01 May 1993

The Interpolating Random Spline Cryptosystem and the Chaotic-Map Public-Key Cryptosystem

Fengi Hwu

Chung You Ho
Missouri University of Science and Technology

THE INTERPOLATING RANDOM SPLINE CRYPTOSYSTEM AND THE CHAOTIC-MAP PUBLIC-KEY CRYPTOSYSTEM

F. Hwu* and C. Y. Ho

CSc-93-09

Department of Computer Science

University of Missouri - Rolla

Rolla, MO 65401 (314)341-4491

*This report is substantially the Ph.D. dissertation of the first author, completed May 1993.

ABSTRACT

The feasibility of implementing the interpolating cubic spline function as encryption and decryption transformations is presented. The encryption method can be viewed as computing a transposed polynomial. The main characteristic of the spline cryptosystem is that the domain and range of encryption are defined over real numbers, instead of the traditional integer numbers. Moreover, the spline cryptosystem can be implemented in terms of inexpensive multiplications and additions.

Using spline functions, a series of discontiguous spline segments can execute the modular arithmetic of the RSA system. The similarity of the RSA and spline functions within the integer domain is demonstrated. Furthermore, we observe that such a reformulation of RSA cryptosystem can be characterized as polynomials with random offsets between ciphertext values and plaintext values. This contrasts with the spline cryptosystems, so that a random spline system has been developed. The random spline cryptosystem is an advanced structure of spline cryptosystem. Its mathematical indeterminacy on computing keys with interpolants no more than 4 and numerical sensitivity to the random offset t_i increases its utility.

This article also presents a chaotic public-key cryptosystem employing a one-dimensional difference equation as well as a quadratic difference equation. This system makes use of the El Gamal's scheme to accomplish the encryption process. We note that breaking this system requires the identical work factor that is needed in solving discrete logarithm with the same size of moduli.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

I. INTRODUCTION

In order to plan for the information society in the twenty-first century, we must understand the community which we now inhabit and anticipate the future. The networked population in this society has increased enormously in the past decade. A great deal of international trade and commerce is already being mediated via computers and communication mediums. Traditionally, hand-written papers dominated most human activities. The explosive utilization of computers and networks has drastically reduced the distances among human beings and among nations. The cooperation and collaboration of various entities and units are mediated by electronically transmitted data. Business practices based on paper will continue to decrease to an absolute minimum, and more uses will be made of digital technology. Therefore, the quicker we adapt to the new technology, the greater the benefits we will obtain.

From a negative point of view, information on networks can be stolen, sabotaged, and used for extortion. In this digitalized environment, the integration of computers and telecommunication networks becomes an attractive target. There is usually no security problem involved when a single computer is used in one room by one person for one application only. However, when we start to share databases and computers, there is a chance for confidential information to be misused. Along with the fact that networks will extend access; it is clear that such networks inherently increase the risk to information security. To protect the property of individuals, organizations and governments in the forthcoming information society, cryptography provides us a promising technique. Cryptography is inextricably bound to information technology in general and to computers in particular. The advent of this technology could be used to convert the future communication traffic from public to secret formats.

Encryption is the process of scrambling data into an unfamiliar form for privacy. The process is a function of the input data and a key. There are two classes of cryptographic systems; in a secret-key cryptosystem the legitimate sender and receiver employ the same key for scrambling and unscrambling data, in a public-key cryptosystem each user has one public key for scrambling data and one private key for reproducing the original data.

Traditionally, cryptography has been used for military and diplomatic missions. Cryptography can be used for civilian users as well as military users. In the civilian sector, cryptography is used for internetwork commerce, electronic mail, international banking, and electronic money exchange. It provides both privacy and authentication. This includes the identification and authentication of digital signatures instead of hand-written signatures. Public-key cryptosystems can be explicitly used for these purposes.

Public-key cryptography has implications far beyond simple data scrambling and unscrambling. It allows people to do things securely over computer networks which are impossible in any other way. For instance, there are password protections, digital signatures, fair coin tosses, and bit commitments, etc. Regarding conventional password protection, the host computer stores the password in encrypted form, which can create serious security problems. For one, when the user types the password into the system, anyone accessing the data path can read it. He might be accessing his computer through a transmission path that passes through industrial competitors, foreign countries, and some universities, any one of which can look at his password through its machine. Two, anyone with access to the processor memory of the system can see the password before the system encrypts it, and can compare it with the system encrypted form in the password file.

Public-key cryptography solves this problem by allowing the host computer to keep a file of every user's public key, each user keeps his own secret key. This secret key is generated by the user's hardware or communication software. This requires a trusted and intelligent terminal, but neither the host nor the communication path needs to be secure. When logging in, the host sends a challenge number to the user. The user encrypts the random number with his secret key, and sends it back to the host computer. The host computer decrypts the message using the user's public key. If the decrypted message matches what the host computer sent to the user in the first place, the computer allows the user to access to the system. No one else has access to the user's secret key, so no one else can impersonate the user. More important, the user never sends his secret key over the communication channel to the host computer. No one listening to the channel can learn the secret key and impersonate the user.

Digital signature protection is another important application. Encrypt a document with your secret key, and you have a secure digital signature. Anyone with the public key can decrypt it, so anyone can read it. Only you have access to your secret key, so no one else could sign it. And finally, anyone who modifies the encrypted document will produce gibberish when decrypted, so no one can modify the signed document. With the digital signature protocol, a trusted authority should sign both communicating users' public keys to prevent the potential impersonation attack. The signed keys would include a signed certification. Now both parties would know that the public key that they received over the communication channel actually belongs to each other.

Assume that two persons who do not trust each other can flip a coin over some communication media. The fair coin toss protocol could prevent each one from cheating. Assume that Alice and Bob both generate a secret and public key pair. Alice generates

two messages, one indicating heads and the other indicating tails. Alice encrypts both messages with her public key and sends them to Bob. Bob who cannot read either message, chooses one randomly. He encrypts it with his public key and sends it back. Alice, who cannot read this message, decrypts it with her secret key and sends it back to Bob. Bob decrypts the message with his secret key to reveal the result of coin toss. He sends the decrypted message back to Alice. Alice reads the result of the coin toss and verifies that the message is correct. Both Alice and Bob reveal their public and secret keys so that both can verify that the other did not cheat. This protocol is self-enforcing. Either party can immediately detect cheating on the other party.

The bit commitment protocol is to let Alice commit to a prediction, but does not reveal that prediction to Bob until sometime later. On the other hand, Bob wants to make sure that Alice cannot change her mind after she has committed to her prediction. First, both Alice and Bob each generate some random bit strings. Bob hands Alice his string. Alice creates a message consisting of her random string, the bit she wishes to commit to, and Bob's random string. She then encrypts it with her public key and sends the result back to Bob. Bob cannot decrypt the message, so he does not know what the bit is. When it comes time for Alice to reveal her bit, she decrypts it using her secret key. Bob then ensures himself that the bit is valid by checking that his random string is correct. Moreover, two parties could be involved in long distance contract signing. No one is willing to sign first. We could establish a public-key cryptographic protocol for signing contracts by computer, so that both parties are bound by the contract at the same time. Physical proximity and written signatures are what we want to avoid.

We notice that identification of individuals is one of the critical requirements of access control, whether it is for access into buildings, authorization of credit at a point of

sale, or into computers and communication networks. In most access control systems, personal data of each user is stored in a secure computer. This personal data must be retrieved whenever someone wants to access the computer networks. To avoid the extra communication to the computer storing this database, off-line systems require the user to provide this data that card readers can verify. Magnetic stripe cards have been used to hold the data necessary for anyone to authenticate the user's identification. But they are forgeable. The smart card or chip card is designed for the replacement of the magnetic stripe card of the 90's. Smart cards are plastic cards with an embedded integrated circuit capable of performing computations.

The smart card is intended to be a multipurpose card. Its protected memory can store identification data other than payment transactions. In the off-line system, public-key systems provide a means of authenticating this crucial data by providing digital signatures that any terminal can verify. A certification center creates a digital signature for the data on each smart card, and this signature is also included in the smart card data. Any alternation of the personal identification data in the smart card will result in an incorrect certification center digital signature, which can be immediately detected. Smart cards will probably replace all magnetic stripe cards.

Cryptography is progressing rapidly, both in the development of algorithms and cryptographic protocols, and in the related discipline of designing an encryption device. Integrated circuits for encryption are now produced economically. More and more chip level encryption products will be produced. Public-key cryptography will be available in this form. One of the most difficult aspects of cryptographic applications is key management. Poorly done, it can endanger secure communication. Overdone, it can present a burdensome expense. The chosen method of key management must be consistent with

the overall security requirement and architecture. Thus, key generation, distribution, and organization will be the most critical issues of cryptographic application. Furthermore, cryptographic systems must be international in scope and address the problems related to the common encryption algorithms, secure key management and efficient distribution irrespective of national boundaries. Cryptographic devices must become one of the inseparable parts of computer design. Making use of secure cryptographic systems and efficient key management, we could design secure cryptographic protocols to allow people to do business securely and fairly over computer networks.

The future of computer technology is becoming quite clear, with artificial intelligence machines, and microchip technology being built into everything around us. Worldwide business relies on electronic communication via computers. Any decision made and action taken is a result of computerized outputs as well as reliable information inputs. Information integrity must be absolutely maintained in order to insure the quality and accuracy of the result. A communication environment designed with the aid of cryptography should provide security that will be needed. Regarding the internal security feature of computer systems, an individual or an organization could regulate their security policy. People can access encrypted files and data only when they have legal authority. Summing up, we understand that the internal security policy of computer systems should be regulated by permission-based and cryptographic techniques, depending on how sensitive the protected entities are. The external secure communication systems should be fully regulated by computationally secure cryptographic techniques, no matter what data streams are transmitted. Key features of the information society in the twenty-first century will include cryptographic communication channels and cryptographic protocols.

A. PRELIMINARY

Cryptology is a difficult and esoteric science. Especially, when one seeks particular cryptographic functions without compromising their security. The aim of this introduction is to sketch the intellectual outlines of this subject. Two newly designed cryptographic systems are presented; spline cryptosystem and chaotic-map public-key cryptosystem.

The term, *cryptology*, stems from Greek roots meaning *hidden* and *word*, it is used to describe the study of secret communications. Generally speaking, cryptology can be separated rather cleanly into two sectors: cryptography and cryptanalysis. Cryptographic research is concerned with the secrecy and/or authenticity of data communication. Cracking systems and protocols are the major research areas in cryptanalysis. It sounds hostile, but decent cryptanalysts can expose the unsuspected weakness of systems so that modifications can be made to improve security.

Cryptography involves the study of transformations E on data; we use the notation

$$E : \bar{X} \rightarrow \bar{Y} = E(\bar{X})$$

in which $\bar{X}$ is called the clear message, or simply the *plaintext*. $\bar{X}$ is encrypted into the ciphertext message, or just the *ciphertext*, or most often the *cryptogram* $\bar{Y} = E(\bar{X})$ by the cryptographic transformation E, where $\bar{X} \in \mathbf{M}$ the finite *message space* and $\bar{Y} \in \mathbf{C}$ the *ciphertext space*. E denotes the *encipherment(encryption, encoding)* of plaintext $\bar{X}$ into the ciphertext $\bar{Y}$. The only requirement on E is the obvious one; it must be possible to reverse the process of encipherment (called *decipherment*) and recover plaintext from the ciphertext. Thus the transformation E must have an inverse D, so that $ED = I$ or $DE = I$

where I is the identity transformation. We use the notation

$$D : \bar{Y} \rightarrow \bar{X} = D(\bar{Y}) = D(E(\bar{X}))$$

which is read *ciphertext* $\bar{Y}$ is decrypted to *plaintext* $\bar{X} = D(\bar{Y})$ by the inverse cryptographic transformation D.

Originally, cryptographic transformations were made by hand. Later, mechanical devices were introduced to carry out the transformation and these were succeeded by electromechanical devices. Today, encipherment is the result of the execution of a program. If encipherment is to hide information, a pair of users must tailor the transforma tion to their specific communication. A cryptographic system(*cryptosystem*) is a family **T** = $\{E_k, D_k: k \in K\}$ of cryptographic transformations. The subscript k on E_k and D_k, is called the *key*. We can assume that k is a sequence of alphanumeric ASCII characters(or a sequence of bit streams) and E_k is an algorithm(or a program) which takes input $\bar{X}$ and produces output $\bar{Y}$. The *key space* **K** is the totality of key values.

A cryptanalyst's chore is to *break* a cryptosystem; this means that the cryptanalyst will attempt to deduce the meaning of ciphertexts, or to determine a decryption algorithm that matches an encryption algorithm. The analyst can do any or all of the following three things:

a. attempt to break a single message,

b. attempt to recognize patterns in encrypted messages, in order to break subsequent ones by applying a straightforward decryption algorithm,

c. attempt to find general weaknesses in an encryption algorithm, not necessarily having any messages.

Using the *Kerckhoff's principle*, an analyst works on many things. Such as encrypted messages, known encryption algorithms, intercepted plaintext, data items known or suspected in ciphertext, mathematical or statistical tools, and techniques and properties of languages. There are three attacks with which cryptanalysts have used:

a. Ciphertext-only attack: to make use of statistical tools on intercepted ciphertext to crack the system.

b. Known-plaintext attack: to compare the known ciphertext with the corresponding plaintext so that the secret key can be derived.

c. Chosen-plaintext attack or chosen-ciphertext attack: to submit unlimited plaintext or ciphertext of his own and then inspect the difference of outputs resulted from the different inputs. So that secret keys can be derived.

Most cryptosystems in use today are intended to be secure against the chosen-plaintext attack or chosen-ciphertext attack.

Historically, there are two simple and classical groups of cipher; substitution cipher and transposition cipher. For the former, one letter is replaced with another in the same alphabet, and for the latter, the order of letters is rearranged. These classical ciphers are relatively easy to break using information about letter frequency distribution.

Because of Diffie and Hellman's speculation[DiH76], cryptographic research moved into the new era of *public-key systems*. Computationally hard problems with number theory were introduced as the building blocks of secure cryptosystems. Most of these systems can be implemented in a public-key style in which it is easy for everyone to encrypt messages, whereas only the legitimate receiver can decrypt. Such systems sometimes

include the concept of a *digital signature*, which is a marker resulting from the signer and message. So the legal receiver of the encrypted message can be assured that the message was sent by a legal sender. The security of these systems partly depends on our current inability to solve certain number-theoretical problems efficiently. The security of RSA depends on the difficulty in factoring large composite integers and solving the discrete logarithm problem defined over a finite field. To set RSA up, one must be able to recognize whether large numbers are prime or not by primality test.

More specifically, Figure 1. illustrates a complete cryptosystem architecture consisting of three parts: an enciphering process, a deciphering process, and a key generator dominated by an initial condition, where $\bar{X}$ is the plaintext and $\bar{Y}$ is the ciphertext.

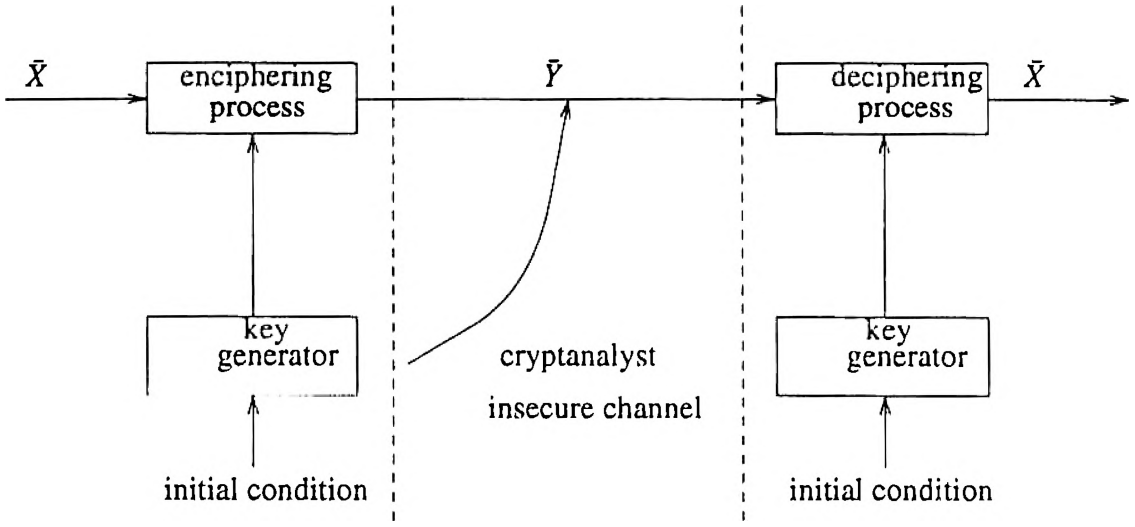

Figure 1. Complete architecture of a secret-key cryptosystem.

Actually, the previous architecture is a secret-key structure, and Figure 2., specifies a public-key architecture with the key generator controlled by the receiver, where K and k are public-key and secret-key respectively.

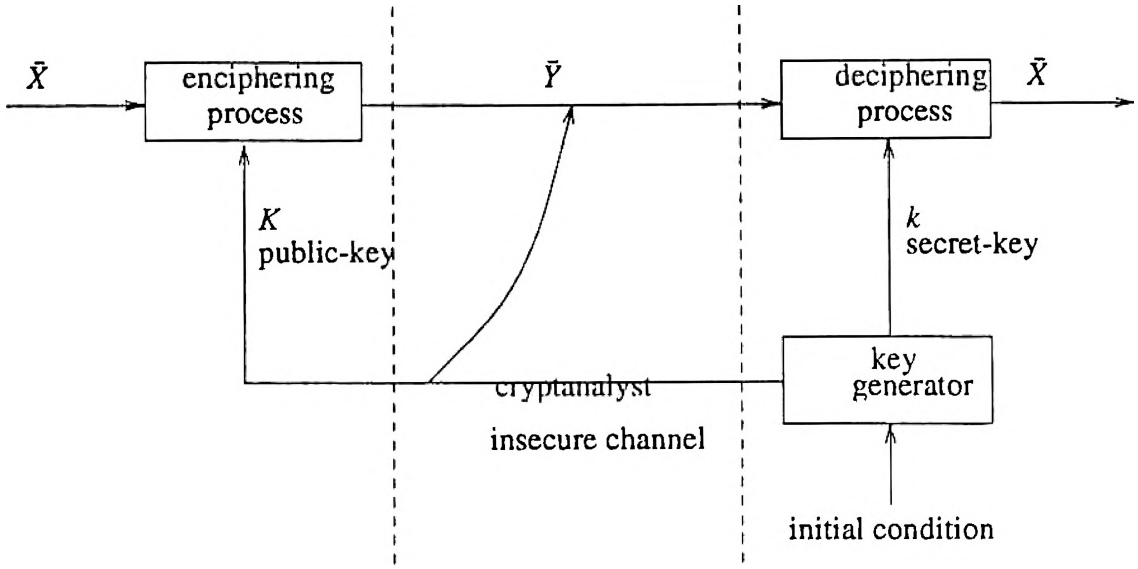

Figure 2. Complete architecture of a public-key cryptosystem

Strictly speaking, the strength of an encryption algorithm depends on the computational security, instead of theoretical security. An encryption algorithm may be *breakable*, meaning that given enough time and data, an analyst could determine the algorithm. However, practicality is also an issue. For instance, a particular cipher scheme may have an inverse decryption scheme that requires 10^{30} operations. Assuming a current-technology computer, which performs on the order of 10^{10} operations per second, this decipherment would require 10^{20} seconds, or roughly 10^{12} years. In this case, although we know that theoretically, a decryption algorithm exists, the decryption algorithm can be ignored as infeasible using current technology.

B. RESEARCH MOTIVATION

Current cryptosystems are built upon the assumption that the associated number-theoretical problems cannot be solved in a practical amount of time. However as computers become more powerful, it may become possible in the near future to break existing cryptosystems. Thus a *spline cryptosystem* which does not follow the traditional number-theoretical assumption may be a useful alternative. In 1991, we proposed a spline cryptosystem[HqH92], which can be classified as a transposed polynomial system. In that paper, the plaintext values represent values of $S(t)$ from which a multi-segment cubic spline may be defined, and corresponding values of t at x-axis are given. Since, the resulting cubic spline is continuously defined over the interval [0,1], it is possible to choose values t' in each subinterval and compute the corresponding $S(t')$ as the ciphertext.

The spline cryptosystem used a fixed offset between t and t', which is 0.5. This fixed offset contrasts with the RSA system and would probably be the target for cryptanalysts. In the case of the RSA system, randomness is the major characteristic. Under the assumption of randomness in the RSA system, mapping the plaintext value into the ciphertext value will result in the ciphertext value uniformly distributed among the range. In fact, each ciphertext value will occur only once, and no ciphertext value will occur before all values have been generated. Thus, a random approach has been applied to the spline cryptosystem, so that the offset is not fixed at 0.5. Furthermore, the system is sensitive to the variation of offsets and is mathematical indeterminacy without sufficient boundary conditions. These characteristics increase its utility.

Chaotic behavior has been studied in many fields. It is desired to inject chaos into cryptography, so that the designated system becomes dynamic and non-linear. Recently, a chaotic-map secret-key cryptosystem[HaN91] was introduced. This cryptosystem employs a one-dimensional iterated map as the encryption transformation. It encrypts 64-bit plaintexts into about 147-bit ciphertexts, using a 64-bit key α. Plaintexts, ciphertexts, and keys are all real numbers $\in$ [0,1]. Whereas, a public-key version of chaotic-map cryptosystem is proposed. The El Gamal's[EGt85] public-key encryption scheme has been added into this public-key cryptosystem, in addition to the chaotic map. The motivation of designing such a system is that the one-dimensional difference equation(iterated map) is well suited to be a one-way function.

II. REVIEW OF LITERATURE

A. NUMBER-THEORETIC NOTATIONS AND DEFINITIONS

We begin with a study of properties of multiplication and division of integers. In particular, we investigate prime numbers, the computation of inverse and factorization, since these topics have major implications in the advanced encryption transformation. We also study a restricted number system, called a *field*. The fields we consider are finite and have convenient properties that make them useful for representing cryptosystem. Especially, modular arithmetic defined over finite fields has been heavily used as a cryptosystem tool.

