-1$ (mod p). □

LEMMA 7.2.15 [KIST87] *If* $n=p^k+m$ $(0 \leqq k \leqq h-1,\ 0 \leqq m \leqq p-3)$, *then we have* $G(x) \mid V_n(x)=\sum_{i=0}^{p-1} x^{nci}$. □

Proof: Substitute g for x in $V_n(x)$. Thus we have

$$V_n(g)=g^{nd}\cdot\sum_{i=0}^{p-1}(t+i)^{p^k+m}$$
$$=g^{nd}\cdot\sum_{i=0}^{p-1}(t^{p^k}+i)(t+i)^m.$$

If $m=0$ we have $V_n(g)=g^{nd}\{\sum_{i=0}^{p-1} 1 + \sum_{i=0}^{p-1} i\}$. From Lemma 7.2.14, it follows that $V_n(g)=0$. On the other hand, if $1 \leqq m \leqq p-3$, then

$$V_n(g) = g^{nd}\{t^{p^k} \sum_{i=0}^{p-1} (t+i)^m + \sum_{i=0}^{p-1} i(t+i)^m\}$$
$$= g^{nd}\{t^{p^k} U_m(t) + W_m(t)\},$$

where $U_m(t) = \sum_{i=0}^{p-1} (t+i)^m$ and $W_m = \sum_{i=0}^{p-1} i(t+i)^m$. Here we have

$$U_m(t) = \sum_{i=0}^{p-1} \sum_{r=0}^{m} {}_mC_r \cdot t^{m-r} \cdot i^r$$
$$= pt^m + \sum_{r=1}^{m} \{{}_mC_r \cdot t^{m-r} \sum_{i=0}^{p-1} i^r\}$$
$$= \sum_{r=1}^{m} {}_mC_r \cdot t^{m-r} \cdot S_r \pmod p.$$

Recalling Lemma 7.2.14, we have $S_r=0 \pmod p$ $(1 \leqq r \leqq m)$, and thus $U_m(t)=0$. Furthermore we have

$$W_m(t) = \sum_{i=0}^{p-1} i(t+i)^m$$
$$= \sum_{i=0}^{p-1} i \sum_{j=0}^{m} {}_mC_j \cdot t^{m-j} \cdot i^j$$
$$= \sum_{j=0}^{m} \{{}_mC_j \cdot t^{m-j} \sum_{i=0}^{p-1} i^{j+1}\}$$
$$= \sum_{j=0}^{m} {}_mC_j \cdot t^{m-j} \cdot S_{j+1}.$$

For $0 \leqq j \leqq m \leqq p-3$, we have $S_{j+1}=0 \pmod p$ and thus $W_m(t)=0$. (See Lemma 7.2.14) Hence we can conclude that $V_n(g)=0$, and this implies that $G(x) \mid V_n(x)$. □

LEMMA 7.2.16 [KIST87] *Assume that $p-1 \leqq n$. Here $G(x) \mid V_n(x)$, where $V_n(x) = \sum_{i=0}^{p-1} x^{ni}$, iff ${}_nC_{p-1}, {}_nC_{2(p-1)}, \cdots, {}_nC_{j(p-1)}$ $(0 < j(p-1) \leqq n)$ are all divisible by p.* □

Proof: Note that

$$V_n(g) = g^{nd} \sum_{i=0}^{p-1} (t+i)^n$$
$$= g^{nd} \sum_{i=0}^{p-1} \sum_{r=0}^{m} {}_nC_r \cdot t^{n-r} \cdot i^r$$
$$= g^{nd}\{pt^n + \sum_{r=1}^{n} {}_nC_r \cdot t^{n-r} \sum_{i=0}^{p-1} i^r\}$$
$$= g^{nd} \sum_{r=1}^{n} {}_nC_r \cdot t^{n-r} \cdot S_r \pmod p.$$

Recalling Lemma 7.2.14, we have

$$V_n(g) = g^{nd} \sum_{\substack{r=0 \pmod{p-1} \\ 0<r\leq n}} {}_nC_r \cdot t^{n-r}(p-1) \pmod{p}.$$

Then $V_n(g)=0$ iff for every r such that $r=0 \pmod{p-1}$ $(0< r\leqq n)$, ${}_nC_r=0 \pmod{p}$. □

Note that if the public knapsack vector c includes at least three elements whose values are close to each other, we can find the primitive polynomial G(x), which is the fraction of the secret-key. Here we will show that we can derive the entire secret-key if the primitive polynomial G(x) is known.

THEOREM 7.2.17 [KIST87] *If the primitive polynomial G(x) is given, then the entire secret-key will be derived.* □

Proof: Define d' and b_i' as follows:

$$g^{d'}=g^{cx}-g^{cy},$$
$$b_i'=c_i-d' \quad (0\leqq i\leqq p-1).$$

Then we have

$$g^{bi'}=g^{ci-d'}=g^{bi+d-d'}=g^{bi}\cdot(n_x-n_y)^{-1}$$
$$=(t+perm(i))\cdot n,$$

where $n=(n_x-n_y)^{-1} \pmod{p}$. Recalling the construction of the cryptosystem, we have $g^{bi}=t+perm(i)$ $(1\leqq i\leqq p-1)$. Let t be represented by $t=e_0+e_1g+\cdots+e_{h-1}g^{h-1}$. Then

$$\begin{aligned} g^{b0} &= t+perm(0) = e_{00} + e_1g+\cdots+e_{h-1}g^{h-1}, \\ g^{b1} &= t+perm(1) = e_{10} + e_1g+\cdots+e_{h-1}g^{h-1}, \\ &\vdots \\ g^{bp-1} &= t+perm(p-1) = e_{p-1,0} + e_1g+\cdots+e_{h-1}g^{h-1}, \end{aligned}$$

where $e_{i0}=e_0+perm(i)$. Notice that $\{e_{00}, e_{10}, \cdots, e_{p-1,0}\}$ forms *GF*(p) and $g^{bi}-e_{i0}$ $(0\leqq i\leqq p-1)$ are all identical. Recalling the relation that $g^{bi'}=(t+perm(i))\cdot n$, then it follows that

$$g^{b0'} = \{t+perm(0)\}\cdot n = e_{00}' + e_1'g+\cdots+e_{h-1}'g^{h-1},$$
$$g^{b1'} = \{t+perm(1)\}\cdot n = e_{10}' + e_1'g+\cdots+e_{h-1}'g^{h-1},$$
$$\vdots$$
$$g^{bp-1'} = \{t+perm(p-1)\}\cdot n = e_{p-1,0}' + e_1'g+\cdots+e_{h-1}'g^{h-1},$$

where $e_{i0}' = e_{i0}\cdot n \pmod p$ and $e_i' = e_i\cdot n \pmod p$ $(0 \leqq i \leqq p-1)$. Thus $\{e_{00}', e_{10}', \cdots, e_{p-1,0}'\}$ forms $GF(p)$ and $g^{bi'} - e_{i0}'$ $(0 \leqq i \leqq p-1)$ are all identical. Define $t' = g^{bi'} - e_{i0}'$ and $perm'(i) = e_{i0}$. Then $\{g, d', t', perm'\}$ will be secret-key, because it satisfies the conditions SG2, SG3, SG5 and SG6 in *SYSTEM GENERATION*. □

The secret-key $\{g, d', t', perm'\}$ is not necessarily identical to the original secret-key $\{g, d, t, perm\}$. Thus we have following Theorem.

THEOREM 7.2.18 [KIST87] *Assume that a public knapsack vector c is given. Then there exist some secret-key other than the original one.* □

In order to show the *validity* of the attacking method, we demonstrate the following simple example:

EXAMPLE 7.2.19:

Let p=7 and h=3. Define the secret-key as follows:

$G(x) = x^3 + 6x + 2$,
$t = g^5 = 5g^2 + g + 5$,
$d = 100$,
$perm: \{0,1,2,3,4,5,6\} \rightarrow \{6,0,1,2,3,4,5\}$.

Then public-key knapsack vector c will be

$c = (34, 105, 163, 80, 108, 198, 232)$.

Finding the Primitive Polynomial G(x):

Recalling Theorem 7.2.13, we define c_i, c_j, c_k such that $(c_i, c_j, c_k) = (80, 105, 108)$. Then we have $f(x) = mx^{28} - x^{25} + 1 - m$, which

is divisible by G(x) for some m. Since *deg* f is small enough, we can apply directly polynomial factorization algorithm [Ber68, Ber70,Moe77]. In this case, we can derive only one primitive polynomial $G(x)=x^3+6x+2$ for m=5.

Derivation for the Rest of Secret-Key:

Choose $(c_x, c_y)=(80,34)$, then d' and b' are given by d'=43 and b'=(333,62,120,37,65,155,189), respectively. Define the polynomials such that $E_i(g)=g^{b_i'}=e_{i0}'+e_{i1}'g+e_{i2}'g^2$ $(0 \leqq i \leqq 6)$. The following table gives the coefficients of $E_i(g)$.

TABLE 7.2.20 Coefficients of $E_i(g)$

i	b_i'	e_{i0}'	e_{i1}'	e_{i2}'
0	333	6	5	4
1	62	4	5	4
2	120	2	5	4
3	37	0	5	4
4	165	5	5	4
5	155	3	5	4
6	189	1	5	4

Observing TABLE 7.2.20, we can decide secret-key t' and *perm'* as follows (See Theorem 7.2.17.):

$t'=4g^2+5g$,

perm': $\{0,1,2,3,4,5,6\} \rightarrow \{6,4,2,0,5,3,1\}$.

Computing $c_i=b_i+d' \pmod{7^3-1}$, we have the public knapsack vector c as follows:

$c_0=b_0'+d' \pmod{7^3-1}=333+43 \pmod{342}=34$,
$c_1=b_1'+d' \pmod{7^3-1}=62+43 \pmod{342}=105$,
$c_2=b_2'+d' \pmod{7^3-1}=120+43 \pmod{342}=163$,
$c_3=b_3'+d' \pmod{7^3-1}=37+43 \pmod{342}=80$,

$c_4 = b_4' + d' \pmod{7^3-1} = 65+43 \pmod{342} = 108,$

$c_5 = b_5' + d' \pmod{7^3-1} = 155+43 \pmod{342} = 198,$

$c_6 = b_6' + d' \pmod{7^3-1} = 189+43 \pmod{342} = 232.$

Thus we have c=(34,105,163,80,108,198,232). This implies that the secret-key {g,d',t',*perm'*} provides the same public-key c, hence the secret-key {g,d',t',*perm'*} is a *valid* secret-key to recover the massages. □

7.2.3 DISCUSSION

This Section has analyzed the security of *Knapsack Type* PKC's. In subsection 7.2.1, we have studied the security of *general* Knapsack Type PKC's and have shown several conditions for Knapsack Type PKC's to be solved in *linear time* [IKT84]. The conditions are somewhat weaker than the conventional one, which is generally referred to as *super-increasing* [MH78]. On the other hand, in subsection 7.2.2 we have developed an concrete *Attacking Method* [KIST87] for the Multiplicative Knapsack Type PKC [CR84]. The attacking method is composed of two steps; The first step is to find the primitive polynomial G(x), the fraction of the secret-key. We have proven that the primitive polynomial G(x) can be obtained if the public knapsack vector c includes at least three elements whose values are close to each other [KIST87]. The second step is to derive the rest of secret-key. We have shown that the rest of secret-key can be derived if the primitive polynomial G(x) is known [KIST87]. In addition, we have proven that there exist some secret-key other than the original one [KIST87].

The results in subsection 7.2.1 *implicitly* suggests that the values of all the elements in public knapsack vector must be uniform for secure PKC's. On the other hand, the results in subsection 7.2.2 *definitely* points out that the public knapsack vector must not include the elements of close value for secure PKC. Hence those results are remarkable because of their complementality, however, they will provide the criterion for the design of *secure* Knapsack-Type PKC's.

7.3 PKC BASED ON A SYSTEM OF NON-LINEAR EQUATIONS

This Section presents a PKC based on the difficulty of solving a system of non-linear equations over finite fields and analyzes the security of the PKC against the possible attacks.

7.3.1 PKC BASED ON A SYSTEM OF NON-LINEAR EQUATIONS

This subsection proposes a PKC based on the difficulty of solving a system of non-linear equations over finite fields, especially over $GF(2^m)$. The central idea for the construction of the PKC is to compose of easily invertible transforms. More formally, the key-maker chooses easily invertible transforms such as $F_1, F_2, \cdots, F_k$ and defines practically non-invertible transform F such that $F = F_1 \times F_2 \times \cdots \times F_k$. Here F will be made public as a public-key, while $F_1, F_2, \cdots, F_k$ will be stored as a secret-key. The above construction of PKC's, i.e., the composition of easily invertible transforms, is referred to as *OBSCURE REPRESENTATION* (denoted *OR*) [IM85]. This subsection will develop a PKC by OR introducing a new concept, called *SEQUEN-*

TIALLY SOLVABLE (denoted *SS*) transform [TKIFM86,TKIFM87].

