Survey of Computational Assumptions Used in Cryptography Broken or Not by Shor's Algorithm

by

Hong Zhu
School of Computer Science
McGill University
Montréal, Canada

December, 2001

A Thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master in Science

Abstract

We survey the computational assumptions of various cryptographic schemes, and discuss the security threat posed by Shor's quantum algorithm.

One-way functions form the the basis of public-key cryptography. Although we have candidate hard problems that are believed to be one-way, none has been proven to be so. Therefore the security of the corresponding cryptographic schemes depends on the the intractability assumptions of these problems. Two major species of such problems, factoring and discrete logarithm, are widely believed to be intractable, and serve as the basis of many popular schemes. However, these two problems turned out to be polynomial-time solvable on a hypothetical quantum computer using Shor's algorithm. This is the most worrisome long-term threat to current public-key cryptosystems.

In the thesis we provide a review of existing cryptosystems, with a focus on their underlying computational assumptions and the security. Other than factoring and discrete logarithm, schemes have been proposed based on error-correcting codes, subset-sum and subset-product problems, lattice, polynomials, combinatorial group theory, number fields, etc. Many are to be furtherly evaluated in future research.

Résumé

Nous faisons un survol des hypothèses calculatoires de plusieurs schémas cryptographiques, et nous discutons de la menace posée par l'algorithme quantique de Shor.

Les fonctions à sens unique forment la base des systèmes à clés publiques en cryptographie. Malgré que nous ayons sélectionné des problèmes difficiles et qui semblent à sens unique, personne n'a prouvé qu'ils le sont. Donc, la sécurité des schémas cryptographiques dépend d'hypothèses sur la complexité de ces mêmes problèmes. La factorisation et les logarithmes discrets, deux problèmes de ce type, sont largement acceptés comme étant intraitables et servent de base à ces schémas. Cependant, ces deux problèmes peuvent être résolus en un temps polynomial à l'aide d'un hypothétique ordinateur quantique en utilisant l'algorithme de Shor. À long terme, cela représente une grande menace aux systèmes cryptographiques à clés publiques.

Dans cette thèse, nous présentons un survol des systèmes cryptographiques existants, en insistant sur leur côté calculatoire et leur sécurité. À part la factorisation et les logarithmes discrets, des schémas ont été proposés en se basant sur la théorie des codes, la somme et le produit de sous-ensembles, la théorie des treillis, la théorie des polynômes, la théorie des groupes combinatoires et d'autres théories. Plusieurs problèmes devront être évalués dans de futures recherches.

Acknowledgments

First I wish to express my gratitude to my thesis supervisor, Claude Crépeau, for his stimulating guidance and constant encouragement during this thesis work. Among the many things, I'm especially impressed by his commitment to high standards and his very friendly way of relating to people. It has been a pleasure for me to work with such a nice person. I am immensely thankful to Paul Dumais for his generous help. His good-humored criticism has been an indispensable source of motivation. I highly appreciate the time and efforts that Claude and Paul put in proofreading and revising my thesis manuscript.

I also wish to thank members of the Crypto & Quantum Info Lab, Geneviève, Hugo, Simon-Pierre, Alex, Martin, and Thanh Vinh, for always being so nice and helpful. Special thanks to our system administrator, Andrew, for his time and patience. Also thanks to Lise, Teresa, Vera, and Lucy for their considerable assistance.

Finally I would like to thank my fellow students of the School of Computer Science, and the many friends at McGill. Thanks to Wen and Samuel for being sunshine to my life.

Contents

List of Tables

Chapter 1

Introduction

Cryptography refers to a wide range of security issues in the storage, transmission and protection of information such as massive file storage, electronic commerce through public networks, the use of smart cards, etc.

Three of the most important services provided by cryptosystems are secrecy, authenticity, and integrity. *Secrecy* refers to keeping information secret from all but those who are authorized to see it. *Authenticity* refers to validating the source of a message; i.e., that it was transmitted by a properly identified sender. *Integrity* refers to assurance that a message was not modified accidentally or deliberately in transit, by replacement, insertion or deletion. Traditional cryptography deals mainly with the secrecy aspect.

A cryptosystem for message transmission means a map from ordinary text (plaintext) to encrypted form (ciphertext). The idea of using arithmetic operations to construct such a map goes back to the days of Roman Empire. Until the late 1970's, encryption schemes were based on the sender and receiver of a message knowing and using the same secret key. In such a cryptosystem (known as *symmetric-key* cryptosystem) two users who want to communicate secretly must exchange keys in a safe way.

The course of cryptography was totally altered when Diffie and Hellman introduced the concept of *public-key* cryptosystem in 1976. The idea behind public-key cryptography is fairly simple: Anyone can put something in a box and close the lock, but only the person who knows the lock combination can open the box again. In a public-key cryptosystem, each person gets a pair of keys, one called the public key and the other called the private key. Each person's public key is published while the private key is kept secret. Suppose Alice has a public key, which she publishes. Then anyone can encrypt a message to send her. And everyone uses the same method of encryption using her public key. But only Alice knows the private key which allows her to invert the process and decrypt the message.

At the heart of this concept is the idea of using a one-to-one *one-way function* for encryption. Speaking roughly, a function f from X to Y is "one-way" if it is easy to compute $f(x)$ for any $x \in X$ but very hard on average to compute x from the value of $f(x)$. The functions used for encryption belong to a special class of one-way functions called *trapdoor* one-way functions. A trapdoor one-way function is a one-way function where the inverse direction is easy, given a certain piece of information (the trapdoor), but difficult otherwise. This trapdoor serves as the decryption key. A trapdoor one-way function remains one-way only if the decryption key is kept secret.

All practical public-key cryptosystems are based on functions that are believed to be one-way, but no function has been proven to be so. The existence of polynomial-time computable one-way functions is still an open question.

This means that it is theoretically possible that an algorithm will be discovered that can compute the inverse direction easily without a trapdoor. This development would render any cryptosystem based on that one-way function insecure and useless.

Two candidate one-way functions of importance in cryptography today are integer factorization and discrete logarithm. The former has given rise to the RSA cryptosystem and the latter to the discrete logarithm based systems.

The RSA cryptosystem, invented by Rivest, Shamir and Adleman in 1978, is the most popular public-key cryptosystem. This system is based on the fact that multiplication and primality testing are easy but prime factorization is much harder. So far it has resisted all kinds of attacks [MvOV96]. The difficulty of discrete logarithm problem is the foundation of several public-key cryptosystems, including the ElGamal public-key cryptosystem. The discrete logarithm problem bears the same relation to these systems as factoring does to RSA. More recently, the fact that every elliptic curve defined over a finite field has a group structure is used in constructing of elliptic curve cryptosystems.

Factoring algorithms have been studied for hundreds of years, general discrete logarithm algorithms have been extensively studied since the early 1970s, and elliptic curve discrete logarithms have been studied since the mid-1980s. It is impossible to predict when a mathematical breakthrough might occur.

It is an unfortunate fact that discrete logarithms and integer factorization are so close that many algorithms developed for one problem can be modified to apply to the other. For security, it would be better to have much more diversity. However, the many attempts to find public-key schemes based on other principles have been less than successful – most have been broken, and the rest are either unpractical or still under investigation.

The most worrisome long-term threat to RSA and discrete logarithm cryptosystems comes from quantum computers.

Quantum computers use the dynamics of atomic-scale objects to store and manipulate information. The behavior of atomic-scale objects is governed by quantum mechanics rather than by classical physics. The state of a quantum computer is a superposition of exponentially many basis states, each of which corresponds to a state of a classical computer of the same size. By taking advantage of interference and entanglement in this system, a quantum computer could naturally perform a myriad of operations in parallel (known as quantum parallelism). As a result, significant speedup is possible for certain problems using appropriate quantum algorithms.

In 1994, Peter Shor [Sho97] of AT&T Laboratories showed that if such machines were built, integer factorization and discrete logarithms (including elliptic curve discrete logarithms, as shown by [BL95]) could be computed in polynomial time.

The implications of Shor's factoring algorithm on the world of cryptography is staggering. The integer factorization and discrete logarithms problems are generally believed to be intractable for classical algorithms, and most schemes are based on this assumption. The ability to break the RSA and discrete logarithms based systems will render almost all current channels of communication insecure.

Shor's discovery stimulated great interests and an explosion in research on quantum computers. Experimentalists try to build quantum computers and theorists try to find other quantum algorithms. Quantum computing suddenly came into a dynamic and rapidly developing field.

While there is still some debate on whether quantum computers are feasible, no fundamental obstructions to their constructions have been found, and novel approaches are regularly suggested. The one comforting factor is that all experts agree that a lot of ground needs to be broken before the first quantum computer can be built. It will likely take many years to do so, at least for machines on a scale that will threaten modern public-key systems.

It is likely that quantum computing will be the next revolution in computer science. Because of the threat that Shor's algorithm poses to existing encryption techniques, there is also a great deal of interest in developing alternate public-key cryptosystems. A few candidate hard problems are:

- Error-correcting codes
- Subset-sum (knapsack)
- Subset-product
- Lattice
- Polynomials
- Combinatorial group theory
- Number fields

The remainder of this thesis is organized as follows. Chapter 2 contains a brief covering of the relevant mathematics used in this thesis. Chapter 3 introduces the basics of cryptography. Chapter 4 and 5 discuss the two important problems, factoring and discrete logarithm, as well as some major practical cryptosystems based on them. Chapter 6 explains the famous Shor's quantum algorithm and how it solves factoring and discrete logarithm problems. We thus show the great impact of Shor's algorithm on cryptography and quantum computing. The rest of the thesis surveys computational assumptions that survive Shor's attack, i.e., assumptions other than factoring and discrete logarithm. The list includes error-correcting codes, subset-sum, lattices, polynomials, braid groups, subset product, number fields, etc. The list here is not assumed to be exhaustive. We aim to provide an up-to-date view of the research directions in cryptography as facing the long-term threat posed by Shor's algorithm.

Chapter 2

Mathematical Background

2.1 Complexity theory

Computational complexity provides a foundation for analyzing cryptographic techniques. Complexity theory classifies a problem according to the minimum time and space needed to solve the hardest instances of the problem on a Turing Machine (or some other abstract model of computation). If a problem is polynomial solvable on a Turing Machine (TM), then it is polynomial solvable on a real system and vice versa.

2.1 Definition [1] An *algorithm* is a well-defined computational procedure that takes a variable input and halts with an output.

2.2 Definition The *running time* of an algorithm on a particular input is the number of primitive operations or "steps" executed.
Often a step is taken to mean a bit operation. For some algorithm it will be more convenient to take step to mean something else such as a comparison, a machine instruction, a machine clock cycle, a modular multiplication, etc.

The following definitions involves asymptotic notations O and o. Readers not familiar with them can refer to [MvOV96]. Intuitively, $f(n) \in O(g(n))$ means that f

[1]Unless otherwise indicated, the definitions in this chapter are based on [MvOV96].

grows no faster asymptotically than $g(n)$ within a constant multiple. $f(n) \in o(g(n))$ means that $g(n)$ is an upper bound for $f(n)$ that is not asymptotically tight, i.e., $f(n)$ becomes insignificant relative to $g(n)$ as n gets larger.

The following two notations are borrowed from Paul Dumais [Dum99, page 5].

2.3 Notation A *polynomial-time algorithm* is an algorithm whose worst-case running time function is of the form

$$poly(n) = \bigcup_{k \geq 1} O(n^k),$$

where n is the input size.

2.4 Notation A *superpolynomial-time algorithm*[2] is an algorithm whose worst-case running time function is of the form

$$superpoly(n) = \bigcap_{k \geq 1} \Omega(n^k),$$

where n is the input size.

2.5 Example (superpolynomial running time) Let A be an algorithm whose inputs are either elements of a finite field $\mathbb{F}_q$, or an integer q. If the expected running time of A is of the form

$$L_q[\alpha, c] = O(e^{((c+o(1))(\ln q)^{\alpha}+(\ln \ln q)^{1-\alpha}}),$$

where c is a positive constant, and α is a constant satisfying $0 < \alpha < 1$, then A is a superpolynomial-time algorithm. Observe that for $\alpha = 0$, $L_q[0, c]$ is a polynomial in $\ln q$, while for $\alpha = 1$, $L_q[1, c]$ is a polynomial in q, and thus fully exponential in $\ln q$.

2.6 Definition An algorithm whose running time is given by $O(k^{h(n)})$ for constant $k > 1$ and polynomial $h(n)$ is called an *exponential-time algorithm* [Den82].

[2] The term "superpolynomial" is often exchangeable with "subexponential" when describing the complexity class that is asymptotically faster than exponential while asymptotically slower than polynomial. In this thesis, we stick to the term "superpolynomial".

Informally speaking, problems that are solvable in polynomial time are called *tractable* because they can usually be solved for reasonable size inputs. Problems that cannot be systematically solved in polynomial time are called *intractable* or simply "hard", because as the size of the input increases, their solution becomes infeasible on even the fastest computers.

Complexity theory restricts its attention to decision problems, i.e., problems which have either YES or NO as an answer. In practice, computational problems can be phrased as decision problems, so that an efficient algorithm for the decision problem yields an efficient algorithm for the computational problem, and vice versa.

2.7 Definition Let L_1 and L_2 be two decision problems. L_1 is said to *polytime reduce* to L_2, written $L_1 \leq_P L_2$, if there is an algorithm that solves L_1 which uses, as a subroutine, an algorithm for solving L_2, and which runs in polynomial time if the algorithm for L_2 does.

Informally, if $L_1 \leq_P L_2$, then L_2 is at least as difficult as L_1, or, equivalently, L_1 is no harder than L_2. Consequently, if L_1 is widely believed to be intractable, then proving that $L_1 \leq_P L_2$ provides strong evidence of the intractability of L_2.

2.8 Definition Let L_1 and L_2 be two decision problems. If $L_1 \leq_P L_2$ and $L_2 \leq_P L_1$, then L_1 and L_2 are said to be *computationally equivalent*, written $L_1 \equiv_P L_2$.

2.9 Definition The complexity class **P** is the set of all decision problems that are solvable in polynomial time.

2.10 Definition The complexity class **NP** is the set of all decision problems for which a YES answer can be verified in polynomial time using some extra information, called a *certificate*.

The class **P** consists of all problems solvable in polynomial time. The class **NP** (nondeterministic polynomial) consists of all problems solvable in polynomial time on a nondeterministic TM. This means if the machine guesses the solution, it can check its correctness in polynomial time. Apparently, this does not really "solve" the problem, because there is no guarantee the machine will guess the correct answer, and a nondeterministic TM is not a realistic model of computation. However, all problems in **NP** have this nice property that an (instance, solution) pair can be verified efficiently. These (instance, solution) pairs can be built efficiently for some problems. Put simply, most of the interesting problems that currently cannot be solved in polynomial time are in **NP**.