1. Modular Arithmetic. Recall that a modulus applied to a nonnegative integer means *remainder after division*. So that any two integers are congruent under modulus n if their results of mod n are equal. This is denoted as:

$$x \equiv_n y \text{ if and } \textit{only} \text{ if } x \equiv y \ (\textit{mod } n), \tag{2.1.1.1}$$

alternately:

$$x \equiv_n y \text{ if and } \textit{only} \text{ if } (x - y) = k \times n \text{ for } \textit{some } k. \tag{2.1.1.2}$$

Modular arithmetic on nonnegative integers form a construct called, a *commutative ring*, with the operations of addition and multiplication. All rings have the properties of associativity and distributivity. Commutative rings, as their name implies, commutativity. Furthermore, if every number other than 0 has an inverse under multiplication, the number system is called a *field*. Inverses under multiplication produce a finite field called a Galois field. The integers, mod n, consist of a Galois field(GF). Cryptography is based

on *GF(p)*, where *p* is a prime, and there is a shortcut way to compute modulo *n*. That is, we could either obtain the residue modulo *n* and then do the operation or do the operation first and then find the residue modulo *n*.

To perform a secure encryption, we need a procedure for finding the inverse of an arbitrary element under modulo *n*, even for very large values of *n*, see Figure 3. The formal description is the following:

Given: $a \in \{0,n-1\}$, $GCD(a, n)= 1$, and $n>1$

Find: $x \in \{0,n-1\}$, such that $a \times x \equiv 1 \pmod{n}$

where *GCD* stands for greatest common divisor.

This algorithm[Knd81] is a fast approach that makes use of Euclid's algorithm for finding the greatest common divisor, as well as, computing the inverse of *a* under modulo *n*.

```
procedure inverse(a,n) /* find the inverse of a with respect to n */
begin
    g[0] ← n;
    g[1] ← a;
    i ← 1;
    for(i = 1; g[i] ≠ 0; i++)
    begin
        y ← g[i-1] div g[i];
        g[i+1] ← g[i-1] - y × g[i];
        u[i+1] ← u[i-1] - y × u[i];
        v[i+1] ← v[i-1] - y × v[i];
        i++;
    end;
    if (v[i-1] ≥ 0) then inverse ← v[i-1];
                    else inverse ← v[i-1] + n;
end;
```

Figure 3. Extended Euclid algorithm

Many encryption algorithms are based on modular exponentiation, so computing the power of a number is an important operation. There is a fast algorithm, so-called repeated squaring and multiplying, shown in Figure 4. Squaring when the bits of binary representation of exponents are 0 and multiplying when they are 1.

```
procedure fastexp(a,z,n) /* calculate a ** z (mod n)  */
begin
    x ← 1;
    a ← a mod n;
    while(z ≠ 0)
    begin
        while(z mod 2 ≡ 0)
        begin
            z >>= 1;
            a ← (a × a) mod n;
        end;
        z--;
        x = (x × a) mod n;
    end;
    return(x);
end;
```

Figure 4. Repeated squaring and multiplying.

2. Notations And Definitions. Let Z_n^* be the set of elements in $Z_n = \{0, 1, \cdots, n-1\}$ that are relatively prime to n(or called *reduced set of residues*). The size of Z_n^* is defined as $\phi(n)$, known as *Euler's totient function*, satisfies the equation:

$$\phi(n) = n \times \Pi(1 - \frac{1}{p}),$$

where p runs over all primes dividing n.

Especially, if p is a prime, then $\phi(p) = p - 1$. The following three theorems are an important part of the theoretical backbone of current cryptosystems. Especially the *Chinese Remainder Theorem* is a powerful cryptographical tool.

Theorem 2.1: (Euler's Theorem)

For all integer $n > 1$, $a^{\phi(n)} \equiv 1 \ (mod\ n)$, for all a $\in Z_n^*$

Theorem 2.2: (Fermat's Theorem)

If p is a prime, $a^{p-1} \equiv 1 \ (mod\ p)$, for all $a \in Z_p^*$

Theorem 2.3: (Chinese Remainder Theorem)

Let $n = n_1 \times n_2 \times \cdots \times n_k$, where the n_i are pairwise prime. Consider the correspondence $a = (a_1, a_2, \cdots, a_k)$, where $a \in Z_n$, $a_i = a(mod\ n_i) \in Z_{n_i}$. In other words, the system of congruence, $a \equiv a_i \ (mod\ n_i)$; $i = 1, 2, \ldots, k$, has a common solution in $[0, n-1]$.

The following section introduces some difficult number theory problems that can be used to protect cryptosystems.

B. COMPUTATIONAL COMPLEXITY

A recent trend in encryption is to consider problems that are hard to solve, and for which the number of possible solutions is large. Then, even with computer support, an exhaustive brute force search is expected to be infeasible. Thus, computational complexity provides a foundation for analyzing the computational requirements of cryptanalytic technique, and for studying the inherent difficulty of solving ciphers. So that the strength of a cipher is determined by the computational complexity of the algorithm used to solve the cipher.

In 1979, Diffie and Hellman[DiH79] suggested applying computational complexity to the design of the encryption algorithms. They noted that NP-complete problems might be excellent candidates for cryptosystems, because they cannot be solved in polynomial time by any known efficient techniques. Problems that are computationally more difficult than the problems in NP are not suitable for designing encryption algorithms, since the encryption and decryption must be fast(i.e. in polynomial time). Therefore, the endeavor of breaking any polynomial-time encryption algorithm must be in NP. They speculated that information could be encrypted in such a way that breaking the system would require an extensive and difficult computation. With a decryption key, however, a short-cut solution would be possible.

Diffie and Hellman's speculation brought cryptography from traditional number games into the massive computational complexity problems. It introduced the RSA cryptosystem and the knapsack cryptosystem to the public. Those related cryptographic research areas are blossoming. Researchers who work on cryptanalysis are burning their energy to discover efficient polynomial-time algorithms, so that those successfully

developed systems are breakable.

The *Discrete Logarithm Problem* and *Factorization Problem* are two hot issues in cryptanalysis. If both could be solved efficiently, then most of the current cryptosystems and cryptographical protocols would not be secure any more. The following two instances provide a more detailed description:

Instance 1: discrete logarithm problem.

Given: q is a primitive root of a prime p, $s \in \{1,2,..,p\text{-}1\}$, and

$q^k = s \ mod \ p$.

Find: k.

Instance 2: factorization problem.

Given: N is a large composite integer(over 200 digits), which is the result of multiplication of two approximately equal primes.

Find: p and q such that $N = p \times q$.

The discrete logarithm problem has found great popularity in cryptography for message authentication, secrecy system, and protocol design. It is quite infeasible to compute the value k given p, q, and s[PoH78]. Exponential complexity is needed[Oda84] in solving the discrete logarithm over finite fields. Similar discrete logarithm problems can be found in a finite field with prime characteristic p, $GF(p^k)$, or in the group of points on the elliptic curve[Kon87]. Note that it should be a genuine one-way function, instead of keeping an intrinsic trap-door secret. This is a contrast to the RSA[RSA78] trap-door one-way function.

Regarding the factorization problem, if the composite number N is very large, then factoring N to be two equally sized prime numbers becomes computationally infeasible. So far, the best algorithms found for solving the factorization problem are of exponential order.

The knapsack problem was used as a backbone by Merkle and Hellman[MeH78] to design a public-key cryptosystem in 1978. Soon after that, it was broken. But, it built a bridge between NP-complete problems and cryptography. There is no known efficient polynomial time algorithm that is able to solve an arbitrary instance of the NP-complete problem. Breaking such a knapsack system does not mean we have found an efficient algorithm to solve all instances of such problems, because the knapsack cryptosystem was based on an easy knapsack problem for which the instance might be solved using a linear time algorithm. Despite the fact that the knapsack cryptosystem is insecure, it is still being used.

C. ADVANCED CRYPTOSYSTEM

Older encryption schemes using substitution ciphers and permutation ciphers were relatively straightforward. Cryptanalysts now have new tools for analyzing codes. Even taking character frequency counts is tedious and error-prone when done by hand, while it is fast, and reliable by computers. So, we will discuss four important encryption algorithms, which represent the state of the art of the encryption algorithm. All of these four algorithms require extensive computation. Interestingly, most of these algorithms were presented at about the same time.

1. Secret-key Cryptosystem (DES). DES is based on an early proposal, *Lucifer*. It was developed by IBM to solve the growing needs for data security. This algorithm became known as the Data Encryption Standard, although its proper name is DEA(Data Encryption Algorithm) in the U.S. and DEA1 in other countries. It was adopted by the National Bureau of Standards and has withstood all the attacks published in open literatures. DES is a one-key, secret-key, or symmetric system, in which both of the joined parties use the same key to encrypt and decrypt messages.

DES is a careful and complex combination of two fundamental encryption techniques, substitution and permutation. It splits a data block in half, scrambles each half independently, combines the key calculated via a key schedule algorithm with one half, and swaps the two halves. The process is repeated 16 times, using table lookups and simple bit operations. The substitution parts are *S boxes* the only non-linear part of DES and the security of the cryptosystem depends on their constructions. Permutation is used to rearrange the output of the *S boxes* in order to produce a random effect. This depends on having as many *S boxes* as possible. The security of this standard depends on the secrecy of key instead of the encryption algorithm. The scheme can be implemented quite efficiently. Input to DES is divided into blocks of 64 bits, which are transformed using a 56-bit key (actually, the key length is 64 bits, however, parity bits on positions 8, 16,...,64 are discarded). The scheme is described as follows:

given

$$\text{plaintext } \bar{x} = (x_0, x_1, \ldots, x_{63}),$$

$$\text{ciphertext } \bar{y} = (y_0, y_1, \ldots, y_{63}),$$

key $\bar{k} = (k_0, k_1, \ldots, k_{55})$,

define

$$DES : \bar{x} \rightarrow \bar{y} = DES(\bar{k}, \bar{x}),$$

where DES is the product of mappings

$$DES = IP^{-1} \times T_{16} \times \cdots \times \phi \times T_1 \times IP,$$

and

IP is initial permutation,

T_i is the mapping on the i-th round,

ϕ is the interchange involution,

$$\phi : (x_0, x_1, \cdots, x_{31}, x_{32}, x_{33}, \cdots, x_{63}) \rightarrow (x_{32}, x_{33}, \cdots, x_{63}, x_0, \cdots, x_{31}).$$

At the last iteration, the left and the right halves are not exchanged, instead the two halves are concatenated and input to the final permutation IP^{-1}. The algorithm can be used both to encrypt and decrypt. Figure 5 shows one specific round of the DES algorithm, where + is the exclusive-OR operation.

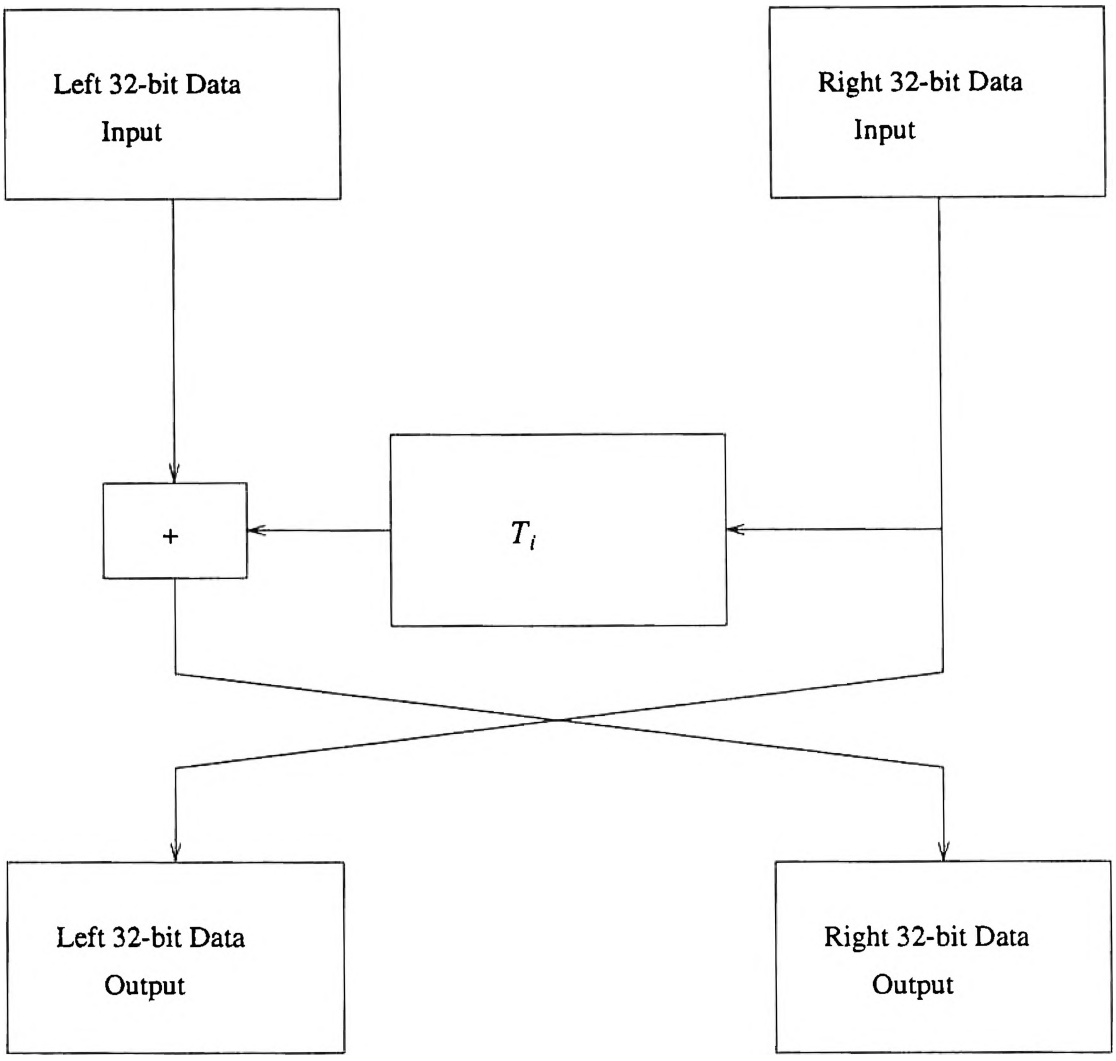

Figure 5. One round iteration of DES

A short description of the DES algorithm appeared in [DiH79] and a complete description is readily available in [DaE77]. Nowadays, the consensus of cryptanalysis is that the key size is the weakness of DES. It could be cracked by a brute-force attack.

Diffie and Hellman[DiH77] proposed a parallel architecture to exhaustively search the entire key space, which has a size of 2^{56}. It can be searched in about one day, but the cost of this architecture is over millions of dollars. Hellman[Hem80] also proposed a chosen-plaintext attack using a special purpose machine which also requires an impractical pre-processing time. Recently, Biham and Shamir[BiS90] published a chosen-plaintext attack on DES. They announced that DES with up to 15 iteration rounds could be cryptanalyzed using a different procedure than an exhaustive search. This attack finds particular differences in plaintext pairs resulting in the difference of the corresponding ciphertext pairs. Thus, if one knows the probability of the possible keys, one is able to locate the most probable key.

2. Public-key Cryptosystem. Diffie and Hellman[DiH76] proposed a fantastic encryption system in 1976. In that paper, they proposed a revolutionary public-key system and also described two subtle definitions: First, a *one-way function f* is defined such that for every x in the domain of f, $f(x)$ is easy to compute, but it is computationally infeasible to find $f^{-1}(y)$. This function has been applied to design a secure computer-login procedure. Second, a *trap-door one-way function* f_k is defined such that given the parameter k, it is easy to compute $f_k(x)$ and $f_k^{-1}(y)$, for all x and y in the domain and range. Then, they proposed a *key-exchange protocol* eliminating the key distribution problem in secret-key cryptosystem. To break this protocol seems to be as hard as solving the discrete logarithm problem. But it has not been proven that breaking the system is equivalent to computing the discrete logarithm problem. The scenario of this protocol is as follows:

a. The numbers X and P are known to each subscriber, where X is a primitive root of P.

b. Each subscriber i chooses a secret number $S_i \in \{1, 2, \ldots, P-1\}$ and publishes the number $Y_i = X^{S_i} \bmod P$ in the open domain or transmits to the intended subscriber j over public channel.

c. While the subscriber j receives Y_i, he raises the number to S_j. Thus, $Y_{ij} = X^{S_i S_j}$ is common to subscribers as well.

More concretely, the protocol employed by subscriber A and subscriber B is illustrated in Figure 6.

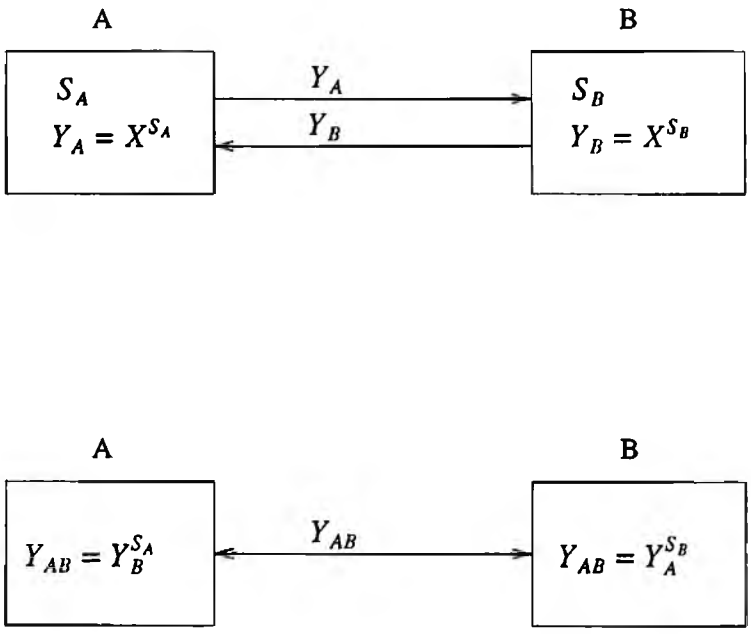

Figure 6. Key-exchange protocol

An impersonation attack [RiS84] on this scheme makes use of the fact that the identity of Y_i is not assured. Assume that the interceptor C can control the communication channel between A and B. Upon receiving Y_A from A he sends $Y_C = X^{Sc}$ to B. Identically, he sends Y_C to A instead of Y_B. Eventually, he will completely decrypt, modify, and replay messages through this channel (See Figure 7).

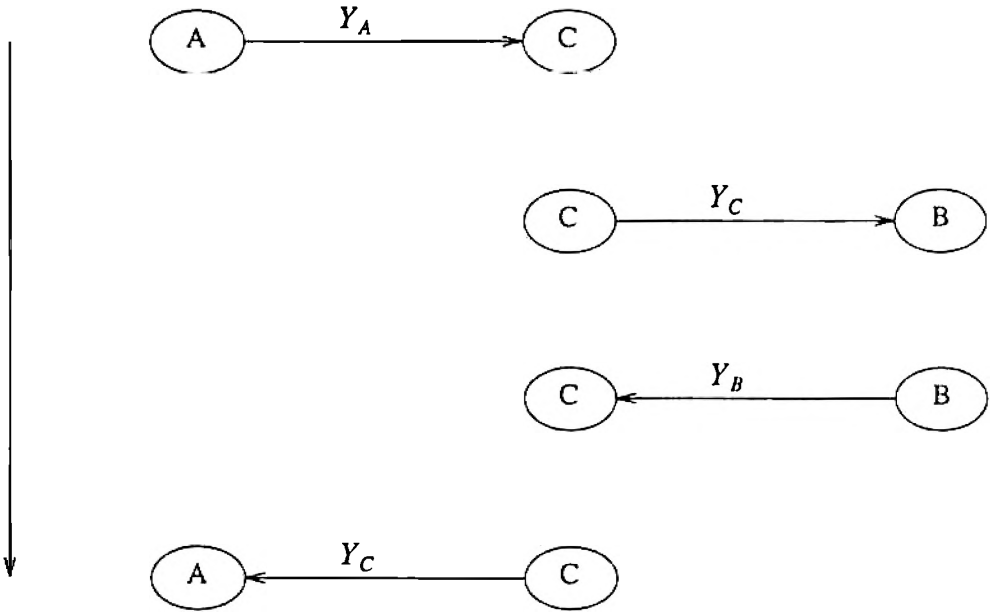

Figure 7. An impersonation attack

With a public key system, each user would have a key that does not have to be kept secret. The public nature of the key would not inhibit the secrecy of the system. The scenario of public-key system is as follows:

Each joined user must have two keys; one public key for encryption and one secret key for decryption. There must be a relation between the public key and the secret key, so that the legal designer knowing the trapdoor of the relation could derive one key easily given the other key. Indeed, the user might publish the public key freely.

There are several conditions which have to be fulfilled so that a public key cryptosystem can work properly:

a. Calculation of the public key and secret key of each subscriber should be finished in *polynomial time.*

b. Sender A, knowing the public key of receiver B, can encrypt a message sent to B in *polynomial time.*

c. Receiver B, using his/her secret key, can decrypt the received ciphertext in *polynomial time.*

d. Not knowing the trapdoor between the public key and the corresponding secret key, the attacker faces an *intractable numerical problem* or an *infeasible computation.*

e. The attacker faces an infeasible problem of trying to recover the message by viewing the ciphertext sent to some receiver and the public key of the receiver.

In general, conditions a, b, and c must belong to the class *P*, and conditions d and e must belong to either the *NP-complete* class or the *NP-incomplete* class. Actually, conditions b and e define a *one-way function*. This means the encryption transformation is easy, but recovering the message is infeasible, even if the public key and ciphertext are known.

3. The RSA Cryptosystem. The RSA[RSA78] was introduced by three people, Ronald Rivest, Adi Shamir, and Leonard Adleman. They made the most spectacular contribution to public-key cryptography in 1978. It has remained secure, to date. The RSA system results from the number theory, and makes use of the fact that finding large(e.g. 200 digits) composite numbers is computationally easy. But, it is computationally infeasible to factor this composite number to be two prime numbers.

The RSA cryptosystem is a block cipher in which the plaintexts and ciphertexts are integers between 0 and $N - 1$ for some composite number, N. It resembles the exponential key-exchange system[DiH76] using exponentiation in modular arithmetic for its encryption and decryption operations. Unlike that system, RSA must do its arithmetics not over prime numbers, but over composite ones. The knowledge of a plaintext M, a modulus N, and an exponent are sufficient for calculation of $M^e \ mod \ N$. Exponentiation, however, is a one-way function with respect to the extraction of roots as well as logarithms. Depending on the characteristics of N, M, and e, it may be very difficult to invert.