DEFINITION 7.3.1 [TKIFM86,TKIFM87] *Let* $f:X^n \to Y^n$ *be an 1 to 1 transform such that* $y=(y_1,y_2,\cdots,y_n)=f(x_1,x_2,\cdots,x_n)=f(x)$. *Then the transform is called* ***sequentially solvable*** *(denoted SS) iff* x_i *is given by* $x_i=f^{-1}(y;x_1,x_2,\cdots,x_{i-1})$ *for* $1\leqq i\leqq n$. □

Applying the above concept *SS* transform, we construct a PKC based on a system of non-linear equations as follows:

SYSTEM GENERATION 1:

SG1. Let $x,w\in GF(2^m)^n$ be a plaintext vector and an intermediate vector, respectively. For an $n\times n$ non-singular matrix A over $GF(2^k)$, define $w=Ax$, where $GF(2^k)$ is a subfield of $GF(2^m)$, i.e., $k\mid m$;

SG2. Define $v\in GF(2^m)^n$ such that $v=F(w)\cdot w$, where

$$F(w)=\begin{bmatrix} f_{11}(w_1) & f_{12}(w_2,\cdots,w_n) & f_{13}(w_2,\cdots,w_n)\cdots f_{1n}(w_2,\cdots,w_n) \\ 0 & f_{22}(w_2) & f_{23}(w_3,\cdots,w_n)\cdots f_{2n}(w_3,\cdots,w_n) \\ \vdots & 0 & \vdots \\ \vdots & \vdots & \vdots \quad \cdots\cdots\cdots \quad \vdots \\ 0 & 0 & 0 \qquad\qquad f_{nn}(w_n) \end{bmatrix},$$

$$f_{ii}(w_i)=(a_iw_i+b_i)/w_i(c_iw_i+d_i),$$

$$f_{ij}(w_{i+1},\cdots,w_n)=\left(\sum_{t=i+1}^{n} a_{ijt}w_t+b_{ij}\right)/\left(\sum_{s=i+1}^{n} c_{ijs}w_s+d_{ij}\right).$$

Note that every coefficient of f_{ij}'s $\in GF(2^k)$;

SG3. Let $y\in GF(2^m)^n$ be a ciphertext vector and let B be an $n\times n$ non-singular matrix over $GF(2^k)$. Define $y=Bv$;

SG4. *Public-Key* will be

$$y_i=\sum_{j=1}^{r} g_{ij}(x_1,x_2,\cdots,x_n)/d_i(x_1,x_2,\cdots,x_n),$$

where g_{ij}, d_i are polynomials of degree 1 of $x_1,x_2,\cdots,x_n$

and r=n(n+1)/2, and *Secret-Key* will be B^{-1}, F(w) and A^{-1}. □

Remark 7.3.2 From the construction of F(w), it is clear that F(w)·w is SS. (See S2 in *SYSTEM GENERATION 1.*) □

ENCRYPTION:

For the plaintext $x \in GF(2^m)^n$, substitute x to the public-key and compute the ciphertext y. □

DECRYPTION:

D1. For the ciphertext y, compute $v=B^{-1}y$;

D2. Compute w from v. Note that v=F(w)·w is SS;

D3. For w, compute the plaintext $x=A^{-1}w$. □

Unfortunately *Hasegawa* and *Kaneko* have found an efficient attacking method for the above scheme [HK87]. Here we present a stronger version of SS PKC introducing the generalized concept, called *partially sequentially solvable* (denoted *PSS*) transform.

DEFINITION 7.3.3 [TFH88a,TFH88b] *Let* $f:X^n \to Y^n$ *be 1 to 1 transform such that* $y=(y_1,y_2,\cdots,y_n)=f(x_1,x_2,\cdots,x_n)=f(x)$. *Then the transform f is called partially sequentially solvable (denoted PSS) iff for i* $(1 \leqq i \leqq n)$, $(x_1,x_2,\cdots,x_i)$ *is given by* $(x_1,x_2,\cdots,x_i)=f^{-1}(y)$ *and for* $i+1 \leqq j \leqq n$ x_j *is given by* $x_j=f^{-1}(y;x_1,x_2,\cdots,x_{i-1})$. □

It is clear that the above concept PSS is the generalization of SS. Here we will show the example of PSS transform as follows:

EXAMPLE 7.3.4:

Let $v,w \in GF(2^m)^n$. Here we define the following relation between (w_{n-1},w_n) and (v_{n-1},v_n):

$$v_n = \frac{a_1 w_n + a_2 w_{n-1} + a_3}{a_4 w_n + a_5 w_{n-1} + a_6},$$

$$v_{n-1} = \frac{b_1 w_n + b_2 w_{n-1} + b_3}{b_4 w_n + b_5 w_{n-1} + b_6}.$$

Then w_{n-1} and w_n can be given by v_{n-1} and v_n as follows:

$$w_n = \frac{d_1(v_{n-1}, v_n)}{d_0(v_{n-1}, v_n)},$$

$$w_{n-1} = \frac{d_2(v_{n-1}, v_n)}{d_0(v_{n-1}, v_n)},$$

where $d_i(v_{n-1}, v_n)$ $(0 \leqq i \leqq 2)$ are defined by

$$d_0 = \begin{vmatrix} a_4 v_n + a_1 & a_5 v_n + a_2 \\ b_4 v_{n-1} + b_1 & b_6 v_{n-1} + b_2 \end{vmatrix},$$

$$d_1 = \begin{vmatrix} a_6 v_n + a_3 & a_5 v_n + a_2 \\ b_6 v_{n-1} + b_3 & b_5 v_{n-1} + b_2 \end{vmatrix},$$

$$d_2 = \begin{vmatrix} a_4 v_n + a_1 & a_6 v_n + a_3 \\ b_4 v_{n-1} + b_1 & b_6 v_{n-1} + b_3 \end{vmatrix}.$$

Thus we have the equivalent representation as follows:

$$(w_{n-1}, w_n)^T = K(v_{n-1}, v_n)(v_{n-1}, v)^T,$$

where $K(v_{n-1}, v_n)$ is given by

$$K(v_{n-1}, v_n) = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix},$$

$$k_{11} = \frac{(a_1 b_6 + a_3 b_4) v_{n-1} + a_6 b_4 v_{n-1} v_n + a_3 b_1}{v_{n-1} \cdot d_0(v_{n-1}, v_n)},$$

$$k_{12} = \frac{(a_4 b_3 + a_6 b_1) v_n + a_4 b_6 v_{n-1} v_n + a_1 b_3}{v_n \cdot d_0(v_{n-1}, v_n)},$$

$$k_{21} = \frac{(a_3 b_5 + a_2 b_6) v_{n-1} + a_6 b_5 v_{n-1} v_n + a_2 b_3}{v_{n-1} \cdot d_0(v_{n-1}, v_n)},$$

$$k_{22} = \frac{(a_6 b_2 + a_5 b_3) v_n + a_5 b_6 v_{n-1} v_n + a_3 b_2}{v_n \cdot d_0(v_{n-1}, v_n)}.$$

Here we refer to the above transform $K(\cdot, \cdot)$ as the *Kernel* of

PSS transform. □

Here we present the construction of the PSS transform. Define the following n× n matrix F(w):

$$F(v)=\begin{bmatrix} f_{11} & 0 & \cdots\cdots & 0 \\ 0 & f_{22} & 0 \cdots & 0 \\ \vdots & 0 & \cdots\cdots & 0 \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \\ 0 & 0 & K(v_{n-1},v_n) & \end{bmatrix},$$

where f_{ii} $(1\leqq i\leqq n-2)$ are functions both of degree 1 of v_i and of higher degree of $v_{i+1},\cdots,v_n$ and K is the *Kernel* of the PSS transform. For example, define $w=F(v)\cdot v$ for $w,v\in GF(2^m)^n$, where

$$f_{ii}(v)=\frac{a_i(v)\cdot v_i+b_i(v)}{v_i\cdot d_0(v_{n-1},v_n)}\ (1\leqq i\leqq n-2),$$

$a_i(v)$: linear polynomial of $v_{i+1},\cdots,v_n$,

$b_i(v)$: quadratic polynomial of $v_{i+1},\cdots,v_n$.

It is not difficult to see that the transform $w=F(v)\cdot v$ is PSS. Then we construct a PKC applying the above PSS transform.

SYSTEM GENERATION 2:

SG1. Let $x\in GF(2^m)^n$ be a plaintext. For an n× n non-singular matrix A over $GF(2^k)$, define intermediate vector $v\in GF(2^m)^n$ by v=Ax, where $GF(2^m)$ is a subfield of $GF(2^m)$, i.e., k| m;

SG2. Define an intermediate vector w by $w=F(v)\cdot v$, where the transform $w=F(v)\cdot v$ is PSS;

SG3. For an n× n invertible matrix B over $GF(2^k)$, define an intermediate vector u=Bw;

SG4. Define an intermediate vector z by $z=G(u)\cdot u$, where the transform $z=G(u)\cdot u$ is PSS;

SG5. For an $n\times n$ invertible matrix C over $GF(2^k)$, define the ciphertext $y=Cz$;

SG6. *Public-Key* will be $y=CG(u)BF(v)Ax$, thus it is given by $y_i=r_i(x)/r_d(x)$ $(1\leqq i\leqq n)$, where $r_i(x)$ and $r_d(x)$ are polynomials of degree 4 of $x_1, x_2, \cdots, x_n$, and *Secret-Key* will be A,B,C,F(v) and G(u). □

ENCRYPTION:

For the plaintext $x\in GF(2^m)^n$, substitute x to the public-key and compute the ciphertext y. □

DECRYPTION:

D1. For the ciphertext$\in GF(2^m)^n$, compute $z=C^{-1}y$;

D2. Compute u from z, where $z=G(u)\cdot u$ is PSS;

D3. Compute w by $w=B^{-1}u$;

D4. Compute v from w, where $w=F(v)\cdot v$ is PSS;

D5. Compute the plaintext x by $x=A^{-1}v$. □

7.3.2 ANALYSIS ON THE PKC

This subsection proposes possible attacks and analyzes the security of the PKC.

Recalling *SYSTEM GENERATION 2*, the PKC can be regarded as the composition of two PSS transforms. Here we develop an attacking method (denoted *AM1*) to find the intermediate vector u from the ciphertext y. This implies that the AM1 inverts the first PSS transform. The primary principle of the AM1 can be simply described as follows: Assume that every coefficient of

each secret-key is an unknown variable and we reconstruct the PKC from the secret-key with unknown variables. Then we compare the reconstructed PKC with the given public-key and evaluate the coefficients of each secret-keys.

Here we analyze the running cost of AM1 in the case that n=5. Then we have the following Theorem.

THEOREM 7.3.4 [TFH88a,TFH88b] *To find the Kernel of PSS transform $z=G(u)\cdot u$, AM1 requires to solve a system of 126 quartic equations of 71 variables and a system of 126 quadratic equations of 14 variables.* □

Remark 7.3.5 The proof of Theorem 7.3.4 is not so difficult, however, somewhat complicated. (See [TFH88a].) □

Assume that AM1 succeeds to find the Kernel of the PSS transform $z=G(u)\cdot u$. Then the following Theorem shows the running cost to find the coefficients of the PSS transform and C.

THEOREM 7.3.6 [TFH88a,TFH88b] *Assume that the Kernel of the PSS transform $z=G(u)\cdot u$ is known. Then, to find the all coefficients of the PSS transform that $z=G(u)\cdot u$ and C, AM1 requires to solve three systems of 126 quadratic equations of at most 46 variables.* □

Remark 7.3.7 The proof of Theorem 7.3.6 is also easy, however, somewhat complicated. (See [TFH88a].) □

Thus we can conclude that the proposed PKC is *secure* against the above attacking method AM1.

Furthermore we will demonstrate another attacking method (denoted *AM2*). The attacking method AM2 is suggested by *Okamoto* and *Nakamura* [ON86] and the primary principle of AM2 can be

briefly summarized as follows: Notice the decryption procedure of the PKC and estimate the degree of the inverse transform of the PKC. The degree of the inverse transform gives the total number of terms N of the inverse transform. Represent the inverse transform with N unknown variables. Then substitute arbitrary plaintexts to the public-key and compute the ciphertexts until N linearly independent equations will be obtained. Thus we have the following Theorem in the case that n=5.

THEOREM 7.3.8 [TFH88a,TFH88b] *To find the inverse transform of the PKC, AM2 requires to solve a system of linear equations of at least* 4.225×10^{13} *unknown variables.* □

Remark 7.3.9 Theorem 7.3.8 can be easily proven by a simple combinatorial discussion. See [TFH88a] on details. □

Thus we can conclude that the proposed PKC is *secure* against the attacking method AM2.

7.3.3 DISCUSSION

This Section has developed a PKC based on the difficulty of solving a system of non-linear equations over $GF(2^m)$. In order to realize the PKC, we have defined a new concept, called *partially sequentially solvable* (denoted *PSS*) transform. Furthermore we have demonstrated possible attacks for the PKC and have analyzed the security of the PKC. Hence we can conclude that the PKC is supposed to be secure against possible attacks.

The proposed PKC has the following preferable features:

F1. The encryption and the decryption can be carried out in $O(m^2)$ [TAI87], where $m=|x|$;

F2. The public-key size and the secret-key size are comparatively small.

The following table illustrates the public-key size (denoted *PKS*) and secret-key size (denoted *SKS*), respectively.

TABLE 7.2.10 PKS and SKS of the PKC

n	k	PKS [Kbit]	SKS [Kbit]
5	8	6.05	1.48
6	8	11.76	2.18

7.4 PKC BASED ON THE FACTORIZATION

This Section presents a PKC based on the intractability of factoring the product of two large primes [KIT87,KIT88a,KIT88b]. The PKC has a remarkable property that inverting the PKC is equivalent to factoring the composite moduli. Thus the PKC is *provable secure* under the assumption that factoring is hard. Furthermore the PKC in this Section has a simple structure and a preferable feature that the ciphertexts are uniquely decryptable. The related works can be found in [Ra79,Wil80,Wil85].

