STUDY ON ELLIPTIC CURVE PUBLIC KEY CRYPTOSYSTEMS WITH APPLICATION OF PSEUDORANDOM NUMBER GENERATOR

BY

YUEN CHING WAH

A THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTER OF PHILOSOPHY

DIVISION OF INFORMATION ENGINEERING

THE CHINESE UNIVERSITY OF HONG KONG

JUNE 1998

簡介

近來有很多密碼系統都採用橢圓曲線，因爲不少學者和用者都相信橢圓曲線密碼系統擁用同一樣的保安效能但它的私鑰和公鑰長度卻比較其他密碼系統爲短。

這篇論文主要研究數碼簽署。 在現今的數碼簽署，每一個簽署都需要一個新的僞隨機數， 所以僞隨機數產生器對於數碼簽署系統是很重要。 論文中會講及如果 Digital Signature Algorithm 和 Nyberg-Rueppel Digital Signature 這兩個數碼簽署系統的僞隨機數產生器是 LCG 、 ICG 或 PN-sequence generator 時，採用橢圓曲找出私鎖的時間和採用離散對數的時間差不多。 所以無論採用橢圓曲線或離散對數時，用者要注意僞隨機數產生器的特質 。

Acknowledgement

This thesis would not have been possible without my supervisor, Prof. Victor Wei not only because of his helpful guidance but because his assertive style of doing research. Working with him is a true privilege, and I have benefited tremendously from his knowledge of science, both in depth and broadness. While being a constant source of valuable insights and advice, he has always allowed me high degree of freedom and independence to pursue my own research and study.

My special thanks to Prof. Kwok-wai Cheung and Prof. Yu-ngai Hui for their interest in my work and their precious time serving on my defense examinations committees.

I am also grateful to all the members of the Information Integrity Laboratories especially, S.L. Cheng, C.K. Lai, C.W. Lam, S.K. Lau, C.H. Lee, S.W. Ng and Duncan Wong. They made the last two years really enjoyable.

Above all, I would like to express my greatest gratitude to my family and my best friends. Thanks to Jason, Ming-ming, Ethel, Dorothy, Jacqueline, Nico, Caroline, Heidee, Goretti, Cynthia, Kris and Eliza for their friendship and their patient ears. During the last two years, my family gave me unconditional encouragement and support.

Abstract

Recently, there are many cryptosystems which are implemented with elliptic curve since many researchers and users believe that elliptic curve cryptosystems provide a shorter key length and the same security when comparing with any well-known public-key cryptosystem.

This thesis mainly concerns on the elliptic curve discrete logarithm problem based digital signature schemes and discrete logarithm problem based digital signature scheme. Many digital signature schemes require the signer to generate a new random number with every signature, so pseudorandom number generator is a critical factor in these schemes.

When LCG, ICG or PN-sequence generator is used in Digital Signature Algorithm or Nyberg-Rueppel Digital Signature Schemes, it takes approximately equal time to crack the schemes operated in the elliptic curve abelian group and the schemes operated in the finite field group given that they use the same key sizes.

Therefore, one should take caution against which pseudorandom number generator is used no matter which digital signature scheme is used and which implementation is underlied . Although the random number seems to be hidden in the digital signature scheme, we also need to pay more attention to it. In other words, this thesis also illustrates the high vulnerability of the digital signature schemes to the weaknesses in the random number generation process.

Contents

Chapter 1

Introduction

1.1 Why use cryptography?

The direction of communication and information exchange has been radically altered by emerging computer and communications technologies. Along with the speed, efficiency and cost-saving benefits of the digital revolution, come new challenges to the security and privacy of communications and information traversing the global communications infrastructure.

In response to these challenges, the security mechanisms of traditional paper-based communications media – envelopes and locked filing cabinets – are being replaced by cryptographic security techniques.

Through the use of cryptography, communication and information stored and transmitted by computers can be protected against interception to a very high degree. Recently, non-governmental demand for encryption is increased dramatically. Modern encryption technology – a mathematical process involving the use of formulas (or algorithms) – was traditionally deployed most widely to protect the confidentiality of military and diplomatic communications.

With the advent of the computer revolution, and recent innovations in the science of encryption, a new market for cryptographic products was developed. Electronic communications are now widely used in the civilian sector and have become an integral component of the global economy. Computers store and exchange an ever-increasing amount of highly personal information, including medical and financial data. In this electronic environment, the need for privacy-enhancing technologies is apparent. Communications applications such as electronic mail and electronic fund transfers require secure means of encryption and authentication – features that can only be provided if cryptographic know-how is widely available and unencumbered by government regulation.

Encryption ensures the confidentiality of personal records, such as medical information, personal financial data, and electronic mail. In a networked environment, such information is increasingly at risk of theft or misuse.

1.2 Why is authentication important ?

There is a tremendous potential for fraud in the electronic world. Transactions take place at a distance without the benefit of physical clues that permit identification, making impersonation easy. The ability to make perfect copies and undetectable alterations of digitised data complicates the matter. Traditionally hand-written signatures serve to determine the authenticity of an original document. However, in the electronic world hand-written signatures does not exist and we need to provide a method to prevent impersonation. In the electronic world, the concept of an original document is problematic, but a digital signature can verify data integrity, and provide authentication and non-repudiation functions to certify the sender of the data.

1.3 What is the relationship between authentication and digital signature?

A digital signature functions for electronic documents just as a handwritten signature does for printed documents. The signature is an unforgeable piece of data that asserts that a named person wrote or otherwise agreed to the document to which the signature is attached. A digital signature actually provides a greater degree of security than a handwritten signature. The recipient of a digitally signed message can verify both that the message originated from the person whose signature is attached and that the message has not been altered either intentionally or accidentally since it was signed. Furthermore, secure digital signatures cannot be repudiated; the signer of a document cannot later disown it by claiming the signature was forged. In other words, digital signatures enable "authentication" of digital messages, assuring the recipient of a digital message of both the identity of the sender and the integrity of the message.

1.4 Why is random number important?

A cryptographic system can only be as strong as the encryption algorithms, digital signature algorithms, one-way hash functions, random number used in system and message authentication codes it relies on. Therefore, breaking one of them will imply that the system is broken. The most weak link in cryptographic system is the random number used in system.

In August 1995, Damien Doligez, a student at the Ecole Polytechnique in Paris, used a network of 120 computers to crack an encrypted session in eight days with generate a Netscape secret key. But, using this brute force techniques, the cost to break the encryption of any single credit card transaction far exceeds the possible financial reward.

Several weeks later, when two graduate students at Berkeley, Ian Goldberg and David Wagner, tried to figured out a quicker way to crack the Netscape code than Damien Doligez did. They discovered that Netscape uses a formula based on date, time and other known information to generate a random number to create secret key. Therefore, they could find out the secret key easily and could read all the data and messages which are transmitted through Netscape. This implied that they could break Netscape's security coding system in less than a minute. Therefore, no one using the software can be certain of protecting credit card information, bank account numbers or other types of information that Netscape is supposed to keep private during on-line transactions.

The Berkeley students identified a basic flaw in the Netscape random number generation, were able to narrowly focus their attack to quickly break the code, with far less computer power.

When comparing with the Berkeley students and Ecole Polytechnique student, if a cracker focuses on predicting the random number, the system is much easily cracked. Moreover, if a weak random number generator is used in the system, it will greatly reduce the job of a cracker. In other words, a weak random number generator will cause a vital risk in the cryptographic system.

Chapter 2

Background

2.1 Cryptography

Cryptography has been used for millenia to safeguard military and diplomatic communication. There are two types of cryptography : 1) Symmetric key cryptography, 2) Asymmetric key cryptography.

2.1.1 Symmetric key cryptography

Figure 2.1 shows the basic symmetric key cryptography model.
The original intelligible message (M) , referred to as *plaintext*, is converted into apparently random nonsense, referred to as *ciphertext* ($E_K(M)$). The encryption process consists of an algorithm (E) and a key(K). The key is generated by the random source and which is a value independent of the plaintext that controls the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. Changing the key changes the output of the algorithm.

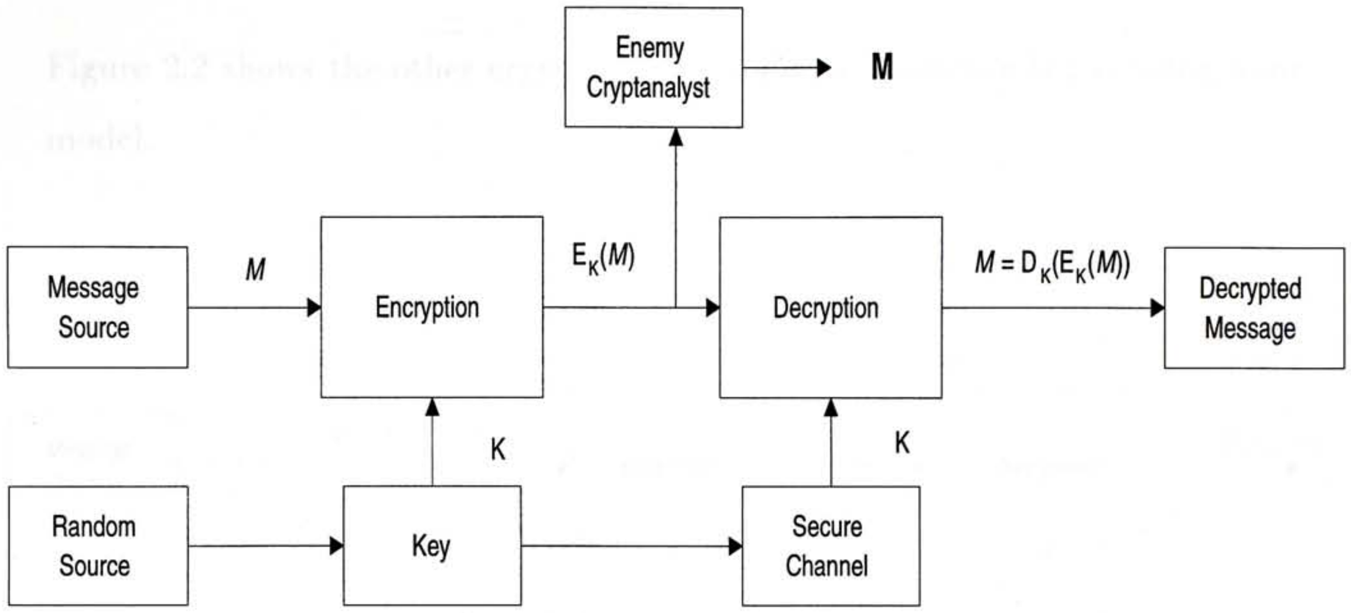


Figure 2.1: Basic symmetric key cryptography model

Once the ciphertext is produced, it is transmitted. Upon, reception, the ciphertext can be transformed back to the original plaintext by using a decryption algorithm and the same key that was used for encryption.

The security of conventional cryptosystem depends on the several factors. First, the encryption algorithm must be powerful enough so that it is impractical to decrypt a message on the basis of the ciphertext alone. Beyond that, the security of the algorithm. That is, it is assumed that it is impractical to decrypt a message on the basis of the ciphertext plus knowledge of the encryption/decryption algorithm. In other words, we do not need to keep the algorithm secret; we need to keep only the key secret.

2.1.2 Asymmetric key cryptography

Figure 2.2 shows the other cryptography model, asymmetric key cryptography model.

