Security and Efficiency Analyses of Public Key Cryptosystems

(公開鍵暗号の安全性解析と高速化手法)

國 廣 昇

Abstract

This thesis analyzes the security and efficiency of public key cryptosystems. New attacks for several cryptosystems are proposed and the effectiveness of the attacks is evaluated. Furthermore, solutions are given to several unsolved problems in computational number theory and algebraic geometry theory that are closely related to the security of public key cryptosystems. Moreover, new calculation methods are proposed to speed up encryption and decryption.

This thesis consists of the following eight chapters.

Chapter 1 is the introduction. We explain the main purpose of our studies and overview previous works related to our studies. Chapter 2 gives the preliminaries. We summarize the mathematics and cryptosystems appearing in this thesis.

We analyze the security of several cryptosystems from Chapter 3 to Chapter 6. In Chapter 3, we investigate how the elliptic curve factoring method, which is an efficient attack for public key cryptosystems, especially RSA cryptosystem, can be speeded up. In Chapter 4, we analyze the security of a certain type of elliptic curve cryptosystem defined over a composite modulus. We also investigate the difficulty of a known problem — the problem of counting the number of points on an elliptic curve over the ring $\mathbf{Z}/n\mathbf{Z}$ —. This problem is assumed to be as difficult to solve as the cryptosystem is to break. We prove that this problem is computationally equivalent to a factoring problem. In Chapter 5, we investigate the difficulty of an elliptic curve discrete logarithm problem over a super-anomalous elliptic curve. We prove that this problem can be solved in deterministic polynomial time. In Chapter 6, the multi-variate RSA cryptosystem is defined and its security and efficiency are evaluated. We prove that this cryptosystem can be broken under an unusual usage.

In Chapter 7, we describe how to speed up public key cryptosystems. We propose new methods to generate short addition chains. Moreover, we evaluate their efficiency.

Chapter 8 is the conclusion. We summarize results obtained in this thesis.

Contents

Chapter 1

Introduction

1.1 Background

Modern cryptography was born by Diffie and Hellman's proposal of public key cryptosystems in 1976 [15] and by the adoption of DES as a U. S. Federal Information Standard for encrypting unclassified information in 1977 [46]. In 1978 Rivest, Shamir, and Adleman discovered the first practical public key cryptosystem and signature scheme, which is now referred to as RSA [57]. The security of an RSA cryptosystem is based on the difficulty of factoring a large composite number. After the proposal of the RSA scheme, several cryptosystems were developed whose security were based on the difficulty of other mathematical problems, for example, discrete logarithm problems or elliptic curve discrete logarithm problems. The ElGamal cryptosystem [17] is an example of former and Elliptic Curve Cryptosystems [26][41] are examples of the latter. It is interesting that these problems can be applied in an extremely practical field, i.e., cryptography, although they are considered as purely mathematical problems.

Since these problems are closely related to number theory and computational theory, many researchers have begun to research cryptography and some have developed cryptography theories. In spite of these efforts by researchers, however, a lot of problems remain unsolved. For example, it is not clear how difficult a factoring problem is. In addition, it has not been proven whether the factoring problem is difficult or easy.

On the other hand, cryptography is attracting notice from another angle. Economic activities such as electronic trade are rapidly spreading. Unfortunately, network used in these activities is not guaranteed to be secure. Hence, there are many dangerous points, such as eavesdropping and message forging. Electronic trade cannot be made safe without eliminating such dangerous points. Therefore, "cryptography" has been recognized as an essential tool for secure electronic commerce.

It depends on the answers to the following two questions whether a cryptosystem can spread globally or not. One is " Is this cryptosystem secure?" and the other is "Are its encoding and decoding speeds fast enough?" If a cryptosystem is not secure, it cannot be called a cryptosystem, and even if it is secure, if its encoding and decoding speeds are slow,

it has no meaning as a cryptosystem.

As we have seen, cryptography has been attracting notice from both academic and practical aspects and has been making steady progress. The important *academic* question of : "How long does it take to factorize a big composite number?" can be translated into the important *practical* question of: "How secure is RSA cryptosystem?"

As we have described above, however, there remain many unsolved problems in cryptosystem security. In this thesis, therefore, we give answers to some unsolved problems. By using these results, the reader can make communications via insecure networks more safe. In addition, we propose several algorithms to increase the encoding and decoding speeds.

Modern cryptography can be classified into *public key cryptography* and *secret key cryptography*. In this thesis, our concern is only public key cryptography.

1.2 Overview

In this section, we overview previous researches analyzing the security and efficiency of public key cryptosystems.

1.2.1 Security analyses on public key cryptosystems

Ever since the first proposal of the RSA cryptosystem [57], two important problems have been attracting the attention of many researchers. One is "How difficult is the factoring problem? Are there factoring algorithms able to run in polynomial time?" The other one is "Is breaking RSA cryptosystem computationally equivalent to the factoring problem?" These two problems remain unsolved although 23 years have passed since the proposal. Nonetheless, these problems continue to be steadily analyzed.

The Elliptic Curve factoring Method (ECM) [33][43] and Number Field Sieve Method (NFSM) [34] are effective integer factoring methods. In the following discussion, let n be the composite to be factored and p be the smallest prime factor of n. The running time of ECM depends on the size of prime factor p. Concretely speaking, the running time is evaluated to be the sub-exponential time of $\log p$. Note that this sub-exponential time is longer than polynomial time and shorter than exponential time. On the other hand, the running time of NFSM depends on the size of composite n. Concretely speaking, the running time is evaluated to be the sub-exponential time of $\log n$. We can easily verify that NFSM is more effective than ECM if composite n is the product of two same-sized primes, assuming an RSA cryptosystem; ECM is more effective otherwise.

However, the above statements merely involve the best known algorithms and their running times and do not mention whether there are faster algorithms than the above two algorithms. Hence, research on integer factoring can be expected to go on. In this thesis, a new efficient ECM is proposed.

The relationship between the difficulty of breaking RSA cryptosystem and solving the factoring problem has been analyzed. In 1998, Boneh and Venkatesan provided evidence

that breaking RSA cannot be equivalent to solving the factoring problem [6]. Moreover, some attacks have been proposed where it is not necessary to know prime factors of the composite. In 1990, Wiener proved that RSA can be easily broken if the size of the secret key is less than 1/4 of the composite size [68]. Developing this analysis, in 1999, Boneh proved that RSA can be easily broken if the size of the secret key is less than 0.292 of the composite size [5].

Many variants of RSA cryptosystems have been proposed and have been analyzed in terms of security. The Rabin cryptosystem [53], KMOV cryptosystem [27], and Koyama scheme [28] are famous examples of such cryptosystems. It has been proven that the difficulty of breaking the Rabin cryptosystem is computationally equivalent to the difficulty of factoring problem [53]. In addition, it has been pointed out that the Koyama scheme can be easily broken if this scheme is used under special conditions [8]. In this thesis, we describe the relationship between the factoring problem and a certain algebraic-geometric problem related to the difficulty of breaking the KMOV cryptosystem. Moreover, we generalize the RSA cryptosystem by using multi-variate rational functions and analyze its security.

The ElGamal cryptosystem [17] and elliptic curve cryptosystems [26][41] are some other effective public key cryptosystems. Let's relate these cryptosystems to a mathematical problem. It is strongly believed that a discrete logarithm problem is difficult. However, it has not been proven how difficult such a problem is yet. The best method for solving this problem is to use the *index calculus method* [1][11][18][52][12][19][20]. This method runs in sub-exponential time.

The first Elliptic Curve Cryptosystem was proposed [26][41] by substituting an operation over elliptic curves for an operation over a finite field in the ElGamal cryptosystem. The security of this cryptosystem is based on the difficulty of an elliptic curve discrete logarithm problem. It is considered that this problem is more difficult to solve than an ordinary discrete logarithm problem since the *index calculus method* cannot be applied directly [39]. This allows us to use short key lengths. Hence, elliptic curve cryptosystems can encrypt and decrypt faster than the ElGamal cryptosystem and RSA cryptosystems although operations of encryption and decryption are more complicated than these cryptosystems.

The Discrete Logarithm Problem (DLP) over general elliptic curves is considered to be difficult. The solution requires exponential time. However, it has been reported that DLP for a special elliptic curve can be solved easily. More specifically, in 1991, Menezes, Okamoto, and Vanstone reported that DLP for a supersingular elliptic curve can be solved in sub-exponential time [38]. In 1997, Semaev [61], Smart [65], Satoh-Araki [58], and Rück [55] independently reported that DLP for anomalous elliptic curves is solvable in polynomial time. This means that an attacker can easily read a whole message in an instant if we happen to choose an anomalous elliptic curve in Elliptic Curve Cryptosystem. In this thesis, we extend an anomalous elliptic curve to a super-anomalous elliptic curve and show that DLP for such a super-anomalous elliptic curve can be solved in deterministic polynomial time.

1.2.2 Efficiency of public key cryptosystems

The RSA cryptosystem can achieve fast calculation by reducing the number of multiplications as small as possible and/or speeding up the multiplication operations. In this thesis, we consider the former approach, i.e., how to reduce the number of multiplications.

It is well known that a power exponentiation can be efficiently calculated by using an addition chain [25]. Obtaining the shortest chain, however, is an NP-hard problem [16]. Hence, we cannot obtain the shortest chain in a practical time. Due to this problem, many algorithms have been proposed to obtain a sub-optimal chain, for example, the Bos-Coster method [9], the Yacobi method [67], the m-ary or 2^κ-ary method [25], the Lou-Chang method [35], the Window method [25], and the Extended Window method with the Tunstall-like algorithm [29][32], etc. In this thesis, we propose two efficient algorithms able to generate short addition chains for a number whose binary sequence has relatively many bit 1's.

1.3 Outline of this thesis

In Chapter 2, we summarize the mathematics and cryptosystems appearing in this thesis.

In Chapter 3, we propose a new elliptic curve factoring method. We show its efficiency by theoretical analysis.

In Chapter 4, we analyze the security of a certain type of elliptic curve cryptosystem defined over a composite modulus. We also investigate the difficulty of a certain problem — the problem of counting the number of points on an elliptic curve over the ring $\mathbf{Z}/n\mathbf{Z}$ —. This problem is assumed to be as difficulty as the cryptosystem is to break. We prove that this problem is computationally equivalent to the factoring problem. In Chapter 5, we investigate the difficulty of an elliptic curve discrete logarithm problem over a super-anomalous elliptic curve. We prove that this problem can be solved in deterministic polynomial time. In Chapter 6, the multi-variate RSA cryptosystem is defined and its security and efficiency are evaluated. We prove that this cryptosystem can be broken under unusual usages.

In Chapter 7, we describe how to speed up public key cryptosystems. We propose new methods to generate short addition chains. Moreover, we evaluate the efficiency of these methods.

In the last chapter, we summarize results obtained in this thesis.

Chapter 2

Preliminaries

2.1 Mathematical notation

In this section, we summarize the mathematical notation used in this thesis, except for Chapter 7.

2.1.1 Number theory

Let p be a prime and $\mathbf{F}_p$ be a finite field with p elements. Let $\mathbf{Z}$ be an integer ring and $\mathbf{Z}/n\mathbf{Z}$ be a residue ring modulo n, where n is a positive integer. Let $(\mathbf{Z}/n\mathbf{Z})^*$ be a set of integers in the interval $[1, n]$, which are relatively prime to n. Let $\mathbf{Z}_p$ be a ring of p-adic integers. We denote the residue of a modulus n as $a \bmod n$. We write $a \equiv b (\bmod\ n)$ if n divides $(a - b)$, where a and b are integers.

The Legendre symbol and the Jacobi symbol are important for number theory.

Definition 2.1 Let p be a prime and a be an integer. The symbol (a/p) will have the value 1 if the congruence $x^2 \equiv a \pmod{p}$ has a solution, -1 if this congruence has no solution, and 0 if $p|a$. The symbol (a/p) is called the *Legendre symbol.*

Definition 2.2 Assume that the composite $n = {p_1}^{e_1}{p_2}^{e_2}\cdots{p_k}^{e_k}$. The Jacobi symbol (a/n) is defined as $(a/n) = (a/p_1)^{e_1}(a/p_2)^{e_2}\cdots(a/p_k)^{e_k}$. Each $(a/p.)$ is the Legendre symbol.

These symbols are easily computable.

We define the Euler ϕ function.

Definition 2.3 For $n > 1$, let $\phi(n)$ denote the number of integers in the interval $[1, n]$, which are relatively prime to n. The function ϕ is called the Euler phi function.

If $n = {p_1}^{e_1}{p_2}^{e_2}\cdots{p_k}^{e_k}$ is the prime factorization of n, then

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_k}\right).$$

Note that $\phi(n)$ is equal to the number of elements in $(\mathbf{Z}/n\mathbf{Z})^*$. We have an important theorem.

Euler's Theorem If $\gcd(x, n) = 1$, then $x^{\phi(n)} \equiv 1 \pmod{n}$. This theorem is extended as follows.

Lagrange's Theorem Let G be an abelian group, $\mathcal{O} \in G$ be an identity element, and $+$ be a binary operation on G. Let $m \cdot x$ be m-times the element of x, or $m \cdot x = \underbrace{x + x + \cdots x}_{m \text{ times}}$. Let $\#G$ be the number of elements in G. For all $x \in G$, $\#G \cdot x = \mathcal{O}$.

By substituting $G = (\mathbf{Z}/n\mathbf{Z})^*, \#G = \phi(n)$, and $\mathcal{O} = 1$, we can easily confirm that Euler's Theorem is a special case of Lagrange's Theorem.

In this thesis, we assume that a prime p is neither 2 nor 3 and a composite n is neither divided by 2 nor 3.

2.1.2 Elliptic curve theory

In this section, we summarize the elliptic curve theory used in this thesis. Refer to [64][39] for the details on this elliptic curve theory.

An elliptic curve $E(\mathbf{F}_p)$ is the set of points $(x, y) \in {\mathbf{F}_p}^2$ on the curve $E/\mathbf{F}_p : y^2 = x^3 + a_p x + b_p$, where $a_p, b_p \in \mathbf{F}_p$ and assuming $4a^3 + 27b^2 \neq 0$, including a point at infinity $\mathcal{O}_p$. Let $\#E(\mathbf{F}_p)$ be the number of points over $E/\mathbf{F}_p$. If $\#E(\mathbf{F}_p)$ is p, this curve is called *anomalous*. If $\#E(\mathbf{F}_p)$ is $p + 1$, this curve is called *supersingular*.

It is well known that the points on an elliptic curve form an abelian group under a certain addition. The addition rules are given below.

Let $P = (x_1, y_1), Q = (x_2, y_2) \in E(\mathbf{F}_p)$. If $x_1 = x_2$ and $y_1 = -y_2$, then $P + Q = \mathcal{O}_p$. Otherwise, $P + Q = (x_3, y_3)$, where

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

and

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

Let $kP = \underbrace{P + P + \cdots P}_{k \text{ times}}$, where k is a positive integer. The computation amount of calculating kP is $\mathrm{O}(\log k \cdot (\log p)^2)$. In particular, if $k \approx p$, the amount is $\mathrm{O}((\log p)^3)$. In this case, we can easily compute kP.

2.1.3 Elliptic curve over the ring $\mathbf{Z}/n\mathbf{Z}$

Next, we explain the elliptic curve over the ring $\mathbf{Z}/n\mathbf{Z}$. Let the composite $n = \prod_{i=1}^{k} {p_i}^{e_i}$, where k is the number of distinct prime factors of n, and E is the elliptic curve $E : y^2 = x^3 + ax + b$. A group $E(\mathbf{Z}/n\mathbf{Z})$, a set of points on E over $\mathbf{Z}/n\mathbf{Z}$, is defined as the direct sum of k groups $E(\mathbf{Z}/n\mathbf{Z}) = \oplus_{i=1}^{k} E(\mathbf{Z}/{p_i}^{e_i}\mathbf{Z})$[39]. Hence, each element R over $E(\mathbf{Z}/n\mathbf{Z})$ can be represented by $P_i \in E(\mathbf{Z}/{p_i}^{e_i}\mathbf{Z})$, $1 \leq i \leq k$, and we denote it as $R = \langle P_1, P_2, \ldots, P_k \rangle$. In this group $E(\mathbf{Z}/n\mathbf{Z})$, we define a point at infinity of $E(\mathbf{Z}/n\mathbf{Z})$: $\mathcal{O}$ to be the point

$\langle \mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_k \rangle$, where $\mathcal{O}_i$s are points at infinity of $E(\mathbf{Z}/{p_i}^{e_i}\mathbf{Z})$, respectively. In addition, we define points at semi-infinity to be such points $\langle P_1, P_2, \ldots, P_k \rangle$ that at least one P_i is $\mathcal{O}_i$. For simplicity, we often use E_n instead of $E(\mathbf{Z}/n\mathbf{Z})$. When we emphasize coefficients a and b, we write $E_n(a,b)$ instead of $E(\mathbf{Z}/n\mathbf{Z})$.

The number of points over $E(\mathbf{Z}/n\mathbf{Z})$, $\#E(\mathbf{Z}/n\mathbf{Z})$, is equal to

$$\#E(\mathbf{Z}/n\mathbf{Z}) = \prod_{i=1}^{k} \#E(\mathbf{Z}/{p_i}^{e_i}\mathbf{Z}) = \prod_{i=1}^{k} \#E(\mathbf{F}_{p_i}) \cdot {p_i}^{e_i-1}.$$

Note that $\#E(\mathbf{Z}/{p_i}^{e_i}\mathbf{Z}) = \#E(\mathbf{F}_{p_i}) \cdot {p_i}^{e_i-1}$ since $E(\mathbf{Z}/{p_i}^{e_i}\mathbf{Z}) = E(\mathbf{F}_{p_i}) \oplus \mathbf{F}_{{p_i}^{e_i-1}}$. We can also define the number of points as

$$\begin{aligned} \#'E(\mathbf{Z}/n\mathbf{Z}) &= \mathbf{LCM}(\#E(\mathbf{Z}/{p_1}^{e_1}\mathbf{Z}), \#E(\mathbf{Z}/{p_2}^{e_2}\mathbf{Z}), \ldots, \#E(\mathbf{Z}/{p_k}^{e_k}\mathbf{Z})) \\ &= \mathbf{LCM}(\#E(\mathbf{F}_{p_1}), \#E(\mathbf{F}_{p_2}), \ldots, \#E(\mathbf{F}_{p_k})) \cdot \prod_{i=1}^{k} {p_i}^{e_i-1}. \end{aligned}$$

If all $\#E(\mathbf{F}_{p_i})$ are relatively prime, $\#'E(\mathbf{Z}/n\mathbf{Z})$ is equivalent to $\#E(\mathbf{Z}/n\mathbf{Z})$.

Ordinary points are any points other than the point at infinity or the points at semi-infinity. We define $E^*(\mathbf{Z}/n\mathbf{Z})$ as the set of ordinary points. Note that $E^*(\mathbf{Z}/n\mathbf{Z})$ is equivalent to the set of (x_0, y_0) satisfying $y_0^2 \equiv x_0^3 + ax_0 + b \pmod{n}$. We can denote the ordinary points as $P_i = R \bmod {p_i}^{e_i}$, $1 \leq i \leq k$. Or, letting $R = (x_R, y_R)$ and $P_i = (x_i, y_i)$, we can also write $x_i = x_R \bmod {p_i}^{e_i}$ and $y_i = y_R \bmod {p_i}^{e_i}$. Note that $\#E^*(\mathbf{Z}/n\mathbf{Z}) = \prod_{i=1}^{k} \#E^*(\mathbf{Z}/{p_i}^{e_i}\mathbf{Z}) = \prod_{i=1}^{k} (\#E(\mathbf{F}_{p_i}) - 1) \cdot {p_i}^{e_i-1}$.

We explain the most simple case, $n = p_1 p_2$. Each element R over $E(\mathbf{Z}/n\mathbf{Z})$ can be represented by $R = \langle R_1, R_2 \rangle$, where $R_1 \in E(\mathbf{F}_{p_1}), R_2 \in E(\mathbf{F}_{p_2})$. A point at infinity of $E(\mathbf{Z}/n\mathbf{Z})$: $\mathcal{O}$ is $\langle \mathcal{O}_1, \mathcal{O}_2 \rangle$. Points at semi-infinity are any points such that $\langle \mathcal{O}_1, R_2 \rangle$ or $\langle R_1, \mathcal{O}_2 \rangle$, where $R_1 \in E^*(\mathbf{F}_{p_1}), R_2 \in E^*(\mathbf{F}_{p_2})$. Ordinary points are any points such that $\langle R_1, R_2 \rangle$, where $R_1 \in E^*(\mathbf{F}_{p_1}), R_2 \in E^*(\mathbf{F}_{p_2})$.

We define an addition on the point of $E(\mathbf{Z}/n\mathbf{Z})$ as follows. If $n = p_1 p_2$ and $P = \langle P_1, P_2 \rangle$ and $Q = \langle Q_1, Q_2 \rangle$, where $P, Q \in E(\mathbf{Z}/n\mathbf{Z})$, $P_1, Q_1 \in E(\mathbf{F}_{p_1})$, and $P_2, Q_2 \in E(\mathbf{F}_{p_2})$, the addition $P + Q = \langle P_1 + Q_1, P_2 + Q_2 \rangle$, where the first elements are added over $E(\mathbf{F}_{p_1})$ and the second elements are added over $E(\mathbf{F}_{p_2})$ by using the same addition rules as in Section 2.1.2. Under this addition, $E(\mathbf{Z}/n\mathbf{Z})$ is a group.

We cannot execute this addition when we do not know the prime factors of n. Therefore, we define "a pseudo-addition" on the point of $E^*(\mathbf{Z}/n\mathbf{Z})$ by using the same addition rules. This pseudo-addition can be executed without knowing the prime factors of n. Unfortunately, $E(\mathbf{Z}/n\mathbf{Z})$ is not a group under this addition. This is evident since the addition is not always defined: if $\gcd(x_2 - x_1, n) > 1$ (for the case $P \neq Q$), or if $\gcd(2y, n) > 1$ (for the case $P = Q$), then we cannot compute λ. In this case, we will obtain the prime factors of n.

We can view this phenomenon from the following two points. One is that we can ignore this case since this probability is negligible. The other is that we can factorize a composite n by using this property. Both are meaningful.

2.2 Collection of public key cryptosystems

In this section, we explain some number theory problems and three public key cryptosystems related to this thesis.

2.2.1 Intractable mathematical problems

The followings are famous intractable problems in number theory. First, we formulate these problems.

Factoring Problem: Given composite n, find the complete prime factorization of n.

Discrete Logarithm Problem (DLP): Given prime p, and two elements g and $h \in \mathbf{F}_p{}^*$, find positive integer α such that $h = g^\alpha \bmod p$.

Elliptic Curve Discrete Logarithm Problem (ECDLP): Given elliptic curve $E(\mathbf{F}_p)$, and two points G and $H \in E(\mathbf{F}_p)$, find positive integer α such that $H = \alpha G$ over $E(\mathbf{F}_p)$.

In this thesis, we discuss other problems, which are not always intractable. These problems are counting the number of points on Elliptic Curve over Ring $\mathbf{Z}/n\mathbf{Z}$, ECDLP over $\mathbf{Z}/n\mathbf{Z}$, and ECDLP over a super-anomalous elliptic curve. We formulate these problems in Sections 4.1, 4.1, and 5.2, respectively.

Table 2.1 shows the correspondence between problems and examples of cryptosystems based on the difficulties of the cryptosystems.

Table 2.1. Problems and cryptosystems

Problem	Examples of schemes	Difficulty
Factoring Problem	RSA, Rabin, KMOV	Sub-exponential time
Discrete Logarithm Problem	ElGamal	Sub-exponential time
ECDLP	ECC	Exponential time
Problem of Counting the Number of Points on Elliptic Curve over Ring $\mathbf{Z}/n\mathbf{Z}$	KMOV	Equivalent to factoring Described in Section 4.1
ECDLP over $\mathbf{Z}/n\mathbf{Z}$	–	Described in Section 4.1
ECDLP over a Super-anomalous Elliptic Curve	OU-Identity based scheme	Tractable Described in Section 5.2

RSA: Rivest-Shamir-Adleman cryptosystem [57]
KMOV: Koyama-Maurer-Okamoto-Vanstone cryptosystem [27]
ECC: Elliptic Curve Cryptosystem [26][41]
OU-Identity based scheme: Okamoto and Uchiyama Identity based scheme [48]

In this table, "Exponential time" means that the best algorithms proposed until now need time $O(\exp(c \log t))$ to solve the problem for input t, or input length $\log t$, where c is

a constant. "Sub-exponential time" means that the best algorithms need time $L(t, c, \alpha) \equiv O(\exp(c(\log t)^{\alpha}(\log\log t)^{1-\alpha}))$ to solve the problem, where c is a constant and $0 < \alpha < 1$.

Elliptic Curve Method and Number Field Sieve Method are efficient for solving factoring problem. The computation amount for Elliptic Curve Method depends on the size of the smallest prime factor of composite n. It is evaluated as $L(p, \sqrt{2}, 1/2) = O(\exp(\sqrt{2\log p \log\log p}))$. On the other hand, the computation amount for Number Field Sieve Method depends on the size of composite n. It is evaluated as $L(n, 1.923, 1/3) = O(\exp(1.923(\log n)^{1/3}(\log\log n)^{2/3}))$.

The Index Calculus method is efficient to solve DLP and its computation time has been evaluated as $L(p, c, 1/3)$ for some constant c. DLP is believed to be as difficult to solve as factoring problem.

The Pohlig-Hellman method and Pollard method, etc., are efficient for solving ECDLP for general elliptic curves. These methods have a variety of improvements. However, the computation time of each of these methods is $O(\sqrt{p})$. ECDLP for general curves is believed to be more difficult to solve than factoring problem and DLP. ECDLP for special curves (supersingular curves or anomalous curves) is an exception of the above discussion.

2.2.2 Collection of Public Key Cryptosystems

In this section, we describe RSA cryptosystem, ElGamal cryptosystem, and Elliptic Curve Cryptosystem.

The RSA cryptosystem is described as follows.

RSA cryptosystem

Public key: $(e, n(= pq))$, where p and q are distinct primes.
Secret key: $(d, (p, q))$, where $ed \equiv 1 (\text{mod } \phi(n))$
Plaintext: M, where $0 < M < n$
Encryption: Compute $C = M^e \bmod n$
Ciphertext: C
Decryption: Compute $M = C^d \bmod n$

The encryption and decryption of the RSA cryptosystem work by the following reason. Since $ed \equiv 1 \pmod{\phi(n)}$, there exists an integer k such that $ed = 1 + k\phi(n)$. By Euler's theorem, in decryption, $C^d \bmod n$ is transformed as follows.

$$C^d \equiv (M^e)^d \equiv M^{ed} \equiv M^{1+k\phi(N)} \equiv M \cdot (M^{\phi(n)})^k \equiv M \cdot 1^k \equiv M \pmod{n}.$$

If we can factorize n, we can break the RSA scheme. In addition, if we can obtain $\phi(n)$, we can break the RSA scheme. However, it is not known whether we can or cannot factorize n when we can break the RSA scheme.

Next, the ElGamal Cryptosystem is described as follows.

ElGamal cryptosystem

Public key: (p, g, g^r), where g is a generator of multiplicative group $\mathbf{F}_p{}^*$ of the integers modulo p and r is a random integer, $1 \leq r \leq p - 2$.

Secret key: r

Plaintext: M, where $0 < M < p$

Encryption: Compute $C_1 = g^k \bmod p$ and $C_2 = m \cdot (g^r)^k \bmod p$, where k is a random integer, $1 \leq k \leq p-2$.

Ciphertext: $C = (C_1, C_2)$

Decryption: Compute $M = C_2 \cdot C_1^{-r} \bmod p$

The encryption and decryption of the ElGamal cryptosystem work by the following reason. In decryption, $C_2 \cdot C_1^{-r} \bmod p$ is transformed as follows.

$$C_2 \cdot {C_1}^{-r} \equiv m \cdot (g^r)^k \cdot (g^k)^{-r} \equiv m \cdot g^{rk} \cdot g^{-kr} \equiv m \pmod{p}.$$

If we can solve a discrete logarithm problem, we can break the ElGamal cryptosystem. On the other hand, it is not known whether we can or cannot solve a discrete logarithm problem when we can break the ElGamal scheme.