7.4.1 PKC BASED ON THE FACTORIZATION

In this subsection, we present a PKC based on the intractability of factoring a large composite number. In order to show the construction of the PKC, we provide some mathematical definitions, lemmas and the related works.

DEFINITION 7.4.1 [Kra86] *Call an* $x \in Z_m$ *quadratic residue*

(mod m), if $x=y^2$ *(mod m) for some y; otherwise* $x \in Z_m$ *is quadratic non-residue (mod m).* □

Remark 7.4.2 Throughout this Section, the set of all quadratic (non) residues (mod m) is denoted by QR_m (QNR_m). □

DEFINITION 7.4.3 [Kra86] *Let p be an odd prime number and let* $x \not\equiv 0$ *(mod p). Define the following symbol such that*

$(x/p)=1$, $x \in QR_p$,

$(x/p)=-1$, $x \in QNR_p$.

Here (x/p) is called Legendre symbol of x mod p. □

DEFINITION 7.4.4 [Kra86] *Let* $m=p_1 p_2 \cdots p_r$, *where* p_i $(1 \leqq i \leqq r)$ *are primes, not necessarily distinct. Then Jacobi symbol (x/m) is defined by* $(x/m)=(x/p_1)(x/p_2)\cdots(x/p_r)$, *where* (x/p_i) $(1 \leqq i \leqq r)$ *denote Legendre symbol.* □

Here we describe the *Rabin's Scheme* (denoted *RS*) [Ra79] and the *Williams Scheme* (denoted *WS*) [Wil80] as follows:

RABIN's SCHEME (RS):

Secret-Key Two large primes p and q;

Public-Key N(=pq),d;

Plaintext m (0< m< N);

Encryption $c=m(m+d) \pmod N$;

Decryption Solve the following two quadratic equations:

$x^2+dx-c=0 \pmod p$,

$x^2+dx-c=0 \pmod q$.

Applying *Chinese Remainder Theorem* [Kra86], evaluate the four roots of $x^2+dx-c=0 \pmod N$ and find the original plaintext m. □

Rabin has proven that inverting RS is as hard as factoring

N [Ra79]. However in RS, the ciphertexts cannot be uniquely decrypted, because we have no way to identify the original plaintext m in the four possible roots of $x^2+dx-c=0 \pmod N$.

WILLIAMS' SCHEME (WS):

Secret-Key Two large primes p and q, where $p=q=3 \pmod 4$;

Public-Key N(=pq)

Plaintext m, where $0< m< N/2$ and $(m/N)=1$;

Encryption $c=m^2 \pmod N$;

Decryption Solve the following two quadratic equations:

$$x^2-c=0 \pmod p,$$
$$x^2-c=0 \pmod q.$$

Applying the Chinese Remainder Theorem, evaluate the four roots of $x^2-c=0 \pmod N$ and find the original plaintext m such that $0< m< N/2$ and $(m/N)=1$. □

It can be shown that inverting WS is equivalent to factoring N(=pq) [Wil80]. In addition, in WS the ciphertexts are uniquely decryptable [Wil80]. Here we will present a PKC based on the intractability of factoring a large composite number [KIT87,KIT88a,KIT88b]. Throughout this Section, we will refer to the PKC [KIT87,KIT88a,KIT88b] as *KIT*.

SYSTEM GENERATION:

SG1. Choose arbitrary two large primes p and q and compute the product N(=pq);

SG2. Choose an arbitrary d such that $(d/p)=(d/q)=-1$;

SG3. *Public-Key* will be N(=pq) and d, and *Secret-Key* will be p and q. □

ENCRYPTION:

For the plaintext m, where $0 \leq m < N$ and $GCD(m,N)=1$, compute $c=m+dm^{-1} \pmod N$. If $(m/N)=1$, then $s=0$; otherwise define $s=1$. Furthermore if $dm^{-1} \pmod N > m$, then $t=0$; otherwise define $t=1$. Then the ciphertext c is given by $c=(c,s,t)$. □

DECRYPTION:

D1. Solve the quadratic equation $x^2-cm+d=0 \pmod N$ and compute the four roots m_{ij} $(i,j \in \{+,-\})$ such that

$$m_{++}=[m_{p+},m_{q+}],$$
$$m_{--}=[m_{p-},m_{q-}],$$
$$m_{+-}=[m_{p+},m_{q-}],$$
$$m_{-+}=[m_{p-},m_{q+}],$$

where $m_{+-}=[m_{p+},m_{q-}]$ shows that $m_{+-}=m_{p+} \pmod p$, $(m_{+-}/p)=1$, $m_{+-}=m_{q-} \pmod q$ and $(m_{+-}/q)=-1$. (The other cases are similarly defined.)

D2. Find the original plaintext m in the possible four roots according to the values of "s" and "t" as follows:

TABLE 7.4.5 Rules of the Decryption of *KIT*

s \ t	0	1
0	$\min(m_{++},m_{--})$	$\max(m_{++},m_{--})$
1	$\min(m_{+-},m_{-+})$	$\max(m_{+-},m_{-+})$

The *validity* of the decryption for *KIT* will be shown in the following. □

THEOREM 7.4.6 [KIT87,KIT88a] *The decryption procedure described above identifies the original plaintext m in the*

. *four possible roots of $x^2-cx+d=0 \pmod N$.* □

Proof: Notice that m_{p+} and m_{p-} are the two possible roots of $x^2-cx+d=0 \pmod p$ while m_{q+} and m_{q-} are also the two possible roots of $x^2-cx+d=0 \pmod q$. Recalling the assumption that $(c/p)=-1$, it follows that $(m_{p+}/p)(m_{p-}/p)=(c/p)=-1$. Thus we have $(m_{p+}/p)=1$ and $(m_{p-}/p)=-1$. Similarly recalling that $(c/q)=-1$, it follows that $(m_{q+}/q)=1$ and $(m_{q-}/q)=-1$. Then we have $(m_{++}/N)=(m_{++}/p)(m_{++}/q)=(m_{p+}/p)(m_{q+}/q)=1\cdot 1=1$. Furthermore we have $(m_{--}/N)=1$ and $(m_{+-}/N)=(m_{-+}/N)=-1$. Hence the bit "s" enables us to identify that if s=0, then the original plaintext $m=m_{++}$ or m_{--}; otherwise $m=m_{+-}$ or m_{-+}. Here we assume that s=0. Then we have $m_{++}m_{--}=[m_{p+}m_{p-},m_{q+}m_{q-}]=[d,d]=d \pmod N$, where $m=[x,y]$ denotes that $m=x \pmod p$ and $m=y \pmod q$. It follows that $m_{--}=d(m_{++})^{-1} \pmod N$. Hence the bit "t" enables us to identify that if t=0 then the original plaintext $m=\min(m_{++},m_{--})$; otherwise $m=\max(m_{++},m_{--})$. In the case that s=1, the bit "t" also enables us to identify that if t=0, then $m=\min(m_{+-},m_{-+})$; otherwise $m=\max(m_{+-},m_{-+})$. Thus KIT uniquely decrypts the ciphertexts. □

Here we will show the following example to demonstrate the *validity* of KIT [KIT87,KIT88a].

EXAMPLE 7.4.7:

Let that p=11, q=13 and d=2. Then it follows that N=143 and $(2/11)=(2/13)=-1$.

Encryption: Let m=24. Then $c=24+2\cdot 24^{-1} \pmod{143}=36$. By noting $(24/143)=1$ and $2\cdot 24^{-1} \pmod{143}=12<24(=m)$, we have s=0 and t=1. Thus the ciphertext is c=(34,0,1). □

Decryption: Solve $x^2-36x+2=0 \pmod{11}$. Then we have $m_{11+}=1$ and $m_{11-}=2$. Furthermore the roots of $x^2-36x+2=0 \pmod{13}$ are $m_{13+}=12$ and $m_{13-}=11$. Noting that $s=0$, it follows that $m=m_{++}=[1,12]=12$ or $m=m_{--}=[2,11]=24$. In addition, since $t=1$, then we have $m=\max(m_{++},m_{--})=24$. □

7.4.2 ANALYSIS ON *KIT*

This subsection proves that inverting KIT is intractable as factoring the composite moduli N [KIT87,KIT88a]. To show the equivalency of inverting KIT and factoring the composite moduli $N(=pq)$, we must prove the following two statements:

S1. *If there exists a polynomial time algorithm FACT to factor the composite moduli $N(=pq)$, then there exists a polynomial time algorithm INV to find the plaintext for any given ciphertext;*

S2. *If there exists a polynomial time algorithm INV to find the plaintext for any given ciphertext, then there exists a polynomial time algorithm FACT to factor the composite moduli $N(=pq)$.*

It is not so difficult to see that S1 is correct. Thus we concentrate our interest on the proof of S2. To prove S2 in the above, we present the following auxiliary lemma:

LEMMA 7.4.8 [KIT87,KIT88a,KIT88b] *Neither $x^2-cx+d=0 \pmod{p}$ nor $x^2-cx+d=0 \pmod{q}$ has a multiple root.* □

Proof: Notice that m_{p+} and m_{p-}, the two possible roots of $x^2-cx+d=0 \pmod{p}$, satisfy $(m_{p+}/p)=1$ and $(m_{p-}/p)=-1$, respectively. Thus it follows that $m_{p+} \neq m_{p-}$. In the similar way, we

can prove that $m_{q+} \neq m_{q-}$. □

The following Theorem formulates the main result in this Section, i.e., the equivalency of inverting KIT and factoring the composite moduli N(=pq).

THEOREM 7.4.9 [KIT87,KIT88a] *If there exists a polynomial time algorithm INV to find the plaintext for any ciphertext of KIT, then there exists a probabilistic polynomial time algorithm FACT to factor the composite moduli N(=pq) with probability 1/4.* □

Proof: Choose at random d (0< d< N). Note that d satisfies (d/p)=(d/q)=-1 with probability 1/4. Let (N,d) be a public-key of KIT. For any plaintext m, evaluate the ciphertext c=(c,s,t). Define the ciphertext c'=(c,s',t), where s'=s+1 (mod 2), and compute the plaintext m' for c' by the polynomial time algorithm *INV*. Assume that $m=[m_{p+},m_{q+}]$. Since s'=s+1 (mod 2), we have $m'=[m_{p+},m_{q-}]$ or $m'=[m_{p-},m_{q+}]$. Here we consider the case that $m'=[m_{p+},m_{q-}]$. (For the case that $m'=[m_{p-},m_{q+}]$, the proof is the same.) Then $m-m'=[m_{p+},m_{q+}]-[m_{p+},m_{q-}]=[0,m_{q+}-m_{q-}]$. Recalling Lemma 7.4.8, we have $m_{q+}-m_{q-} \neq 0 \pmod q$, and this implies that $m-m'=0 \pmod p$ and $m-m' \neq 0 \pmod q$. Hence it follows that GCD(m-m',N)=p. Note that randomly chosen d (0< d< N) satisfies (d/p)=(d/q)=-1 with probability 1/4 and the algorithm *INV* can be carried out in polynomial time. Thus the above algorithm factors the composite moduli N(=pq) in polynomial time with probability 1/4. □

Remark 7.4.10 The probability that the above probabilistic algorithm fails after 100 trials is $(3/4)^{-100} \simeq 10^{-13}$,

which is *negligibly small.* □

The following Theorem is the stronger version of Theorem 7.4.9.

THEOREM 7.4.11 [KIT87,KIT88a] *If there exists a polynomial time algorithm INV to find the plaintext for 1/k of all the ciphertexts of KIT, then there exists a probabilistic polynomial time algorithm FACT to factor the composite moduli N(=pq) with probability 1/4k.* □

Proof: Let X be the set of ciphertexts for which the polynomial time algorithm *INV* can be applied. Choose at random a plaintext m ($0< m< N$). The ciphertext for m belongs to X with probability 1/k. Thus we can apply the same algorithm in the proof of Theorem 7.4.9 to the plaintext m. Hence we can construct a polynomial time algorithm *FACT* to factor the composite moduli N(=pq) with probability 1/4k. □

The following shows the construction of the algorithm *FACT* employing the polynomial time algorithm *INV*.

ALGORITHM 7.4.12:

S1. *Input N;*
S2. *Choose at random d (0< d< N);*
S3. *Choose at random m (0< m< N);*
S4. *Encrypt m and compute the ciphertext c=(c,s,t);*
S5. *Define c'=(c,s',t), where s'=s+1 (mod 2);*
S6. *Decrypt c' and compute the plaintext m' by INV;*
S7. *Compute Fact=GCD(m-m',N);*
S8. *If Fact=1 or N goto S2;*
S9. *Output Fact.*

□

7.4.3 DISCUSSION

This Section has developed a PKC based on the hardness of factoring a large composite number (denoted *KIT*) [KIT87,KIT88a, KIT88b] and in addition has proven that inverting KIT is equivalent to factoring the composite moduli N(=pq) [KIT87,KIT88a]. The PKC *KIT* has a simple structure and a preferable feature that the ciphertexts are uniquely decryptable.