We know that all problems in **P** are also in **NP**, but we do not know whether or not all problems in **NP** are in **P**. The class **NP** includes the class **P** because any problem polynomial solvable on a deterministic TM is polynomial solvable on a nondeterministic one. Although many problems in **NP** seem much "harder" than the problems in **P**, no one has yet proved that $\mathbf{P} \neq \mathbf{NP}$.

NP-complete problems is the set of equivalent problems in **NP** such that if any one of the problems is in **P**, then all **NP** problems are in **P** and $\mathbf{P} = \mathbf{NP}$. Therefore the **NP-complete** problems are the "hardest" problems in **NP**. The fastest known algorithms for systematically solving these problems are superpolynomial-time algorithms.

2.2 Number theory

The set of integers is denoted by $\mathbb{Z}$.

2.11 Definition (Division algorithm for integers)
Let $a, b \in \mathbb{Z}, b \geq 1$, then there exist unique $q,\ r \in \mathbb{Z}$ such that

$$a = q\,b + r,\ 0 \leq r < b$$

q is called the *quotient*, denoted by a div b; and r (the *remainder*) denoted by a mod b. If $r = 0$, we say b *divides* a, and denote this by $b|a$.

2.12 Definition A non-negative integer d is the *greatest common divisor* of integers a and b, denoted by $d = \gcd(a, b)$ if

1. $d|a$ and $d|b$; and

2. whenever $c|a$ and $c|b$, then $c|d$.

2.13 Definition for $n \geq 1$, 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.*

2.14 Fact

1. If p is a prime, then $\phi(p^e) = (p-1)p^{e-1}$.

2. If $gcd(m, n) = 1$, then $\phi(mn) = \phi(m) \cdot \phi(n)$.

3. If $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$, then $\phi(n) = (p_1 - 1)p_1^{e_1-1}(p_2 - 1)p_2^{e_2-1} \dots (p_k - 1)p_k^{e_k-1}$.

Let n be a positive integer.

2.15 Definition If a and b are integers, then a is said to be *congruent to b modulo n*, written $a \equiv b \pmod{n}$, if n divides $(a - b)$.

2.16 Definition The *integers modulo n*, denoted $\mathbb{Z}_n$, is the set of (equivalence classes of) integers $\{0, 1, 2, \dots, n-1\}$. Addition, subtraction, and multiplication in $\mathbb{Z}_n$ are performed modulo n.

2.17 Definition Let $a \in \mathbb{Z}_n$. The *multiplicative inverse* of a modulo n is an integer $x \in \mathbb{Z}_n$ such that $ax \equiv 1 \pmod{n}$. If such an x exist, then it is unique, and a is said to be *invertible*; the inverse of a is denoted by a^{-1}.

2.18 Fact Let $a \in \mathbb{Z}_n$. Then a is invertible if and only if $gcd(a, n) = 1$. We can compute a^{-1} using the *extended Euclidean algorithm* (refer to [MvOV96, page 71]).

2.19 Theorem (*Chinese remainder theorem*) If the integers $n_1, n_2, \ldots, n_k$ are pairwise relatively prime, the the system of simultaneous congruences

$$\begin{aligned} x &\equiv a_1 \pmod{n_1} \\ x &\equiv a_2 \pmod{n_2} \\ &\vdots \\ x &\equiv a_k \pmod{n_k} \end{aligned}$$

has a unique solution modulo $n = n_1 n_2 \ldots n_k$, which is given by

$$x = \sum_{i=1}^{k} a_i N_i M_i \bmod n,$$

where $N_i = n/n_i$ and $M_i = N_i^{-1} \bmod n_i$. These computations can be performed in $O((\lg n)^2)$ bit operations.

2.20 Definition The *multiplicative group* of $\mathbb{Z}_n$ is $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid gcd(a, n) = 1\}$.

2.21 Definition Let $a \in \mathbb{Z}_n^*$. The *order* of a, denoted $ord(a)$, is the least positive integer t such that $a^t \equiv 1 \pmod{n}$.

2.22 Definition Let $\alpha \in \mathbb{Z}_n^*$. If the *order* of α is $\phi(n)$, then α is said to be a *generator* or a *primitive element* of $\mathbb{Z}_n^*$. If $\mathbb{Z}_n^*$ has a generator, then $\mathbb{Z}_n^*$ is said to be *cyclic*.

2.23 Fact $\mathbb{Z}_n^*$ has a generator if and only if $n = 2,\ 4,\ p^k$, or $2p^k$, where p is an odd prime and $k \geq 1$.

2.24 Definition Let $a \in \mathbb{Z}_n^*$. The integer a is said to be a *quadratic residue* modulo n, if there exists an $x \in \mathbb{Z}_n^*$ such that $x^2 \equiv a \pmod{n}$. If no such x exists, then a is called a *quadratic non-residue* modulo n. The set of all quadratic residues modulo n is denoted by Q_n and the set of all quadratic non-residues is denoted by $\overline{Q}_n$.

2.25 Definition Let $a \in Q_n$. If $x \in \mathbb{Z}_n^*$ satisfies $x^2 \equiv a \pmod{n}$, x is called a *square root* of a modulo n.

2.26 Definition Let p be an odd prime and a an integer, the *Legendre symbol* is defined to be

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{if } p|a, \\ 1, & \text{if } a \in Q_p, \\ -1, & \text{if } a \in \overline{Q}_p. \end{cases}$$

2.27 Definition Let $n \geq 3$ be odd with prime factorization $n = p_1^{e_1} p_2^{e_2} \ldots p_k^{e_k}$. The the Jacobi symbol is defined to be

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{e_1} \left(\frac{a}{p_2}\right)^{e_2} \cdots \left(\frac{a}{p_k}\right)^{e_k}.$$

Note that if n is prime, the *Jacobi symbol* is just the Legendre symbol. There exists a polynomial-time algorithm [MvOV96, page 73] to compute the Legendre symbol. Based on results from number theory, we can evaluate a Jacobi symbol in polynomial time without factoring n using the same algorithm.

2.28 Definition Let $n \geq 3$ be and odd integer, and let $J_n = \{a \in \mathbb{Z}_n^* \mid \left(\frac{a}{n}\right) = 1\}$. The set of *pseudosquares* modulo n, denoted $\widetilde{Q}_n$, is defined to be the set $J_n - Q_n$.

Chapter 3

Basic Concepts

Cryptography is the study of mathematical techniques related to aspects of information security. Cryptographic primitives and computational difficulty are linked in a fundamental way, as cryptographic primitives can be constructed based on various intractability assumptions. At the very heart of cryptography is the notion of *one-way function*, which was shown to be necessary and sufficient for many cryptographic primitives [OW93]. In this section we define one-way function and describe the role of one-way functions in various cryptographic contexts.

3.1 One-way functions

In the construction of cryptographic schemes, we are concerned with both the computational efficiency and the infeasibility of violating the scheme. The computations of the legitimate users of the scheme ought be efficient; whereas violating the security features (by an adversary) ought to be infeasible. A complexity gap (i.e., between the complexity of proper usage and the complexity of defeating the security) is required. Hence, one-way functions play a central role in cryptography. A one-way function (OWF) is a mathematical function that is significantly easier to perform in one direction (the forward direction) than in the opposite direction (the inverse direction).

3.1 Definition (OWF, intuitive definition) [MvOV96, page 327]
A *one-way function* is a function f such that for each x in the domain of f, it is easy to compute $f(x)$; but for essentially all y in the range of f, it is computationally infeasible to find any x such that $y = f(x)$.

Note that by saying "for essentially all y", we don't exclude the possibility that for a few values y it is easy to find an x such that $y = f(x)$. For a better understanding of this, we include below (see 3.2) the more rigorous definition of one-way function by Goldreich[1].

3.2 Definition (OWF, Goldreich's definition) [Gol98, page 29]
A function $f : X \to Y$, where $X, Y \subseteq \{0,1\}^*$, is called *(strongly) one-way* if the following two conditions hold

1. easy to compute: There exist a (deterministic) polynomial-time algorithm, A, so that on input x algorithm A outputs $f(x)$ (i.e., $A(x) = f(x)$).

2. hard to invert: For every probabilistic polynomial-time inverting algorithm, A', every polynomial $p(\cdot)$, and all sufficiently large n

$$\Pr(A'(f(U_n), 1^n) \in f^{-1}(f(U_n))) < \frac{1}{p(n)},$$

where n is the length of input x, U_n denotes a random variable uniformly distributed over $X_n \subseteq \{0,1\}^n$, and $p(\cdot)$ is a polynomial depending on one variable.

"Hardness to invert" is interpreted as an upper bound on the success probability of efficient inverting algorithms. Clearly, the success probability obtained by repeating the algorithm polynomial (in n) many times is still negligible. Hence, defining negligible success as "occurring with probability smaller than any polynomial fraction" is analog to defining feasible as "computed within expected polynomial-time".

[1] Goldreich called it "strong one-way function" [Gol98, page 29].

In fact, there are no known instances of functions which are provably one-way (with no assumptions) [MvOV96, page 328]. All instances of "one-way functions" to date should thus be more properly qualified as "conjectured" or "candidate" one-way functions. Although it is widely believed that one-way functions do exist, it remains possible that they do not. As a fact, almost all of Modern Cryptography rises or falls with the question of whether one-way functions exist.

The following are two examples of candidate one-way functions.

3.3 Example (OWF – multiplication of large primes) For primes p and q, $f(p, q) = pq$ is a one-way function: given p and q, computing $n = pq$ is easy; but given n, finding p and q is difficult. The difficult direction is known as the *integer factorization* problem, RSA and many other cryptographic systems rely on this example.

3.4 Example (OWF – exponentiation in prime fields) Given a generator α of Z_p^*, for most appropriately large prime p, $f(x) = \alpha^x \pmod{p}$ is a one-way function. $f(x)$ is easily computed given α, x, and p; but for most choices p it is difficult, given (y, p, α), to find an x in the range $0 \leq x \leq p-2$ such that $\alpha^x \pmod{p} = y$. The difficult direction is known as the *discrete logarithm* problem. Exponentiation in other groups is also a reasonable candidate for a one-way function, provided that the discrete logarithm problem for the group is believed to be hard. For example, the logarithm problem in the group of points on an elliptic curve.

However, a one-way function is not sufficient for public-key cryptography if it is equally hard for the legitimate receiver and the adversary to invert. So rather, we need a trapdoor one-way function. A trapdoor one-way function is a one-way function where the inverse direction is easy, given a certain piece of information (the trapdoor), but difficult otherwise.

3.5 Definition (TDOWF) A *trapdoor one-way function* is a one-way function $f : X \to Y$ with the additional property that given some extra information that depends only on f not on x (called the trapdoor information) it becomes feasible to find for any given $y \in Im(f)$, an $x \in X$ such that $f(x) = y$.

Public-key cryptosystems are based on one-to-one (presumed) trapdoor one-way functions. The public key gives information about the particular instance of the function; the private key gives information about the trapdoor. Whoever knows the trapdoor can perform the function easily in both directions, but anyone lacking the trapdoor can perform the function only in the forward direction. The forward direction is used for encryption and signature verification; the inverse direction is used for decryption and signature generation.

Definitions of OWF and TDOWF can be extended to that of *one-way permutation* and *trapdoor one-way permutation* simply by substituting "function" with "permutation".

Since the existence of one-way functions has not been proved, the existence of trapdoor one-way functions/permutations is also unknown. However, there are a number of good candidates, and some of them will be discussed in Chapter 4 – 9.

3.2 Some main topics in cryptography

One-way functions are fundamental to cryptography in that they were shown to be necessary and sufficient for many cryptographic primitives. OWF is necessary and sufficient for pseudorandom bit generators, digital signatures, computational symmetric-key cryptography, coin-flipping, and identification [NY89, Rom90, IL89, BCG89, OW93]. TDOWF is sufficient for public-key cryptography, and oblivious transfer (therefore any two-party protocols). In this section we present some basic concepts including encryption, symmetric-key and public-key, digital signatures, etc.

3.2.1 Encryption schemes

The traditional and most basic problem of cryptography is that of providing secret communication over insecure media. The general setting consists of two parties communication through a channel which is possibly tapped by an adversary. The parties want to exchange information without leaking the content to the adversary.

Loosely speaking, an encryption scheme is a protocol that allows these parties to communicate secretly. Typically, the encryption scheme consists of a pair of algorithms. One algorithm, called *encryption*, is applied by the sender (i.e., the party sending a message), while the other algorithm, called *decryption*, is applied by the receiver. Hence, in order to send a message, the sender first applies the encryption algorithm to the message, and sends the result, called the *ciphertext*, over the channel. Upon receiving a ciphertext, the receiver applies the decryption algorithm to it, and retrieves the original message (called the *plaintext*).

For real security, each algorithm is indeed a set of transformations characterized by parameters and/or auxiliary inputs known as the *key*. The range of possible values of the key is called the *keyspace*

3.6 Definition A *cryptosystem* has five components [Sti95]:

1. A plaintext space, $\mathcal{P}$;
2. A ciphertext space, $\mathcal{C}$;
3. A keyspace, $\mathcal{K}$;
4. A family of encryption transformations, $\mathcal{E}$;
5. A family of decryption transformations, $\mathcal{D}$;

For each $K \in \mathcal{K}$, there is an encryption rule $e_K \in \mathcal{E}$ and a corresponding decryption rule $d_K \in \mathcal{D}$. Each $e_K : \mathcal{P} \to \mathcal{C}$ and $d_K : \mathcal{C} \to \mathcal{P}$ are functions such that $d_K(e_K(x)) = x$ for every plaintext $x \in \mathcal{P}$.

3.2.2 Security

There are two approaches to defining security [Gol99]:

The first ("classic") approach is *information theoretic*. It is concerned with the "information" about the plaintext which is "present" in the ciphertext. Loosely speaking, if the ciphertext contains information about the plaintext then the encryption scheme is considered insecure. It has been shown that such high level of security can be achieved only if the key in use is at least as long as the total length of the messages sent via the encryption scheme. The fact, that the key has to be longer than the exchanged information, is indeed a drastic limitation on the practical uses of such schemes.

The second ("modern") approach is based on *computational complexity.* It comes from the observation that it does not matter whether the ciphertext contains information about the plaintext, but rather whether this information can be efficiently extracted. In other words, we ask whether it is feasible for the eavesdropper to extract this information, instead of asking whether it is possible for him to do so. It turns out that this approach may offer security even if the key is much shorter than the total length of the messages sent via the encryption scheme.