As we have seen in Section 2.1.2, the points on an elliptic curve over a finite field F_p form an abelian group. Next, the ElGamal cryptosystem is extended to Elliptic Curve Cryptosystem by changing $\mathbf{F}_p{}^*$ in ElGamal to $E(\mathbf{F}_p)$.

Elliptic Curve Cryptosystem (ECC)

Public key: $(E(\mathbf{F}_p), G, rG)$, where G is a generator of group $E(\mathbf{F}_p)$ and r is a random integer, $0 < r < \#E(\mathbf{F}_p)$.

Secret key: r

Plaintext: $M \in E(\mathbf{F}_p)$

Encryption: Compute $C_1 = kG$ and $C_2 = m + k(rG)$ over $E(\mathbf{F}_p)$, where k is a random integer, $0 < k < \#E(\mathbf{F}_p)$.

Ciphertext: $C = (C_1, C_2)$

Decryption: Compute $M = C_2 - rC_1$

The encryption and decryption of Elliptic Curve Cryptosystem work by the following reason. In decryption $C_2 - rC_1$ over $E(\mathbf{F}_p)$ is transformed as follows.

$$C_2 - rC_1 = m + k(rG) - r(kG) = m + krG - rkG = m \text{ over } E(\mathbf{F}_p).$$

If we can solve ECDLP, we can solve ECC. ECDLP is believed to be more difficult than DLP to solve. Hence, ECDLP can use a shorter key if the same security level is adapted. It is faster than the above RSA cryptosystem and ElGamal cryptosystem.

In this thesis, we also discuss other cryptosystems, e.g., the KMOV scheme, Koyama scheme, Okamoto-Uchiyama ID-based scheme, etc. The details of these schemes are described later.

Chapter 3

Speeding up Elliptic Curve Factoring Method

An Elliptic Curve Method (ECM) is an effective integer factorization method. After the first proposal of this method, many researchers developed ECM. Some researchers proposed that these methods should use elliptic curves whose orders have a given small factor (8, 12, or 16) so as to reduce the ECM's computation time. In this chapter, we discuss the ability of extending this idea. At first, on the assumption that there exists an algorithm to select elliptic curves whose orders have a bigger factor than ordinary ones, we discuss to what degree we can reduce the ECM's time. Finally, a theoretical analysis shows the effectiveness of the proposed method. For example, the proposed method runs 3 or 4 times faster than the original ECM.

3.1 Introduction

It is important to find efficient factoring algorithms for evaluating the security of cryptosystems. The ρ method [50], $p-1$ method [49], $p+1$ method [21], continued fraction method [51], quadratic sieve method [45], number field sieve method [34], elliptic curve factoring method [33][43], etc., have been proposed. These methods have been classified into two types: algorithms depending on prime factors and algorithms depending on composite numbers.

An Elliptic Curve Method is one of the former examples. Letting p be the smallest prime factor of composite n, the running time of this method is evaluated as $O(\exp\sqrt{2\log p\log\log p})$. A Number Field Sieve Method is one of the latter examples. Letting n be the composite to be factored, the running time of this method is evaluated as $O(\exp(1.923(\log n)^{1/3}(\log\log n)^{2/3}))$.

The Elliptic Curve Method (ECM) [33][43] can factorize the composite when the number of points over the elliptic curve becomes a product of small primes. If we choose a random elliptic curve for ECM, such a probability is very small. In order to overcome this, methods using special elliptic curves, whose number of points has a given small factor (12 or 16), and ways indicating how to construct such curves, have been proposed. Recently, Izu proposed

methods for constructing curves with a bigger factor, such as 27 or 36 [24].

In this chapter, we discuss the ability of extending this idea. At first, on the assumption that there exists an algorithm to select elliptic curves whose orders have a bigger factor than ordinary ones, we discuss to what degree we can reduce an ECM's time. Finally, a theoretical analysis shows the effectiveness of the proposed method. For example, the proposed method runs 3 or 4 times faster than the original ECM.

3.2 Original ECM

3.2.1 Elliptic Curve Factoring Method

In this section, we review the original elliptic curve factoring method [33]. Let $n(=pq)$ be a composite whose two prime factors are unknown. Suppose that $P \in E(\mathbf{Z}/n\mathbf{Z})$ and MP is M times the point of P as $MP = \underbrace{P+P+\cdots P}_{M \text{ times}} \equiv \langle P_1', P_2' \rangle$. We compute MP using the *tangent-and-chord* operation modulo n without knowing the prime factors p and q. Suppose that MP is calculated successively in a binary method as $M_1P(=P), M_2P, \ldots, M_lP(=MP)$. If MP is a point at semi-infinity as $MP = \langle \mathcal{O}_p, MP_2(\neq \mathcal{O}_q) \rangle$, then at least one M_iP $(1 \leq i \leq l)$ is a point at semi-infinity. In this case, in the process to compute MP, we cannot obtain M_iP using the tangent-and-chord operation modulo n, which includes a calculation of the multiplicative inverse of a number not prime to n. However, we can find a prime factor p of n. Hence, if $P_1' = \mathcal{O}_p$ and $P_2' \neq \mathcal{O}_q$, we can find prime factor p.

Denoting the order of point P over E_p by $\Phi_p(P)$, we can rewrite the above condition as follows. That is, if $\Phi_p(P) | M$ and $\Phi_q(P) \not| M$, we can find prime factor p.

We can raise the probability of success by adequately setting M. For example, $M = l!$, where $l \in \mathbf{Z}$ or $M = 2^{e_1}3^{e_2}\cdots p_k{}^{e_k}$, where p_i is the i-th prime and e_i is a positive integer [33][43]. In setting M like this, we can succeed in factorizing n when the largest and the second largest prime factors of $\Phi_p(P)$ are not so large.

In this chapter, we deal with the case of $n = pq$.

3.2.2 Selection of elliptic curves

From the first proposal of an ECM in 1985 [33], many improvements have been proposed. In this section, we review previous selections of elliptic curves.

We assume the following about the distribution of the number of points over elliptic curves in choosing random coefficients.

Assumption 3.1 There exists such a constant $C_{\max}$ and $C_{\min}$ that the probability $\Pr(t)$ that the number of points of elliptic curves defined over $\mathbf{F}_p$ becomes exactly $t(\in [p+1-2\sqrt{p}, p+1+2\sqrt{p}])$ holds that $C_{\min}/\sqrt{p} \leq \Pr(t) \leq C_{\max}/\sqrt{p}$.

From Assumption 3.1, we can regard $\#E_p$ as a almost random number about p. Hence, for a large p, the probability that the largest and the second largest prime factors of $\#E_p$ are small, i.e., the probability of success in the factorization, is very small. We can raise

this probability by choosing the elliptic curve adequately. Montgomery [43] proposed a construction method for elliptic curves whose number of points has a factor 12 or 16. The reason why the probability rises up by choosing such curves is as follows.

> Let $\#E_p$ be the number of points over an elliptic curve. When the elliptic curve is set so that the number has a factor 16, $\#E_p/16$ is regarded as a almost random number. Hence, the randomness of the number decreases from $\#E_p$ to $\#E_p/16$. The probability that the largest and the second largest prime factors of the number of points becomes small is expected to become bigger. Hence, the probability of success in the factorization becomes bigger.

It is written in [64] that setting such elliptic curves reduces the average computation time by 1/1.5.

The following theorem is concerned with the construction of such curves.

Theorem 3.1 The number of points over the following elliptic curve has a factor 16 for an arbitrary prime field $\mathbf{F}_p$.

$$E_p(a,b) : y^2 \equiv x^3 + ax + b \pmod{p}, \text{ where}$$

$a = -27(1 + 32d + 448d^2 + 3584d^3 + 17664d^4 + 51200d^5 + 51200d^6 - 237568d^7 - 1183744d^8 - 1900544d^9 + 3276800d^{10} + 26214400d^{11} + 72351744d^{12} + 117440512d^{13} + 117440512d^{14} + 67108864d^{15} + 16777216d^{16})$

$b = 54(1+48d+1056d^2+14080d^3+126336d^4+795648d^5+3505152d^6+9916416d^7+9922560d^8 - 61276160d^9 - 408551424d^{10} - 1519386624d^{11} - 4565499904d^{12} - 12155092992d^{13} - 26147291136d^{14} - 31373393920d^{15} + 40642805760d^{16} + 324941119488d^{17} + 918854565888d^{18} + 1668594794496d^{19} + 2119566360576d^{20} + 1889785610240d^{21} + 1133871366144d^{22} + 412316860416d^{23} + 68719476736d^{24})$, and $d \in \mathbf{F}_p$.

The following point lies on the above curve.
$P = (3(1 + 16d + 96d^2 + 160d^3 - 832d^4 - 4864d^5 - 9216d^6 - 4096d^7 + 4096d^8), -864d^3(1 + 2d)(1 + 4d)^3(-1 + 8d^2)(1 + 4d + 8d^2))$

Proof. This is easily proven from [3] and [54]. ■

Recently, Izu proposed methods for constructing curves with a bigger factor, such as 27 or 36 [24]. We discuss the ability of extending this idea. At first, on the assumption that there exists an algorithm to select elliptic curves whose orders have a bigger factor than ordinary ones, we discuss to what degree we can reduce an ECM's time.

3.3 New ECM

3.3.1 Outline of a new ECM

We propose a new elliptic curve factoring method. Figure 3.1 shows the outline of this method. First, this method selects adequate elliptic curves (First Step). Second, the method

executes the original ECM (Second Step).

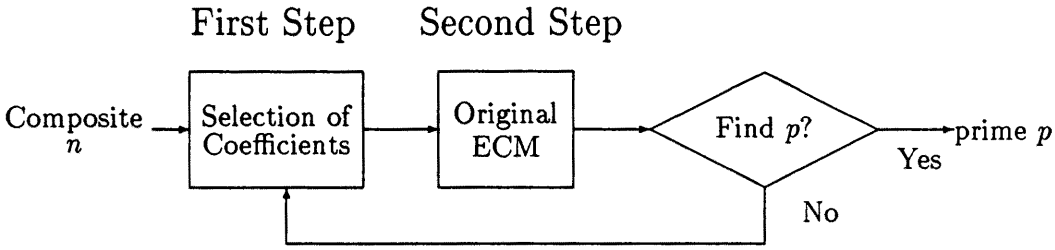

Figure 3.1. Outline of a new ECM

Figure 3.2 shows the outline of a routine to select elliptic curves. First, a random number $d \in Z_n$ is generated. Second, coefficients a and b are calculated over $\mathbf{Z}/n\mathbf{Z}$ based on Theorem 3.1. If $n = pq$, $\#E_n(a,b)$ is a multiple of 16^2 because $\#E_n(a,b) = \#E_p(a,b) \cdot \#E_q(a,b)$. Next, this routine checks whether $D|\#E_n(a,b)$, where D is a product of odd primes less than or equal to r for an adequately chosen r. This routine outputs an elliptic curve whose number of points is a multiple of 16^2D.

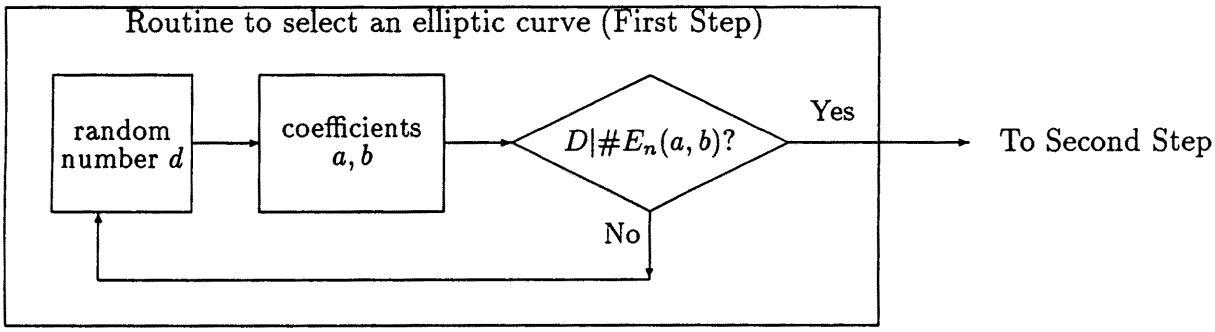

Figure 3.2. Routine to select an adequate elliptic curve

3.3.2 Divisor check algorithm

The routine to select an adequate elliptic curve is formalized as follows.

Routine to select an adequate elliptic curve – Fig. 3.2

Input composite n and $D = 3 \cdot 5 \cdot 7 \cdots r$, where r is adequately chosen.

Output an elliptic curve $E_n(a,b) : y^2 \equiv x^3 + ax + b \pmod{n}$ whose number of points is a multiple of 16^2D

Step 1 Generate a random number $d \in Z_n$.

Step 2 Calculate coefficients a and b from Theorem 3.1.

Step 3 Check whether $\#E_n(a,b) \bmod D = 0$, from now we call this algorithm a divisor check algorithm. If the above is *yes* , output a and b. Otherwise, return to Step 1.

It is an open problem whether there exists an efficient divisor check algorithm. We estimate the computational amount of the new ECM by assuming the existence of a divisor check algorithm whose computational amount is $O(\log^u D \log^v n)$, where u and v are constants.

In this chapter, we call a method without executing Step 3 an *old ECM.*

3.3.3 Estimation of the computation amount of the new ECM

Let us now estimate the computational amount of the new ECM.

Analysis of the computation amount at the First Step

From Assumption 3.1, the probability that the number of points over elliptic curve is divided by D is bigger than $4C_{\min}/D$ and less than $4C_{\max}/D$. Hence, the first step stops at $O(D)$ cycles in an average case. Hence, the computation amount of the first step, $T_1(D)$, is given as follows.

$$T_1(D) = O(D \cdot \log^u D \log^v p) \approx O(e^r r^u \log^v n). \tag{3.1}$$

Note that the latter is led by the approximation equation: $D = C \cdot e^r$, where C is a constant.

Analysis of the computation amount at the Second Step

Since we use the original ECM as the second step, the computation amount of the second step, $T_2(D)$, is given as follows.

$$T_2 = O\left(\exp C_2 \sqrt{(\log p)(\log\log p)}\right), \tag{3.2}$$

where $C_2 = 1/\sqrt{2}$.

Estimation of the number of trial curves

We estimate the necessary number of trial curves to factorize. Let L_1, L_2 be parameters of the ECM, where $L_1 < L_2$. Assume that the number of points over the elliptic curve, $\#E_n$, is factorized as follows.

$$\#E_n = \#E_p \cdot \#E_q = (p_1 p_2 p_3 \cdots)(q_1 q_2 q_3 \cdots), \tag{3.3}$$

where $p_i \geq p_{i+1}$ and $q_i \geq q_{i+1}$. For simplicity, we set $p_1 \leq q_1$. If $L_2 \geq p_1$ and $L_1 \geq p_2$, the ECM can factorize n.

Next, we analyze the probability $\Pr(L_2 \geq p_1 \text{ and } L_1 \geq p_2)$. In analyzing this, we use the following lemma.

Lemma 3.1 (Bach and Peralta [7]) We define $G(\alpha,\beta)$ to be the probability that the largest prime factor of a large number N is less than N^β and the second largest prime factor is less than N^α. If α is the interval $[1/15, 1/10]$ and β is in $[1/9, 1/6]$, $G(\alpha,\beta)$ is approximated as follows.

$$G(\alpha,\beta) = \exp(c_0) \times \alpha^{c_1/\alpha} \times \beta^{c_2/\beta}, \tag{3.4}$$

where $c_0 = 4.55219$, $c_1 = 0.933064$, and $c_2 = 0.280283$.

By using Lemma 3.1, we prove the following theorem. This theorem describes the ratio of probability of success in using the new ECM and original ECM.

Theorem 3.2 Let P_{rnd} be the probability of success when an elliptic curve is randomly chosen. Let $P_{\mathrm{div}}(m)$ be the probability of success when an elliptic curve $E_p(a,b)$ is chosen so that $\#E_p(a,b)$ is divided by m. The ratio of $P_{\mathrm{div}}(m)$ to P_{rnd} is given as follows.

$$\frac{P_{\mathrm{div}}(m)}{P_{\mathrm{rnd}}} = \mathrm{pow}\left(m, \sum_{i=1}^{2} \frac{c_i}{\log L_i} \log \frac{e \log p}{\log L_i}\right), \tag{3.5}$$

where the function $\mathrm{pow}(x,y)$ means x^y.

Proof.

Let $\#E_p$ be the number of points over a randomly chosen elliptic curve and $\#\overline{E_p}$ be the number of points over such an elliptic curve that its number of points is divided by m. From Assumption 3.1, we can regard $\#E_p$ as a random integer about p and $\#\overline{E_p}/m$ as a random integer about p/m.

First, we evaluate P_{rnd}. P_{rnd} is equal to the probability that the largest prime factor of $\#E_p$ is less than L_2 and the second largest prime factor is less than L_1. Letting $\alpha \equiv \frac{\log L_1}{\log p}$ and $\beta \equiv \frac{\log L_2}{\log p}$, we obtain $P_{\mathrm{rnd}} = G(\alpha,\beta)$. Similarly, letting $\overline{\alpha} \equiv \frac{\log L_1}{\log p/m} = \frac{\log L_1}{\log p - \log m}$ and $\overline{\beta} \equiv \frac{\log L_2}{\log p/m} = \frac{\log L_2}{\log p - \log m}$, we obtain $P_{\mathrm{div}}(m) = G(\overline{\alpha},\overline{\beta})$.

By using Lemma 3.1, we evaluate $P_{\mathrm{div}}(m)/P_{\mathrm{rnd}}$.

$$\frac{P_{\mathrm{div}}(m)}{P_{\mathrm{rnd}}} = \frac{G(\overline{\alpha},\overline{\beta})}{G(\alpha,\beta)} = \prod_{i=1}^{2} \left(\frac{\log p - \log m}{\log L_i}\right)^{\frac{c_i \log m}{\log L_i}} \times \prod_{i=1}^{2} \left(1 - \frac{\log m}{\log p}\right)^{-\log p \frac{c_i}{\log L_i}} \tag{3.6}$$

The former part of Eq. (3.6) is transformed into the following by ignoring $\log m$ since $\log p \gg \log m$.

$$\left(\frac{\log p - \log m}{\log L_i}\right)^{\frac{c_i \log m}{\log L_i}} \Rightarrow \mathrm{pow}\left(m, \frac{c_i}{\log L_i} \log \frac{\log p}{\log L_i}\right)$$

The latter part of Eq. (3.6) is transformed into the following by using the approximation equation: $\left(1 + \frac{t}{x}\right)^x \approx e^t$, if x is large.

$$\left(1 - \frac{\log m}{\log p}\right)^{-\log p \frac{c_i}{\log L_i}} \underset{\log p \to \infty}{\to} e^{\log m \frac{c_i}{\log L_i}} = \mathrm{pow}\left(m, \frac{c_i}{\log L_i}\right)$$

Hence, we can approximate $P_{\mathrm{div}}(m)/P_{\mathrm{rnd}}$ as follows.

$$\frac{P_{\mathrm{div}}(m)}{P_{\mathrm{rnd}}} \approx \mathrm{pow}\left(m, \sum_{i=1}^{2} \frac{c_i}{\log L_i} \log \frac{e \log p}{\log L_i}\right)$$

■

Note that $P_{\mathrm{div}}(m)/P_{\mathrm{rnd}}$ in Eq. (3.5) is bigger than 1. This institutional reason is why the function $x^{1/x}$ monotonically increases between the interval $(0,1]$ and $\alpha < \overline{\alpha}, \beta < \overline{\beta}$. The exponent part of Eq. (3.5) can be approximated as the follows.

Theorem 3.3 Let

$$g(p, L_1, L_2) \equiv \sum_{i=1}^{2} \frac{c_i}{\log L_i} \log \frac{e \log p}{\log L_i}.$$

Note that $P_{\mathrm{div}}(m)/P_{\mathrm{rnd}} = \mathrm{pow}(m, g(p, L_1, L_2))$. Suppose that we choose L_1 and L_2 so that $\log L_1 = \sqrt{(\log p)(\log\log p)}/\sqrt{2}, L_2 = 40 \times L_1$, [64]. Then, $g(p, L_1, L_2)$ is approximated as follows.

$$g(p) \equiv g(p, L_1, L_2) \approx C_g \sqrt{\frac{\log\log p}{\log p}}, \tag{3.7}$$

where $C_g = (c_1 + c_2)/\sqrt{2} \approx 0.9$.

We can verify that $g(p)$ lies from 0.23 to 0.19 for $10^{30} \leq p \leq 10^{50}$.

Total computational amount of the new ECM

From the above analyses, the total computational amount of the new ECM becomes as follows.

Theorem 3.4 Let T_2 be the computational amount of the ECM per one curve. Let K be the number of trial curves until success in the factorization when an elliptic curve is randomly chosen. Hence, the computational amount of the original ECM, T_{ORG}, becomes $T_{\mathrm{ORG}} = T_2 \times K$. Furthermore, the computational amount of the old ECM, which chooses an elliptic curve by Theorem 3.1, T_{OLD}, becomes $T_{\mathrm{OLD}} = T_2 \times K/16^{g(p)}$. The computational amount of new ECM to set the optimization parameters as D, T_{NEW}, becomes

$$\begin{aligned} T_{\mathrm{NEW}} &= (T_1(D) + T_2) \times \frac{1}{P_{\mathrm{div}}\left(16\sqrt{D}\right)} \\ &= \left(O\left(D \log^u D \log^v n\right) + O\left(\exp C_2 \sqrt{\log p \log\log p}\right)\right) \times \frac{K}{16^{g(p)} D^{g(p)/2}} \end{aligned} \tag{3.8}$$

Proof. By executing the First Step, $\#E_n$ comes to be divided by $16^2 D$. Hence, we can regard that both $\#E_p$ and $\#E_q$ are divided by natural numbers about $16\sqrt{D}$. The probability of success for the new ECM becomes $P_{\mathrm{div}}\left(16\sqrt{D}\right)$ and the number of trial curves becomes $1/P_{\mathrm{div}}\left(16\sqrt{D}\right)$. Since $P_{\mathrm{rnd}} = 1/K$, the average number of trial curves becomes $K/\ \mathrm{pow}\left(16\sqrt{D}, g(p)\right)$. ■

3.4 How to select optimization parameters

In this section, we discuss how to choose optimization parameters D and r. Note that $D = 3 \cdot 5 \cdot 7 \cdot 11 \cdots r$. Then, we evaluate the computation amount of the new ECM.

3.4.1 Selection of optimization parameter D

With regard to the selection of D, the following theorem holds.

Theorem 3.5 (Selection of D) If D satisfies

$$T_1(D) = \frac{g(p)}{2 - g(p)} T_2, \tag{3.9}$$

Equation (3.8) becomes minimum.

Proof. From Eq. (3.1), we can write $T_1(D) = C_1 D \log^u D$, where C_1 is a positive constant. Hence, we can describe T_{NEW} as follows.

$$T_{\mathrm{NEW}}(D) = (C_1 D \log^u D + T_2) \times \frac{K}{16^{g(p)} D^{g(p)/2}}. \tag{3.10}$$

When $T'_{\mathrm{NEW}}(D) = 0$, $T_{\mathrm{NEW}}(D)$ becomes minimum. By solving $T'_{\mathrm{NEW}}(D) = 0$, we obtain

$$T_2 = \left(\frac{2}{g(p)} - 1\right) C_1 e^r r^u \left(1 + \frac{2u}{(2 - g(p))r}\right) \approx \left(\frac{2}{g(p)} - 1\right) T_1. \tag{3.11}$$

■

In other words, the total computational amount of the new ECM becomes minimum if the selection time for elliptic curves is $g(p)/(2 - g(p))$ times the computational time per one curve. Let D^* be the solution of the equation $T_1(D) = T_2 g(p)/(2 - g(p))$. By using D^*, the ratio of T_{OLD} to T_{NEW} becomes as follows.

$$\frac{T_{\mathrm{OLD}}}{T_{\mathrm{NEW}}} = \frac{2 - g(p)}{2} D^{*g(p)/2}. \tag{3.12}$$

In particular, if $10^{30} \leq p \leq 10^{50}$, the above value is approximated as $0.9 D^{*0.1}$.

Next, we estimate the optimal D^*, or r^*. From [25], T_2 can be written as $T_2 = C_{T_2} \exp \sqrt{(\log p)(\log\log p)/2}$ for some C_{T_2}. Suppose that $T_1(D) = C_{T_1} D \log^u D \log^v n$ for some C_{T_1}. By solving the equation $T_1(D) = T_2 g(p)/(2 - g(p))$, we approximately obtain

$$r^* \approx \log D^* \approx \frac{1}{\sqrt{2}} \sqrt{\log p \log\log p} - \log \frac{2 - g(p)}{g(p)} + \log \frac{C_{T_2}}{C_{T_1}}. \tag{3.13}$$

By substituting D^* into Eq. (3.12), we obtain the following.

Theorem 3.6 The ratio of the computational time of the old ECM, T_{OLD}, to the new ECM, T_{NEW}, becomes as follows.

$$\frac{T_{\mathrm{OLD}}(p)}{T_{\mathrm{NEW}}(p)} = \frac{2 - g(p)}{2} \left(\frac{C_{T_2}}{C_{T_1}}\right)^{\frac{g(p)}{2}} \left(\frac{2}{g(p)} - 1\right)^{\frac{g(p)}{2}} (\log p)^{0.318} \overset{p \to \infty}{\to} (\log p)^{0.318}. \tag{3.14}$$

Proof. $D^{*g(p)/2}$ is transformed as follows.

$$D^{*g(p)/2} = \exp(r^* g(p)/2) = \left(\frac{C_{T_2}}{C_{T_1}}\right)^{\frac{g(p)}{2}} \left(\frac{2}{g(p)} - 1\right)^{\frac{g(p)}{2}} \exp\left(\frac{0.9}{2\sqrt{2}} \log\log p\right) \to (\log p)^{0.318}.$$

■

3.4.2 Efficiency of the new ECM

We estimate the optimal r^* and D^* from Eq. (3.13) for the proper p. Furthermore, we calculate $T_{\mathrm{OLD}}/T_{\mathrm{NEW}}$ and $T_{\mathrm{ORG}}/T_{\mathrm{NEW}}$. We cannot obtain correct values for C_{T_1} and C_{T_2}, which depend on the implementation. Hence, for simplicity, we assume that $C_{T_1} = C_{T_2}$. Table 3.1 shows the optimal r^* and a ratio of the old ECM to new ECM, and a ratio of the original ECM to the new ECM.

Table 3.1. Optimal r^* and ratios of computation time

p	10^{10}	10^{20}	10^{30}	10^{40}	10^{50}	10^{60}	10^{70}	10^{80}	10^{90}	10^{100}
r^*	5	7	11	13	13	17	17	19	23	23
$T_{\mathrm{OLD}}/T_{\mathrm{NEW}}$	1.7	2.3	2.7	3.0	3.3	3.6	3.8	4.0	4.2	4.4
$T_{\mathrm{ORG}}/T_{\mathrm{NEW}}$	4.4	4.7	5.0	5.3	5.5	5.7	5.9	6.1	6.3	6.5

We use Eq. (3.13) in setting r^* and D^* to simplify the analysis although we *should* use Eq. (3.9) to be exact. In this case, the optimal r^* and the ratios slightly vary from Table 3.1.

We can easily verify that the new ECM is three or four times faster than the old ECM. In addition, for adequate prime factors such as $10^{30} \leq p \leq 10^{50}$, we should set $r = 11$ or $r = 13$.

3.5 Conclusion

We theoretically proved that on the assumption that there exists an algorithm to select elliptic curves whose orders have a big factor, we can decrease the computational amount of the elliptic curve factoring method. As a result, we proved that the new ECM is three or four times faster than the old ECM. Furthermore, we proved that the new ECM is $(\log p)^{0.318}$ times faster than the old ECM asymptotically for a prime factor p. It is an open problem whether there exists an efficient Divisor Check Algorithm or not.

Chapter 4

Difficulty of counting the number of points on elliptic curve over the ring $\mathbf{Z}/n\mathbf{Z}$

For composite n, we prove that counting the number of points on elliptic curves over the ring $\mathbf{Z}/n\mathbf{Z}$ is randomly computationally equivalent to factoring n. That is, we prove that if we can count it, we can easily factor n. Furthermore, we also prove that if we can solve the elliptic curve discrete logarithm problem modulo n, we can easily factor n.