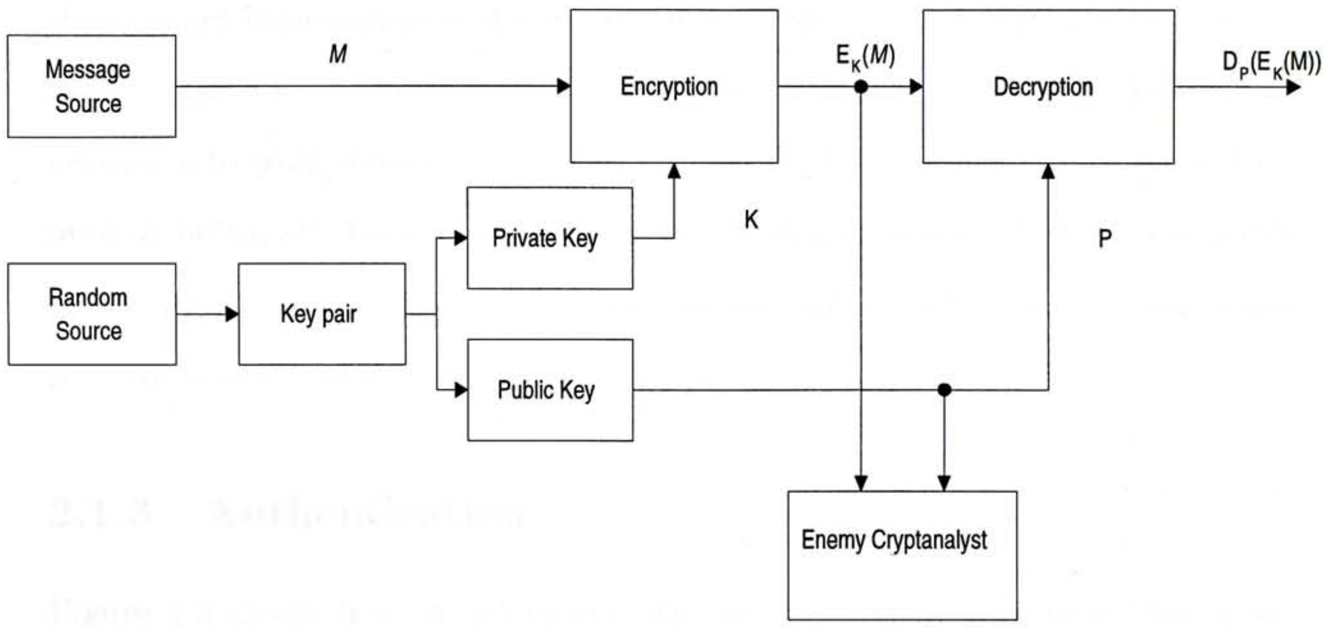


Figure 2.2: Basic asymmetric key cryptography model

The main problem of symmetric key cryptography is that the key should be transmitted in a secure channel. If the sender and the receiver are in separate physical locations, they must trust a courier, or a phone system, or some other transmission medium to prevent the disclosure of the key being communicated. Anyone who overhears or intercepts the key in transit can later read, modify, and forge all messages encrypted or authenticated using that key.

Public-key cryptography was introduced in 1976 by Whitfield Diffie and Martin Hellman for solving the key management problem in symmetric key cryptography. Where, key management means the generation, transmission and storage of keys. Each person owns a pair of keys, one is called the public key(P) and the other is called the private key(K). Each person's public key is published while the private key is kept secret. The need for the sender and receiver to share secret information is eliminated since all the communications involve only public keys, and no private key is ever transmitted or shared. No longer is it necessary to trust some communications channel to be secure against eavesdropping or betrayal. Anyone can send a confidential message by just using public information, but the message can only be decrypted with a private key, which is in the sole possession of the intended recipient.

2.1.3 Authentication

Figure 2.3 shows how an adversary can decrypt messages intended for a second entity without breaking the encryption system. The adversary sends Alice's(sender's) a public key e' which Alice assumes(incorrectly) to be the public key of Bob to impersonated Bob. Then, the adversary intercepts encrypted messages from Alice to Bob, decrypts with its own private key d', re-encryptes the message under Bob's public key e, and sends it to Bob. The adversary impersonates Bob successfully. This points out the necessity to authenticate public keys to achieve data origin authentication of the public keys themselves.

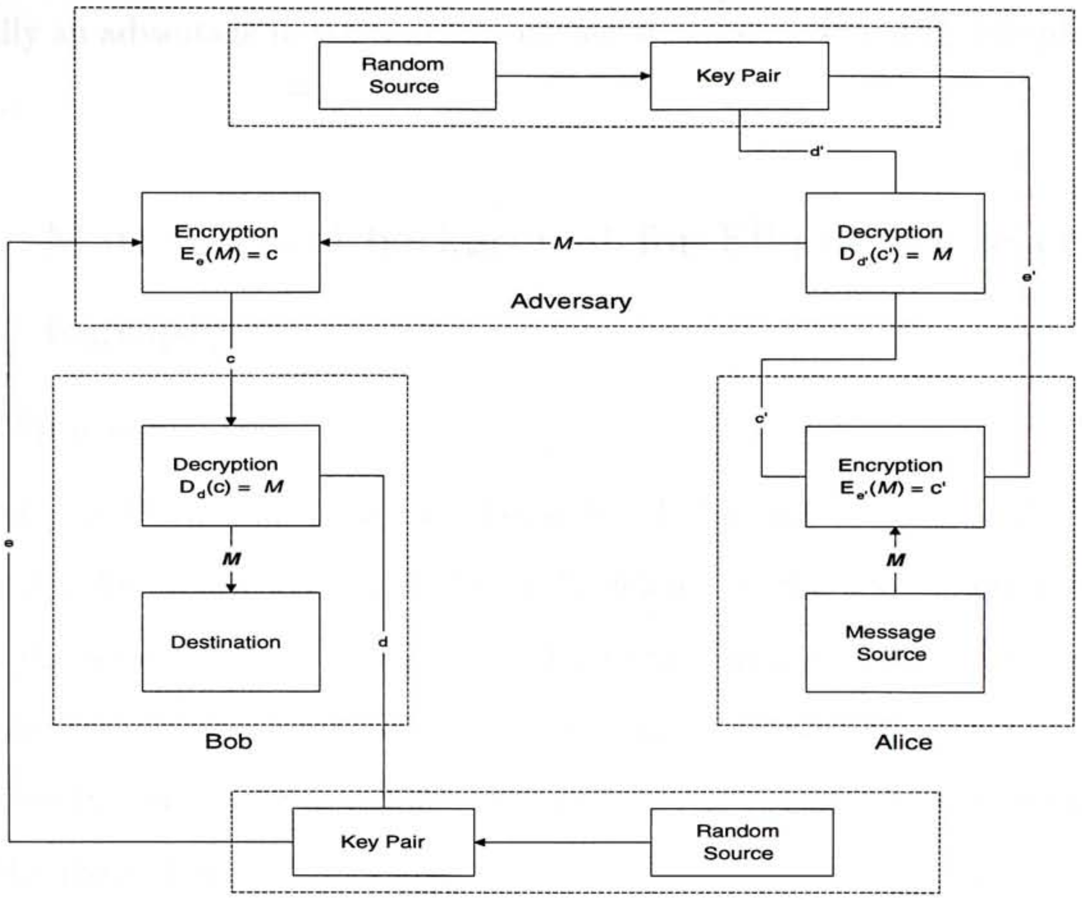


Figure 2.3: Scenario of an adversary impersonate a person

2.2 Elliptic curve cryptography

Elliptic curve cryptosystems were first suggested by Miller [8] and Koblitz [9]. And recently elliptic curve has been used in algorithms for factoring integers [10] [11], primality proving [12] , pseudorandom number generation [13], one-way permutations [14] and public-key cryptography [15]. Elliptic curve cryptography is the highest strength-per-key-bit of any known public-key scheme. That is, it provides equivalent security as existing public key schemes but with shorter key lengths. Moreover, it provides relatively small block sizes, high-speed software and hardware implementation. When elliptic curves cryptosystem is compared with RSA, it is regarded as being faster and requires less processing power. It is especially an advantage in the implementation of smartcards and mobile phones system.

2.2.1 Mathematical background for Elliptic curve cryptography

1. Elliptic curves over F_p

   Let $p > 3$ be a prime number. Let a, $b \in F_p$ be such that $4a^3 + 27b^2 \neq 0$ in F_p. An elliptic curve $E(F_p)$ over F_p defined by the parameters a and b is the set of solutions (x, y), $x, y \in F_p$, to the equation: $y^2 = x^3 + ax + b$, together with an extra point $\mathcal{O}$, the point at infinity and serves as the identity element. The number of points in $E(F_p)$ is denoted by $\#E(F_p)$. The Hasse Theorem tells that

   $$p + 1 - 2\sqrt{p} \leq \#E(F_p) \leq p + 1 + 2\sqrt{p}$$

   The set of points $E(F_p)$ forms a group with the following addition rules [16]: Let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ and $P_3 = (x_3, y_3)$. where $x_1 \neq x_2$

   (a) $\mathcal{O} + \mathcal{O} = \mathcal{O}$

   (b) $P_1 + \mathcal{O} = \mathcal{O} + P_1 = P_1$

   (c) $-P_1 = (x_1, -y_1)$

   (d) $P_1 + (-P_1) = \mathcal{O}$

   (e) Rule for adding two distinct points that are not inverses of each other if $P_3 = P_1 + P_2$, then $-P_3$ is the third point of intersection of the line P_1P_2 and

- $x_3 = \lambda^2 - x_1 - x_2$, $y_3 = \lambda(x_1 - x_3) - y_1$, where $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$

(f) Rule for doubling a point if $P_3 = P_1 + P_1$, then $-P_3$ is the third point of intersection of the tangent line to the curve at P_1 and

- $x_3 = \lambda^2 - 2x_1$, $y_3 = \lambda(x_1 - x_3) - y_1$, where $\lambda = \frac{3x_1^2 + a}{2y_1}$

From the last two equations, we can find that the group $E(F_p)$ is abelian, that is $P_1 + P_2 = P_2 + P_1$, for all point P_1 and P_2 in $E(F_p)$.

2. Elliptic curves over F_{2^m}

A non-supersingular elliptic curve $E(F_{2^m})$ over F_{2^m} defined by the parameters a, $b \in F_{2^m}$, where $b \neq 0$, is the set of solutions (x, y), $x \in F_{2^m}$, $y \in F_{2^m}$, to the equation $y^2 + xy = x^3 + ax^2 + b$ together with an extra point $\mathcal{O}$, the point at infinity. The number of points in $E(F_{2^m})$ is denoted by $\# E(F_{2^m})$. The Hasse Theorem tells us that

$$q + 1 - 2\sqrt{q} \leq \#E(F_q) \leq q + 1 + 2\sqrt{q}$$

where $q = 2^m$. Further,$\#E(F_p)$ is even.

(a) $\mathcal{O} + \mathcal{O} = \mathcal{O}$

(b) $P_1 + \mathcal{O} = \mathcal{O} + P_1 = P_1$

(c) $-P_1 = (x_1, x_1 + y_1)$

(d) $P_1 + (-P_1) = \mathcal{O}$

(e) Rule for adding two distinct points that are not inverses of each other if $P_3 = P_1 + P_2$, then $-P_3$ is the third point of intersection of the line P_1P_2 and

- $x_3 = \lambda^2 + \lambda + x_1 + x_2 + a$, $y_3 = \lambda(x_1 + x_3) + x_3 + y_1$, where $\lambda = \frac{y_2+y_1}{x_2+x_1}$

(f) Rule for doubling a point if $P_3 = P_1 + P_1$, then $-P_3$ is the third point of intersection of the tang ent line to the curve at P_1 and

- $x_3 = \lambda^2 + \lambda + a$, $y_3 = +x_1^2 + (\lambda + 1)x_3$, where $\lambda = x_1 + \frac{y_1}{x_1}$

From the last two equations, we can find that the group $E(F_{2^m})$ is abelian, that is $P_1 + P_2 = P_2 + P_1$, for all point P_1 and P_2 in $E(F_{2^m})$.