3.2.3 Symmetric-key vs. public-key

There are two general forms of key-based encryption schemes: symmetric-key and public-key.

Traditional encryption schemes are based on the sender and receiver of a message knowing and using the same secret key: the sender uses the secret key to encrypt the message, and the receiver uses the same secret key to decrypt the message. This method is known as the *secret-key* or *symmetric-key* scheme[2]. In fact all the encryption schemes used prior to the 1980's are symmetric-key schemes. The eavesdropper in these schemes must be ignorant of the encryption key, and consequently the key distribution problem arises (i.e., how can two parties wishing to communicate over an insecure channel agree on a secret encryption/decryption key).

In contrast, the computational complexity approach allows the introduction of encryption schemes where the encryption key may be given to the eavesdropper without

[2]Symmetric-key systems are also referred to as private-key systems. To avoid confusing with the private key in public-key systems, we use the term "symmetric-key" throughout this thesis.

compromising the security of the scheme. Clearly, the decryption key in such schemes is different and furthermore infeasible to compute from the encryption key. The concept of *public-key* cryptography was introduced in 1976 by Diffie and Hellman [DH76]. In their concept, each person gets a pair of keys, one called the public key and the other called the private key. Each person's public key is published while the private key is kept secret.The key distribution problem thus is trivially resolved since all communications involve only public keys, and no private key is ever transmitted or shared. When Alice wishes to send a secret message to Bob, she looks up Bob's public key in a directory, uses it to encrypt the message and sends it off. Bob then uses his private key to decrypt the message and read it. No one listening in can decrypt the message. Anyone can send an encrypted message to Bob but only Bob can read it.

3.2.4 Digital signatures

A signature scheme (also called digital signature) is a method of signing a message stored in electronic form. In comparison to "conventional" handwritten signatures, digital signatures are message dependent: signatures are created by a signing transformation of the message, and verified by a verification transformation also involving the message.

Digital signatures can be based on OWF or TDOWF. Again, there are symmetric-key and public-key versions consists of three algorithms corresponding to the key-generation, signing and verification tasks. The difference between the two types lies in the definition of security (i.e., whether the adversary is given access to the verification-key). Public-key signature schemes produce signatures which are universally verifiable, since the verification-key is publicly available. In contrast, symmetric-key signature schemes are only used to authenticate messages sent among a small set to mutually trusting parties. Therefore symmetric-key signature schemes are commonly referred to as message authentication scheme.

There is a class of digital signatures which arise from public-key encryption techniques. For example, the RSA signature scheme derives directly from the RSA public-key encryption. As in the case of decryption, the signing-key is the secret information which distincts the legitimate signer from all other users. Other users only have the corresponding verification-key allowing them to verify signatures (but not to produce them).

3.2.5 Other applications of OWF

Next we briefly describe some other important applications of OWF.

There are many situations in cryptography where random numbers or bit-strings are needed. In practice it is common to use a *pseudo-random bit generator* (PRBG). A PRBG starts with a short random bit-string (a "seed") and expands it into a much longer "random-looking" bit string. In other words, although the output of a PRBG is not really random, it is infeasible to tell the difference. A PRBG can be constructed from any OWF. In fact, PRBG exists if and only if OWF exists [Lub96].

It turns out that PRBG plays a central role in the construction of others primitives, such as symmetric-key cryptosystems, digital signatures, zero-knowledge proofs, and bit commitment.

A proof refers to a process by which the validity to an assertion is established. Proofs in cryptographic protocols are often dynamic interactive processes, in which one party P (the prover) tries to prove a certain fact to the other party V (the verifier). Loosely speaking, *zero-knowledge proofs* are proofs which yield nothing beyond the validity of the assertion. That is, a verifier obtaining such a proof gains no knowledge beyond the conviction in the validity of the assertion.

An essential tool used in zero-knowledge proofs is bit commitment schemes. A *bit commitment* simply means that a player in the protocol is able to choose a bit and commit to his choice such that he can no longer change his mind. A bit commitment scheme consists of two phases. In the commit phase, P commits to a bit b, and sends the encrypted form of b (called a blob) to V. In the release phase, P can "open" the blob to reveal b and it is guaranteed that he cannot reveal a value other than the one committed. Bit commitment schemes are of great interest because they are a key ingredient in the construction of any two-party protocols. Their simple functionality enables complicated, otherwise seemingly impossible tasks.

Chapter 4

Candidate OWFs Reducible to Factoring

Factoring is the hard direction of a conjectured OWF. In this Chapter we examine the factoring problem, and three problems that are reducible to factoring — the RSA problem, the Rabin problem, and the quadratic residuosity problem. All of them are candidate OWFs.

4.1 The factoring problem

4.1 Definition The integer factorization problem (FACTORING)
Given a positive integer n, find its prime factorization; that is, write $n = p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ where the p_i are pairwise distinct primes and each $e_i \geq 1$.

Factoring is widely believed to be a hard problem, yet this has not been proven. The worst cases turn out to be when n is a product of large primes. Mathematicians and computer scientists have been very actively searching for efficient factoring algorithms. The best algorithms known (see [MvOV96] for a summary) have time complexity $L_n[\alpha, c]$, where $\alpha = 1/2, 1/3$. It is superpolynomial in the size (the number of digits, i.e., $\log n$). The fastest algorithm, the number field sieve, achieves $\alpha = 1/3$.

There remains a possibility that an easy factoring algorithm will be discovered. There is also the possibility that someone will prove that factoring is difficult. Above all this, there is the threat from a quantum computer — if one is ever developed — on which factoring can be solved efficiently using Shor's algorithm [Sho97]. We will cover Shor's algorithm later in Chapter 6.

4.2 The RSA problem

4.2 Definition The RSA problem (RSAP)
Given a positive integer n that is a product of two distinct odd primes p and q, a positive integer e such that $\gcd(e, (p-1)(q-1)) = 1$, and an integer c, find an integer m such that $m^e \equiv c \pmod{n}$

In other words, the RSA problem is that of finding the e^{th} roots modulo a composite integer n. The underlying one-way function, $f(x) = x^e \pmod{n}$, $(f : Z_n \to Z_n)$ is called the RSA function. The inverse is $f(x)^{-1} = x^d \pmod{n}$, where $d \equiv e^{-1} \pmod{\phi(n)}$. The conditions imposed on the problem parameters n and e ensure that the function is in fact a permutation over its domain. It is conjectured that the RSA function is a trapdoor one-way permutation [Gol98], with the factors of n serving as the trapdoor information.

If an opponent knows the trapdoor (p, q), he can compute $\phi(n) = (p-1)(q-1)$ and then compute d as the inverse of e using the extended Euclidean algorithm, thus easily solve the RSA problem. This fact is stated next.

4.3 Fact RSAP $\leq_P$ FACTORING. That is, the RSA problem polytime reduces to the integer factorization problem. [MvOV96]

However, it is unknown whether there might be other easier ways of breaking RSA without factoring n. The best algorithms known for inverting RSA proceed by (explicitly or implicitly) factoring n except for small d[1]. It is widely believed that without the knowledge of the factorization of n, it is infeasible to invert RSA, yet no proof of this is known. In other words, we have no proof that shows how secure RSA really is. This problem became the motivation of designing *provably secure* cryptosystems whose security can be mathematically proved to be equivalent to the difficulty of factoring.

The RSA cryptosystem is one of the most well-known and popular public-key cryptosystem. It was invented in 1977 by Rivest, Shamir, and Adleman [RSA78]. It may be used to provide both secrecy and digital signatures.

In the RSA public-key encryption, the RSA function serves as the encryption function, and the inverse function as the decryption function. Suppose Alice wants to send a message m to Bob. Bob's public key is (n, e), and his private key is (d, p, q). Alice creates the ciphertext c by exponentiating: $c = m^e \bmod n$. She sends c to Bob. To decrypt, Bob also exponentiates: $m = c^d \bmod n$; the relationship between e and d ensures that Bob correctly recovers m. Since only Bob knows d, only Bob can decrypt.

Because the encryption and decryption functions are mutual inverses, the RSA scheme can be used for digital signatures as well. Suppose Bob wants to send a message m to Alice in such a way that Alice is assured that the message is authentic and is from Bob. Bob creates a digital signature s by exponentiating: $s = m^d \bmod n$, where d is Bob's private key. He sends m and s to Alice. To verify the signature, Alice exponentiates and checks that the message m is recovered: $m = s^e \bmod n$, where (n, e) is Bob's public key.

The RSA pseudorandom bit generator [ACGS88] is based on the assumption that the RSAP is intractable. The generator first selects a random seed, x_0, then computes the sequence $x_1, x_2, \ldots, x_l$ by successively applying the RSA function. The

[1] An attack on RSA with short d is known from Wiener [Wie90]. This attack will discover d where $|d| < |n|/4$. More recent results improve Wiener's attack to $|d| < 0.292|n|$ [BD98, DN00]. These attacks pose no threat to normal case RSA where $|d| \sim |n|$.

sequence of pseudorandom bits is formed by the sequence of the least significant bit of x_i. The efficiency is furtherly improved in the Micali-Schnorr pseudorandom bit generator [MS91] by generating more bits per exponentiation by e. Yet the security of Micali-Schnorr PRBG is stronger than requiring that the RSAP is intractable.

A variety of provably secure public-key schemes [Rab79, Wil80, Wil85, SW95] have been developed whose security is computationally equivalent to the difficulty of factoring. The basic idea underlying most these systems is to replace the exponent e in the RSA system by λe, where λ is a small prime (usually, $\lambda = 2$ *or* 3, but larger values of λ are possible, specifically $\lambda = 5$ [SW95]. Upon raising a ciphertext to the secret exponent d, the receiver obtains not the original message, but its λth power. As a result, the sender needs to provide a clue indicating which of the λth roots of this power is the correct message. All these schemes were shown to to be as difficult to break as it is to factor n.

4.3 The quadratic residuosity problem

4.4 Definition The quadratic residuosity problem (QRP)
Given an odd composite integer n and an integer a having Jacobi symbol $\left(\frac{a}{n}\right) = 1$, decide whether or not a is a quadratic residues modulo n.

Let n be a product of two distinct primes p and q, it can be shown that if $a \in J_n$, then $a \in Q_n$ if and only if $a \in Q_p$ and $a \in Q_q$. Thus, if the factorization of n is known, the quadratic residuosity problem can be solved simply by computing the Legendre symbol $\left(\frac{a}{p}\right)$. This observation can be generalized to all integers n and leads to the following fact.

4.5 Fact QRP $\leq_P$ FACTORING. That is, the QRP polytime reduces to the FACTORING problem. [MvOV96]

On the other hand, if the factorization of n is unknown, there is no known efficient procedure for solving QRP. It is widely believed that QRP is as hard as the integer factorization problem, although no proof is this is known.

The intractability of quadratic residuosity problem forms the basis for the security of the Goldwasser-Micali probabilistic public-key encryption scheme. The Goldwasser-Micali public-key cryptosystem [GM84] encrypts one bit at a time. A 0 bit is encrypted to a random quadratic residues modulo n; a 1 bit is encrypted to a random pseudosquare modulo n. The receiver uses his trapdoor knowledge (i.e. the factorization of n) to determine whether the element he receive is a quadratic residue or a pseudosquare, therefore the original bit.

4.4 The square root modulo n problem

4.6 Definition The square root modulo n problem (SQROOT)
Given a composite integer n and $a \in Q_n$ (the set of quadratic residues modulo n), find a square root of a modulo n; that is , an integer x such that $x^2 \equiv a \pmod{n}$

Note that the SQROOT problem is not a special case of the RSA problem: since $p-1$ is even, it follows that e is odd, and in particular $e \neq 2$. The conjectured one-way function in the SQROOT problem is $f(x) = x^2 \pmod{n}$, with (p, q) as the trapdoor. This function induces a 4-to-1 mapping on the multiplicative group modulo n.

If the the factors p and q of n are known, then the SQROOT problem can be solved efficiently by first finding square roots of a modulo p and modulo q, and then combining them using the Chinese remainder theorem (2.19) to obtain the square root of a modulo n.

Conversely, if one can solve the SQROOT problem, then the FACTORING problem is easy. It works as follows. First compute $a = x^2 \bmod n$ for a random x coprime to n. Then find a square root y by solving the SQROOT problem with $(a,\, n)$. If $y \equiv \pm x \pmod{n}$, then the trial fails, and the above procedure is repeated with a new x. Otherwise, $y \not\equiv \pm x \pmod{n}$, $\gcd(x - y, n)$ is guaranteed to be a non-trivial factor of n, namely, p or q. The procedure runs in expected polynomial time.

Therefore we have the following fact.

4.7 Fact SQROOT and FACTORING are computationally equivalent. [MvOV96]

If n is a Blum integer (i.e., n is a product of two distinct primes each congruent to 3 modulo 4), then the function defined above is a permutation. When $p, q \equiv 3 \pmod 4$, the computation of square roots is very easy. The two square root of a modulo p are $\pm a^{(p+1)/4} \bmod p$, and the two square root of a modulo q are $\pm a^{(q+1)/4} \bmod q$ [Sti95]. It is then straightforward to obtain the four square roots of a modulo n using the Chinese remainder theorem.

The Rabin public-key scheme [Rab79] is based on the above trapdoor one-way permutation. It was the first *provably secure* public-key encryption and signature scheme — that is, the underlying problem of the scheme is provably as difficult as some computational problem that is widely believed to be difficult, such as FACTORING or DLP.

The Blum-Blum-Shub (BBS) pseudorandom bit generator [BBS86] is based on the assumption that integer factorization is intractable. It works in a similar way to the RSA PRBG, using $f(x) = x^2 \pmod n$ where n is a Blum integer. The BBS generator forms the basis for the Blum-Goldwasser probabilistic public-key encryption scheme [BG85]. The scheme uses the BBS generator to generate a pseudorandom bit sequence which is then XORed with the plaintext. The resulting ciphertext, together with an encryption of the random seed used, is sent to the receiver. The receiver uses his trapdoor information to recover the seed and subsequently reconstruct the pseudorandom bit sequence and the plaintext.

Chapter 5

Candidate OWFs DL and GDL

In this Chapter we look at the discrete logarithm problem and its generalized version. Like the factoring problem, the discrete logarithm problem is believed to be difficult and also to be the hard direction of a one-way function.