RSA treats a plaintext block as an unsigned integer and operates with arithmetic *mod N*. Two keys, d and e, are used for decryption and encryption. Furthermore, these two keys are interchangeable. In the RSA system, a participant creates his own public and secret keys with the following scenario:

a. The receiver selects two large prime numbers at random, say, 100 digits each, p and q, and multiplies them together to obtain a composite modulus N.

b. The receiver suitably chooses a small odd encrypting exponent, e, that is relatively prime to $\phi(N) = (p-1) \times (q-1)$. Using $\phi(N)$ the receiver can calculate[Knd81] a number d as the multiplicative inverse of e, such that

$e \times d \equiv 1 \ mod \ \phi(N)$, is uniquely defined.

c. The receiver publishes the pair $K = (e, N)$ as the public key and keep the secret parameter d as the secret key k.

For this scheme, plaintext, ciphertext, and two keys(public key and secret key) are in the set $Z_N = \{0,1,...,N-1\}$. The encryption of a message M associated with a public key K is:

$$E_K(M) = M^e \ (mod \ N). \tag{2.3.3.1}$$

The decryption of a ciphertext C associated with a secret key k is:

$$D_k(C) = C^d \ (mod \ N). \tag{2.3.3.2}$$

The encryption process can be carried out by anyone who knows the public key. But, only the genuine receiver, who knows the factors of N, can reverse the process and decrypt. Because of symmetry in modular arithmetic, encryption and decryption are mutual inverses and commutative.

The quantity of $\phi(N)$ plays a critical role in Euler's theorem, which says that for any number a that is invertible modulo N

$$a^{\phi(N)} \equiv 1 \ (mod \ N), \tag{2.3.3.3}$$

or more generally

$$a^{k\phi(N)} \equiv 1 \ (mod \ N). \tag{2.3.3.4}$$

When the cryptogram $M^e \ mod \ N$ is raised to the power d, the result is

$$(M^e)^d = M^{ed} = M^{k\phi(N)+1} \equiv M \ (mod \ N), \tag{2.3.3.5}$$

the original message M.

We know that the strength of the exponential key-exchange protocol is not known to be equivalent to the difficulty of extracting discrete logarithms. Similarly, the strength of RSA has not been proven equivalent to factoring. There might be some method of taking the eth root of M^e without calculating d and thus breaking RSA without factoring. Rabin[Ram79] produced a variant of RSA, subsequently improved by William[Wih80], that is equivalent to factorization. Diffie and Rivest have observed[Dif82] the precise equivalence of Rabin's.

4. El Gamal Cryptosystem. El Gamal[EGt85] proposed a cryptosystem which was a variant of the key-exchange protocol of Diffie and Hellman. The general concept is given a generator g of cyclic group G_n with order s, where s is in $O(2^n)$, such that group elements can be represented as n-bit strings. The sender chooses a secret key k_1 uniformly distributed in $[0..n-1]$ and the receiver's public key is $y = g^{k_2}$, such that the encryption E is

$$E(m) = (g^{k_1}, y^{k_1} + m) = (c_1, c_2), \tag{2.3.4.1}$$

and the decryption D is

$$D(c_1, c_2) = c_1^{k_2} + c_2, \tag{2.3.4.2}$$

where + represents exclusive-OR which may be replaced by any invertible operation.

To attack this system is as hard as to break the Diffie and Hellman's key-exchange protocol. Note that the secret key k_1 is never used more than once in order to randomize the encryption process. The modulus is about the same size as the RSA's, thus the ciphertext is double the size of the plaintext.

5. Knapsack Cryptosystem. Merkle and Hellman[MeH78] proposed an interesting knapsack cryptosystem based on a known knapsack problem which belongs to the NP-complete class. They built trapdoors into the knapsack one-way function to produce the trap-door knapsack public-key cryptosystem. The general idea of knapsack system is transforming an easy, superincreasing, knapsack problem(in the special case where the components are 1, 2, 4, 8, etc., this is the elementary operation of binary decomposition), solved at linear time into a general knapsack problem.

Given a *knapsack vector* of integers $W = (w_1, w_2, \cdots, w_n)$, it is easy to add up the elements of any specified *subvector*. Presented with an integer S, however, it is not easy to find a *subvector* of W whose elements sum up to S, even if such a subvector is known to exist. This knapsack problem is well known in combinatorics and is believed to be extremely difficult in general. It belongs to the class of NP-complete problems, problems thought not solvable in polynomial time on any deterministic algorithms.

Merkle identified the knapsack problem as a theoretically attractive basis for a one-way function. The knapsack vector W can be used to encrypt an n-bit message $X = (x_1, x_2, \cdots, x_n)$, $x_i \in \{0, 1\}$, by taking the dot product $S = W \cdot X$ as the ciphertext. This process is easy and simply requires n additions. Inverting the function by finding a binary vector X such that $WX = S$ solves the knapsack problem. It is believed to be computationally infeasible if W is randomly selected. Despite this difficulty in general, many cases of the knapsack problems are quite easy. Merkle contrived to build a trapdoor into the knapsack one-way function by starting with a simple knapsack vector and converting it into a more complex form[MeH78]. The scenario of the system is as follows:

a. The participator first chooses the initial condition which is a sequence of super-increasing integers $W = (w_1, w_2, \cdots, w_n)$, that is $\sum_{i=1}^{j-1} w_i < w_j$, where $2 \le j \le n$.

b. The participator transforms the superincreasing vector into an general problem by determining a suitable field Z_m and an integer $t \in Z = \{0, 1, 2,..,m\text{-}1\}$, where m is randomly provided such that $m > \sum_{i=1}^{n} w_i$.

c. According to the following congruences, $k_i = w_i \times t(mod\ m)$, the vector $K = \{k_1, \cdots, k_n\}$ has no superincreasing property. Thus, K is the public key and (W, m, t) is the secret key

d. An intended sender sends an encrypted message which has been computed by $C = \sum_{i=1}^{n} k_i \times m_i$ to a legal receiver. The receiver decrypts it by computing t^{-1} and $M' = C \times t^{-1}$, such that the receiver uses the secret key W and M' to obtain the genuine message M.

From a geometric perspective, the knapsack cryptosystem is based on the intersection of two n dimensional planes, shown in Figure 8. The plaintexts lie along the intersecting pencil. If the solution exists, it must be a point on that line consisting of all integer coordinations and unique as well.

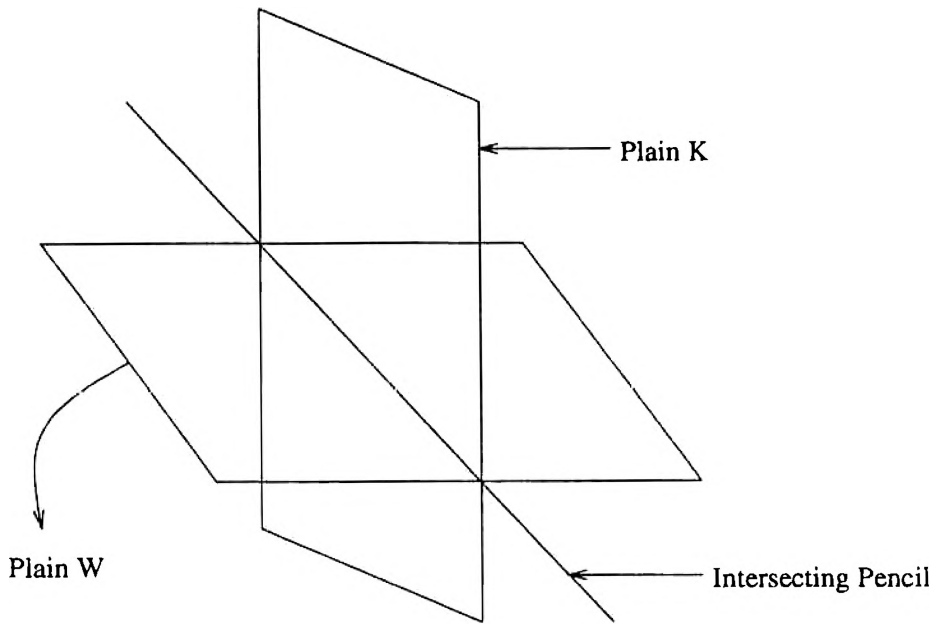

Figure 8. Graphical interpretation of knapsack cryptosystem

This process can be iterated to produce a sequence of knapsack vectors with more and more difficult knapsack problems by using transformations (t_1, m_1), (t_2, m_2), etc. The overall transformation is not, in general, equivalent to any single (t, m) transformation. The trap-door knapsack system is not designed for digital signature. This does not interfere with the use of the system for sending private messages, but requires special adaptation for signature applications.

Nineteen eighty-two, was an exciting year for public-key cryptanalysis. In March, Shamir sent out a research announcement: he had broken the single iteration Merkle-

Hellman knapsack system[Sha84]. Shamir had learned how to take a public knapsack vector and discover t' and m' that would convert it back into a superincreasing secret knapsack vector, not necessarily the same one that the originator had used, but the one that would suffice for decrypting messages encrypted with the public knapsack vector. The idea behind the attack is to consider the transformation of the easy knapsack problem *(W, t, m)* into the hard knapsack problem *(K, C, m)*,

$$K = t \times W \ (mod\ m). \tag{2.3.5.1}$$

From equation (2.3.5.1), there exists integers $y_1, y_2, \ldots, y_n$ such that

$$t^{-1}k_i - y_i m = w_i. \tag{2.3.5.2}$$

Therefore,

$$\frac{t^{-1}}{m} - \frac{y_i}{k_i} = \frac{w_i}{mk_i}. \tag{2.3.5.3}$$

Since the w_i's are superincreasing, $w_i \le 2^{i+1-n}m$ and $k_i \ge \frac{m}{n^2}$, we have

$$\frac{y_i}{y_1} - \frac{k_i}{k_1} = O(2^{i+1-n}n^2m^{-1}). \tag{2.3.5.4}$$

Thus

$$(\frac{y_2}{y_1}, \frac{y_3}{y_1}, \cdots, \frac{y_{d+1}}{y_1}) \tag{2.3.5.5}$$

is a simultaneous diophantine approximation to the vector

$$(\frac{k_2}{k_1}, \frac{k_3}{k_1}, \cdots, \frac{k_{d+1}}{k_1}), \tag{2.3.5.6}$$

for $d \ge 2$. Once the y_i's are found, it is easy to break the system. Shamir's attack is based on the assumption that displaying a good approximation might be unique and could be obtained by an algorithm[Leh83] to solve simultaneous diophantine approximations.

Many of the crucial observations about the weakness of the basic Merkle-Hellman system had been made earlier by Desmedt[DGo84]. Finally, the era of the knapsack system was ended by Brickell[Bre84][Bre85][Bre88], who invented a polynomial time algorithm allowing the secret key to be created from the public key. Although the knapsack system has been broken, its temporary success encouraged researchers to investigate other NP-complete problems for other possible encryption schemes.

6. The McEliece Coding Scheme. Within a short time, McEliece's[Mcr78] proposed another public-key system. McEliece's system makes use of the existence of a class of error-correcting codes, the Goppa codes, for which a fast decoding algorithm is known. His idea was to construct a Goppa code and disguise it as a general linear code, for which the decoding problem is NP-complete. There is a strong parallel with the trapdoor knapsack system in which a superincreasing knapsack vector, with knapsack problem which is simple to solve, is disguised as a general knapsack vector whose knapsack problem is NP-complete.

In a knapsack system, the secret key consists of a superincreasing knapsack vector v, together with the multiplier w and the modulus m that disguise it. In McEliece's system, the secret key consists of the $(k \times n)$ generator matrix G for a Goppa code together with a $(k \times k)$ nonsingular scrambling matrix S, and a $(n \times n)$ permutation matrix P that disguise it. The public-key appears as the encoding matrix $G' = SGP$ of an arbitrary linear code, for which a fast algorithm for error-correcting is not known.

a. To encrypt a k-bit data block X into a n-bit message Y, the sender multiplies it by the receiver's public encoding matrix G', then adds a locally generated n-bit noise block e of weight t.

b. To decrypt, the receiver multiplies the received message Y by P^{-1}, decodes $YP^{-1} = (XS)G + eP^{-1}$ to get a Goppa code then multiplies this by S^{-1} to recover the sender's data block X.

McEliece's system has never achieved wide acceptance and has probably never even been considered for implementation for any real application. This may be due to the large public key, requiring in the order of a million bits. Or, it may be because the McEliece's system bears a structural similarity to knapsack systems which can be broken. Whereas, cryptographic researchers are still trying to improve the McEliece's system, because of its error-correcting feature. Adams[AdM89] shows that carefully choosing parameters k and t, the dimension of code and the maximum correcting errors, will increase the cryptanalysis's endeavor and decrease its data expansion.

With this article, Adams demonstrates the optimal values of t is 37, instead of 50, so that the work factor of attack is about 2^{84} and the dimension k increases from 524 to 654. Thus, the data expansion is reduced. However, Korzhik and Turkin[KoT91] proposed an approach to attack this well-known algebraic coding system, that is based on iterative optimization algorithm[KoT91]. This algorithm guarantees correction of a linear code with at most t errors. Their experiment shows that a (1024,654) BCH code with $t = 65$ can be attacked within 60 hours on personal computers.

7. Cryptanalysis of RSA Cryptosystem. Ever since RSA was published, researchers have tried to break it. They are trying to solve the associated two hard problems:

a. Factoring large numbers: if the modulus N can be factored, then the secret key would be derived from the public key.

b. Discrete logarithm problem: since the public key e and the ciphertext C are known(i.e. $M^e = C \ (mod \ N)$), recovering the message M becomes a feasible attempt. On the other hand, determining if a large number is prime is also a related work. But RSA has resisted various kinds of attacks.

Many papers provide comprehensive understanding on factoring problems[Leh86][Leh87][Mop87][MoS90][Pom84], and on discrete logarithm problems[Cod84][Oda84][LaO90]. Pomerance[Pom84] proposed the quadratic sieve factoring algorithm, which has been applied with parallel techniques in order to break RSA. The idea is if N is to be factored and integers r and s can be found so that

$$r^2 = s^2 \ (mod \ N), \tag{2.3.7.1}$$

then

$$(r + s)(r - s) = 0 \ (mod \ N). \tag{2.3.7.2}$$

If either

$$N > gcd \ (N, r+s) \ \ or \ \ N > gcd \ (N, r - s) \tag{2.3.7.3}$$

then a factor of N is determined.

To find such pairs (r, s), we start with a randomly selected $t \in [2, N - 1]$ and define $u = t^2 \ (mod \ N)$. The number u is defined as a quadratic residue of N. It might happen that u is a square, but in general, this will not happen. So that we choose different values t_i and compute $u_i = t_i^2 \ (mod \ N)$. By multiplying some of u_i, we obtain a perfect square, so that the quadratic sieve algorithm can be employed.

An interesting fact is that the best algorithm for solving the discrete logarithm problem under modulus p and the best algorithm for factoring N require about the same

amount of computations as $p \approx N$. Most of the fast factoring algorithms have been shown to run in time

$$\exp\,[(1 + o(1))((log\ n)(log\ \log n))^{1/2}],$$

as n becomes large; then it is of interest in cryptography. One of the fascinating questions about RSA is; is it as secure as factoring? There are no known attacks on RSA that are faster than factoring the modulus.

As mentioned above, RSA requires two very large prime numbers, p and q, to construct its moduli. How to test primality becomes a critical issue. The prime number theorem states that the approximation $\frac{n}{\ln n}$ specifies accurate estimate of $\pi(n)$, the number of primes less than or equal to n.

For instance, to find a 100-digit prime number we require to test approximately $\ln 10^{100} \approx 230$ randomly chosen 100-digit numbers for primality. Actually, the numbers can be cut into half by considering odd integers only. Recall Fermat's theorem, it says if p is a prime then

$$a^{p-1} \equiv 1\ (mod\ p), \tag{2.3.7.4}$$

for every a relatively prime to p. Conversely, if $a^{p-1} \not\equiv 1\ (mod\ p)$ for any a then p must be composite. Note that if the equation (2.3.7.4) is satisfied with arbitrary number a, then p is said to be pseudoprime to base a and a is a witness to the compositeness of n. In particular, there exists such a integer n, the so-called carmichael number, and equation (2.3.7.4) holds for all a relatively prime to n, so it is not sufficient to merely use Fermat's theorem for primality test. Thus, Rabin[Rab80] proposed a probabilistic algorithm for testing primality:

a. Randomly choose s integers $a_1, a_2, \ldots, a_s$ with $1 < a_i < n$.

b. Test if the equation (2.3.7.4) holds,

   1). if it does not hold for any a_i then outputs <COMPOSITE NUMBER>,

   2). else declares n to be a prime number with error probability at most $\frac{1}{2^s}$.

8. Key Management. Key management is a major problem in networks protected by cryptographic technique. For example, regarding the secret-key system, if there are n subscribers in the network, one will need $n(n-1)/2$ different secret keys for every possible pair of subscribers. This is an impractical situation in a large network. A popular solution to this problem is to avoid the assumption that the two parties share a secret key. They use an entity, called *Key Distribution Center(KDC)*, trusted by all network processors. They share a key with each subscriber and use these in a process to provide additional keys to subscribers as needed.

a. *KDC* delivers a randomly chosen key k_i to subscriber i in the system, for $i = 1, 2, \ldots, n$.

b. When subscriber i wishes to communicate securely with subscriber j, he sends a request to *KDC*.

c. *KDC* randomly generates a new session key k_{ij} and encrypts it under k_i and k_j. Then *KDC* sends each cryptogram to subscriber i and subscriber j.

d. Subscriber i and j decrypt the cryptograms they have just received and thereby obtain the session key k_{ij}, which is to be used for encryption by two subscribers.

As the protocol described, every subscriber shares a secret key with *KDC* and can only authenticate messages explicitly meant to him. In a public-key network, each subscriber has the public key of *KDC* which is a trusted distributor of each subscriber's public key. Then the key distribution protocol becomes:

a. each subscriber has to register his/her public key to *KDC*,

b. subscriber *i* sends a request(communicating to subscriber *j*) to *KDC*, and then *KDC* encrypts *j*'s public key under *KDC*'s secret key. This cryptogram, so-called *certificate*, is sent to subscriber *i*.

Now the subscriber *i* believes the public-key's authentication and is happy to use it. These protocols are called *three party protocols*, and have been studied extensively in[BBF83][BMR90][DeS81].

D. AUTHENTICATION AND DIGITAL SIGNATURE

Cryptography has two main applications over data security, secrecy, and user/message authentication. Securely authenticated exchange is essential for network security. Authentication issues become crucial due to networks spreading worldwide with users accessing via remote terminals. In general, messages are transmitted via insecure channels so that the following issues might happen; modifying, reusing, and blocking the contents of messages. The receiver must determine that the messages comes from the genuine sender and that the contents of the messages have been not altered.

1. Authentication. The extensive use of open networks and distributed systems poses increasing threats to the security of data communications involving end-users and network components. Authentication is nothing more than the determination by the authorized receiver, or perhaps the trusted third party, that a particular message was sent by the authorized sender under the existing authentication protocol, which has not been altered or substituted for.

The essential feature in the authentication scheme[BsM90] is that authentication depends on the measure of redundant information inherently present in the structure of the message. In other words, authentication is concerned with schemes that do not accept incorrect messages, without the legal party recording the message. This captures the fact that the adversary cannot fool one side to accept a message without the other side being involved. On the other hand, coding theory is concerned with codes that introduce redundancy in such a way, that the most likely alternations to the encoded messages are in some sense close to the codeword they derive from. From cryptographic point of view, there are two different implementations.

a. Secret-key cryptosystem: generally speaking, a secret-key system delivers secrecy and authentication. For example, using the cipher-block chaining mode of DES it produces a 64-bit message authentication code(MAC) appending to the message to be authenticated.

b. Public-key cryptosystem: authentication scheme is dominated by the sender, that is, the sender executes $C = D_k(M)$ and the receiver applies the sender's public key to authenticate the message. So, the public-key system delivers either secrecy or authentication. If you want to have both, you must use the public-key system twice, one for secrecy and the other for authentication.

It is not difficult to design authentication protocols, but many of them are not secure. For instance, Figure 9 demonstrates a trivial attack to the secret-key authentication scheme. Here, X is randomly generated by party A, and E(X) is the value of X encrypted by E. Attacker C intercepts the first flow sent by A to B. He wishes to pretend to be B, but he cannot decrypt E(X) directly. Whereas, the attacker takes advantage of A to accomplish this purpose. The attacker C, starts a second flow with A, pretending to be party B starting this communication. When this protocol is finished by the forth flow, A does communicate with attacker C, instead of party B.

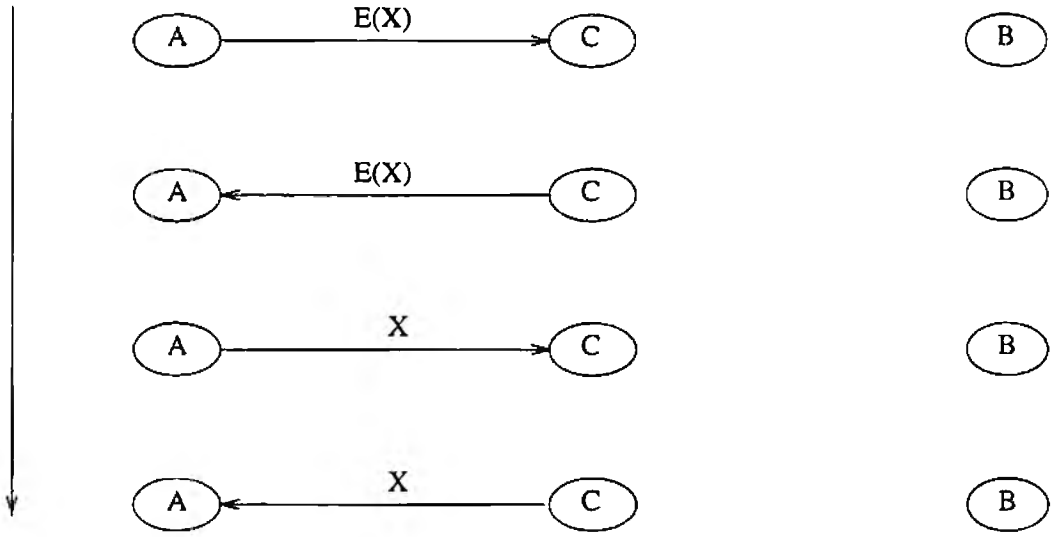

Figure 9. An impersonation attack on authentication scheme.

The above protocol is defined as one-way authentication. Two-party authentication protocols are designed for correctly authenticating messages exchanged between two parties in a communication network. The basic idea behind it is to authenticate a message from the sender. We use a challenge previously sent by the receiver. Usually, the sender

combines the challenge with the authenticated message, and the receiver verifies such a combination. In one-way authentication, it is sufficient that one party marks acceptance; in the two-way authentication we require both parties to mark acceptance.