2.3 Pseudorandom number generator

Cryptographic applications demand much more out of a pseudorandom number generator than most applications do. For a source of bits to be cryptographically random, it must be computationally impossible to predict what the Nth random bit will be given complete knowledge of the algorithm or hardware generating the stream and the sequence of 0th through N-1st bits, for all N up to the lifetime of the source [19], [20], [21], [22].

A software generator (also known as pseudo-random) has the function of expanding a truly random seed to a longer string of apparently random bits. This seed must be large enough not to be guessed by the opponent. Ideally, it should also be truly random (perhaps generated by a hardware random number source).

2.3.1 Linear Congruential Generator

One of the most popular pseudorandom number generator is linear congruential generator(LCG). This type of generator uses a method similar to the folding schemes in chaotic maps. The general formula is,

$$k_{i+1} = a \times k_i + b \pmod M \quad \forall i \geq 0$$

The values a, b and M are pre-selected constants. a is known as the miltiplier, b is the increment, and M is the modulus. The quality of the generator is strongly dependent upon the choice of these constants(a significant part of [1] is described the property of linear congruential generator fully.) The main advantage of LCG is that it is very easy to program. When the parameters a, b and M are known, given k_0 all the other k_i can be easily computed. Even if the parameters a, b and M are unknown, the sequences k_i are still predictable when some of the k_i are given [23]. One fairly obvious goal is to make the period(the time before the generator repeats the sequence) long, this implies that M should be as large as possible.

2.3.2 Inversive Congruential Generator

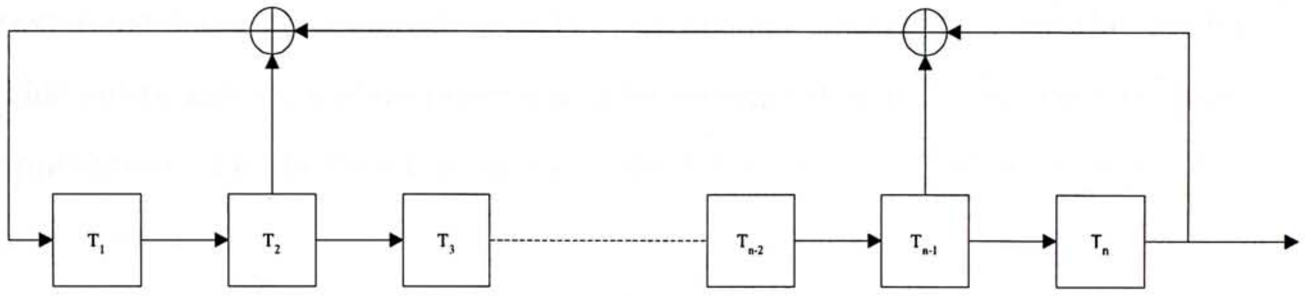
Inversive congruential generator(ICG) is a very promising new approach to produce uniform pseudorandom numbers. From [2], the properity of ICG is being discussed. For a given prime number p, and for $c \in Z_p$, let $\bar{c}$ is the inverse of c , so $\bar{c} = 0$ if $c = 0$ and $\bar{c} = c^{-1}$ if $c \neq 0$. In other words, $\bar{c}$ equals the number c^{p-2} (mod p). Inversive congruential generators are due to Eichenauer and Lehn [3]. The general formula is,

$$k_{i+1} = a \times \bar{k}_i + b \pmod p \quad \forall i \geq 0$$

A prominent feature of the ICG with prime modulus is the absence of any lattice structure, in sharp contrast to linear congruential generators.

2.3.3 PN-sequence generator

PN-sequence generator is a kind of shift register generator. A linear shift register is shown in the following figure. PN-sequence has the following properties:



1. Length of a maximal-length code $= 2^n - 1$ when generated by n shift registers.
2. Every possible n-bit word exits in the n-stage shift register at some time during the generation of a complete code cycle and only once during the cycle.
3. Each period of the sequence contain 2^{n-1} ones
4. Each period of the sequence contain $2^{n-1} - 1$ zeros

2.4 Digital Signature Scheme

A digital signature is created by running message text through a hashing algorithm. This yields a message digest. The message digest is then encrypted using the private key of the individual who is sending the message, turning it into a digital signature. The digital signature can only be decrypted by the public key of the same individual. The recipient of the message decrypts the digital signature and then recalculates the message digest. The value of this newly calculated message digest is compared to the value of the message digest found from the signature. If the two match, the message will not have been tampered with. Since the public key of the sender was used to verify the signature, the text must have been signed with the private key known only by the sender. This entire authentication process will be incorporated into any security-aware application. The following is an example of how a digital signature is used to authenticity.

Suppose Alice wants to send a signed message to Bob. She creates a message digest by using a hash function on the message. The message digest serves as a "digital fingerprint" of the message; if any part of the message is modified, the hash function returns a different result. Alice then encrypts the message digest with her private key. This encrypted message digest is the digital signature for the message. Alice sends both the message and the digital signature to Bob. When Bob receives them, he decrypts the signature using Alice's public key, thus revealing the message digest. To verify the message, he then hashes the message with the same hash function Alice used and compares the result to the message digest he received from Alice. If they are exactly equal, Bob can be confident that the message is indeed sent from Alice and has not changed since she signed it. If the message digests are not equal, the message either originated elsewhere or was altered after it was signed. Note that using a digital signature does not encrypt the message itself. If Alice wants to ensure the privacy of the message, she must also encrypt it using Bob's public key. Then only Bob can read the message by decrypting it with his private key. It is not feasible for anyone to either find a message which hashes to a given value or to find two messages that hash to the same value. If either were feasible, an intruder could attach a false message onto Alice's signature. Specific hash functions have been designed to have the property that finding a match is not feasible, and are therefore considered suitable for use in cryptography.

2.5 Babai's lattice vector algorithm

In this thesis, we will use Babai's lattice vector alogrithm [4] as a part of subroutine to solve the simultaneous equations in different moduli. In [4], Laszlo Babai suggests two algorithms to find a point of a given lattice. Some terms will be introduced first.

Lattice: Given a set of l linear independent vectors in $\mathbf{R^l}$, $E = \{e_1, \ldots, e_l\}$, we define the lattice spanned by E(the basis of the lattice) be the set of all the possible linear combinations of e_i's with integral coefficients, that is:

$$L(E) = \{\sum_{i=1}^{l} c_i e_i \text{ such that } c_i \in Z \quad \forall i\}$$

Nearby Lattice Point Problem: Given E and x, find $w \in L$ such that $||x - w|| \leq C||x - v||$, where C is a function of l and v is the nearest neighor of x in E.

Lovasz lattice reduction algorithm [5] originally designed to give nearly optimal simultaneous diophantine approximation which, in turn, arose, as far as Lovasz was concerned, from the need to eliminate the annoying full-dimensionality condition from the ellipsoid method in linear programming [6], [7].

2.5.1 First Algorithm: Rounding Off

Let $x = \sum_{i=1}^{l} \beta_i e_i$ and let α_i be the integer nearest to β_i. Set $w = \sum_{i=1}^{l} \alpha_i e_i$. If the basis B is Lovasz-reduced, then the ROUNDING OFF procedure finds a lattice point w, the nearest to x within a factor of $C = 1 + 2l(9/2)^{l/2}$.

2.5.2 Second Algorithm: Nearest Plane

Let $U = \sum_{i=1}^{d-1} Re_i$ be the linear subspace generated by $e_1, e_2, \ldots, e_{l-1}$ and let $L' = \sum_{i=1}^{d-1} Ze_i$ be the corresponding sublattice of L. Find $u \in L$ such that the distance between x and the affine subspace $U + u$ be minimal. Let x' denote the orthogonal projection of x on $U + u$. Recursively, find $y \in L'$ near $x' - u$. Let $w = y + u$. If the basis of E is Lovasz-reduced, then the NEAREST PLANE procedure finds a lattice point w, nearest to x within a factor of $C = 2^{d/2}$.

Chapter 3

Several Digital Signature Schemes

Every digital signature algorithm contains two parts. One is signature generation and the other is signature verification. Moreover, every message m of arbitrary length will be as the input of the hash function to product the data f such that $f \in Z_p$ for some integer q. We mainly concern on the following four digital signature schemes:

1. Digital Signature Algorithm (DSA)
2. Nyberg-Rueppel Signature Scheme
3. Elliptic Curve Analog of the DSA (EC_DSA)
4. Elliptic Curve Analog of the Nyberg-Rueppel Signature Scheme (EC_Nyberg-Rueppel SS)

3.1 DSA

In August of 1991, the US National Institute of Standards and Technology (NIST) proposed a digital signature algorithm (DSA) [18]. The DSA has become a U.S. Federal Information Processing Standard(FIPS 186) called the Digital Signature Standard(DSS). The algorithm is a variant of the ElGamal scheme which is based on the hardness of computing the discrete logarithm in some finite fields. and DSA is a digital signature with appendix.

Key Generation :

1. Select a prime number q such that $2^{159} < q < 2^{160}$.
2. Select a prime number p such that $p > 2^{511}$ and $(p-1)$ is divisible by q .
3. Select a generator α of the unique cyclic group of order q in Z_p^*
   Z_p^* means the nonzero elements in Z_p
4. Select a random e such that $1 \leq e \leq q-1$, e is the signer's private key.
5. Calculate $d = \alpha^e \pmod{p}$, d is the signer's public key .

Signature Generation:

- Input -
  i. The signer's private key e
  ii. The parameters p, α, q associated with e
     where p is the size of the field used
     $\alpha^q = 1 \pmod{p}$
  iii. The data, which is an integer f such that $0 \leq f < q$

- Output -

  The signature, which is a pair of integers (c, l), where $1 \leq c < q$ and $1 \leq l < q$. The signature (c, l) shall be computed by the following steps:

  1. Select a random integer k, $1 \leq k \leq q-1$
  2. Calculate $l = (\alpha^k \pmod{p}) \pmod{q}$
  3. Calculate $c = k^{-1}(f + el) \pmod{q}$. If $c = 0$, then go to step (1)

Signature Verification:

- Input -

  i. The signer's public key d

  ii. The parameters p, α ,q associated with d where p is the size of the field used

  $\alpha^q = 1 \pmod{p}$

  iii. The signature (c, l)

  iv. The data, which is an integer f such that $0 \leq f < q$

- Output -

  "Valid" or "Invalid" .

  The signature can be verified by the following steps :

  1. Check $1 \leq l < q$ and $1 \leq c < q$. If not, output "invalid".
  2. Compute $w = c^{-1} \pmod{q}$
  3. Compute $u_1 = wf \pmod{q}$ and $u_2 = lw \pmod{q}$
  4. Compute $v = (\alpha^{u_1} d^{u_2}) \pmod{q}$

5. Output "Valid" if and only if $v = l$

A random k must be selected for each message. Otherwise, the signer's private key can be recovered easily. For example, if two messages are signed with the same random k,

$$\begin{aligned} kc_1 &= (f_1 + el_1) \pmod q \\ kc_2 &= (f_2 + el_2) \pmod q \\ e &= \frac{kc_1 - f_1}{l_1} = \frac{kc_2 - f_2}{l_2} \pmod q \\ k &= \frac{f_1 l_2 - f_2 l_1}{c_1 l_2 - c_2 l_1} \pmod q \end{aligned}$$

k can be found from the above equation. When k is known, using the equation $e = \frac{kc_1 - f_1}{l_1} \pmod q$ can recover the private key.