4.1 Introduction

Elliptic curves can be applied to public-key cryptosystems, and as such several schemes have been proposed [14][26][27][28][39][41]. There are two typical elliptic curve cryptosystems: ElGamal-type scheme [26][41] and RSA-type schemes [14][27][28]. The security of the ElGamal-type elliptic curve cryptosystem is based on the difficulty of solving a discrete logarithm over elliptic curve modulo a prime. However, the security of an RSA-type elliptic curve cryptosystem is based on the difficulty of factoring a large composite. It has been conjectured that completely breaking the original RSA is computationally equivalent to the factoring the used composite, although this has NOT been proved yet. In a certain RSA-type elliptic curve (or cubic curve) cryptosystem proposed in [28], however, this equivalence between the two problems was proved. In general RSA-type elliptic curve cryptosystems, including RSA-type cubic curve cryptosystems, the equivalence has not been proved. As the order $\phi(n)$ of $(\mathbf{Z}/n\mathbf{Z})^*$ for a composite n have played a significant role in analyzing the security of the original RSA scheme, it is important to evaluate the complexity of counting the number of points on an elliptic curve over the ring $\mathbf{Z}/n\mathbf{Z}$ for RSA-type elliptic curve cryptosystems.

We are interested in reductions of factoring to other problems in elliptic curve theory over $\mathbf{Z}/n\mathbf{Z}$. In this chapter, we will consider the following problems.

FCT(n) : Given composite n, find the complete prime factorization of n.

COMP($\phi(n)$) : Given composite n, compute the Euler phi function $\phi(n) = |(\mathbf{Z}/n\mathbf{Z})^*|$, which is the number of integers in the interval $[1, n]$, each of which are relatively prime to n.

COMP($\#E_n(a,b)$) : Given composite n and integers a and b, compute $\#E_n(a,b)$, which is the number of points over an elliptic curve $E_n(a,b) : y^2 \equiv x^3 + ax + b \pmod{n}$.

COMP($\#E_n(a,b)$ mod d) : Given composite n, an elliptic curve $E_n(a,b)$ and prime d ($= O(\log n)$), compute $\#E_n(a,b) \mod d$.

ECDLP mod p (Elliptic Curve Discrete Logarithm Problem mod p): Given *a prime* p, an elliptic curve E_p and two points G and A over E_p, find the positive integer α such that $\alpha G = A$.

ECDLP mod n (Elliptic Curve Discrete Logarithm Problem mod n): Given *a composite* n, an elliptic curve E_n and two points G and A over E_n, find the positive integer α such that $\alpha G = A$.

We would like to emphasize that for a prime p (not a composite), COMP($\#E_p(a,b)$) and COMP($\#E_p(a,b) \mod d$) are computable in polynomial time [60]. Conversely, it is not known whether an algorithm exists for solving an ECDLP mod p in polynomial time (or even in sub-exponential time).

The purpose of this chapter is to show possible reductions between the factoring problem and some problems in elliptic curve theory over $\mathbf{Z}/n\mathbf{Z}$. We prove the equivalence of FCT(n) and COMP($\#E_n(a,b)$) in Section 4.3 and prove the equivalence of FCT(n) and COMP($\#E_n(a,b) \mod d$) in Section 4.4. Finally, we prove that FCT(n) is randomly polynomial time reducible to ECDLP mod n in Section 4.5.

4.2 Preliminaries

First, we define the notation used in the computational relationships and then describe some of the previous work relating to the original RSA scheme. The overview of elliptic curve over the ring $\mathbf{Z}/n\mathbf{Z}$ and elliptic curve factoring method are written at Sections 2.1.3 and 3.3.1, respectively.

4.2.1 Notations of Computational Relationship

Let A, B and C be computational problems. We can define computational relationship among these problems in the following terms: $A \leq_P B$, $A \leq_{RP} B$, $A =_P B$, $A =_{RP} B$ and $A \leq_P B \oplus C$.

Definition 4.1 We say that problem A is *polynomial time reducible* to problem B, written as $A \leq_P B$, if there is an algorithm that solves A which uses *an oracle* (or a subroutine) for B, and the algorithm runs in polynomial time.

Definition 4.2 We say that problem A is *randomly polynomial time reducible* to problem B, written as $A \leq_{RP} B$, if there is an algorithm that solves A which uses *an oracle* (or a subroutine) for B, and the algorithm runs in randomly polynomial time.

Roughly speaking, $A \leq_P B$ or $A \leq_{RP} B$ implies that if solving B is easy, then solving A is also easy.

Definition 4.3 If $A \leq_P B$ and $B \leq_P A$, then we say that A and B are *computationally equivalent*, written as $A =_P B$.

Definition 4.4 If $\{A \leq_P B \text{ and } B \leq_{RP} A\}$, or $\{A \leq_{RP} B \text{ and } B \leq_P A\}$, or $\{A \leq_{RP} B \text{ and } B \leq_{RP} A\}$, then we say that A and B are *randomly computationally equivalent*, written as $A =_{RP} B$.

Definition 4.5 We say that A is *polynomial time reducible* to B and C, written as $A \leq_P B \oplus C$, if there is an algorithm that solves A which uses both *an oracle* for B and *an oracle* for C, and the algorithm runs in polynomial time.

4.2.2 Previous Work

For COMP($\phi(n)$) and FCT(n), the following two facts are widely known.

Fact 4.1 Let n be a composite that is a product of distinct odd primes. On the assumption that *Extended Riemann Hypothesis* (ERH) is true, it holds that [40]

$$\text{COMP}(\phi(n)) =_P \text{FCT}(n).$$

Without the ERH assumption, it holds that [36]

$$\text{COMP}(\phi(n)) \leq_P \text{FCT}(n) \text{ and } \text{FCT}(n) \leq_{RP} \text{COMP}(\phi(n)),$$

i.e.,

$$\text{FCT}(n) =_{RP} \text{COMP}(\phi(n)).$$

Fact 4.2 For composite n' that is a product of two distinct odd primes p and q, it holds that $\text{COMP}(\phi(n')) =_P \text{FCT}(n')$.

Proof. Using an oracle for FCT(n'), we can obtain p and q. Therefore, we can obtain $\phi(n')$ by computing $(p-1)(q-1)$. Hence, it follows that $\text{COMP}(\phi(n')) \leq_P \text{FCT}(n')$. Conversely, using an oracle for COMP($\phi(n')$), we can obtain $p+q$ by using the fact that $p+q = n' + 1 - \phi(n')$. Thus, p and q can be determined by solving the quadratic equation: $x^2 - (n' + 1 - \phi(n'))x + n' = 0$ over a real field R. This leads to

$$x = \frac{n' + 1 - \phi(n') \pm \sqrt{(n' + 1 - \phi(n'))^2 - 4n'}}{2},$$

from which it follows that FCT(n') $\leq_P$ COMP($\phi(n')$). Thus, COMP($\phi(n')$) and FCT(n') are computationally equivalent, COMP($\phi(n')$) $=_P$ FCT(n'). ■

Woll [69] studied the reductions between numerous number theoretic problems, including FCT(n) and COMP($\phi(n)$).

4.2.3 Generalized KMOV scheme

We describe generalized KMOV scheme and show that it is important to study the reduction between FCT(n) and COMP($\#E_n(a,b)$).

The generalized KMOV scheme is described as follows.

generalized KMOV scheme

Public key: $(e, n(=pq))$, where p and q are distinct primes and $E_n(a,b) : y^2 \equiv x^3 + ax + b \pmod{n}$

Secret key: $(d,(p,q))$, where $ed \equiv 1 \pmod{\#E_n(a,b)}$

Plaintext: $M \in E_n(a,b)$

Encryption: Compute $C = eM$ over $E_n(a,b)$

Ciphertext: $C \in E_n(a,b)$

Decryption: Compute $M = dC$ over $E_n(a,b)$

The encryption and decryption of the generalized RSA scheme work by the following reason. Since $ed \equiv 1 \pmod{\#E_n(a,b)}$, there exists an integer k such that $ed = 1 + k\#E_n(a,b)$. By Lagrabge's theorem, in decryption, dC over $E_n(a,b)$ is transformed as follows.

$$dC = d(eM) = deM = (1 + k\#E_n(a,b))M = M + k(\#E_n(a,b)M) = M.$$

Remark 4.1 If $p \equiv 2 \pmod 3$, $q \equiv 2 \pmod 3$, and $a = 0$, this scheme becomes KMOV scheme. By setting as the above, people who know p and q can easily compute $\#E_n$. In this case, $\#E_n = (p+1)(q+1)$.

If we can factorize n, we can break the generalized scheme. In addition, if we can obtain $\#E_n(a,b)$, we can break this scheme. However, it is not known what relationship FCT(n) and COMP($\#E_n(a,b)$) have.

KMOV scheme has following two good properties.

- KMOV scheme is stronger than RSA scheme against the Håstad's attack [23].
- Even if an algorithm which solves an elliptic curve discrete logarithm problem efficiently is found, KMOV scheme remains still secure. This means that even if ECC is broken, KMOV scheme remains still secure.

4.3 FCT(n) and COMP($\#E_n(a,b)$)

In this section, we examine FCT(n) and COMP($\#E_n(a,b)$). We shall prove Theorem 4.1 (described below). The following is well known.

Fact 4.3 Let $n(=\prod_{i=1}^{k} p_i^{e_i})$ be a product of odd primes, where all of $p_i(\neq 2,3)$ are distinct odd primes and all of e_i are positive integers. It holds that

$$\mathrm{COMP}(\#E_n(a,b)) \leq_P \mathrm{FCT}(n).$$

Proof. Using an oracle for FCT(n), we can obtain prime factorization of n as $n = \prod_{i=1}^{k} p_i^{e_i}$. We can compute $\#E_{p_i}$ for each p_i by Schoof algorithm [60] and obtain $\#E_n = \prod_{i=1}^{k} \#E_{p_i} p_i^{e_i-1}$. Since k is less than $\log_2 n$, we can obtain $\#E_n$ in polynomial time. ■

We will now prove the converse of Fact 4.3.

Theorem 4.1 Under the same condition of Fact 4.3, it holds that

$$\mathrm{FCT}(n) \leq_{RP} \mathrm{COMP}(\#E_n(a,b)) \text{ and } \mathrm{FCT}(n) =_{RP} \mathrm{COMP}(\#E_n(a,b)).$$

Proof. First, we show an algorithm for computing $n'(=\prod_{i=1}^{k} p_i)$ from $n = \prod_{i=1}^{k} p_i^{e_i}$ by using an oracle for COMP$(\#E_n(a,b))$.

Input: Composite $n(=\prod_{i=1}^{k} p_i^{e_i})$

Output: Composite $n'(=\prod_{i=1}^{k} p_i)$, which is a product of distinct prime.

Step 1 Randomly set elliptic curve $E_n(a,b) : y^2 \equiv x^3 + ax + b \pmod{n}$ satisfying $\mathrm{GCD}(n, 4a^3 + 27b^2) = 1$.

Step 2 Using an oracle for COMP$(\#E_n(a,b))$, obtain $\#E_n(a,b)$.

Step 3 Output $\dfrac{n}{\mathrm{GCD}(n, \#E_n(a,b))}$. This value becomes n' with high probability.

The following algorithm can factorize n' in randomly polynomial time using an oracle for COMP$(\#E_{n'}(a,b))$. Once we obtain $p_1, p_2, \ldots, p_k$, we can easily determine $e_1, e_2, \ldots, e_k$.

Factoring Algorithm using an oracle for COMP$(\#E_n(a,b))$

Input: Composite $n(=\prod_{i=1}^{k} p_i)$

Output: Prime factors $p_1, p_2, \cdots, p_k$

Step 1 Set a parameter S and set elliptic curve $E_n(a,b) : y^2 \equiv x^3 + ax + b \pmod{n}$ satisfying $\mathrm{GCD}(n, 4a^3 + 27b^2) = 1$ and a point P over E_n.

1.1 Set S that is the largest prime less than $\lfloor \log n \rfloor$.

1.2 Set the point $P = (x_0, y_0)$ randomly and set a randomly.

1.3 Calculate $b = {y_0}^2 - {x_0}^3 - ax_0$.

Step 2 Using an oracle for COMP($\#E_n(a,b)$), obtain $\#E_n(a,b)$.

Step 3 Check the divisibility of $\#E_n(a,b)$ by S. If $S|\#E_n$ and $S^2 \not| \#E_n$, proceed to step 4. Otherwise, return to step 1.2.

Step 4 Set $M \equiv \frac{\#E_n(a,b)}{S}$ and try to compute MP over $E_n(a,b)$. Suppose that MP is calculated as $M_1P(=P), M_2P, \ldots, M_lP(=MP)$.

- If at least one M_iP is a *semi-zero point*, then we can find a prime factor p_i. When n/p_i is a prime, factoring is completed. When n/p_i is a composite, set $n = n/p_i$, return to step 1.
- If MP is a *zero point*, then we cannot find a prime factor. Return to step 1.2.

The following analysis leads us to the conclusion that the above algorithm runs in randomly polynomial time $O((\log n)^5)$.

We use properties of *elliptic curve* and *elliptic curve factoring method* [33] described in Chapter 2. Let $\Phi_{p_i}(P)$ be the order of point P over E_{p_i}. In the elliptic curve method, the order of points plays an important role. Note that, in the above algorithm, we only assumed an oracle to compute the number of points: $\#E_n$, instead of an oracle to compute the order of a point.

First, we will prove that the point MP in step 4 always becomes a zero point or a semi-zero point. In general, mP is a zero point if $\Phi_{p_i}(P)|m$ for all i, and mP is a semi-zero point if $\Phi_{p_i}(P)|m$ for at least one i and mP is not a zero point. Note that $\Phi_{p_i}(P)|\#E_{p_i}$ and $\#E_{p_i}|\#E_n$ and $\Phi_{p_i}(P)|\#E_n$ for all i and all P over E_n. Let p_1 be the prime factor that satisfies $S|\#E_{p_i}$. Since $S|\#E_n$ and $S^2 \not| \#E_n$, such p_1 uniquely exists. Since $S|\#E_{p_1}$ and $S \not| \#E_{p_i}$ for $2 \leq i \leq k$, M is denoted by $M = \frac{\#E_{p_1}}{S} \prod_{i=2}^{k} \#E_{p_i}$ and then $\#E_{p_i}|M$ for $2 \leq i \leq k$. Note that $\#E_{p_1} \not| M$ since $\#E_{p_1}$ is a multiple of S and M is not a multiple of S. Letting $P = \langle P_1, P_2, \ldots, P_k \rangle$, MP is denoted as $MP = \langle MP_1, MP_2, \ldots, MP_k \rangle$. Since $\Phi_{p_i}(P)|\#E_{p_i}$, then $\Phi_{p_i}(P)|M$ and $MP_i = \mathcal{O}_{p_i}$ for $2 \leq i \leq k$ and all P over E_n. Hence MP is a zero point or a semi-zero point because $\Phi_{p_i}(P)|M$ for at least one i.

This classification of the above two cases depends on whether MP_1 is a zero point $\mathcal{O}_{p_1}$ or an ordinary point. This dependency corresponds to the divisibility of $\Phi_{p_1}(P)$ by S as follows. If $S|\Phi_{p_1}(P)$, we can write $M = \frac{\Phi_{p_1}(P)}{S} \cdot C$, where C is not a multiple of S. Since M is not a multiple of $\Phi_{p_1}(P)$, MP_1 is not a zero point. Hence, MP is a semi-zero point if $S|\Phi_{p_1}(P)$. If $S \not| \Phi_{p_1}(P)$, we can write $M = \Phi_{p_1}(P) \cdot C$, where C is not a multiple of S. Since M is a multiple of $\Phi_{p_1}(P)$, MP_1 is a zero point $\mathcal{O}_{p_1}$. Hence, MP is a zero point if $S \not| \Phi_{p_1}(P)$.

Next, we will evaluate the probability of passing step 3. If R is a uniformly distributed random number, then the probability that $S^j|R$ is $\frac{1}{S^j}$, where j is a small integer and S is a prime. For prime p and randomly chosen integers a and b, the value of $\#E_p(a,b)$ behaves

as a pseudo random number. Strictly speaking, $\#E_p(a,b)$ is not uniformly distributed in the range $p - 2\sqrt{p} + 1 \leq \#E_p(a,b) \leq p + 2\sqrt{p} + 1$. However, we put the assumption: $\Pr\{S^j | \#E_p(a,b)\} = \frac{1}{S^j}$. The probability that $S|\#E_n$ and $S^2 \not| \#E_n$ is equal to the probability that there exists p_i such that $S|\#E_{p_i}$ and $S^2 \not| \#E_{p_i}$ and $S \not| \#E_{p_j}$ for all $j(\neq i)$. Since $\Pr\{S|\#E_{p_i}$ and $S^2 \not| \#E_{p_i}\}$ is equal to $\frac{1}{S}(1 - \frac{1}{S})$ and $\Pr\{S \not| \#E_{p_j}\}$ is equal to $(1 - \frac{1}{S})$ for each j on the above assumption,

$$\Pr\{S|\#E_n \text{ and } S^2 \not| \#E_n\} = \frac{1}{S}\left(1 - \frac{1}{S}\right) \cdot \left(1 - \frac{1}{S}\right)^{k-1} \cdot k = \left(1 - \frac{1}{S}\right)^k \frac{k}{S}.$$

Next, we will evaluate the probability of finding a prime factor in step 4. From the previous analysis, this probability is equal to the probability that a random point P over E_{p_1} satisfies that $S|\Phi_{p_1}(P)$. This probability is given as $1 - \frac{1}{S}$.

Hence, the probability $Q(k,S)$ of finding a prime factor per curve is given by

$$Q(k,S) \equiv \left(1 - \frac{1}{S}\right)^{k+1} \frac{k}{S}. \tag{4.1}$$

Note that the average number of trial curves is given by $1/Q(k,S)$.

By theoretical analysis, we find that $Q(k,S)$ is monotonically increasing in $k \leq S$, for a given S. From this property and that $k \leq \log n \approx S$, an integer which minimizes $Q(k,S)$ is $k = 2$. In this case the probability $Q(k, \log n) \geq Q(2, \log n) = (1 - \frac{1}{\log n})^3 \frac{2}{\log n} = O(\frac{1}{\log n})$. Hence, the average number of trial curves needed to find one prime factor is less than $O(\log n)$. Since the number of prime factors is less than $\log n$, the average number of total trial curves to find the complete prime factors is $O((\log n)^2)$.

Next, we will determine the computation amount per curve. Computing MP needs $O(\log M)$ group operations. Since $M \leq \#E_n \approx n$, the number of group operations is $O(\log n)$. It is known that the computation amount per group operation is $O((\log n)^2)$. These results lead to the conclusion that the above algorithm runs in randomly polynomial time $O((\log n)^5)$. Thus, $\mathrm{FCT}(n) \leq_{RP} \mathrm{COMP}(\#E_n(a,b))$ has been proven.

This property and Fact 3 ($\mathrm{COMP}(\#E_n(a,b)) \leq_P \mathrm{FCT}(n)$) imply
$\mathrm{FCT}(n) =_{RP} \mathrm{COMP}(\#E_n(a,b))$. ■

We show a simple example.

Example 4.1: Let $n = 22657$. When we set $S = 5$, $a = 22651$ and $b = 9310$, we have $\#E_n = 28688$ and $5 \not| \#E_n$. This case fails at step 3. When we set $S = 5$, $a = 20837$ and $b = 8047$, we have $\#E_n = 26400$ and $5^2 | \#E_n$. This case also fails at step 3. Next, we show an example of success. In step 1, we set $S = 5$ and $a = 18405, b = 18024$ and $P = (3926, 16206)$. In step 2, we obtain $\#E_n(a,b) = 22080$ by an oracle. In step 3, we confirm that $5|\#E_n$ and $5^2 \not| \#E_n$. In step 4, since $M \equiv \frac{\#E_n}{5} = 4416$, we compute MP over $E_{22657}(18405, 18024)$ by using a binary method, which is calculated successively as

$$P, 2P, 4P, 8P, 16P, 17P, 34P, 68P, 69P = (15499, 8896).$$

However, $138P$ cannot be calculated by tangent and chord operation because $138P$ is a *semi-zero* point (note that 8896 is not relatively prime to 22657). Since GCD$(8896, n(=22657)) = 139$, factoring is thus completed.

As a result $n = 22657$ is factored by $n = 22657 = pq = 139 \times 163$. Note that $\Phi_p(P) = 138, \Phi_q(P) = 10$ and $M = 4416$ imply $\Phi_p(P)|M$ and $\Phi_q(P) \not| M$, and that factoring is successful (refer to Section. 3.2.1). The reason why factoring is done in computing $138P$ is that "$138P$" occurs while in the process of computing $4416P$ and "138" happens to satisfy $\Phi_p(P)|138$ and $\Phi_q(P) \not| 138$. We may note that $\#E_n = \#E_p \times \#E_q = 138 \times 160 = 2^6 \times 3 \times 5 \times 23$.

Remark 4.2 It may be interesting to compare Fact 4.1 and Theorem 4.1. If ERH is true, then FCT$(n) \leq_P$ COMP$(\phi(n))$ and FCT$(n) \leq_{RP}$ COMP$(\#E_n(a,b))$. Without the assumption that ERH is true, FCT$(n) \leq_{RP}$ COMP$(\phi(n))$ and FCT$(n) \leq_{RP}$ COMP$(\#E_n(a,b))$.

Remark 4.3 We check whether $S|\#E_n$ and $S^2 \not| \#E_n$ in step 3 of the factoring algorithm in the proof of Theorem 4.1. We can construct an algorithm to rule out the condition: $S^2 \not| \#E_n$ in step 3. Since the success probability of reconstructed algorithm is larger than that of the previous algorithm, this algorithm runs in randomly polynomial time. Note that the factoring algorithm in the proof of Theorem 4.1, in which analysis of success probability is easier, also runs in randomly polynomial time.

Remark 4.4 In the factoring algorithm in the proof of Theorem 4.1, we set one S and use *several* trial curves to factorize. Moreover, we can construct a dual algorithm with *one* curve by using *several* primes S, ranging from 2 to B. We define "partially B-smooth" number as the integer whose *smallest*[1] prime factor is less than B. Step 3 and step 4 of the dual algorithm are as follows.

Step 3 If $\#E_n(a,b)$ is *partially B-smooth*, find prime factors less than B of $\#E_n$ by trial division. Otherwise, return to step 1.2. Let's denote $\#E_n = {q_1}^{e_1}{q_2}^{e_2}\cdots{q_s}^{e_s} \times R$, where $q_1, q_2, \cdots, q_s$ are all primes less than B with $e_i \geq 0$ and R is not partially B-smooth.

Step 4 Let $\widetilde{M_{i,j}} = \#E_n(a,b)/{q_i}^j$ for $1 \leq i \leq s$ and $1 \leq j \leq e_i$. Compute $\widetilde{M_{i,j}}P$. If there exists $\widetilde{M_{i,j}}P$ which is a semi-zero point, factoring is successful. Otherwise, i.e., if all $\widetilde{M_{i,j}}P$ are zero points, factoring fails.

We can approximately estimate the probability of success. The probability T that $\#E_n(a,b)$ becomes partially B-smooth is denoted by $T = 1 - \prod_{i=1}^{s}(1 - \frac{1}{q_i})$ if $B \ll \#E_n$. From Mertens's Theorem [56], T is approximated as $1 - \frac{e^{-\gamma}}{\log B}$, where γ is Euler's constant. If B is sufficiently large, we have $T \approx 1$. The probability that there exists q_i with $e_i = 1$ $(1 \leq i \leq s)$ is almost 1. Let q^* be the largest of the primes. The probability U to succeed in factoring in step 4 is more than $1 - \frac{1}{q^*}$. When B is sufficiently large, $U \approx 1$. Hence, the

[1] B-smooth number is an integer whose *biggest* prime factor is less than B.

number of expected trial curves is almost 1. Let $|\widetilde{M_{i,j}}|$ be the number of combination of suffix of $\widetilde{M_{i,j}}$, where $|\widetilde{M_{i,j}}| = \sum_{i=1}^{s} e_i$. Since the average of e_i is given as $\frac{1}{q_i - 1}$, the average of $|\widetilde{M_{i,j}}|$, denoted by $\overline{|\widetilde{M_{i,j}}|}$, is given by the $\overline{|\widetilde{M_{i,j}}|} = \sum_{i=1}^{s} \frac{1}{q_i - 1} \approx \log\log B$ [56]. From $B \ll n$, we have $\overline{|\widetilde{M_{i,j}}|} < \log\log n$. Hence, the computing all $\widetilde{M_{i,j}}P$ is completed in polynomial time. Since $T \approx 1$ and $U \approx 1$, this reconstructed dual algorithm also runs in randomly polynomial time.

In several variants of RSA-type cryptosystems, the multiple of $\phi(n)$, instead of $\phi(n)$ itself, can be easily known from public information. These schemes are insecure because Miller [40] proved that if the multiple of $\phi(n)$ is known, n can be easily factored. In relation to this fact, we prove the following.

Lemma 4.1 Let a value $r\#E_n(a,b)$ be the multiple of $\#E_n(a,b)$, where r is randomly distributed and independent of n, a and b. For composite n that is a product of odd primes, it holds that

$$\mathrm{FCT}(n) \leq_{RP} \mathrm{COMP}(r\#E_n(a,b)).$$

Proof. We can prove this lemma by revising the previous "Factoring Algorithm". This is achieved by replacing an oracle for $\mathrm{COMP}(\#E_n(a,b))$ with an oracle for $\mathrm{COMP}(r\#E_n(a,b))$ in step 2 of the proof of Theorem 4.1. Note that the above assumption about r implies $\Pr\{S^j|r\} = \frac{1}{S^j}$. From this property and $\Pr\{S^j|\#E_{p_i}\} = \frac{1}{S^j}$ for each i and j from the assumption described above, the probability that $S|r\#E_n$ and $S^2 \not| r\#E_n$ is given as $(1 - \frac{1}{S})^{k+1}\frac{k+1}{S}$. The probability of succeeding of finding a prime factor in step 4 is $1 - \frac{1}{S}$ as well as $\mathrm{COMP}(\#E_n(a,b))$. Hence, the probability of successful factoring is $(1 - \frac{1}{S})^{k+2}\frac{k+1}{S}$. As well as the analysis of $\mathrm{COMP}(\#E_n(a,b))$, this revised algorithm finds complete prime factors in randomly polynomial time $\mathrm{O}((\log n)^5)$. ■

4.4 FCT(n) and COMP($\#E_n(a,b) \mod d$)

In this section, we present reductions between two problems,

$$\mathrm{FCT}(n) \text{ and } \mathrm{COMP}(\#E_n(a,b) \mod d).$$

The computation amount of the problem $\mathrm{COMP}(\#E_p(a,b) \mod d)$ is known to be equal $\mathrm{O}((\log p)^3 d^5)$, where p is a prime (not a composite) [60]. Hence, if $d = \mathrm{O}(\log p)$, then $\mathrm{COMP}(\#E_p(a,b) \mod d)$ is computable in polynomial time. The Schoof algorithm[2] for calculating $\#E_p(a,b)$ runs in polynomial time $\mathrm{O}((\log p)^8)$ by using $\mathrm{O}(\log p)$ times of $\mathrm{COMP}(\#E_p(a,b) \mod d)$. However, Theorem 4.2 states that $\mathrm{COMP}(\#E_n(a,b) \mod d)$ is not as easy as $\mathrm{FCT}(n)$. Consequently, before proving Theorem 4.2, we must establish the following lemma.

[2] Charlap, Coley and Robbins [10] showed an algorithm which calculates $\#E_p(a,b)$ in $\mathrm{O}((\log p)^6)$ by revising Schoof algorithm.

Lemma 4.2 Let $n(=\prod_{i=1}^{k} p_i^{e_i})$ be a product of odd primes, where all of $p_i(\neq 2,3)$ are distinct odd primes and all of e_i are positive integers. It holds that

$$\mathrm{COMP}(\#E_n(a,b)) \leq_P \mathrm{COMP}(\#E_n(a,b) \mod d).$$

Proof. Using the following algorithm, $\#E_n(a,b)$ is computable in polynomial time.