Bird[BJM92] proposed a new two-way authenticated protocol which is provably secure, efficient, and practical. The built-in block of such a protocol is based on the CBC-mode of the DES cryptosystem. With their proposition, they indicate that the current authentication protocols are breakable, such as ISO authentication standard, and also present their model's secure strength. They have tried to identify and avoid the security weaknesses of other protocols. Each response to a given challenge is encrypted, so that the attacker cannot pretend to be a legal party by directly relaying each response. Their model uses only three exchanged messages.

2. Digital Signature. With the rapid expansion of computer application and data communication in networks, the paperless electronic offices have been predicted. Digital signatures will replace hand written signatures to sign documents, contracts, and electronic messages of all kinds. In general, a digital signature is a protocol that produces the same effect as a real signature. It is a mark that only the sender can make, but other people can easily verify that it belongs to the sender. Just like a real signature, digital signatures must meet the following conditions; unique, impossible to deny, easy to verify, and not replayable. Indeed, these methods can be divided into two classes:

a. Direct signature authentication performed by receivers only.

b. Indirect signature authentication performed by the third notarized party.

The characteristic attribute of a digital signature must depend on the message and a unique secret number of the signer. From a practical perspective, the signature should not be the same size as the message. Therefore, a compression function must be applied at first. Generally, compression methods are similar to hash functions, that is, a compression function should be designed in such a way that there are no two identical results for two different messages.

3. Secret-Key Digital Signature. With a secret-key encryption system, the secrecy of the key guarantees the authenticity of the messages, as well as its secrecy. However, the secret-key encryption does not prevent impersonation or substitution attack. The receiver can create an identical message sent by the sender, since it also has access to the key. With the secret-key encryption such as DES, an arbiter is needed to prevent counterfeiting. Here is an outline of the digital signature protocol and is shown in Figure 10:

a. The sender has the k_s in common with the arbiter, and the receiver has the key k_r in common with the arbiter.

b. The sender first sends $E_{k_s}(M)$ to the arbiter, and then the arbiter decrypts M using k_s.

c. The arbiter sends $E_{k_r}(M, ID_s, E_{k_s}(M))$ to the receiver. The receiver decrypts it and knows the message coming from the sender.

d. The receiver cannot decrypt $E_{k_s}(M)$ since it is encrypted by k_s. However, the receiver files a copy of M and $E_{k_s}(M)$ in case there is a future dispute.

e. If the sender denies the message, then the receiver produces M and $E_{k_s}(M)$ to the arbiter who decrypts $E_{k_s}(M)$ and certifies that only the sender could have produced $E_{k_s}(M)$.

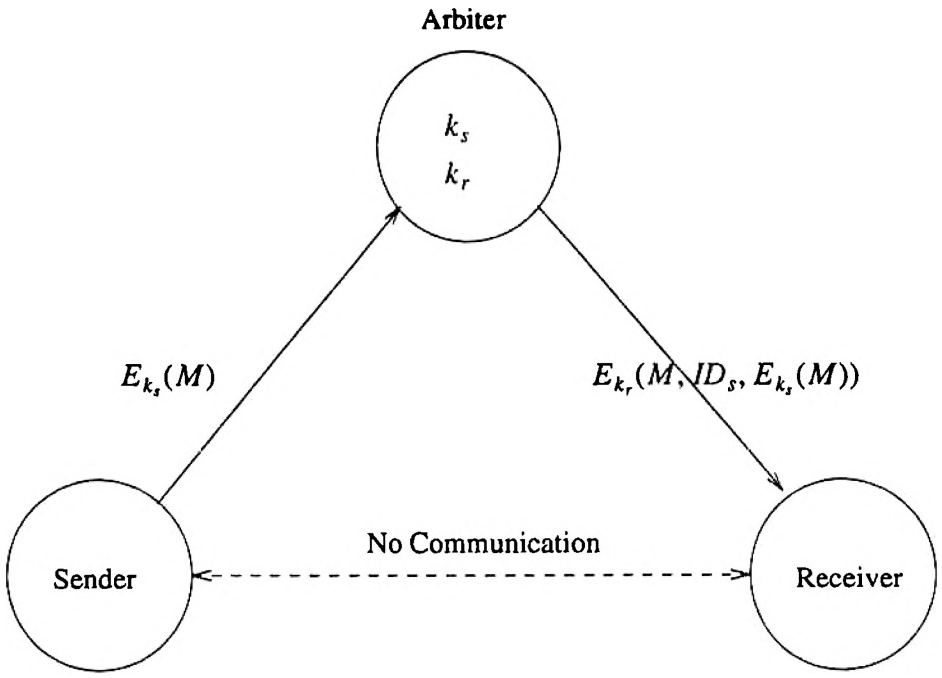

Figure 10. Secret-key digital signature

4. Public-Key Digital Signature. Public-key encryption systems are ideally suited to create a digital signature. The sender should be able to use the secret key to sign a document, whereas all users can verify the authenticity of the signature by his public key. Under RSA, the sender wants to sign message M:

a. it computes a compression function $CF(M)$,

b. $SG(M) = D_A(CF(M))$, where SG represents digital signature,

c. forwards $(M, SG(M))$ to the receiver.

The receiver verifies the signature by:

a. applying the public key of the sender, $CF(M) = E_A(SG(M))$,

b. computing the compression function $CF'(M)$and then accepting the signature if $CF'(M) = CF(M)$.

Under the scheme of El Gamal[EGt85], the signature for m is the pair (u, w), $u, w \in [0, p-1]$. Each subscriber selects two secret numbers, s, t, then compute $a = q^s \ mod \ p$, and $u = q^t \ mod \ p$. Next he puts a into a public directory, so that all users are able to access it. Now the signature (u, w) for m is satisfied

$$m \equiv s \times u + t \times w \ mod \ (p-1). \tag{2.4.4.1}$$

While the verification procedure is checking if

$$q^m = a^u \times u^w. \tag{2.4.4.2}$$

Note that the signature scheme is as secure as the discrete logarithm problem, and the signature is still double the size of the document.

5. Shared Digital Signature. For the sake of security, a document would be signed by t individuals rather than one person. An organization or a company may set up its policy to choose n individuals, and permit any subset consisting of $t \leq n$ persons to sign documents. Desmodt and Frankel[DFr90] proposed the shared generation of the RSA signature. This is better than letting each individual create its own signature. This scheme lessens communication overheads and does not need a large public-key directory. Furthermore, there is no communication among individuals or shareholders. The secret key is not revealed to each individual, and nobody can perform a substitution attack. The

scheme is based on Lagrange interpolation polynomial over integers. The RSA signature scheme is $SG \equiv m^d \ (mod\ n)$. A dealer distributes a partial result $SG_i = m^{s_i} \ (mod\ n)$ to each individual. Then any t individuals can create $SG \equiv m \times \prod SG_i$.

Each share s_i is generated by a polynomial $f(x)$ with degree t-1, such that $f(0) \equiv d - 1 \ mod\ \phi(n)$. Using Lagrange interpolation

$$f(x) = \sum_{i=1}^{t} y_i \times (\prod_{j \neq i}^{t} \frac{x - x_j}{x_i - x_j}). \tag{2.4.5.1}$$

Thus, the share s_i of each user i is

$$s_i = y_i \times [\prod_{j \neq i}^{t} (\frac{0 - x_j}{x_i - x_j})]. \tag{2.4.5.2}$$

6. Undeniable Digital Signature. The concept of undeniable signature was proposed by Chaum and van Antwerpen[Cdv90]. Such a protocol is to guarantee the recipient of this commitment should be able to ensure that the issuer cannot deny it and the recipient should be unable to replay the commitment to anyone else without the issuer's cooperation. The validity of an undeniable signature is verified by the recipient establishing a challenge and response protocol with the signer. If the test is successful, the probability of the signature being valid is high. Otherwise, either the signer denies a valid signature later or the signature is invalid.

The security assumption of undeniable signatures is based on the difficulty of computing discrete logarithms in a cyclic group of prime order. In this scheme, the public keys are g and g^x, where g is a generator of the group and x is the secret key. Given an arbitrary message m, the signature has the form m^x.

E. CRYPTOGRAPHIC PROTOCOL

1. Zero-knowledge Protocol. Zero-knowledge protocol is a new cryptographic research area. It is an interactive-proof system. The verifier just believes that the assertion claimed is true and makes no effect to test his assertion. It is a robust prover against attempts of verifier to extract information by means of interaction. Goldwasser, Micali, and Rackoff[GMR85] make this notion precise. They call an interactive-proof system for language L zero-knowledge, for all $x \in L$. The verifier can compute, after participating in the interaction with the prover, via two probabilistic interactive machines computing in polynomial time on the input x. Initially, both machines access to a common input tape. Each machine has its own tapes and communication tapes, in addition to the common input tape. Moreover, one machine is not allowed to monitor the internal computation of the other or to read the other's coin tosses, current states, and programs. The verifier's output is either accepted or rejected. We require the prover and the verifier to follow the predeterminated protocol, which succeeds in two conditions, completeness and soundness.

By far, the most important result about zero-knowledge was given by Goldrach, Micali, and Wigderson[GMW86]. They showed the following result:

> *If there exists polynomial-time indistinguishable encryption scheme then every NP language has a computational zero-knowledge interactive proof system.*

We exemplify the zero-knowledge proof systems for graph isomorphism. The fact that graph isomorphism has an efficient proof system, is because it is in NP. In the following protocol, the prover needs a probabilistic polynomial-time machine, an auxiliary input, and the isomorphism between the input graphs. Let ϕ denote the isomorphism

between $G_1(V, E_1)$ and $G_2(V, E_2)$. The following steps executed for n times, each time we use random coin tosses:

a. The prover randomly generates an isomorphism H of G_1 and sends it to verifier. This is done by an one way permutation π such that $(\pi(x), \pi(y)) \in$ *edges of* H if and only if $(x,y) \in E_1$.

b. The verifier chooses $\alpha \in \{1,2\}$ at random and sends it to the prover.

c. The prover justifies if $\alpha = 1$ then sends π to the verifier, else sends $\pi\phi^{-1}$.

d. If this one way permutation is valid, then it goes to a., otherwise this protocol is stopped.

If the verifier iterates n times successfully, then he accepts.

This proof is zero-knowledge since whatever the verifier receives is useless, as the prover can generate random isomorphic copies of the input graphs by himself. Since the graph 3-coloring problem has been proved by this zero-knowledge protocol, we recognize that all languages in NP have zero-knowledge proof systems assuming the existence of secure encryption scheme. Reviewing this setting, cryptography is a much wider and vital subject than the classical setting.

2. Multi-party Protocol. In society, communications is group-oriented. That is, it is the communications between an organization and another. Desmedt[Dey88] introduced the group-oriented cryptosystem as a mean of sending messages to a group of n people, so that only certain subsets of these n people are able to decrypt the message. He introduced two different groups. One is called *known*, if the sender has to know each member's

public key. The other is *anonymous*, if there is a public key independent of the members. For both types of groups, the decipherment needs the cooperation of all the members.

This research topic has been further studied by Desmedt and Frankel[DFr90], Frankel[Fry90], and Hwang[Hwt91]. Desmedt and Frankel[DFr90] designed a system where any k honest members can decrypt the received ciphertexts. This system is assumed to have an anonymous group and trusted parties. Meanwhile, Frankel's protocol does not match its name(for large group oriented networks). His proposal is based on the assumption of trusted clerks and tamper-proof modulars. In a large network, these assumptions are impractical and inefficient. However, Hwang proposed a protocol base on the Diffie and Hellman's key-exchange protocol and Shamir's secret sharing scheme[Sha79]. The sender has more authority to decide the messages destination without the cooperation of the receiving group. The most interesting property is that his system does not need the trusted party at all.

Shamir[Sha79] introduced a (t, n) threshold scheme with $n \geq 2t + 1$. It permits a secret value to be shared by a set of n participants(See Figure 11).

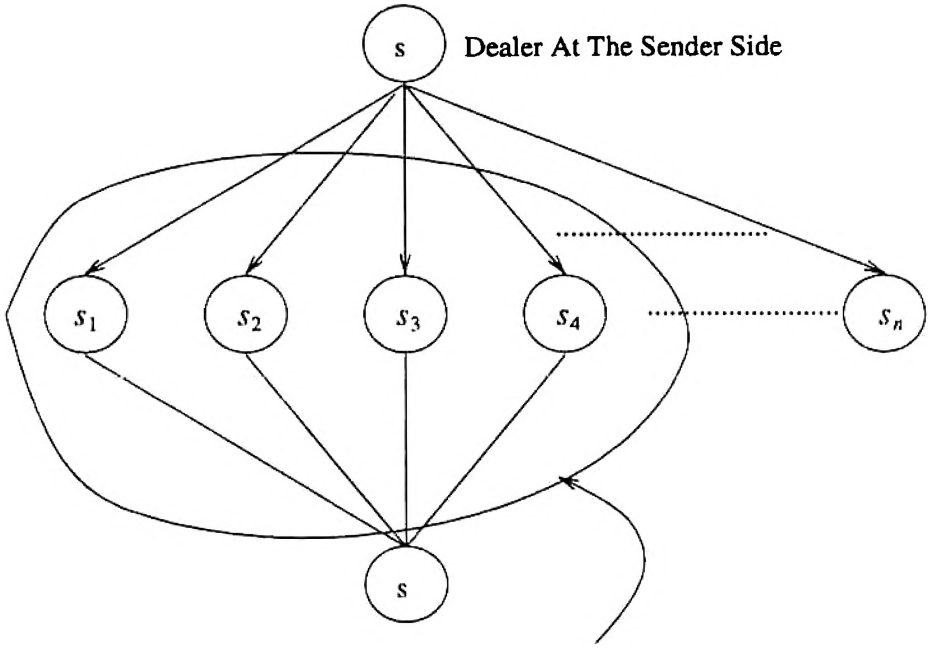

Figure 11. Shamir's secret sharing scheme.

Even when at most t out of n participants are malicious, the secret value can still be recovered. The general concept is based on a polynomial interpolation. Given a polynomial $f(x)$ with degree at most t, $f(x) = a_0 + a_1 x + \cdots + a_t x^t$ where $f(0) = s$, the secret value. A distinguished participant among them, a so-called dealer, evaluates shares of s by computing

$$f(1) = s_1,\ f(2) = s_2, \ldots,\ f(i) = s_i, \ldots,\ f(n) = s_n$$

and distributes them to each participant's identification i.

Given any subset of s_i, which size is t+1, the polynomial can be reconstructed by interpolation, and then the secret is $s = f(0)$, thus s can be used as secret communication and user authentication. In practical situations, some participants might be traitors and might contribute bad shares as their shares of secret. This will prevent the correctness of computation for honest participants. Regarding this, a verifiable secret sharing has been proposed by Chaum, Crepeau, and Damgard[ChC88]. Loosely speaking, a verifiable secret sharing is a n+1-party protocol, through which a sender can distribute to the receivers pieces of a secret s recognizable through an interpolation polynomial $f(x)$. A verifiable secret sharing consists of two phases, commitment and computation. For the first phase, the dealer convinced that the secret value has been committed by all honest participants. And later on, all honest participants will know the secret value. For the second phase, the value is recovered by the cooperation of the honest majority.

The notion of verifiable secret sharing differs from Shamir's secret sharing, in that the secret is recognizable and the piece should be verifiable, as authentic. It has been used as the building block for protocols with the honest majority[BMR90][RaB89]. Moreover, improvements in efficiency and fault-tolerance are also made.

III. THE SPLINE CRYPTOSYSTEM

A. PRELIMINARY

The feasibility of implementing the interpolating cubic spline function as the transformation for encipherment and decipherment[HqH92] has been demonstrated. The encryption method can be viewed as computing a transposed polynomial. The main characteristic of the spline cryptosystem is that the domain and range of encryption are defined over real numbers, instead of traditional integer numbers. In addition, the spline cryptosystem can be implemented in terms of simple matrix computations. Furthermore, an advanced structure of the spline cryptosystem, the random spline cryptosystem, will be introduced in order to resist more sophisticated attacks.

Generally, cryptography is defined over finite fields. Such as the RSA system, the encryption process and the decryption process are under *GF(N)*, where N is a large composite number. A new attempt is to construct a cryptosystem based on real number assumption. It seems to be a *maxim* that the real numbers are not allowed to be applied to any known cryptosystem. In fact, through theoretical verification and experimental results of executing our proposed cryptosystem, the total error (double precision) generated by this system is nearly zero ($<10^{-7}$) and negligibly small. In order to construct such a system, a precise, well-behaved *cubic spline function* is introduced as the fundamental building block of this new cryptosystem, and then it has been named as the *spline cryptosystem*. The first appearance of the cryptosystem is linear, and it has been modified into non-linear transformation via substitution and permutation, thus making it more capable of resisting the cryptoanalysts' attacks.

The spline cryptosystem has two major achievements. First, the system is defined over real numbers. Secondly, without providing precise boundary conditions, recovering messages from corresponding cryptogram is unsuccessful. Moreover, the cryptosystem can be implemented on any current personal computer via simple encryption and decryption processes.

This chapter is organized as follows. The second section introduces the elementary mathematical background of *cubic splines*. In the third section, the encryption transformation and the decryption transformation are built up. In the fourth section the probably generated error is analyzed. A concrete example is demonstrated in the fifth section. Finally, in the sixth section, the security of spline cryptosystem is discussed.

B. DEFINITION AND NOTATION

We consider the interval [0, 1] and subdivide it by a mesh of points corresponding to the location of the nodes:

$$\delta : 0 = x_0 < x_1 < \ldots < x_{n+1} = 1. \tag{3.2.1}$$

The cubic spline[AdR76] $S(x)$ within δ, or the spline on the δ, is a function which is continuous together with its first and second derivatives on [0, 1] and coincides with a cubic polynomial in each subinterval $[x_{j-1}, x_j]$ $(j = 1, 2, \cdots, n+1)$. According to the fundamental theory of spline[Scl81], for arbitrary real number k_i $(1 \le i \le 4)$ representing the required boundary condition, and m_j $(1 \le j \le n)$ representing the value at each node, the spline $S(x)$ on δ is uniquely defined by the following boundary conditions,

$$S'(0) = k_1, S(0) = k_2, S(1) = k_3, S'(1) = k_4, \tag{3.2.2}$$

and the value at each node is defined as

$$S(x_j) = m_j \ (1 \le j \le n). \tag{3.2.3}$$

Now only the case of equally-spaced mesh is considered, i.e.

$$\delta : x_j = j \times h \ (0 \le j \le n+1), \tag{3.2.4}$$

where $h = 1/(n+1)$ is the length of the subinterval. In this case, the spline $S(x)$ on δ can be represented as

$$S(x) = \sum_{j=-1}^{n+2} d_j \times B(t) \text{ and } t = (x - x_j)/h, \ (0 \le x \le 1), \tag{3.2.5}$$

where, $x_j = j \times h$, $(-1 \le j \le n+2)$.

$B(t)$ is the normalized cubic B-spline represented in equation (3.2.6) and shown in Figure 12:

$$B(t) = \begin{cases} -|t|^3/6 + |t|^2 - 2|t| + 4/3 & \text{if} \quad 1 \le |t| \le 2 \\ |t|^3/2 - |t|^2 + 2/3 & \text{if} \quad 0 \le |t| \le 1 \\ 0 & \textit{otherwise}, \end{cases} \tag{3.2.6}$$

and the constant d_j $(j = -1, 0, \cdots, n+2)$ is said to be a *control point.*

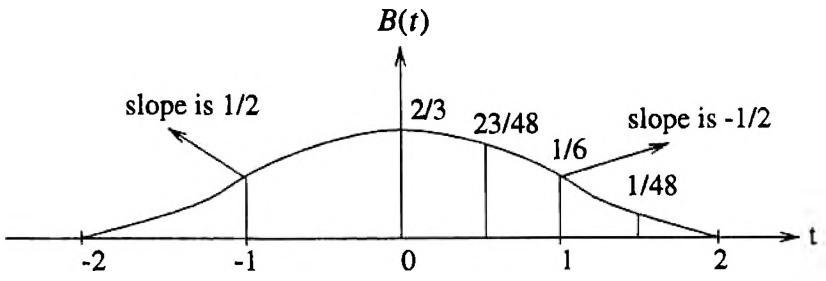

Figure 12. Normalized cubic B-spline curve.

Substituting (3.2.5) into (3.2.2) and (3.2.3) yields the algebra system:

$$P \times [d_{-1}, \cdots, d_{n+2}]^T = [k_1, k_2, m_1, \cdots, m_n, k_3, k_4]^T. \tag{3.2.7}$$

The matrix form of P is

$$P = \begin{bmatrix} B'_{-1}(x_0) & B'_0(x_0) & B'_1(x_0) & \cdots & B'_{n+2}(x_0) \\ B_{-1}(x_0) & B_0(x_0) & B_1(x_0) & \cdots & B_{n+2}(x_0) \\ B_{-1}(x_1) & B_0(x_1) & B_1(x_1) & \cdots & B_{n+2}(x_1) \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ B_{-1}(x_n) & B_0(x_n) & B_1(x_n) & \cdots & B_{n+2}(x_n) \\ B_{-1}(x_{n+1}) & B_0(x_{n+1}) & B_1(x_{n+1}) & \cdots & B_{n+2}(x_{n+1}) \\ B'_{-1}(x_{n+1}) & B'_0(x_{n+1}) & B'_1(x_{n+1}) & \cdots & B'_{n+2}(x_{n+1}) \end{bmatrix},$$

where $B_j(x) = B((x - x_j)/h)$ and P is a $(n+4) \times (n+4)$ matrix. For example, $n = 3$:

$$P = \begin{bmatrix} \frac{-1}{2h} & 0 & \frac{1}{2h} & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 & 0 & \frac{-1}{2h} & 0 & \frac{1}{2h} \end{bmatrix}.$$

Let ξ_j be the middle point of subinterval $[x_{j-1}, x_j]$ $(j = 1, 2, \cdots, n)$, and $c_j = S(\xi_j)$ be the ciphertext. Then from (3.2.5) we have

$$Q \times [d_{-1}, \ldots, d_{n+2}]^T = [k_1, k_2, c_1, \cdots, c_n, k_3, k_4]^T. \tag{3.2.8}$$

The matrix form of Q is

$$Q = \begin{bmatrix} B'_{-1}(x_0) & B'_0(x_0) & B'_1(x_0) & \cdots & B'_{n+2}(x_0) \\ B_{-1}(x_0) & B_0(x_0) & B_1(x_0) & \cdots & B_{n+2}(x_0) \\ B_{-1}(\xi_1) & B_0(\xi_1) & B_1(\xi_1) & \cdots & B_{n+2}(\xi_1) \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ B_{-1}(\xi_n) & B_0(\xi_n) & B_1(\xi_n) & \cdots & B_{n+2}(\xi_n) \\ B_{-1}(x_{n+1}) & B_0(x_{n+1}) & B_1(x_{n+1}) & \cdots & B_{n+2}(x_{n+1}) \\ B'_{-1}(x_{n+1}) & B'_0(x_{n+1}) & B'_1(x_{n+1}) & \cdots & B'_{n+2}(x_{n+1}) \end{bmatrix},$$

where $B_j(x) = B((x - x_j)/h)$ and Q is a $(n+4) \times (n+4)$ matrix. For example, $n = 3$:

$$Q = \begin{bmatrix} \frac{-1}{2h} & 0 & \frac{1}{2h} & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 & 0 \\ \frac{1}{48} & \frac{23}{48} & \frac{23}{48} & \frac{1}{48} & 0 & 0 & 0 \\ 0 & \frac{1}{48} & \frac{23}{48} & \frac{23}{48} & \frac{1}{48} & 0 & 0 \\ 0 & 0 & \frac{1}{48} & \frac{23}{48} & \frac{23}{48} & \frac{1}{48} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 & 0 & \frac{-1}{2h} & 0 & \frac{1}{2h} \end{bmatrix}.$$

Both matrixes P and Q, are tridiagonal and nonsingular.