The encryption of KIT is composed of three parts; The first part is the computation of the function for the plaintext m, i.e., $c=m+dm^{-1} \pmod N$, and the second part is the evaluation of the bit "s", i.e., the evaluation of (m/N). In addition, the third part is the evaluation of the bit "t", i.e., the comparison of m and $dm^{-1} \pmod N$. The running cost of the first and third part is $O_B(m^2)$, respectively, where $|N|=m$ and $O_B(\cdot)$ denotes the order of bit operations. Furthermore the running cost of the second part is also $O_B(m^2)$. Thus the running cost of the encryption is $O_B(m^2)$.

On the other hand, the decryption of KIT is composed of two parts; The first part finds the roots of $x^2-cx+d=0 \pmod N$, and the second part identifies the original plaintext m in the four possible roots of $x^2-cx+d=0 \pmod N$ by the bit "s" and "t". To find the roots of $x^2-cx+d=0 \pmod N$, the first part solves the two quadratic equations $x^2-cx+d=0 \pmod p$, $x^2-cx+d=0 \pmod q$ by $GCD\{x^{(p-1)/2}-1, x^2-cx+d\}$ over $GF(p)$, $GCD\{x^{(q-1)/2}-1, x^2-cx+d\}$ over $GF(q)$, respectively. Then each equation has distinct two roots m_{p+}, m_{p-} and m_{q+}, m_{q-}, respectively, we can find the four

distinct roots of $x^2-cx+d=0 \pmod N$ by *Chinese Remainder Theorem* [Kra86]. Note that the running cost of GCD(·,·) is $O_B(m^3)$ and that of Chinese Remainder Theorem is $O_B(m^2)$. Thus the running cost of the first part is $O_B(m^2)$. It is easy to show that the running cost of the second part is $O_B(m^2)$, thus the running cost of the decryption is $O_B(m^3)$.

The following table summarize the running cost of the encryption and the decryption for several PKC's.

TABLE 7.4.13 Running Cost of PKC's

	KIT	Rabin & Williams	RSA
Encryption	$O_B(m^2)$	$O_B(m^2)$	$O_B(m^2)$ for small e $O_B(m^3)$ for large e
Decryption	$O_B(m^3)$	$O_B(m^3)$	$O_B(m^3)$

7.5 IDENTITY-BASED CRYPTOSYSTEM

This Section presents an *Identity-Based Cryptosystem* based on discrete logarithm problem [TIK87,TI88]. The identity-based cryptosystem (denoted *IDC*) can be materialized by employing ElGamal's PKC [El85a] and is supposed to be one of the earliest concrete examples of the IDC *strictly* in Shamir's sense.

In the recent large-scale telecommunication network, it is required that each entity is able to communicate with arbitrary entities through the network. In the light of the data protection, cryptography is one of the most promising tools for it in such a system. In both private-key cryptosystems and public-key

cryptosystems, however, key management is a *serious* problem in the case that the system includes many entities.

In 1984, *A.Shamir* definitely pointed out the significance of Identity-Based System (denoted *IDS*) [Sha84], i.e., Identity-Based Cryptosystem (denoted *IDC*) and Identity-Based Signature Scheme (denoted *IDSS*), as one of the countermeasures for the key management problem in the large-scale network, and developed a concrete example for IDSS. On the other hand, in 1984 *T.Okamoto* independently proposed a concrete example of the IDSS [Oka84, Oka86] at almost the same time. Kohnfelder and Blom presented the similar ideas in [Ko78,Bl82], however, they did not positively suggest the importance of IDS. Thus the development of the IDS can be regarded as a *natural* requirement for the reduction of the key management problem in a large-scale network.

Then IDC can be categorized such as IDC [Ko78,DQ86,TIK87, TI88], IDSS [Oka84,Sha84,FS86,Oka86,FFS87] and ID-Based Key Distribution System (denoted *IDKDS*) [Bl82,KO87,MI87,Oka87,Ta87]. In IDC, we have already had two concrete examples such as Kohnfelder's scheme [Ko78] and Desmedt and Quisquater's scheme [DQ86]. In [Ko78], the interactive preliminary communications are necessary in each data transmission, while in [DQ86], each entity must allow the assumption that the tamper-free modules are available. (This assumption seems to be too *strong* in the light of the modern technologies!)

In this Section, we present Identity-Based Cryptosystem based on the hardness of discrete logarithm problem [TIK87,TI88] employing ElGamal's PKC [El85a]. The basic idea of the IDC can

be found in [TIK87]. This Section includes the further studies such as the security of the IDC, the identification for senders, the reduction of processing cost, etc. Our scheme does not require any interactive preliminary communications in each message transmission and any assumptions except the intractability of discrete logarithm problem. (This assumption seems to be quite reasonable!) Thus the proposed scheme is one of the earliest realizations for IDC, which satisfies Shamir's original concept [Sha84] in a *strict* sense.

7.5.1 IDC BASED ON DISCRETE LOGARITHM PROBLEM

Here we show the ElGamal's PKC [El85a] in the following:

ElGamal's PKC:

Public-Key

p : a large prime number;
g : a generator of the multiplicative group in $GF(p)$;
$z=g^s \pmod p$ $(0 \leqq s \leqq p-2)$.

Secret-Key

s : an arbitrary constant number $(0 \leqq s \leqq p-2)$.

Encryption Let m $(0 \leqq m \leqq p-2)$ be a plaintext to be transmitted. The Sender chooses a random number r $(0 \leqq r \leqq p-2)$ and computes the ciphertext $C=(c_1, c_2)$, where $c_1=g^r \pmod p$ and $c_2=mz^r \pmod p$. □

Decryption The receiver computes

$$\begin{aligned}(c_1)^s &= (g^r)^s \pmod p \\ &= (g^s)^r \pmod p \\ &= z^r \pmod p,\end{aligned}$$

and recovers the message m by

$$m=\{(c_1)^s\}^{-1}c_2 \pmod p$$
$$=(z^r)^{-1}mz^r \pmod p.$$

The above transform can be performed only by the receiver knowing the secret "s". (The above notations will be used throughout this Section.) □

By modifying the ElGamal's PKC, we can develop an IDC based on discrete logarithm problem. Here we will present the detailed descriptions for the construction of the IDC.

CONSTRUCTION OF THE IDC:

The construction of the IDC is composed of two parts; The first part is the <u>preparation</u> of the *Trusted Center* (denoted *TC*) and *Each Entity* for the establishment of the IDC. The second part prescribes the <u>protocol</u> of the *protected data transmission*. The TC is located only for the establishment of the IDC, thus the TC can be closed after that except for the case of the *re-establishment* of the IDC, the state of *emergency*, etc.

<u>*Subscription to the IDC*</u>

Each entity generates k dimensional binary vector for his identity (denoted *ID*). Define entity i's ID by ID_i as follows:

$$ID_i=(x_{i1},x_{i2},\cdots,x_{ik}),\ x_{ij}\in\{0,1\}\ (1\leqq j\leqq k).$$

All entities registers their ID's to the TC and the TC stores them in the public file. The TC publishes an 1 to 1 one way function $f(\cdot)$, e.g., RSA. Any entity can evaluate the entity i's extended ID, EID_i by

$$EID_i=f(ID_i)=(y_{i1},y_{i2},\cdots,y_{in}),\ y_{ij}\in\{0,1\}\ (1\leqq j\leqq n).$$

The extended ID (denoted *EID*) plays a central role of the

countermeasure against conspiracy of some entities. Details on EID will be discussed in Subsection 7.5.2. The TC chooses an arbitrary large prime number p, e.g., $|p|=512$, and also generates an n dimensional vector a such that $a=(a_1, a_2, \cdots, a_n)$, where $1 \leqq a_i \leqq p-2$ $(1 \leqq i \leqq n)$, and stores it as the TC's Secret Information. Note that for n dimensional binary vectors I and J, the vector a must satisfy the following claim:

$$a \cdot I \not\equiv a \cdot J \pmod{p-1}, \qquad (7.1)$$

to avoid the accidental coincidence of some entities' secret-key. One of the most simple way to generate the vector a is to use Merkle and Hellman's Scheme [MH78]. Then the TC chooses a *super-increasing* sequence $\{a_i'\}$ $(1 \leqq i \leqq n)$ such that

$$\sum_{i=1}^{n} a_i' < p-1.$$

The TC also chooses w satisfying $GCD(w, p-1)=1$, and defines a vector $a=(a_1, a_2, \cdots, a_n)$, where $a_i = a_i' \cdot w \pmod{p-1}$ $(1 \leqq i \leqq n)$.

Remark 7.5.1 It is clear that the above vector a satisfies the claim of Eq.(7.1). Note that the above scheme is just the one scheme to generate an n dimensional vector a satisfying the claim of Eq.(7.1). Here we adopt the above scheme, however, the other ones might be possible. □

The TC chooses an arbitrary generator g of the multiplicative group in $GF(p)$, and defines a vector $h=(h_1, h_2, \cdots, h_n)$ employing the generator g and the vector a, where $h_i = g^{a_i} \pmod{p}$ $(1 \leqq i \leqq n)$. The TC informs each entity of p, g and h as the common public informations in the network. The TC defines the entity i's secret-key s_i by the inner product of a and EID_i,

$s_i = a \cdot EID_i \pmod{p-1}$

$= \sum_{j=1}^{n} a_j y_{ij} \pmod{p-1}$.

The TC sends entity i's secret-key s_i to entity i through a highly secure channel.

System Parameters:

TC's Secret Information

a : n dimensional vector over Z_{p-1};

TC's Public Information

h : n dimensional vector over Z_p;

p : a large prime number;

f : 1 to 1 one way function, e.g., RSA;

g : a generator of the multiplicative group in $GF(p)$;

Entity i's Secret-Key

s_i $(0 < s_i < p-1)$;

Entity i's Public Information

ID_i : k dimensional binary vector;

After the establishment of the IDC, each entity exchanges the protected data by the following protocol:

Protocol of the IDC

Suppose that entity 2 sends his message m to entity 1.

Encryption:

Entity 2 generates the entity 1's EID by

$EID_1 = f(ID_i) = (y_{11}, y_{12}, \cdots, y_{1n})$.

In addition, entity 2 computes z_1 by

$z_1 = \prod_{i=1}^{n} h_i^{y_{1i}} = \prod_{i=1}^{n} (g^{a_i})^{y_{1i}} \pmod{p}$.

Note that $\sum_{1 \le i \le n} a_1 y_{1i} = s_1$, thus we have $z_1 = g^{s_1} \pmod{p}$. Hence

Entity 2 can use z_1 as z in ElGamal's PKC. Let m ($0 \leqq m \leqq p-1$) be an entity 2's message to be transmitted. Entity 2 generates r ($0 \leqq r \leqq p-2$) at random and computes the cipher text $C=(c_1, c_2)$, where $c_1 = g^r \pmod p$ and $c_2 = m(z_1)^r \pmod p$. Entity 2 sends the ciphertext C to entity 1. □

Decryption:

Entity 1 computes $(c_1)^{s_1} \pmod p = (g^r)^{s_1} \pmod p$ using his secret-key s_1. Recalling the definition of c_1, c_2 and z_1, entity 1 can recover entity 2's message by computing

$$\{(c_1)^{s_1}\}^{-1} c_2 = \{(g^r)^{s_1}\}^{-1} m(z_1)^r \pmod p$$
$$= \{(g^r)^{s_1}\}^{-1} m(g^{s_1})^r \pmod p$$
$$= m \pmod p.$$

The above transform can be carried out only by the entity knowing the secret "s_1". □

Here we describe the *informal* but *intuitive* explanation for the structure of the proposed IDC. The TC generates the *Secret-Information* **a**. The TC evaluates the subset sum of **a** according to each entity's EID and issues the subset sum of a to each entity as his secret-key. Then we have following two fundamental questions:

Q1. *If some entities disclose their secret-keys each other, can they find the TC's Secret Information? If possible, how many entities must conspire?*

Q2. *If some entities disclose their secret-keys each other, can they find the other entities' secret-keys? If possible, how many entities must conspire?*

In the subsequent subsections, we study the above ques-

tions and provide further practical improvements.

7.5.2 ANALYSIS ON THE IDC

This subsection analyzes the security of the proposed IDC and presents some remarks for secure system. The analyses on the proposed IDC are classified by *the security of the protected data transmission*, *that of the TC's Secret Information*, and *that of the arbitrary entity's secret-key*.

Security of the Protected Data Transmission:

The security of the proposed IDC is based on the intractability of discrete logarithm problem. Hence p must be chosen large enough, e.g., $|p| = 512$, and $(p-1)$ must have at least one large prime factor [PH78]. (If $(p-1)$ is the product of the small primes, then discrete logarithm problem will be solved in polynomial time [PH78].)