3.2 Nyberg-Rueppel Digital Signature

Nyberg-Rueppel Digital Signature scheme is a variant of the basic ElGamal scheme with message recovery which relies on the difficulty of two related but distinct discrete logarithm problems.

Key Generation :

1. Select a prime number q
2. Select a prime number p such that $(p-1)$ is divisible by q .
3. Select a generator α of the unique cyclic group of order q in Z_p^*

   Z_p^* means the nonzero elements in Z_p
4. Select a random e such that $1 \leq e \leq q-1$, e is the signer's private key.

5. Calculate $d = \alpha^e \pmod{p}$, d is the signer's public key .

Signature Generation:

- Input -

    i. The signer's private key e

    ii. The parameters p, α ,q associated with e

    where p is the size of the field used

    $\alpha^q = 1 \pmod{p}$

    iii. The data, which is an integer f such that $0 \leq f < q$

- Output -

  The signature, which is a pair of integers (c, l), where $1 \leq c < q$ and $0 \leq l < q$. The signature (c, l) shall be computed by the following steps:

  1. Select a random integer k, $1 \leq k \leq q - 1$, and compute $r = \alpha^{-k} \pmod{p}$.
  2. Compute $c = fr \pmod{p}$. If $c = 0$, then go to step (1).
  3. Compute $l = ec + k \pmod{q}$

Signature Verification :

- Input -

    i. The signer's public key d

    ii. The parameters p, α, q associated with d

    where p is the size of the field used

    $\alpha^q = 1 \pmod{p}$

    iii. The signature (c, l)

- Output - Data f or "Invalid"

  The data can be recovered by the following steps:

  1. Check $1 \leq c \leq p$ and $0 \leq l < q$. If not, output "Invalid"
  2. Compute $v = \alpha^l d^{-c} \pmod{p}$ and $f = vc \pmod{p}$
  3. Output data f.

A random k must be selected for each message. Otherwise, the signer's private key can be recovered easily. For example, if two messages are signed with the same random k,

$$\begin{aligned} c_1 &= f_1 r_1 \pmod{q} \\ l_1 &= ec_1 + k \pmod{q} \\ e &= \frac{l_1 - k}{c_1} = \frac{l_2 - k}{c_2} \pmod{q} \\ k &= \frac{c_2 l_1 - c_1 l_2}{c_2 - c_1} \pmod{q} \end{aligned}$$

k can be found from the above equation. When k is known, using the equation $e = \frac{l_1 - k}{c_1} \pmod{q}$ can recover the private key.

3.3 EC_DSA

EC_DSA stands for elliptic curve analog of the DSA.

Key Generation:

1. Select a value of q such that q is a prime number or $q = 2^m$, m is a prime number
2. Select the parameters a, b of the elliptic curve $E(F_q)$ over F_q

(a) If q is a prime number, then the equation of the elliptic curve $E(F_q)$:

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

(b) If q is equal to 2^m, the equation of the elliptic curve $E(F_q)$:

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

3. Select a point G of a prime order r in $E(F_q)$. That is, $rG = \mathcal{O}$ (The identity element)

4. Select a random s such that $0 < s < r$. s is the signer's private key.

5. Calculate $V = sG$. V is the signer's public key.

Signature Generation :

- Input -

  i. The signer's private key s

  ii. The elliptic curve parameters q, a, b, r and G associated with s, where r is the order of the generator G

  iii. The data, which is an integer f such that $0 \le f < r$

- Output -

  The signature, which is a pair of integers (c, d), where $1 \le c < r$ and $1 \le d < r$.

  1. Generate a one-time key pair $(u, V = uG)$ with the same set of parameters as the private key s. Let $V = (x_v, y_v)(V \neq \mathcal{O}$ since V is the one time public key)

2. Convert x_v into an integer i
3. Compute an integer $c = i \pmod r$. If $c = 0$, then go to step 1.
4. Compute an integer $d = u^{-1}(f + sc) \pmod r$. If $d = 0$, then go to step 1.
5. Output the pair (c, d) as the signature.

Signature Verification:

- Input -
  - i. The signer's public key W
  - ii. The elliptic curve parameters q, a, b, r and G associated with W, where r is the order of the generator G
  - iii. The signature (c, d)
  - iv. The data f, which is an integer f such that $0 \le f < r$
- Output -

  "Valid" or "Invalid" .

  The signature can be verified by the following steps:

  1. Check $1 \le c < r$ and $1 \le d < r$. If not, output "invalid".
  2. Compute integers $h = d^{-1} \pmod r$, $h_1 = fh \pmod r$, and $h_2 = ch \pmod r$.
  3. Compute $P = h_1G + h_2W$. If $P = \mathcal{O}$, output "invalid" and stop. Otherwise, $P = (x_p, y_p)$.
  4. Convert x_p into an integer j
  5. $c' = j \pmod r$

6. If $c = c'$, then output "Valid". Otherwise, output "Invalid".

A random number u should be selected for every signature. Otherwise, the signer's private key can be recovered easily. For example, two messages are signed with the same random number u,

$$\begin{aligned} d_1 &= u^{-1}(f_1 + sc_1) \pmod r \\ s &= \frac{d_1 u - f_1}{c_1} = \frac{d_2 - f_2}{c_2} \pmod r \\ d_1 uc_2 - c_2 f_1 &= d_2 uc_1 - c_1 f_2 \pmod r \\ u &= \frac{c_2 f_1 - c_1 f_2}{c_2 - c_1} \pmod r \end{aligned}$$

u can be found from the above equation. When u is known, using the equation $s = \frac{ud_1 - f_1}{c_1} \pmod r$ can recover the private key.

3.4 EC_Nyberg-Rueppel Digital Signature Scheme

EC_Nyberg-Rueppel Digital Signature Scheme stands for elliptic curve analog to the Nyberg-Rueppel Digital Signature Scheme.
The key generation part is the same as the EC_DSA. Signature Generation:

- Input -

  i. The signer's private key s

  ii. The elliptic curve parameters q, a, b, r and G associated with s, where r is the order of the generator G

  iii. The data, which is an integer f such that $0 \leq f < r$

- Output -

  The signature, which is a pair of integers (c, d), where $1 \leq c < r$ and $0 \leq d < r$.

  The signature (c, d) should be computed by the following steps:

  1. Generate a one-time key pair $(u, V = uG)$ with the same set of parameters as the private key s. Let $V = (x_v, y_v)(V \neq \mathcal{O}$ because V is a public key).
  2. Convert x_v into an integer i
  3. Compute an integer $c = i + f \pmod{r}$. If $c = 0$, then go to step (1).
  4. Compute an integer $d = u - sc \pmod{r}$.
  5. Output the pair (c, d) as the signature .

Signature Verification:

- Input -

  i. The signer's public key W

  ii. The elliptic curve parameters q, a, b, r and G associated with W, where r is the order of the generator G

  iii. The signature (c, d)

- Output -

  Data f or "Invalid" .

  The data can be recovered by the following steps :

1. Check $1 \leq c < r$ and $0 \leq d < r$. If not, output "Invalid".
2. Compute $P = dG+cW$. If $P = \mathcal{O}$, output "Invalid". Let $P = (x_p, y_p)$
3. Convert x_p into an integer j
4. Compute an integer $f = c - j \pmod{r}$
5. Output f as the data

A random u should be selected for every signature. Otherwise, the signer's private key can be recovered easily. For example, two messages are signed with the same random u,

$$\begin{aligned}
d_1 &= u - sc_1 \pmod{r} \\
d_2 &= u - sc_1 \pmod{r} \\
s &= \frac{u - d_1}{c_1} = \frac{u - d_2}{c_2} \pmod{r} \\
uc_2 - d_1c_2 &= c_1u - c_1d_2 \pmod{r} \\
u &= \frac{d_1c_2 - d_2c_1}{c_2 - c_1} \pmod{r}
\end{aligned}$$

u can be found from the above equation. When u is known, using the equation $s = \frac{u-d_1}{c_1} \pmod{r}$ can recover the private key.

Chapter 4

Miscellaneous Digital Signature Schemes and their PRNG

In this chapter, a simple lattice based algorithm will be used to solve a system of linear equations in different moduli and apply in different digital signature schemes with different random number generators. This algorithm using the nearest lattice vector approximation algorithm as a subroutine.

Lattice: Given a set of l linear independent vectors in $\mathbf{R}^l$, $E = \{e_1, \ldots, e_l\}$, we define the lattice spanned by E(the basis of the lattice) be the set of all the possible linear combinations of e_i's with integral coefficients, that is:

$$L(E) = \{\sum_{i=1}^{l} c_i e_i \text{ such that } c_i \in Z \quad \forall i\}$$

Given E and a vector $x \in \mathbf{R}^l$ not in $L(E)$, the nearest lattice vector problem asks for a lattice vector $w \in L(E)$ such that $||w - x|| = min_{v \in L(E)} ||v - x||$.

In [4], Babai gave a simple polynomial time algorithm to find an appropriate solution to the nearest lattice vector problem : given the basis E and the target vector x, Babai's algorithm returns a lattice vector w such that $||w - x|| \leq c \quad min_{v \in L(E)} ||v - x||$, where $c = 2^{l/2}$ is an approximation factor depending only on the dimension of the lattice.

4.1 DSA with LCG

[17] suggests a method to solve the simulaneous modular equations and use this method to find out the secret key in the DSA with LCG case. In this section, we will recall [17] result.

Let $U_1, \ldots U_n$ be positive integers and let V_u be the set of vectors $\{\vec{x} \in Z^n | \quad \forall i |x_i| < U_i\}$. Let also $A = \{a_{i,j}\}$ be an two vectors in Z^m. We want to find an integer vector $\overrightarrow{x} \in V_U$ such that $A\overrightarrow{x} = \overrightarrow{b} \pmod{\overrightarrow{M}}$, i.e., $|x_i| < U_i$ for all $i = 1, \ldots n$ and the following modular equations are simultaneously satified.

$$\begin{aligned}
a_{1,1}x_1 + \ldots + a_{1,n}x_n &= b_1 \pmod{M_1} \\
a_{2,1}x_1 + \ldots + a_{2,n}x_n &= b_2 \pmod{M_2} \\
&\vdots \\
a_{m,1}x_1 + \ldots + a_{m,n}x_n &= b_n \pmod{M_n}
\end{aligned}$$

We first assume that the above system has a solution x and that a good approximation to this solution is known and devise a method to find the exact solution.

Definition : Let $\overrightarrow{x}$ and $\overrightarrow{y}$ be two vectors in V_U. We say that vector $\overrightarrow{y}$ c-approximates $\overrightarrow{x}$ iff for all $i = 1, \ldots, n$ we have $|x_i - y_i| < (U_i - |y_i|)/(c\sqrt{n})$.

Lemma : Let c be a constant greater than $2^{(m+n)/2}$. There exists a polynomial time algorithm that on input $U_1, \ldots, U_n$, A, $\overrightarrow{b}, \overrightarrow{M}$ as above and a c-approximation $\overrightarrow{y}$ to a solution $\overrightarrow{x} \in V_U$ to $A \cdot \overrightarrow{x} = \overrightarrow{b} \pmod{\overrightarrow{M}}$, finds a solution $\overrightarrow{w} \in V_U$ to $Adots\overrightarrow{x} = \overrightarrow{b} \pmod{\overrightarrow{M}}$.