The following section demonstrates the processes of applying the interpolating cubic spline to the encryption transformation and the decryption transformation.

C. ENCRYPTION AND DECRYPTION

For our scenario, suppose that A and B (also known as Alice and Bob) are two subscribers of the spline cryptosystem, and the encryption algorithm and the decryption algorithm are represented as E_k and D_k. Obviously, this spline cryptosystem is a symmetric cryptosystem. Therefore, both subscribers use the same key in order to establish a secure communication channel. The security of the system relies on the concealment of the key which is generated by a key-exchange protocol. According to the Diffie and Hellman scheme[DiH76], the common session key is k_{AB}.

Now the system will be outlined as follows. The spline cryptosystem encrypts n-character message with k_{AB}. Meanwhile, there exist seven working subkeys, k_1, k_2, k_3, k_4, k_5, k_6, and k_7, which are derived from k_{AB}. These subkeys provide a secure environment for our system and they are generated as follows:

$k_1 = k_{AB} \bmod 256$, is one-byte length,

$k_2 = LCS(k_1)$ (Left Circular Shift by one bit), is one-byte length,

$k_3 = LCS(k_2)$ (Left Circular Shift by one bit), is one-byte length,

$k_4 = LCS(k_3)$ (Left Circular Shift by one bit), is one-byte length,

$k_5 = LCS(k_4)$ (Left Circular Shift by one bit), is one-byte length,

$k_6 = LCS(k_5)$ (Left Circular Shift by one bit), is one-byte length,

$k_7 = LCS(k_6)$ (Left Circular Shift by one bit), is one-byte length.

k_1, k_2, k_3, and k_4 are four boundary conditions, $k_5 \bmod 10$ is n, the number of characters in each block. k_6, k_7 are used for scrambling the original cryptogram to achieve the non-linearity of the system.

Next, the encryption algorithm and the decryption algorithm are introduced. A tries to send a n-character $(m_1, m_2, \ldots, m_n)$ message to B by executing E_k with the following two steps:

$$step\ 1: P \times \bar{d}^T = [k_1, k_2, m_1, m_2, \cdots, m_n, k_3, k_4]^T,$$

$$step\ 2: Q \times \bar{d}^T = [k_1, k_2, c_1, c_2, \cdots, c_n, k_3, k_4]^T,$$

where m_i is the corresponding plaintext value. With the first step, we obtain the vector

$\vec{d}^T = [d_{-1}, d_0, \ldots, d_{n+2}]$ where matrix P is specifically defined. The second step results in a ciphertext vector. The first two components, as well as the last two components of the ciphertext vector, are given by boundary conditions, and the remaining members, $c_1, c_2, \ldots, c_n$, are corresponding ciphertext values. Because all the algebraic computations in this system are linear, it cannot resist a determined attack. For the sake of security, ciphertext values need to be scrambled via substitution (S-box) and permutation (P-box). More specifically, Figure 13 illustrates these two transformations dominated by k_6 and k_7. The constructions of the S-box and the P-box are given below:

S-box: indeed, we would like to scramble the mantissa and leave the exponent untouched. We shift left circular, with mantissa of each real number c_i, by k_6 mod 8 bits, then insert them into the P-box sequentially.

P-box: we pick up three digits from k_7, say n_1, n_2, and n_3 sequentially from right to left. Let $n_i = n_i \bmod n$, $i = 1, 2, 3$, then move c_{n_1},c_{n_2}, and c_{n_3} to the rightmost sequentially. If any of n_1, n_2, or n_3 are equal to 0 or n, then the real numbers are not moved.

For example, let us consider the case $c = (c_1, c_2, c_3, c_4, c_5, c_6)$, $k_7 = 109$, $n_1 = 9$, $n_2 = 0$, $n_3 = 1$. Then it produces $n_1 = 3$, $n_2 = 0$, $n_3 = 1$, and its output is $c' = (c_2, c_4, c_5, c_6, c_3, c_1)$.

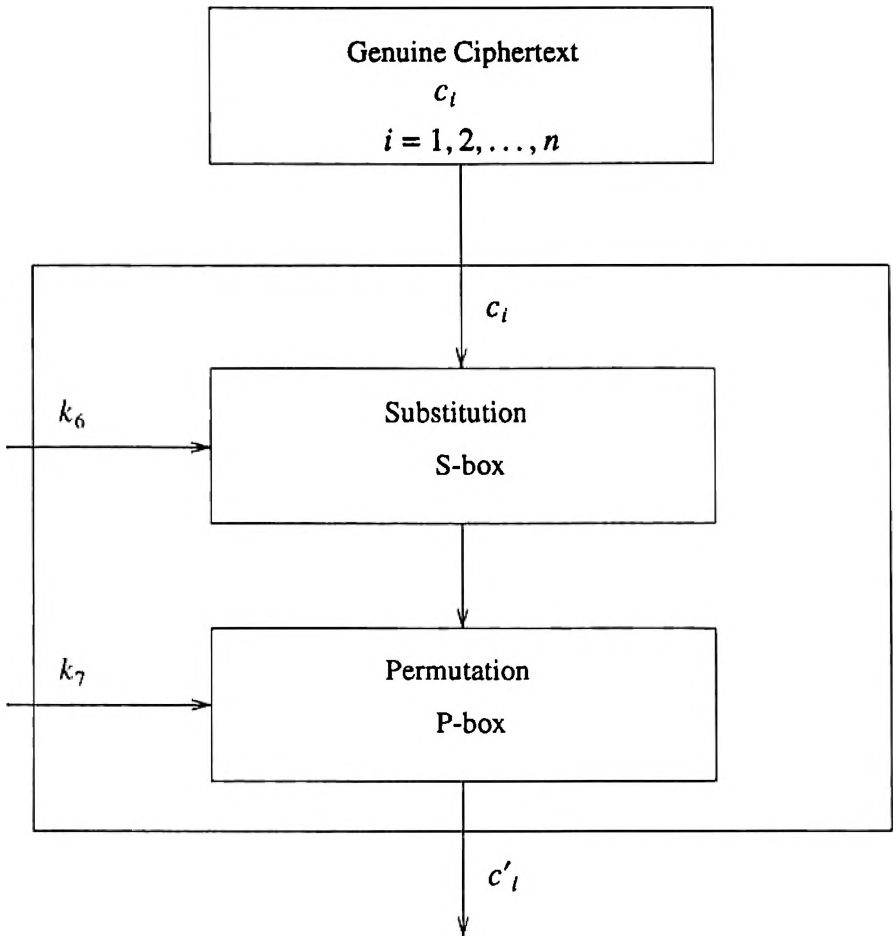

Figure 13. Substitution and permutation parts.

While B receives disguised ciphertext, he recovers genuine ciphertext through P-box first and then S-box. Then he performs the decryption algorithm D_k according to:

$$step\ 3: Q \times \bar{d}^T = [k_1, k_2, c_1, c_2, \cdots, c_n, k_3, k_4]^T,$$

$$step\ 4: P \times \bar{d}^T = [k_1, k_2, m'_1, m'_2, \cdots, m'_n, k_3, k_4]^T,$$

With the third step, B obtains the control point vector $\bar{d}$, where matrix Q is specifically defined. Through the fourth step, the first two members and the last two members of the plaintext vector are obtained by boundary conditions. Finally, by rounding off m'_1,m'_2, ... , and m'_n, B has the corresponding plaintext.

Most of the computations of the encryption and decryption algorithms involve computations of matrix inversion. Therefore, the complexity of the algorithm is $O(n^3)$, where n is the number of characters in each block.

D. ERROR ESTIMATION

The floating-point representation of a real number in computers is of finite length. Thus, there must be errors generated through our system. However, if the absolute error generated through our system is theoretically approximated to **zero**, then the error due to the processes of computation is tolerable and can be ignored.

Let $M = (m_1, m_2, \cdots, m_n)$ be the sending message, and $\bar{M} = (\bar{m}_1, \bar{m}_2, \cdots, \bar{m}_n)$ be the recovered message. It can be stated that $M \approx \bar{M}$.

While the encryption process is started, both matrix P and plaintext vector M are precisely given. The control point vector $\bar{d}$ *derived from step 1 has only machine-generated interior errors, i.e.,* $\varepsilon \approx 0$. *It results in the absolute error for each member of control point vector* $\bar{d}$, *which is less than* ε. *Again, in examining the conditions of matrixes P and Q, they are both tridiagonal matrices and the sum of each row equals one, except the 1st and the last two rows. This type of matrixes is considered to be well-behaved for matrix inversion[RcM71] and will not augment the error during the transformations. Therefore, the solution of this system is considered to be stable. It is also believed that there is no other possible error in the encipherment due to the rearrangement of ciphertext.*

Likewise, the same arguments can be applied to the decipherment. Let $G = Q^{-1}$ *and* $q = \max_i \sum_{j=1}^{n} |G_{ij}|, i = 1, 2, \ldots, n$, *then* $\bar{d}' = G \times \bar{c}$, *where the vector* $\bar{c}$ *consists of the original ciphertext value. Thus, the upper bound of the total error is equal to* $q\varepsilon$. *We have found that the value q is consistently less than 20, so* $q\varepsilon \approx 0$. *Therefore, the error quantity of* $q\varepsilon$ *can be ignored.*

E. A SIMPLE EXAMPLE

As the verification procedure is presented, the four boundary conditions together with the number of encrypted characters in each block are provided for demonstrating correctness of the spline cryptosystem. Considering an 8-character message, UM-ROLLA, we encode each character to be its corresponding ASCII code: A=65, L=76, M=77, O=79, R=82, U=85, and '-'=45. With $k_1 = -100$, $k_2 = -400$, $k_3 = 400$, $k_4 = -100$, $h = 1/9$, P and Q are defined as above. Then the encryption process solves the control point vector $\bar{d}$ in advance,

$$P \times \bar{d}^T = [-100, -400, 85, 77, 45, 82, 79, 76, 76, 65, 400, -100].$$

Then

$$Q \times \bar{d}^T = [-100, -400, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, 400, -100],$$

where

$$c_1 = -208.23162227$$

$$c_2 = 147.44977802$$

$$c_3 = 39.807510174$$

$$c_4 = 64.945181280$$

$$c_5 = 85.036764706$$

$$c_6 = 72.907759896$$

$$c_7 = 88.707195708$$

$$c_8 = 27.263457270,$$

and the control point vector is

d[1]= 332.76468122

d[2]= -760.82678505

d[3]= 310.54245900

d[4]= 28.656949069

d[5]= 36.829744728

d[6]= 94.024072019

d[7]= 79.073967197

d[8]= 63.680059193

d[9]= 122.20579603

d[10]= -96.503243310

d[11]= 653.80717721

d[12]= -118.72546553.

Of course, the original cryptogram should be scrambled before transmission. For simplicity, let us skip the rearrangement of the cryptogram, and directly express the recovered message. The decryption process solves the control point vector $\bar{d}$ as follows,

$$Q \times \bar{d}^T = [-100, -400, c_1, c_2, \cdots, c_8, 400, -100].$$

Then

$$P \times \bar{d}^T = [-100, -400, m_1, m_2, \ldots, m_8, 400, -100],$$

where

$m_1 = 85.000000000(85)$

$m_2 = 77.000000000(77)$

$m_3 = 45.000000000(45)$

$m_4 = 82.000000000(82)$

$m_5 = 79.000000000(79)$

$m_6 = 76.000000000(76)$

$m_7 = 76.000000000(76)$

$m_8 = 64.999999994(65),$

and the control point vector is

d[1]= 332.76468122

d[2]= -760.82678505

d[3]= 310.54245900

d[4]= 28.656949069

d[5]= 36.829744728

d[6]= 94.024072019

d[7]= 79.073967197

d[8]= 63.680059193

d[9]= 122.20579603

d[10]= -96.503243310

d[11]= 653.80717721

d[12]= -118.72546553.

The number in each parenthesis is the recovered ASCII code, and it shows that the message is precisely recovered.

F. THE ANALYSIS OF SECURITY

The security of this system has to be evaluated. However, our speculations lead us to believe that this is a relatively novel approach to cryptography, and that all the known attacks can be applied.

Namely, the fundamental ciphertext-only attack is evaluated in advance. The cubic spline is a continuous function within the range [0,1], and its n interpolation points represent a set of characters that are strongly related. Any change of a single interpolant will result in changes for the rest. Moreover, the frequency distribution of cryptogram based on the spline cryptosystem becomes obscure, since the same characters at different x-axis with the same set are uniformly mapped into real numbers. Obviously, a statistical attack on inspecting ciphertexts is not feasible.

In its elementary form, this cryptosystem undergoes a series of linear algebraic transformations without substitution and permutation. One may attack this system by comparing the plaintext and its corresponding ciphertext, since the subkeys k_1, k_2, k_3, and k_4 of this linear system can be easily reconstructed. Therefore, in addition to these four working subkeys, three subkeys are introduced to modify our system. These

additional subkeys result in the procedures of substitution and permutation of the original cryptogram which make the encipherment and decipherment no longer simple linear transformations.

Inspecting the substitution and permutation parts, the worst working factor in breaking the non-linear transformations, without the knowledge of k_6 and k_7, is $8 \times n!$. n is the length of each block, but it is no more than 10. Thus, this might provide a weak point of the spline cryptosystem.

Considering the previous key generation, there are 256 values for k_1. This indicates that the system could be broken by trying all the values of k_1. In order to avoid such a weakness, the modulus 256 is replaced by a certain 56-bit number. Note that the DES cryptosystem takes 56 bits as the key length. Furthermore, a careless choice of k_1 will result in the block size to be $k_5 \bmod 10 = 0$. The system cannot work in this special instance. See Appendix-invalid values of $k_1 \in [0,255]$.

G. CONCLUSION

A practical cryptosystem has been proposed, which implements under real numbers. It has been demonstrated how the spline cryptosystem uses an interpolation cubic spline as an encryption method. Cryptography has been expanded from the traditional discrete integer domain into real numbers. Recently, a random spline cryptosystem has also been designed. This is a sophisticated system, in which the security is based on its randomness.

IV. MULTI-SEGMENT SPLINES AND THE RSA SYSTEM

A. PRELIMINARY

A formal, general transformation of encipherment and decipherment of the RSA cryptosystem is presented. The spline cryptosystem constructs ciphers based on an interpolating spline function. Using spline functions, a series of discontiguous spline segments can execute the modular arithmetic of the RSA system. Recently, we[HqH92] proposed a new encrypting method called the spline cryptosystem. This system, is based on the interpolating cubic spline function[LaS86]. This system has been improved to be a random version. In order to understand the similar property between random spline cryptosystem and the RSA cryptosystem[RSA78], the RSA cryptosystem is reformulated to be a series of discontiguous spline segments. So that the encryption process and decryption process of RSA can be simulated properly.

The RSA cryptosystem has been reformulated to be multi-segment splines. Applying multi-segment splines[dBc78] one segment is found in which the value of ciphertext corresponds to an integer value of the plaintext. Furthermore, the reformulation of RSA cryptosystem can be characterized as polynomials with random offsets between ciphertext values and plaintext values. In fact, given a plaintext, we can find a spline segment defined over the x-axis, with index i. Therefore, the corresponding ciphertext is the value at the y-axis, and the ciphertext corresponds to a spline segment defined over the y-axis, with index j. We also recognize that given an arbitrary function, f, $f(i, j)$ is not a constant. This is identical to the random encryption process of random spine cryptosystem. That is, with the random spline cryptosystem, the ciphertext corresponding to a given

plaintext is generated by an arbitrary parameter $t_i \in [0.25,0.75]$, where i is the block number. This means the value of t_i could be anywhere within each subinterval.

B. RELATED TRANSFORMATION

In modern and classical encryptions, modular arithmetic is the primary computation. In the RSA cryptosystem, N is assumed to be a composite number, and $D = \{0, 1, \ldots, N-1\}$. Let $M \in D$ and $C \in D$ be the plaintext and the ciphertext respectively, e and d be the encryption key and the decryption key. Let e and d be related to each other, so that

$$E_e : C = M^e \ mod \ N \ \text{ and } \ D_d : M = C^d \ mod \ N, \tag{4.2.1}$$

where

$$ed = 1 \ mod(\phi(N)). \tag{4.2.2}$$

More specifically, the encipherment and the decipherment of the RSA system are reformulated by a series of discontiguous spline segments, multi-segment splines. In general, the encipherment could be simulated by a series of discontiguous e-th degree spline segments, $x^e - iN$, where i is the index of the segment with respect to x-axis. Similarly, the decipherment can be simulated by a series of discontiguous d-th degree spline segments, $y^d - jN$, where j is the index of the segment with respect to y-axis. These formulas consist of monotonically increasing but progressively narrower intervals $[0, N^{1/e})$, $[N^{1/e}, (2N)^{1/e})$, etc. (or $[0, N^{1/d})$, $[N^{1/d}, (2N)^{1/d})$, etc.)

In particular, let $x_i = (iN)^{1/e}$, $i = 0,1,..,N^{e-1}$, $y_j = (jN)^{1/d}$, $j = 0,1,..,N^{d-1}$, where x_i are coordinates on x-axis and y_j are coordinates on y-axis. Then we define

$$f(x) = \sum_{i=0}^{N^{e-1}} \phi_i(x)(x^e - iN), \quad g(y) = \sum_{j=0}^{N^{d-1}} \psi_j(y)(y^d - jN), \tag{4.2.3}$$

where

$$\phi_i(x) = \begin{cases} 1 & \text{if } x_i \le x < x_{i+1} \\ 0 & \textit{otherwise} \end{cases}, \tag{4.2.4}$$

and

$$\psi_j(y) = \begin{cases} 1 & \text{if } y_j \le y < y_{j+1} \\ 0 & \textit{otherwise} \end{cases}, \tag{4.2.5}$$

such that *f(x)* and *g(y)* are multi-segment splines(see Figure 14 and Figure 15).

The scenarios of the encipherment and the decipherment are as follows. To encrypt an integer value M. Assuming $M \in [x_i, x_{i+1})$ if $iN \le M^e < (i+1)N$. Then, compute $f(M) = M^e - iN$. Thus the range of values of $f(M)$ is [0, N]. The decipherment is to compute $g(C) = C^d - jN$ given the value of $C = f(M)$. That is, choose $C \in [y_j, y_{j+1})$ if $jN \le C^d < (j+1)N$. Then the range of the value of $g(C)$ is [0, N]. We demonstrate these processes via the following example: assume that $p = 3$, $q = 11$, $\phi(N) = 20$, $(e, d) = (7, 3)$, and $M = 17$.

The encryption steps are:

1. to compute the segment's index $i = \lfloor M^e/N \rfloor = \left\lfloor 17^7/33 \right\rfloor = 12434505$ such that $M \in [x_i, x_{i+1})$,
2. to compute $f(17) = 17^7 - i \times 33$

$= 410338673 - 12434505 \times 33$

$= 8.$

And the decryption steps are:

1. to compute the segment's index $j = \left\lfloor C^d/N \right\rfloor = \left\lfloor 8^3/33 \right\rfloor = 15$ such that $C \in [y_j, y_{j+1})$,

2. to compute $g(8) = 8^3 - j \times 33$

$= 512 - 15 \times 33$

$= 17.$

It has been shown that the order of above algorithm is identical to the RSA's. Clearly, the relationship between the degrees of the two series of spline segments must be concealed. Otherwise the attacker would be able to determine x, given $f(x)$.

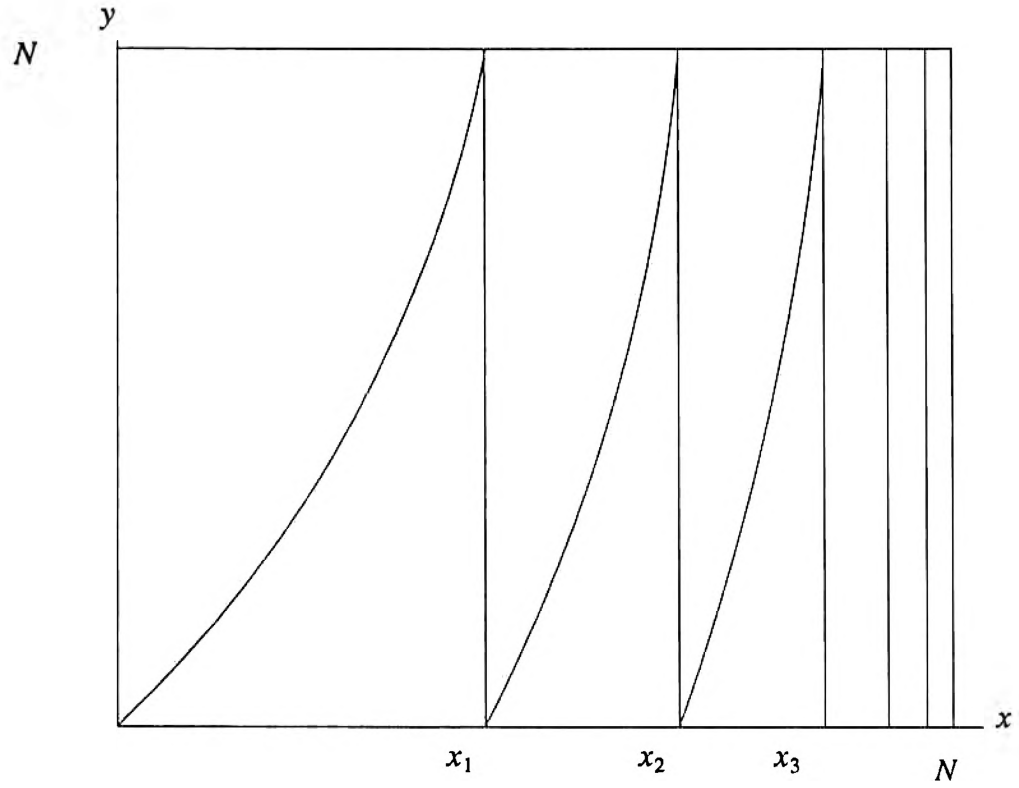

Figure 14. A series of spline segments defined over x.