The intractability of discrete logarithm problem forms the basis of many cryptographic techniques, including Diffie-Hellman key agreement and its derivatives, ElGamal encryption, and the ElGamal signature scheme and its variants [MvOV96]. We will briefly talk about Diffie-Hellman key agreement, ElGamal encryption, ElGamal signature and DSS as examples. The discrete logarithm problem appears to be much harder over arbitrary groups than over finite fields; this is the motivation for cryptosystems based on elliptic curves. Generalized discrete logarithm problem is examined in the setting of elliptic curve.

5.1 The discrete logarithm problem

If G is a group, such as the multiplicative group of a finite field or the group of points on an elliptic curve, and α is an element of G, then (writing the group multiplicatively) α^x is the *discrete exponentiation* of base α to the power x. This operation shares many properties with ordinary exponentiation, so that, for example,

$$\alpha^{x+y} \equiv \alpha^x \cdot \alpha^y \pmod{n}.$$

Finding discrete logarithm is the inverse operation of discrete exponentiation. For simplicity assume that G is cyclic and is generated by α, the formal definition follows:

5.1 Definition Discrete logarithm (DL)
Let G be a finite cyclic group of order m. Let α be a generator of G, and let $\beta \in G$. The *discrete logarithm of β to the base α*, denoted $\log_\alpha \beta$, is the unique integer x, $0 \leq x \leq m-1$, such that $\beta = \alpha^x$.

The number x is called the discrete logarithm, since it again shares many properties with the ordinary logarithm. For example,

$$\log_\alpha(\beta\gamma) \equiv (\log_\alpha \beta + \log_\alpha \gamma) \pmod{m}$$

The problem of finding discrete logarithm defined on $\mathbb{Z}_p^*$ is known as the discrete logarithm problem (DLP).

5.2 Definition The discrete logarithm problem (DLP)
Given a prime p, a generator α of $\mathbb{Z}_p^*$, and an element $\beta \in \mathbb{Z}_p^*$, find the integer x, $0 \leq x \leq p-2$, such that $\alpha^x \equiv \beta \pmod{p}$.

More generally, the discrete logarithm problem can be defined in a finite cyclic group as follows:

5.3 Definition The generalized discrete log problem (GDLP)
Given a finite cyclic group G of order n, a generator α of G, and an element $\beta \in G$, find the integer x, $0 \leq x \leq n-1$, such that $\alpha^x = \beta$.

The groups of most interest in cryptography are the multiplicative group $\mathbb{F}_q^*$ of the finite field $\mathbb{F}_q$, including the particular case of the multiplicative group $\mathbb{Z}_p^*$ of the integers modulo a prime p, and the the multiplicative group $\mathbb{F}_{2^m}^*$ of the finite field $\mathbb{F}_{2^m}$ of characteristic two. Also of interest are the group $\mathbb{Z}_n^*$ where n is a composite integer, the group of points on an elliptic curve defined over a finite field, and the jacobian of a hyperelliptic curve defined over a finite field [MvOV96].

The discrete logarithm problem is a well-studied problem. The best discrete logarithm algorithms have expected running times similar to those of the best factoring algorithms. A summary of the known algorithms for the DLP can be found in [MvOV96].

Currently, the best algorithms to solve the discrete logarithm problem are broken into two classes: index-calculus methods and collision search methods. Index calculus methods are very similar to the fastest current methods for integer factoring and they run in superpolynomial-time. Collision search algorithms have purely exponential running time. Index calculus methods generally require certain arithmetic properties to be present in order to be successful, whereas collision search algorithms can be applied much more generally. Collision search methods is the best known method for attacking the general elliptic curve discrete log problem.

5.2 The Diffie-Hellman problem

The Diffie-Hellman problem is closely related to the discrete logarithm problem. Its assumed intractability forms the basis for the security of many cryptographic schemes, including Diffie-Hellman key agreement and its derivatives, and ElGamal public-key encryption.

5.4 Definition The Diffie-Hellman problem (DHP)
Given a prime p, a generator α of $\mathbb{Z}_p^*$, and elements $\alpha^a \bmod p$ and $\alpha^b \bmod p$, find $\alpha^{ab} \bmod p$.

5.5 Definition The generalized Diffie-Hellman problem (GDHP)
Given a finite cyclic group G, a generator α of G, and group elements α^a and α^b, find α^{ab}.

Suppose that the DLP could be efficiently solved. Then given α, p, $\alpha^a \bmod p$ and $\alpha^b \bmod p$, one could first find a by solving the DLP from α, p, and $\alpha^a \bmod p$, and then compute $(\alpha^b)^a = \alpha^{ab} \bmod p$. This establishes the following relation between the DHP and the DLP.

5.6 Fact DHP $\leq_P$ DLP. That is, DHP polytime reduces to the DLP. More generally, GDHP $\leq_P$ GDLP. [MvOV96]

Whether the GDHP and the GDLP are computationally equivalent remains an open question.

5.3 Diffie-Hellman key agreement

The Diffie-Hellman key agreement protocol is based on the Diffie-Hellman problem. It was developed by Diffie and Hellman [DH76] in 1976 and published in the ground-breaking paper "New Directions in Cryptography". The protocol (together with authentication) allows two users to exchange a secret key over an insecure medium.

The basic approach is that if Alice and Bob wish to create a common secret key, they agree on a group G, and then Alice chooses a random integer a, while Bob chooses a random integer b. Alice then computes α^a and sends it to Bob over a public channel, while Bob computes α^b and sends that to Alice. Now Alice and Bob can both compute

$$\alpha^{ab} = (\alpha^a)^b = (\alpha^b)^a,$$

while an eavesdropper who happens to have overheard the exchange, and thus knows α, α^a, and α^b, will hopefully not be able to compute the secret α^{ab}.

5.4 ElGamal public-key encryption

The ElGamal public-key encryption [ElG85] is based on the Diffie-Hellman problem.

Bob has a private key a and a public key (p, α, β), where $\beta \equiv \alpha^a \pmod{p}$. Suppose Alice wishes to send a message m to Bob. She first generates a random number k less than $p-1$, then computes

$$e_K(m,k) = (y_1, y_2) = (\alpha^k, m\,\beta^k) \bmod p.$$

Alice sends (y_1, y_2) to Bob. Upon receiving the ciphertext, Bob computes

$$d_K(y_1, y_2) = y_2 ({y_1}^a)^{-1} \bmod p$$

.

In the ElGamal cryptosystem, the plaintext m is "masked" by multiplying it by β^k, yielding y_2. The value α^k is also transmitted as part of the ciphertext. Bob knows the secret exponent a, which enables him to compute β^k from α^k. Then he can "remove the mask" by dividing y_2 by β^k to obtain m. Clearly, the ciphertext depends both on the plaintext m and the random value k chosen by Alice. Therefore, the ElGamal Cryptosystem is probabilistic, as there will be many ciphertexts that are encryptions of the same plaintext.

The problem of breaking the ElGamal encryption scheme is equivalent to solving the Diffie-Hellman problem [MvOV96]. In fact, the ElGamal encryption scheme can be viewed as simply comprising a Diffie-Hellman key exchange to determine a session key α^{ak}, and then encrypting the message by multiplication with that session key. For this reason, the security of the ElGamal encryption scheme is said to be *based* on the discrete logarithm problem in $\mathbb{Z}_p^*$, although such an equivalence has not been proven.

Analysis based on the best available algorithms for both factoring and discrete logarithms shows that RSA and ElGamal have similar security for equivalent key lengths [Lab00]. The main disadvantage of ElGamal is the need for randomness, and its slower speed (especially for signing). Another potential disadvantage of the ElGamal system is that message expansion by a factor of two takes place during encryption. However, such message expansion is negligible if the cryptosystem is used only for exchange of secret keys.

5.5 The ElGamal signature scheme and DSS

The ElGamal signature algorithm is similar to the encryption algorithm in that the public key and private key have the same form. However, encryption is not the same as signature verification, nor is decryption the same as signature creation as in RSA. The ElGamal signature scheme is designed specifically for the purpose of signatures.

The ElGamal signature scheme was, in part, the basis for several later signature schemes, including one by Schnorr [Sch89], which in turn was the basis for DSS, the Digital Signature Standard. DSS makes use of computation of discrete logarithms in certain subgroups of $\mathbb{Z}_p^*$, where p is allowed up to 1024 bits. In 1994, the DSS was adopted by the U.S. National Institute of Standards and Technology (NIST) to be the digital authentication standard of the U.S. government.

The security of the ElGamal signature scheme and its variants relies on the discrete logarithm problem. However, it remains unproven that these schemes are secure even if the discrete logarithm problem is hard.

5.6 The elliptic curve discrete logarithm problem

The discrete logarithm problem is typically described in the setting of the multiplicative group $\mathbb{Z}_p^*$, but can be easily generalized to work in any finite cyclic group G. Depending on the cyclic group used, the discrete logarithm problem may be easy or (apparently) difficult. It is therefore useful to study other groups in the hope of finding other settings where the discrete logarithm problem seems to be intractable. The groups that have received the most attention are:

1. The multiplicative group $\mathbb{F}_{2^m}^*$ of the finite field $\mathbb{F}_{2^m}$ of characteristic two.

2. The group of points on an elliptic curve over a finite field.

In this section, we examine the GDLP in the setting of elliptic curve.

5.6.1 Introduction to elliptic curves

Elliptic curves has been the subject of many mathematical studies since the 19th century.

An elliptic curve can be defined over any field (e.g., real, rational, complex). However, elliptic curves used in cryptography are mainly defined over finite fields. An elliptic curve consists of elements (x, y) satisfying the equation[1]

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

where $a, b \in \mathbb{Z}_p$ are constants such that $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$, together with a special element $\mathcal{O}$ called the point at infinity.

There is a rule for adding two points on an elliptic curve E to get a third elliptic curve point. Such an operation is called *addition*, and denoted by $+$. Under this addition rule, the set of points on E forms an abelian group, with $\mathcal{O}$ serving as its identity.

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on an elliptic curve E.

1. $P + \mathcal{O} = \mathcal{O} + \mathcal{P} = \mathcal{P}$;
2. If $x_1 = x_2$ and $y_1 = -y_2$, then $P + Q = \mathcal{O}$;
3. Otherwise $P + Q = (x_3, y_3)$, where

$$x_3 = \lambda^2 - x_1 - x_2$$

$$y_3 = \lambda(x_1 - x_3) - y_1$$

and

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

[1] This equation can be used to define an elliptic curve over any field $\mathbb{F}_{p^n}$, for $p > 3$ prime, $n > 1$. An elliptic curve over $\mathbb{F}_{2^n}$ or $\mathbb{F}_{3^n}$ is defined by a slightly different equation. [Sti95]

It can be proven that the above addition rule indeed makes the points on E an abelian group [ST92]. This implies that if $P \in E$ and $Q \in E$, then it holds that $P + Q \in E$.

The addition operation in an elliptic curve is the counterpart to modular multiplication in common public-key cryptosystems, and multiple addition is the counterpart to modular exponentiation. Table 5.1 summarizes the correspondence between $\mathbb{Z}_p^*$ and E_p.

	Multiplicative Group	Elliptic Curve Group
Group	$\mathbb{Z}_p^*$	E or E_p
Elements	$\{1, 2, \dots, p-1\}$	Points (x, y) on E plus $\mathcal{O}$
Operation	Multiplication modulo p	Addition over E
Arithmetic notation	Elements: g, h	Elements: P, Q
	Multiplication: gh	Addition $p + Q$
	Inverse: h^{-1}	Negative: $-P$
	Division: g/h	Subtraction: $P - Q$
	Exponentiation: g^a	Multiple: aP
Discrete logarithm problem	Given g and $h = g^a$, find a	Given P and $Q = aP$, find a

Table 5.1: Notational correspondence between $\mathbb{Z}_p^*$ and E_p

5.6.2 The elliptic curve discrete logarithm problem (ECDLP)

Since the group of points on an elliptic curve E_p forms a cyclic group, we can extend the DLP to the elliptic curve discrete logarithm problem (ECDLP).

5.7 Definition The elliptic curve discrete logarithm problem (ECDLP)

Given an elliptic curve E_p, a point $P \in E_p$ of order n and a point $Q \in E_p$, find $a \in \mathbb{Z}_n$ such that $Q = aP$, provided that such an a exists.

The ECDLP has received much attention over the past decade. It is conjectured to be harder than the DLP and the factoring problem [Wie98]. The DLP on a finite field can be solved in superpolynomial time by the index calculus method. By contrast, the best attacks on the ECDLP in general[2] are brute-force methods, which run in exponential time. The index calculus method does not work for elliptic curves because elliptic curves don't have certain properties that may facilitate cryptanalysis.

As a result, shorter key sizes can be used to achieve the same security as larger keys in general finite field discrete log cryptosystems.

5.6.3 Elliptic curve cryptosystem

The use of elliptic curves in public-key cryptography was proposed independently by Koblitz [Kob87] and Miller [Mil85] in 1985.

Elliptic curve cryptosystems are analogs of existing public-key cryptosystems (such as RSA and ElGamal) in which modular multiplication is replaced by elliptic curve addition operation. One can easily construct elliptic curve encryption, signature, and key agreement schemes by making analogs of ElGamal, DSA, and Diffie-Hellman.

Elliptic curve cryptosystems have emerged as a promising new area in public-key cryptography in recent years due to their potential for offering similar security to established public-key cryptosystems with reduced key sizes. Shorter key-lengths bring about simpler arithmetic processors, and smaller band-width and memory requirements.

It should be noted that at equivalent key sizes elliptic curve cryptosystems are much slower than other public-key methods. If a superpolynomial-time algorithm

[2]For certain choices of elliptic curves there do exist more efficient attacks [MVO91]. However these cases are readily classified and easily avoided.

were found for the ECDLP, key sizes would have to increase greatly, and elliptic curve cryptosystems would no longer be competitive as a public-key method.

Discrete logarithm problem and factoring seem to enjoy the same level of difficulty. Historically, any algorithmic advance in one problem equally affects the other. Like factoring, discrete logarithm problem (including ECDLP) can be solved efficiently using Shor's algorithm, as we will detail in Chapter 6.

Chapter 6

Shor's Algorithm

In 1994, Peter Shor discovered polynomial-time algorithms [Sho97] to solve factoring or discrete logarithm problems on a hypothetical quantum computer. Boneh and Lipton [BL95] showed that using a variant of Shor's, discrete logarithm problem is solvable over any group including Galois fields and elliptic curves. Shor's discovery thus has a deep impact on cryptography, and spurs widespread interests in quantum computing.

6.1 Quantum computing

6.1.1 A brief history

Quantum computing is a new field in computer science that brings together ideas from classical information theory, computer science, and quantum physics. It holds the key to computers that may run exponentially faster than any known algorithm that runs on conventional computers for certain problems.

