Chapter 1

Introduction

1.1 Electronic Commerce

As we enter the new millennium, the Internet has played a very important role in our everyday life. Internet is being used almost everywhere including schools, homes, workplaces and in our leisure activities. One of the most important applications of the Internet is in marketplace and commerce.

Electronic commerce applications started in the early 1970's. It acted as the electronic funds transfer (EFT) which was used in large corporation financial institutions to transfer the transaction information via the network. Another form of electronic commerce is the electronic data interchange (EDI). It allows the standard format computer readable data exchange between corporations. This application expanded the transactions from financial to processing and the participating companies from financial institution to manufacturers, service parties and retailers. In the early 1990's, with the commercialization of the internet and its rapid growth to millions of customers, the Electronic Commerce applications expanded rapidly to everything and everywhere. This covers not only the commercial activity in the
physical marketplace, but opened new forms of commercial activity which is not possible in physical world.

The major electronic applications [1] include retailing (direct marketing, online customer service, electronic intermediaries, electronic department store etc), service industry (travel and tourism service, employment placement and job market, real estate, trading stock on line, cyberbanking and personal finance, auction, online publishing), business to business (supplier oriented, buyer oriented, intermediary oriented, just in time delivery), advertisement, and electronic payment system.

1.2 The Cornerstone of Security

The Internet network system is an Open Systems. This means the system is open for communication with other systems. There are impostors, interceptors, eavesdroppers, intruders and malicious parties who might intercept, modify and tamper with the messages between the receiver and the sender. So in the open network environment, security is the major issue.

The National Computer Security Association (NCSA) in the United States [1] has identified four cornerstones of the secure electronic commerce. It includes: Authenticity, Privacy, Integrity, and Non-repudiation.

- Authenticity

Authenticity provides the mechanism in which the authorities of the message can prove itself. There are various authentication schemes, each providing different levels of security. These schemes include: password-based authentication, digital signatures, use of certificate, Kerberos and so on.

- Privacy
Privacy means that the content of a message is secret and only known to the authorized parties. The main methods of achieving privacy is by encrypting it using various cryptographic techniques, which include symmetric key encryption and asymmetric encryption.

- **Integrity**

  Integrity provides an assurance that the content of information has not been modified in any way. One method of ensuring the integrity of a message is an authentication code, another one is the hash function.

- **Non repudiation**

  Non repudiation means the owner or the sender of the message cannot deny that he has sent the message. The digital signature is the most important scheme as it is hard to dispute whether you were involved or not.

### 1.3 The Scope of the Research

The contribution of this dissertation lies in the area of new concepts, new methods and more efficient solutions for securing electronic commerce services.

Our contribution includes:

- **$m$ out of $n$ oblivious transfer.**

  The Oblivious Transfer notation was developed in 1982 [2] with the implementation based on the RSA public key cryptographic scheme. After that, a series of related concepts and its implementation were given, such as one out of two bits oblivious transfer, one out of two bits chosen oblivious transfer, one out of two strings oblivious transfer, one out of two strings chosen oblivious transfer, one out of $n$ oblivious transfer, quantum oblivious transfer,
CHAPTER 1. INTRODUCTION

$p$-oblivious transfer and so on. At the same time, it also became a very useful primitive in modern cryptography. It is widely used in electronic commerce, mobile phone security etc. to protect user’s privacy. However, there is still no general notation and method to generalize this implementation method. In this dissertation, we present a generalized $m$ out of $n$ oblivious transfer notion. A general implementation, based on public key system and symmetric key system is present. As this scheme has a drawback of computational complex, a more efficient method is developed, which is based on the discrete log problem. We also compare our scheme with other schemes, and other schemes can be seen as a special case of our scheme. As one application of this $m$ out of $n$ OT protocol scheme, a more efficient blind signature is given.

- **A new Non-interactive method for $m$ out of $n$ oblivious transfer.**

  The concept of non-interactive oblivious transfer was given in 1989 [3]. In the same paper, the non-interactive oblivious transfer protocol was presented and the implementation was based on the Diffie-Hellman problem. However, this non interactive scheme was only used for one out of two oblivious transfer protocol. Then he extended it to two out of three and limited to $t - 1$ out of $t$ oblivious transfer. In this dissertation, an $m$ out of $n$ noninteractive oblivious transfer protocol is developed which is based on the discrete log problem. We also present the public key generation algorithm for this scheme and analyze its security. A more secure scheme which can protect the receiver even though the sender is dishonest is shown. Finally, we discuss the application of the scheme.