Security of TC's Secret Information:

Consider the case that $m(\geqq n)$ entities conspire to derive the TC's Secret Information a. For the notational simplicity, we assume that those entities are numbered as $1, 2, \cdots, m$. Each entity i's $(1 \leqq i \leqq m)$ has partial information of a in the form that $a \cdot EID_i = s_i \pmod{p-1}$ $(1 \leqq i \leqq m)$. Thus we have the system of linear congruences as follows:

$$\begin{bmatrix} EID_1 \\ EID_2 \\ \vdots \\ \vdots \\ EID_m \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ \vdots \\ s_m \end{bmatrix} \pmod{p-1} = D \cdot a \pmod{p-1}.$$

If the matrix D involves n linearly independent row vectors

over Z_{p-1}, then the TC's Secret Information a can be uniquely recovered by the m entities' conspiracy. However, *K.Nakamura et.al.* [NOTM87] and *D.Coppersmith* [Co87a] suggested that even in the case that matrix D does not involve n linearly independent row vectors over Z_{p-1}, m entities i's ($1 \leqq i \leqq m$) can derive a', which is equivalent to the original TC's Secret Information. Here we show the attacking method in the following:

ATTACK 7.5.2 [NOTM87,Co87a] *If m entities disclose their secret-keys each other, then they can derive an n dimensional vector a' over Z_{p-1}, which is equivalent to the TC's original Secret Information.* □

Proof (Sketch): Since the IDC proposed above is based on the intractability of discrete logarithm problem, (p-1) must have at least one large prime factor. For simplicity, we assume that $(p-1)=2^t q$, where $t << |p|$ and q is an odd prime. Without loss of generality, for some n entities among the m entities, we have the following matrix D' such that

$$D' = \begin{bmatrix} EID_1 \\ EID_2 \\ \vdots \\ \vdots \\ EID_n \end{bmatrix},$$

which satisfies $|D'| \neq 0 \pmod q$. (We assume that the entities are numbered as $1,2,\cdots,n$ for the notational simplicity.) Here we have an n dimensional vector $a'=(a_1', a_2', \cdots, a_n')$ over $GF(q)$, where $a_i' = a_i \pmod q$ $(1 \leqq i \leqq n)$. Define an arbitrary entity k's secret-key s_k' by $s_k' = EID_k \cdot a' \pmod p$. Recalling $a_i' = a_i \pmod q$ $(1 \leqq i \leqq n)$, s_k' is not necessarily identical to the entity k's

secret-key s_k, however, the difference of s_k and s_k' is some constant times of q.

Hence the entity k's original secret-key s_k can be derived at most in "t" trials, where $p-1=2^t q$ and $t<<|p|$. □

Thus up to (n-1) entities never can derive the TC's Secret Information by ATTACK 7.5.2, while more than n entities can find an n dimensional vector a', which is equivalent to the TC's Secret Information. If $m(>n)$ entities find the a', they can efficiently evaluate an arbitrary entity's secret-key, which implies that the proposed IDC is *completely* destroyed! Further improvements will described in Subsection 7.5.3.

Security of Each Entity's Secret-Key:

As discussed above, up to (n-1) entities never can derive the TC's Secret Information by the ATTACK 7.5.2 [NOTM87,Co87a], while more than n entities can compute a', which is equivalent to the TC's Secret Information. Here we consider the security of each entity's secret-key s_i against $t(<n)$ entities' conspiracy. If $t(<n)$ entities disclose their secret-key each other, they have the system of linear congruences such that

$$\begin{bmatrix} EID_1 \\ EID_2 \\ \vdots \\ \vdots \\ EID_t \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ \vdots \\ s_m \end{bmatrix} \pmod{p-1} = D'' \cdot \mathbf{a} \pmod{p-1},$$

where for the notational simplicity, we assume that the t entities are numbered as $1,2,\cdots,t$. If there exists a t dimensional vector $c=(c_1,c_2,\cdots,c_n)$ over Z_{p-1} such that for an entity k, $c \cdot D'' = \sum_{1 \le i \le t} c_i EID_i \pmod{p-1} = EID_k \pmod{p-1}$, $t(<n)$ entities

can compute the k's secret-key s_k by $s_k = \sum_{1 \le i \le t} c_i s_i \pmod{p-1}$. Noting that Z_{p-1} does not construct a field, it is easily shown that $t(< n)$ entities' conspiracy can generate at most 2^t other entities' secret-key by $\sum_{1 \le i \le t} d_i s_i \pmod{p-1}$ for some d_i's.

Hence the probability P_t that $t(< n)$ entities derive the other entities' secret-key is at most $2^t/2^n = 2^{t-n}$. Thus we have the following Theorem:

THEOREM 7.5.3 [TIK87,TI88] *If $t(< n)$ entities disclose their secret-keys each other, then the probability P_t that they can derive the other entities' secret-key will be bounded by $P_t < 2^{t-n}$.* □

Here we consider the necessity of the 1 to 1 one way function $f(\cdot)$ to define EID's. Assume that $t(< n)$ entities succeeds to find the entity k's secret-key s_k. Then they intend to make their spy with ID_k infiltrate into the system. By their conspiracy, they know the entity k's extended ID EID_k, however, it is practically impossible for t entities to evaluate ID_k because $EID_k = f(ID_k)$ and $f(\cdot)$ is one way function. Thus they cannot make an illegal entity infiltrate into the IDC and at the same time this implies that $t(< n)$ entities cannot disclose the specified entity's secret-key.

7.5.3 PRACTICAL IMPROVEMENTS

This subsection presents some practical improvements such that the construction of the the IDC over cyclic group Z_N, the identification for senders, the enhancement of the security and the processing cost, etc. The modification in this subsection

is of practical importance rather than of theoretical interest.

Construction of the IDC over Cyclic Group Z_N:

The IDC proposed above is based on the intractability of discrete logarithm over Z_p, where p is an odd prime. Here we consider an IDC based on finding index problem over cyclic group Z_N, where N=pq. (The factorization of N is known only to the TC.) By this minor change, the system in subsection 7.5.2 is slightly modified such that (p-1) is replaced by $\lambda(N)$, where $\lambda(N)$ is Carmichael function of N, and a generator g of the multiplicative group in Z_p is replaced by that in Z_N. For the modified IDC, *D.Coppersmith* showed an attacking method [Co87b], which derives the TC's Secret Information by the disclosure of secret-keys of (n+1) entities.

Attack 7.5.4 [Co87b] *If (n+1) entities disclose their secret-keys each other, then they can derive an n dimensional vector a' over Z_N, which is equivalent to the original TC's Secret Information.* □

Proof: For notational simplicity, we assume that (n+1) entities are numbered as $1,2,\cdots,n+1$. When (n+1) entities disclose their secret-keys each other, then they have the system of linear congruences such that

$$\begin{bmatrix} EID_1 \\ EID_2 \\ \vdots \\ \vdots \\ EID_{n+1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ \vdots \\ s_m \end{bmatrix} \pmod{\lambda(N)}.$$

Since each EID_i is an n dimensional vector over Z_N, there exists an (n+1) dimensional vector $c=(c_1, c_2, \cdots, c_{n+1})$ over integer ring

satisfying $\Sigma_{1\leq i\leq n+1} c_i EID_i = 0$. Thus we have $\Sigma_{1\leq i\leq n+1} c_i EID_i = 0 \pmod{\lambda(N)}$ and this implies that $\Sigma_{1\leq i\leq n+1} c_i s_i = A\lambda(N)$ for some integer A. If $A \neq 0$, then the (n+1) entities can have a constant times of $\lambda(N)$ and can factor the composite moduli N [Mil76]. Then the similar method with ATTACK 7.5.2 is applicable to the modified scheme, hence the TC's Secret Information can be derived by the disclosure of secret-keys of (n+1) entities. □

Identification for Senders:

The modified IDC in the above has almost the same security with the original IDC, however, in the modified IDC it is possible for each receiver to identify the senders introducing further modifications.

The further modified IDC is materialized as follows: Let e and d be an encryption and decryption-key of *RSA* for the TC, respectively. Each entity i receives $t_i = (ID_i)^d \pmod N$ from the TC and stores it secretly. The TC makes public the encryption key e to all entities. (Note that any entities does not know the decryption key d.) Furthermore the TC modifies the TC's Public Information h to $h' = (h_1', h_2', \cdots, h_n')$, where $h_i' = (g^{a_1})^e \pmod N$ $(1 \leqq i \leqq n)$. Assume that entity 2 transmits his message m to entity 1. Then encryption and decryption of the further modified IDC are defined as follows:

Encryption:

Entity 2 computes z_1 from EID_1 and h' such that

$$z_1 = \prod_{i=1}^{n} (h_i')^{y_{1i}} \pmod N = (g^{s_1})^e \pmod N,$$

Then entity 2 computes the ciphertext $C = (c_1, c_2)$ for message m,

where $c_1 = t_2 g^r \pmod N$ and $c_2 = m z_1^r \pmod N$. Note that $t_2 = (ID_2)^d \pmod N$ and r is a random number. Entity 2 sends the ciphertext C to entity 1. The above transform can be carried out only by the entity knowing "t_2". Thus the above scheme enables the entity 1 to identify entity 2. □

Decryption:

Using the secret-key s_1, entity 1 computes

$$\begin{aligned}\{(c_1)^e ID_2^{-1}\}^{s_1} &= (t_2 g^{re} ID_2^{-1})^{s_1} \pmod N \\ &= (ID_2^{-1} g^{re})^{s_1} \pmod N \\ &= (g^{re})^{s_1} \pmod N \\ &= z_1^r \pmod N,\end{aligned}$$

Thus entity 1 can recover the message m by computing

$$\begin{aligned}[\{(c_1)^e ID_2^{-1}\}^{s_1}]^{-1} c_2 &= (z_1^r)^{-1} m z_1^r \pmod N \\ &= m \pmod N.\end{aligned}$$

The above transform can be performed only by the entity knowing "s_1". □

By attaching some redundancies to the massage, e.g., name, address, password, etc., the further modified IDC enables the receivers to identify the senders because the decryption procedure is defined to use the sender's ID.

Enhancement of Security and Processing Cost:

The TC's Secret Information of the original IDC is derived by n entities' conspiracy. (See ATTACK 7.5.2.) Here we consider the practical countermeasure for the enhancement of the security of the IDC. For simplicity, assume that n=512 hereafter.

The TC partitions 512 dimensional binary vector B into 256 segments by every two bits such that

$$B = (b_1, b_2; b_3, b_4; \cdots; b_{511}, b_{512})$$

$=(seg_1, seg_2, \cdots, seg_{256})$,

and defines $a(i;jk)$ $(1 \leqq i \leqq 256, j,k \in \{0,1\})$. Then for each seg_i, the TC computes $h(i;jk)=g^{a(i;jk)} \pmod p$ $(1 \leqq i \leqq 256, j,k \in \{0,1\})$, and publishes the table including every $h(i;jk)$. Furthermore the TC computes each entity's secret-key $s_k = \Sigma_{1 \leq i \leq 256} a(i;seg_{ki}) \pmod p$ depending on the EID_i, where seg_{ki} implies *i-th* segment of EID_k. Then the TC distributes it to each entity through a highly secure channel. The following table, TABLE 7.5.5, shows the example of $h(i,jk)$'s.

TABLE 7.5.5 Example of h(i,jk)

h(1;00)= 5	h(2;00)=21	h(3;00)= 4	h(4;00)=16
h(1;01)=13	h(2;01)=17	h(3;01)=23	h(4;01)= 2
h(1;10)=12	h(2;10)= 7	h(3;10)=15	h(4;10)= 8
h(1;11)= 9	h(2;11)=11	h(3;11)=18	h(4;11)=22

Suppose that entity 2 sends his message m to entity 1.

Encryption:

Entity 2 computes $z_1' = \Pi_{1 \leq i \leq 256} h(i;seg_{1i})$ from EID_1 and the published table. Entity 2 makes use of z_1' as z_1 in the original IDC (in subsection 7.5.2.) to encrypt the message m. □

Decryption:

It can be carried out exactly in the same way with the original IDC in subsection 7.5.2. □

Recalling the original IDC, the TC's Secret Information is derived by 512 entities' conspiracy, whereas in the above IDC, it is derived by 1024(=4× 256) entities' conspiracy. Furthermore the running cost for the encryption-key generation of the

above IDC is about half of that in the original IDC, however, the TC's Public Information in the above IDC is about twice of that in the original IDC. Further generalizations, e.g., partitioned of each EID_i into 128 segments by every 4 bits, etc., are possible and such schemes are regarded as the hybrid system of the IDC and the conventional PKC.

7.5.4 DISCUSSION

This Section has developed an IDC based on the intractability of discrete logarithm problem [TIK87,TI88] employing ElGamal's PKC [El85a]. In addition, this Section has analyzed the security of the IDC and has presented several practical improvements for the *secure* and *efficient* system.

H.Tanaka has stated a *pessimistic* personal opinion on the establishment of the *robust* IDC against the conspiracy of entities [Ta88]. His opinion on the IDC is that the establishment of robust IDC's is impossible. We personally think that there exists a *jump of logic* in his discussion, thus we will not give wholehearted support to his personal view. Under existing circumstances, we have no robust examples of IDC in Tanaka's sense and expect that the establishment of such a scheme is *extremely* difficult. Thus the possibility for establishing robust IDC's must be theoretically studied.

7.6 CONCLUSION

In this Section, we have proposed several PKC's and an IDC and have studied the security against possible attacks. Here we

restate the results in this Section and describe further works.

Subsection 7.2 has analyzed the security of the *general* and *multiplicative* Knapsack-Type PKC's. For general Knapsack-Type PKC's, we have shown some sufficient conditions for Knapsack Problems to be solved in *linear* time [IKT84]. On the other hand, for the multiplicative Knapsack-Type PKC proposed by *Chor* and *Rivest* [CR84], we have developed an attacking method and have proven that the secret-key can be derived if the knapsack vector c includes at most three elements whose values are close to each other [KIST87]. Furthermore we have shown that the secret-key of the PKC is not unique [KIST87].