Counting the number of points ($\#E_n(a,b)$) Algorithm using an oracle for $\mathrm{COMP}(\#E_n(a,b) \mod d)$

Input: Composite $n(=\prod_{i=1}^{k} p_i^{e_i})$, elliptic curve E_n: $y^2 \equiv x^3 + ax + b \pmod n$

Output: $\#E_n(a,b)$

Step 1 Choose a number L such that $(n+1)^2 \leq \prod l$, where the product ranges over all primes l between 3 and L.

Step 2 Compute $\tau_l = \#E_n(a,b) \mod l$ by using an oracle for $\mathrm{COMP}(\#E_n(a,b) \mod l)$ for $l = 3,5,7,11,\cdots,L$, respectively.

Step 3 Solve the system of the congruences: $\#E_n \equiv \tau_l \pmod l$ by the Chinese Remainder Theorem, where l is all the primes between 3 and L.

The following analysis leads to the conclusion that this algorithm runs in polynomial time $\mathrm{O}((\log n)^2)$.

Since $L \approx \log(n+1)^2$, we have $L = \mathrm{O}(\log n)$. Therefore, obtaining each τ_l is executable by an oracle for $\mathrm{COMP}(\#E_n(a,b) \mod l)$. Since step 2 consists of at most L iterations to obtain τ_l, step 2 is completed in polynomial time. Since step 3 is computable in polynomial time $\mathrm{O}((\log n)^2)$, it follows from the above discussion that this algorithm completes in polynomial time to compute $\#E_n(a,b)$. ■

Theorem 4.2 Let $n(=\prod_{i=1}^{k} p_i^{e_i})$ be a product of odd primes, where all of $p_i(\neq 2,3)$ are distinct odd primes and all of e_i are positive integers. It holds that

$$\mathrm{FCT}(n) =_{RP} \mathrm{COMP}(\#E_n(a,b) \mod d).$$

Proof. We have $\mathrm{FCT}(n) \leq_{RP} \mathrm{COMP}(\#E_n(a,b) \mod d)$ from Theorem 4.1 and Lemma 4.2 because it satisfies that $\mathrm{FCT}(n) \leq_{RP} \mathrm{COMP}(\#E_n(a,b)) \leq_P \mathrm{COMP}(\#E_n(a,b) \mod d)$. In addition,

$$\mathrm{COMP}(\#E_n(a,b) \mod d) \leq_P \mathrm{FCT}(n),$$

which is trivially proven, and $\mathrm{FCT}(n) \leq_{RP} \mathrm{COMP}(\#E_n(a,b) \mod d)$ imply the latter part of Theorem 4.2, $\mathrm{FCT}(n) =_{RP} \mathrm{COMP}(\#E_n(a,b) \mod d)$. ■

4.5 FCT(n) and ECDLP mod n

In this section, we will present the reductions between FCT(n) and the elliptic curve discrete logarithm problem modulo n, ECDLP mod n. ECDLP mod n is analogously related with DLP mod n, the ordinary discrete logarithm problem modulo n. Letting DLP mod p be an ordinary discrete logarithm problem modulo prime p, it is known that FCT(n) $\leq_{RP}$ DLP mod n and DLP mod $n \leq_P$ FCT(n) $\oplus$ DLP mod p [4]. Similar reductions also hold in FCT(n) and ECDLP mod n.

Theorem 4.3 Let $n(= \prod_{i=1}^{k} p_i^{e_i})$ be a product of odd primes, where all of $p_i(\neq 2, 3)$ are distinct odd primes and all of e_i are positive integers. It holds that

$$\text{FCT}(n) \leq_{RP} \text{ECDLP mod } n.$$

Proof. We will present a factoring algorithm using an oracle for ECDLP mod n. This algorithm is constructed by revising the factoring algorithm described in the proof of Theorem 4.1. By similar discussion as in Theorem 4.1, we consider the case that n is a product of distinct primes.

Factoring Algorithm using an oracle for ECDLP mod n

Input: Composite $n(= \prod_{i=1}^{k} p_i)$

Output: k prime factors $p_1, p_2, \ldots, p_k$

Step 1 Set a parameter S and set elliptic curve $E_n(a, b) : y^2 \equiv x^3 + ax + b$ (mod n) and a point P over E_n.

Step 2 Choose a random integer α and compute $-\alpha P$.

Step 3 Compute the smallest positive integer M'' satisfying $M''P = -\alpha P$ using an oracle for ECDLP mod n. Set $M' = M'' + \alpha$.

Step 4 If $S|M'$ and $S^2 \not| M'$, proceed to the next step. Otherwise, return to step 1.

Step 5 Set $M \equiv \dfrac{M'}{S}$ and compute MP.

- If MP is an ordinary point, then we fail to find a prime factor. Return to step 1.
- Otherwise, since MP is a *semi-zero point*, then we can find a prime factor p_i. When n/p_i is a prime, factoring is completed. When n/p_i is a composite, set $n = n/p_i$ and return to step 1.

The following analysis leads to the conclusion that this algorithm runs in randomly polynomial time $O((\log n)^5)$.

Note that M' is the point order of P. Since M' is the smallest positive integer satisfying $M'P = \mathcal{O}$, MP is not a zero point for $M < M'$. Note that either $\{S \not| \Phi_{p_i}(P)\}$ or, $\{S|\Phi_{p_i}(P)$ and $S^2 \not| \Phi_{p_i}(P)\}$ satisfies for each i since $M' = \mathbf{LCM}(\Phi_{p_1}(P), \Phi_{p_2}(P), \ldots, \Phi_{p_k}(P))$. If $S|\Phi_{p_i}(P)$ and $S^2 \not| \Phi_{p_i}(P)$ for all i, then $\Phi_{p_i}(P) \not| M$ for all i since M is not a multiple of S and $\Phi_{p_i}(P)$ is a multiple of S. Hence, each MP_i is not a zero point $\mathcal{O}_{p_i}$. Hence in this case, MP is an ordinary point. Otherwise, i.e., if there exists i that satisfies $S \not| \Phi_{p_i}(P))$, MP is a semi-zero point.

The probability that MP is a semi-zero point is bigger than $Q(k,S) = (1-\frac{1}{S})^{k+1}\frac{k}{S}$ in Eq. (4.1), since that $S|M'$ and $S^2 \not| M'$ implies that $S|\#E_n$ and $S^2 \not| \#E_n$ and $S|\Phi_{p_i}(P)$.

Hence, similar analysis as in Section 4.3 leads us to the conclusion that this algorithm runs in randomly polynomial time $\mathrm{O}((\log n)^5)$. ■

Next, we will prove that if both FCT(n) and ECDLP mod p are tractable, ECDLP mod n is also tractable.

Theorem 4.4 For composite n that is a product of distinct odd primes, it holds that ECDLP mod $n \leq_P$ FCT(n) $\oplus$ ECDLP mod p.

Proof. The following algorithm solves ECDLP mod n in polynomial time.

Solving ECDLP mod n algorithm using an oracle for FCT(n) and an oracle for ECDLP mod p

Input: composite $n(=\prod_{i=1}^{k} p_i)$, elliptic curve $E_n : y^2 \equiv x^3 + ax + b \pmod{n}$ and two points G and A over E_n.

Output: Integer α such that $\alpha G = A$ over E_n.

Step 1 Using an oracle for FCT(n), find k prime factors of n as $p_1, p_2, \ldots, p_k$.

Step 2 Using an oracle for ECDLP mod p_i, solve $\alpha_i G = A$ over E_{p_i} for each i.

Step 3 Calculate $\#E_{p_i}(a,b)$ for each i.

Step 4 Obtain α which satisfies the system of the congruences: $\{\alpha = \alpha_i \bmod \#E_{p_i}\}_{i=1}^{k}$ by using the Chinese Remainder Theorem.

This algorithm is completed in polynomial time, $\mathrm{O}((\log n)^6)$. Structures of the direct sum of G and A are denoted by $G = \langle G_1, G_2, \ldots, G_k\rangle$ and $A = \langle A_1, A_2, \ldots, A_k\rangle$, respectively, or simply as $G = \langle G_i\rangle_{i=1}^{k}$ and $A = \langle A_i\rangle_{i=1}^{k}$. It holds that $\alpha_i G_i = A_i$ for each i from step 2. It follows that

$$\alpha G = \alpha\langle G_i\rangle_{i=1}^{k} = \langle \alpha G_i\rangle_{i=1}^{k} = \langle (\alpha \bmod \#E_{p_i})G_i\rangle_{i=1}^{k} = \langle \alpha_i G_i\rangle_{i=1}^{k} = \langle A_i\rangle_{i=1}^{k} = A.$$

Hence, α obtained in step 4 is the solution to the equation, $\alpha G = A$. ■

4.6 Conclusion

We investigated the factoring problem and some problems in elliptic curve theory over $\mathbf{Z}/n\mathbf{Z}$. We proved that if we know $\#E_n$, we can easily factor n. We also proved that if we know $\#E_n \mod d$ for all primes d less than $2\log n$, instead of $\#E_n$ itself, we can easily factor n. Finally, we also proved that if we can solve the elliptic curve discrete logarithm problem modulo n, we can easily factor n.

Chapter 5

Discrete log problem for super-anomalous elliptic curves

Super-anomalous elliptic curves over a ring $\mathbf{Z}/n\mathbf{Z}$ ($n = \prod_{i=1}^{k} {p_i}^{e_i}$) are defined by extending anomalous elliptic curves over a prime filed $\mathbf{F}_p$. They have n points over a ring $\mathbf{Z}/n\mathbf{Z}$ and p_i points over $\mathbf{F}_{p_i}$ for all p_i. We generalize Satoh-Araki-Smart algorithm [58][59] and Rück algorithm [55], which solve a discrete logarithm problem over anomalous elliptic curves. We prove that a "discrete logarithm problem over super-anomalous elliptic curves" can be solved in *deterministic* polynomial time without knowing prime factors of n.

5.1 Introduction

Elliptic curve cryptosystems (ECC) have been widely recognized since 1985 [26][41]. The security of the ECC is based on the difficulty of solving a discrete logarithm problem over elliptic curves. The ECC is believed to have higher security than the RSA scheme [57], except for special elliptic curves. A few attacks have been proposed for such special curves including anomalous elliptic curves defined over a prime field $\mathbf{F}_p$.

Recently, Rück [55], Semaev [61], Smart [65], and Satoh-Araki [58][59] proposed efficient algorithms for solving a discrete logarithm problem over anomalous elliptic curves in polynomial time. These four algorithms are classified into two approaches: (1) number theoretic approach by Satoh-Araki and Smart and (2) algebraic geometrical approach by Semaev and Rück [2]. We generalize Satoh-Araki-Smart algorithm and Rück algorithm by applying them to elliptic curves defined over the ring $\mathbf{Z}/n\mathbf{Z}$, respectively[1]. First, we define "super-anomalous elliptic curves" by extending anomalous elliptic curves. Second, we prove that the two generalized algorithms solve the discrete logarithm problem over super-anomalous elliptic curves in deterministic polynomial time without knowing prime factors. In appendix B, we show another algorithm for solving this problem.

[1] We do not generalize Semaev algorithm since the Semaev algorithm was extended to the Rück algorithm.

5.2 Super-anomalous elliptic curves (SAEC)

First, we explain the mathematical notations used in this chapter. Second, we define *super-anomalous* elliptic curves. An (ordinary) anomalous elliptic curve is a special case in these curves.

5.2.1 Definitions

Let $\tilde{E}(\mathbf{F}_p)$ be an elliptic curve. If $\#\tilde{E}(\mathbf{F}_p)$, the number of points over $\tilde{E}(\mathbf{F}_p)$, is equal to p, this curve is called *anomalous*. In particular, we define $\tilde{E}^*(\mathbf{F}_p) = \tilde{E}(\mathbf{F}_p) - \{\mathcal{O}_p\}$. The lifting curve E is defined as follows. The lifting $E : y^2 = x^3 + ax + b$ is an elliptic curve satisfying $a \equiv a_p \pmod{p}$ and $b \equiv b_p \pmod{p}$ and $a_p, b_p \in \mathbf{F}_p$ and $a, b \in \mathbf{Z}$ (or $\mathbf{Z}/p^2\mathbf{Z}$). Next, we explain the lifting of a point over an elliptic curve. A point $(x_p, y_p) \in \tilde{E}^*(\mathbf{F}_p)$ is lifted to a point $(x, y) \in E(\mathbf{Z}_p)$ (or $E(\mathbf{Z}/p^2\mathbf{Z})$) satisfying $x \equiv x_p \pmod{p}$ and $y \equiv y_p \pmod{p}$. The inverse of the lifting is called a reduction. In this chapter, we use $\tilde{E}$ for an original elliptic curve and E for a lifted elliptic curve. The elliptic curve over the ring $\mathbf{Z}/n\mathbf{Z}$ is written at Section 2.1.3.

5.2.2 Super-anomalous elliptic curves (SAEC)

We define "super-anomalous elliptic curves" as follows.

Definition 5.1 Let $E : y^2 \equiv x^3 + ax + b \pmod{n}$ be an elliptic curve over $\mathbf{Z}/n\mathbf{Z}$ and n be the composite with k prime factors $n = \prod_{i=1}^{k} {p_i}^{e_i}$, where $p_i \neq 2, 3$. In addition, set $a_i = a \bmod p_i$ and $b_i = b \bmod p_i$ and let E_i be the elliptic curve $E_i : y^2 = x^3 + a_i x + b_i$ over $\mathbf{F}_{p_i}$ for all i. Then, the elliptic curve E is called *a super-anomalous elliptic curve*, when it satisfies that the number of points on E_i defined over $\mathbf{F}_{p_i}$: $\#E_i(\mathbf{F}_{p_i}) = p_i$ for all i.

Remark 5.1 If $k = 1$ and $e_1 = 1$, this curve is an (ordinary) anomalous elliptic curve.

Remark 5.2 It holds that $\#E(\mathbf{Z}/{p_i}^{e_i}\mathbf{Z}) = {p_i}^{e_i - 1} \cdot \#E(\mathbf{F}_{p_i})$. Hence, $\#E(\mathbf{Z}/{p_i}^{e_i}\mathbf{Z}) = {p_i}^{e_i}$ if $E(\mathbf{F}_{p_i})$ is anomalous. Since $E(\mathbf{Z}/n\mathbf{Z}) = \oplus_{i=1}^{k} E(\mathbf{Z}/{p_i}^{e_i}\mathbf{Z})$, it holds that $\#E(\mathbf{Z}/n\mathbf{Z}) = n$ for super-anomalous elliptic curve E. Note that $\#E(\mathbf{Z}/{p_i}^{e_i}\mathbf{Z}) \neq \#E(\boldsymbol{F}_{{p_i}^{e_i}})$.

Remark 5.3 Consider the curve $E(\mathbf{Z}/n\mathbf{Z})$ satisfying $\#E(\mathbf{Z}/n\mathbf{Z}) = n$, and $n = \prod_{i=1}^{k} {p_i}^{e_i}$ and $\#E(\mathbf{F}_{p_i}) \neq p_i$ for some i. A typical example of such curve satisfies that $n = p_1 p_2$, $\#E(\mathbf{F}_{p_1}) = p_2$ and $\#E(\mathbf{F}_{p_2}) = p_1$. Although $\#E(\mathbf{Z}/n\mathbf{Z}) = n$, this curve is not super-anomalous. We call this curve as *pseudo super-anomalous elliptic curve.*

Example5.1 We show an example of a super-anomalous elliptic curve. Let $E(\mathbf{Z}/143\mathbf{Z})$ be $y^2 \equiv x^3 + 42x + 58 \pmod{143}$. By factoring 143, we get $p_1 = 11$ and $p_2 = 13$. Therefore, we set $a_1 = 42 \bmod 11 = 9$ and $b_1 = 58 \bmod 11 = 3$ and $a_2 = 3$ and $b_2 = 6$. Let $E_1(\mathbf{F}_{11}) : y^2 \equiv x^3 + 9x + 3 \pmod{11}$ and $E_2(\mathbf{F}_{13}) : y^2 \equiv x^3 + 3x + 6 \pmod{13}$. Since $\#E_1(\mathbf{F}_{11}) = 11$ and $\#E_2(\mathbf{F}_{13}) = 13$, $E(\mathbf{Z}/143\mathbf{Z})$ is a super-anomalous elliptic curve.

5.3 Generalized Satoh-Araki-Smart algorithm

A discrete logarithm problem (DLP) over super-anomalous elliptic curves is to obtain a positive integer α such that $P = \alpha G$, where P and G are distinct two points over a super-anomalous elliptic curve $\tilde{E}(\mathbf{Z}/n\mathbf{Z})$. When n can be factored, this problem is solvable by using Satoh-Araki algorithm (or Rück algorithm, etc.) and Chinese remainder theorem. However, our proposed algorithms do not need a factorization of n. This situation is the similar to the relationship between the computation of Legendre symbol and Jacobi symbol.

5.3.1 Discrete Log Algorithm using the map λ_n

We generalize Satoh-Araki [58][59] and Smart [65] algorithms. This generalized algorithm can solve the DLP over super-anomalous elliptic curves $\tilde{E}(\mathbf{Z}/n\mathbf{Z})$. Note that the solution always exists for all P and G, if $G \in \tilde{E}^*(\mathbf{Z}/n\mathbf{Z})$. This algorithm includes Satoh-Araki-Smart algorithm as a special case when n is a prime.

Generalized Satoh-Araki-Smart algorithm (solving DLP over super-anomalous elliptic curve)

Input super-anomalous elliptic curve $\tilde{E}(\mathbf{Z}/n\mathbf{Z}) : y^2 \equiv x^3 + a_n x + b_n \pmod{n}$ and two distinct points $P = (s_1, t_1)$ and $G = (s_2, t_2) \in \tilde{E}^*(\mathbf{Z}/n\mathbf{Z})$

Output the positive integer α such that $P = \alpha G$

Step 0 Set $j = 0$.

Step 1 Set the lifting curve $E^{(j)} : y^2 \equiv x^3 + (a_n + jn)x + b_n \pmod{n^2}$.

Step 2 Set the lifting points $A_P = (s_1,\ t_1 + w_1 n)$ and $A_G = (s_2,\ t_2 + w_2 n) \in E^{(j)}(\mathbf{Z}/n^2\mathbf{Z})$, where $w_i = (s_i^3 + (a_n + jn)s_i + b_n - t_i^2)/(2t_i n) \bmod n$ $(i = 1, 2)$.

Step 3 Compute $\lambda_n(P)$ and $\lambda_n(G)$ by using algorithm for computing λ_n (described later).

Step 4 Output $\alpha = \lambda_n(P)/\lambda_n(G)$ if $\gcd(\lambda_n(G), n) = 1$. Otherwise, return to step 1 by setting $j \leftarrow j + 1$ to use the different lifting curve.

Remark 5.4 In step 2, w_1 (and w_2) can be always defined because it suffices that $\gcd(2t_1, n) = 1$ and $n | (s_1^3 + a_n s_1 + b_n - t_1^2)$. Supposing that $\gcd(2t_1, n) = p_i$, it holds that $s_1^3 + a_n s_1 + b_n \equiv 0 \pmod{p_i}$ and $2 | \#\tilde{E}(\mathbf{F}_{p_i})$. This statement contradicts that $\#\tilde{E}(\mathbf{F}_{p_i})$ is prime.

Remark 5.5 The above algorithm terminates by at most $(k+1)$ iterations as described in the next subsection, where k is the number of distinct prime factors of n.

Next, we describe the algorithm for computing λ_n.

Algorithm for computing λ_n

Input the lifting curve $E(\mathbf{Z}/n^2\mathbf{Z}) : y^2 \equiv x^3 + ax + b \pmod{n^2}$ and the lifting point $A := (X_1, Y_1) \in E(\mathbf{Z}/n^2\mathbf{Z})$ of R (P or G)

Output $\lambda_n(R)$

Step 1 Compute $(X_{n-1}, Y_{n-1}) \equiv (n-1)A \in E(\mathbf{Z}/n^2\mathbf{Z})$ by using an elliptic curve addition formula over $E(\mathbf{Z}/n^2\mathbf{Z})$.

Step 2 Compute

$$\lambda_n(R) = -\left(\frac{X_{n-1} - X_1}{n} \bmod n\right) / (Y_{n-1} - Y_1 \bmod n).$$

Remark 5.6 In computing λ_n, we do not need to know prime factors of n.

Remark 5.7 In the process of computing $(n-1)A$ in step 1, the sum of two points may happen to become undefined, i.e., the denominator is not coprime to n. In this case, the above algorithm cannot output the correct answer. However, this situation causes no problem for finding discrete logarithm by the following two reasons. The first reason is that when the sum becomes undefined, one can find prime factors p_i and can deterministically solve DLP over super-anomalous elliptic curves by solving DLP over anomalous elliptic curves over $\mathbf{F}_{p_i}$. The second reason is that this probability is negligibly small for large n.

Remark 5.8 The above algorithm outputs the same answer even if one use the different lifting points as the input.

Example 5.2 Setting $\widetilde{E} : y^2 \equiv x^3 + 42x + 58 \pmod{143}$, which is super-anomalous as described in Section 5.2, and $G = (107, 108) \in \widetilde{E}^*(\mathbf{Z}/143\mathbf{Z})$ and $P = (10, 68) \in \widetilde{E}^*(\mathbf{Z}/143\mathbf{Z})$, the problem is to find the smallest positive integer α such that $P = \alpha G$.

In step 1 of the generalized algorithm, we set $E : y^2 \equiv x^3 + 42x + 58 \pmod{143^2}$ as the lifting of $\widetilde{E}$. In step 2, we set $A_G = (107 + 0 \times 143,\ 108 + 101 \times 143) \in E(\mathbf{Z}/143^2\mathbf{Z})$ as the lifting of G and $A_P = (10 + 0 \times 143,\ 68 + 44 \times 143) \in E(\mathbf{Z}/143^2\mathbf{Z})$ as the lifting of P. In step 3, we obtain $\lambda_n(G) = 76$ and $\lambda_n(P) = 123$ by using algorithm for computing λ_n as follows.

> In step 1 of algorithm for computing λ_n, we compute $142A_G = (107 + 114 \times 143,\ 35 + 88 \times 143)$. In step 2, we compute $\lambda_n(G) = -\left(\frac{107+114\times143-107}{143}\right)/(35 - 108) \bmod 143 = 76$. Similarly, we can obtain $\lambda_n(P)$.

In step 4 of the generalized algorithm, since $\gcd(76, 143) = 1$, we compute $\alpha = 123/76 \bmod 143 = 75$. One can easily verify that $75G = (10, 68) = P$.

Example 5.3 Set $n = 11^2 \cdot 13 = 1573$, $\widetilde{E} : y^2 \equiv x^3 + 614x + 487 \pmod{1573}$, which is super-anomalous, and $G = (393, 108)$ and $P = (582, 1355) \in \widetilde{E}^*(\mathbf{Z}/1573\mathbf{Z})$. In step 3, we obtain $\lambda_n(G) = 656$ and $\lambda_n(P) = 1438$. In step 4, we obtain $\alpha = 1438/656 \bmod 1573 = 218$. One can easily verify that $218G = (582, 1355) = P$.

We could solve this DLP without knowing prime factors of 143 or 1573 as the above examples showed.

5.3.2 Properties of λ_n

In this subsection, we prove that the generalized Satoh-Araki-Smart algorithm solves DLP over super-anomalous elliptic curves in *deterministic* polynomial time without knowing primes factors. Concretely speaking, Theorem 5.1 describes that the map λ_n (defined in Section 5.3.1) is a homomorphism and Theorem 5.3 shows that λ_n becomes always an isomorphism by adequately choosing the lifting curves. If λ_n is an isomorphism and $P = \alpha G$, then $\lambda_n(P) = \lambda_n(\alpha G) = \alpha\lambda_n(G)$. Hence, we can solve DLP over super-anomalous elliptic curves and obtain $\alpha = \lambda_n(P)/\lambda_n(G)$.

The map λ_n has the following property.

Theorem 5.1 Let $n(= \prod_{i=1}^{k} p_i^{e_i})$ be a product of odd primes, where all of $p_i(\neq 2, 3)$ are distinct odd primes and all of e_i are positive integers. In addition, let $\widetilde{E}$ be a super-anomalous elliptic curve over $\mathbf{Z}/n\mathbf{Z}$ and E be its lifting to $\mathbf{Z}$. In this case, the map $\lambda_n : \widetilde{E}(\mathbf{Z}/n\mathbf{Z}) \to \mathbf{Z}/n\mathbf{Z}$ is a homomorphism. Therefore, letting R_1 and $R_2 \in \widetilde{E}(\mathbf{Z}/n\mathbf{Z})$, we have

$$\lambda_n(R_1 + R_2) \equiv \lambda_n(R_1) + \lambda_n(R_2) \pmod{n}.$$

Remark 5.9 When n is a prime, Theorem 5.1 was proven in [58][59].

In order to prove Theorem 5.1, we use the following three lemmas.

Lemma 5.1 Let $\widetilde{E}$ be a super-anomalous elliptic curve over $\mathbf{Z}/p^e\mathbf{Z}$ and E be its lifting to $\mathbf{Z}$. Letting R_1 and $R_2 \in \widetilde{E}^*(\mathbf{Z}/p^e\mathbf{Z})$, we have

$$\lambda_{p^e}(R_1 + R_2) \equiv \lambda_{p^e}(R_1) + \lambda_{p^e}(R_2) \pmod{p^e}.$$

For $R \in \widetilde{E}(\mathbf{Z}/p^e\mathbf{Z})$, define

$$\theta_{(p^e,j)}(R) = -\left(\frac{X_{jp^e-1} - X_1}{p^e}\right)/(Y_{jp^e-1} - Y_1) \bmod p^e,$$

where $(X_{jp^e-1}, Y_{jp^e-1}) := (jp^e - 1)A \in E(\mathbf{Z}/p^{2e}\mathbf{Z})$, $A := (X_1, Y_1) \in E(\mathbf{Z}/p^{2e}\mathbf{Z})$ is the lifting points of R, and j is a positive integer. In particular, it clearly holds that $\theta_{(p^e,1)}(R) = \lambda_{p^e}(R)$. Lemmas 5.2 and 5.3 combine two maps: $\theta_{(p^e,j)}$ and λ_n.

Lemma 5.2 Let $\widetilde{E}$ be a super-anomalous elliptic curve over $\mathbf{Z}/p^e\mathbf{Z}$ and E be its lifting to $\mathbf{Z}$. Letting $R \in \widetilde{E}^*(\mathbf{Z}/p^e\mathbf{Z})$, we have

$$\theta_{(p^e,j)}(R) \equiv j \cdot \lambda_{p^e}(R) \pmod{p^e}.$$

Lemma 5.3 Let $n = \prod_{i=1}^{k} {p_i}^{e_i}$, where all of p_i are distinct primes. In addition, let $\widetilde{E_i}$ be an anomalous elliptic curve over $\mathbf{Z}/{p_i}^{e_i}\mathbf{Z}$ for all i, and let $\widetilde{E}(\mathbf{Z}/n\mathbf{Z})$ be a super-anomalous curve made from $\widetilde{E_i}(\mathbf{Z}/{p_i}^{e_i}\mathbf{Z})$ for all i. Let E be the lifting of $\widetilde{E}$. For all $R \in \widetilde{E}^*(\mathbf{Z}/n\mathbf{Z})$ and i, we have

$$\lambda_n(R) \equiv \theta_{({p_i}^{e_i}, n/{p_i}^{e_i})}(R) \pmod{{p_i}^{e_i}}.$$

Note that E becomes the lifting curve of $\widetilde{E_i}$ for all i. Hence, all $\lambda_n(R)$ and $\theta_{({p_i}^{e_i}, n/{p_i}^{e_i})}(R)$ can be calculated by using E as the lifting curve.

Proofs of Lemmas 5.1, 5.2 and 5.3 are described in Appendix A.

We show that Theorem 5.1 is induced from Lemmas 5.1, 5.2 and 5.3.

Proof of Theorem 5.1.