Proof : Let $\Gamma = \{\gamma_{i,j}\}$ be the n $\times$ n diagonal matrix defined by $\gamma_{i,i} = \frac{1}{U_i - |y_i|}$ and let M be the m $\times$ diagonal matrix whose diagonal entries are $M_1, \ldots, M_m$. Consider the lattice generated by the columns of the matrix $L = \begin{bmatrix} A & M \\ \Gamma & 0 \end{bmatrix}$ and define the vectors

$$\overrightarrow{X} = \begin{bmatrix} \overrightarrow{b} \\ \Gamma \cdot \overrightarrow{x} \end{bmatrix}$$

$$\overrightarrow{Y} = \begin{bmatrix} \overrightarrow{b} \\ \Gamma \cdot \overrightarrow{y} \end{bmatrix}$$

Notice that $\overrightarrow{X}$ is a lattice vector and

$$\| \overrightarrow{X} - \overrightarrow{Y} \| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2 \gamma_{i,i}^2} \leq \sqrt{\sum_{i=1}^{n} \frac{1}{c^2 n}} = \frac{1}{c}$$

Running Babai's nearest lattice vector alogrithm on lattice E and target vector Y we obtain a lattice vector $\overrightarrow{W}$ such that $\| \overrightarrow{W} - \overrightarrow{Y} \| < c \| \overrightarrow{X} - \overrightarrow{Y} \| \leq 1$. Since the first m elements of $\overrightarrow{W}$ and $\overrightarrow{Y}$ are integers, they must be the same. So, the vector $\overrightarrow{W}$ is equal to $\begin{bmatrix} \overrightarrow{b} \\ \Gamma \cdot \overrightarrow{w} \end{bmatrix}$

for some integer vector $\overrightarrow{w}$ satisfying $A \cdot \overrightarrow{w} = \overrightarrow{b} \pmod{\overrightarrow{M}}$.

Now, for all $i = 1, \cdots, n$ we have

$$(w_i - y_i)^2 \gamma_{i,i}^2 \leq \sum_i (w_i - y_i)^2 \gamma_{i,i}^2 = \| \overrightarrow{W} - \overrightarrow{Y} \|^2 < 1$$

so that $|w_i - y_1| < 1/\gamma_{i,i} = U_i - |y_i|$ and by triangular inequality

$$|w_i| \leq |y_i| + |w_i - y_i| < |y_i| + U_i - |y_i| = U_i$$

Therefore, $\overrightarrow{w} \in V_U$.

For the DSA with LCG case, we combine the two signature equations and with the LCG equation $k_2 = ak_1 + b \pmod{M}$. The following equations will be given out.

$$\begin{aligned} c_1 k_1 &= f_1 + el_1 \pmod{q} \\ c_2 k_2 &= f_2 + el_2 \pmod{q} \\ k_2 &= ak_1 + b \pmod{M} \end{aligned}$$

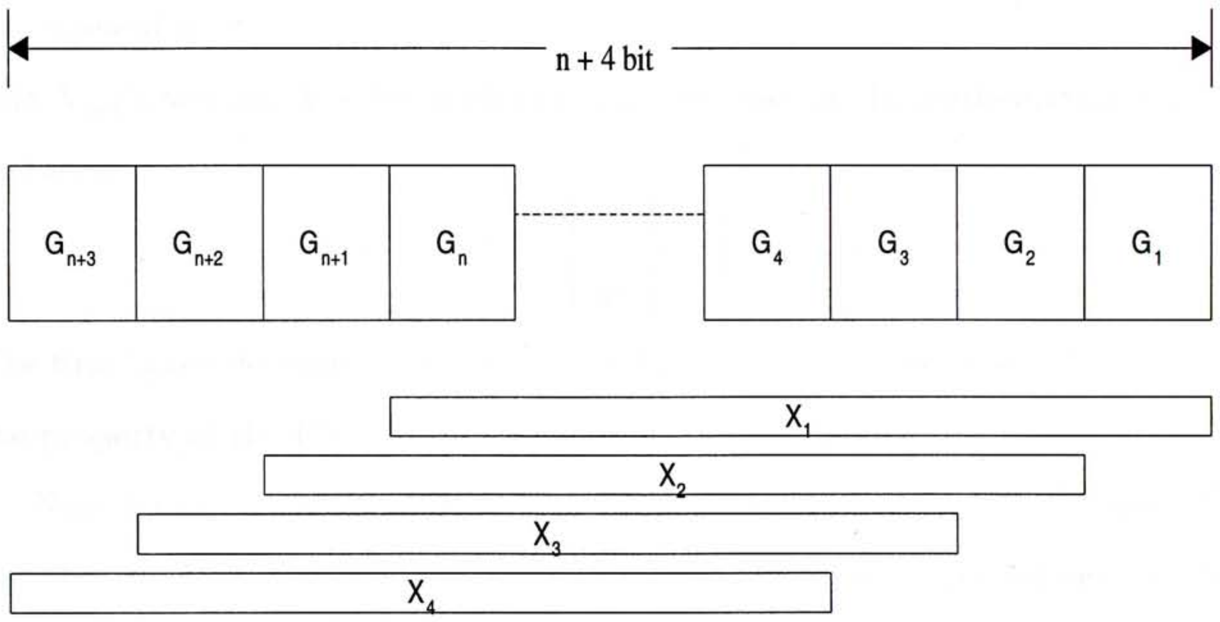
The lattice L should be generated by the columns of the matrix E.

$$\mathbf{E} = \begin{bmatrix} -l_1 & c_1 & 0 & q & 0 & 0 \\ -l_2 & 0 & c_2 & 0 & q & 0 \\ 0 & -a & 1 & 0 & 0 & M \\ r_e^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{k_1}^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{k_2}^{-1} & 0 & 0 & 0 \end{bmatrix}.$$

Using the procedures that described above can find out the secret key easily.

4.2 DSA with PN-sequence

As stated in the previous section, DSA need a random nonce k for each message to sign. In this part, we assume the random nonce is produced by the PN-sequence generator. Fig 4.2 shows an example of 4 random numbers are generated by the PN sequenced generator.



PN-sequence generates n + 3 bits, i.e. G_1 , G_2, ..., G_{n+3} to produce 4 random number, i.e. X_1, X_2, X_3 and X_4 and each random number contains n bits and has the following format:

$$\begin{aligned} X_1 &= G_n G_{n-1} G_{n-2} \ldots G_2 G_1 \\ X_2 &= G_{n+1} G_n G_{n-1} \ldots G_3 G_2 \\ X_3 &= G_{n+2} G_{n+1} G_n \ldots G_4 G_3 \\ X_4 &= G_{n+3} G_{n+2} G_{n+1} \ldots G_5 G_4 \end{aligned}$$

In general, X_i and X_{i+1} has the following relationship:

$$\begin{aligned} X_i &= G_{i_1}G_{i_2}G_{i_3}\ldots G_{i_{n-1}}G_{i_n} \\ X_{i+1} &= *G_{i_1}G_{i_2}\ldots G_{i_{n-2}}G_{i_{n-1}} \end{aligned}$$

* stands for $G_{{i+1}_1}$. Since this value is unknown when X_i is generated, we use * to represent it.

The X_{i+1}'s bits are X_i's bits shifted to right by one bit. In mathematical representation :

$$X_{i+1} = \frac{X_i}{2} - \left\{ \begin{array}{c} \frac{1}{2} \\ 0 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 2^{n-1} \end{array} \right\}$$

The first brace depends on the values of G_{i_n} and the second brace depends on the property of the PN-sequence generator.

Now, let us consider the DSA with PN-sequence in mathematical approach. For the ith data, the random nonce is k_i. So the following equations can be obtained.

$$\begin{aligned} l_i &= (\alpha^{k_i} \pmod p) \pmod q \\ c_i &= k_i^{-1}(f_i + el_i) \pmod q \\ k_{i+1} &= \frac{k_i}{2} - \left\{ \begin{array}{c} \frac{1}{2} \\ 0 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 2^{n-1} \end{array} \right\} \end{aligned}$$

Now, we consider two signatures are received by an adversary. That is the adversary knows c_1, c_2, l_1, l_2, f_1 and f_2.

$$\begin{aligned} c_1k_1 &= f_1 + el_1 \pmod q \\ c_2k_2 &= f_2 + el_2 \pmod q \end{aligned}$$

$$k_2 = \frac{k_1}{2} - \left\{ \begin{array}{c} \frac{1}{2} \\ 0 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 2^{n-1} \end{array} \right\}$$

We change f_1 and f_2 as object of the equations.

$$\begin{aligned} f_1 &= c_1 k_1 - e l_1 \pmod q && (4.1) \\ f_2 &= c_2 k_2 - e l_2 \pmod q && (4.2) \\ -k_1 + 2k_2 &= -\left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 2^n \end{array} \right\} && (4.3) \end{aligned}$$

Solving these equations can obtain the secret key e. However, these equations with difference moduli cause they difficult to solve. Using Babai's algorithm to find out the solution.

4.2.1 Solution

We need to set up a lattice L which is generated by the columns of the matrix $\mathbf{E}$ to find out the solution of secret key.

$$\mathbf{E} = \begin{bmatrix} -l_1 & c_1 & 0 & q & 0 & 0 \\ -l_2 & 0 & c_2 & 0 & q & 0 \\ 0 & -1 & 2 & 0 & 0 & 2^n \\ r_e^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{k_1}^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{k_2}^{-1} & 0 & 0 & 0 \end{bmatrix}$$

Let $e' = \frac{q}{2}$, $k_1' = k_2' = \frac{M}{2}$, $r_e = \min\{e', q - e'\}$, $r_{k_1} = \min\{k_1', M - k_1'\}$, $r_{k_2} = \min\{k_2', M - k_2'\}$ and lattice vector $\mathbf{l_1}$, $\mathbf{l_2}$, $\mathbf{l_3}$ and $\mathbf{l_4}$.

$$\begin{aligned}
\mathbf{I_1} &= (f_1, f_2, 0, \frac{e}{r_e}, \frac{k_1}{r_{k_1}}, \frac{k_2}{r_{k_2}})^T \\
\mathbf{I_2} &= (f_1, f_2, -1, \frac{e}{r_e}, \frac{k_1}{r_{k_1}}, \frac{k_2}{r_{k_2}})^T \\
\mathbf{I_3} &= (f_1, f_2, 2^n, \frac{e}{r_e}, \frac{k_1}{r_{k_1}}, \frac{k_2}{r_{k_2}})^T \\
\mathbf{I_4} &= (f_1, f_2, 2^n - 1, \frac{e}{r_e}, \frac{k_1}{r_{k_1}}, \frac{k_2}{r_{k_2}})^T
\end{aligned}$$

Let i be 1, 2, 3 or 4. For this particular i, $\mathbf{I_i}$ can be obtained by the following steps :

1. The first three columns of the matrix $\mathbf{E}$ are multipled by e, k_1, k_2 respectively.
2. The remaining columns(the last 3 columns) are subtracted by appropriate multiples to perform modular reduction.