The field started in the early 1980s with suggestions by Benioff [Ben80] and Feynman [Fey82, Fey86]. Feynman observed that certain quantum mechanical effects could not be simulated efficiently on a computer. This observation led to speculation that perhaps computation in general could be done more efficiently using quantum effects. In 1985, Deutsch defined the universal quantum Turing machine [Deu85]. However, it wasn't until 1994 that this field saw exciting promises brought by Shor's algorithm. Shor discovered polynomial time quantum algorithms for integer factorization and discrete log, thus attacked many cryptosystems based on the hardness of these problems. Shor's work prompted a flurry of activity, both among experimentalists trying to build quantum computers and theorists trying to find other quantum algorithms and quantum error correcting codes.

Why are we interested in quantum computing? A prime motivation is that quantum mechanics might provide new and possibly very powerful ways of information processing. Highly parallel quantum algorithms can drastically decrease the computational time for some problems, thus promise to solve certain problems which are intractable on digital computers. Moreover, at the current pace, the ongoing miniaturization in chip design will lead to chip components as small as a few atoms within the next two decades (Moore's law [SR00]). That means we will eventually approach the regime where quantum theory is highly relevant to how computing devices function. Atomic scale sets the ultimate physical limits for classic gates. If computers are to become smaller and faster in the future, new, quantum devices must replace or supplement classical ones.

6.1.2 Basic concepts

The power of quantum computing comes from quantum parallelism. A quantum computer promises to be immensely powerful because it can be in multiple states at once (called superposition), and because it can act on all its possible states simultaneously. Thus, a quantum computer could naturally perform a myriad of operations in parallel, using only a single processing unit. A few basic concepts are important for understanding quantum computing.

Quantum superposition In classical computers, the fundamental unit of information is a bit. A bit can represent either a 0 state or a 1 state. In a quantum computer, information is represented using qubits (quantum analog of the classical bit).

6.1 Definition A *qubit* is a quantum state $|\psi\rangle$ of the form

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.

A qubit can be in a linear superposition of the two distinguishable physical states, i.e., can exist simultaneously as $|0\rangle$ or $|1\rangle$, with a complex amplitude for each state. Similarly, two qubits can be in a superposition of the four states ($|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$), and n qubits can be in a superposition of 2^n states.

Quantum parallelism Quantum computers operate on all values stored in any qubit at the same time. A quantum computer with an input of n qubits can execute a single gate operation on all the 2^n encoded values in $O(n)$ time. To perform the same task with a classical computer, 2^n processors would have to work in parallel, or else the computation would have to be repeated 2^n times. This phenomenon is known as *quantum parallelism.*

Quantum entanglement At the heart of quantum parallelism lies entanglement, which refers to quantum correlations of multiple qubits. An entangled state cannot be described as a tensor product[1] of states of the individual qubits. This means a qubit within the entangled state is not, by itself, in a pure state (but is in a statistical mixture of pure states); even though the multiple qubits as a whole are.

Measurement Accessing the results obtained through quantum parallelism proves tricky, because it requires measuring the final state of the qubits. When a

[1]The notion of tensor product is used in describing the state of a multi-bit quantum system. Refer to [Gru99] for more explanation.

qubit is measured the result will be $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with the complementary probability, $|\beta|^2$. Any measurement disturbs the quantum state — whenever a measurement is made, the state is transformed from a possibly complex superposition to a simple state. We cannot "see" a superposition itself, we will "see" one and only one classical state.

This difficulty in accessing values is a severe limitation, requiring highly unconventional algorithms. How do quantum algorithms give the result we look for? It is done through quantum interference.

Quantum interference Quantum interference, the analog of Young's double-slit experiment that demonstrated constructive and destructive interference phenomena of light, is one of the most significant characteristics of quantum computing. Quantum interference improves the probability of obtaining a desired result by constructive interference, and diminishes the probability of obtaining an erroneous result by destructive interference. Thus in some cases, among the exponentially many computations, the correct answer can theoretically be identified with appropriate quantum algorithms.

6.1.3 Quantum algorithm

A central issue of quantum computing is to devise algorithms that take advantage of quantum parallelism. Few good quantum algorithms are known to date. The two main examples are Shor's algorithm [Sho97] and Grover's algorithm [Gro96].

Shor's algorithm for factoring and discrete logarithm runs in polynomial-time, which is an superpolynomial speedup of the fastest classical algorithms. Grover's algorithm searches an unordered list in $O(\sqrt{n})$ time, while classical algorithms require $O(n)$. Although Grover's quadratic speedup is not as dramatic as the (conjectured) superpolynomial speedup achieved by Shor's factoring algorithm, it is provably better than any possible classical search algorithm.

6.1.4 Future development

In the last decade, quantum computing has become a prominent and promising area of theoretical computer science. Realizing this promise requires two things:

1. actually building a quantum computer;
2. discovering tasks where a quantum computer is significantly faster than a classical computer.

In theory, a quantum computer will be able to perform any task that a classical computer can. However, this does not necessarily mean that a quantum computer will outperform a classical computer for all types of task. If we use our classical algorithms on a quantum computer, it will simply perform the calculation in a similar manner to a classical computer. In order for a quantum computer to show its superiority it needs to use quantum algorithms which can exploit the phenomenon of quantum parallelism. Such algorithms are not easy to formulate. It is not yet known whether the power of quantum parallelism can be harnessed to solve a wide variety of problems.

The realization of a practical quantum computer still seems far away. Only a few, small-scale quantum computers have been built to date. The largest quantum computers so far have 100 logic operations on two qubits or 10 operations on seven qubits [SR00]. Moreover, it is unclear how this small-scale machine can be scaled up to a larger practical one, or whether it is even possible to do so [Pre97]. Most people believe that it will be increasingly difficult (and costly) to built bigger quantum computers because of the instability problems (decoherence). On the other hand, there is good reason for optimism since we see no fundamental physical barrier to building large quantum computers.

6.2 The Quantum Fourier Transform

The quantum Fourier transform (QFT) is a variant of the discrete Fourier transform (DFT). The DFT sends a discrete function to another discrete function, conventionally having as its domain equally spaced points $k\frac{2\pi}{N}$ in the interval $[0, 2\pi)$ for some N. By scaling the domain by $\frac{N}{2\pi}$, the quantum Fourier transform outputs a function with domain the integers between 0 and $N-1$.

The quantum Fourier transform operates on the amplitude of the quantum state, by sending

$$\sum_a g(a)|a\rangle \mapsto \sum_c G(c)|c\rangle,$$

where G is the discrete Fourier transform of g,

$$G(c) = \frac{1}{\sqrt{N}} \sum_a \exp(2\pi i a c/N) g(a),$$

and a and c both range over the binary representations for the integers between 0 and $N-1$. If the state is measured after the Fourier transform is performed, the probability that the result is $|c\rangle$ is $|G(c)|^2$.

Fourier transforms in general map from the time domain to the frequency domain. So Fourier transforms map functions of period r to functions which have non-zero values only at multiples of the frequency $1/r$. Thus applying the quantum Fourier transform to a periodic function $g(a)$ with period r, we would expect to end up with $\sum_c G(c)|c\rangle$, where $G(c)$ is zero except for multiples of N/r. Thus, when the state is measured, the result would be a multiple of N/r , say jN/r.

In order for Shor's algorithm to be a polynomial algorithm, the quantum Fourier transform must be performed in polynomial time. This requires [Sho94]: (1) N can be represented with a polynomial number of bits, and (2) that N must be smooth, i.e., must have "small" prime factors. Coppersmith [Cop94] and Deutsch (unpublished, see [EJ96]) independently found an efficient construction for the QFT based on the fast Fourier transform (FFT) algorithm [Knu81].

The QFT is a variant of the FFT which is based on powers of two, and only gives approximate results for periods which are not a power of two. However the larger the power of two used as a base for the transform, the better the approximation. Take

$N = 2^l$, the Fourier transform is

$$\sum_a g(a)|a\rangle \to \sum_c \left(\frac{1}{\sqrt{2^l}} \sum_a \exp(2\pi iac/2^l) g(a) \right) |c\rangle.$$

The classical version requires $O(N \log N)$ operations. In contrast, the QFT takes time $O((\log N)^2)$ by exploiting quantum parallelism. The implementation of the QFT is a network of one-bit and two-bit quantum gates. Specifically, the circuit uses two types of gates. One is a gate to perform the familiar Hadamard transformation, H. We will denote by H_j the Hadamard transformation applied to the jth bit. The other type of gate performs transformations of the form

$$S_{j,k} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\theta_{k-j}} \end{pmatrix}$$

where $\theta_{k-j} = \pi/2^{k-j}$ which acts on the kth element depending on the value of the jth element. The quantum Fourier transform is given by

$$H_0 S_{0,1} \ldots S_{0,l-1} H_1 \ldots H_{l-3} S_{l-3,l-2} S_{l-3,l-1} H_{l-2} S_{l-2,l-1} H_{l-1}$$

followed by a bit reversal transformation. For more details including the quantum circuit, see [Sho97].

Shor shows that the quantum Fourier transform with base 2^l can be constructed using only $l(l-1)/2$ gates [Sho97]. Thus quantum computers can efficiently solve certain problems with a periodic structure, such as factoring and the discrete log problem.

6.3 Shor's algorithm

Shor's algorithm has two phases: first, based on quantum computing; second,on classical computations. The classical phase involves efficient algorithms known from number theory such as continued fraction expansion and Euclid's algorithm. As all of quantum computing, Shor's algorithm is wholly probabilistic. Several repetitions of one or both phases may be necessary to find the correct result.

Most modern factoring algorithms, including Shor's, use a standard reduction of the factoring problem to the problem of finding the period of a function. The algorithm first uses quantum parallelism to compute all the values of the function in one step. Next it performs a quantum Fourier transform, putting all the amplitude of the function into multiples of the reciprocal of the period. With high probability, measuring the state yields the period, which in turn is used to factor the integer n.

6.3.1 Shor's algorithm for factoring

Shor's algorithm is based on calculating the period r of the function $f(x) = a^x \bmod n$ for a randomly selected integer a between 0 and n. Once r is known the factors of n are obtained by calculating the greatest common divisor of n and $a^{r/2} \pm 1$. The algorithm uses two essential quantum registers.

Step 1. Calculating $a^x \bmod n$ in quantum parallelism Choose an integer a arbitrarily. If a is not relatively prime to n, we've found a factor of n. Otherwise apply the rest of the algorithm. Let l be such that[2] $n^2 \leq 2^l < 2n^2$.

Prepare two quantum registers in state $|0, 0\rangle$. The first register has l qubits, and the second $\lceil \log n \rceil$ qubits. Apply a transformation called the Walsh-Hadamard transform to put the first register in the equally weighted superposition of all integers from 0 to $2^l - 1$.

$$|0, 0\rangle \mapsto \frac{1}{\sqrt{2^l}} \sum_{x=0}^{2^l-1} |x, 0\rangle.$$

Then we take advantage of quantum parallelism by computing $f(x) = a^x \bmod n$ for all the values of x in the superposition simultaneously. The values of $f(x)$

[2] This choice is made so that the approximation for non powers of 2 given by the quantum Fourier transform used in Step 3 will be good enough for the rest of the algorithm to work.

are placed in the second register so that after the computation the two registers become entangled:

$$\frac{1}{\sqrt{2^l}}\sum_{x=0}^{2^l-1}|x,0\rangle \mapsto \frac{1}{\sqrt{2^l}}\sum_{x=0}^{2^l-1}|x,f(x)\rangle.$$

Step 2. Measuring the second register Measure the second register, and obtain value $u = f(k)$ for some randomly selected k. This measurement also collapses the state of the first register into a superposition of all states $|x\rangle$ such that $x = k, k+r, k+2r, \ldots$, i.e. all x for which $f(x) = u$.

So the state after measurement is

$$C\sum_x g(x)|x,u\rangle,$$

for some scale factor C where

$$g(x) = \begin{cases} 1 & \text{if } f(x) = u, \\ 0 & \text{otherwise.} \end{cases}$$

The offset k is randomly selected by the measurement of the second register. It's impossible to directly extract r by measuring the first register because of k.

Step 3. Applying QFT The $|u\rangle$ part of the state will not be used, so we will no longer write it. Apply the quantum Fourier transform to the state obtained in Step 2.

$$QFT : \sum_x g(x)|x\rangle \mapsto \sum_c G(c)|c\rangle$$

Standard Fourier analysis tells us that when the period r of $g(x)$ is a power of two, the result of the quantum Fourier transform is

$$C'\sum_j \rho_j |j\frac{2^l}{r}\rangle$$

where $|\rho_j| = 1$. When the period r does not divide 2^l, the transform approximates the exact case so most of the amplitude is attached to integers close to multiples of $2^l/r$. The Fourier transform can be regarded as an interference between the various superposed states in the first register.

Step 4. Extracting the period Measure the state in the standard basis $(|0\rangle, |1\rangle)$ for quantum computation, and call the result v. The remaining part of the algorithm is classical.

In the case where the period happens to be a power of 2 so that the quantum Fourier transform gives exactly multiples of the scaled frequency, the period is easy to extract. In this case, $v = j2^m/r$ for some j. Most of the time j and r will be relatively prime, in which case reducing the fraction $v/2^m$ to it's lowest terms will yield a fraction whose denominator q is the period r.

The fact that in general the quantum Fourier transform only gives approximately multiples of the scaled frequency complicates the extraction of the period from the measurement. When the period is not a power of 2, a good guess for the period can be obtained using the continued fraction expansion of $v/2^m$ [RP98].

Step 5. Finding a factor of n When our guess for the period, q, is even, use the Euclidean algorithm to efficiently check whether either $a^{q/2}+1$ or $a^{q/2}-1$ has a non-trivial common factor with n.

The reason why $a^{q/2}+1$ or $a^{q/2}-1$ is likely to have a non-trivial common factor with n is as follows. If q is indeed the period of $f(x) = a^x \bmod n$, then $a^q = 1 \bmod n$. If q is even, we can write

$$(a^{q/2}+1)(a^{q/2}-1) \equiv 0 \bmod n.$$

Thus, so long as neither $a^{q/2}+1$ nor $a^{q/2}-1$ is a multiple of n, either $a^{q/2}+1$ or $a^{q/2}-1$ has a non-trivial common factor with n.

Step 6. Repeating the algorithm, if necessary Various things could have gone wrong so that this process does not yield a factor of n [RP98]:

1. The value v was not close enough to a multiple of $2^l/r$.
2. The period r and the multiplier j could have had a common factor so that the denominator q was actually a factor of the period not the period itself.
3. Step 5 yields n as n's factor.
4. The period of $f(x) = a^x \bmod n$ is odd.