Subsection 7.3 has presented an PKC based on the difficulty of solving a system of non-linear equations [TKIFM86, TKIFM87,TFH88a,TFH88b]. In order to establish *secure* PKC's, we have defined a *Partially Sequentially Solvable* (denoted *PSS*) transform and a *Kernel* of the PSS transform [TFH88a,TFH88b]. We also have studied the security of the PKC and have shown that the PKC is secure against possible attacks.

Furthermore Subsection 7.4 has proposed a PKC based on the intractability of factoring a large composite number [KIT87, KIT88a,KIT88b]. We have surveyed the related works to the PKC, i.e., *Rabin's scheme* and *Williams' scheme*, and have demonstrated the advantages such as the uniqueness of the decryption, low running cost, etc. In addition, we have proven that inverting the PKC is hard as factoring the composite moduli. This implies that the PKC is *provably secure* under the assumption that fac-

toring a composite number is hard. *K.Kurosawa* has developed a *cryptographically strong pseudo-random sequence generator* [BM84, BBS86] by applying the PKC [Ku87] and he also has extensively studied the *k-th* bit security of the PKC [Ku88].

Finally Subsection 7.5 has developed an IDC based on the difficulty of discrete logarithm problem [TIK87,TI88] employing ElGamal's PKC [El85a]. We have studied the security of the IDC against the conspiracy of entities and have proven that $t(< n)$ entities' conspiracy never derive the TC's Secret Information while it finds the other entity's secret-key with probability at most 2^{t-n}. In addition, we have shown some practical improvements for establishing *secure* and *efficient* IDC.

Recent studies on IDS have found out that IDC could be materialized in two ways, i.e., one is the original IDC and the other is the composition of IDKDS and the private-key cryptosystem. We have already had several realizations for IDKDS [Bl82, KO87,MI87,Oka87,Ta87]. If a robust and efficient IDKDS would be developed, then the composed scheme of IDKDS and the private-key cryptosystem is more advantageous than the original IDC, because the secure and efficient private-key cryptosystems, e.g., DES and FEAL [SM87], are available. *Okamoto' scheme* [Oka87] is constructed by composing *Diffie and Hellman's Key Distribution System* [DH76] and the signature scheme by RSA. The scheme is not only efficient but also fairly robust against the conspiracy of entities under the *reasonable* assumptions, i.e., RSA is hard to invert and Diffie and Hellman's PKDS is secure. Thus it is supposed to be one of the most promising IDKDS's.

In the development of PKC'S and IDS's, the *robustness* and the *efficiency* are significant factors to be considered. For the robustness, careful analyses on the proposed schemes must be required, however, in the light of *secure communications*, it is the most preferable to show the equivalency to some intractable problems. On the other hand, the efficiency of the schemes in general is donimated by running costs, parallelizability, etc. Especially for the finite field arithmetics, most of the operations are composed of the simple ones such as additions, subtractions, multiplications and divisions. Thus the development of the fast algorithms and the compact configurations for those operations are *quite useful* to design high performance cryptographical instruments.

Hence the algorithms developed in Chapter Ⅲ ,Ⅳ ,Ⅴ and Ⅵ are expected to provide considerable efficiency being applied to the PKC's and IDS's.

CHAPTER VII :

CONCLUSIONS AND PERSPECTIVES

In this dissertation, we have studied several algorithms for finite field arithmetics such as multiplications, divisions, finding roots of quadratic equations and deciding quadratic residuosity, and have presented the *systematical* configurations of those algorithms.

Here we restate the main results in each Chapter.

In *CHAPTER III* , we have presented a parallel type multiplier for $GF(2^m)$ based on the canonical basis, which has less circuit size than a parallel type *Massey-Omura* Multiplier. In order to generalize the multiplier based on the canonical basis, we have defined a polynomial of a specified form, called *Equally Spaced Polynomial* (denoted *ESP*), and have shown a necessary and sufficient condition for the ESP's to be irreducible. Furthermore we have developed a parallel type multiplier based on the ESP's, which has the *structural* regularity and modularity.

In *CHAPTER IV* , we have proposed two kinds of fast division algorithms, i.e., one is *sequential* type and the other is *recursive* type. In addition we have shown several lemmas on the generation of normal bases and have developed the two types of the configuration for the recursive algorithm in $GF(2^m)$, both of which have *simultaneous size* $O(m^3)$ and *depth* $O((\log m)^2)$.

In *CHAPTER V* , we have presented an efficient probabilistic algorithms for solving quadratic equations over $GF(2^m)$ and

GF(p), and also have shown the advantageous feature, i.e., the uniform efficiency for the number of random trials.

In *CHAPTER VI* , we have developed an efficient decision algorithm for quadratic residuosity in $GF(p^m)$, where p is an odd prime and $m \geq 2$. (The study is *strongly* related to that in *CHAPTER V* .) Furthermore we have compared the performance with that of the other algorithms and have verified the *availability* of the proposed decision algorithm.

In *CHAPTER VII* , we have analyzed *Knapsack-Type* PKC's and have shown the conditions for the design of *secure* PKC's. Furthermore we have presented two types of PKC's, i.e., the PKC based on a system of non-linear equations and the PKC based on factorization of a large composite number, and have developed *identity-based cryptosystem* (denoted *IDC*) based on the hardness of discrete logarithm problem. For the PKC based on a system of non-linear equations, we have considered some possible attacks and have shown that the PKC is *secure* against the demonstrated possible attacks. For the PKC based on factorization of a large composite number, we have proven that inverting the PKC is as hard as factoring the composite moduli. (This implies that the PKC is *provably secure* under the assumption that factoring is hard.) Furthermore for IDC based on the intractability of discrete logarithm problem, we have analyzed the security against the conspiracy of some entities and in addition have shown several practical improvements to establish a *secure* and *efficient* IDC. We finally discussed the applications of the algorithms developed in *CHAPTER III* , *IV* , *V* and *VI* to the PKC's and

have pointed out the *significance* of finite field arithmetics and the *remarks* for the development of such algorithms.

Here we will describe the perspective for the studies on the *cryptographies* and the *related technical areas*.

In *CHAPTER III*, we have studied the configuration of multipliers in $GF(2^m)$, however, we could not yet have provided an universal method to design multipliers for $GF(2^m)$ with arbitrary m. Hence for every m, the constructive method must be developed to design multipliers for $GF(2^m)$, which have the regularity and the modularity in the structure and can be implemented with small circuit size $O(m^2)$. Furthermore as a theoretical interest, we must figure out which cases satisfy the *minimum term condition* for MOM by *S.A.Vanstone* [Va87]. (See Lemma 3.10.)

In *CHAPTER IV*, we have presented the recursive division algorithm and have developed two types of the divider for recursive algorithm. The divider has *simultaneous size* $O(m^3)$ and *depth* $O((log\ m)^2)$, however, further studies will be required in the light of *Area and Time Optimality*.

We finally refer to the perspective for the studies on *cryptographies*. The PKC itself is a *fairly* attractive and interesting research field, while the concept of the PKC provides several applications, e.g., identification schemes, signature schemes, cryptographically strong pseudo-random bit generator (denoted *CSPRBG*) [BM84,BBS86], interactive proof system [Ba85, GMR85], etc. The most significant point is to establish *provably secure* systems.

In 1985, *S.Goldwasser, et.al.* defined an interactive proof system, in which the prover proves the correctness of his statement to the verifier with no additional information except for the correctness of the statement. They named the above system *Zero-Knowledge Proof System* (denoted *ZKIP*). In their definition, zero-knowledgeness is formulated by *polynomial time indistinguishability*, which is also employed in CSPRBG [BM84,BBS86]. The ZKIP is a considerably useful tool to establish provably secure systems such as identification schemes [FS86,FFS87] and signature schemes. In 1986, *A.Fiat* and *A.Shamir* presented a zero-knowledge identification scheme [FS86] and in 1987 *U.Feige, et.al.* formulated strict definition for zero-knowledge and established a zero-knowledge identification scheme [FFS87]. (Note that [FS86] and [FFS87] are needless to say *provably secure!*) Further theoretical results for ZKIP are as follows:

R1. [GMW86] *If there exists a secure probabilistic encryption, then all NP-statements have the computational ZKIP.* □

R2. [For87] *Unless the polynomial time hierarchy [St77] collapses to the second level, then any NP-complete language does not have the perfect and the statistical ZKIP.* □

R3. [SMP87] *Non-interactive zero-knowledge proof system exists under the assumption that quadratic residuosity is hard.* □

R4. [BFM88] *Non-interactive zero-knowledge proof system exists under the assumption that distinguishing*

products of two primes from those product of three primes is hard. □

As describes above, the analysis for the security of cryptographical schemes requires to prove the security, i.e., to show that cryptographical schemes are secure against any attacks. In order to establish provably secure systems, *theoretical foundations* and *strict formulations* must be studied.

ACKNOWLEDGMENTS:

I would like to thank my advisor, Professor Shigeo Tsujii of Tokyo Institute of Technology, for the guidance that made this dissertation possible. I also would like to thank Professor Kouichi Sakaniwa and Professor Kaoru Kurosawa of Tokyo Institute of Technology for their helpful discussions and suggestions during the preparation of this dissertation. I thank Prof. H.Imai and Prof. T.Matsumoto of Yokohama National University, Dr. K.Koyama of NTT Basic Research Laboratories and all other members of the private research group on cryptographies for their useful comments. I thank Prof. Scott A.Vanstone of the University of Waterloo for his suggestive comments on the configuration of multipliers, and also thank Dr. Don Coppersmith of IBM Thomas J Watson Research Center, Prof. H.Tanaka of Kobe University, Dr. K.Nakamura, Dr.E.Okamoto, Miss K.Tanaka and Mr. S.Miura of NEC Information Basic Research Laboratories for their discussions and cooperations on the analysis of identity-based cryptosystem. Special thanks go to my parents, who have encouraged me, and my dear friends, who have been supportive and cooperative. I finally thank my colleagues, Mr. M.Takeuchi of Takushoku University, Mr. T.Okamoto and Mr. K.Ohta of NTT Communication and Information Processing Laboratories, Dr. H.Shizuya of Tohoku University, Mr. M.Morii of Osaka University, Mr. T.Uematsu of Tokyo Institute of Technology and all other members in my laboratory for their fruitful discussions and kind cooperations and contributions.

PUBLICATIONS

Long Papers:

Toshiya Itoh, Kaoru Kurosawa and Shigeo Tsujii;
"On the Reliability of Public-Key Cryptosystem Using Knapsack Problem," IECE Trans. of A, vol.J67-A, No.12, December 1984, pp.1176-1180

Shigeo Tsujii, Kaoru Kurosawa, Toshiya Itoh,
Atsushi Fujioka and Tsutomu Matsumoto;
"A Public-Key Based on the Difficulty of Solving a System of Non-linear Equations," IECE Trans. of D, vol.J69-D, No.12, December 1986, pp.1963-1970

Kaoru Kurosawa, Toshiya Itoh, Hiroo Shigeta and Shigeo Tsujii;
"An Attacking Method for Multiplicative Knapsack Type Public-Key Cryptosystem Based on Finite Field," IECE Trans. of IECE, vol.E.70, NO.1, January 1987, pp.37-41

Toshiya Itoh and Shigeo Tsujii;
"A Fast Algorithm for Computing Multiplicative Inverses in Finite Fields Using Normal Bases," IEICE Trans. of A, vol.J70-A, No.11, November 1987, pp.1637-1645

Kaoru Kurosawa, Toshiya Itoh and Masashi Takeuchi;
"A Public-Key Cryptosystem Using a Reciprocal with the Same Intractability as Factoring a Large Number," IEICE Trans. of A, vol.J70-A, No.11, November 1987, pp.1632-1636

Toshiya Itoh;
"An Efficient Probabilistic Algorithm for Solving Quadratic Equations over Finite Fields," IEICE Trans. of A, vol.J71-A, No.1, January 1987, pp.95-101

Kaoru Kurosawa, Toshiya Itoh and Shigeo Tsujii;
"Public-Key Cryptosystem Using a Reciprocal Number with the Same Intractability as Factoring a Large Number," to appear in CRYPTOLOGIA, 1988

Toshiya Itoh and Shigeo Tsujii;
"A Fast Algorithm for Computing Multiplicative Inverses in $GF(2^m)$ Using Normal Bases," to appear in INFORMATION AND COMPUTATION, vol.78, No.2, August 1988

Toshiya Itoh and Shigeo Tsujii;

"Configuration of Parallel Type Multipliers in GF(2^m)," submitted to INFORMATION AND COMPUTATION, 1987

Shigeo Tsujii and Toshiya Itoh;

"ID-Based Cryptosystem Based on Discrete Logarithm Problem," submitted to JOURNAL OF CRYPTOLOGY, 1988

Short Notes:

Shigeo Tsujii, Kaoru Kurasawa, Toshiya Itoh, Atsushi Fujioka and Tsutomu Matsumoto;

"Public-Key Cryptosystem Based on the Difficulty of Solving a System Non-Linear Equations," ELECTRONICS LETTERS, vol.23, No.11, May 1987, pp.558-560

Kaoru Kurosawa, Toshiya Itoh and Masashi Takeuchi;

"Public-Key Cryptosystem Using a Reciprocal Number with the Same Intractability as Factoring a Large Number," ELECTRONICS LETTERS, vol.23, No.15, July 1987, pp.809-810

Toshiya Itoh;

"Efficient Probabilistic Algorithm for Solving Quadratic Equations over Finite Fields," ELECTRONICS LETTERS, vol.23, No.17, November 1987, pp.869-870