So, the secret key e can be recovered by lattice vector $\mathbf{I_i}$ easily.
Running Babai's nearest lattice vector algorithm on lattice $L(E)$ and target vector.

$$\begin{aligned}
\mathbf{J_1} &= (f_1, f_2, 0, \frac{e'}{r_e}, \frac{k_1'}{r_{k_1}}, \frac{k_2'}{r_{k_2}})^T \\
\mathbf{J_2} &= (f_1, f_2, -1, \frac{e'}{r_e}, \frac{k_1'}{r_{k_1}}, \frac{k_2'}{r_{k_2}})^T \\
\mathbf{J_3} &= (f_1, f_2, 2^n, \frac{e'}{r_e}, \frac{k_1'}{r_{k_1}}, \frac{k_2'}{r_{k_2}})^T \\
\mathbf{J_4} &= (f_1, f_2, 2^n - 1, \frac{e'}{r_e}, \frac{k_1'}{r_{k_1}}, \frac{k_2'}{r_{k_2}})^T
\end{aligned}$$

We obtain lattice vector $\mathbf{N}$ such that $||\mathbf{J_i} - \mathbf{N_i}|| < \mathbf{8}||\mathbf{J_i} - \mathbf{I_i}||$. Now, if $| e - e' | < \frac{r_e}{14}$, $| k_1 - k_1' | < \frac{r_{k_1}}{14}$, $| k_2 - k_2' | < \frac{r_{k_2}}{14}$, then $||\mathbf{J_i} - \mathbf{N_i}|| < \mathbf{1}$ and since the first three entries of $\mathbf{N_i}$ are integers, they must coincide with the corresponding entries in $\mathbf{J_i}$ and we have

$$\begin{aligned}
\mathbf{N_1} &= (f_1, f_2, 0, \frac{e''}{r_e}, \frac{k_1''}{r_{k_1}}, \frac{k_2''}{r_{k_2}})^T \\
\mathbf{N_2} &= (f_1, f_2, -1, \frac{e''}{r_e}, \frac{k_1''}{r_{k_1}}, \frac{k_2''}{r_{k_2}})^T \\
\mathbf{N_3} &= (f_1, f_2, 2^n, \frac{e''}{r_e}, \frac{k_1''}{r_{k_1}}, \frac{k_2''}{r_{k_2}})^T \\
\mathbf{N_4} &= (f_1, f_2, -1 + 2^n, \frac{e''}{r_e}, \frac{k_1''}{r_{k_1}}, \frac{k_2''}{r_{k_2}})^T
\end{aligned}$$

for some e'', k_1'', k_2'' satisfy the equations 4.1, 4.2 and 4.3. Moreover,

$$\begin{aligned}
e'' &= e'' - e' + e' \geq -r_e + r_e = 0 \\
e'' &= e'' - e' + e' < r_e + (q - r_e) = q \\
k_1'' &= k_1'' - k_1' + k_1' \geq -r_{k_1} + r_{k_1} = 0 \\
k_1'' &= k_1'' - k_1' + k_1' < r_{k_1} + (q - r_{k_1}) = M \\
k_2'' &= k_2'' - k_2' + k_2' \geq -r_{k_2} + r_{k_2} = 0 \\
k_2'' &= k_2'' - k_2' + k_2' < r_{k_2} + (q - r_{k_2}) = M
\end{aligned}$$

So,

$$\begin{aligned}
0 &\leq e'' < q \\
0 &\leq k_1'' < M \\
0 &\leq k_2'' < M
\end{aligned}$$

If the vector **N** does not have the desired form, this implies that the initial guess (e', k_1', k_2') was not so good. At this point, we need to change the value of initial guess. Change the initial guess by the following steps:

1. Let $j = 1$
2. $e' = q/2 + (1 - (1 - \frac{1}{8})^j q/2))$
3. $k_1' = M/2 + (1 - (1 - \frac{1}{8})^j M/2))$
4. $k_2' = M/2 + (1 - (1 - \frac{1}{8})^j M/2))$
5. Run Babai's algorithm. Check the output **N** if **N** is the desired form or not.
    - If yes, output **N**.
    - If no, check e', k_1' and k_2'.
        - If $e' \le q$, $k_1' \le M$ or $k_2' \le M$, then $j = j + 1$ and go to step (2) .
        - Otherwise, go to step(6).
6. Let $j = 1$
7. $e' = q/2 - (1 - (1 - \frac{1}{8})^j q/2))$
8. $k_1' = M/2 - (1 - (1 - \frac{1}{8})^j M/2))$
9. $k_2' = M/2 - (1 - (1 - \frac{1}{8})^j M/2))$
10. Run Babai's algorithm. Check the output **N** if **N** is the desired form. If yes, output **N**. Otherwise,$j = j + 1$ and go to step (7) .

Once we have found a solution e', k_1' and k_2' to the equations 4.1, 4.2 and 4.3, we can check that we actually found the secret key e by using the unsed formula, $d = \alpha^e \pmod p$. Comparing $d' = \alpha^{e'} \pmod p$ with d. If they are not equal, repeat the subroutine with different initial guess (e', k_1', k_2'). So the secret key e can be found in this way.

4.3 DSA with ICG

In this part, we assume the DSA uses ICG (inversive congruential generator) for generating the random nonce k. Therefore, the following equations will be obtained.

$$\begin{aligned} k_{i+1} &= a\bar{k_i} + b \pmod M \\ k_2 &= a\bar{k_1} + b \pmod M \\ c_i &= k_i^{-1}(f_i + el_i) \pmod q \\ c_1k_1 &= f_1 + el_1 \pmod q \\ c_2k_2 &= f_2 + el_2 \pmod q \end{aligned}$$

We change f_1 and f_2 as objects of the equations.

$$\begin{aligned} f_1 &= c_1k_1 - el_1 \pmod q && (4.4) \\ f_2 &= c_2k_2 - el_2 \pmod q && (4.5) \\ k_2k_1 &= a + bk_1 \pmod M && (4.6) \end{aligned}$$

Solving these equations can obtain the secret key e. However, these equations with difference moduli cause they difficult to solve. Using Babai's algorithm to find out the solution.

4.3.1 Solution

We need to set up a lattice L which is generated by the columns of the matrix $\mathbf{E}$ to find out the solution of secret key.. The lattice is generated by the columns of the matrix.

$$\mathbf{E} = \begin{bmatrix} -l_1 & c_1 & 0 & 0 & q & 0 & 0 \\ -l_2 & 0 & c_2 & 0 & 0 & q & 0 \\ 0 & -b & 0 & 1 & 0 & 0 & M \\ r_e^{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{k_1}^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{k_2}^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{k_1k_2}^{-1} & 0 & 0 & 0 \end{bmatrix}$$

Let $e' = \frac{q}{2}$, $k_1' = k_2' = \frac{M}{2}$, $r_e = \min\{e', q-e'\}$, $r_{k_1} = \min\{k_1', M-k_1'\}$, $r_{k_2} = \min\{k_2', M-k_2'\}$ and lattice vector $\mathbf{I}$

$$\mathbf{I} = (f_1, f_2, a, \frac{e}{r_e}, \frac{k_1}{r_{k_1}}, \frac{k_2}{r_{k_2}}, \frac{k_1k_1}{r_{k_1k_2}})^T$$

$\mathbf{I}$ can be obtained by the following steps :

1. The first four columns of the matrix $\mathbf{E}$ are multipled by e, k_1, k_2, k_1k_2 respectively.

2. The remaining columns(the last 3 columns) are subtracted by appropriate multiples to perform modular reduction.

So, the secret key e can be recovered by lattice vector $\mathbf{I}$ easily.

Running Babai's nearest lattice vector algorithm on lattice $L(E)$ and target vector.

$$\mathbf{J} \;=\; (f_1, f_2, a, \frac{e'}{r_e}, \frac{k_1'}{r_{k_1}}, \frac{k_2'}{r_{k_2}}, \frac{k_1 k_2'}{r_{k_1 k_2}})^T$$

We obtain lattice vector $\mathbf{N}$ such that $||\mathbf{J} - \mathbf{N}|| < \mathbf{2^{7/2}}||\mathbf{J} - \mathbf{I}||$. Now, if $| e - e' | < \frac{r_e}{23}$, $| k_1 - k_1' | < \frac{r_{k_1}}{23}$, $| k_2 - k_2' | < \frac{r_{k_2}}{23}$, then $||\mathbf{J} - \mathbf{N}|| < \mathbf{1}$ and since the first three entries of $\mathbf{N}$ are integers, they must coincide with the corresponding entries in $\mathbf{J}$ and we have

$$\mathbf{N} \;=\; (f_1, f_2, a, \frac{e''}{r_e}, \frac{k_1''}{r_{k_1}}, \frac{k_2''}{r_{k_2}}, \frac{k_1 k_2''}{r_{k_1 k_2}})^T$$

for some e'', k_1'', k_2'' satisfy the equations 4.4, 4.5 and 4.6. Moreover,

$$\begin{aligned}
e'' &= e'' - e' + e' \geq -r_e + r_e = 0 \\
e'' &= e'' - e' + e' < r_e + (q - r_e) = q \\
k_1'' &= k_1'' - k_1' + k_1' \geq -r_{k_1} + r_{k_1} = 0 \\
k_1'' &= k_1'' - k_1' + k_1' < r_{k_1} + (q - r_{k_1}) = M \\
k_2'' &= k_2'' - k_2' + k_2' \geq -r_{k_2} + r_{k_2} = 0 \\
k_2'' &= k_2'' - k_2' + k_2' < r_{k_2} + (q - r_{k_2}) = M
\end{aligned}$$

So,

$$\begin{aligned}
0 &\leq e'' < q \\
0 &\leq k_1'' < M \\
0 &\leq k_2'' < M
\end{aligned}$$

If the vector **N** does not have the desired form, this implies that the initial guess (e', k_1', k_2') was n ot so good. At this point, we need to change the value of initial guess. Change the initial guess by the following steps:

1. Let $j = 1$
2. $e' = q/2 + (1 - (1 - \frac{1}{8})^j q/2))$
3. $k_1' = M/2 + (1 - (1 - \frac{1}{8})^j M/2))$
4. $k_2' = M/2 + (1 - (1 - \frac{1}{8})^j M/2))$
5. Run Babai's algorithm. Check the output **N** if **N** is the desired form or not.
    - If yes, output **N**.
    - If no, check e', k_1' and k_2'.
        - If $e' \leq q$, $k_1' \leq M$ or $k_2' \leq M$, then $j = j + 1$ and go to step (2) .
        - Otherwise, go to step(6).
6. Let $j = 1$
7. $e' = q/2 - (1 - (1 - \frac{1}{8})^j q/2))$
8. $k_1' = M/2 - (1 - (1 - \frac{1}{8})^j M/2))$
9. $k_2' = M/2 - (1 - (1 - \frac{1}{8})^j M/2))$
10. Run Babai's algorithm. Check the output **N** if **N** is the desired form. If yes, output **N**. Otherwise, check $e' \leq q$, $k_1' \leq M$ or $k_2' \leq M$, then j = j+1 and go to step (7) .

Once we have found a solution e', k_1' and k_2' to the equations 4.1, 4.2 and 4.3, we can ch eck that we actually found the secret key e by using the unsed formula, $d = \alpha^e \pmod p$. Comparing $d' = \alpha^{e'} \pmod p$ with d. If they are not equal, repeat the subroutine with different initial guess (e', k_1', k_2'). So the secret key e can be found in this way.

4.4 EC_DSA with PN-sequence

In this part, we assume the EC_DSA uses PN-sequence to generate the random nonce u. Therefore, the following equations will be obtained.

$$\begin{aligned}
d_i &= u_i^{-1}(f_i + sc_i) \pmod r \\
u_{i+1} &= \frac{u_i}{2} - \left\{ \begin{array}{c} \frac{1}{2} \\ 0 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 2^{n-1} \end{array} \right\} \\
d_1 u_1 &= f_1 + sc_1 \pmod r \\
d_2 u_2 &= f_2 + sc_1 \pmod r \\
u_2 &= \frac{u_1}{2} - \left\{ \begin{array}{c} \frac{1}{2} \\ 0 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 2^{n-1} \end{array} \right\}
\end{aligned}$$

We change f_1 and f_2 as object of the equations.

$$\begin{aligned}
f_1 &= d_1 u_1 - sc_1 \pmod r \\
f_2 &= d_2 u_2 - sc_2 \pmod r \\
-u_1 + 2u_2 &= \left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 2^n \end{array} \right\}
\end{aligned}$$

Solving these equations can obtain the secret key d. However, these equations with difference moduli cause they difficult to solve. Again, using Babai's algorithm to find out the solution.