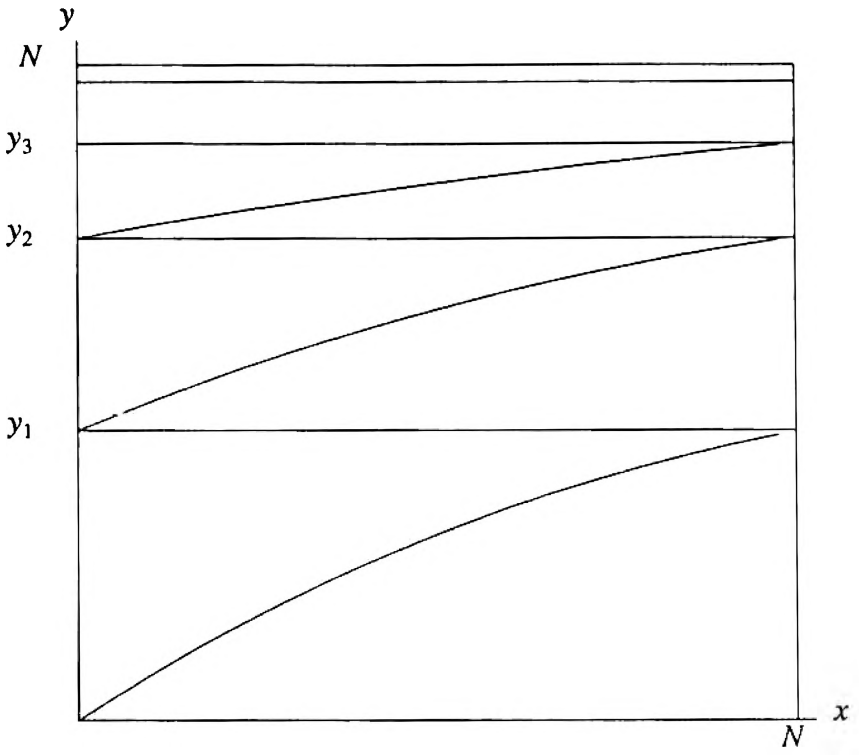

Figure 15. A series of spline segments defined over y.

Is encipherment and decipherment using multi-segment splines feasible?. The following equations answer this question:

$$f(M) = M^e - iN = M^e \ mod \ N = C = E_e \ \ by \ (4.2.1), \tag{4.2.6}$$

and

$$g(C) = C^d - jN = (M^e - iN)^d - jN = M^{ed} \ mod \ N = M = D_d \ \ by \ (4.2.1). \tag{4.2.7}$$

Generally speaking, with RSA or the exponential encipherment, $D(E(M)) = M$ or

$E(D(C)) = C$. But, with the spline segments, $g(f(x)) = x$ or $f(g(y)) = y$ is always true when $x, y \in \{0, 1, \cdots, N-1\}$. This means that only integer values of x may be encoded and decoded. Whereas such formulas do not work over real numbers. This is because there may not be any intersection between $f(x)$ and $g(y)$ for arbitrary pairs x and y.

C. DISCUSSION

A geometric interpretation of encryption and authentication is provided. The RSA encryption was defined in the previous section. From a geometric perspective, there are a set of curves along the x-axis and a set of curves along the y-axis. It follows that $f(x) = y$ and $g(y) = x$. However, both x and y are integers. Viewing that at all intersections between $f(x)$ and $g(y)$(see Figure 14. and Figure 15.), there are $N^{e-1} \times N^{d-1}$ such intersections, but only N integer intersections. Clearly, a certain spline segment expressed in x will intersect some spline segments expressed in y at integer coordinates, there is possible no integer intersection.

So these integer intersections indicate a one-to-one mapping. That is, the solution must be unique in x and y. Notice that the offset between x and y is random. This means it enables the scheme to withstand any chosen-plaintext attack. Indeed, the spline cryptosystem[HqH92] can be characterized as a transposed polynomial system. In that paper, the offset between t and t' is fixed, this is in contrast to the multi-segment spline approach.

D. CONCLUSION

The significance of multi-segment splines was demonstrated for the execution behavior of the RSA system. That is, under the integer domains of $f(x)$ and $g(y)$, the encipherment and the decipherment can be executed correctly as the RSA's. Furthermore, it was shown that reformulation of both the encryption and decryption processes are necessary conditions for inspecting the similar characteristic of random spline cryptosystem. This implies that random spline cryptosystem functions properly.

Given very large e, d, and N, such a reformulation becomes impractical. Because M^e of $f(M)$ and C^d of $g(C)$ generate extremely large results that cannot be represented correctly in computers. A bridge has been established between multi-segment splines and the RSA system. This may provide a new topic for cryptographic communities.

V. THE RANDOM SPLINE CRYPTOSYSTEM

A. PRELIMINARY

In this chapter, a random encryption scheme named the *random spline cryptosystem* is presented. The mechanism is an advanced structure of our earlier work, spline cryptosystem. Its mathematical indeterminacy on computing keys with interpolants no more than four and numerical sensitivity to the offset t_i increases its utility. From the theoretical point of view, the notion of interpolating cubic splines is easy to understand and implement. From practical viewpoint, this system is capable of handling real numbers and it can be enforced by simple operations.

The preliminary design of spline cryptosystem was disclosed in the ACM SIGSAC Review in winter of 1992[HqH92]. An advanced mechanism, called the random spline cryptosystem, having mathematical indeterminacy and numerical sensitivity, is presented. For any given interpolant points, representing the plaintext, encipherment consists of evaluating the spline at the midpoint of each segment. This produces a new set of interpolant points, through which a new cubic spline passes. Conversely, given the ciphertext, the previously defined cubic spline can be reproduced precisely. That is the amazing behavior of the cubic spline.

With the earlier structure of the spline cryptosystem, its security solely relied on scrambling of the ciphertext by substitution and permutation. Viewing Biham and Shamirs'[BiS90] paper, they developed a new type of cryptanalytic attack. This attack can be applied to any variation of DES-like substitution and permutation ciphers.

Conceivably, the plaintext may still be recovered with the application of the spline cryptosystem. With this speculation, an advanced technique to enhance the security of the system is in order. The substitution and permutation steps are eliminated and a variable parameter t_i is provided which is dominated by an initial condition. More defining information is involved than just the intermediate point of each subinterval. In other words, each block in the sequence is encrypted and decrypted by the fixed matrix P together with a randomized matrix Q_i, where Q_i is derived by the parameter t_i, $i = 1, 2, \cdots$. The specified initial condition and the boundary conditions are considered to be the secret keys, together with the block i. Both the encryption process and the decryption process are achieved simply by multiplication and addition operations, which are simple calculations.

It is found that the ciphertext is sensitive to the offset t_i and boundary conditions. If the precise seed and boundary conditions are not known, then the system should be difficult to attack. The interpolant $n = 3$ is the safety margin and boundary conditions can be systematically constructed. The same boundary conditions never are used more than once within each block. Another interesting characteristic of the system is that the ciphertexts are uniformly distributed in real numbers.

The remainder of this chapter is organized as follows. The second section reviews the theoretical essence of *cubic splines*. In the third section, encipherment and key management of the random spline system are discussed. We also analyze the security and sensitivity of the randomized system in the fourth section. Finally, future research with this system is mentioned.

B. THEORETICAL FOUNDATION

The *spline* was heavily used in the past in the shipbuilding and aircraft industries. The spline bends in such a way that its internal energy due to bending is minimal, consistent with the interpolation constraints imposed on it. If the formula of the interpolating spline is viewed as the function $S(x)$, then $S(x)$ can be constructed by interpolating at the knots $a = x_0 < x_1 < \cdots < x_{n+1} = b$, so that the bending energy is minimal.

The *cubic spline*[AdR76][dBc78][Scl81] is well-suited as the candidate for constructing the spline cryptosystem, because the higher degree polynomials are characterized by their oscillatory behavior and large swings. In order to construct the cubic function $S(x)$, a basis is required for $S(x)$, that is, the *cubic B-splines*. For simplicity, equally spaced knots are employed. First, the cubic B-spline is defined for all real numbers. Then the normalized blending function $B(t)$ is given for the cubic B-spline, represented in equation (5.2.1) and shown in Figure 12.

$$B(t) = \begin{cases} -|t|^3/6 + |t|^2 - 2|t| + 4/3 & \text{if} \quad 1 \le |t| \le 2 \\ |t|^3/2 - |t|^2 + 2/3 & \text{if} \quad 0 \le |t| \le 1. \\ 0 & \textit{otherwise} \end{cases} \tag{5.2.1}$$

Considering the interval $x_{-1} < x_0 < \cdots < x_{n+1} < x_{n+2}$, there are a few more cubic B-splines whose support is not entirely in $[x_0, x_{n+1}]$ and either start at the knot x_{-1} or terminate at the knot x_{n+2}. Thus, the complete set of cubic B-splines restricted to $[x_0, x_{n+1}]$ is the following translate of $B(x)$:

$$B_j(x) = B((x - jh)/h), \; j = -1, 0, \cdots, n+2, \tag{5.2.2}$$

where h is the length of the subinterval $[x_i, x_{i+1}]$, $i = 0, 1, \cdots, n$. Therefore, any cubic

spline $S(x)$ with the knots $x_0, \cdots, x_{n+1}$ can be written in the form

$$S(x) = \sum_{j=-1}^{n+2} d_j \times B_j(x), (0 \le x \le 1), \tag{5.2.3}$$

where $x_j = j \times h$, and d_j is said to be the *control point*.

The resulting cubic spline $S(x)$ is a continuous function within the interval [0, 1], with continuous first and second derivatives. With respect to the interpolating cubic spline, m_j $(1 \le j \le n)$, the plaintext, represents the interpolant at each knot, x_j. Four boundary conditions, viewed as the secret keys of our system, are given as below

$$S'(0) = k_1, S(0) = k_2, S(1) = k_3, S'(1) = k_4,$$

where k_1 and k_4 are the slopes at 0 and 1, k_2 and k_3 are the positions at 0 and 1. So that equation (5.2.3) can be reformulated in the matrix form

$$P \times [d_{-1}, \cdots, d_{n+2}]^T = [k_1, k_2, m_1, \cdots, m_n, k_3, k_4]^T, \tag{5.2.4}$$

where P is a $(n+4) \times (n+4)$ matrix. For example, when $n = 3$:

$$P = \begin{bmatrix} \frac{-1}{2h} & 0 & \frac{1}{2h} & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 & 0 & \frac{-1}{2h} & 0 & \frac{1}{2h} \end{bmatrix}.$$

Recall that, in our paper[HqH92], a new set of interpolant points c_j was calculated corresponding to the intermediate points ξ_j, given a set of interpolant points m_j corresponding to x_j(see Figure 16). Furthermore, in order to increase the utility and security of the spline system, it enforces the matrix Q_i randomly by using the formula $\xi_j^{(i)} = (1-t_i)\times x_{j-1} + t_i \times x_j$, where $0 < t_i < 1$, $i = 1, 2, \cdots$, $j = 1, 2, \cdots, n$. $\xi_j^{(i)}$ could be anywhere within the interval $[x_{j-1}, x_j]$ and $c_j = S(\xi_j^{(i)})$, instead of fixed at the middle point of $[x_{j-1}, x_j]$. Meanwhile, t_i should be generated such that $\lim_{i\to\infty} t_i = 0.5$. For instance, $t_i = 0.5\times(1+(-1)^{i+SEED}/(i+SEED+1))$, and $SEED$ must be integer. Such a formula should be dominated by an initial condition and the output be oscillated about *0.5*. In other words, the value of t_i jumps back and forth around the optimal value *0.5*, but it never comes close to 0 or 1. This means $t = 0.5$ will produce a new set of interpolant points with the middle offset t from the original ones.

Indeed, the system is executed by multiplying a random matrix Q_i with the control points $d_{-1}, d_0, \cdots, d_{n+1}, d_{n+2}$

$$Q_i \times [d_{-1}, \ldots, d_{n+2}]^T = [k_1, k_2, c_1, \cdots, c_n, k_3, k_4]^T, \tag{5.2.5}$$

where the matrix Q_i is formulated as a $(n+4)\times(n+4)$ matrix. For example, when $n = 3$:

$$Q_l = \begin{bmatrix} \frac{-1}{2h} & 0 & \frac{1}{2h} & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 & 0 \\ \omega_1 & \omega_2 & \omega_3 & \omega_4 & 0 & 0 & 0 \\ 0 & \omega_1 & \omega_2 & \omega_3 & \omega_4 & 0 & 0 \\ 0 & 0 & \omega_1 & \omega_2 & \omega_3 & \omega_4 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 & 0 & \frac{-1}{2h} & 0 & \frac{1}{2h} \end{bmatrix},$$

where

$$\omega_1(t_l) = -(1 + t_l)^3/6 + (1 + t_l)^2 - 2(1 + t_l) + 4/3$$

$$\omega_2(t_l) = t_l^3/2 - t_l^2 + 2/3$$

$$\omega_3(t_l) = (1 - t_l)^3/2 - (1 - t_l)^2 + 2/3$$

$$\omega_4(t_l) = -(2 - t_l)^3/6 + (2 - t_l)^2 - 2(2 - t_l) + 4/3 .$$

Combining equations (5.2.4) and (5.2.5), the control points d_j's are eliminated so that it is possible to compute the new interpolant points directly from the original ones.

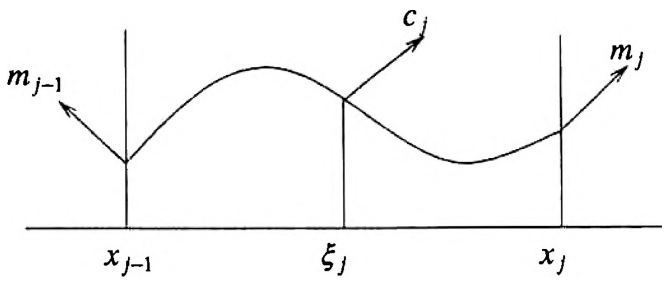

Figure 16.The ciphertext c_j with respect to the plaintext m_j.

C. ENCIPHERMENT AND KEY MANAGEMENT

In this section the encryption process and the decryption process are introduced. Meanwhile, a simple example is presented and a concrete architecture is illustrated. Finally, we systematically construct the keys in order to manage the system easily.

1. Encipherment. Notice that the encryption process E_k and the decryption process D_k of random spline cryptosystem are straightforward. The plaintext is viewed as 16-bit integer(two characters), but the ciphertext is viewed as 32-bit real number. For each block i

$$E_k : Q_i \times P^{-1} \times [k_1, k_2, m_1, m_2, \cdots, m_n, k_3, k_4]^T = [k_1, k_2, c_1, c_2, \cdots, c_n, k_3, k_4]^T,$$

and

$$D_k : P \times Q_i^{-1} \times [k_1, k_2, c_1, c_2, \cdots, c_n, k_3, k_4]^T = [k_1, k_2, m_1, m_2, \cdots, m_n, k_3, k_4]^T,$$

where Q_i is dominated by t_i, $i = 1, 2, \cdots$, and $Q_i \neq Q_j$ for $i \neq j$ and T is the transposition operator. Then by rounding plaintexts m_i in terms of the decryption, the plaintext can be recovered. As a matter of fact,

$$D_k(E_k(M)) = P \times Q_i^{-1} \times Q_i \times P^{-1} \times [k_1, k_2, m_1, \cdots, m_n, k_3, k_4] = M' \approx M,$$

but these two vectors M' and M are identical by rounding each element of M'.

An example:Let $n = 4$, $h = 0.2$, $SEED = 10$, $t_1 = 0.458$, $k_1 = -100$, $k_2 = -400$, $k_3 = 400$, and $k_4 = -100$. Then the matrixes P and Q_1 are

$$P = \begin{bmatrix} -2.5 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 & 0 & 0 & -2.5 & 0 & 2.5 \end{bmatrix}$$

and

$$Q_1 = \begin{bmatrix} -2.5 & 0 & 2.5 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 \\ \omega_1 & \omega_2 & \omega_3 & \omega_4 & 0 & 0 & 0 & 0 \\ 0 & \omega_1 & \omega_2 & \omega_3 & \omega_4 & 0 & 0 & 0 \\ 0 & 0 & \omega_1 & \omega_2 & \omega_3 & \omega_4 & 0 & 0 \\ 0 & 0 & 0 & \omega_1 & \omega_2 & \omega_3 & \omega_4 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 & 0 & 0 & -2.5 & 0 & 2.5 \end{bmatrix},$$

where w_1 = 2.6487750773E-02, w_2 = 5.0473813658E-01, w_3 = 4.5272714119E-01, and w_4 = 1.6046971452E-02. The plaintexts are **TEXT**. Then we execute the encryption process as follows:

$$Q_1 \times P^{-1} \times [-100, -400, 84, 69, 88, 84, 400, -100]^T = [ciphernumbers]^T$$

where ciphernumbers are same as those in Table 2. The decryption process is executed as follows:

$$P \times Q_1^{-1} \times [-100, -400, ciphernumbers, 400, -100]^T = [returnplain]^T,$$

by rounding each member of returnplain. We obtain the plaintexts just the same as the original ones.

More specifically, let us turn to the problem of computing the encryption process, E_k, or the decryption process, D_k. As described before, there is an inverse matrix P^{-1} involving in E_k, so that this computation suffers in practice from numerical instability. That is, the round-off errors tend to accumulate when floating-point number representations are used instead of real numbers. Fortunately, there is an approach, called the *LUP decomposition*, which is numerically stable and has the further advantage of $n^3/3$ operations instead of n^3 operations.

This decomposition algorithm could be presented within any numerical analysis references. The idea behind the *LUP decomposition* is to find three $n \times n$ matrices, L, U, and P for an arbitrary nonsingular matrix A such that $LU = PA$,

where

a. L is a unit lower-triangular matrix,

b. U is an upper-triangular matrix,

c. P is a permutation matrix.

Recall that the encipherment

$$Q_i \times P^{-1} \times [k_1, k_2, m_1, m_2, .., m_n, k_3, k_4]^T = [k_1, k_2, c_1, c_2, .., c_n, k_3, k_4]^T$$

is composed of two steps, equation (5.2.4) and equation (5.2.5). Conceivably, equation (5.2.4) involves the computation of the matrix inversion first. However, it can be replaced by the efficient *LUP decomposition*. On the other hand, decipherment produces the same replacement. After the decomposition, the process is accelerated by forward substitution and backward substitution.

2. Architecture. More concisely, the following illustration(Figure 17.) is a concrete architecture of random spline cryptosystem. It is composed of the encryption process, the random matrix generator, and the key generator, but does not specify the decryption part for simplicity.

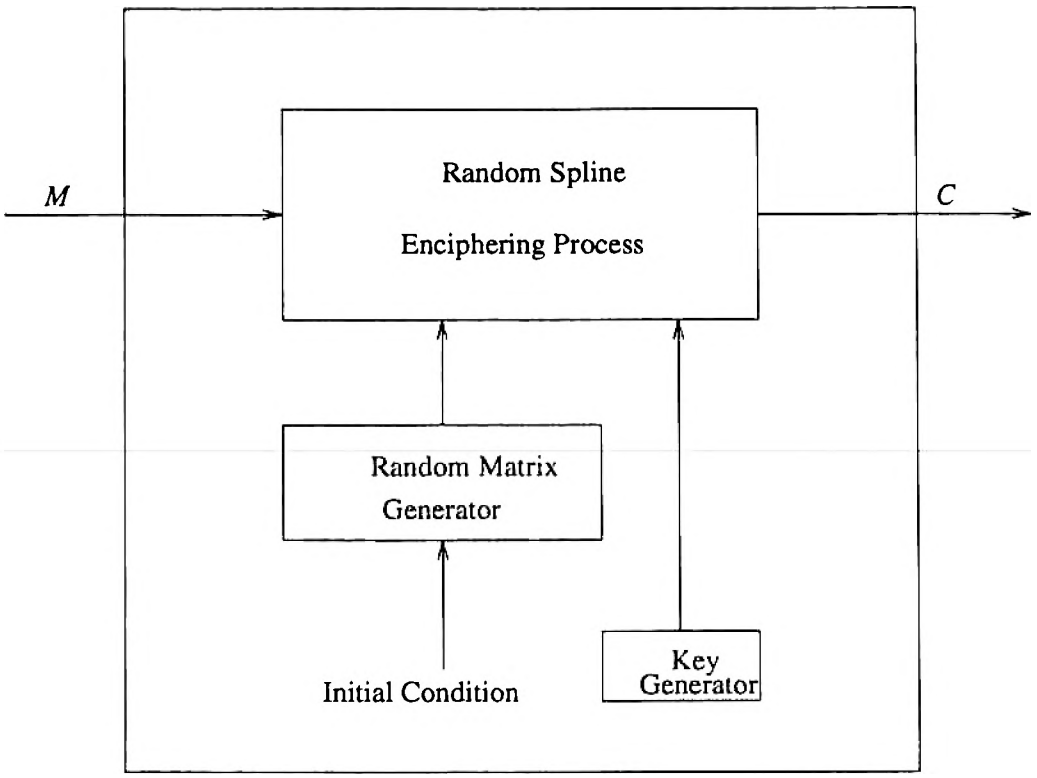

Figure 17. Architecture of random spline cryptosystem.

3. Key Management. Understandably, the smoothness of the cubic spline makes the system vulnerable, since the ciphertext does not vary much with respect to the corresponding plaintext. The spline curve is dominated by four boundary conditions, especially the starting point and the ending point of a spline curve. The curve should be as vibrant as we wish, in order to obtain further distance between the ciphertext and the corresponding plaintext. For example, in Figure 18, the two curves are generated by different boundary conditions. The upper one is generated by $k_1 = -100$, $k_2 = -400$, $k_3 = 400$, $k_4 = -100$,

and $SEED = 9$ with block 2. The lower one is generated by $k_1 = -100$, $k_2 = -40$, $k_3 = 40$, $k_4 = -100$, and $SEED = 9$ with block 2. The upper curve demonstrates better feature than the lower one. Table 1 illustrates the corresponding outputs with the different boundary conditions. The strategies for choosing keys are given below:

a. The magnitudes of k_2 and k_3 both are ≥ 400, but the signs of k_2 and k_3 are reversed.

b. Both k_2 *mod* $10 = 0$ and k_3 *mod* $10 = 0$.

c. The increasing or decreasing magnitude of k_2 or k_3 is *10* if necessary.

d. The reasonable bounds are -128 to 127 for k_1 and k_4.

e. The initial condition *SEED* could be an 8-bit integer.