A few repetitions of this algorithm yields a factor of n with high probability. It can be shown that $O(\log\log r)$ repetitions suffice to find the period r with a high probability. Given r, step 5 succeeds with probability at least $1 - 1/2^{k-1}$, where k is the number of distinct prime factors of n. For composite numbers, $k \geq 2$, so that the probability is at least $1/2$.

Shor's factoring algorithm takes $O((\log n)^2(\log\log n)(\log\log\log n))$ steps on a quantum computer, along with a polynomial (in $\log n$) steps on a classical computer. The overall time complexity is polynomial. The total number of elementary logic gates required is a polynomial of n.

6.3.2 Shor's algorithm for discrete log

The discrete log problem is to find $r \bmod p-1$ such that $x \equiv g^r \pmod{p}$, given p, g, and x. The OWF function involved is $f(r) = g^r \bmod p$. We have the following preliminary fact:

6.2 Fact Suppose that we know the order k of the base g. Then if $g^r \equiv x \pmod{p}$, the following equation

$$g^a x^{-b} \equiv u \pmod{p}$$

is satisfied for all integers $a, b \geq 0$ such that $a = k + br$.

Shor's algorithm for discrete log uses three quantum registers.

Step 1. Calculating $g^a x^{-b} \bmod p$ in quantum parallelism Prepare two quantum registers in state $|0,0\rangle$, and put both in the uniform superposition of all integers from 0 to $p-2$.

$$|0,0\rangle \mapsto \frac{1}{p-1}\sum_{a,b=0}^{p-2} |a,b\rangle.$$

We then compute $g^a x^{-b} \bmod p$ in the third register. This leaves our machine in the entangled state

$$\frac{1}{p-1}\sum_{a=0}^{p-2}\sum_{b=0}^{p-2} |a,b,g^a x^{-b} \bmod p\rangle.$$

Step 2. Measuring the third register Observe the third register, and obtain value $u = g^{k_a} x^{-k_b}$ for some randomly selected (k_a, k_b). The state after measurement is

$$C\sum_{a,b} h(a,b)|a,b,u\rangle,$$

for some scale factor C where

$$h(a,b) = \begin{cases} 1 & \text{if } g^a x^{-b} = u \\ 0 & \text{otherwise} \end{cases}$$

Step 3. Applying QFT The $|u\rangle$ part of the state will not be used, so we will no longer write it. Since there is a correspondence between a and b as stated in the preliminary fact, we are reduced to the superposition state

$$\frac{1}{\sqrt{p-1}}\sum_{b=0}^{p-2} |br+k \bmod (p-1), b\rangle$$

As before, apply the quantum Fourier transform to send $|a\rangle \mapsto |c\rangle$ and $|b\rangle \mapsto |d\rangle$.

$$QFT : \frac{1}{\sqrt{p-1}} \sum_{b=0}^{p-2} |br + k \bmod (p-1), b\rangle \mapsto \sum_{c,d} H(c,d)|c,d\rangle.$$

In the easy case where $p-1$ is smooth, we carry out the QFT over $\mathbb{Z}_{p-1}$.[3] However, in the general (hard) case, we need to find a smooth q such that $p < q < 2p$, and perform our transform in $\mathbb{Z}_q$.

Step 4. Extracting the DL Finally, we observe the state of the quantum computer, extract r from the pair c, d, and repeat the algorithm if necessary.

In the easy case, extracting r is easy: our probability will be non-zero if and only if $rc + d \equiv 0 \pmod{p-1}$, i.e.,

$$r \equiv -\frac{d}{c} \pmod{p-1}$$

If c and $p-1$ are relatively prime, we can find r by division. Because we are choosing among all possible c's with equal probability, the chance that c and $p-1$ are relatively prime is $\phi(p-1)/(p-1)$, which is greater than $1/\log(p)$. Thus by repeating the algorithm, we will obtain r with high probability.

In the hard case, we are able to obtain c, d such that

$$\left| \frac{d}{q} + r \frac{c(p-1) - [c(p-1) \bmod q]}{(p-1)q} \right| \leq \frac{1}{2q}$$

with relatively high probability. Note q divides $c(p-1) - [c(p-1) \bmod q]$. We need only round d/q off to the nearest multiple of $1/(p-1)$ and divide $\pmod{p-1}$ by the integer

$$c' = \frac{c(p-1) - [c(p-1) \bmod q]}{q}$$

to find a candidate r.

It is shown that the algorithm need only be repeated a polynomial number of rounds to find the correct r.

[3] As Shor notes, there already exist classical polynomial-time algorithms [PH78] for the $p-1$ smooth case.

Chapter 7

Surviving Assumptions

7.1 Error Correcting Codes Assumptions

Decoding an arbitrary linear code is an NP-hard problem [ERvT78]. The McEliece cryptosystem is based on this assumption. The McEliece system uses an easy special case of the NP-hard problem, disguised as a general-looking instance of the problem. The two major drawbacks of the systems are large key size and message expansion.

7.1.1 Introduction to linear codes

7.1 Definition [Sti95] Let k, n be positive integers, $k \leq n$. An $[n, k]$ *linear code*, C, is a k-dimensional subspace of $(\mathbb{Z}_2)^n$, the vector space of all binary n-tuples.

7.2 Definition A *generating matrix* for an $[n, k]$ code, C, is a $k \times n$ binary matrix whose rows form a basis for C.

7.3 Definition Let $x, y \in (\mathbb{Z}_2)^n$, where $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$. The *Hamming distance* is

$$d(x, y) = |\{i : 1 \leq i \leq n, x_i \neq y_i\}|,$$

i.e., the number of coordinates in which x and y differ.

7.4 Definition Let C be an $[n, k]$ code. The minimal *distance* of C is the quantity

$$d(C) = min\{d(x, y) : x, y \in C, x \neq y\}.$$

An $[n, k]$ code with minimal distance d is denoted as an $[n, k, d]$ linear code.

The purpose of an error-correcting code is to correct random errors that occur in the transmission of (binary) data through a noisy channel. Let G be a generating matrix for an $[n, k, d]$ code. Suppose Alice wants to transmit a binary k-tuple x. She encodes x as $y = xG$ and sends y through the channel. Suppose Bob receives the n-tuple r, which may or may not be the same as y. He will decode r using the strategy of *nearest neighbor decoding*. That is, to find the codeword y' that has minimum distance to r. Then he decodes r to y', and, finally, determines the k-tuple x' such that $y' = x'G$. Hopefully $y' = y$, so $x' = x$, meaning any transmission errors have been corrected. If at most $(d - 1)/2$ errors occurred, then nearest neighbor decoding can correct all the errors.

An efficient approach for nearest neighbor decoding is through the use of parity-check matrices. A *parity-check matrix* for an $[n, k, d]$ linear code is an $(n - k) \times n$ binary matrix with the property that, for each vector of the code, the product (mod 2) of the matrix by the vector is zero. This product can actually be computed for any vector and is called the *syndrome*. A syndrome is a column vector with $n - k$ components. A vector is a codeword if and only if its syndrome is zero.

7.5 Definition The syndrome decoding problem
Given a linear code and the syndrome of a vector, find the nearest codeword to the vector.

The syndrome decoding problem is NP-hard [ERvT78]. Fischer and Stern [FS96] constructed an efficient PRBG based on this problem, and proved that the generator is as secure as a hard instance of the problem. NP-hardness ensures that there is no known polynomial-time algorithm for solving the problem in the worst case. However, for many special classes of codes, polynomial-time algorithms are known to exist. One such class of codes, the Goppa codes, are used as the basis of McEliece cryptosystem. For each irreducible polynomial $g(x)$ of degree t over $\mathbb{F}_{2^m}$, there exists a binary Goppa code of length $n = 2^m$ and dimension $k \leq n - mt$ capable of correcting any pattern of t or fewer errors. These codes have efficient decoding algorithms [MS77]. Moreover, they are easy to generate and there are a large number of Goppa codes with the same parameters.

7.1.2 McEliece cryptosystem

The McEliece cryptosystem [McE78] is a public-key cryptosystem based on the intractability of the syndrome decoding problem. The idea is to first select a particular code for which an efficient decoding algorithm is known, and then disguise it as a general-looking linear code. A description of the original code can serve as the private key, while a description of the transformed code serves as the public key. Goppa codes are used in the McEliece cryptosystem.

In the setup, the receiver first chooses an easily solvable Goppa code C with generating matrix G; then transforms this matrix into G', a generating matrix for a seemingly arbitrary linear code. To transmit a k-bit message x, the sender multiplies it with G' and adds a random error vector e to the resulting codeword xG'. The Hamming weight of the error vector is equal to the number of errors the code can correct. Therefore, the legitimate receiver can recover the message by using a decoding algorithm for the original code C.

The McEliece cryptosystem has resisted cryptanalysis to date. Berson [Ber97] showed that the scheme fails under message-resend and related-message attacks. However, such attacks do not break the cryptosystem in the sense that they do not recover the private key.

Compared with other public-key cryptosystems which involve modular exponentiation, the McEliece scheme has the advantage of high-speed encryption and decryption. An implementation of this scheme would be two to three orders of magnitude faster than RSA. In addition, the McEliece scheme employs probabilistic encryption, which is better than other types of deterministic encryption in preventing the elimination of any information leaked through public-key cryptography. However, the McEliece scheme suffers from the drawback that the public key is very large. A (less significant) drawback is that there is message expansion by a factor of n/k. For the recommended parameters $n = 1024, t = 38, k \geq 644$, the public key is about 2^{19} bits in size, while the message expansion factor is about 1.6. For these reasons, the scheme receives little attention in practice. [MvOV96]

7.2 Knapsack Assumption

The subset sum problem is an NP-complete problem [GJ79]. Knapsack cryptosystems are based on the subset sum problem. The Merkle-Hellman knapsack encryption scheme [MH78] was the first public-key encryption scheme invented. Many variations have subsequently been proposed but broken.

7.2.1 Knapsack one-way function

The knapsack problem is to select objects with given weights and profits in such a way that a specified capacity is not exceeded and a specified target profit is attained [Sti95]. A special case of the knapsack problem — the subset-sum problem — forms the basis of knapsack cryptosystems.

7.6 Definition The subset-sum problem
Given positive integers (or weights) $s_1, \dots, s_n$, and T, determine whether there is a subset of the s_i's that sums to T. This is equivalent to determining whether there is a 0-1 vector $x = (x_1, \dots, x_n)$ such that

$$\sum_{j=1}^{n} x_i s_i = T$$

7.7 Definition Given $s_1, \dots, s_n$ and T, the *density* of the subset-sum problem is $\delta = n / \log T$.

If one thinks of T as the capacity of a knapsack and the s_i as the sizes of various items, then the question is whether the knapsack can be filled exactly by some collection of those items.

The subset-sum problem is known to be NP-complete [GJ79]. Among other things, this means that there is no known polynomial-time algorithm that solves the worst case in general. However, the subset-sum problem is easy for special cases such as having "hidden structure" and/or "low density". For instance, if s is chosen to be a superincreasing sequence where

$$s_j > \sum_{i=1}^{j-1} s_i$$

for $2 \leq j \leq n$, then the resulting knapsack problem is easy to solve.

The value of T is compared to each value of s, starting with the largest, s_n. If s_n is larger than T then it is not in the subset; if it is smaller then it is in the subset. Reduce T by s_n if in, and move to the next largest number in the sequence. Repeat until finished. Thus in the same manner as decomposing a binary number, we can trivially solve the problem in time $O(n)$, and the solution x (if it exists) must be unique.

Clearly depending on the choice of s, we may or may not have a one-way function[1]. The strategy therefore is to transform a superincreasing easy knapsack into a non-superincreasing general one, and use this transformation as the trapdoor. The most famous transformation is the modular multiplication used by Merkle and Hellman.

7.2.2 Merkle-Hellman cryptosystem

The Merkle-Hellman knapsack cryptosystem [MH78] is a public-key cryptosystem based on the intractability of the subset sum problem. Like the McEliece system, the Merkle-Hellman knapsack system uses an easy special case of the NP-hard problem,

[1] The conjecture that the subset-sum problem is one-way is based on the failure of known algorithm to handle random "high density" instances [Gol98].

disguised as a general instance of the problem. It uses a superincreasing sequence as an easy knapsack, and disguises it with modular multiplication. The original knapsack set can serve as the private key, while the transformed knapsack set serves as the public key.

In the setup, the receiver first chooses a prime p exceeding the sum of all s_i, as well as a multiplier a. Then he disguises his superincreasing sequence into a seemingly arbitrary one by a modular transformation

$$t_i = as_i \bmod p.$$

t will appear to have no special structure. Any solution $(x_1, \ldots, x_n)$ to the knapsack problem $\sum_{i=1}^{n} x_i t_i = y$ is also a solution to the knapsack problem $\sum_{i=1}^{n} x_i s_i = z$, where $z = a^{-1}y \bmod p$. The public key is t, the private key is (p, a, s).

The system can be made even stronger, by iterating this process of hiding structure. In the multiple-iterated version of Merkle-Hellman scheme, the easy superincreasing sequence is disguised by a series of modular multiplications.

7.2.3 Attacks on knapsack systems

The basic Merkle-Hellman scheme was broken by Shamir [Sha84]. The attack relies on the fact that the modular multiplication method does not disguise completely the easy knapsack that is the basis of the construction. Since multiplying each of the weights by $a \bmod p$ produces the same knapsack problem, the goal is to find two values a', p' that transforms the non-superincreasing problem into one that is. By using an integer programming algorithm of Lenstra, Shamir was able to narrow down the search and discover Bob's trapdoor (or an equivalent trapdoor). Attacks have been developed for more general knapsack systems, and now any system which uses modular multiplication to disguise an easy-to-solve knapsack can be broken efficiently.

In addition to the attacks on specific knapsack systems, the most powerful general attack known uses the L^3-lattice basis reduction algorithm [LLL82]. It is typically successful on so-called low-density knapsacks, namely those in which the weights s_i

are large. There is a basic link between lattices and knapsacks. Solving the subset sum problem amounts to find a $0,1$-solution of an inhomogeneous linear equation, which can be naturally viewed as a closest vector problem (see section 7.3.2). A deterministic polynomial-time reduction from the subset sum problem to the closest vector problem is given in [Mic01]. In the case of low-density knapsacks, one can derive from this reduction a provable method to solved the problem in polynomial-time with high probability [CJL$^+$92].