4.4.1 Solution

We need to set up a lattice L which is generated by the columns of the matrix $\mathbf{E}$ to find out the solution of secret key..

$$\mathbf{E} = \begin{bmatrix} -c_1 & d_1 & 0 & r & 0 & 0 \\ -c_2 & 0 & d_2 & 0 & r & 0 \\ 0 & -1 & 2 & 0 & 0 & 2^n \\ r_s^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{u_1}^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{u_2}^{-1} & 0 & 0 & 0 \end{bmatrix}$$

Multiple the first column of the matrix by s, the second column by u_1, the third column of u_2 and subtracting the appropriate multiples of the remaining columns to perform modular reduction. After these operations, the following lattice vector will be obtained.

$$\mathbf{I} = (f_1, f_2, -\left\{\begin{matrix} 0 \\ 1 \end{matrix}\right\} + \left\{\begin{matrix} 0 \\ 2^n \end{matrix}\right\}, \frac{s}{r_s}, \frac{u_1}{r_{u_1}}, \frac{u_2}{r_{u_2}})$$

Using this lattice vector we can recover the secret key s easily.

The procedures for finding out the solution of secret key s is similar to the previous sections.

4.5 EC_DSA with LCG

In this part, we assume the EC_DSA uses LCG to generate the random nonce u. Therefore, the following equations will be obtained.

$$\begin{aligned} d_i &= u_i^{-1}(f_i + sc_i) \pmod r \\ u_{i+1} &= au_i + b \pmod M \\ d_1u_1 &= f_1 + sc_1 \pmod r \\ d_2u_2 &= f_2 + sc_1 \pmod r \\ u_2 &= au_1 + b \pmod M \end{aligned}$$

We change f_1 and f_2 as the object of the equations.

$$\begin{aligned} f_1 &= d_1u_1 - sc_1 \pmod r \\ f_2 &= d_2u_2 - sc_2 \pmod r \\ b &= u_2 - au_1 \pmod M \end{aligned}$$

Solving these equations can obtain the secret key s. However, these equations with difference moduli cause they difficult to solve. Again, using Babai's algorithm to find out the solution.

4.5.1 Solution

We need to set up a lattice L which is generated by the columns of the matrix $\mathbf{E}$ to find out the solution of secret key..

$$\mathbf{E} = \begin{bmatrix} -c_1 & d_1 & 0 & r & 0 & 0 \\ -c_2 & 0 & d_2 & 0 & r & 0 \\ 0 & -a & 1 & 0 & 0 & M \\ r_s^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{u_1}^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{u_2}^{-1} & 0 & 0 & 0 \end{bmatrix}$$

Multiple the first column of the matrix by s, the second column by u_1, the third column of u_2 and subtracting the appropriate multiples of the remaining columns to perform modular reduction. After these operations, the following lattice vector will be obtained.

$$\mathbf{I} = (f_1, f_2, b, \frac{s}{r_s}, \frac{u_1}{r_{u_1}}, \frac{u_2}{r_{u_2}})$$

Using this lattice vector we can recover the secret key s easily.

The procedures for finding out the solution of secret key s is similar to the previous sections.

4.6 EC_DSA with ICG

In this part, we assume the EC_DSA uses ICG to generate the random nonce u. Therefore, the following equations will be obtained.

$$d_i = u_i^{-1}(f_i + sc_i) \pmod r$$

$$\begin{aligned} u_{i+1} &= a\bar{u}_i + b \pmod{M} \\ d_1 u_1 &= f_1 + sc_1 \pmod{r} \\ d_2 u_2 &= f_2 + sc_1 \pmod{r} \\ u_2 u_1 &= a + bu_1 \pmod{M} \end{aligned}$$

We change f_1 and f_2 as the objects of the equations.

$$\begin{aligned} f_1 &= d_1 u_1 - sc_1 \pmod{r} \\ f_2 &= d_2 u_2 - sc_2 \pmod{r} \\ a &= u_2 u_1 - bu_1 \pmod{M} \end{aligned}$$

Solving these equations can obtain the secret key d. However, these equations with difference moduli cause they difficult to solve. Again, using Babai's algorithm to find out the solution.

4.6.1 Solution

We need to set up a lattice L which is generated by the columns of the matrix $\mathbf{E}$ to find out the solution of secret key..

$$\mathbf{E} = \begin{bmatrix} -c_1 & d_1 & 0 & 0 & r & 0 & 0 \\ -c_2 & 0 & d_2 & 0 & 0 & r & 0 \\ 0 & -b & 0 & 1 & 0 & 0 & M \\ r_s^{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{u_1}^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{u_2}^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{u_1 u_2}^{-1} & 0 & 0 & 0 \end{bmatrix}.$$

Multiple the first column of the matrix by s, the second column by u_1, the third column of u_2 and subtracting the appropriate multiples of the remaining columns to perform modular reduction. After these operations, the following lattice vector will be obtained.

$$\mathbf{I} = (f_1, f_2, a, \frac{s}{r_s}, \frac{u_1}{r_{u_1}}, \frac{u_2}{r_{u_2}}, \frac{u_1u_2}{r_{u_1u_2}})$$

Using this lattice vector we can recover the secret key s easily.

The procedures for finding out the solution of secret key s is similar to the previous sections.

4.7 Nyberg-Rueppel Digital Signature with PN-sequence

In this part, we assume the Nyberg-Rueppel Digital Signature uses PN-sequence to generate random nonce k. Therefore, the following equations will be obtained.

$$\begin{aligned}
l_i &= dc_i + k_i \pmod q \\
k_{i+1} &= \frac{k_i}{2} - \left\{ \begin{array}{c} \frac{1}{2} \\ 0 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 2^{n-1} \end{array} \right\} \\
l_1 &= dc_1 + k_1 \pmod q \\
l_2 &= dc_2 + k_2 \pmod q \\
-k_1 + 2k_2 &= -\left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 2^n \end{array} \right\}
\end{aligned}$$

Solving these equations can obtain the secret key d. However, these equations with difference moduli cause they difficult to solve. Again, using Babai's algorithm to find out the solution.

4.7.1 Solution

We need to set up a lattice L which is generated by the columns of the matrix $\mathbf{E}$ to find out the solution of secret key..

$$\mathbf{E} = \begin{bmatrix} c_1 & 1 & 0 & q & 0 & 0 \\ c_2 & 0 & 1 & 0 & q & 0 \\ 0 & -1 & 2 & 0 & 0 & 2^n \\ r_d^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{k_1}^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{k_2}^{-1} & 0 & 0 & 0 \end{bmatrix}$$

Multiple the first column of the matrix by d, the second column by k_1, the third column of k_2 and subtracting the appropriate multiples of the remaining columns to perform modular reduction. After these operations, the following lattice vector will be obtained.

$$\mathbf{I} = (l_1, l_2, -\left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\} + \left\{ \begin{matrix} 0 \\ 2^n \end{matrix} \right\}, \frac{s}{r_s}, \frac{k_1}{r_{k_1}}, \frac{k_2}{r_{k_2}})$$

Using this lattice vector we can recover the secret key d easily.

The procedures for finding out the solution of secret key d is similar to the previous sections.

4.8 Nyberg-Rueppel Digital Signature with LCG

In this part, we assume the Nyberg-Rueppel Digital Signature uses LCG to generate random nonce k. Therefore, the following equations will be obtained.

$$\begin{aligned}
l_i &= dc_i + k_i \pmod q \\
k_{i+1} &= ak_i + b \pmod M \\
l_1 &= dc_1 + k_1 \pmod q \\
l_2 &= dc_2 + k_2 \pmod q \\
k_2 &= ak_1 + b \pmod M
\end{aligned}$$

Solving these equations can obtain the secret key d. However, these equations with difference moduli cause they difficult to solve. Again, using Babai's algorithm to find out the solution.

4.8.1 Solution

We need to set up a lattice L which is generated by the columns of the matrix $\mathbf{E}$ to find out the solution of secret key.

$$\mathbf{E} = \begin{bmatrix} c_1 & 1 & 0 & q & 0 & 0 \\ c_2 & 0 & 1 & 0 & q & 0 \\ 0 & -a & 1 & 0 & 0 & M \\ r_d^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{k_1}^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{k_2}^{-1} & 0 & 0 & 0 \end{bmatrix}$$

Multiple the first column of the matrix by d, the second column by k_1, the third column of k_2 and subtracting the appropriate multiples of the remaining columns to perform modular reduction. After these operations, the following lattice vector will be obtained.

$$\mathbf{l} = (l_1, l_2, b, \frac{s}{r_s}, \frac{k_1}{r_{k_1}}, \frac{k_2}{r_{k_2}})$$

Using this lattice vector we can recover the secret key d easily.

The procedures for finding out the solution of secret key s is similar to the previous sections.

4.9 Nyberg-Rueppel Digital Signature with ICG

In this part, we assume the Nyberg-Rueppel Digital Signature uses ICG to generate random nonce k. Therefore, the following equations will be obtained.

$$\begin{aligned}
l_i &= dc_i + k_i \pmod q \\
k_{i+1} &= a\bar{k_i} + b \pmod M \\
l_1 &= dc_1 + k_1 \pmod q \\
l_2 &= dc_2 + k_2 \pmod q \\
k_2 &= a\bar{k_1} + b \pmod M \\
k_2 k_1 &= a + bk_1 \pmod M
\end{aligned}$$

Solving these equations can obtain the secret key d. However, these equations with difference moduli cause they difficult to solve. Again, using Babai's algorithm to find out the solution.

4.9.1 Solution

We need to set up a lattice L which is generated by the columns of the matrix $\mathbf{E}$ to find out the solution of secret key.

$$\mathbf{E} = \begin{bmatrix} c_1 & 1 & 0 & 0 & q & 0 & 0 \\ c_2 & 0 & 1 & 0 & 0 & q & 0 \\ 0 & -b & 0 & 1 & 0 & 0 & M \\ r_d^{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{k_1}^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{k_2}^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & r_{k_1 k_2}^{-1} & 0 & 0 & 0 \end{bmatrix}$$

Multiple the first column of the matrix by d, the second column by k_1, the third column of k_2 and subtracting the appropriate multiples of the remaining columns to perform modular reduction. After these operations, the following lattice vector will be obtained.

$$\mathbf{I} = (l_1, l_2, a, \frac{s}{r_s}, \frac{k_1}{r_{k_1}}, \frac{k_2}{r_{k_2}})$$

Using this lattice vector we can recover the secret key d easily.

The procedures for finding out the solution of secret key d is similar to the previous sections.