Shigeo Tsujii, Toshiya Itoh and Kaoru Kurosawa;

"ID-Based Cryptosystem Using Discrete Logarithm Problem," ELECTRONICS LETTERS, vol.23, No.24, November 1987, pp.1318-1320

Toshiya Itoh and Shigeo Tsujii;

"An Effective Recursive Algorithm for Computing Multiplicative Inverses in GF(2^m)," ELECTRONICS LETTERS, vol.24, No.6, March 1988, pp.334-335

Toshiya Itoh and Shigeo Tsujii;

"An Efficient Algorithm for Deciding Quadratic Residuosity in Finite Fields GF(p^m)," submitted to INFORMATION PROCESSING LETTERS, 1988

Conference:

Shigeo Tsujii, Kaoru Kurosawa, Toshiya Itoh,
Atsushi Fujioka and Tsutomu Matsumoto;
"A Public-Key Cryptosystem Based on the Difficulty of Solving a System of Non-Linear Equations," 1986 IEEE International Symposium on Information Theory, 1986

Toshiya Itoh and Shigeo Tsujii;
"An Effective Recursive Algorithm for computing Multiplicative Inverses in GF(2^m)," 1988 IEEE International Symposium on Information Theory, 1988

Kaoru Kurasawa, Toshiya Itoh and Masashi Takeuchi;
"Public-Key Cryptosystem Using a Reciprocal Number with the Same Intractability as Factoring a Large Number," 1988 IEEE Symposium on Information Theory, 1988

REFERENCES

[Ad79] Adleman, L., "A Subexponential Algorithm for Discrete Logarithm Problem with Applications to Cryptography," *Proc. of IEEE 20th Annual Symposium on Foundations of Computer Science*, 1979, pp.55-60

[Ad83] Adleman, L., "On Breaking the Iterated Merkle-Hellman Knapsack Public-Key Cryptosystem," Advances in Cryptology-Crypto'83, *Plenum* 1984, pp.303-308

[Ba85] Babai, L., "Trading Group Theory for Randomness," *17th ACM Symposium on Theory of Computing*, 1985, pp.421-429

[BBS86] Blum, L., Blum, M. and Shub, M., "A Simple Unpredictable Pseudo-random Number Generator," *SIAM* J.Comput., vol.15, No.2, May 1986, pp.364-383

[BCG86] Beth, T., Cook, B.M. and Gollmann, D., "Architectures for Exponentiation in $GF(2^m)$," Lecture Notes in Computer Science 263, Advances in Cryptology-Crypto'86, *Springer-Verlag*, Berlin 1986, pp.302-310

[BCS83] Ben-Or, M., Chor, B. and Shamir, A., "On the Cryptographic Security of Single RSA Bits," *15th ACM Symposium on Theory of Computing*, April 1983, pp.421-430

[BDS82] Brickell, E.F., Davis, J.A. and Simmons, G.J., "A Preliminary Report on the Cryptanalysis of Merkle-Hellman Knapsack Cryptosystems," Advances in Cryptology, Proc. of Crypto'82, *Plenum* 1983, pp.289-301

[Be85] Beth, T., "On the Arithmetics of Galios Fields and the Like," Lecture Notes in Computer Science 229, AAECC-3, *Springer-Verlag*, Berlin 1985, pp.2-16

[Ber68] Berlekamp, E.R., "Algebraic Coding Theory," *McGraw-Hill*, 1968

[Ber70] Berlekamp, E.R., "Factoring Polynomials over Large Finite Fields, *Math.Comp.*, vol.24, No.111, July 1970, pp.713-735

[Ber82] Berlekamp, E.R., "Bit-Serial Reed-Solomon Encoders," *IEEE* Trans. on Inform. Theory, vol.IT-28, No.6, November 1982, pp.869-874

[BFM88] Blum, M., Feldman, P. and Micali, S., "Non-Interactive Zero-Knowledge and Its Applications," *20th ACM Symposium on Theory of Computing*, Chicago, May 1988, pp.103-112

[BFMV84] Blake, I.F., Fuji-hara, R., Mullin, R.C. and Vanstone, S.A., "Computing Logarithms in Finite Fields of Characteristic Two," *SIAM* J.Alg.Disc.Math., vol.5, No.2, June 1984, pp.276-285

[BKS79] Berkovits, S., Kowalchuk, J. and Schanning, B., "Implementing Public-Key Scheme," *IEEE Communication Magazine*, vol.17, May 1979, pp.2-3

[Bl82] Blom, R., "Non-Public Key Cryptosystem," Advances in Cryptology, Proc. of Crypto'82, *Plenum*, New York 1983, pp.231-236

[BLO83] Brickell, E.F., Lagarias, J.C. and Odlyzko, A.M., "Evaluation of the Adleman Attack on Multiply Iterated Knapsack Cryptosystems," Advances in Cryptology, Proc. of Crypto'83, *Plenum* 1984, pp.39-42

[BM84] Blum, M. and Micali, S., "How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits," *SIAM* J.Comput., vol.13, No.4, November 1984, pp.850-864

[CG84] Chor, B. and Goldreich, O., "RSA/Rabin least significant bits are 1/2+1/*poly*(*log* N) secure," Lecture Notes in Computer Science 196, Advances in Cryptology-Crypto'84, *Springer-Verlag*, Berlin 1985, pp.303-313

[Co87a] Coppersmith, D., *private communication*, August 1987

[Co87b] Coppersmith, D., *private communication*, November 1987

[CR84] Chor, B. and Rivest, R., "A Public-Key Cryptosystem Based on Arithmetic in Finite Fields," Lecture Notes in Computer Science 196, Advances in Cryptology-Crypto'84, *Springer-Verlag*, Berlin 1985, pp.54-65

[CW82] Coppersmith, D. and Winograd, S., "On the Asymptotic Complexity of Matrix Multiplication," *SIAM* J.Comput., vol.11, No.3, August 1982, pp.472-492

[Den82] Denning, D.E.R., "Cryptography and Data Security," *Addison-Wesley*, 1982

[DES77] *Data Encryption Standard*, Federal Information Processing Standard (FIPS) Publication 46, National Bureau of Standards, U.S. Department of Commerce, January 1977

[DH76] Diffie, W and Hellman, M.E., "New Directions in Cryptography," *IEEE* Trans. on Inform. Theory, vol.IT-22, No.6, November 1976, pp.644-654

[DQ86] Desmedt, Y. and Quisquater, J.J., "Public-Key System Based on the Difficulty of Tampering (Is there a difference between DES and RSA?)," Lecture Notes in Computer Science 263, Advances in Cryptology-Crypto'86, *Springer-Verlag*, Berlin 1987, pp.111-117

[El85a] ElGamal, T., "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithm," *IEEE* Trans. on Inform. Theory, vol.IT-31, No.4, July 1985, pp.469-472

[El85b] ElGamal, T., "A Subexponential-Time Algorithm for Computing Discrete Logarithms over $GF(p^2)$," *IEEE* Trans. Inform. Theory, vol.IT-31, No.4, July 1985, pp.473-481

[FS86] Fiat, A. and Shamir, A., "How to Prove Yourself: Practical Solution to Identification and Signature Problems," Lecture Notes in Computer Science 263, Advances in Cryptology-Crypto'86, *Springer-Verlag*, Berlin 1987, pp.186-194

[FFS87] Feige, U., Fiat, A. and Shamir, A., "Zero Knowledge Proofs of Identity," *19th ACM Symposium on Theory of Computing*, May 1987, pp.210-217

[For87] Fortnow, L., "The Complexity of Perfect Zero-Knowledge," *19th ACM Symposium on Theory of Computing*, New York City, May 1987, pp.204-209

[GHY85a] Galil, Z., Haber, S. and Yung, M., "Symmetric Public-Key Encryption," Lecture Notes in Computer Science 218, Advances in Cryptology-Crypto'85, *Springer-Verlag*, Berlin 1986, pp.128-137

[GHY85b] Galil, Z., Haber, S. and Yung, M., "A Private Interactive Test of a Boolean Predicate and Minimum-Knowledge Public-Key Cryptosystems," *26th IEEE Symposium on Foundation of Computer Science*, 1985, pp.360-371

[Gil77] Gill Ⅲ, J.T., "Computational Complexity of Probabilistic Turing Machines," *SIAM* J.Comput., vol.6, No.4, December 1977, pp.675-695

[GJ79] Garey, M.R. and Johnson, D.S., "Computers and Intractability : A Guide to the Theory of NP-Completeness," *Freeman*, San Francisco 1979

[GM84] Goldwasser, S. and Micali, S., "Probabilistic Encryption," *Journal of Computer and System Sciences*, vol.28, 1984, pp.270-299

[GMR85] Goldwasser, S., Micali,S. and Rackoff,C., "The Knowledge Complexity of Interactive Proof-System," *17th ACM Symposium on Theory of Computing*, 1985, pp.291-304

[GMT82] Goldwasser, S., Micali, S. and Tong, P., "Why and How to Establish a Private Code on a Public Network," *23th IEEE Symposium on Foundation of Computer Science*, November 1982, pp.134-144

[GMW86] Goldreich, O., Micali, S. and Wigderson, A., "Proofs that Yields Nothing But their Validity and a Methodology of Cryptographic Protocol Design," *27th IEEE Symposium of Foundations of Computers Science*, 1986, pp.174-187

[Gol84] Goldreich, O., "On the Number of Close and Equal Pairs of Bits in a String (with Implication on the Security of RSA's L.S.B)," Lecture Notes in Computer Science 209, Advances in Cryptology-Eurocrypto'84, *Springer-Verlag*, Berlin 1985, pp.127-141

[HK87] Hasegawa, S. and Kaneko, T., "An Attacking Method for a Public-Key Cryptosystem Based on the Difficulty of Solving a System of Non-linear Equations," *10th Symposium on Information Theory and Its Applications*, Enoshima Island, Japan, November 1987, pp.113-118

[HR82] Hellman, M.E. and Reyneri, J.M., "Fast Computation of Discrete Logarithms in *GF*(p)," Advances in Cryptology, Proc. of Crypto'82, *Plenum*, New York 1983, pp.3-13

[IKT84] Itoh, T., Kurosawa, K. and Tsujii, S., "On the Reliability of Public-Key Cryptosystems Using Knapsack Problem," *IECE* Trans. of A, vol.J67-A, No.12, December 1984, pp.1176-1180

[IM85] Imai, H. and Matsumoto, T., "Algebraic Methods for Constructing Asymmetric Cryptosystems," *Proc. of AAECC-3*, Grenoble France, July 1985

[IMT87] Itoh, T., Miyata, T. and Tsujii, S., "Irreducible Polynomials over *GF*(2) and Their Application to Multiplier," *Proc. of 10th Symposium on Information Theory and Its Applications*, Enoshima Island, Japan, November 1987, pp.125-130

[IT87a] Itoh, T. and Tsujii, S., "A Fast Algorithm for Computing Multiplicative Inverses in Finite Fields Using Normal Bases," *IEICE* Trans. of A, vol.J70-A, No.11, November 1987, pp.1637-1645

[IT87b] Itoh, T. and Tsujii, S., "Configuration of Parallel Type Multipliers in $GF(2^m)$," submitted to *Information and Computation*, 1987

[IT87c] Itoh, T. and Tsujii, S., "A Decision Algorithm for Quadratic Residuosity in Finite Fields," *Technical Group of IEICE*, vol.IT87-3, May 1987, pp.13-20

[IT88a] Itoh, T. and Tsujii, S., "A Fast Algorithm for Computing Multiplicative Inverses in $GF(2^m)$ Using Normal Bases," to appear in *Information and Computation*, vol.78, No.2, August 1988

[IT88b] Itoh, T. and Tsujii, S., "An Effective Recursive Algorithm for Computing Multiplicative Inverses in $GF(2^m)$," *Electronics Letters*, vol.24, No.6, March 1988, pp.334-335

[IT88c] Itoh, T. and Tsujii, S., "An Efficient Algorithm for Deciding Quadratic Residuosity in Finite Fields $GF(p^m)$," submitted to *Information Processing Letters*, 1988

[IT88d] Itoh, T. and Tsujii, S., "An Effective Recursive Algorithm for Computing Multiplicative Inverses in $GF(2^m)$," to appear in *1988 IEEE International Symposium on Information Theory*, Kobe, Japan, 1988

[Ito87] Itoh, T., "Efficient Probabilistic Algorithm for Solving Quadratic Equations over Finite Fields," *Electronics Letters*, vol.23, No.17, August 1987, pp.869-870

[Ito88a] Itoh, T., "An Efficient Probabilistic Algorithm for Solving Quadratic Equations over Finite Fields," *IEICE* Trans. of A, vol.J71-A, No.1, January 1988, pp.95-101

[Ito88b] Itoh, T., "On Generation of Normal Bases in $GF(2^m)$," *Manuscript*, April 1988

[KIST87] Kurosawa, K., Itoh, T., Shigeta, H. and Tsujii, S., "An Attacking Method for Multiplicative Knapsack Type Public-Key Cryptosystem Based on Finite Field," *IECE* Trans. of IECE, vol.E.70, No.1, January 1987, pp.37-41

[KIT87] Kurosawa, K., Itoh, T. and Take-uchi, M., "A Public-Key Cryptosystem Using a Reciprocal with the Same Intractability as Factoring a Large Number," *IEICE* Trans. of A, vol.J70-A, No.11, November 1987, pp.1632-1636