Summing up, the key size may be described as:

$$k_1 + k_2 + k_3 + k_4 + SEED = 56,$$

since k_1 and k_4 both are 8-bit integers, k_2 and k_3 both are 16-bit integers and *SEED* is an 8-bit integer. However, from the security point of view, the size of keys are too small. This system could be attacked by a brute-force search. In order to enhance its utility, the size of keys can be adaptable. For instance, k_2 and k_3 both are 32-bit integers, and *SEED* is a 64-bit integer. Thus, the total size of keys can be 144 bits.

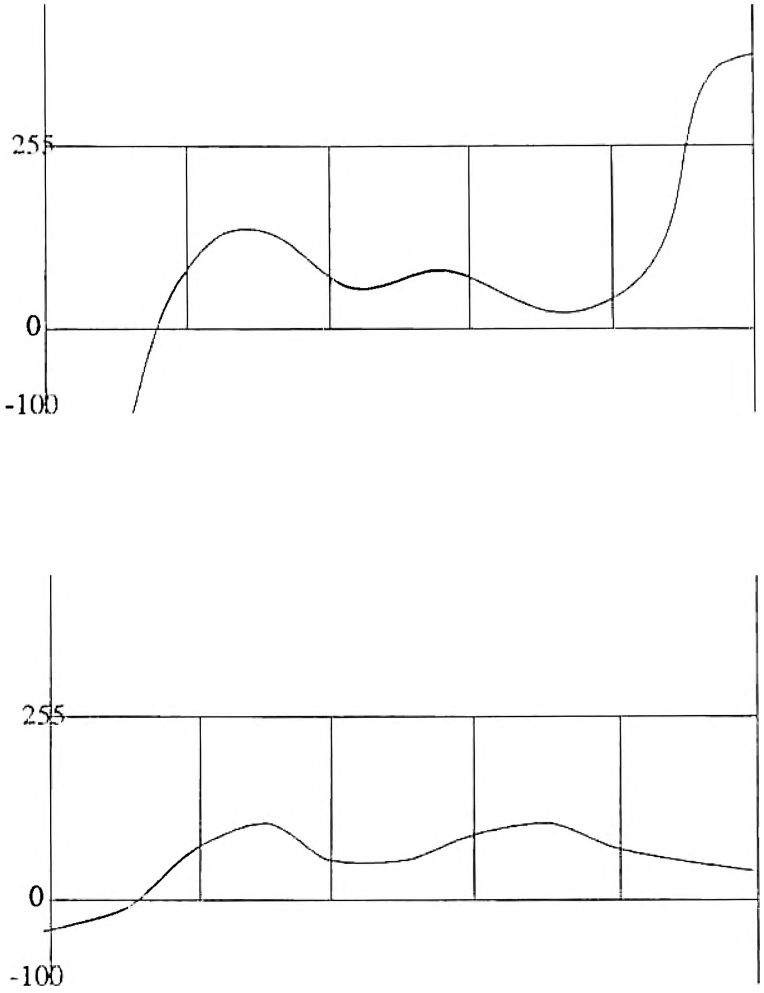

Figure 18. Two curves generated by different boundary conditions.

In the random spline cryptosystem, the well-conditioned property of both P and Q_i[RcM71] diminishs the error accumulated through computation, so that the message is recovered completely. Note that, the matrix Q_i of each block is completely decided by the corresponding t_i, but the matrix P is identical everywhere within the system.

Therefore, it is safe to put the matrix P in a public file, so that each user of the spline cryptosystem can access it. In general, there is no complicated computation involved in encipherment and decipherment, thus it is easy to implement the model.

D. INFORMATION RATE, SENSITIVITY, AND SECURITY

In this section, we analyze the information rate of encryption process, and the sensitivity of parameters related to security. We also show how the mathematical indeterminacy of the system helps to improve security.

1. Information Rate. Note that the n bits of plaintext are encrypted as m bits of ciphertext approximately. Then the information rate R of the encipherment is n/m. Viewing the system performance, the plaintext is represented as a 16-bit integer number, but the ciphertext is represented as a 32-bit floating-point representation. So the information rate is 0.5(or data expansion is 2). In order to decrease the data expansion, we allow the plaintext to have 4 characters producing a 32-bit integer which still a 32-bit ciphertext.

2. Sensitivity of Parameters. In general, the ciphertext is extremely sensitive to variations of the corresponding value t_i(see Table 2). This enhances the secure nature of the system. The initial condition *SEED* is also a secret key coupled with the system. Therefore there are five secret keys within the random spline system, k_1, k_2, k_3, k_4, and *SEED*. Table 2 is organized as follows:

a. The first cluster is the original output with the [HqH92] proposed specification. That is, the plaintext, TEXT, together with four boundary conditions(k_1 = -100, k_2 = -400, k_3 = 400, k_4 = -100) and the parameter, t = 0.5. The ciphertext is fixed at the intermediate point of each subinterval.

b. The second, third, and fourth clusters have been collected by executing the system coupled with SEED = 9. We also consider block 1, block 2, and block 3 sequently.

If we define the float-point representation of ciphertext to be double precision, then the mantissa will be a 16 digits decimal number. Plaintext is allowed to be 16 digits(8 characters) number, in order to recover the plaintext correctly. This makes the system very sensitive to boundary conditions. Minor changes of boundary conditions result in quite different ciphertext. Thus, boundary conditions are defined over arbitary real numbers, and the previous key scheduler becomes unexisted.

Regarding the sensitivity of the offset t_i, Figure 19 shows the distribution of the relative errors between the ciphertexts and the plaintexts. For which, t_i is divided into 20 intervals, with fixed boundary conditions. When t_i is close to *1*, the relative error is close to *0*. This means that the ciphertext is identical to the plaintext and attackers can recognize the pattern easily. Whereas, if t_i is close to *0*, the ciphertext only shifts to the left for one subinterval. This means attackers would observe partial information from ciphertexts. This is not allowed in cryptography. However, when t_i is around *0.5*. the relative error becomes either larger or smaller. Such a phenomenon should confuse the attackers. Therefore, we assume that t_i should be generated between *0.25* and *0.75*. This conclusion resulted from our proposed equation.

3. Security. Since, this is a relatively novel approach to cryptography, all possible breaches need to be assessed. Namely, the basic *ciphertext-only attack* is considered first. Because the cubic spline is a continuous function, its interpolating points are strongly related to each other. This means the change of an interpolating point will result in the change for the rest. Moreover, we consider whether attacks would reconstruct the cubic spline function by just viewing the ciphertexts. If they could, then the plaintexts would be obtained by computing the function's values at each corresponding x-coordinate. From interpolating viewpoint, a cubic polynomial $f(x)$ can be constructed only if there are at least four points on this curve, such as (ξ_1, c_1), (ξ_2, c_2), (ξ_3, c_3), and (ξ_4, c_4). While attackers could easily intercept ciphertexts c_1, c_2, and c_3, they still do not know exact boundary conditions. One cannot rebuild the cubic spline not knowing the exact boundary conditions. Secondly, if boundary conditions are changed slightly, the ciphertexts become obviously different. Summing up, a ciphertext-only attack can be ruled out.

On the other hand, we analyze the possibility of reconstructing the secret-key by comparing the ciphertext and the corresponding plaintext. Because of the similar process for encrypting each block, only one block of message will be analyzed here.

Let $U = Q_i \times P^{-1} = (u_{mn}^{(i)})_{(n+4)\times(n+4)}$, where each $u_{mn}^{(i)}$ is a function of t_i of each block i. Consider

$$V_i = \begin{bmatrix} u_{33} & u_{34} & \cdot & \cdot & u_{3,n+2} \\ u_{43} & u_{44} & \cdot & \cdot & u_{4,n+2} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ u_{n+2,3} & u_{n+2,4} & \cdot & \cdot & u_{n+2,n+2} \end{bmatrix}$$

and

$$W_i = \begin{bmatrix} u_{31} & u_{32} & u_{3,n+3} & u_{3,n+4} \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ u_{n+2,1} & u_{n+2,2} & u_{n+2,n+3} & u_{n+2,n+4} \end{bmatrix},$$

where V_i is a $n \times n$ matrix and W_i is a $n \times 4$ matrix. Then encipherment is formed as follows:

$$[c_1, \cdots, c_n]^T = V_i \times [m_1, \cdots, m_n]^T + W_i \times [k_1, k_2, k_3, k_4]^T. \qquad (5.4.3.1)$$

As long as $1 \le n < 4$, equation (5.4.3.1) shows the mathematical indeterminacy of the computing keys. The unknown parameters are always *greater than* the equation numbers. Such solution spaces are infinite. Thus, a brute-force attack should not be possible. The mathematical indeterminacy of deciding keys forces attackers to give up the technique of trying all possible solutions. Therefore, the reconstruction of the secret key by comparing the ciphertext with the corresponding plaintext can be thwarted.

The relative error between the plaintext and the ciphertext is analyzed in order to recognize that the spline curve is not only dominated by boundary conditions but is also dominated by the interpolant numbers n. Note that the relative error is defined as $\sum_{i=1}^{n} | c_i - m_i | / m_i$, where c_i is the ciphertext and m_i is the respective plaintext. It was found that $n = 3$ is a good candidate to construct the spline curve(see Table 3). Because, the constructed curve is as vibrant as we wish. The interpolant numbers n was chosen by considering the maximal relative error.

In addition, another probable weakness of the random spline cryptosystem is that if t_i was generated as in our original proposal, that is, $\lim_{i \to \infty} t_i = 0.5$. When t_i comes very close to 0.5 for some block i, Q_i is just the same as the previously defined matrix Q[HqH92], which was fixed in all encryption and decryption processes. This makes it easier to break the system by using the fixed matrixes Q and P to recover plaintexts with block number j, $j \geq i$. Based on this observation, we replace the formula whose output t_i is oscillated about 0.5 by a random number generator whose output $y_{i+1} = f(y_i)$ is between 0 and 1, where $t_i = y_i/2 + 0.25$ for $i = 1, 2, \cdots$ and y_0 is the *SEED*. This reformulation makes $t_i \in [0.25, 0.75]$.

Furthermore, if the *SEED* is compromised, then this cryptosystem is insecure. It can be protected by letting users select its size based on the users' security requirement. Meanwhile, four boundary conditions can be solved by a set of known plaintext and ciphertext pairs easily. Assume that attackers obtain four pairs of plaintext and ciphertext, $(\bar{M}_i, \bar{C}_i)$, $(\bar{M}_j, \bar{C}_j)$, $(\bar{M}_k, \bar{C}_k)$, and $(\bar{M}_l, \bar{C}_l)$, where $\bar{M}$ and $\bar{C}$ are plaintext vector and ciphertext vector respectively. Then attackers can make use of any two of four pairs to derive the interdependence of four boundary conditions. So that four boundary conditions

can be obtained by repeating the process. To avoid this possibility the same four boundary conditions should be never used more than once within each block. The unknown numbers are always greater than the equation numbers, that is, $4 \times i > 3 \times i$ where i is the encrypted block number and $i = 1, 2, \cdots$.

E. CONCLUSION

It has been found that the interpolating cubic polynomial, especially the cubic spline, can be employed as an encryption method. The random spline cryptosystem has the additional features of the mathematical indeterminacy of computing keys and sensitivity of boundary conditions and offset t_i, making it more difficult to break. The possible weakness of random spline cryptosystem has been investigated. It is also suggested that readers might use more sophisticated attacks, such as a chosen-plaintext attack, or might try to find possible trapdoors in this system.

The random spline cryptosystem also can be used as a message authentication system. The receiver checks whether the derived boundary conditions are just the same as the committed ones. If they are different, then the receiver will know that the ciphertext has been altered. Notice that the random spline cryptosystem is a totally different mechanism from existing ones[Der82][DiH76][RSA78]. Conceivably, it is not intended to replace all other methods, but rather as an alternative way to enlarge cryptographic research. In this regard, the random spline cryptosystem offers us a promising answer. Hopefully, cryptologists will use this approach to penetrate beyond the traditional theory of numbers.

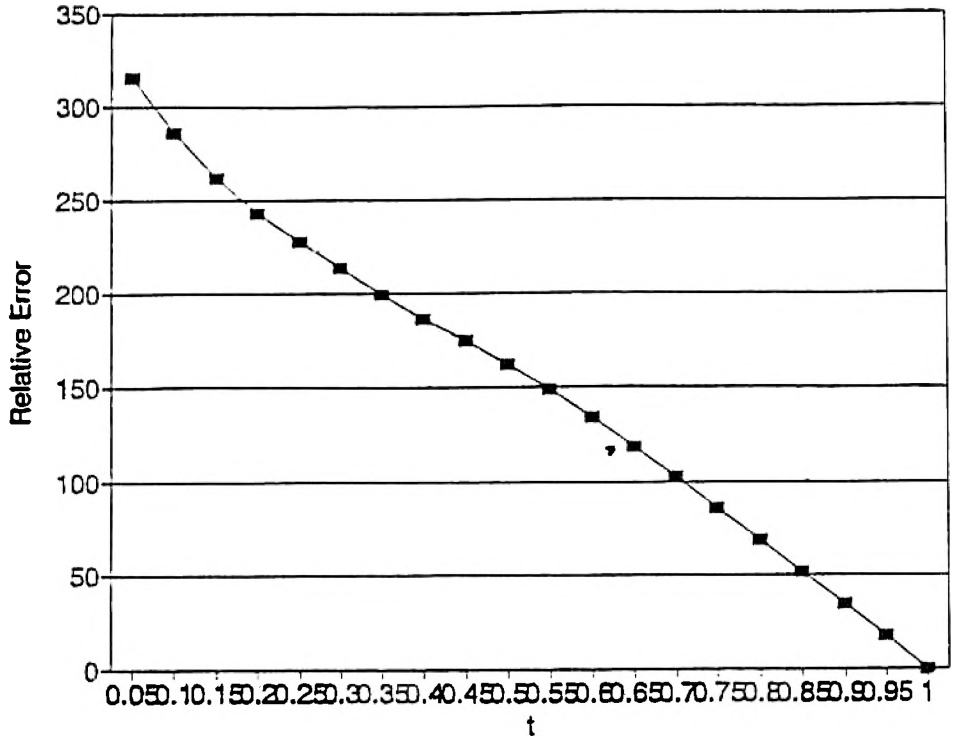

Figure 19. The sensitivity of ciphertext in terms of the offset t.

Table I. Demonstration ciphernumbers resulting from different boundaries conditions.

SEED:9, i:2, t:0.458

key:-1.0000000000E+02 -4.0000000000E+02 4.0000000000E+02 -1.0000000000E+02

	ciphernumber	plainnumber	returnplainnumber	
T	-2.8515978175E+02	8.4000000000E+01	8.4000000002E+01	T
E	1.4032802497E+02	6.9000000000E+01	6.8999999998E+01	E
X	4.4771365312E+01	8.8000000000E+01	8.8000000003E+01	X
T	6.6112338435E+01	8.4000000000E+01	8.3999999994E+01	T

SEED:9, i:2, t:0.458

key:-1.0000000000E+02 -4.0000000000E+01 4.0000000000E+01 -1.0000000000E+02

	ciphernumber	plainnumber	returnplainnumber	
T	-2.3271075614E+01	8.4000000000E+01	8.3999999999E+01	T
E	8.6606807868E+01	6.9000000000E+01	6.9000000002E+01	E
X	5.4981069259E+01	8.8000000000E+01	8.7999999997E+01	X
T	7.8994739749E+01	8.4000000000E+01	8.4000000009E+01	T

Table II. The same plainnumbers enciphered via increased block indexes.

t:0.5

key:-1.0000000000E+02 -4.0000000000E+02 4.0000000000E+02 -1.0000000000E+02

	ciphernumber	plainnumber	returnplainnumber	
T	-2.7134483911E+02	8.4000000000E+01	8.4000000000E+01	T
E	1.5711286618E+02	6.9000000000E+01	6.9000000000E+01	E
X	5.8471886807E+01	8.8000000000E+01	8.7999999999E+01	X
T	6.8647971271E+01	8.4000000000E+01	8.4000000002E+01	T

SEED:9, i:1, t:0.545

key: 1.0000000000E+02 4.0000000000E+02 4.0000000000E+02 -1.0000000000E+02

	ciphernumber	plainnumber	returnplainnumber	
T	-2.0875135794E+02	8.4000000000E+01	8.4000000001E+01	T
E	1.3091914199E+02	6.9000000000E+01	6.8999999999E+01	E
X	5.9757735128E+01	8.8000000000E+01	8.8000000001E+01	X
T	5.6105514795E+01	8.4000000000E+01	8.3999999999E+01	T

SEED:9, i:2, t:0.458

key:-1.0000000000E+02 -4.0000000000E+02 4.0000000000E+02 -1.0000000000E+02

	ciphernumber	plainnumber	returnplainnumber	
T	-2.8515978175E+02	8.4000000000E+01	8.4000000005E+01	T
E	1.4032802497E+02	6.9000000000E+01	6.8999999990E+01	E
X	4.4771365312E+01	8.8000000000E+01	8.8000000021E+01	X
T	6.6112338435E+01	8.4000000000E+01	8.3999999942E+01	T

SEED:9, i:3, t:0.538

key:-1.0000000000E+02 -4.0000000000E+02 4.0000000000E+02 -1.0000000000E+02

	ciphernumber	plainnumber	returnplainnumber	
T	-2.1471779046E+02	8.4000000000E+01	8.4000000000E+01	T
E	1.3178317323E+02	6.9000000000E+01	6.9000000001E+01	E
X	5.8669303291E+01	8.8000000000E+01	8.8000000000E+01	X
T	5.6765376014E+01	8.4000000000E+01	8.4000000001E+01	T

Table III. Demonstration of the optimal interpolant numbers.

SEED:10, i:1, t:0.458, n:1

key:-1.0000000000E+02 -4.0000000000E+02 4.0000000000E+02 -1.0000000000E+02

	ciphernumber	plainnumber	returnplainnumber	
A	-2.9499674479E+02	6.5000000000E+01	6.5000000004E+01	A

Relative Error = 5.5384615

SEED:10, i:1, t:0.458, n:2

key:-1.0000000000E+02 -4.0000000000E+02 4.0000000000E+02 -1.0000000000E+02

	ciphernumber	plainnumber	returnplainnumber	
A	-2.8942334587E+02	6.5000000000E+01	6.5000000004E+01	A
B	1.1193991609E+02	6.6000000000E+01	6.5999999987E+01	B

Relative Error = 6.1487261

SEED:10, i:1, t:0.458, n:3

key:-1.0000000000E+02 -4.0000000000E+02 4.0000000000E+02 -1.0000000000E+02

	ciphernumber	plainnumber	returnplainnumber	
A	-2.8853993572E+02	6.5000000000E+01	6.4999999999E+01	A
B	1.3095418940E+02	6.6000000000E+01	6.6000000003E+01	B
C	2.0796384136E+01	6.7000000000E+01	6.6999999991E+01	C

Relative Error = 7.1128363

SEED:10, i:1, t:0.458, n:4

key:-1.0000000000E+02 -4.0000000000E+02 4.0000000000E+02 -1.0000000000E+02

	ciphernumber	plainnumber	returnplainnumber	
A	-2.8741208861E+02	6.5000000000E+01	6.5000000001E+01	A
B	1.2567214878E+02	6.6000000000E+01	6.5999999998E+01	B
C	4.0796699520E+01	6.7000000000E+01	6.7000000004E+01	C
D	4.6002598284E+01	6.8000000000E+01	6.7999999989E+01	D

Relative Error = 7.0404329

VI. THE CHAOTIC-MAP PUBLIC-KEY CRYPTOSYSTEM

A. PRELIMINARY

Now we present a chaotic public-key cryptosystem employing a one-dimensional difference equation as well as a quadratic difference equation. This system also makes use of the El Gamal's scheme to accomplish the encryption process. The trapdoor was built by allowing the designer to know the iteration times of a certain difference equation.

Chaotic behavior has been studied in many fields. It is desired to inject chaos into cryptography, so that the designated system becomes dynamic and non-linear. Recently, a chaotic-map secret-key cryptosystem[HaN91] has been introduced. The cryptosystem employs a one-dimensional iterated map as the encryption transformation. It encrypts 64-bit plaintexts into about 147-bit ciphertexts, using 64-bit keys α. Plaintexts, ciphertexts, and keys are all real numbers $\in$ [0,1]. A major weakness of this system is its linearity which is not expected in cryptographic system design. Moreover, the system has been cryptanalyzed by Biham[Bie91] soon after their presentation in professional conference.

A public-key version of chaotic-map cryptosystem is proposed. The El Gamal's[EGt85] public-key encryption scheme is added into this public-key system, in addition to the chaotic map. The iterated map[ThS86] is an excellent candidate for constructing dynamic systems. In particular, a one-dimensional difference equation(iterated map) is well suitable to be a one-way function. Viewing this, a trapdoor was built by letting the legitimate receiver know iterated times of certain difference equation. Note that

the security of this system depends on the infeasibility of solving discrete logarithm over finite fields. As was true with the El Gamal's system, such an equivalence has never been proven.

The work is organized as follows: the public-key protocol is constructed in the second section. In the third section, concrete examples are illustrated. Finally, the security of chaotic-map public-key system is analyzed in the fourth section.

B. PUBLIC KEY PROTOCOL

First, the chaotic-map secret-key cryptosystem is reviewed. With such a system, arbitrary plaintext is encrypted into 2^n ciphertexts. Whereas, the decryption process guarantees the plaintext must be obtained by tracing the ciphertext backwards. In particular, this system is defined over real numbers, and for the sake of security, the iterated lower bound $n = 75$. The encryption process is

$$a_{i-1} = \begin{cases} \alpha a_i & \text{if } r_i = 0 \\ (\alpha - 1)a_i + 1 & \text{if } r_i = 1 \end{cases}, \tag{6.2.1}$$

where

$$a_{75} = X, \text{ and } a_0 = Y.$$

X is the plaintext, Y is the ciphertext, and α is the control parameter of map. X, Y, and α are $\in [0,1]$. Alternatively, the decryption process is

$$a_{i+1} = \begin{cases} a_i/\alpha & \text{if } 0 \le a_i \le \alpha \\ (a_i - 1)/(\alpha - 1) & \text{if } \alpha < a_i \le 1 \end{cases}. \tag{6.2.2}$$

Regarding the El Gamal's scheme, the general concept is given a generator g of cyclic group G_n with order s, where s is in $O(2^n)$, such that group elements can be represented as n-bit strings. The sender chooses a secret key k_1 uniformly distributed in $[0..n-1]$ and the receiver's public key is $y = g^{k_2}$, such that the encryption E is

$$E(m) = (g^{k_1}, y^{k_1} \times m), \tag{6.2.3}$$

and the decryption D is

$$D(c_1, c_2) = \frac{c_2}{c_1^{k_2}}, \tag{6.2.4}$$

where $\times$ might be replaced by any invertible operations.