From Lemmas 5.2 and 5.3, it suffices that

$$\begin{aligned} \lambda_n(R) &\equiv \theta_{({p_i}^{e_i}, n/{p_i}^{e_i})}(R) \\ &\equiv \frac{n}{{p_i}^{e_i}} \cdot \lambda_{{p_i}^{e_i}}(R) \pmod{{p_i}^{e_i}}. \end{aligned} \tag{5.1}$$

From Eq. (5.1) and the Chinese remainder theorem, there exist constants $D_i \in \mathbf{Z}/n\mathbf{Z}$ for $1 \leq i \leq k$ such that

$$\lambda_n(R) \equiv \sum_{i=1}^{k} D_i \cdot \lambda_{{p_i}^{e_i}}(R) \pmod{n}. \tag{5.2}$$

Hence, from Eq. (5.2) and Lemma 5.1, for R_1 and $R_2 \in \widetilde{E}^*(\mathbf{Z}/n\mathbf{Z})$, we have

$$\begin{aligned} \lambda_n(R_1 + R_2) &\equiv \sum_{i=1}^{k} D_i \cdot \lambda_{{p_i}^{e_i}}(R_1 + R_2) \equiv \sum_{i=1}^{k} D_i \cdot (\lambda_{{p_i}^{e_i}}(R_1) + \lambda_{{p_i}^{e_i}}(R_2)) \\ &\equiv \sum_{i=1}^{k} (D_i \cdot \lambda_{{p_i}^{e_i}}(R_1)) + \sum_{i=1}^{k} (D_i \cdot \lambda_{{p_i}^{e_i}}(R_2)) \equiv \lambda_n(R_1) + \lambda_n(R_2) \pmod{n}. \end{aligned}$$

Therefore, Theorem 5.1 has been proven. ∎

Theorem 5.1 states only that λ_n is a homomorphism. The following explains the image of the map λ_n.

Theorem 5.2 Under the same condition as Theorem 5.1, the image of the map λ_n becomes one of the following three sets: (i) $(\mathbf{Z}/n\mathbf{Z})^*$, (ii) $n'(\mathbf{Z}/n\mathbf{Z})^*$, where n' is a non-trivial factor of n, or (iii) $\{0\}$. The result depends on the choice of lifting curve E.

Note 5.1 In the case of (i), λ_n is surjective and becomes an isomorphism.

Note 5.2 In the case of (ii), we can obtain non-trivial factors n' and n/n' of n by computing $\gcd(\lambda_n(P), n) = n'$.

When the image of λ_n falls under (ii) or (iii), we cannot solve DLP directly (of course, we can solve DLP by knowledge on prime factors of n in case of (ii)). However, the following theorem guarantees that these cases can be avoided by adequately choosing the lifting curves.

Theorem 5.3 Under the same condition as Theorems 5.1 and 5.2, define an elliptic curve $E^{(j)}$ by $E^{(j)} : y^2 \equiv x^3 + (a + nj)x + b \pmod{n^2}$. At least one of the following $(k+1)$ maps: λ_n for $E(= E^{(0)})$, $E^{(1)}, E^{(2)} \cdots$, and $E^{(k)}$ is an isomorphism to $(\mathbf{Z}/n\mathbf{Z})^*$.

Proofs of Theorems 5.2 and 5.3 are described in Appendix A.

Theorem 5.3 guarantees that even if one cannot solve DLP with one lifting E, one can solve it with at most $k+1$ liftings: $E^{(0)}(= E), E^{(1)}, E^{(2)}, \cdots$ and $E^{(k)}$, where k is the number of prime factors of n. Note that Satoh-Araki's case corresponds to $k = 1$.

Hence, it has been proven that the generalized Satoh-Araki-Smart algorithm solves DLP over super-anomalous elliptic curves in deterministic polynomial time without knowing primes factors.

5.4 Generalized Rück algorithm

In this section, we consider a case of n, where n is a product of only distinct primes, that is, $n = \prod_{i=1}^{k} p_i$.

5.4.1 Discrete Log Algorithm using the map Λ_n

In this section, we generalize Rück algorithm [55]. Before describing the generalized algorithm, we introduce the following operation $\oplus$. Letting $R_1 = (x_1, y_1), R_2 = (x_2, y_2) \in \tilde{E}(\mathbf{Z}/n\mathbf{Z})$ and $r_1, r_2 \in \mathbf{Z}/n\mathbf{Z}$,

$$(R_1, r_1) \oplus (R_2, r_2) = (R_1 + R_2, r_1 + r_2 + \Theta(R_1, R_2)).$$

The first element is operated by elliptic curve addition over $\mathbf{Z}/n\mathbf{Z}$ and the second element is operated by addition over $\mathbf{Z}/n\mathbf{Z}$. $\Theta(R_1, R_2)$ is defined as follows.

$$\Theta(R_1, R_2) = \begin{cases} \dfrac{3x_1^2 + a_n}{2y_1}, & \text{if } R_1 = R_2 \\ 0, & \text{if } R_1 = -R_2 \\ \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{otherwise.} \end{cases}$$

In addition, we define $\otimes$ operation as follows. For positive integer j, $j \otimes (R_1, r_1) \equiv \underbrace{(R_1, r_1) \oplus \cdots \oplus (R_1, r_1)}_{j \text{ times}}$.

Generalized Rück algorithm (solving DLP over a super-anomalous elliptic curve)

Input and Output are the same as the generalized Satoh-Araki-Smart algorithm.

Step 1 Compute $n \otimes (P, 0)$ and $n \otimes (G, 0)$ by using the above operation.

Step 2 Set $\Lambda_n(P)$ as the second element of $n \otimes (P, 0)$ and $\Lambda_n(G)$ as the second element of $n \otimes (G, 0)$.

Step 3 Output $\alpha = \Lambda_n(P)/\Lambda_n(G) \bmod n$.

Remark 5.10 This generalized algorithm includes Rück algorithm as a special case when n is a prime.

Remark 5.11 The map Λ_n is an isomorphism as described at the next subsection. Hence, the above algorithm always terminates and outputs the correct answer. The total calculation time is $O((\log n)^3)$.

Remark 5.12 We do not need to know prime factors of n in executing the generalized Rück algorithm.

Remark 5.13 In the process of computing $n \otimes (P, 0)$ in step 1, the sum of two points may happen to become undefined. In this case, the above algorithm cannot output the correct answer. However, this situation causes no problem for finding discrete logarithm by the same reason described in Remark 5.7.

Remark 5.14 The above algorithm can output the correct answer if $n = \prod_{i=1}^{k} p_i$, where p_i are distinct primes.

Example 5.4 We use the same example as Example 5.2. In step 1 of the generalized algorithm, we compute $143 \otimes (P, 0) = (\mathcal{O}, 86)$ and $143 \otimes (G, 0) = (\mathcal{O}, 45)$. In step 2, we set $\Lambda_n(P) = 86$ and $\Lambda_n(G) = 45$. In step 3, we compute $\alpha = \Lambda_n(P)/\Lambda_n(G) = 86/45 \bmod 143 = 75$. One can easily verify that $75G = (10, 68) = P$ and this is the same answer as Example 5.2.

Remark 5.15 We found that Rück algorithm is from 2.2 to 2.3 times as fast as Satoh-Araki algorithm by numerical experiments. For example, the former solves in 0.15 seconds and the latter solves in 0.32 seconds for 160 bits.

5.4.2 Properties of Λ_n

In this subsection, we prove that the map Λ_n (defined in Section 5.4.1) is an isomorphism. If Λ_n is an isomorphism, we can also solve DLP over a super-anomalous elliptic curve by the generalized Rück algorithm.

Theorem 5.4 Let $n(= \prod_{i=1}^{k} p_i)$ be a product of odd primes, where all of p_i are distinct odd primes. In addition, let $\widetilde{E}$ be a super-anomalous elliptic curve over $\mathbf{Z}/n\mathbf{Z}$. In this case, the map $\Lambda_n : \widetilde{E}(\mathbf{Z}/n\mathbf{Z}) \to \mathbf{Z}/n\mathbf{Z}$ is an isomorphism.

Proof of Theorem 5.4 is described in Appendix A.

Remark 5.16 When n is a prime, Theorem 5.4 has been proven in [55].

Hence, it has been proven that the generalized Rück algorithm solves DLP over super-anomalous elliptic curves in deterministic polynomial time without knowing primes factors.

5.5 Application to Okamoto-Uchiyama's scheme

At Eurocrypt'98, Okamoto and Uchiyama [48] proposed an ID-based cryptosystem (OU scheme), which is based on the difficulty of the "anomalous E_n–Diffie–Hellman problem." In other words, this problem is the Diffie-Hellman problem [15] over super-anomalous elliptic curves over $\mathbf{Z}/n\mathbf{Z}$. Hereafter, we call this problem "DHP over super-anomalous elliptic curves." Although they proposed a new scheme, they found by themselves that it is not secure. In [48], the new scheme and breaking method are described.

This problem is formulated as follows.

DHP over super-anomalous elliptic curves: given a composite n (whose prime factors p and q are secret) and a super-anomalous elliptic curve $E(\mathbf{Z}/n\mathbf{Z})$, and a point $G \in E(\mathbf{Z}/n\mathbf{Z})$, and two points $\alpha G \in E(\mathbf{Z}/n\mathbf{Z})$ and $\beta G \in E(\mathbf{Z}/n\mathbf{Z})$, find $\alpha\beta G \in E(\mathbf{Z}/n\mathbf{Z})$.

If DLP over super-anomalous elliptic curves is solvable, DHP over super-anomalous elliptic curves is also solvable. The reason is as follows. Suppose that the former problem is solvable. Then, given G, and αG, and βG, one can find α from G and αG by solving the problem and computing $\alpha(\beta G) = \alpha\beta G \in E(\mathbf{Z}/n\mathbf{Z})$. Hence, in order to prove that DHP over super-anomalous elliptic curves is solvable, it is sufficient to prove that DLP over super-anomalous elliptic curves is solvable.

Okamoto and Uchiyama [48] proved that given a super-anomalous elliptic curve, a prime factors of n is revealed with probability 1/2 by using a technique similar to that in our another paper [30]. After obtaining prime factors p_1 and p_2, one can solve DLP over anomalous elliptic curves $E(\mathbf{F}_{p_1})$ and $E(\mathbf{F}_{p_2})$ by Satoh-Araki algorithm (or Rück algorithm, etc.) and obtain the solution of DLP over super-anomalous elliptic curves by using the Chinese Remainder Theorem. Therefore, they proved that DHP over super-anomalous elliptic curves is solvable and that the OU scheme is broken in *probabilistic* polynomial time.

In contrast to their attack, we point out that our algorithms described in Sections 5.3 and 5.4 are applicable to the direct attack to their cryptosystem. A remarkable point is that our algorithms *deterministically* solve DLP over super-anomalous elliptic curves and do not need to know prime factors of n. This implies that DHP over super-anomalous elliptic curves is solvable and that the OU scheme is broken in *deterministic* polynomial time. Furthermore, this means that all schemes based on the difficulty of DLP over super-anomalous elliptic curves are solvable in polynomial time.

5.6 Related works

We have shown that our algorithms can solve DLP over super-anomalous elliptic curves without knowing prime factors. Similarly, some number theoretic or elliptic curve theoretic algorithms execute in polynomial time without knowing prime factors. However, some algorithms are proven not to be executable in polynomial time without knowing prime factors. We show one example of the former algorithms (Legendre symbol and Jacobi symbol) and one example of the latter algorithms (Schoof algorithm [60]).

Example without need of factoring: Legendre symbol and Jacobi symbol

The Legendre symbol was extended to the Jacobi symbol. Letting n be a composite with prime factors: $n = p_1 p_2$, the Jacobi symbol (a/n) is defined to be $(a/p_1)(a/p_2)$. The Legendre symbol (a/p_1) and (a/p_2) are computable in polynomial time. Hence, one can compute the Jacobi symbol using prime factors p_1 and p_2. However, it is well known that the Jacobi symbol (a/n) is also computable in polynomial time without knowing prime factors by the same algorithm for computing the Legendre symbol. This situation is similar to our algorithms.

Example with need of factoring: Schoof algorithm

The Schoof algorithm [60] and its improved versions can count the number of points $\#E(\mathbf{F}_p)$ in $O((\log p)^8)$ and $O((\log p)^6)$. Letting n be a composite, consider the following two questions.

Q1. Can one count the number of points $\#E(\mathbf{Z}/n\mathbf{Z})$ by using the Schoof algorithm without knowing prime factors in advance?

Q2. Can one count the number of points $\#E(\mathbf{Z}/n\mathbf{Z})$ in polynomial time without knowing prime factors in advance?

The answer for Q2 is "No" because the problem of counting the number of points $\#E(\mathbf{Z}/n\mathbf{Z})$ and the problem of factoring n are computationally equivalent. This is proved in [30]. Therefore, the answer for Q1 is also "No." Hence, the Schoof algorithm cannot count the correct number of points in polynomial time without knowing prime factors. This situation is different from our algorithms.

5.7 Conclusion

In this chapter, we proved that a discrete logarithm problem for super-anomalous elliptic curves is solvable in deterministic polynomial time without knowing prime factors. We also proposed algorithms which solve this problem by generalizing Satoh-Araki-Smart algorithm [58][59] and Rück algorithm [55]. Finally, we showed related works of our proposed algorithms.

Chapter 6

Multi-variate RSA cryptosystems and their security analyses

Many variants of RSA cryptosystems have been proposed. One of them is the Koyama scheme proposed at Eurocrypt'95. In this chapter, we generalize the Koyama scheme by using multi-variate rational functions. Then, we evaluate the speed of encryption and decryption of this generalized scheme with chaining. We also analyze its security when an attacker knows partial information about a plaintext. Finally, we show another attack on the two algebraic related plaintexts of the same user.

6.1 Introduction

Many variants of RSA cryptosystems have been proposed. One of them is the Koyama scheme [28]. This scheme is based on operations over singular cubic curves: $y^2 + axy = x^3$. It has been proven that breaking this scheme (or obtaining the whole plaintext) is computationally equivalent to breaking the original RSA scheme. Furthermore, Bleichenbacher [8] proved that if an attacker knows half of the plaintext, he can recover the other half of the plaintext in polynomial time.

In this chapter, we generalize the Koyama scheme by using multi-variate rational functions. Then, we evaluate the speed of encryption and decryption of the generalized scheme. We also analyze its security when an attacker knows partial information about the plaintext. Finally, we show another attack on the two related plaintexts of the same user.

6.2 Multi-variate RSA cryptosystem

In this section, we generalize the Koyama scheme to define multi-variate RSA cryptosystems.

6.2.1 The Koyama scheme

The Koyama scheme is described [28] as follows.

Public key: $(e, n(=pq))$, where p and q are distinct primes.
Secret key: $(d, (p, q))$, where $ed \equiv 1 (\mathrm{mod}\ \phi(n))$
Plaintext: two blocks (M_1, M_2), where $0 < M_1, M_2 < n$
Encryption:

$$m = \frac{{M_1}^3}{{M_2}^2} \bmod n, \quad C = m^e \bmod n, \quad a = \frac{{M_1}^3 - {M_2}^2}{M_1 M_2} \bmod n$$

Ciphertext: two blocks (C, a)
Decryption:

$$m = C^d \bmod n, \quad M_1 = \frac{a^2 m}{(m-1)^2} \bmod n, \quad M_2 = \frac{a^3 m}{(m-1)^3} \bmod n$$

This scheme is based on operations over singular cubic curves: $y^2 + axy = x^3$. Note that ${M_2}^2 + aM_1M_2 \equiv {M_1}^3 \pmod n$. It has been proven that breaking this scheme (or obtaining the whole plaintext (M_1, M_2)) is computationally equivalent to breaking the original RSA scheme [28]. Furthermore, Bleichenbacher [8] proved that if we know M_1 (or M_2), we can recover M_2 (or M_1) in polynomial time.

6.2.2 Mathematical Preparation

Consider the set of t-variate $2t$ rational functions, $f : \mathcal{S}_1 (\subseteq (\mathbf{Z}/n\mathbf{Z})^t) \to \mathbf{Z}/n\mathbf{Z}$, $g_i : \mathcal{S}_1 \to \mathbf{Z}/n\mathbf{Z}$, for $1 \le i \le t-1$ and $h_i : \mathcal{S}_2 (\subseteq (\mathbf{Z}/n\mathbf{Z})^t) \to \mathbf{Z}/n\mathbf{Z}$, for $1 \le i \le t$, satisfying the following three conditions, where n is the product of two odd primes and $\mathcal{S}_2$ is the range of $(f(x_1, \ldots, x_t), g_1(x_1, \ldots, x_t), g_2(\cdot), \ldots, g_{t-1}(\cdot))$ for all $(x_1, \ldots, x_t) \in \mathcal{S}_1$.

Condition 1

$$h_i(f(x_1, \ldots, x_t), g_1(\cdot) \ldots, g_{t-1}(\cdot)) = x_i \text{ for } (x_1, \ldots, x_t) \in \mathcal{S}_1,\ 1 \le i \le t.$$
$$f(h_1(x_1, \ldots, x_t), h_2(\cdot), \ldots, h_t(\cdot)) = x_1 \text{ for } (x_1, \ldots, x_t) \in \mathcal{S}_2.$$
$$g_i(h_1(x_1, \ldots, x_t), h_2(\cdot), \ldots, h_t(\cdot)) = x_{i+1} \text{ for } (x_1, \ldots, x_t) \in \mathcal{S}_2,\ 1 \le i \le t-1.$$

Condition 2 $|\mathcal{S}_1| = |\mathcal{S}_2| \approx n^t$ and the maps f, g_i and h_j are one-to-one.
Condition 3 The degree of f, g_i and h_i in x_j is $\mathrm{O}((\log n)^c)$ for all i and j, where c is a constant.

6.2.3 Encryption and decryption

We define t-variate RSA schemes as follows by using the above functions f, g_i and h_i. Suppose that functions f, g_i, h_j are public. This general scheme can be applied to both secret communication and digital signature. For simplicity, we mention the general scheme for secret communication.
Public key: $(e, n(=pq))$, where p and q are distinct primes.
Secret key: $(d, (p, q))$, where $ed \equiv 1 (\mathrm{mod}\ \phi(n))$

Plaintext: t blocks: $(M_1, M_2, \ldots, M_t) \in \mathcal{S}_1$

Encryption:

$$\begin{aligned} m &= f(M_1, M_2, \ldots, M_t) \bmod n \\ C &= m^e \bmod n \\ a_i &= g_i(M_1, M_2, \ldots, M_t) \bmod n, \text{ for } 1 \le i \le t-1 \end{aligned}$$

Ciphertext: t blocks: $(C, a_1, a_2, \ldots, a_{t-1})$

Decryption:

$$\begin{aligned} m &= C^d \bmod n \\ M_j &= h_j(m, a_1, \ldots, a_{t-1}) \bmod n, \text{ for } 1 \le j \le t \end{aligned}$$

Note 6.1 If $(M_1, M_2, \ldots, M_t) \notin \mathcal{S}_1$, then the above scheme does not work for secret communication.

This general scheme has several examples.

Example 6.1: [Linear functions] Consider $t \times t$ matrix $V = \{v_{ij}\}$. Suppose that V has the inverse matrix $W = \{w_{ij}\}$. Then, the functions satisfy the above three conditions by the following setting.

$$\begin{aligned} f(x_1, \ldots, x_t) &= \sum_{j=1}^{t} v_{1j} x_j \\ g_i(x_1, \ldots, x_t) &= \sum_{j=1}^{t} v_{(i+1),j} x_j \text{ , for } 1 \le i \le t-1 \\ h_i(x_1, \ldots, x_t) &= \sum_{j=1}^{t} w_{i,j} x_j \text{ , for } 1 \le i \le t. \end{aligned}$$

In this case, $\mathcal{S}_1 = (\mathbf{Z}/n\mathbf{Z})^t, \mathcal{S}_2 = (\mathbf{Z}/n\mathbf{Z})^t$, and $|\mathcal{S}_1| = |\mathcal{S}_2| = n^t$.

Example 6.2 [Cubic rational functions] (Koyama scheme): The Koyama scheme is a special case of the generalized scheme. Let $t = 2$.

$$f(x_1, x_2) = \frac{{x_1}^3}{{x_2}^2} \quad , \quad g_1(x_1, x_2) = \frac{{x_1}^3 - {x_2}^2}{x_1 x_2},$$
$$h_1(x_1, x_2) = \frac{{x_2}^2 x_1}{(x_1 - 1)^2} \quad , \quad h_2(x_1, x_2) = \frac{{x_2}^3 x_1}{(x_1 - 1)^3}.$$

In this case, $\mathcal{S}_1 = \{(x_1, x_2) \in (\mathbf{Z}/n\mathbf{Z})^2 : \gcd(x_1, n) = \gcd(x_2, n) = \gcd({x_1}^3 - {x_2}^2, n) = 1\}$ and $\mathcal{S}_2 = \{(x_1, x_2) \in (\mathbf{Z}/n\mathbf{Z})^2 : \gcd(x_1, n) = \gcd(x_1 - 1, n) = \gcd(x_2, n) = 1\}$. If a plaintext $(M_1, M_2) \notin \mathcal{S}_1$, then we can find the prime factors of n with high probability.

Example 6.3 (Numerical example of Example 6.2): Let $n = 323, e = 5$ and the plaintext $(M_1, M_2) = (115, 254)$. We can obtain a ciphertext $(C, a) = (106, 175)$ by computing

$$C = \left(\frac{(115)^3}{(254)^2}\right)^5 = 106, a = \frac{115^3 - 254^2}{115 \cdot 254} = 175.$$

Let the plaintext $(M_1', M_2') = (61, 115)$. A ciphertext is $(C', a') = (277, 125)$ by computing

$$C' = \left(\frac{(61)^3}{(115)^2}\right)^5 = 277, a' = \frac{61^3 - 115^2}{61 \cdot 115} = 125.$$

Example 6.4: [Construction by chaining of primitive functions] Suppose that the set of 2-variate functions F, G, H_1, H_2, which are primitive functions, satisfies the above three conditions. We can construct a t-variate RSA scheme for arbitrary t by chaining F and G (or H_1 and H_2) as follows.

$$\begin{aligned}
g_1(x_1, \ldots, x_t) &= G(x_1, x_2) \\
g_2(x_1, \ldots, x_t) &= G(F(x_1, x_2), x_3) \\
g_3(x_1, \ldots, x_t) &= G(F(F(x_1, x_2), x_3), x_4) \\
&\vdots \\
g_{t-1}(x_1, \ldots, x_t) &= G(F(F(\cdots(F(x_1, x_2), x_3), x_4), \cdots), x_t) \\
f(x_1, \ldots, x_t) &= F(F(F(\cdots(F(x_1, x_2), x_3), x_4), \cdots), x_t) \\
h_t(x_1, \ldots, x_t) &= H_2(x_1, x_t) \\
h_{t-1}(x_1, \ldots, x_t) &= H_2(H_1(x_1, x_t), x_{t-1}) \\
h_{t-2}(x_1, \ldots, x_t) &= H_2(H_1(H_1(x_1, x_t), x_{t-1}), x_{t-2}) \\
&\vdots \\
h_2(x_1, \ldots, x_t) &= H_2(H_1(H_1(\cdots(H_1(x_1, x_t), x_{t-1}), x_{t-2}), \cdots), x_2) \\
h_1(x_1, \ldots, x_t) &= H_1(H_1(H_1(\cdots(H_1(x_1, x_t), x_{t-1}), x_{t-2}), \cdots), x_2)
\end{aligned}$$

For example, suppose that $F(x_1, x_2) = x_1 + x_2, G(x_1, x_2) = x_1 + 2x_2, H_1(x_1, x_2) = 2x_1 - x_2$ and $H_2(x_1, x_2) = -x_1 + x_2$. For arbitrary t, $C = m^e = (\sum_{j=1}^{t} M_j)^e \bmod n, a_i = \sum_{j=1}^{i} M_j + 2M_{i+1} \bmod n$, for $1 \leq i \leq t-1, M_1 = 2^{t-1}m - \sum_{j=1}^{t-1} 2^{j-1}a_j \bmod n$ and $M_i = -2^{t-i}m + a_{i-1} + \sum_{j=i}^{t-1} 2^{j-i}a_j \bmod n$, for $2 \leq i \leq t$.

6.2.4 Efficiency of the general scheme

We analyze the efficiency of the general scheme in using chaining.

The encryption consists of $(t-1)$ operations of F and G and one e-power operation. The decryption consists of $(t-1)$ operations of H_1 and H_2 and one d-power operation. Suppose that the operation of F (or G) is executed by the u_1 (or u_2) multiplication and the e-power operation is executed by the $w(e)$ multiplication. Then, the total number of multiplications in the encryption becomes $w(e) + (u_1 + u_2)(t-1)$. Since this encryption treats t blocks at one time, the average number of multiplications per block becomes $\frac{w(e)}{t} + \frac{(u_1+u_2)(t-1)}{t}$. This value converges $u_1 + u_2$ as $t \to \infty$. The multi-variate RSA scheme is asymptotically $\frac{w(e)}{u_1+u_2}$ times as fast as the original RSA scheme since the latter needs $w(e)$ multiplications.

Consider the chaining of the Koyama scheme. This scheme needs four multiplications and two divisions in computing f and g_1. Supposing that one division corresponds to 10

multiplications, $u_1 + u_2 = 24$. In the case of a large e, letting e be 1024 bit lengths and $w(e)$ be $1024 \times \frac{6}{5}$, it becomes $\frac{w(e)}{u_1+u_2} = 51.2$. In other words, the multi-variate RSA scheme is about 50 times as fast as the original scheme. Note that in the case of a small e, since $w(e)$ is also small, the multi-variate RSA scheme is not efficient.

6.3 Security Analysis of whole or partial plaintext

In this section, we analyze the security of the multi-variate RSA cryptosystems.

6.3.1 Security of whole plaintext

Theorem 6.1 Given any ciphertext and the public key, obtaining the whole plaintext $(M_1, \ldots, M_t)$ in the general scheme is computationally equivalent to breaking the original RSA scheme.

Proof. We prove that the following two problems are computationally equivalent.

Problem A: Given a ciphertext $(C, \{a_i\}_{i=1}^{t-1})$ and public keys (e, n), find the whole plaintext $(\{M_i\}_{i=1}^{t})$.

Problem B: Given C, e and n, find the integer m such that $C = m^e \bmod n$.

In other words, we prove that if we can solve Problem A, we can also prove B in deterministic polynomial time and that if we can solve Problem B, we can also prove A in deterministic polynomial time.

Algorithm for Solving Problem B using the algorithm $\mathbf{ALGO_A}$ for solving A
Input C, e, n
Output the positive integer m such that $C = m^e \bmod n$
Step 1 choose $a_i \in (\mathbf{Z}/n\mathbf{Z})^*$ randomly for $1 \leq i \leq t-1$.
Step 2 run $\mathbf{ALGO_A}(C, \{a_i\}_{i=1}^{t-1}, e, n) \rightarrow (\{M_i\}_{i=1}^{t})$.
Step 3 compute $m = f(M_1, M_2, \ldots, M_t)$.

We can easily verify that the above algorithm can solve Problem B.

Algorithm for Solving Problem A using the algorithm $\mathbf{ALGO_B}$ for solving B
Input $(C, \{a_i\}_{i=1}^{t-1}), e, n$
Output $(\{M_i\}_{i=1}^{t})$ for $(C, \{a_i\}_{i=1}^{t-1})$
Step 1 run $\mathbf{ALGO_B}(C, e, n) \rightarrow m$.
Step 2 compute $M_i = h_i(m, a_1, a_2, \ldots . a_{t-1})$ for $1 \leq i \leq t$.

We can easily verify that the above algorithm can solve Problem A.

Hence, the above two problems are computationally equivalent. ■

Note 6.2 The above algorithms do not get the secret keys. If an attacker can factorize n, he can break the general scheme and the original RSA. This means that he can get secret key and recover plaintexts.

6.3.2 Security of partial plaintext

Next, we analyze the security of a certain part of the plaintext.

Theorem 6.2 If an attacker knows one block, for example M_i, in the plaintext then he can recover the whole plaintext ($\{M_j\}_{j=1}^t$) in polynomial time.

Proof. Consider simultaneous congruences

$$\begin{cases} M_i \equiv h_i(x, a_1, a_2, \ldots, a_{t-1}) \pmod{n} \\ x^e \equiv C \pmod{n}. \end{cases}$$

The above congruences have only one solution: $x = m$. Computing

$$\gcd(h_i(x, a_1, a_2, \ldots, a_{t-1}) - M_i, x^e - C),$$

we can obtain $C'(x - m) \bmod n$ and get m. By substituting $x = m$, we can recover the whole plaintext M_j by computing $h_j(m, a_1, \ldots, a_{t-1}) \bmod n$ for all j. ■

Theorem 6.2 shows that when an attacker happens to know one of $M_1, M_2, \ldots, M_t$, he can recover the whole plaintext. Then, Theorems 6.1 and 6.2 show that breaking the RSA scheme and obtaining one block of plaintext in the multi-variate RSA scheme are computationally equivalent.