- **Maintaining privacy in buying digital goods**

  There are many ways to protect the user’s privacy in electronic commerce, such as anonymous electronic cash payment systems, which protect people’s
privacy by hiding who the buyer is. An alternative way that we present is
to hide what people buy and even how much they spend and when they buy.
It presents priced oblivious transfer protocol in this way to protect people’s
privacy in [4]. In this dissertation, we discuss this scheme and give an improved
scheme to maintain the privacy in buying and subscribing the digital goods
using the $m$ out of $n$ priced oblivious transfer protocol. We present the $m$
out of $n$ priced oblivious transfer protocol and give a concrete homomorphic
encryption scheme. We also discussed the zero knowledge proof and secret
sharing etc. tools which are used in buying digital goods scheme.

1.4 Outline of Dissertation

This dissertation is organised as follows. Chapter 2 gives the basic notions and
background which are used in modern cryptography and cryptographic techniques.

Chapter 3 introduces the basic protocols such as hash function, public key cryp-
tosystem, digital signature, Identification protocol, zero-knowledge proofs of knowl-
edge, secretes sharing and oblivious transfer protocol, which are used in this disser-
tation.

Chapter 4 discusses the $m$ out of $n$ oblivious transfer protocol and its application
in blind signatures. We define the concept of $m$ out of $n$ oblivious transfer, and give
the implementation of the $m$ out of $n$ oblivious transfer protocol, and then give a
more efficient $m$ out of $n$ oblivious transfer protocol. An application of the $m$ out
of $n$ oblivious transfer protocol is also presented.

Chapter 5 introduces the non-interactive protocol. We give the definition of the
non-interactive $m$ out of $n$ oblivious transfer. Next, we construct the first method
to realize the non-interactive $m$ out of $n$ oblivious transfer protocol and we discuss
the security of the proposed scheme.
CHAPTER 1. INTRODUCTION

In chapter 6, we first introduce the different ways of protecting the user's privacy in electronic commerce services, then we present the basic tools which are used for this chapter. After that, we give the implementation of maintaining privacy protocol in buying and subscribing digital goods. We also give the details of $m$ out of $n$ priced oblivious transfer protocol and the homomorphic encryption protocol, and discuss how to prove the committed number between two boundaries using the zero knowledge proof technology.

In chapter 7, we summarize the contributions in this dissertation and give the direction for future research.
Chapter 2

Basic Notations and Background

In this chapter, we introduce some basic notations and background which are associated with this dissertation. This chapter can be seen as part of an introduction of modern cryptography. Readers who are familiar with these topics can skip this chapter. Some materials in this chapter are borrowed from [5, 6, 7, 1, 8, 9, 10]

2.1 Algorithms and Complexity Theory

2.1.1 Algorithms

In Encyclopedia of Computer science and Engineering, Algorithms are defined as "the precise characterization of a method of solving a problem" and further notes that any algorithm must have the following properties:

- Finiteness. Application of the algorithm to a particular set of data must result in a finite sequence of actions.
- Unique initialization. The action which starts off the algorithm must be unique.
• Unique succession. Each action in the algorithm must be followed by a unique successor action.

• Solution. The algorithm must terminate with a solution to the problem, or it must indicate that for the given data, the problem is insoluble by the algorithm.

For example, the Euclidean algorithm is one of the most important algorithms used in cryptography. It is a method for calculating the greatest common divisor (gcd) of two positive integers, i.e., the largest positive integer that divides both of them.

2.1.2 Complexity of the Algorithm

Complexity of the Algorithm gives us a methodology for analyzing the computational complexity of different cryptographic techniques and algorithms. Information theory tells us that all the cryptographic algorithms can be broken, but the complexity theory can tell us how long it will take to break the algorithm.

The complexity of the algorithm is often measured by two variables: $T$ and $S$. $T$ and $S$ stand for time complexity and space complexity. If the concern is for a fast algorithm, we use the time complexity $T$. If the minimum storage is the object, we use the space complexity $S$. Both of them are the function of the input size $n$. Usually, it is expressed as “big O” notation, i.e., the order of magnitude of the computational complexity.

The algorithms are classified according to the time complexity. When the complexity of the algorithm is $O(n^m)$ ( $m$ is a constant), we call it a polynomial-time algorithm.

• If $m = 0$ it is called as constant algorithm.

• If $m = 1$ it is called as linear algorithm.
2.1. ALGORITHMS AND COMPLEXITY THEORY

- If \( m = 2 \) it is called as quadratic algorithm.

- If \( m = 3 \) it is called as cubic algorithm.

When the complexity of the algorithm is \( O(t^{f(n)}) \), where \( t \) is a constant greater than one and \( f(n) \) is polynomial function of \( n \), we call it an exponential time algorithm.

When \( c \) is constant and the \( f(n) \) is more than constant but less than linear, we call it a super polynomial time algorithm. It is a subset of the exponential time algorithm.

2.1.3 Complexity of the Problem

We sometimes use the concept of the Turing machine. A Turing machine is a finite state machine with an infinite read and write memory tape. We can think of a Turing machine as a realistic model of computation.

If the problem can be solved with a polynomial time algorithm, we call it tractable. If the problem cannot be solved with polynomial time algorithm, we call it in-tractable. Intractable problems are also called hard.