Therefore, the chaotic-map public-key system is constructed as follows. Assume that each subscriber i of the system selects and publishes initial values a_0, a_n, and α which are uniformly distributed between 1 and p-1, where p is a large prime number($\approx$ 200 digits) such that p-1 has a large prime factor, as well as α is a primitive element of p, but the subscriber keeps n secret. a_n, the n-th iteration of a_0, can be generated from the difference equation

$$a_{i+1} \equiv \alpha a_i \ (mod\ p). \tag{6.2.5}$$

The following Figure 20 is a graphical interpretation of equation (6.2.5), where $\alpha = 2$ is defined as the slope of the line segments.

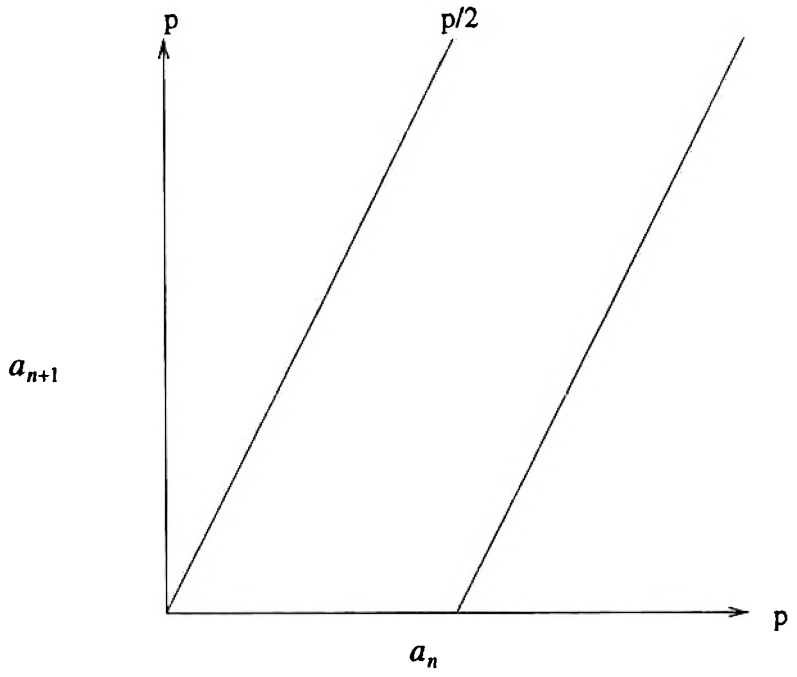

Figure 20. Graphical interpretation of equation 6.2.5.

Obviously, this process is concerned with the discrete case, instead of the continuous tent map of equation (6.2.1). In order to increase its security, it is anticipated that each subscriber will use a different modulus p instead of the universal modulus. Alternatively, the above one-dimensional difference equation can be replaced by any quadratic difference equation, such as

$$a_{i+1} = a_i^2 \ (mod\ p). \tag{6.2.6}$$

Thus, the encryption process of chaotic public-key system consists of three steps.

a. The sender randomly generates a positive secret integer k, and find out the receiver's public keys, a_0, a_n, and α.

b. The sender iterates a_0 to be a_k for k times, as well as iterates a_n to be a_{n+k}.

c. The ciphertexts are constructed as

$$c_1 = a_k,\ c_2 = m \times a_{n+k}\ (mod\ p),$$

where $m \in [0,\ p\text{-}1]$ is served as the plaintext. With the third step, it simulates the El Gamal's scheme and the ciphertext is double the size of the plaintext.

Whereas, on the receiver's side, the decryption process is composed of two steps.

a. The receiver i iterates c_1 by equation (6.2.5) n times to be s, where n is known to the receiver i only.

b. This step is to divide c_2 by s to recover the plaintext m.

Similarly, constructing the public keys by means of equation (6.2.5) can be replaced by quadratic difference equation (6.2.6), so that the encryption process and decryption process are not different. We will argue that the encryption process of chaotic-map public key system works correctly. All parameters involving in argument are identical to previous definitions, otherwise they will be specified particularly. The encryption process is

$$E(m) = (c_1, c_2) = (\alpha^k a_0, m\alpha^{n+k} a_0), \tag{6.2.7}$$

and the decryption process is to iterate c_1 to be s for n times, so

$$D(c_1, c_2) = \frac{c_2}{s} = \frac{m\alpha^{n+k}a_0}{\alpha^{n+k}a_0} = m. \tag{6.2.8}$$

Thus the result is the original message m. More specifically, two concrete examples are presented in the next section.

C. EXAMPLES

Assume that the receiver's public keys are $a_0 = 137$, $a_n = 64$, $p = 311$, $\alpha = 43$ and the secret key $n = 30$ such that a_n is calculated by equation (6.2.5). Given a message $m = 76$, the sender

a. selects $k = 15$ and then computes $a_{15} = 59$ as well as $a_{45} = 275$,

b. computes $c_1 = 59$ and $c_2 = m \times a_{45} \ (mod\ p) = 63$.

On the receiver side, only he knows $n = 30$, he then

a. iterates c_1 for 30 times and obtains $s = 275$,

b. computes $\frac{c_2}{s} \ (mod\ p) = 63 \times 95 \ (mod\ p) = 76$.

Thus, the message is recovered correctly.

Alternatively, we consider the quadratic difference equation. Each public key is identical to the above instance, except that $a_n = 200$. Thus, the sender

a. computes $a_{15} = 282$, as well as $a_{45} = 267$

b. computes $c_1 = 282$ and $c_2 = m \times a_{45} \ (mod\ p) = 77$.

Whereas, the receiver

a. iterates c_1 for 30 times to get $s = 267$, and then

b. computes $\frac{c_2}{s} \ (mod\ p) = 77 \times 106 \ (mod\ p) = 76$.

This system is shown to work properly for this example.

D. SECURITY

Reviewing El Gamal's scheme, he did not suggest that the sender's secret number is useable more than once. To know one block of message enables attackers to compute other blocks. Similarly, this system generates the different secret number k uniformly distributed in [0, p-1] for each encrypted block.

We claim that breaking the system is as hard as solving discrete logarithm over finite fields. First, considering if one can derive the secret key n from public keys. Given a_0, a_n, α, and p such that

$$a_n \equiv \alpha^n a_0 \ (mod\ p). \tag{6.4.1}$$

Then we have

$$\alpha^n \equiv \frac{a_n}{a_0} \equiv t \ (mod\ p). \tag{6.4.2}$$

Thus, the value n is computable from equation (6.4.2) only if the discrete logarithm over finite fields is also efficiently solvable. Whereas, the best algorithm[Oda84][LaO90] to solve discrete logarithm over finite fields remains exponential order. Meanwhile, the same argument can be employed for the secret number k. Secondly, we argue if one can recover the plaintext m by inspecting c_1 and c_2. Given c_1, c_2, and known parameters such that

$$c_1 \equiv a_k \equiv \alpha^k a_0 \ (mod\ p). \tag{6.4.3}$$

And we have

$$c_2 \equiv a_{n+k} m \equiv \alpha^{n+k} a_0 m \ (mod\ p). \tag{6.4.4}$$

Thus, if equation (6.4.2) is solvable to n, then it is easy to compute the plaintext m from $l \equiv \frac{c_2}{c_1} \equiv \alpha^n m \ (mod\ p)$. Furthermore, the size of modulus is considered as the safety margin of the RSA[RSA78] system and the discrete logarithm problem.

For the encryption process, there are two iterations plus one multiplication needed. At the worst case, each iteration makes use of $2 \log p$ multiplications in *GF(p)*. On the other hand, the decryption process needs only one iteration as well as one division. In order to enhance its utility, higher order chaotic map or difference equations can be employed as iteration equations in this public-key system. Such as $a_{n+1} \equiv a_n^r + d(mod\ p)$, where $r \geq 2$ and d is an arbitrary constant. Conceivably, the higher the order is, the more the work factor of attack is needed.

VII. CONCLUSION AND FUTURE RESEARCH

We have proposed a practical spline cryptosystem employing an interpolating cubic spline as an encryption method. It was found that the interpolating cubic polynomial, especially the cubic spline, can be employed as an encrypting method. The random spline cryptosystem has the additional features of the mathematical indeterminacy of computing keys and sensitivity of boundary conditions and offset t_i, making it more difficult to break. Notice that the random spline cryptosystem is a totally different mechanism from existing ones. We understand that it is not intended to replace all other methods, but rather as an alternative way to enlarge cryptographic research. In this regard, the random spline cryptosystem offers us a promising answer. In addition, there are additional research topics related to spline cryptosystem. For instance, we may analyze its security intensively, design an efficient key management, promote the spline cryptosystem to be a public-key version, and design cryptographic protocols under public-key version.

The significance of multi-segment splines was demonstrated for the execution behavior of the RSA system. Under the integer domains of $f(x)$ and $g(y)$, the encipherment and the decipherment can be executed correctly as the RSA's. Furthermore, it was shown that reformulation of both, the encryption and decryption processes, are necessary conditions for inspecting the similar characteristic of random spline cryptosystem. This implies that the random spline cryptosystem functions properly.

In addition, a chaotic-map public-key cryptosystem, based on the chaotic behavior or bifurcation of one-dimensional difference equations that are defined over finite fields, was proposed. It was found that such equations are suitable to be one-way functions in

terms of constructing public-key cryptosystems. One-dimensional equations were chosen rather than higher-dimensional ones because higher-dimensional equations are invertible. The security of such a system relies on the infeasible computation of discrete logarithm over finite fields. For the sake of security, the sizes of ciphertexts and public keys are the double size of the RSA system.

APPENDIX

26 INVALID KEY VALUES OF $k_1 \in [0,255]$

k_1	k_5	k_5 mod 10
0(00000000)	0(00000000)	0
5(00000101)	80(01010000)	0
10(00001010)	160(10100000)	0
15(00001111)	240(11110000)	0
35(00100011)	50(00110010)	0
40(00101000)	130(10000010)	0
45(00101101)	210(11010010)	0
65(01000001)	20(00010100)	0
70(01000110)	100(01100100)	0
75(01001011)	180(10110100)	0
100(01100100)	70(01000110)	0
105(01101001)	150(10010110)	0
110(01101110)	230(11100110)	0
130(10000010)	40(00101000)	0
135(10000111)	120(01111000)	0
140(10001100)	200(11001000)	0
160(10100000)	10(00001010)	0
165(10100101)	90(01011010)	0
170(10101010)	170(10101010)	0
175(10101111)	250(11111010)	0
195(11000011)	60(00111100)	0
200(11001000)	140(10001100)	0

205(11001101)	220(11011100)	0
225(11100001)	30(00011110)	0
230(11100110)	110(01101110)	0
235(11101011)	190(10111110)	0

BIBLIOGRAPHY

[AdM89]. Adams, C. M. and Meijer, H. "Security-related Comments Regarding McEliece's Public-key Cryptosystem", *IEEE Trans. Info. Th.*, vol. IT-35, pp454-457, March 1989

[AdR76]. Adams, J. Alan, and Rogers, David F. *Mathematical Elements for Computer Graphics"*, Mcgraw-Hill, New York, 1976

[BBF83]. Bauer, R.K., Berson, T.A., and Freiertag, R.J. "A Key Distribution Protocol Using Event Markers", *ACM TOCS 13,* 1983, pp.249-255

[Bie91]. Biham, E. "Cryptanalysis of The Chaotic-Map Cryptosystem Suggested at EUROCRYPT'91", in *Adv. in Cryptology-Eurocrypt'91,* pp.532-534, 1991

[BiS90]. Biham, E. and Shamir, A. "Differential Cryptanalysis of DES'like Cryptosystem", *The Weizmann Institute of Science, Department of Applied Mathematics,* Technical Report CS90-16, 1990

[BJM92]. Bird, R., Gopal, A., Herzberg, A., Jerson, P., Kutten, S., Molva, R., and Yung, M. "Systematic Design of Two-party Authentication Protocols", *In Adv. in Cryptology - proceedings of CRYPTO'91.* Lecture Notes in Computer Science, pp.44-61, Spring-Verlag, 1992

[BMR90]. Beaver, D., Micali, S., and Rogaway, P. "The Round Complexity of Secure Protocols", *In proc. 22th ACM Symposium on Theory of Computing,* May 1990

[Bre84]. Brickell, E.F. "Solving Low Density Knapsacks", *Advances in Cryptology(Proc. Crypto83),* New York, pp.25-37, 1984

[Bre85]. Brickell, E.F. "Breaking The Iterated Knapsacks", *Advances in Cryptology(Proc. Crypto84),* Springer-Verlag, Berlin, pp.342-358, 1985

[Bre88]. Brickell, E.F. "The Cryptanalysis of Knapsack Cryptosystems", *Appl. Discrete Math.*, SIAM, pp.3-23, 1988

[BsM90]. Bellovin, S.M. and Merritt, M. "Limitations of The Kerberos Authentication System", *ACM Computer Communication Review 30,* 1990, pp.119-132

[Cdv90]. Chaum, D., and van Antwerpen, H. "Undeniable Signatures", *In Adv. in Cryptology - proceedings of CRYPTO'89,* Lecture Notes in Computer Science, Springer-Verlag, 1990

[ChC88]. Chaum, D., Crepeau, C., and Damgard, I. "Multi-party Unconditionally Secure Protocols", *In proc. 20th ACM Symposium on Theory of Computing,* Chicago, ACM 1988

[Cod84]. Coppersmith, D. "Fast Evaluation Of Discrete Logarithms In Field Of Characteristic two", *IEEE Trans. Inform. Theory 30* pp.587-594, 1984

[DaE77]. "Data Encryption Standard", *FIPS PUB46, National Tech. Info. Service,* Springfield, VA, 1977

[dBc78]. de Boor, Carl, *A practical Guide to Spline,* Springer-Verlag, New York, 1978

[Der82]. Denning, R. D. *Cryptography and Data Security,* Addison-Wesley, 1982

[DeS81]. Denning, D.E. and Sacco, G.M. "Timestamps in Key Distribution Systems", *CACM 24* Aug. 1981, pp.533-536

[Dey88]. Desmedt, Y., "Society and Group Oriented Cryptography:A New Concept", *In Advances in Cryptology - proc. of CRYPTO 87,* Lecture Notes in Computer Science, pages 120-127, 1988

[DFr90]. Desmedt, Y. and Frankel, Y. "Threshold Cryptosystem". *Advances in Cryptology - proc. of CRYPTO 89,* Lecture Notes in Computer Science, pages 307-315, 1990

[DGo84]. Desmedt, Y.G., Vandewalle, J.P., and Govaerts, R.J.M. "A Critical Analysis of The Security of Knapsack Public-key Algorithm", *IEEE Trans. Inform. Theory* IT-30, pp.601-611, 1984

[Dif82]. Diffie, W. "Conventional versus Public-key Cryptosystem", *AAAS Selected Symposium 69.*, Westview Press, Boulder, CO, 1982

[DiH76]. Diffie, W and Hellman, M.E. "New Directions In Cryptography", *IEEE Trans. Inform. Theory,* vol. IT-22, pp.644-654, Nov. 1976

[DiH77]. Diffie, W. and Hellman, M.E. "Exhaustive Cryptanalysis of the NBS Data Encryption Standard", *Computer,* vol. 10, no. 6, pp.74-84, June 1977

[DiH79]. Diffie, W. and Hellman, M.E. "Privacy and Authentication:An Introduction to Cryptography", *Proc. IEEE,* vol. 67, pp.397-427, Mar. 1979

[EGt85]. El Gamal, T. "A Public-Key Cryptosystem And Signature Scheme Based on Discrete Logarithms", *IEEE Trans. Inform. Theory,* vol. IT-31. no.4, pp.469-472, July 1985

[Fry90]. Frankel, Y. "A Practical Protocol for Large Group Oriented Networks". *Advances in Cryptology - proc. of EUROCRYPT 89,* Lecture Notes in Computer Sciences, pages 56-61, 1990

[GMR85]. Goldwasser, S., Micali, S., and Rackoff, C. "The Knowledge Complexity Of Interactive Proof System", *Proc. 17th ACM symposium on theory of computing,* Providence RI, pp.291-304, 1985

[GMW86]. Goldrach, O., Micali, S., and Wigderson, A. "Proof That Yields Nothing But Their Validity And A Methodology Of Cryptographic Protocol Design", *Proc. IEEE of the 27th IEEE symposium on foundations of computer science,* Toronto, pp.174-187, 1986

[HaN91]. Habutsu, T., Nishio, Y., Sasase, I., Mori, S. "A Secret-key Cryptosystem by Iterating Chaotic Map", In *Adv. in Cryptology-Eurocrypt'91,* pp.127-140, 1991

[Hem80]. Hellman, M.E. "A Cryptanalytic Time-Memory Tradeoff", *IEEE Tran. Inform. Theory,* vol. 26, no. 4, pp.401-406, July 1980

[HqH92]. Ho, C.Y., Xuan, Q., and Hwu, F. "Spline Cryptosystem", *Sigsac ACM,* vol. 10, pp.6-15, winter 1992

[Hwt91]. Hwang, T. "Cryptosystem for Group Oriented Cryptography", *Advances in Cryptology - proc. of EUROCRYPT 90,* Lecture Notes in Computer Science, pp.352-360, 1991

[Knd81]. Knuth, D. "Semi Numerical Algorithm", *in the Art of computer programming,* vol. 2, 2nd ed. Reading, MA. Addison-Wesley, pp.316-336, 1981

[Kon87]. Koblitz, N. "Elliptic Curve Cryptosystem", *Math. of Computation 48,* 1987, pp.203-209

[KoT91]. Korzhik, V. and Turkin, A.I. "Cryptanalysis of McEliece's Public-key Cryptosystem", *Adv. in Cryptology-EUROCRYPT'91,* Lecture Notes in Computer Science, pp.68-70, 1991

[LaO90]. LaMacchia, B.A. and Odlyzko, A.M. "Computation Of Discrete Logarithms In Finite Field", 1990

[LaS86]. Lancaster, P., Salkauskas, K. *Curve and Surface fitting:An introduction,* Academic Press, San Diego, 1986

[Leh83]. Lenstra, H.W. "Integer Programming with a Fixed Number of Variables", *Math. Operation Research,* vol. 8, no. 4, pp.538-548, Nov. 1983

[Leh86]. Lenstra, H.W. "Elliptic Curves and Number-theoretic Algorithm", *Proc. Int'l Cong. Math.* Berkeley, CA. 1986

[Leh87]. Lenstra, H.W. "Factoring Integers With Elliptic Curves", *Ann. of Math,* 126, pp.649-673, 1987

[Mcr78]. McEliece, R.J. "A Public-key Cryptosystem Based on Algebraic Coding Theory", *JPL DSN progress Rep.42-44,* pp.114-116, Jan.-Feb. 1978

[MeH78]. Merkle, R.C. and Hellman, M.E. "Hiding Information and Signatures in Trapdoor knapsacks", *IEEE Trans. Inform. Theory,* vol. IT-24, pp.525-530, Sept. 1978

[Mop87]. Montgomery, P.L. "Speeding The Pollard And Elliptic Curves Methods Of Factorization", *Math. Comp.,* 48, pp.243-264, 1987

[MoS90]. Montgomery, P.L. and Silverman, R.D. "An FFT Extension To The p-1 Factoring algorithm", *Math. Comp.,* 1990

[Oda84]. Odlyzko, A.M. "Discrete Logarithms In Finite Fields And Their Cryptographic Significance", *Adv. in Cryptology(Proc. of Eurocrypt84) Lecture Notes in Computer Science,* vol. 209, springer-Verlag, New York, pp.224-314, 1984

[PoH78]. Pohlig, S.C. and Hellman, M.E. "An Improved Algorithm for Computing Logarithms over GF(p) and its Cryptographic Significance", *IEEE Trans, on Inform. Theory 24,* pp.106-110, 1978

[Pom84]. Pomerance, C. "The Quadratic Sieve Factoring Algorithm", *Proc. EUROCRYPT 84: Advances in Cryptology,* New york-London, PP.169-182, 1984

[Ram79]. Rabin, M.O. "Digitalized Signatures and Public-key Functions as Intractable as Factorization", *MIT Lab. for Computer Science,* MIT/LCS/TR-212, Jan. 1979

[Rab80]. Rabin, M.O. "Probabilistic Algorithm for Testing Primality", *Journal of Number Theory,* 12:128-138, 1980

[RaB89]. Rabin, T. and Ben-Or M. "Verifiable Secret Sharing and Multiparty Protocols with Honest Majority", *In proc. 21th ACM STOC,* 1989

[RcM71]. Rao, C. R. and Mitra, S. K. *Generalized inverse of matrices and its applications.* John Wiley & Sons, New York, 1971

[RiS84]. Rivest, R.L. and Shamir, A. "How to Expose an Eavesdropper", *Comm. of the ACM 27,* 1984, pp.393-395

[RSA78]. Rivest, R.L., Shamir, A., and Adleman, L. "A Method for Obtaining Digitals Signatures and Public-key Cryptosystems", *Commun. ACM,* vol. 21, pp.120-126, Apr. 1978

[Scl81]. Schumaker, Larry L. *Spline functions:Basic Theory,* John Wiley & Sons, New York, 1981

[SeP89]. Seberry, J. and Pieprzyk, J., *Cryptography:An Introduction to Computer Security,* Prentice Hall, New York, 1989

[Sha79]. Shamir, A. "How to Share a Secret", *Comm. of the ACM,* 22:612-613, November 1979

[Sha84]. Shamir, A. "A Polynomial Time Algorithm for Breaking the Basic Merkle-Hellman Cryptosystem", *IEEE Tran. Inform. Theory,* vol. IT-30, no.5, pp.699-704, Sept. 1984

[ThS86]. Thompson, J. M. T. and Stewart, H. B. *Nonlinear Dynamics and Chaos,* John Wiley and Sons, Chichester, 1986

[Wih80]. Williams, H.C. "A Modification of The RSA Public-key Cryptosystem", *IEEE Trans. Inform. Theory,* Vol. IT-26, no.6, pp.726-729, Nov. 1980