6.4 Security analysis of related messages

In this section, we analyze the security when related messages are sent.

6.4.1 Security for the relationship between blocks

Suppose that an attacker knows a certain algebraic relationship between blocks instead of the value of block. The next theorem shows that he can also recover the whole plaintext in this case.

Theorem 6.3 If an attacker knows the non-trivial algebraic relationship $P(M_1, \ldots, M_t) \equiv 0 \pmod{n}$ between $M_1, \ldots, M_t$, he can recover the whole plaintext ($\{M_j\}_{j=1}^t$) in polynomial time.

Note 6.3 The relationship $P(x_1, \ldots, x_t)$ must be a rational function. For example, $M_1 + M_2 + \cdots + M_t = C'$, or $P(x_1, x_2, \ldots, x_t) = x_1 + x_2 + \cdots x_t - C'$. This corresponds that the sum of blocks are known. This value may be used in error detecting.

Proof. It satisfies that $P(M_1, \ldots, M_t) = P(h_1(m, a_1, \ldots, a_{t-1}), \ldots, h_t(m, a_1, \ldots, a_{t-1})) = 0$. Consider simultaneous congruences:

$$P(h_1(x, a_1, \cdots, a_{t-1}), \ldots, h_t(x, a_1, \ldots, a_{t-1})) = 0$$

and $x^e - C = 0$. In the similar way as the proof of Theorem 6.2, we can recover the whole plaintext by solving the above congruences. ■

6.4.2 Security for the relationship between two plaintexts

Suppose that the partially same plaintext are sent to the same receiver.

Theorem 6.4 Suppose that the public key e is $O((\log n)^c)$, where c is a constant. Consider that a user sends $(M_1, \ldots, M_t)$ the first time and $(M_1', \ldots, M_t')$ the second time. If an attacker knows the fact that $M_i = M_j'$, the whole plaintexts $(\{M_i\}_{i=1}^t)$ and $(\{M_i'\}_{i=1}^t)$ are recovered in deterministic polynomial time.

Proof. Suppose that a user sends a ciphertext $(C, \{a_i\}_{i=1}^{t-1})$ for a plaintext $(\{M_i\}_{i=1}^t)$ and a ciphertext $(C', \{a_i'\}_{i=1}^{t-1})$ for a plaintext $(\{M_i'\}_{i=1}^t)$. Then,

$$
\begin{aligned}
m &= f(M_1, \ldots, M_t) \bmod n \\
C &= m^e \bmod n \\
a_i &= g_i(M_1, \ldots, M_t) \bmod n, \text{ for } 1 \le i \le t-1 \\
m' &= f(M_1', \ldots, M_t') \bmod n \\
C' &= m'^e \bmod n \\
a_i' &= g_i(M_1', \ldots, M_t') \bmod n, \text{ for } 1 \le i \le t-1.
\end{aligned}
$$

Suppose that an attacker knows $C, C', \{a_i\}_{i=1}^{t-1}$ and $\{a_i'\}_{i=1}^{t-1}$ and the fact that $M_i = M_j'$. It satisfies that $M_i = h_i(m, a_1, \cdots, a_{t-1})$ and $M_j' = h_j(m', a_1', \cdots, a_{t-1}')$. Letting

$$
R(x, y) = h_i(x, a_1, \cdots, a_{t-1}) - h_j(y, a_1', \cdots, a_{t-1}'),
$$

it satisfies that $R(m, m') \equiv 0 \pmod n$. Consider the following three 2-variate simultaneous congruences:

$$
\begin{cases}
R(x, y) & \equiv 0 \pmod n \\
x^e - C & \equiv 0 \pmod n \\
y^e - C' & \equiv 0 \pmod n.
\end{cases}
$$

The above congruences have only one solution: $(x, y) = (m, m')$. When e is small, the solution can be easily obtained [13]. In the Appendix C, we shall describe how to solve. After obtaining m and m', we can recover the whole plaintext $(\{M_i\}_{i=1}^t)$ and $(\{M_i'\}_{i=1}^t)$ by computing $M_i = h_i(m, a_1, \cdots a_{t-1})$ and $M_i' = h_i(m', a_1', \cdots a_{t-1}')$ for $1 \le i \le t$. ∎

Example 6.5 We use the values in Example 6.3. Since $M_1 = h_1(m, a) = h_2(m', a') = M_2'$, it satisfies that

$$
\frac{a^2 m}{(m-1)^2} = \frac{a'^3 m'}{(m'-1)^3}.
$$

By substituting $a = 175$ and $a' = 125$ and replacing m with x and m' with x', we can obtain

$$
267yx^2 + (60y^3 + 143y^2 + 292y + 263)x + 267y \equiv 0 \pmod{323} \tag{6.1}
$$

Consider the three congruences: Eq. (6.1), $x^5 - 106 \equiv 0 \pmod{323}$ and $y^5 - 277 \equiv 0 \pmod{323}$. By solving these congruences, we obtain $m = 140$ and $m' = 292$. How to solve them is described in Appendix C. We can obtain $M_1 = h_1(140, 175) = 115$, $M_2 = 254$, $M_1' = 61$ and $M_2' = h_2(292, 125) = 115$. We can easily verify that $M_1 = M_2'$.

Theorem 6.4 can be generalized as follows.

Theorem 6.5 Consider that the non-trivial algebraic relationship:

$$Q(M_1, \ldots, M_t, M_1', \ldots, M_t') \equiv 0 \pmod{n}$$

are given instead of $M_i = M_j'$. Suppose that Q is a $2t$-variate rational function and Q includes both at least one of $\{M_i\}_{i=1}^t$ and $\{M_i'\}_{i=1}^t$. Then, we can recover the both of whole plaintexts $(\{M_i\}_{i=1}^t)$ and $(\{M_i'\}_{i=1}^t)$.

Note 6.4 For example, $Q(x_1, \ldots, x_t, x_1', \ldots, x_t') = x_1 - x_1'$. This corresponds that the beginnings of plaintext are the same.

Proof. It satisfies that

$$\begin{aligned} &Q(h_1(m, a_1, \ldots, a_{t-1}), \ldots, h_t(m, a_1, \ldots, a_{t-1}), \\ &h_1(m', a_1', \ldots, a_{t-1}'), \ldots, h_t(m', a_1', \cdots, a_{t-1}')) \equiv 0 \pmod{n}. \end{aligned} \tag{6.2}$$

By replacing m with x and m' with x', we can obtain $Q(x, y) \equiv 0 \pmod{n}$. By solving simultaneous congruences: $Q(x, y) \equiv 0 \pmod{n}$, $x^e - C \equiv 0 \pmod{n}$ and $y^e - C' \equiv 0 \pmod{n}$, we can obtain m and m'. By substituting m and m', we can recover the whole plaintext $\{M_i\}_{i=1}^t$ and $\{M_i'\}_{i=1}^t$. ■

Note 6.5 If only $\{M_i\}_{i=1}^t$ appear in Q, we can recover all of $\{M_i\}_{i=1}^t$ from Theorem 6.3.

6.5 Conclusion

We have generalized the Koyama scheme by using multi-variate rational functions and evaluated the speed of its encryption and decryption. We also analyzed the security of generalized scheme when an attacker knows partial information about the plaintext. Finally, we showed another attack on two related plaintexts of the same user.

Chapter 7

How to generate short addition chains

Power exponentiation is an important operation in modern cryptography. This operation can be efficiently calculated using the concept of the addition chain. In this chapter, two new systematic methods, a Run-length method and a Hybrid method, are proposed to generate a short addition chain. The performance of these two methods are theoretically analyzed and it is shown that the Hybrid method is more efficient and practical than known methods. The proposed methods can reduce the addition chain length by 8%, in the best case, compared to the Window method.

7.1 Introduction

The encryption and decryption in the RSA scheme [57] consist of the power exponentiation: $M^e \bmod n$. This operation is very important and is also used in prime testing algorithms, integer factoring algorithms, and so on. Fast calculation of this operation is realized by reducing the number of multiplications as much as possible and/or speeding up the operation of multiplication. In this chapter, we consider the former problem, i.e., how to reduce the number of multiplications.

It is well known that the power exponentiation can be efficiently calculated using the addition chain [25]. Suppose that M^e can be calculated as $M^1 \rightarrow M^{a_1} \rightarrow M^{a_2} \rightarrow \cdots \rightarrow M^{a_r}(= M^e)$ based on a rule:

$$M^{a_i} = M^{a_j} \times M^{a_k}, \quad j, k < i. \tag{7.1}$$

The sequence of exponents a_i, $\langle a_0(= 1), a_1, a_2, \cdots, a_r(= e)\rangle$, is called an addition chain since this sequence satisfies the relation

$$a_i = a_j + a_k, \quad j, k < i. \tag{7.2}$$

Note that the chain length r is equal to the number of multiplications used to calculate M^e.

Finding a short addition chain for a given e leads to the fast calculation of the power exponentiation and, hence, fast encryption and decryption in the RSA scheme. However, obtaining the shortest chain is an NP-hard problem [16]. Therefore, many algorithms have been proposed to obtain a sub-optimal chain, for example, the Bos-Coster method [9], the Yacobi method [67], the m-ary or 2^κ-ary method [25], the Lou-Chang method [35], the Window method [25], and the Extended Window method with the Tunstall-like algorithm [32], etc. In [32], we showed that the Window method is optimal under the following two conditions. One is that a power exponent e is uniformly randomly distributed, and the other is that only the doubling rule $a_i = 2a_{i-1}$ and the star chain rule $a_i = a_{i-1} + a_k$, $k < i$, are allowed in Eq. (7.2).

However, non-uniform cases have not been analyzed in details. In this chapter, we mainly treat non-uniform cases. In order to derive a short addition chain, we propose two methods, a Run-length method[1] and a Hybrid method, which are different from the known methods. We theoretically show that the Hybrid method is as efficient as the Extended Window method, which is optimal even for non-uniform cases under the above second condition, and the Run-length method is efficient when the Hamming weight of the binary representation of the power exponent e is large. It is also shown that the performance of the Hybrid method is better than the other known methods.

The Hybrid method and Run-length method are also applicable to elliptic curve cryptosystems [26][41]. However, the scalar multiplication over elliptic curves can use an addition-subtraction chain instead of an addition chain since an inverse element can be easily obtained. The computation rule of the addition-subtraction chain is given by $a_i = a_j \pm a_k$, $j, k < i$ instead of Eq. (7.2). An efficient method for obtaining a short addition-subtraction chain based on the canonical signed binary representation of e is described in [32].

Many algorithms speeding up the operation of multiplication have been proposed as reducing the number of multiplications as much as possible. One famous example is *Montgomery Reduction* [42]. We can simultaneously use these algorithms and our ones, and establish faster encryption and decryption.

In this chapter, the base of logarithm is always 2. Let $Seq = s_1 s_2 \cdots s_L, s_l \in \{0, 1\}$, be the ordinary binary representation of a positive integer e, and $(Seq)_{10} = e$. $|Seq|$ stands for the length of Seq, i.e., L, which is equal to $\lfloor \log e \rfloor + 1$. $w(e)$ represents the Hamming weight of e, in other words, $w(e)$ is the number of "1" in $Seq = s_1 s_2 \cdots s_L$, i.e., $w(e) = \sum_{i=1}^{L} s_i$. We assume that the bits "0" and "1" occur with the probability of p and q, respectively, i.e., $q = \Pr\{s_l = 1\} = w(e)/L$ and $p = \Pr\{s_l = 0\} = 1 - q$. The addition chain length r must satisfy $r \geq L + \log(qL) - 2.13$ for any e [66].

[1] Run-length method is firstly proposed in [31].

7.2 Window methods

In this section, we briefly review the Window method [25] and the Extended Window method with the Tunstall-like algorithm [32].

7.2.1 Window method

The addition of the Window method is restricted by the following two rules[2]:

$$\text{doubling rule: } b_i = b_{i-1} + b_{i-1} = 2b_{i-1}, \tag{7.3}$$

$$\text{star chain rule: } b_i = b_{i-1} + a_k,\ a_k \in \mathcal{D}, \tag{7.4}$$

where $\mathcal{D}$ is a set of all odd integers less than 2^κ. Hereafter, we call $\mathcal{D}$ as "dictionary". Note that the operations based on the doubling rule $b_i = 2b_{i-1}$ and star chain rule $b_i = b_{i-1} + a_k$ correspond to $M^{b_i} = (M^{b_{i-1}})^2$ and $M^{b_i} = M^{b_{i-1}} \cdot M^{a_k}$ in the power exponents, respectively.

The Window method is formulated as follows.

Input an integer e (or the binary representation of e, Seq).

Output an addition chain for e, $\langle c_0(=1), c_1, \cdots, c_r = e\rangle$.

Step 1 Determine the window size κ appropriately from L and $w(e)$.

Step 2 Make the shortest addition chain for the dictionary $\mathcal{D}$. In this case, it is given by $\langle 1, \underline{2}, 3, 5, \cdots, 2^\kappa - 1\rangle$, where the underline represents the number not included in the dictionary.

Step 3 [Main Procedure]

Process the sequence Seq from the most significant bit in the following two phases.

Phase 1 Read κ bits from Seq. Let the κ bits be $1s_1 \cdots s_{\kappa-l-1}\underbrace{0\cdots 0}_{l}$, where $s_{\kappa-l-1} = 1$ and $(1s_1 \cdots s_{\kappa-l-1})_{10} = a$. First apply the doubling rule $b_i = 2b_{i-1}$, $\kappa - l$ times. Next, apply the star chain rule $b_i = b_{i-1} + a$ and finally apply the doubling rule l times. Remove the κ bits from Seq.

Phase 2 If $Seq = \underbrace{0\cdots 0}_{l_0}1\cdots$, then apply the doubling rule $b_i = 2b_{i-1}$, l_0 times. Remove the l_0 bits from Seq and return to Phase 1.

Step 4 Concatenate $\langle 1, 2, 3, 5, \cdots, 2^\kappa - 1\rangle$ in Step 2 and $\{b_i\}$ in Phases 1 and 2 of Step 3 to obtain the whole addition chain $\langle c_0, c_1, \ldots, c_r\rangle$.

Note 7.1 After applying the doubling rule or the star chain rule, the index i is increased one. All methods described in Sections 7.2 and 7.3 obey this rule.

[2] All methods described in Sections 7.2 and 7.3 use these two rules. However, only the concatenation point of the addition chains obtained by Steps 2 and 3 described later can break this rule in all methods.

Note 7.2 The sequence is parsed by κ bits in Phase 1 of Step 3, but the parse length in Phase 2 depends on the sequence.

Note 7.3 When $\kappa = 1$, the above method is equivalent to the binary method [25].

Note 7.4 The average chain length is given as follows [32].

$$\left(L - \left(\kappa - \frac{p - p^{\kappa}}{1 - p}\right)\right) + \frac{L}{\kappa + \frac{p}{1-p}} + 2^{\kappa-1}, \tag{7.5}$$

where the first and second terms are the average numbers of the applied doubling and star chain rules in Step 3, respectively. The third term is the chain length for the dictionary $\mathcal{D}$ in Step 2. When $L = 512$ and $p = 1/2$, the optimal window size κ is equal to 5 and the average chain length is 609.

Note 7.5 The 2^{κ}-ary method [25] is similar to the Window method. This method can be easily implemented, but it is less efficient than the Window method. For instance, the chain length obtained by the 2^{κ}-ary method is $L-\kappa+L/\kappa+2^{\kappa}$, which becomes 640 for $\kappa = 5$ in the above case. The average chain length obtained by its variant, the Lou-Chang method [35], is 613 in the same case.

7.2.2 Extended Window method with Tunstall-like Algorithm

As we have seen in the previous subsection, the set of all odd positive integers less than 2^{κ} is adopted as the dictionary in the Window method. On the other hand, a more flexible dictionary can be used in the Extended Window Method (EWM) proposed in [32]. Moreover, the "Tunstall-like algorithm" was introduced to make the optimal dictionary in the same paper. In this subsection, we review the EWM and the Tunstall-like algorithm.

The EWM is formulated as follows.

Input an integer e (or the binary representation of e, Seq).

Output an addition chain for e, $\langle c_0(=1), c_1, \cdots, c_r = e\rangle$.

Step 1 Determine the primitive dictionary[3] $\mathcal{D}_s = \{Seq_0, Seq_1, \cdots Seq_K\}$ appropriately from L and $w(e)$ and obtain the dictionary $\mathcal{D} = \{a_0(=1), a_1, \ldots, a_K\} = \{(Seq'_j)_{10}\}_{j=0}^{K}$, where Seq'_j is the sequence obtained by removing trailing zeros, in Seq_j, ending at the least significant bit position[4].

Step 2 Make the shortest addition chain for $\mathcal{D}$.

Step 3 [Main Procedure]

Process the sequence Seq from the most significant bit in the following two phases.

Phase 1

[3]The most significant bit of each Seq_j is "1".

[4]If $Seq_j = 1s_1s_2\cdots s_l0\cdots 0$, then $Seq'_j = 1s_1s_2\cdots s_l$.

(a) Find Seq_j that is equal to the prefix of Seq.

(b) Apply the doubling rule $b_i = 2b_{i-1}$, $|Seq_j'|$ times.

(c) Apply the star chain rule $b_i = b_{i-1} + a_j$.

(d) Again apply the doubling rule $|Seq_j| - |Seq_j'|$ times.

(e) Remove the prefix of Seq.

Phase 2

(f) If the prefix of Seq is $\underbrace{0\cdots0}_{l_0}1$, then apply the doubling rule $b_i = 2b_{i-1}$, l_0 times.

(g) Remove the prefix $\underbrace{0\cdots0}_{l_0}$ from Seq and return to Phase 1.

Step 4 Concatenate the addition chains obtained in Steps 2 and 3.

Note 7.6 The primitive dictionary $\mathcal{D}_s$ is a set such that $\mathcal{D}_s$ and 0^*, zero sequences with arbitrary length, uniquely parse any Seq.

Note 7.7 In Step 2, it is possible to make the shortest addition chain for $\mathcal{D}$ unless K is very large. But it becomes harder as K becomes larger.

Note 7.8 In general, finding Seq_j in Phase 1 (a) of Step 3 is executed by using a tree obtained from the primitive dictionary determined in Step 1, and the optimal primitive dictionary is obtained by the Tunstall-like algorithm described later. Hence, the implement of the EWM is a little complicated than other methods, which can be implemented without a tree structure, such as the Window method, the Run-length method (described in Section 7.3.1) and the Hybrid method (described in Section 7.3.3).

Note 7.9 At the first parsing of Seq, (b) of Phase 1 can be skipped.

The optimal primitive dictionary $\mathcal{D}_s$ is constructed as follows.

Tunstall-like algorithm [32]

Input a dictionary size K, bit length of e, i.e. L, and the Hamming weight of e, i.e. $w(e)$.

Output a primitive dictionary $\mathcal{D}_s$.

Step 1 Make the root of a tree with weight 1. Set $q = w(e)/L$ and $p = 1 - q$.

Step 2 While the number of leaves is less than $K + 1$, repeat the following.
Let l and $weight(l)$ be the leaf with largest weight and its weight, respectively. Create two children with weight $p \times weight(l)$ and $q \times weight(l)$, and connect them to l via edges labeled with "0" and "1", respectively.

Step 3 Get binary sequences by reading along all paths from the root to all leaves. Then each element of $\mathcal{D}_s$ is obtained by concatenating "1" to each sequence as the most significant bit.

Note that the optimal dictionary depends on the Hamming weight.

In [32], the Tunstall-like algorithm is derived from the duality between the minimization problem of the chain length in the EWM and the optimization problem of variable-to-fixed length codes in data compression theory. We showed in [32] that the EWM reduces to the Window method when $p = 0.5$. Furthermore, it is also proved that the average chain length asymptotically converges $L + H(p)\frac{L}{\log L}$, where $H(p) \equiv -p\log p - (1-p)\log(1-p)$. This result and Eq. (7.5) imply that in case of $p < 0.5$, the addition chains of the EWM and the Window method become shorter and longer, respectively, as the Hamming weight becomes larger. Hence, the Window method is not always optimal, especially when e has a large Hamming weight. In the next section, we propose two new algorithms that can be more easily implemented than the EWM. In addition, the two methods are as efficient as the EWM in the case of a large Hamming weight, i.e., small p.

7.3 New methods

In this section, we propose the Run-length method and the Hybrid method. The former can make a shorter addition chain than the Window method when p is very small. The latter is a hybrid of the Window method and the Run-length method.

7.3.1 Run-length method

First, consider a case with small p. The optimal primitive dictionary obtained by the Tunstall-like algorithm, for example when $K = 512$ and $p = 0.1$, becomes

$$\mathcal{D}_s = \{10, 110, 1110, \cdots, \underbrace{1\cdots1}_{13}0, \underbrace{1\cdots1}_{14}\}.$$

If the above $\mathcal{D}_s$ is adopted, the rule of parsing in Phase 1 can be described as follows.

> Read Seq until bit "0" appears or the run-length of bit "1" becomes t, where t is the maximal run-length of "1".

The above example corresponds to the case of $t = 14$.

The EWM with the above primitive dictionary $\mathcal{D}_s = \{10, 110, \cdots, \underbrace{1\cdots1}_{t-1}0, \underbrace{1\cdots1}_{t}\}$, which gives the dictionary $\mathcal{D} = \{2^i - 1\}_{i=1}^{t}$, can be simplified as follows. The "Run-length method" is named from its similarity to the "Run-length method" in data compression theory.

Run-length method

Input and Output are the same as the EWM.

Step 1 Determine the maximal run-length t appropriately from L and $w(e)$.

Step 2 Make the shortest addition chain for $\mathcal{D} = \{2^i - 1\}_{i=1}^{t}$. In this case, it is given by $\langle 1, \underline{2}, 3, \cdots, 2^i - 1, \underline{2^{i+1} - 2}, 2^{i+1} - 1, \cdots, \underline{2^t - 2}, 2^t - 1\rangle$. Note that the underlines represent the numbers not included in $\mathcal{D}$. Simply speaking, the shortest chain for $\mathcal{D}$ is obtained by repeating, $t - 1$ times, the operations of doubling and adding 1.

Step 3 [Main Procedure]

Phase 1 Read Seq until bit "0" appears or the run-length of "1" becomes t. Let l be the run-length of "1". First apply the doubling rule l times. Next, apply the star chain rule $b_i = b_{i-1} + (2^l - 1)$. Remove the l bits from the prefix of Seq.

Phase 2 Read Seq until bit "1" appears. Let l_0 be the run-length of "0". Apply the doubling rule l_0 times. Remove the l_0 bits from the prefix of Seq and return to Phase 1.

Step 4 Concatenate the addition chains obtained in Steps 2 and 3.

Example 7.1:

As a simple example, we treat the case of $e = 445$ and $Seq = 110111101$. In Step 1, we set $t = 3$. In Step 2, the addition chain $\langle 1, \underline{2}, 3, \underline{6}, 7\rangle$ is obtained for $\mathcal{D} = \{1(= 2^1 - 1), 3(= 2^2 - 1), 7(= 2^3 - 1)\}$. In Step 3, the binary sequence Seq is parsed as follows. 11/0//111//1/0//1//, where "/" means the parsing of Phase 1 and "//" stands for the end of Phase 2. These parsed sequence lead to the addition chain, $\langle 3, 6, 12, 24, 48,_{(+7)} 55, 110,_{(+1)} 111, 222, 444,_{(+1)} 445\rangle$, where "," and ",$_{(+a)}$" mean that the doubling and star chain rules are applied, respectively. Hence, the final addition chain for 445 becomes

$$\langle 1, 2, 3, 6, 7, 12, 24, 48, 55, 110, 111, 222, 444, 445\rangle$$

in Step 4, and the addition chain length is 13. Hence, x^{445} can be calculated by 13 multiplications if we use the Run-length method with $t = 3$.

7.3.2 Average chain length of the Run-length method

In this subsection, we evaluate the average chain length attained by the Run-length method. We showed in [32] that the average chain length obtained by the "EWM" is given by

$$\left(L - \sum_{i=0}^{K} |Seq_i'| P_i\right) + \frac{L}{\sum_{i=0}^{K} |Seq_i| P_i + \frac{p}{1-p}} + (\text{chain length for } \mathcal{D}), \tag{7.6}$$

where P_i is the probability of Seq_i in Phase 1 of Step 3. The first and second terms are the average numbers of the doubling and star chain rules applied in Step 3, respectively. The third term is the chain length for the dictionary $\mathcal{D}$ in Step 2.

By letting $K = t - 1, Seq_i = \underbrace{1 \cdots 1}_{i+1}0$, $P_i = p \cdot q^i$ $(0 \le i < t-1), Seq_{t-1} = \underbrace{1 \cdots 1}_{t}$, $P_{t-1} = q^{t-1}$, and the third term $= 2(t-1)$, we obtain the following theorem.

Theorem 7.1 The average chain length for a positive integer e obtained by the Run-length method is given as

$$\left(L - \frac{1 - q^t}{p}\right) + \frac{L}{\frac{1-q^{t-1}}{p} + \frac{1}{q}} + 2(t-1), \tag{7.7}$$

where $L = \lfloor \log e \rfloor + 1$, $q = w(e)/L$, $p = 1 - q$, and t is a parameter to be optimized.

When $L = 512, p = 0.1$, and $t = 14$, the above value becomes 590.0 while the chain length obtained by the Window method ($\kappa = 6$) is 621.9.

7.3.3 Hybrid method

The optimal primitive dictionary obtained by the Tunstall-like algorithm, for instance when $L = 512$ and $p = 0.15$, becomes

$$\mathcal{D}_s = \{100, 1010, 1100, 10110, 10111, 11010, 11011,$$
$$11100, 11101, 11110, \underbrace{1\cdots1}_{5}0, \ldots, \underbrace{1\cdots1}_{14}0, \underbrace{1\cdots1}_{15}\}. \tag{7.8}$$

The average chain length attained by the above primitive dictionary is only 602.9 while the chain length obtained by the Window method ($\kappa = 6$) is 621.0. However, the EWM with this primitive dictionary has two demerits compared with the Window method. One is Step 3, especially Phase 1 (a), which is more complicated than that in the Window method. The reason is that the EWM needs to use a tree structure in order to quickly find Seq_j since it is a little cumbersome to find Seq_j from many dictionary sequences. The other is the difficulty of obtaining the shortest addition chain for the dictionary in Step 2. In order to remove these defects, we introduce a simple primitive dictionary $\widetilde{\mathcal{D}_s}$, which is similar to the above $\mathcal{D}_s$.

$$\widetilde{\mathcal{D}_s} = \{1\{s_1 s_2 \cdots s_{\kappa-1}\}^*, \underbrace{1\cdots1}_{\kappa}0, \underbrace{1\cdots1}_{\kappa+1}0, \ldots, \underbrace{1\cdots1}_{t-1}0, \underbrace{1\cdots1}_{t}\}, \tag{7.9}$$

where $\{s_1 \cdots s_{\kappa-1}\}^*$ are all $(2^{\kappa-1} - 1)$ sequences with length $\kappa - 1$ excluding $\underbrace{1\cdots1}_{\kappa-1}$.

Note 7.10 When $\kappa = 5, t = 15$, $\widetilde{\mathcal{D}_s}$ of (7.9) becomes "similar" to $\mathcal{D}_s$ given by (7.8).

Note 7.11 The primitive dictionary $\widetilde{\mathcal{D}_s}$ and 0^* can uniquely parse any sequences.

If $\widetilde{\mathcal{D}_s}$ is used instead of $\mathcal{D}_s$, the EWM's defects described above are removed in the following way. Step 3 can be simplified because the parsing rule with $\widetilde{\mathcal{D}_s}$ can be described as follows.

> Read κ bits from Seq. If all κ bits are "1", read Seq until "0" appears or the run-length of "1" becomes $t - \kappa$ and parse Seq there.

Note that this parsing rule can be easily implemented without a tree structure.