Basically all knapsack cryptosystems have been broken, either by specific (often lattice-based) attacks or by the low-density attacks. The last significant candidate that had resisted such attacks was the Chor-Rivest knapsack cryptosystem [CR88]. The Chor-Rivest cryptosystem involves a combination of number theory ideas and knapsacks. It is the only known knapsack public-key encryption scheme that does not use some form of modular multiplication to disguise an easy subset sum problem. The Chor-Rivest cryptosystem was broken by Vaudenay [Vau98] with algebraic (not lattice) methods.

7.3 Lattice Assumptions

Lattices have been widely used in cryptology, and they have found application both in cryptanalysis and cryptography: lattice algorithms can be used to break cryptographic protocols, while hard lattice problems can be used to design secure cryptosystems.

7.3.1 introduction to lattices

7.8 Definition Let $B = \{b_1, b_2, \dots, b_m\}$ be a set of linearly independent vectors in $\mathbb{R}^n$ (so that $m \leq n$). The set L of all integer linear combinations of $b_1, b_2, \dots, b_m$ is called a *lattice* of *dimension* m; that is, $L = \mathbb{Z}b_1 + \mathbb{Z}b_2 + \cdots + \mathbb{Z}b_m$. The set B is called a *basis* for the lattice L.

Lattices are regular arrangements of points in n-dimensional space. A lattice can have infinitely many bases when $m \geq 2$. A basis consisting of vectors of relatively small lengths is called *reduced*. Since a lattice is discrete, it has a shortest nonzero vector. The goal of lattice reduction is to find interesting lattice bases, such as bases consisting of reasonably short and almost orthogonal vectors.

Lattice reduction techniques have been a powerful tool in cryptanalysis. For example, the famous L^3-lattice basis reduction algorithm [LLL82] has played a crucial role in cryptanalyzing factoring and the subset-sum based public-key schemes [Sch91, CJL$^+$92].

7.3.2 Lattice problems

The most famous lattice problem is the shortest vector problem.

7.9 Definition The shortest vector problem (SVP)
Given a basis of a lattice L, find the shortest nonzero vector $v \in L$.

7.10 Definition The closest vector problem (CVP)
Given a basis of a lattice L and a vector $v \in \mathbb{R}^n$, find a lattice vector minimizing the distance to v.

7.11 Definition The smallest basis problem (SBP)
Find a lattice basis minimizing the maximum of the lengths of its elements.

All these problems are NP-hard, and no polynomial-time algorithm is known for approximating either SVP, CVP, or SBP in $\mathbb{Z}^n$ to within a polynomial factor in n [Ngu99]. The best polynomial-time approximation algorithms are based on the L^3 algorithm, and achieve exponential bounds.

7.3.3 Lattice-based cryptosystems

Two lattice-based public-key cryptosystems have received wide attention: the Ajtai-Dwork (AD) cryptosystem and the Goldwasser-Goldreich-Halevi (GGH) cryptosystem.

The Ajtai-Dwork cryptosystem

The Ajtai-Dwork cryptosystem [AD97] works in $\mathbb{R}^n$. Its security is based on a variant of the SVP. The drawbacks of the system are huge public-key size and message expansion. Nguyen and stern [NS98] showed that any realistic implementation of the AD scheme was insecure. Therefore unless major improvements are found, the scheme is only of theoretical importance.

The Goldwasser-Goldreich-Halevi cryptosystem

The Goldwasser-Goldreich-Halevi cryptosystem [GGH96] is based on the CVP. It is the lattice analog of the McEliece cryptosystem. In both schemes, a ciphertext is the addition of a random noise vector to a vector corresponding to the plaintext. The public key and the private key are different representations of the same object (a lattice for GGH, and a linear code for McEliece). The private key has a particular structure allowing to cancel noise vectors up to a certain bound.

The GGH cryptosystem is much more efficient than the AD cryptosystem. It was attacked by Nguyen [Ngu99], who showed that decryption can be reduced to solving CVP instances that are much easier than the general problem.

7.4 Polynomials

Patarin [Pat96] proposed two public-key schemes based on multivariate polynomials, Hidden Field Equations (HFE) and Isomorphism of Polynomials (IP). HFE is based on the MQ problem and HFE trapdoor OWF. IP is a generalization of the Graph Isomorphism (GI).

7.4.1 Hidden Field Equations

Hidden Field Equations (HFE) is a public-key cryptosystem using polynomial operations over finite fields. It uses system of several modular equations in several variables, which could be considered as natural generalization of RSA scheme (one modular equation in one variable). HFE has two levels of security: it is based on NP-complete problem Multivariate Quadratic and it hides algebraic structure used to generate its keys.

The MQ problem

The general problem of solving quadratic equations is known as the MQ problem, and proved NP-complete [PGC$^+$, GJ79].

7.12 Definition Multivariate Quadratic problem over a ring K (MQ)
Given a system of m quadratic equations with n variables in K. Find (one) solution of the system.

7.13 Fact Every function $f : K^n \to K^n$ can be written as :

- a univariate polynomial $b = f(a)$, or
- n multivariate polynomials with n variables over K

$$f : \{b_i = f_i(a_1, \dots, a_n)\}_{i=1..n}.$$

Pell's equation is a special case of the MQ problem.

7.14 Definition Pell's equation
Given a positive non-square integer d, Pell's equation is $x^2 - dy^2 = 1$ and the goal is to find its integer solutions.

Pell's equation is one of the oldest studied problems in number theory. The best algorithms known for solving Pell's equation have superpolynomial time complexity [Len02]. Recently Sean Hallgren gave a quantum algorithm that runs in polynomial time [Hal01].

The HFE problem

The principle of HFE is the following:

1. It is possible to find a solution of a univariate polynomial over a big finite field provided the degree of the polynomial is not too big.

2. For some polynomial functions it is possible to represent them as n quadratic equations with n variables which hide the polynomial structure and makes it look quite as (almost) any other polynomial of any degree. In addition we make an initial and final affine variable changes. This idea of "disguising" is also familiar to us from the Merkle-Hellman knapsacks and McEliece cryptosystem.

7.15 Fact [Cou01] Let f be a polynomial over $\mathbb{F}_{q^n}$ of the special form:

$$f(a) = \sum_i \alpha_i \cdot a^{q^{s_i}+q^{t_i}}$$

Then f can be written as n multivariate quadratic equations over the $a_i \in \mathbb{F}_q$.

7.16 Fact (HFE trapdoor) [Cou01]
If f is a polynomial of degree at most d then $f^{-1}(\{b\})$ can be computed in time $d^2(\ln d)^{O(1)}n^2$ $\mathbb{F}_q$ operations.

7.17 Definition HFE problem [Cou01]
Let S and T be two random secret bijective and affine multivariate variable changes. Let $g = T \circ f \circ S$. Given the multivariate representation g of f and a random b. Find a solution a such that $g(a) = b$.

It is known that the MQ problem is NP-complete for any field K. Computing g^{-1} without knowledge of f requires exponential number of operations with the best known algorithm.

In the HFE public-key scheme, the private-key is S, T, and f, and the public-key is n quadratic polynomials

$$g : \{y_i = g_i(x_1, \dots, x_n)\}_{i=1..n}.$$

g looks like system of random quadratic equations. The idea is to hide the univariate representation with invertible affine variable changes S and T. Since f is bounded degree and univariate, we can invert it.

There are several modifications of the basic HFE: HFE$^-$ (Asiacrypt'98), HFEv (Eurocrypt'99), Quartz (RSA'2000), Flash (RSA'2000), etc. The basic HFE conceals algebraic structure with invertible affine variable changes. Variants of HFE destroys the algebraic structure by non-invertible operations. Attacks on the basic HFE are superpolynomial in general and polynomial when d is fixed, while HFE modifications have resisted all known attack [Cou01].

7.4.2 Isomorphism of Polynomials

7.18 Definition Isomorphism of Polynomials
Given two sets of u multivariate polynomials with n variables over a finite field $\mathbb{F}_q$, $\mathcal{A}$ and $\mathcal{B}$. $\mathcal{A}$ and $\mathcal{B}$ are *isomorphic* if there exist two affine bijections S and T such that

$$\mathcal{B} = T \circ \mathcal{A} \circ S.$$

7.19 Definition The Isomorphism of Polynomials problem (IP)
Given $\mathcal{A}$ and $\mathcal{B}$, find S and T such that $\mathcal{B} = T \circ \mathcal{A} \circ S$.

When S and T are not necessary bijective, we call the corresponding problem the *Morphism of Polynomials problem* (MP). IP is also a generalization of the Graph

Isomorphism (GI). Its difficulty lies in between GI (easy but not polynomial) and MP (hard)[PGC98].

Although we are not able to prove the security of HFE or IP, it has long been regarded as the most promising cryptosystem of the kind. HFE can be used to do signatures, encryption or authentication. HFE is considered stronger and faster than RSA. It generate very short signatures (e.g., 128 bits), and short encryptions of short messages (e.g., 128-bit blocks). IP can be used for signatures and zero-knowledge authentication.

7.5 Combinatorial group theory

Another approach is to use hard problems in combinatorial group theory. Combinatorial group theory (CGT) is the study of groups by means of generators and defining relators. Cryptosystems based on CGT use topological, combinatorial and group theoretical problems that are intractable according to our current mathematical knowledge.

Informally, a finitely represented group G is specified by a finite set of generators and relators. Generators are just abstract symbols. A word w in G is a finite string made up of generators and their inverses. The inverse word w^{-1} consists of all the inverses of symbols of w written in reverse order. The group G consists of equivalence classes of all possible words. Two words are equivalent in G if we can transform one to the other by a finite sequence of replacement rules.

The fundamental questions of CGT include the word problem (WP), the conjugacy problem (CP), and the isomorphism problem (IsoP) as defined and investigated by M. Dehn in the period 1910-1912. The word problem and the conjugacy problem are defined as follows:

7.20 Definition The word problem (WP)
Given a group G and words x, y, determine whether x is equivalent to y in G.

7.21 Definition The conjugacy problem (CP)
Given two arbitrary words $x, y \in G$, determine whether $y = wxw^{-1}$ for some $w \in G$.

An n-braid is a collection of n strands connecting n fixed points at the top with another set of n fixed points at the bottom. The ends of the strands are regularly spaced at the top and the bottom of the braid. Each strand must always hang downwards from the top of the braid to the bottom, never looping up or knotting itself.

Two braids are considered equal if one can be deformed to the other continuously in the set of braids. Two braids on the same number of strands can be concatenated to form their product by positioning one on the top of the other.

The identity element of the group is the braid in which the strands are straight lines which do not intertwine at all. The inverse of a braid is its reflection in a mirror perpendicular to the preferred direction.

This multiplication gives the set of n-strand braids the structure of a group, called B_n, which is non-commutative if $n > 2$. B_n is defined by generators $\sigma_1, \dots, \sigma_{n-1}$.

The word problem of braid groups is easy, but the generalized conjugacy search problem (GCSP) of braid groups is believed to be difficult.

7.22 Definition The conjugacy search problem in B_n
Given two braids $\alpha, \alpha' \in B_n$, determine whether $\alpha = \gamma\alpha'\gamma^{-1}$ for some $\gamma \in B_n$.

Ko et al. [KLC$^+$00] proposed a generalized version of the CP:

7.23 Definition The generalized conjugacy search problem in B_n (GCSP)
Given two braids $\alpha, \alpha' \in B_n$, determine whether $\alpha = \gamma\alpha'\gamma^{-1}$ for some $\gamma \in LB_m$, where $m \leq n$. LB_m denotes the subgroup of B_n generated by $\sigma_1, \dots, \sigma_{m-1}$.

Ko et al. constructed a trapdoor one-way function based on the GCSP in the braid groups.

$$f : LB_l \times B_{l+r} \to B_{l+r} \times B_{l+r}, \quad f(a, x) = (axa^{-1}, x),$$

where LB_l is the subgroup of B_{l+r} generated by $\sigma_1, \ldots, \sigma_{l-1}$. It is a one-way function because given a pair (a, x), it is easy to compute axa^{-1} but all the known attacks need exponential time to compute a from (axa^{-1}, x). Ko et al. also proposed a key agreement scheme and a public-key cryptosystem using the trapdoor one-way function.

7.6 Subset-product

The subset-product problem is to determine, given positive integers (or weights) $s_1, \ldots, s_n$, and K, whether there is a subset of the s_i's such that the product (i.e. result of multiplying together all in the subset) is equal to K.

Recall that in the subset-sum problem a super-increasing sequence is transformed to a seemingly intractable problem. Similarly, in the subset-product problem, the prime factorization serves as the trapdoor. However, almost all transformation tricks from a trapdoor subset-product to another subset-product have been cryptanalyzed due to their linearity and low density [OTU00].

Okamoto et al. [OTU00] proposed a quantum public-key scheme based on the subset-product problem. Their paper presents a new paradigm of cryptography, *quantum public-key cryptosystems.* The security of the scheme is based on the assumption that the subset-product problem is intractable in such cryptosystems.

The basic idea by Okamoto et al. is to employ a non-linear transformation, exponentiation (and logarithm), given that computing a logarithm is feasible. The scheme employs a ring of integers of an algebraic number field, which is randomly selected from exponentially many candidates. In this scheme, the underlying quantum (not classical) mechanism is only Shor's discrete logarithm algorithm, which is employed in the key generation stage. (i.e., off-line stage). Encryption and decryption (i.e., on-line stage) require only classical computations.

7.7 Number field

The class group of algebraic number fields is of cryptographic interest.

The DLP in class groups of imaginary number fields is at least as difficult as factoring the corresponding discriminant. Furthermore, the best known algorithms for solving the DLP in class groups of higher degree require the solution an index calculus problem and many SVP in lattices.

Buchmann and Paulus [BP97] proposed a new one-way function based on the difficulty of finding shortest vectors in lattices. This new function consists of composition of ideals in subrings of quadratic number fields and multiplication by algebraic numbers which can both be performed in polynomial time. The only method currently known for inverting it requires computing shortest vectors in lattices whose dimension is the degree of the number field. The best known algorithm [Kan87] for this requires exponential time in the degree of the number field. Buchmann and Paulus stated that inverting the function is at least as hard as factoring integers. Further analysis is needed to evaluate the security of their proposal.

Buchmann and Williams [BW89] presented a new Diffie-Hellman key exchange system based on real quadratic fields. Sean Hallgren [Hal01] claims that his quantum algorithm for the principal ideal problem breaks this system.

Chapter 8

Conclusion

In the evaluation of cryptographic schemes, we are concerned with both the computational efficiency and the infeasibility of violating the scheme.