4.10 EC_ Nyberg-Rueppel Digital Signature with LCG

In this part, we will assume using the LCG to produce the random number for sign the signature and using EC_Nyberg-Rueppel Digital Signature.
Let the sequences of the random number be u_1, u_2, ..., u_l and the corresponding public key are $V_1 = (x_{V_1}, y_{V_1})$, $V_2 = (x_{V_2}, y_{V_2})$, ..., $V_i = (x_{V_i}, y_{V_i})$.
The corresponding integer representation of x_{V_1}, x_{V_2}, ... , x_{V_l} be V_{x_1} , V_{x_2}, ..., V_{x_l}. Thus we have the following equations.

$$\begin{aligned} c_i &= V_{x_i} + f_i \pmod r & (4.7) \\ d_i &= u_i - sc_i \pmod r & (4.8) \\ u_i &= a \times u_{i-1} + b \pmod M & (4.9) \end{aligned}$$

From (1) and (2),

$$\begin{aligned} d_i &= u_i - s(V_{x_i} + f_i) \pmod r \\ s(V_{x_i} + f_i) &= u_i - d_i \pmod r \end{aligned}$$

Now, we only consider two signatures are received by an adversary. That is the adversary knows f_1 , f_2 , V_{x_1} and V_{x_2}. One additional assumption is that the parameters a , b and M were known. So the following equations can be obtained easily.

$$\begin{aligned} d_1 &= u_1 - sc_1 \pmod r \\ d_2 &= u_2 - sc_2 \pmod r \\ b &= a \times u_1 - u_2 \pmod M \end{aligned}$$

Solving these equations can obtain the value of secret key s. However, these equations with different moduli cause it is difficult to solve them.

4.10.1 Solution

We need to set up a lattice L which is generated by the columns of the matrix $\mathbf{E}$ to find out the solution of secret key.

$$\mathbf{E} = \begin{bmatrix} -c_1 & 1 & 0 & r & 0 & 0 \\ -c_2 & 0 & 1 & 0 & r & 0 \\ 0 & -a & 1 & 0 & 0 & M \\ r_s^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{u_1}^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{u_2}^{-1} & 0 & 0 & 0 \end{bmatrix}$$

Multiple the first column of the matrix by s, the second column by u_1, the third column of u_2 and subtracting the appropriate multiples of the remaining columns to perform modular reduction. After these operations, the following lattice vector will be obtained.

$$\mathbf{I} = (d_1, d_2, b, \frac{s}{r_s}, \frac{u_1}{r_{u_1}}, \frac{u_2}{r_{u_2}})$$

Using this lattice vector we can recover the secret key s easily.
The procedures for finding out the solution of secret key s is similar to the previous sections.

4.11 EC_ Nyberg-Rueppel Digital Signature with PN-sequence

In this part, we assume the EC_Nyberg-Rueppel Digital Signature uses PN-sequence to generate the random nonce u. Therefore, the following equations will be obtained.

$$\begin{aligned} c_i &= V_{x_i} + f_i \pmod r \\ d_i &= u_i - sc_i \pmod r \\ u_{i+1} &= \frac{u_i}{2} - \left\{ \begin{array}{c} \frac{1}{2} \\ 0 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 2^{n-1} \end{array} \right\} \end{aligned}$$

We change d_1 and d_2 as the objects of the equations.

$$\begin{aligned} d_1 &= u_1 - sc_1 \pmod r \\ d_2 &= u_2 - sc_2 \pmod r \\ u_2 &= \frac{u_1}{2} - \left\{ \begin{array}{c} \frac{1}{2} \\ 0 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 2^{n-1} \end{array} \right\} \\ -u_1 + 2u_2 &= \left\{ \begin{array}{c} 0 \\ 1 \end{array} \right\} + \left\{ \begin{array}{c} 0 \\ 2^{n} \end{array} \right\} \end{aligned}$$

Solving these equations can obtain the secret key d. However, these equations with difference moduli cause they difficult to solve. Again, using Babai's algorithm to find out the solution.

4.11.1 Solution

We need to set up a lattice L which is generated by the columns of the mat rix $\mathbf{E}$ to find out the solution of secret key.

$$\mathbf{E} = \begin{bmatrix} -c_1 & 1 & 0 & r & 0 & 0 \\ -c_2 & 0 & 1 & 0 & r & 0 \\ 0 & -1 & 2 & 0 & 0 & 2^n \\ r_s^{-1} & 0 & 0 & 0 & 0 & 0 \\ 0 & r_{u_1}^{-1} & 0 & 0 & 0 & 0 \\ 0 & 0 & r_{u_2}^{-1} & 0 & 0 & 0 \end{bmatrix}$$

Multiple the first column of the matrix by s, the second column by u_1, the third column of u_2 and subtracting the appropriate multiples of the remaining columns to perform modular reduction. After these operations, the following lattice vector will be obtained.

$$\mathbf{I} \; = \; (d_1, d_2, -\left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\} + \left\{ \begin{matrix} 0 \\ 2^n \end{matrix} \right\}, \frac{s}{r_s}, \frac{u_1}{r_{u_1}}, \frac{u_2}{r_{u_2}})$$

Using this lattice vector we can recover the secret key s easily.
The procedures for finding out the solution of secret key s is similar to the previous sections.

4.12 EC_Nyberg-Rueppel Digital Signature with ICG

In this part, we assume the EC_Nyberg-Rueppel Digital Signature uses ICG to generate the random nonce u. Therefore, the following equations will be obtained.

$$\begin{aligned}
c_i &= V_{x_i} + f_i \pmod r \\
d_i &= u_i - sc_i \pmod r \\
u_{i+1} &= a\bar{u}_i + b \pmod M \\
d_1 &= u_1 + sc_1 \pmod r \\
d_2 &= u_2 + sc_2 \pmod r \\
u_2 u_1 &= a + bu_1 \pmod M
\end{aligned}$$

Solving these equations can obtain the secret key d. However, these equations with difference moduli cause they difficult to solve. Again, using Babai's algorithm to find out the solution.

4.12.1 Solution

We need to set up a lattice L which is generated by the columns of the matrix $\mathbf{E}$ to find out the solution of secret key.

$$\mathbf{E} = \begin{bmatrix}
-c_1 & 1 & 0 & 0 & r & 0 & 0 \\
-c_2 & 0 & 1 & 0 & 0 & r & 0 \\
0 & -b & 0 & 1 & 0 & 0 & M \\
r_s^{-1} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & r_{u_1}^{-1} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & r_{u_2}^{-1} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & r_{u_1 u_2}^{-1} & 0 & 0 & 0
\end{bmatrix}$$

Multiple the first column of the matrix by s, the second column by u_1, the third column of u_2 and subtracting the appropriate multiples of the remaining columns to perform modular reduction. After these operations, the following lattice vector will be obtained.

$$\mathbf{I} = (d_1, d_2, b, \frac{s}{r_s}, \frac{u_1}{r_{u_1}}, \frac{u_2}{r_{u_2}}, \frac{u_1u_2}{r_{u_1u_2}})$$

Using this lattice vector we can recover the secret key s easily.
The procedures for finding out the solution of secret key s is similar to the previous sections.

Chapter 5

Conclusion

In general, the subexponential algorithm used to tackle the discrete logarithm cannot be adapted to the elliptic curve environment. And there are some exponential algorithms to solve the elliptic curve discrete logarithm problem. That means, finding a solution to elliptic curve discrete logarithm problem is becoming infeasible much faster than finding a solution to discrete logarithm problem. However, some elliptic curve logarithm problem can be reduced to the discrete logarithm problem. That is, if an elliptic curve with field size q, the elliptic curve discrete logarithm problem can be reduced to the discrete logarithm problem in F_{q^k}, where k is an positive integer [24]. From the above, we can conclude that elliptic curve discrete logarithm is much more difficult to solve than discrete logarithm problem is. Elliptic curve cryptosystem is more secure than discrete logarithm problem's cryptosystem is.

In chapter 4, we showed that two digital signature schemes(Digital Signature Algorithm and Nyberg-Rueppel Signature Scheme) are implemented in finite field group and elliptic curve abelian group with different types of pseudorandom number generators. All of these combinations schemes can find out the secret key by using the method which described in [17] although the secret key appears to be protected by the schemes. If someone thinks that using elliptic curve cryptosystem is more secure, then use shorter key sizes and does not take caution against which pseudorandom number generator is used, it is more risky than discrete logarithm problem's cryptosystem. Therefore, one should stick to cryptographically secure pseudorandom number generators.

Bibliography

[1] D.E. Knuth. The art of Computer Programming, vol2: Seminumerical Alogrithms", Addison-Wesley, Reading Mass., 1969

[2] P. Hellekalek, Inversive Pseudorandom Number Generators: Concepts, Results and Links. http://random.mat.sbg.ac.at.

[3] Eichenauer, J. and J. Lehn. A Non-linear Congruential Pseudo-random Number Generator. Statistic Papers, vol27: P.315-326, 1986.

[4] L. Babai. On Lovasz' Lattice Reduction and the Nearest Lattice Point Problem. Combinatorica, vol6(1), P.1-P.13, 1986.

[5] A.K. Lenstra, H.W. Lenstra, Jr. and L. Lovasz. Factoring polynomials with rational coefficients, Math. Ann., vol261, P.515-534, 1982.

[6] L. Lovasz, Private Communications, 1982 - 1982.

[7] M. Grotschel, L. Lovasz and A. Schrijver. The ellipsoid method and its conssequences in combinatorial optimization, Combinatorica 1, P.186-197, 1981.

[8] V. Miller. Uses of Elliptic Curves in Cryptography. Advances in Cryptology-Crypto 85, P.417-426, 1986.

[9] N. Koblitz. Elliptic Curve Cryptosystems. Mathematics of Computation, vol48. P.203- 209, 1987.

[10] P. Montgomery, Speeding the Pollard and Elliptic Curve Methods of Factorization. Mathematics of Computation, vol48, P.243-264, 1987.

[11] A.O.L. Atkin and F. Morain, Finding Suitable Curves for the Ellipticc Curve Method of Factorization. Mathematics of Computation, vol60. P.399-405, 1993

[12] A.O.L. Atkin and F. Morain. Elliptic Curves and Primality Proving. Mathematics of Computation,1993

[13] B. Kaliski. A Pseudorandom Bit Generator Based on Elliptic Logarithms. Advances in Cryptology - CRYPTO 86, Lecture Notes in COmputer Science, vol293, P.84-103, 1987.

[14] B. Kaliski. One-way Permutations on Elliptic Curves. Journal of Crptology, vol3, P.187 - 199, 1991.

[15] G. Harper, A. Menezes and S. Vanstone, Public-key Cryptosystems with very small key lengths. Advances in Cryptology - Eurocrypt 92, 1992.

[16] A. J. Menezes. Elliptic Curve Public Key Cryptosystems. Kluwer Academic Publishers, 1993.

[17] M. Bellare, S. Goldwasser, D.Micciancio. "Pseudorandom" Number Generation within Cryptographic Algorithms: the DSS case. Advances in Cryptology - Crypto97 Proceedings, Lecture Notes in Computer Science, vol1294, 1997.

[18] National Institute of Standards and Technology (NIST). FIPS Publication 186: Digital Signature Standard, May 1994.

[19] M.Blum and S. Micali. How to Generate Cryptographically Strong Sequences of Pseudo-random Bits. SIAM J. Computing, vol13(4). P.850-863, 1984.

[20] O. Goldreich, S. Goldwasswer, and S. Micali. How to Construct Random Functions. Proc. 25th IEEE Symp. on Foundations of Comp. Science, P.464-479, 1984.

[21] Adi Shamir. The Generation of Cryptographically Strong Pseudo-random Sequences. Advances in Cryptology : Crypto 81, P.1-1, 1982.

[22] A. C. Yao. Theory and Application of Trapdoor Functions. Proc. 23rd IEEE Symp. on Foundations of Comp. Science. P.80-91, 1982.

[23] J. Plumstead Boyar. Inferring a sequence generated by a linear congruence. Proc. 23rd IEEE Symp. on Foundations of Comp. Science. P.153-159, 1982.

[24] A. Menezes, T. Okamoto and S. Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a finite field. Proc. 23rd Annual ACM Symp. on Theory of Computing, P.80-89, 1991.