Furthermore, the shortest optimal addition chain for the dictionary is easily obtained. Note that the dictionary $\widetilde{\mathcal{D}}$ for $\widetilde{\mathcal{D}_s}$ is given by

$$\{1, 3, 5, \ldots, 2^\kappa - 3, 2^\kappa - 1, 2^{\kappa+1} - 1, 2^{\kappa+2} - 1, \ldots 2^t - 1\},$$

or, in short, $\widetilde{\mathcal{D}} = \{\{2i-1\}_{i=1}^{2^{\kappa-1}}, \{2^{\kappa+i} - 1\}_{i=1}^{t-\kappa}\}$. The optimal addition chain for $\widetilde{\mathcal{D}}$ is easily obtained as $\langle 1, \underline{2}, 3, 5, \ldots, 2^\kappa - 3, 2^\kappa - 1, \underline{2^{\kappa+1} - 2}, 2^{\kappa+1} - 1, \ldots, \underline{2^t - 2}, 2^t - 1\rangle$, where the underlined numbers are not included in $\widetilde{\mathcal{D}}$. This chain length is $2^{\kappa-1} + 2(t - \kappa)$.

The above dictionary is equivalent to the dictionary of the Window method when $\kappa = t$. Furthermore, it is equivalent to the dictionary of the Run-length method when $\kappa = 2$. Hence, the following method is a hybrid of the Window method and the Run-length method.

Hybrid method

Input and Output are the same as the EWM.

Step 1 Determine the parameters κ and t appropriately from L and $w(e)$.

Step 2 Make the shortest addition chain for $\mathcal{D} = \{\{2i-1\}_{i=1}^{2^{\kappa-1}}, \{2^{\kappa+i}-1\}_{i=1}^{\kappa-t}\}$. In this case, it is given by $\langle 1, 2, 3, 5, \cdots, 2^{\kappa}-3, 2^{\kappa}-1, 2^{\kappa+1}-2, \cdots, 2^{t}-2, 2^{t}-1\rangle$.

Step 3 [Main Procedure]

Phase 1 Read κ bits from Seq. If bit "0" is included in the κ bits, execute (a). Otherwise, execute (b).

(a) Assume that the κ bits be $1s_1 \cdots s_{\kappa-l-1}\underbrace{0\cdots 0}_{l}$, where $s_{\kappa-l-1}=1$ and $(1s_1\cdots s_{\kappa-l-1})_{10} = a$. Then, apply the doubling rule $\kappa - l$ times, the star chain rule $b_i = b_{i-1} + a$ once, and the doubling rule l times in this order. Remove the κ bits from the prefix of Seq.

(b) Read Seq until bit "0" appears or the run-length of bit "1" becomes $t-\kappa$. Let l be the total run-length of "1". Then, apply the doubling rule l times, and the star chain rule $b_i = b_{i-1} + (2^l - 1)$. Remove the l bits from the prefix of Seq.

Phase 2 Read Seq until bit "1" appears. Let l_0 be the run-length of "0". Then, apply the doubling rule l_0 times. Remove the l_0 bits from the prefix of Seq and return to Phase 1.

Step 4 Concatenate the addition chains obtained in Steps 2 and 3.

Note that (a) and (b) correspond to Phase 1 of Step 3 in the Window and Run-length methods, respectively.

Example 7.2:

Consider $e = 75064310$,

$$Seq = 10001111001011000111111010 110.$$

In Step 1, we set $\kappa = 3$ and $t = 5$. In Step 2, the addition chain $\langle 1, \underline{2}, 3, 5, 7, \underline{14}, 15, \underline{30}, 31\rangle$ is derived for $\mathcal{D} = \{1, 3, 5, 7, 15, 31\}$. In Step 3, Seq is parsed as follows.

$$100/0//1111/00//101//100/0//11111//101//10//,$$

where "/" and "//" are the same as the example treated in section 3.1. This parsed sequence leads to the following chain,

$\langle 1, 2, 4, 8, 16, 32, 64, 128_{,(+15)}\ 143, 286, 572, 1144, 2288, 4576_{,(+5)}\ 4581, 9162_{,(+1)}\ 9163, 18326,$
$36652, 73304, 146608, 293216, 586432, 1172864, 2345728_{,(+31)}\ 2345759, 4691518, 9383036,$
$18766072_{,(+5)}\ 18766077, 37532154_{,(+1)}\ 37532155, 75064310(=e)\rangle$.

In Step 4, the whole addition chain is obtained by concatenating the chains obtained in Steps 2 and 3. Hence, the total chain length for e becomes 8 (Step 2) + 32 (Step 3) = 40 in this example.

Note 7.12 When $\kappa = 1$ and $t = 1$, Hybrid method is equivalent to the binary method.

7.3.4 Average chain length of the Hybrid method

The average chain length obtained by the Hybrid method can also be derived from Eq. (7.6). Let $K = 2^{\kappa-1} + t - \kappa - 1$. In case of $Seq_i = 1\{s_1 \cdots s_{\kappa-1}\}^*$, $0 \le i \le 2^{\kappa-1} - 2$, where $\{s_1 \cdots s_{\kappa-1}\}^*$ represents all $(2^{\kappa-1} - 1)$ sequences excluding $\underbrace{1 \cdots 1}_{\kappa-1}$, P_i in Eq. (7.6) is given by $P_i = p^{w_1} \cdot q^{w_2}$ where w_1 and w_2 are the numbers of "0" and "1" included in $s_1 \cdots s_{\kappa-1}$, respectively. Furthermore,

$$P_i = p \cdot q^{i-2^{\kappa-1}+\kappa} \text{ if } Seq_i = \underbrace{1 \cdots 1}_{\kappa \cdots t-1}0,\ 2^{\kappa-1} - 1 \le i \le 2^{\kappa-1} + t - \kappa - 2 (= K - 1),$$

and

$$P_K = q^{t-1} \text{ if } Seq_K = \underbrace{1 \cdots 1}_{t}.$$

Finally, since the third term is equal to $2^{\kappa-1} + 2(t - \kappa)$, we obtain the following theorem.

Theorem 7.2 The average chain length for positive integer e obtained by the Hybrid method is given by

$$\left(L - (\kappa - \frac{p - p^\kappa}{1-p} + \frac{q^\kappa - q^t}{p})\right) + \frac{L}{\kappa + \frac{q^{\kappa-1} - q^{t-1}}{p} + \frac{p}{1-p}} + 2^{\kappa-1} + 2(t - \kappa), \qquad (7.10)$$

where $L = \lfloor \log e \rfloor + 1$, $q = w(e)/L$, and $p = 1 - q$. κ and t are parameters to be optimized.

When $L = 512, p = 0.15, \kappa = 5$, and $t = 15$, the above value is 605.0. This value is a little larger than the chain length 602.9, which is attained by the EWM with the primitive dictionary given by (7.8). However, the parsing in the Hybrid method is much easier than the EWM as described in Section 7.3.3. In addition, the case of $\kappa = 5$ and $t = 15$ is not optimal for the Hybrid method. In fact, the optimal parameters are $\kappa = 4$ and $t = 12$ and the minimum value is 601.2. See Table 7.1.

Although the optimal parameters (κ, t) must be chosen in Step 1, they can easily be determined by evaluating Eq. (7.10). It is worth noting that Eq. (7.10) is equivalent to Eq. (7.5), which is the average chain length by the Window method, for $\kappa = t$, and Eq. (7.10) is equal to Eq. (7.7), which is the average chain length by the Run-length method, for $\kappa = 2$.

7.4 Numerical results

In this section, we show some numerical results for $L = 512$. Table 7.1 and Figure 7.1 represent the minimum values of average addition chain length obtained by the EWM with the Tunstall-like algorithm, the Window method, the Run-length method, and the Hybrid method, which are calculated from Eqs. (7.6), (7.5), (7.7), and (7.10), respectively. For each method, the optimal size of $\mathcal{D}$ and parameters κ, t, or (κ, t) that minimize the theoretical average chain length are used for each p.

In order to confirm the validity of the above theoretical results, we have some simulation results, which are obtained in the following way. Random sequences are generated with

Table 7.1. Performance of four methods

p	$w(e)$	EWM		Window		Run-length		Hybrid	
		length	$\#\mathcal{D}$	length	κ	length	t	length	(κ, t)
0.95	25	536.6	1	536.6	1	536.6	1	536.6	(1,1)
0.90	51	554.2	6	557.4	3	559.5	2	557.4	(3,3)
0.85	77	567.4	7	571.2	4	579.6	2	571.2	(4,4)
0.80	102	576.8	10	582.0	4	597.3	3	582.0	(4,4)
0.75	128	585.2	14	589.1	5	612.2	3	589.1	(5,5)
0.70	154	591.7	15	594.6	5	625.0	4	594.6	(5,5)
0.65	179	598.0	21	599.2	5	634.7	4	599.2	(5,5)
0.60	205	602.8	21	603.1	5	642.5	4	603.1	(5,5)
0.55	230	606.0	17	606.4	5	647.3	5	606.4	(5,5)
0.50	256	609.2	17	609.3	5	650.1	5	609.3	(5,5)
0.45	282	613.1	22	611.8	5	650.2	6	611.8	(5,5)
0.40	307	613.6	22	614.0	5	648.0	7	613.9	(5,6)
0.35	333	615.1	21	616.0	5	643.5	8	615.0	(5,6)
0.30	358	614.3	19	617.7	5	636.8	9	614.9	(5,7)
0.25	384	611.7	19	619.2	6	628.0	10	613.2	(5,8)
0.20	410	608.6	22	620.2	6	617.0	11	609.4	(5,10)
0.15	435	602.9	21	621.1	6	604.4	12	601.2	(4,12)
0.10	461	590.0	14	621.9	6	590.0	14	589.2	(3,14)
0.05	486	574.2	17	622.6	6	574.2	17	573.9	(3,17)

a given $p = \Pr\{s_l = 0\}$ and addition chains are constructed by the above four methods. The optimal parameters are searched for each method to attain the minimum value of the average chain length.

These simulation values coincide with Table 7.1 very well except that each values are larger than Table 7.1 by about 0.4. This difference is caused from the fact that the last parsing is stopped by the end of a sequence.

From Table 7.1 and Figure 7.1, the compared three methods have the following properties.

1. The Hybrid method can generate an addition chain whose length is almost equal to that of the EWM with the Tunstall-like algorithm for any p.

2. In $0 < p < 0.1$, the average chain length obtained by the Run-length method is almost equal to the EWM and much smaller than the Window method.

3. In $0.1 \leq p \leq 0.4$, the Hybrid method can generate a shorter addition chain than both the Window method and the Run-length method.

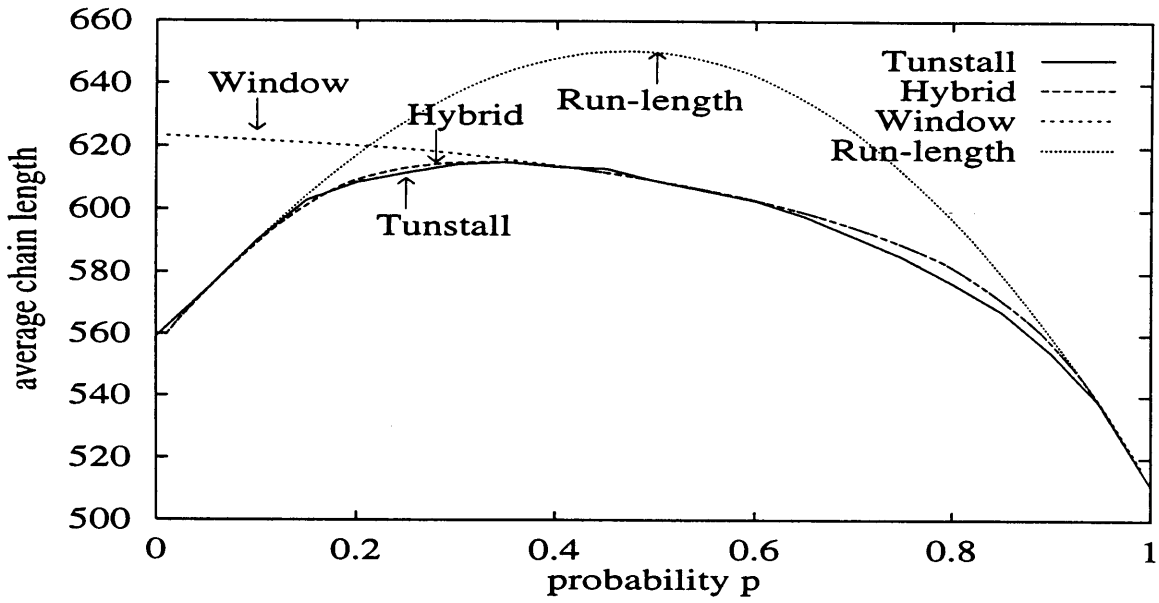

Figure 7.1. Comparison of four methods

4. In $0.4 < p < 1$, the performance of the Hybrid method is almost equal to the Window method.

Note that when $p = 0.45$, 0.35, 0.15, 0.10, and 0.05, the Hybrid method generates a shorter chain than the EWM with the Tunstall-like algorithm. This is caused from the fact that the EWM includes more extra numbers in the addition chain for $\mathcal{D}$ than the Hybrid method in the above cases.

Next, we compare the Window method and the Hybrid method for the case of a large Hamming weight ($p = 0.05$) and larger exponents ($L = 1024, 2048$, and 4096). Table 7.2 shows the average chain length obtained by these two methods. In Table 7.2, Speedup1 is calculated by

$$\text{Speedup1} = 1 - \frac{\text{(Hybrid method)}}{\text{(Window method)}}.$$

In all cases, the Hybrid method can generate a shorter addition chain than the Window method by 7−8%. We showed in [32] that the average chain length asymptotically converges $L + H(p)\frac{L}{\log L}$ in using EWM. Hence, we compare the difference between the average chain length obtained by two methods and the above convergence. In Table 7.2, Speedup2 is calculated by

$$\text{Speedup2} = 1 - \frac{\text{(Hybrid method)} - \left(L + H(p)\frac{L}{\log L}\right)}{\text{(Window method)} - \left(L + H(p)\frac{L}{\log L}\right)}.$$

From the above observation and our theoretical analysis in Section 7.3.3, it follows that the Hybrid method is more efficient and practical compared to the Window method, the EWM with the Tunstall-like algorithm, and the Run-length method.

Table 7.2. Performance of the Window method and the Hybrid method when $p = 0.05$

L	Window method		Hybrid method		Speedup1	Speedup2
	length	κ	length	(κ, t)	(%)	(%)
512	622.6	6	573.9	$(3, 17)$	7.8	52
1024	1219.2	6	1124.0	$(3, 22)$	7.8	57
2048	2395.4	7	2213.7	$(3, 30)$	7.6	62
4096	4724.7	8	4379.0	$(3, 39)$	7.3	65

Finally, we compare the Hybrid method with two other methods: the Bos-Coster method [9] and the Yacobi method [67]. The Bos-Coster method is similar to the Window method. However, their method uses a large window (for example, $\kappa = 10$) and must make a short addition chain for a large dictionary $\mathcal{D}$ with many elements, which is a hard task. Hence, this method is not practical and not suited for implementation. The Yacobi method uses an adaptive dictionary such as the EWM. However, since his method is based on the LZ78 code in data compression theory and the primitive dictionary $\mathcal{D}_s$ is created unboundedly by the incremental parsing, the chain length cannot become short for practical size exponents, e.g., $L = 512, 1024,$ and 2048.

These results lead us to the conclusion that the Hybrid method is more efficient than the known methods.

7.5 Conclusion

In this chapter, we have proposed the Run-length method and the Hybrid method. We theoretically analyzed these two methods and showed that the Hybrid method is more efficient and practical, especially in the case of the large Hamming weight, than known methods. Roughly speaking, the Hybrid method can attain 8% reduction, in the best case, in the average addition chain length compared with the Window method.

Chapter 8

Conclusions

In this thesis, we investigated the security and efficiency of public key cryptosystems.

As for the security, we propose a new elliptic curve factoring method in Chapter 3. We show its efficiency by theoretical analysis. In Chapter 4, we analyze the security of a certain type of elliptic curve cryptosystem defined over a composite modulus. We also investigate the difficulty of a certain problem — the problem of counting the number of points on an elliptic curve over the ring $\mathbf{Z}/n\mathbf{Z}$ —. This problem is assumed to be as difficulty as the cryptosystem is to break. We prove that this problem is computationally equivalent to the factoring problem. In Chapter 5, we investigate the difficulty of an elliptic curve discrete logarithm problem over a super-anomalous elliptic curve. We prove that this problem can be solved in deterministic polynomial time. In Chapter 6, the multi-variate RSA cryptosystem is defined and its security and efficiency are evaluated. We prove that this cryptosystem can be broken under unusual usages.

As for the efficiency, we describe how to speed up public key cryptosystems in Chapter 7. We propose new methods to generate short addition chains. Moreover, we evaluate the efficiency of these methods.

8.1 Summary

The results are summarized as follows.

1. An Elliptic Curve Method (ECM) is an effective integer factoring method. We propose a new ECM, which runs faster than the original ECM, and analyze its efficiency. The new ECM is effective if we assume that there exists an algorithm to determine whether the number of points on an elliptic curve defined over the residue ring is divided by a given integer. Let p be a prime factor of the composite to be factored. We showed that the new ECM runs 3 or 4 times faster than the original ECM if p lies between 10^{30} to 10^{50} on average (Table 3.1). Moreover, we showed that the new ECM asymptotically runs $(\log p)^{0.318}$ faster than the original ECM (Theorem 3.6).

2. The KMOV scheme is one of of RSA-type cryptosystems. The security of the KMOV scheme is closely related to the difficulty of the following mathematical problem – the problem of counting the number of points on an elliptic curve over the ring $\mathbf{Z}/n\mathbf{Z}$ –. We investigated the difficulty of this problem and proved that this problem is computationally equivalent to factoring problem (Theorem 4.1). Next, we investigated an elliptic curve discrete logarithm problem over the ring $\mathbf{Z}/n\mathbf{Z}$. We proved that we can solve the factoring problem if we can solve this problem (Theorem 4.3). In addition, we proved that we cannot solve this problem even if we can solve the factoring problem (Theorem 4.4). Moreover, we discussed the problem described in Section 3.3: the problem of determining whether the number of points over $E(\mathbf{Z}/n\mathbf{Z})$ is divided by a given number (Theorem 4.2).

3. First, we defined a super-anomalous elliptic curve. Next, we investigated the difficulty of an elliptic curve discrete logarithm problem over this curve. We proved that we can solve this problem in deterministic polynomial time (Theorem 5.1, Theorem 5.4). (Note that we can do it without knowing prime factors.) Moreover, we described two algorithms able to solve this problem and mathematically proved that these algorithms output the correct answer. Finally, we mentioned a cryptosystem whose security is based on the difficulty of this problem and showed that we can easily break this cryptosystem (Section 5.5).

4. First, we defined a Multi-variate RSA cryptosystem by extending an RSA cryptosystem. Next, we investigated the efficiency and security of this cryptosystem. We showed that the speed of this cryptosystem in encryption and decryption is faster than the original RSA (Section 6.2.4). With regard to its security, we proved that the difficulty of obtaining a whole message is computationally equivalent to the difficulty of breaking the original RSA (Theorem 6.1). Furthermore, we proposed that we can easily obtain the whole message in the following three cases.

   - An attacker knows one block of the message (Theorem 6.2).
   - An attacker knows the non-trivial algebraic relationship between some blocks of the message (Theorem 6.3).
   - A sender sends a message related to a previously sent message (Theorem 6.4).

5. We can speed up the encryption and decryption of public key cryptosystems by using a short *addition chain*. We proposed two algorithms, – the Run-length method (Section 7.3.1) and Hybrid method (Section 7.3.3) –, which generate short addition chains. We theoretically analyzed their performance (Theorem 7.1, Theorem 7.2) and showed that these methods are effective in numerical experiments. We showed that if the Hamming weight of the binary representation of exponents is large, these methods are especially effective. Let q be the probability that the bit 1 occurs within the binary representation. Concretely speaking, we showed the following (Section 7.4).

(a) If $0.8 < q < 1$, the Run-length method is best.

(b) If $0.6 < q < 0.8$, the Hybrid method is best.

(c) If $0 < q < 0.6$, the Window method is best.

8.2 Further research

As for the security, a lot of problems still remain unsolved as shown in Chapter 1. The followings are their famous examples.

- Are there factoring algorithms able to run in polynomial time?
- Is breaking RSA cryptosystem computationally equivalent to the factoring problem?
- Are there algorithms for solving discrete logarithm problem able to run in polynomial time?
- Is breaking ElGamal cryptosystem computationally equivalent to the discrete logarithm problem?
- Are there algorithms for solving elliptic curve discrete logarithm problem able to run in polynomial time or sub-exponential time?

Acknowledgment

The author would like to express his thanks to Professor Hirosuke Yamamoto for his suggestion and critically reading the manuscript. He also would like to thank Professor Hideki Imai, Professor Kokichi Sugihara, Professor Kazuyuki Aihara, and Associate Professor Satoru Iwata for useful and helpful comments.

He would like to appreciate the kindly support and encouragement for writing this thesis by NTT, especially Dr. Koh'ichi Matsuda, the ex-Executive manager of NTT Science and Core Technology Laboratory group, and Dr. Yoh'ichi Tohkura, the Executive manager of NTT Science and Core Technology Laboratory group, Dr. Kenichiro Ishii, the Director of NTT communication Science Laboratories, Dr. Kiyoshi Kogure, the ex-leader of our research group, and Dr. Kiyoshi Shirayanagi, the leader of our research group.

The Chapters 3, 4, 5, and 6 of this thesis are the joint works of the late Dr. Kenji Koyama, the ex-leader of our research group. The author is also thankful to him.

The author would like to express his sincere thanks to his parents for their continual support and encouragement.

Bibliography

[1] L. M. Adleman, "A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography," Proc of FOCS, pp. 55-60, 1979.

[2] K. Araki, T. Satoh, and S. Miura, "Overview of Elliptic Curve Cryptography," Proc. of Public Key Cryptography, LNCS 1431, Springer-Verlag, pp. 27–49, 1998.

[3] A. O. L. Atkin and F. Morain, "Finding Suitable Curves for the Elliptic Curve Method of Factorization", Mathematics of computation, vol.60, no. 201, pp. 399–405, 1993.

[4] E. Bach, "Discrete logarithm and factoring," Computer Science Division (EECS), University of California, UCB/CSD 84/186, 1984.

[5] D. Boneh and G. Durfee, "Cryptanalysis of RSA with Private Key d Less than $N^{0.292}$," Proc. of Eurocrypt'99, LNCS 1592, Springer-Verlag, pp. 1–11, 1999.

[6] D. Boneh and R. Venkatesan, "Breaking RSA May Not Ne Equivalent to Factoring," Proc. of Eurocrypt'98, LNCS 1403, Springer-Verlag, pp. 59–71, 1998.

[7] E. Bach and R. Peralta "Asymptotic Semismoothness Probabilities," Mathematics of computation, vol.65, no. 216, pp. 1701–1715, 1996.

[8] D. Bleichenbacher, "On the Security of the KMOV Public Key Cryptosystem," Proc. of CRYPTO'97, LNCS 1294, pp. 235–248, 1997.

[9] J. Bos and M. Coster, "Addition chain heuristics," Advances in Cryptology – CRYPTO'89, LNCS 435, pp. 400–407, 1989.

[10] L. Charlap, R. Coley, and D. Robbins, "Enumerate of rational points on elliptic curves over finite fields," preprint, 1991.

[11] D. Coppersmith, "Fast Evaluation of Logarithms in Fields of Characteristic Two," IEEE Trans. on Information Theory, IT-30, pp.587–594, 1984.

[12] D. Coppersmith, A. M. Odlyzko, and R. Schroeppel, "Discrete Logarithms in $GF(p)$," *Algorithmica*, Vol. 1, pp.1–15, 1986.

[13] D. Coppersmith, M. Franklin, J. Patarin, and M. Reiter, "Low-exponent RSA with related messages," Proc. of EUROCRYPT'96, LNCS 1070, pp. 1–9, 1996.

[14] N. Demytko, "A new elliptic curve based analogue of RSA," Proc. of EUROCRYPT'93, LNCS765, pp.40-49, 1994.

[15] W. Diffie and M. Hellman, "New Directions in Cryptography," IEEE Trans. on Information Theory, IT-22, 6, pp. 644–654, 1976.

[16] P. Downey, B. Leong and R. Sthi, "Computing sequences with addition chains," SIAM J. Computing, vol.10, no.3, pp.638–646, 1981.

[17] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Trans. on Information Theory, IT-31, no. 4, pp. 469–472, 1985.

[18] T. ElGamal, "A Subexponential-Time Algorithm for Computing Discrete Logarithms over $GF(p^2)$," IEEE Trans. on Information Theory, IT-31, pp. 473–481, 1985.

[19] D. M. Gordon, "Discrete Logarithm in $GF(p)$ Using the Number Field Sieve," to appear in SIAM Journal on Discrete Math.

[20] D. M. Gordon, "Designing and Detecting Trapdoors for Discrete Log Cryptosystems," Proc. of CRYPTO'92, LNCS 740, pp. 66-75, 1992.

[21] R. K. Guy, "How to factor a number," Proc. of the Manitoba Conf. on Numerical Math., pp. 49–89, 1975.

[22] G. H. Hardy and E. M. Wright, "An introduction to the theory of numbers," Oxford Univ. Press, 1979.

[23] J. Håsted, "On Using RSA with Low Exponent in a Public-Key Network," Proc. of CRYPTO'85, pp.403–408, 1986.

[24] T. Izu, "Generating Elliptic Curves Suitable for Factorization,", Proc. of SCIS2000, SCIS2000-b20, 2000 (in Japanese).

[25] D. E. Knuth, "Seminumerical algorithm (arithmetic) The art of Computer Programming vol.2," Addison Wesley, 1981.

[26] N. Koblitz, "Elliptic Curve Cryptosystems," Mathematics of Computation, 48, 177, pp. 203–209, 1987.

[27] K. Koyama, U. Mauer, T. Okamoto and S. Vanstone, "New public-key schemes based on elliptic curves over the ring Z_n," Proc. of CRYPTO'91, LNCS576, pp.345–357, 1992.

[28] K. Koyama, "Fast RSA-type scheme based on singular curves $y^2 + axy \equiv x^3 \pmod{n}$," Proc. of EUROCRYPT'95, LNCS921, pp.329–340, 1995.

[29] N. Kunihiro, "Study on fast algorithms for exponentiation," M. E. Thesis, Dept. of Math. Eng. and Inform.Physics, Univ. of Tokyo, 1996 (in Japanese).

[30] N. Kunihiro and K. Koyama, "Equivalence of Counting the Number of Points on Elliptic Curve over the Ring $\mathbf{Z}_n$ and Factoring n," Proc. of Eurocrypt'98, LNCS 1403, Springer-Verlag, pp. 47–58, 1998.

[31] N. Kunihiro and H. Yamamoto, "Optimal addition chain classified by Hamming weight", Technical report of IEICE, ISEC96-74, pp.127–132, 1997 (in Japanese).

[32] N. Kunihiro and H. Yamamoto, "Window and Extended Window Methods for Addition Chain and Addition-Subtraction Chain", IEICE Trans. Fundamentals, vol. E81-A, no. 1, pp.72–81, 1998.

[33] H. W. Lenstra, "Factoring Integers with Elliptic Curves," Annals of Mathematics, vol. 126, pp. 649–673, 1987.

[34] A. K. Lenstra, H. W. Lenstra, M. S. Manasse, and J. M. Pollard, "The Number Field Sieve," Proc. of STOC, pp.564–572, 1990.

[35] D. C. Lou and C. C. Chang, "An adaptive exponentiation method," The Journal of Systems and Software, vol. 42, pp. 59–69, 1998.

[36] D. L. Long, "Random Equivalence of Factorization and Computation of Orders," Princeton University, Department of Electrical Engineering and Computer Science, no.284, 1981.

[37] B. Mazur, "Rational isogenies of prime degree," Invent. Math. vol.44, pp. 129–162, 1978.

[38] A. Menezes, T. Okamoto, and S. Vanstone, "Reducing Elliptic Curve Logarithms to Logarithms in a Finite Fields," Proc. of STOC, pp. 80–89, 1991.

[39] A. Menezes, "Elliptic Curve Public Key Cryptosystems," Kluwer Academic Publishers, 1993.

[40] G. L. Miller, "Riemann's hypothesis and tests for primality," Journal of Computer and System Sciences, vol. 13, pp.300-317, 1976.

[41] V. S. Miller, "Use of Elliptic Curves in Cryptography," Proc. of Crypto'85, LNCS 218, Springer-Verlag, pp. 417–426, 1985.