Since the discovery of public-key cryptography by Diffie and Hellman [DH76], very few proposals have withstood cryptanalysis. The two most successful groups are factoring or discrete logarithm based schemes. In the thesis, we first discuss three problems that are reducible to factoring — the RSA problem, the Rabin problem, and the quadratic residuosity problem, which respectively underlie the RSA, Rabin, and Goldwasser-Micali cryptosystems. We also looked into the discrete logarithm problem and its generalized version. The discrete logarithm problem forms the basis of schemes such as Diffie-Hellman key exchange and ElGamal encryption. The ECDLP is conjectured to be harder than the DLP and the factoring problem, and no significant weakness have been reported.

Shor's quantum algorithm poses the most worrisome long-term threat to RSA and discrete logarithm cryptosystems. Taking advantage of quantum parallelism, appropriate quantum algorithms can achieve exponential speedup for certain problems, as is the case of Shor's. This threat motivates people to seek alternative public-key cryptosystems. The security of these systems are base on assumed difficulties of other hard problems. Many of such problems are not well-known. The goal of this thesis was then to survey the various underlying computational assumptions, in hope to inspire new cryptosystems and quantum algorithms.

Our attention was devoted to the following hard problems:

Error-correcting codes : syndrome decoding problem

Subset-sum (knapsack) : subset-sum problem

Subset-product : subset-product problem

Lattice : SVP, CVP, SBP

Polynomials : MQ, HFE, IP

Combinatorial group theory : CP, GCSP

Number fields : DLP, SVP

With academic advances in cryptography, new schemes will continue to be proposed and tested. Among the examples presented here, many are to be furtherly evaluated in future research. We will be watching with great interest the future development of cryptography as well as that of quantum computers.

Bibliography

[ACGS88] Werner Alexi, Benny Chor, Oded Goldreich, and Claus-P. Schnorr. RSA and Rabin functions: certain parts are as hard as the whole. *SIAM Journal on Computing*, 17(2):194–209, April 1988.

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 284–293, New York, May 1997. ACM Special Interest Group for Automata and Computability Theory, ACM Press.

[BBS86] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

[BCG89] M. Bellare, L. Cowen, and S. Goldwasser. Secret key exchange. DIMACS Workshop on Distributed Computing and Cryptography, 1989.

[BD98] Dan Boneh and Glenn Durfee. New results on the cryptanalysis of low exponent RSA (extend abstract, preprint), 1998.

[Ben80] P. A. Benioff. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics*, 22:563–591, 1980.

[Ber97] T. A. Berson. Failure of the McEliece public-key cryptosystem under message-resend and related-message attack. In *Advances in Cryptology: CRYPTO '97: Proceedings* [Int97], pages 213–220.

[BG85] M. Blum and S. Goldwasser. An efficient probabilistic public-key encryption scheme which hides all partial information. In *Advances in Cryptology: CRYPTO '84: Proceedings*, number 196 in Lecture Notes in Computer Science, pages 289–299, Berlin, August 1985. International Association for Cryptologic Research, Springer-Verlag.

[BL95] D. Boneh and R. J. Lipton. Quantum cryptanalysis of hidden linear functions. In *Advances in Cryptology: CRYPTO '95: Proceedings*, number 963 in Lecture Notes in Computer Science, pages 424–437, Berlin, August 1995. International Association for Cryptologic Research, Springer-Verlag.

[BP97] J. Buchmann and S. Paulus. A one way function based on ideal arithmetic in number fields. In *Advances in Cryptology: CRYPTO '97: Proceedings* [Int97], pages 385–394.

[BW89] Johannes Buchmann and Hugh C. Williams. A key exchange system based on real quadratic fields. In *Advances in Cryptology: CRYPTO '89: Proceedings* [Int89], pages 335–343.

[CJL+92] Matthijs J. Coster, Antoine Joux, Brian A. LaMacchia, Andrew M. Odlyzko, Claus-Peter Schnorr, and Jacques Stern. Improved low-density subset sum algorithms. *Computational Complexity*, 2:111–128, 1992.

[Cop94] Coppersmith. An approximate Fourier transform useful in quantum factoring. Research Report RC 19642, IBM, 1994.

[Cou01] Nicolas Courtois. The security of hidden field equations (HFE). Cryptographers' Track RSA Conference 2001, 2001.

[CR88] B. Chor and R. L. Rivest. A knapsack type public-key cryptosystem based on arithmetic in finite fields. *IEEE Trans. Inform. Theory*, 34(5):901–909, September 1988.

[Den82] Dorothy Elizabeth Robling Denning. *Cryptography and data security.* Addison-Wesley, Reading, MA, USA, 1982.

[Deu85] David Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London Ser. A*, A400:97–117, 1985.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Trans. Inform. Theory*, IT-22:644–654, November 1976.

[DN00] Glenn Durfee and Phong Nguyen. Cryptanalysis of the RSA schemes with short secret exponent from Asiacrypt '99. In *Advances in Cryptology: ASIACRYPT '00: Proceedings*, number 1976 in Lecture Notes in Computer Science, pages 14–29, Kyoto, December 2000. International Conference on the Theory and Application of Cryptology, Springer-Verlag, 2000.

[Dum99] Paul Dumais. Zero-knowledge et infomatique quantique: approche de la théorie de la complexité. `http://www.iro.umontreal.ca/ dumais/divers/predoc.ps`, 1999.

[EJ96] A Ekert and R Jozsa. Shor's quantum algorithm for factorizing numbers. *Reviews of Modern Physics*, pages 733–753, July 1996.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[ERvT78] E.R.Berlekamp, R.J.McEliece, and H.C.A. van Tilborg. On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, 24:384–386, 1978.

[Fey82] Richard Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6&7):467–488, 1982.

[Fey86] R. P. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16(6):507–531, 1986.

[FS96] J. B. Fischer and J. Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In *Advances in Cryptology: EUROCRYPT '96: Proceedings* [Int96], pages 245–255.

[GGH96] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(056), 1996.

[GJ79] M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.

[Gol98] Oded Goldreich. Foundations of cryptography: Fragments of a book. `http://theory.lcs.mit.edu/~oded/frag.html`, February 1998.

[Gol99] Oded Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. Number 17 in Algorithms and Combinatorics. Springer-Verlag, Berlin, 1999.

[Gro96] Lov. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, pages 212–219, New York, May 1996. ACM Special Interest Group for Algorithms and Computation Theory, ACM Press.

[Gru99] Jozef Gruska. *Quantum Computing.* McGraw-Hill, Maidenhead, UK, 1999.

[Hal01] Sean Hallgren. Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. `http://www.cs.caltech.edu/%7Ehallgren/pell.ps`, 2001. Submitted.

[IL89] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography. In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235, Los Alamitos, CA, October 1989. IEEE Computer Society, IEEE Computer Society Press.

[Int85] International Association for Cryptologic Research. *Advances in Cryptology: CRYPTO '85: Proceedings*, number 218 in Lecture Notes in Computer Science, Berlin, August 1985. Springer-Verlag.

[Int89] International Association for Cryptologic Research. *Advances in Cryptology: CRYPTO '89: Proceedings*, number 435 in Lecture Notes in Computer Science, Berlin, August 1989. Springer-Verlag.

[Int96] International Association for Cryptologic Research. *Advances in Cryptology: EUROCRYPT '96: Proceedings*, number 1070 in Lecture Notes in Computer Science, Berlin, May 1996. Springer.

[Int97] International Association for Cryptologic Research. *Advances in Cryptology: CRYPTO '97: Proceedings*, number 1294 in Lecture Notes in Computer Science, Berlin, August 1997. Springer-Verlag.

[Int98] International Association for Cryptologic Research. *Advances in Cryptology: CRYPTO '98: Proceedings*, number 1462 in Lecture Notes in Computer Science, Berlin, August 1998. Springer-Verlag.

[Int00] International Association for Cryptologic Research. *Advances in Cryptology: CRYPTO '00: Proceedings*, number 1880 in Lecture Notes in Computer Science, Berlin, August 2000. Springer.

[Kan87] R. Kannan. Minkowski's convex body theorem and integer programming. *Math. Oper. Res.*, 12:415–440, 1987.

[KLC+00] K.H. Ko, S.J. Lee, J.H. Cheon, J.W. Han, J.S. Kang, and C.S. Park. New public-key cryptosystem using braid groups. In *Advances in Cryptology: CRYPTO '00: Proceedings* [Int00], pages 166–183.

[Knu81] Donald E. Knuth. *The Art of Computer Programming*, volume 2: Seminumerical Algorithms. Addison-Wesley, 1981.

[Kob87] N. Koblitz. Elliptic curve cryptosytems. *Mathematics of Computation*, 48(177):203–209, 1987.

[Lab00] RSA Laboratories. *RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1*, 2000.

[Len02] H.W. Lenstra Jr. Solving the Pell equation. *Notices of the American Mathematical Society*, 49(2), February 2002.

[LLL82] A. K. Lenstra, H. W. Lenstra Jr., and L. Lovasz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:514–534, 1982.

[Lub96] Michael Luby. *Pseudorandomness and Cryptographic Applications.* Princeton Computer Science Notes. Princeton University Press, Princeton, NJ, 1996.

[McE78] R. J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.

[MH78] R. Merkle and M. Hellman. Hiding information and signatures in trapdoor knapsacks. *IEEE Trans. Inform. Theory*, IT-24:525–530, September 1978.

[Mic01] D. Micciancio. The hardness of the closest vector problem with preprocessing, 2001.

[Mil85] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology: CRYPTO '85: Proceedings* [Int85], pages 417–426.

[MS77] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland, Amsterdam, Holland, 1977.

[MS91] S. Micali and C. P. Schnorr. Efficient, perfect polynomial random number generators. *Journal of Cryptology*, 3(3):157–172, 1991.

[MVO91] A. Menezes, S. Vanstone, and T. Okamoto. Reducing elliptic curve logarithms to logarithms in a finite field. In *Proc. 23rd Ann. ACM Symp. on Theory of Computing*, pages 80–89, New Orleans, LA, 1991.

[MvOV96] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.

[Ngu99] Phong Nguyen. Cryptanalysis of the goldreich-goldwasser-halevi cryptosystem from crypto '97. In *Advances in Cryptology: CRYPTO '99: Proceedings*, number 1666 in Lecture Notes in Computer Science, pages 288–304, Berlin, August 1999. International Association for Cryptologic Research, Springer.

[NS98] Phong Nguyen and Jacques Stern. Cryptanalysis of the Ajtai-Dwork cryptosystem. In *Advances in Cryptology: CRYPTO '98: Proceedings* [Int98], pages 223–242.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 33–43, New York, May 1989. ACM Special Interest Group for Automata and Computability Theory, ACM Press.

[OTU00] Tatsuaki Okamoto, Keisuke Tanaka, and Shigenori Uchiyama. Quantum public-key cryptosystems. In *Advances in Cryptology: CRYPTO '00: Proceedings* [Int00], pages 147–165.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way functions are essential for non-trivial zero-knowledge. In *2nd Israel Symposium on Theory of Computing and Systems*, pages 3–17. IEEE Comp. Soc. Press, 1993.

[Pat96] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In *Advances in Cryptology: EUROCRYPT '96: Proceedings* [Int96], pages 33–48.

[PGC$^+$] Jacques Patarin, Louis Goubin, Nicolas Courtois, + papers of Eli Biham, Aviad Kipnis, T. T. Moh, and et al. Asymmetric cryptography with multivariate polynomials over a small finite field. Available from J.Patarin@frlv.bull.fr.

[PGC98] Jacques Patarin, Louis Goubin, and Nicolas Courtois. Improved algorithms for isomorphism of polynomials. In *Advances in Cryptology: EUROCRYPT '98: Proceedings*, number 1403 in Lecture Notes in Computer Science, pages 184–200, Berlin, May 1998. International Association for Cryptologic Research, Springer-Verlag.

[PH78] Stephen Pohlig and Martin Hellman. An improved algorithm for computing discrete logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.

[Pre97] J. Preskill. Quantum computing: Pro and con. `http://xxx.lanl.gov/abs/quant-ph/9705032`, 1997.

[Rab79] Michael Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report MIT/LCS/TR-212, Laboratory for Computer Science, Massachusetts Institute of Technology, January 1979.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 387–394, New York, May 1990. ACM Special Interest Group for Automata and Computability Theory, ACM Press.

[RP98] Eleanor Rieffel and Wolfgang Polak. An introduction to quantum computing for non-physicists, 1998.

[RSA78] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21:120–126, February 1978.

[Sch89] C. P. Schnorr. Efficient identification and signatures for smart cards. In *Advances in Cryptology: CRYPTO '89: Proceedings* [Int89], pages 239–252.

[Sch91] C. P. Schnorr. Factoring integers and computing discrete logarithms via diophantine approximation. In *Advances in Cryptology: EUROCRYPT '91: Proceedings*, number 576 in Lecture Notes in Computer Science, pages 171–181, Berlin, May 1991. International Association for Cryptologic Research, Springer-Verlag, 1992.

[Sha84] A. Shamir. A polynomial time algorithm for breaking the basic Merkle-Hellman cryptosystem. *IEEE Transactions on Information Theory*, IT-30(5):699–704, September 1984.

[Sho94] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science: Proceedings*, pages 124–134, Santa Fe, NM, November 1994. IEEE Computer Society, IEEE Computer Society Press.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, October 1997.

[SR00] Andrew M. Steane and Eleanor G. Rieffel. Beyond bits: The future of quantum information processing. *Computer*, 33(1):38–45, January 2000.

[ST92] J.H. Silverman and J. Tate. *Rational Points on Elliptic Curves*. Springer-Verlag, 1992.

[Sti95] Douglas R. Stinson. *Cryptography, Theory and Practice*. CRC Press, Boca Raton, FL, 1995.

[SW95] Renate Scheidler and Hugh C. Williams. A public-key cryptosystem utilizing cyclotomic fields. *Designs, Codes and Cryptography*, 6(2):117–131, 1995.

[Vau98] Serge Vaudenay. Cryptanalysis of the Chor-Rivest cryptosystem. In *Advances in Cryptology: CRYPTO '98: Proceedings* [Int98], pages 243–256.

[Wie90] Michael J. Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory*, 36(3):553–558, 1990.

[Wie98] Michael J. Wiener. Performance comparison of public-key cryptosystems. *CryptoBytes*, 4(1):1, 3–5, Summer 1998.

[Wil80] H. C. Williams. A modification of the RSA public-key encryption procedure. *IEEE Transactions on Information Theory*, IT-26(6):726–729, 1980.

[Wil85] H. C. Williams. An M^3 public-key encryption scheme. In *Advances in Cryptology: CRYPTO '85: Proceedings* [Int85], pages 358–368.