[KIT88a] Kurosawa, K., Itoh, T. and Take-uchi, M., "Public-Key Cryptosystem Using a Reciprocal Number with the Same Intractability as Factoring a Large Number," to appear in *Cryptologia*, 1988

[KIT88b] Kurosawa, K., Itoh, T. and Take-uchi, M., "Public-Key Cryptosystem Using a Reciprocal Number with the Same Intractability as Factoring a Large Number," to appear in *1988 IEEE Symposium on Information Theory*, Kobe, Japan, 1988

[Knu81] Knuth, D., "The Art of Computer Programming," *Addison-Wesley*, 2nd. Edition, vol.2, 1981

[Ko78] Kohnfelder, L.M., "A Method for Certification," MIT Lab. for Computer Science, Cambridge, Mass., May 1978

[KO87] Koyama, K. and Ohta, K., "Identity-Based Conference Key Distribution System," Lecture Notes in Computer Science 293, Advances in Cryptology-Crypto'87, *Springer-Verlag*, Berlin 1988, pp.175-184

[Kra86] Kranakis, E., "Primality and Cryptography," Wiley-Teubner Series in Computer Science, *John Wiley & Sons*, 1986

[Ku87] Kurosawa, K., "A Cryptographically Secure Pseudo Random Sequence Generator Based on Reciprocal Number Cryptosystem," *Proc. of 1987 Symposium on Cryptography and Information Security*, February 1987

[Ku88] Kurosawa, K., "The *k-th* Least Significant of Reciprocal Number Cryptosystem," *IEICE* Trans. of A, vol.J71-A, No.1, January 1988, pp.102-105

[La83] Lautemann, C., "BPP and the Polynomial Hierarchy," *Information Processing Letters*, vol.17, No.8, November 1983, pp.215-217

[Lag83] Lagarias, J.C., "Knapsack Public Key Cryptosystems and Diophantine Approximation," Advances in Cryptology, Proc. of Crypto'83, *Plenum* 1984, pp.3-23

[LN83] Lidl, R. and Niederreiter, H., "Finite Fields," ENCYCLOPEDIA OF MATHEMATICS and Its Applications, vol.20, *Addison-Wesley*, London 1983

[LO83] Lagarias, J.C. and Odlyzko, A.M., "Solving Low-Density Subset Sum Problems," *24th IEEE Symposium on Foundation of Computer Science*, 1983, pp.1-10

[McE78] McEliece, R.J., "A Public-Key Cryptosystem Based on Algebraic Coding Theory," *DSN Progress Report* 42-44, January and February 1978, pp.114-116

[McE87] McEliece, R.J., "Finite Fields for Computer Scientists and Engineers,." *Kluwer Academic Publishers*, 1987

[MH78] Merkle, R.C. and Hellman, M.E., "Hiding Information and Signatures in Trapdoor Knapsacks," *IEEE* Trans. on Inform. Theory, vol.IT-24, No.5, September 1978, pp.525-530

[Mi83] Miyaguchi, S., "A Fast Computing Scheme for RSA Public-Key Cryptosystem and Its VLSI Organization," *IPSJ* Trans. of IPSJ, vol.24, No.6, November 1983, pp.764-771

[MI87] Matsumoto, T. and Imai, H., "On the Key Predistribution System: A Practical Solution to the Key Distribution Problem," Lecture Notes in Computer Science 293, Advances in Cryptology-Crypto'87, *Springer-Verlag*, Berlin 1988, pp.185-193

[Mil76] Miller, G., "Riemann's Hypothesis and Tests for Primality," *J.Comput.Sci.*, vol.13, 1976, pp.300-317

[MK88] Morii, M. and Kasahara, M., "Notes on Multiplication and Division over Finite Fields," *IEICE* Trans. of A, vol.J71-A, No.3, March 1988, pp.845-853

[MKN86] Morii, M., Kasahara, M. and Namekawa, T., "A Knapsack Type Public-Key Cryptographic System Based on Chinese Remainder Theorem," *Technical group of IECE*, vol.IN86-5, April 1986, pp.25-30

[MO81] Massey, J.L. and Omura, J.K., patent Application of "Computational Method and Apparatus for Finite Field Arithmetic," submitted in 1981

[Moe77] Moenck, R.T., "On the Efficiency of Algorithm for Polynomial Factoring," *Math.Comp.*, vol.31, No.137, January 1977, pp.235-250

[MS77] MacWilliams, F.J. and Sloane, N.J.A., "The Theory of Error-Correcting Codes," *North-Holland*, New York, 1977

[Nie86] Niederreiter, H., "Knapsack-Type Cryptosystems and Algebraic Coding Theory," *Problems of Control and Information Theory*, vol.15, No.2, 1986, pp.159-166

[NOTM87] Nakamura, K., Okamoto, E., Tanaka, K. and Miura, S., *private communication*, August 1987

[Oka84] Okamoto, T., "Secure User Authentication without Password Files," *Technical Group of IECE*, vol.IN83-92, January 1984, pp.43-48

[Oka86] Okamoto, T., "A Single Public-Key Authentication Scheme for Multiple Users," *IECE* Trans. of D, vol.J69-D, No.10, October 1986, pp.1481-1489

[Oka87] Okamoto, E., "Key Distribution System Based on Identification Information," Lecture Notes in Computer Science 293, Advances in Cryptology-Crypto'87, *Springer-Verlag*, Berlin 1988, pp.194-202

[ON86] Okamoto, E. and Nakamura, K., "Cryptanalytic Attacks on the Recently Proposed Public-Key Cryptosystems," *Proc. of 1986 Symposium on Cryptography and Information Security*, January 1986

[Pan78] Pan, V.Y., "Strassen's Elimination is not Optimal," Proc. 19th *Annual IEEE symposium on the Foundation of Computer Science*, 1978, pp.166-176

[Pan81] Pan, V.Y., "New Combinations of Methods for Acceleration of Matrix Multiplication," *Comput.Math. with Appl.*, vol.7, 1981, pp.73-125

[PH78] Pohlig, S.C. and Hellman, M.E., "An Improved Algorithm for Computing Logarithms over *GF*(p)," *IEEE* Trans. on Inform. Theory, vol.IT-24, No.1, January 1978, pp.106-110

[Ra76] Rabin, M.O., "Probabilistic Algorithms," in Algorithms and Complexity : New Directions and Recent Results, J.F. Traub *ed.*, *Academic Press*, New York 1976, pp.21-39

[Ra79] Rabin, M.O., "Digitalized Signatures and Public-Key Functions as Intractable as Factorization," *Tech. Rep.*, MIT/LCS/TR-212, MIT Lab. Comput. Sci., 1979

[Ra80a] Rabin, M.O., "Probabilistic Algorithm in Finite Fields," *SIAM* J. Comput., vol.9, No.2, May 1980, pp.273-280

[Ra80b] Rabin, M.O., "Probabilistic Algorithm for Primality Testing," *Journal of Number Theory*, vol.12, 1980, pp.128-138

[Riv80] Rivest, R.L., "A Description of a Single-chip Implementation of the RSA Public-Key Cryptosystem," *NTC Conference Record*, vol.3, 1980

[RSA78] Rivest, R.L., Shamir, A. and Adleman, L., " A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol.21, No.2, February 1978, pp.120-126

[RT75] Reed, I.S. and Truong, T.K., "The Use of Finite Fields to Compute Convolutions," *IEEE* Trans. on Inform. Theory, vol.IT-21, No.2, March 1975, pp.208-213

[SA84] Schnorr, C.P. and Alexi, W., "RSA bits are .5+ε Secure," Lecture Notes in Computer Science 209, Advances in Cryptology-Eurocrypto'84, *Springer-Verlag*, Berlin 1985, pp.113-126

[Sch77] Schonhage, A., "Schnelle Multiplikation von Polynomen uber Korpern der Charakteristic 2," *Acta Informatica*, vol.7, 1977, pp.395-398

[Sch86] Schroeder, M.R., "Number Theory in Science and Communication," *Springer-Verlag*, Berlin 1986

[Sha82] Shamir, A., "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem," *23th IEEE Symposium on Foundation of Computer Science*, 1982, pp.145-152

[Sha84] Shamir, A., "Identity-Based Cryptosystem and Signature Scheme," Lecture Notes in Computer Science 196, Advances in Cryptology-Crypto'84, *Springer-Verlag*, Berlin 1985, pp.47-53

[SM87] Shimizu, A. and Miyaguchi, S., "Fast Data Encipherment Algorithm FEAL," *Proc. of Eurocrypto'87*, April 1987

[SMP87] Santis, A., Micali, S. and Persiano,G., "Non-Interactive Zero-Knowledge Proof System," Lecture Notes in Computer Science 293, Advances in Cryptology-Crypto'87, *Springer-Verlag*, Berlin 1988, pp.52-72

[SS78] Solovay, R. and Strassen, V., "A Fast Monte Carlo Test for Primality," *SIAM* J.Comput., vol.6, No.1, March 1978, pp.84-85

[St69] Strassen, V., "Gaussian Elimination is not Optimal," *Numer.Math.*, vol.13, 1969, pp.354-356

[St77] Stockmeyer, L.J., "The Polynomial-Time Hierarchy," *Theoretical Computer Science*, vol.3, 1977, pp.1-22

[STP86] Scott, P.A., Tavares, S.E. and Peppard, L.E., "A Fast VLSI Multiplier for $GF(2^m)$," *IEEE* Journal of Selected Area of Communications, vol.SAC-4, No.1, January 1986, pp.62-66

[Ta86] Tanaka, H., "A Realization Scheme for the Identity-Based Cryptosystem," *Technical group of IECE*, vol.IT86-38, July 1986, pp.29-32

[Ta87] Tanaka, H., "A Realization Scheme for the Identity-Based Cryptosystem," Lecture Notes in Computer Science 293, Advances in Cryptology-Crypto'87, *Springer-Verlag*, Berlin 1988, pp.340-349

[Ta88] Tanaka, H., "A Personal View on ID-Based Cryptosystem," *Proc. of 1988 Symposium on Cryptography and Information Security*, February 1988

[TAI87] Tsujii, S., Akiyama, R. and Itoh, T., "Implementation of Moon Letter Public-Key Cryptosystem," *Proc. of the 1987 Workshop on Cryptography and Information Security*, Noda, Japan, July 1987, pp.143-152

[TFH88a] Tsujii, S., Fuji-oka, A. and Hirayama, Y., "A Public-Key Cryptosystem Based on the Difficulty of Solving a System of Non-linear Equations (Second Version)," *Proc. of 1988 Symposium on Cryptography and Information Security*, February 1988

[TFH88b] Tsujii, S., Fuji-oka, A. and Hirayama, Y., "Generalization of the Public-Key Cryptosystem Based on the Difficulty of Solving a System of Non-linear Equations," submitted to *IEICE*, May 1988

[TI88] Tsujii, S. and Itoh, T., "ID-Based Cryptosystem Based on Discrete Logarithm Problem,." submitted to *Journal of Cryptology*, 1988

[TIK87] Tsujii, S., Itoh, T. and Kurosawa, K., "ID-Based Cryptosystem Using Discrete Logarithm Problem," *Electronics Letters*, vol.23, No.24, November 1987, pp.1318-1320

[TKIFM86] Tsujii, S., Kurosawa, K., Itoh, T., Fuji-oka, A. and Matsumoto, T., "A Public-Key Cryptosystem Based on the Difficulty of Solving a System of Non-linear Equations," *IECE* Trans. of D, vol.69-D, No.12, December 1986, pp.1963-1970

[TKIFM87] Tsujii, S., Kurosawa, K., Itoh, T., Fuji-oka, A. and Matsumoto, T., "Public-key cryptosystem based the difficulty of solving a system of non-linear equations, *Electronics Letters*, vol.23, No.11, 1987, pp.558-560

[TO88] Tanaka, K. and Okamoto, E., "Realization of RSA Cryptosystem Using Digital Signal Processor," to appear in *1988 IEEE International Symposium on Information Theory*, Kobe, Japan, 1988

[Va87] Vanstone, S.A., *Special Lecture* at NTT Basic Research Lab., May 1987

[VV83] Vazirani, U.V. and Vazirani, V.V., "RSA bits are $0.732+\varepsilon$ secure," Advances in Cryptology, Proc. of Crypto'83, *Plenum* 1984, pp.369-375

[Wil80] Williams, H.C., "A Modification of the RSA Public-Key Encryption Procedure," *IEEE* Trans. on Inform. Theory, vol.IT-26, No.6, November 1980, pp.726-729

[Wil85] Williams, H.C., "Some Public-Key Crypto-Functions as Intractable as Factorization," *Cryptologia*, vol.9, No.3, July 1985, pp.223-237

[WTSDOR85] Wang, C.C., Truong, T.K., Shao, H.M., Deutsch, L.J., Omura, J.K. and Reed, I.S., "VLSI Architectures for Computing Multiplications and Inverses in $GF(2^m)$," *IEEE* Trans. on Comput., vol.C-34, No.8, August 1985, pp.709-716

[WW84] Wah, P.K.S. and Wang, M.Z., "Realization and Application of Massey-Omura Lock," *Proc. of International Zurich Seminar*, March 1984, pp.175-182

[Zie68] Zierler, N., "On Primitive Trinomials (mod 2)," *Information and Computation*, vol.13, 1968, pp.541-554