[42] P. L. Montgomery, "Modular Multiplication Without Trial Division," Mathematics of computation, vol.44, no. 170, pp. 519–521, 1985.

[43] P. L. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization," Mathematics of computation, vol.48, no. 177, pp. 243–264, 1987.

[44] L. J. Moredll, "On the rational solutions of the indeterminate equations of the 3rd and 4th degree," Proc. Camb. Phil. Soc., pp. 179–192, 1992.

[45] M. A. Morrison and J. Brillhart, "A method of factoring and the factorization of F_7," Mathematics of computation, vol.29, pp. 183–205, 1975.

[46] National Bureau of Standards (U. S.). "Data Encryption Standard," Federal Information Processing Standards Publication 46, National Technical Information Services, Springfield, VA, 1977.

[47] T. Okamoto and K. Ohta, "Cryptography, Zero-knowledge Proof and Number Theory," Kyoritu Syuppan, Tokyo, 1995.

[48] T. Okamoto and S. Uchiyama, "Security of an Identity-Based Cryptosystem and the Related Reductions," Proc. of Eurocrypt'98, LNCS 1403, Springer-Verlag, pp. 546–560, 1998.

[49] J. M. Pollard, "Theorems on factorization and primality testing," Proc. Camb. Philos. Soc., 76, pp. 521–528, 1974.

[50] J. M. Pollard, "A Monte Carlo method for factorization," BIT, 15, pp. 331–334, 1975.

[51] C. Pomerance, "Analysis and comparison of some integer factoring algorithms," in Number Theory and Computers, Math. Centrum Tracts, Number 154, Part I and Number 155, Part II, Amsterdam, pp. 89–139, 1983.

[52] C. Pomerance, "Fast, Rigorous Factorization and Discrete Logarithm Algorithm, Discrete Algorithms and Complexity," Proc. of the Japan-U.S. Joint Seminar, Academic Press, pp. 119-143, 1987.

[53] M. O. Rabin, "Digital Signatures and Public-Key Encryptions as Intractable as Factorization," MIT, Technical Report, MIT/LCS/TR-212, 1979.

[54] M A. Reichert, "Explicit Determination of Nontrivial Torsion Structures of Elliptic Curves Over Quadratic Number Fields," Mathematics of computation, vol.46, no. 174, pp. 637–658, 1986.

[55] H. G. Rück, "On the Discrete Logarithm in the Divisor Class Group of Curve," Mathematics of Computation, 68, 226, pp. 805–806, 1999.

[56] P. Riebenboim, "The little book of big primes," Springer–Verlag, 1991.

[57] R.L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signature and public key cryptosystems," *Comm. of ACM*, vol.21 no. 2, pp.120–126, 1978.

[58] T. Satoh and K. Araki, "Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curve," Commentarii Math. Univ. Sancti Pauli, 47, 1, pp. 81-92, 1998.

[59] T. Satoh and K. Araki, "Errata to the paper: Fermat Quotients and the Polynomial Time Discrete Log Algorithm for Anomalous Elliptic Curve," to appear.

[60] R. Schoof, "Elliptic Curves over Finite Fields and the Computation of Square Roots mod p," Mathematics of computation, vol.44, no. 170, pp. 483–494, 1985.

[61] I. A. Semaev, "Evaluation of Discrete Logarithms in a Group of p-torsion Points of an Elliptic Curve in Characteristic p," Mathematics of Computation, 67, 221, pp. 353–356, 1998.

[62] J. Silverman, "The Arithmetic of Elliptic Curves," GTM 106, Springer–Verlag, 1986.

[63] J. H. Silverman and J. Tate, "Rational Points on Elliptic Curves," Springer–Verlag, Berlin, 1992.

[64] R. D. Silverman and S. S. Wagstaff Jr., "A Practical Analysis of the Elliptic Curve Factoring Algorithm," Mathematics of computation, vol.61, no. 203, pp. 445–462, 1993.

[65] N. P. Smart, "The Discrete Logarithm Problem on Elliptic Curves of Trace One," to appear in Journal of Cryptology.

[66] A. Schönhage, "A lower bound for the length of addition chains," Theoretical Computer Science, vol.1, pp.1–12, 1975.

[67] Y. Yacobi, "Exponentiating faster with addition chains," Advances in Cryptology – EUROCRYPT'90, LNCS 473, pp. 222–229, 1990.

[68] M. Wiener, "Cryptanalysis of short RSA secret exponents," IEEE Trans. on Information Theory, IT-36, no. 3, pp. 553–558, 1990.

[69] H. Woll, "Reductions among Number Theoretical Problems," Information and Computation, vol.72, pp.167–179, 1987.

List of Publication

1. N. Kunihiro, Y. Tsuruoka, and K. Koyama, "Integer factorization based on elliptic curves whose orders have a given factor", Proc. of SCIS'97, 15-B, 1997 (in Japanese).

2. N. Kunihiro and K. Koyama, "Equivalence of Counting the Number of Points on Elliptic Curve over the Ring $\mathbf{Z}_n$ and Factoring n", Proc. of Eurocrypt'98, LNCS 1403, Springer-Verlag, pp. 47–58, 1998.

3. N. Kunihiro and K. Koyama, "Two Discrete Log Algorithms for Super-Anomalous Elliptic Curves and Their Applications", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E83-A, no.1, pp. 10–16, 2000.

4. N. Kunihiro and K. Koyama, "Multi-variate RSA Cryptosystems and their Security Analyses", Proc. SCIS2000, A14, 2000.

5. N. Kunihiro and H. Yamamoto, "New methods for generating short addition chains ", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E83-A, no.1, pp. 60–67, 2000.

List of Tables

List of Figures

Appendix A

Proofs of theorems and lemmas in Chapter5

Proof of Lemma 5.1.

We denote the p-adic number field and the ring p-adic integers by $\mathbf{Q}_p$ and $\mathbf{Z}_p$. Let λ'_{p^e} be a composition of the following maps,

$$\lambda'_{p^e} : \tilde{E}(\mathbf{Z}/p^e\mathbf{Z}) \xrightarrow{u} E(\mathbf{Q}_p) \xrightarrow{h_{p^e}} \mathrm{Ker}\ \pi \xrightarrow{\mathcal{L}} p^e\mathbf{Z}_p \xrightarrow{\mathrm{mod}\, p^{2e}} p^e\mathbf{Z}_p/p^{2e}\mathbf{Z}_p \cong \mathbf{Z}/p^e\mathbf{Z},$$

where π is an inverse map of u and h_{p^e} is multiplication of p^e. Concerning the map $\mathcal{L}$, see [58, p84].

We can finish the proof of Lemma 5.1 by proving that (1) λ'_{p^e} is an isomorphism and (2) it holds that $\lambda_{p^e}(R) = \lambda'_{p^e}(R)$ for any $R \in \tilde{E}(\mathbf{Z}/p^e\mathbf{Z})$. By a similar discussion of [58, p.85], (1) can easily be proven. Moreover, by a similar discussion of [58, pp.86-87], (2) can also be proven. ■

Proof of Lemma 5.2.

Let $\theta^{(1)}_{(p^e,j)}$ and $\theta^{(2)}_{(p^e,j)}$ be compositions of the following maps,

$$\theta^{(1)}_{(p^e,j)} : \tilde{E}(\mathbf{Z}/p^e\mathbf{Z}) \xrightarrow{h_j} \tilde{E}(\mathbf{Z}/p^e\mathbf{Z}) \xrightarrow{u} E(\mathbf{Q}_p) \xrightarrow{h_{p^e}} \mathrm{Ker}\ \pi \xrightarrow{\mathcal{L}} p^e\mathbf{Z}_p \xrightarrow{\mathrm{mod}\, p^{2e}} p^e\mathbf{Z}_p/p^{2e}\mathbf{Z}_p \cong \mathbf{Z}/p^e\mathbf{Z},$$

$$\theta^{(2)}_{(p^e,j)} : \tilde{E}(\mathbf{Z}/p^e\mathbf{Z}) \xrightarrow{u} E(\mathbf{Q}_p) \xrightarrow{h_j} E(\mathbf{Q}_p) \xrightarrow{h_{p^e}} \mathrm{Ker}\ \pi \xrightarrow{\mathcal{L}} p^e\mathbf{Z}_p \xrightarrow{\mathrm{mod}\, p^{2e}} p^e\mathbf{Z}_p/p^{2e}\mathbf{Z}_p \cong \mathbf{Z}/p^e\mathbf{Z}.$$

The difference of these two maps is the order of u and h_j.

Let $A \equiv u(R) \in E(\mathbf{Z}_p)$ for $R \in \tilde{E}(\mathbf{Z}/p^e\mathbf{Z})$. A point jA (or, $h_j(A)$) is in $E(\mathbf{Q}_p)$ and becomes the lifting point of jR, because π is a homomorphism and $\pi(jA) = j\pi(A) = jR$. Since both jA and $u(jR)$ are the lifting points of jR, we obtain

$$\theta^{(1)}_{(p^e,j)}(R) = \theta^{(2)}_{(p^e,j)}(R)$$

from Remark 5.8. Moreover, we obtain $\theta^{(1)}_{(p^e,j)}(R) = \lambda_{p^e}(jR) = j\lambda_{p^e}(R)$. We can finish the proof of Lemma 5.2 by proving that it holds that $\theta^{(2)}_{(p^e,j)}(R) = \theta_{(p^e,j)}(R)$ for any $R \in \tilde{E}(\mathbf{Z}/p^e\mathbf{Z})$. By a similar discussion of [58, pp.86-87], this can be easily proven. ■

Proof of Lemma 5.3.

For simplicity, we set $e_i = 1$ for all i. Denote $R = (s,t) \in \tilde{E}(\mathbf{Z}/n\mathbf{Z})$, where $s,t \in \mathbf{Z}/n\mathbf{Z}$. Let $A^{(n)} = (X_1^{(n)}, Y_1^{(n)}) \equiv (s_1^{(n)} + v_1^{(n)}n,\ t_1^{(n)} + w_1^{(n)}n)$ be the lifting points of R to $E(\mathbf{Z}/n^2\mathbf{Z})$, where $s_1^{(n)}, v_1^{(n)}, t_1^{(n)}, w_1^{(n)} \in \mathbf{Z}/n\mathbf{Z}$. In addition, let $A^{(i)} \equiv (s_1^{(i)} + v_1^{(i)}p_i,\ t_1^{(i)} + w_1^{(i)}p_i)$ be the lifting point of R to $E(\mathbf{Z}/{p_i}^2\mathbf{Z})$, where $s_1^{(i)}, v_1^{(i)}, t_1^{(i)}, w_1^{(i)} \in \mathbf{F}_{p_i}$.

By multiplying $A^{(n)}$ and $A^{(i)}$ by $n-1$ over $E(\mathbf{Z}/n^2\mathbf{Z})$ and $E(\mathbf{Z}/{p_i}^2\mathbf{Z})$, respectively, we denote

$$(X_{n-1}^{(n)},\ Y_{n-1}^{(n)}) = (n-1)A^{(n)} = (s_{n-1}^{(n)} + v_{n-1}^{(n)}n, t_{n-1}^{(n)} + w_{n-1}^{(n)}n),$$

where $s_{n-1}^{(n)}, v_{n-1}^{(n)}, t_{n-1}^{(n)}, w_{n-1}^{(n)} \in \mathbf{Z}/n\mathbf{Z}$, and denote $(n-1)A^{(i)} = (s_{n-1}^{(i)} + v_{n-1}^{(i)}p_i,\ t_{n-1}^{(i)} + w_{n-1}^{(i)}p_i)$,where $s_{n-1}^{(i)}, v_{n-1}^{(i)}, t_{n-1}^{(i)}, w_{n-1}^{(i)} \in \mathbf{F}_{p_i}$. From the relation between $A^{(n)}$ and $A^{(i)}$, we have $s_1^{(n)} = s$, $v_1^{(n)} = 0$, $t_1^{(n)} = t$, $w_1^{(n)} = (s^3 + as + b - t^2)/(2tn)$, $s_1^{(i)} = s_1^{(n)} \bmod p_i$, $v_1^{(i)} = (s_1^{(n)} - s_1^{(i)})/p_i \bmod p_i$, and $t_1^{(i)} = t_1^{(n)} \bmod p_i$. Since $\tilde{E}$ and $\widetilde{E_i}$ are super-anomalous and anomalous, we obtain $s_{n-1}^{(n)} = s_1^{(n)}$ and $t_{n-1}^{(n)} + t_1^{(n)} \equiv 0 \pmod n$ and $s_{n-1}^{(i)} = s_1^{(i)}$ and $t_{n-1}^{(i)} + t_1^{(i)} \equiv 0 \pmod{p_i}$. Furthermore, we obtain

$$(s_{n-1}^{(n)} + v_{n-1}^{(n)}p_i) \bmod {p_i}^2 = s_{n-1}^{(i)} + v_{n-1}^{(i)}p_i. \tag{A.1}$$

By definition, we obtain

$$\begin{aligned}
\lambda_n(R) &= -\frac{X_{n-1}^{(n)} - X_1^{(n)}}{n}/(Y_{n-1}^{(n)} - Y_1^{(n)}) \bmod n \\
&= -\frac{(s_{n-1}^{(n)} + v_{n-1}^{(n)}n) - (s_1^{(n)} + v_1^{(n)}n)}{n}/(t_{n-1}^{(n)} - t_1^{(n)}) \\
&= \frac{v_{n-1}^{(n)}}{2t_1^{(n)}} \bmod n
\end{aligned}$$

and also

$$\theta_{(p_i, n/p_i)}(R) = (v_{n-1}^{(i)} - v_1^{(n)})/(2t_1^{(i)}) \bmod p_i.$$

similarly. Note that $p_i \cdot n/p_i = n$. From equation (A.1), we obtain

$$(v_{n-1}^{(n)} - v_{n-1}^{(i)})p_i \equiv -(s_{n-1}^{(n)} - s_{n-1}^{(i)}) \equiv -(s_1^{(n)} - s_1^{(i)}) \equiv -v_1^{(i)}p_i \pmod{{p_i}^2}.$$

Therefore, it suffices that $v_{n-1}^{(n)} - v_{n-1}^{(i)} \equiv -v_1^{(i)} \pmod{p_i}$. Hence, we have

$$\lambda_n(R) \bmod p_i = \frac{v_{n-1}^{(n)} \bmod p_i}{2t_1^{(n)} \bmod p_i} \bmod p_i = \frac{v_{n-1}^{(i)} - v_1^{(i)}}{2t_1^{(i)}} \bmod p_i = \theta_{(p_i, n/p_i)}(R).$$

Therefore, Lemma 5.3 has been proven. ■

Proof of Theorem 5.2.

From Eq. (5.1), $\lambda_n(R) \equiv C'\lambda_{{p_i}^{e_i}}(R) \pmod{{p_i}^{e_i}}$ holds. Hence, if all of $\lambda_{{p_i}^{e_i}}$ are non-zero maps, the image of λ_n is $(\mathbf{Z}/n\mathbf{Z})^*$. Second, suppose that $\lambda_{{p_1}^{e_1}}, \lambda_{{p_2}^{e_2}}, \ldots, \lambda_{{p_l}^{e_l}}$ are zero maps and the others $\lambda_{{p_{l+1}}^{e_{l+1}}}, \ldots, \lambda_{{p_k}^{e_k}}$ are non-zero maps, i.e.isomorphism. Letting n' be $\prod_{i=1}^{l} {p_i}^{e_i}$, $\lambda_n \equiv 0 \pmod{n'}$. Therefore, the image of λ_n is $n'(\mathbf{Z}/n\mathbf{Z})^*$. Next, suppose that all of $\lambda_{{p_i}^{e_i}}$ are zero maps. In this case, λ_n is a zero map. ■

Proof of Theorem 5.3.

Let $E_p : y^2 \equiv x^3 + ax + b \pmod{p^2}$ and $E_p' : y^2 \equiv x^3 + (a+p)x + b \pmod{p^2}$, where the reduction of E_p and E_p' are assumed to be anomalous. In [59], it is proved that either λ_p for E_p or λ_p for E_p' is a non-zero map. Furthermore, it is proved that if $\gcd(p, j_1) = \gcd(p, j_2) = 1$ and $(j_1 \neq j_2)$, then either λ_p for $E^{(j_1)}$ or λ_p for $E^{(j_2)}$ is a non-zero map, where $E_p^{(j)} : y^2 \equiv x^3 + (a + jp)x + b \pmod{p^2}$. This general property can be proven by a similar discussion in [59]. Therefore, at most one λ_p for $E_p^{(j)}$ $(0 \leq j \leq p-1)$ is a zero map. Let $n = \prod_{i=1}^{k} p_i^{e_i}$ and $E^{(j)} : y^2 \equiv x^3 + (a + jn)x + b \pmod{n^2}$. At most one of $\lambda_{p_i^{e_i}}$ for $E^{(j)}$ $(0 \leq j \leq p_i - 1)$ is a zero map for each p_i $(1 \leq i \leq k)$. Hence, there exists $E^{(j)}$ in $(0 \leq j \leq k)$ such that all of $\lambda_{p_i^{e_i}}$ for $E^{(j)}$ are non-zero maps. In this case, λ_n for $E^{(j)}$ is an isomorphism. ■

Proof of Theorem 5.4.

Letting $R_1, R_2 \in \tilde{E}(\mathbf{Z}/n\mathbf{Z})$, we denote $R_l = \langle R_{l1}, R_{l2}, \cdots, R_{lk} \rangle$ (or in short, $\langle R_{li} \rangle_{i=1}^{k}$) for $l = 1, 2$, where R_{1i} and R_{2i} are the points over $\tilde{E}(\mathbf{F}_{p_i})$. From the definition,

$$\Theta(R_1, R_2) \bmod p_i = \Theta(R_{1i}, R_{2i})$$

for all i. Furthermore, we denote $P = \langle P_i \rangle_{i=1}^{k}$ and $\Gamma^{(j)}(P)$ be the second element of $j \otimes (P, 0)$ over $\mathbf{Z}/n\mathbf{Z}$ and $\gamma^{(j)}(P_i)$ be the second element of $j \otimes (P_i, 0)$ over $\mathbf{F}_{p_i}$. From the above property of $\Theta(R_1, R_2)$, it clearly satisfies that $\gamma^{(j)}(P_i) = \Gamma^{(j)}(P) \bmod p_i$ for all j and i. Hence, in case of $j = n$, $\gamma^{(n)}(P_i) = \Gamma^{(n)}(P) \bmod p_i = \Lambda_n(P) \bmod p_i$. It also satisfies that

$$\gamma^{(n)}(P_i) = (n/p_i)\Lambda_{p_i}(P_i) \bmod p_i.$$

From the above $2i$ equations and the Chinese remainder theorem, it holds that $\Lambda_n(P) \equiv \sum_{i=1}^{k} D_i \Lambda_{p_i}(P_i) \pmod{n}$, where all D_i are the constants. Since Λ_{p_i} is an isomorphism, the map Λ_n is also an isomorphism. ■

Appendix B

Another algorithm for solving DLP over SAEC

In Section 5.5, we mentioned that Okamoto and Uchiyama proposed an algorithm able to factorize the composite when a super-anomalous elliptic curve is given as input [48]. This algorithm can factorize the composite in polynomial time with probability 1/2. In this chapter, we extend their algorithm. This extended algorithm runs in deterministic polynomial time on the assumption that the extended Riemann Hypothesis is true.

Theorem B.1 Assume that the Extended Riemann Hypothesis is true. Given a super-anomalous elliptic curve $E(\mathbf{Z}/n\mathbf{Z}) : y^2 \equiv x^3 + ax + b \pmod{n}$, where n is a product of two distinct primes, we can factorize n in deterministic polynomial time.

Proof of Theorem B.1. The following algorithm can factorize the composite in the deterministic polynomial time.

Input Super-anomalous elliptic curve $E(\mathbf{Z}/n\mathbf{Z}) : y^2 \equiv x^3 + ax + b \pmod{n}$

Output p_1 and p_2, which are prime factors of n.

Step 1 Choose a random integer $x_0 \in \mathbf{Z}/n\mathbf{Z}$. By setting y_0 adequately, the point $G = (x_0, y_0)$ lies on one of the four curves described in Note B.1. In particular, if $(x_0^3 + ax_0 + b/p_1) = +1$ and $(x_0^3 + ax_0 + b/p_2) = +1$, G lies on $E(\mathbf{Z}/n\mathbf{Z})$.

Step 2 Compute nG over $E(\mathbf{Z}/n\mathbf{Z})$ by using an addition formula described in the x-coordinate only. We need not care whether $G \in E(\mathbf{Z}/n\mathbf{Z})$ or not.

Step 3 If nG is a point at semi-infinity, we can factorize n. Otherwise, go to Step 4.

Step 4 Find a positive integer c so that $(c/n) = -1$. Set the elliptic curve $E'(\mathbf{Z}/n\mathbf{Z}) : y^2 \equiv x^3 + ac^2x + bc^3 \pmod{n}$.

Step 5 Compute nG over $E'(\mathbf{Z}/n\mathbf{Z})$ formally as in Step 2. In this case, nG always becomes a point at semi-infinity. Hence, we can factorize n.

Note B.1 By setting y_0 adequately, the point G lies on one of the following four curves. Let $(x_0^3 + ax_0 + b/p_1) = l_1$ and $(x_0^3 + ax_0 + b/p_2) = l_2$. If $l_1 = l_2 = +1$, $G \in E(\mathbf{F}_{p_1}) \oplus E(\mathbf{F}_{p_2})(= E(\mathbf{Z}/n\mathbf{Z}))$. If $l_1 = +1, l_2 = -1$, $G \in E(\mathbf{F}_{p_1}) \oplus E_t(\mathbf{F}_{p_2})$. If $l_1 = -1, l_2 = +1$, $G \in E_t(\mathbf{F}_{p_1}) \oplus E(\mathbf{F}_{p_2})$. If $l_1 = l_2 = -1$, $G \in E_t(\mathbf{F}_{p_1}) \oplus E_t(\mathbf{F}_{p_2})$, where E_t is a twist of E.

If $G \in E(\mathbf{F}_{p_1}) \oplus E_t(\mathbf{F}_{p_2})$ or $G \in E_t(\mathbf{F}_{p_1}) \oplus E(\mathbf{F}_{p_2})$, nG becomes $\langle \mathcal{O}_{p_1}, nG_2(\neq \mathcal{O}_{p_2})\rangle$ or $\langle nG_1(\neq \mathcal{O}_{p_1}), nG_2\rangle$, respectively. This is because $\#E_t(\mathbf{F}_{p_1}) = p_1 + 2, \#E_t(\mathbf{F}_{p_2}) = p_2 + 2$. Hence, if G lies on the above two curves, we succeed in factorizing n in Step 3. Conversely, if $G \in E(\mathbf{F}_{p_1}) \oplus E(\mathbf{F}_{p_2})$ or $G \in E_t(\mathbf{F}_{p_1}) \oplus E_t(\mathbf{F}_{p_2})$, we fail in factorizing n in Step 3.

In Step 4, suppose that such a c is selected. For simplicity, we assume that $(c/p_1) = +1$ and $(c/p_2) = -1$. If $G \in E(\mathbf{F}_{p_1}) \oplus E(\mathbf{F}_{p_2})$ then $p_1 | \#E'(\mathbf{F}_{p_1})$ and $p_2 \not| \#E'(\mathbf{F}_{p_2})(= p + 2)$. Hence, nG becomes a point as semi-infinity over $E'(\mathbf{Z}/n\mathbf{Z})$ and we can succeed in factorizing n in Step 5. If $G \in E_t(\mathbf{F}_{p_1}) \oplus E_t(\mathbf{F}_{p_2})$ then we can also succeed in factorizing n in Step 5. The same is true if $(c/p_1) = -1, (c/p_2) = +1$.

In Step 4, we can deterministically find c such that $(c/n) = -1$ in polynomial time. This is led from the following lemma.

Lemma B.1 Assume that the Extended Riemann Hypothesis is true. The minimum number c so that $(c/n) = -1$ is $O((\log n)^2)$.

Hence, we can find c by $O((\log n)^2)$ times computing (c/n) in the worst case.

From the above discussion, this algorithm factorizes the composite n in deterministic polynomial time. ■

Appendix C

How to solve special simultaneous congruences

We can solve the simultaneous congruences:

$$\begin{cases} R(x,y) & \equiv 0 \pmod{n} \\ x^e - C & \equiv 0 \pmod{n} \\ y^e - C' & \equiv 0 \pmod{n} \end{cases}$$

by using the following method if there exists only one solution (m, m') and e and the degree of $H(x,y)$ in x and y are polynomials of $\log n$.

The polynomial $R(x,y)$ can be rewritten into

$$R(x,y) = k_0(y)x^i + k_1(y)x^{i-1} + \cdots + k_i(y), \tag{C.1}$$

where $k_j(y)$, for $0 \leq j \leq i$, are polynomial in y. First, we eliminate a variable x from $R(x,y) \equiv 0 \pmod{n}$ and $x^e - C \equiv 0 \pmod{n}$. This can be done by calculating the resultant of $R(x,y)$ and $x^e - C$. The resultant Resultant$(R(x,y), x^e - C)$ is given by computing the determinant of the following matrix. This is called Sylvester's elimination method. For convenience, we write k_i instead of $k_i(y)$.

$$\det \begin{pmatrix} k_0(y) & k_1(y) & \cdots & k_i(y) & 0 & \cdots & \cdots & 0 \\ 0 & k_0(y) & k_1(y) & \cdots & k_i(y) & 0 & \cdots & 0 \\ 0 & \cdots & \cdots & 0 & k_0(y) & k_1(y) & \cdots & k_i(y) \\ 1 & 0 & \cdots & 0 & -C & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 & -C & 0 & \cdots \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & -C \end{pmatrix}$$

This Resultant$((R(x,y), x^e - C)(\equiv R'(y))$ is a univariate polynomial in y. By computing $\gcd(R'(y), y^e - C')$, we obtain

$$\gcd(R'(y), y^e - C') = C''(y - m') \bmod n. \tag{C.2}$$

Note that the degree of the right hand side of Eq. (C.2) is 1 since there exists only one solution. Thus, we can get $y = m'$. By computing $\gcd(R(x, m'), x^e - C)$, we also obtain $x = m$.

Example C.1

$$\begin{cases} k_0(y)x^2 + k_1(y)x + k_0(y) & \equiv 0 \pmod{323} \\ x^5 - 106 & \equiv 0 \pmod{323} \\ y^5 - 277 & \equiv 0 \pmod{323}, \end{cases}$$

where $k_0(y) = 267y$ and $k_1(y) = 60y^3 + 143y^2 + 292y + 263$.

Resultant$(k_0(y)x^2 + k_1(y)x + k_0(y), x^5 - 106) =$

$$\det \begin{pmatrix} k_0(y) & k_1(y) & k_0(y) & 0 & 0 & 0 & 0 \\ 0 & k_0(y) & k_1(y) & k_0(y) & 0 & 0 & 0 \\ 0 & 0 & k_0(y) & k_1(y) & k_0(y) & 0 & 0 \\ 0 & 0 & 0 & k_0(y) & k_1(y) & k_0(y) & 0 \\ 0 & 0 & 0 & 0 & k_0(y) & k_1(y) & k_0(y) \\ 1 & 0 & 0 & 0 & 0 & -106 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -106 \end{pmatrix}$$

$= 32y^{15} + 166y^{14} + 321y^{13} + 267y^{12} + 105y^{11} + 262y^{10} + 319y^9 + 52y^8 + 101y^7 + 54y^6 + 118y^5 + 293y^4 + 251y^3 + 141y^2 + 25y + 291 \bmod 323 (\equiv R'(y))$. By computing $\gcd(R'(y), y^5 - 277) = (y - 292)$, we obtain $y = 292$. By computing $\gcd(R(x, 292), x^5 - 106) = (x - 140)$, we obtain $x = 140$.

Note C.1 If $R(x, y) = k_0(y)x^2 + k_1(y)x + k_2(y)$,

$$\begin{aligned} R'(y) &\equiv \text{Resultant}(R(x, y), x^e - C) \\ &= C^2 k_0(y)^e + C k_1(y)^e + k_2(y)^e - eC k_0(y) k_1(y) k_2(y) (k_1(y)^2 - k_0(y) k_2(y))^{(e-3)/2}. \end{aligned}$$