Based on the complexity of the solution, we can divide the problem into complexity classes.

- Class P includes all the problems that can be solved in polynomial time.

- Class NP includes all the problems that can be solved in polynomial time only on a nondeterministic Turing machine.

- Other classes include PSPACE and EXPTIME.
2.2 Interactive and Non-Interactive Protocols

Protocol is a special class of algorithms. The protocol is defined as following [6].

Definition 2.1 A protocol is a series of steps, involving two or more parties, designed to accomplish a task.

The protocols have the following characteristics:

- Everyone involved in the protocol must know the protocol and all of the steps to follow in advance.

- Everyone involved in the protocol must agree to follow it.

- The protocol must be unambiguous, each step must be well defined and there must be no chance of a misunderstanding.

- The protocol must be complete, there must be a specified action for every possible situation.

The interactive protocol is informally defined in [11] as follows:

Definition 2.2 A s-space, t-round interactive protocol \((A,B)\) is a pair of Turing machine \((A,B)\) which share an input tape (read only). Both have a private read/write work tape and read-only random tape. There are two communication tape: one which \(B\) has written only access to, and \(A\) has read only access to, and one which \(A\) has write-only access to, and \(B\) has read only access to. We think the first tape as containing messages sent to, or "questions" asked of \(A\), and the second as messages sent to, or "answers" to \(B\). The machine take \(T\) turns being active with \(B\) going first. Before each message to \(A\), all but the first \(a\) bits of \(A\)'s private work tape are erased. \(A\) is computationally unbounded and \(B\) is polynomial time bounded.
2.3 BASIC ABSTRACT ALGEBRA

<table>
<thead>
<tr>
<th>Turing Machine A</th>
<th>Communication Tape</th>
<th>Turing Machine B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public Input tape (R)</td>
<td></td>
<td>Public Input tape(R)</td>
</tr>
<tr>
<td>Public Output tape (W)</td>
<td></td>
<td>Public Output tape</td>
</tr>
<tr>
<td>Private Input tape (R)</td>
<td></td>
<td>Private Input tape(R)</td>
</tr>
<tr>
<td>Private Random tape (R)</td>
<td></td>
<td>Private Random tape(R)</td>
</tr>
<tr>
<td>Private work tape (RW)</td>
<td></td>
<td>Private work tape(RW)</td>
</tr>
</tbody>
</table>

\[ A:Output(W);Input(R):B \]

\[ A:Input(R);Output(W):B \]

Private Output tape (W)                     Private Output tape(W)

Figure 2.1: The Model of interactive protocol

We can show the protocol model in the Figure 2.1.

Non-interactive Protocol can be seen as a special case of protocol which has only one way communication tape from the Turing machine A to the Turing machine B.

2.3 Basic Abstract Algebra

We often use the special groups in cryptographic techniques. We give some of the most important notations in this section (mainly borrowed from [9, 10]).
2.3.1 Group

The ring structure depends on two closed binary operations, but the group structure only involves one closed binary operation. Now we give the definition of the group.

**Definition 2.3** If \( G \) is a non-empty set and \( \circ \) is a binary operation on \( G \), then \( (G, \circ) \) is called a group if the following conditions are satisfied.

- For all \( a, b \in G \), \( a \circ b \in G \) (Closure of \( G \) under \( \circ \))
- For all \( a, b, c \in G \), \( a \circ (b \circ c) = (a \circ b) \circ c \). (The associative Property)
- There exists \( e \in G \) with \( a \circ e = e \circ a = a \), for all \( a \in G \). (The existence of the Identity)
- For each \( a \in G \) there is an element \( b \in G \) such that \( a \circ b = b \circ a = e \). (Existence of Inverses)

In addition, if \( ab = ba \) for all \( a, b \in G \), then \( G \) is called commutative, or abelian group (The name of Norwegian mathematician Niels Henrik Abel)

**Definition 2.4** For any group \( G \) the number of elements in \( G \) is called the order of \( G \) and this is denoted by \( |G| \). When the number of the element is not finite we say that \( g \) has infinite order.

**Definition 2.5** A group \( G \) is called cyclic if there is an element \( x \in G \) such that for each \( a \in G \), \( a = x^n \) for some \( n \in \mathbb{Z} \).

2.3.2 Integer Group

The most important group in cryptographic system is \( \mathbb{Z}_N \) and \( \mathbb{Z}_N^* \).

\( \mathbb{Z}_N \) is the set of the integers modulo \( N \):
2.4. NUMBER THEORY

\[ \mathbb{Z}_N = \{x | 0 \leq x < N\}. \]

\(\mathbb{Z}_N\) is a cyclic additive group with the order of \(N\).

\(\mathbb{Z}_N^*\) is the set of positive integers smaller than \(N\) and relatively prime to \(N\):

\[ \mathbb{Z}_N^* = \{x | 1 \leq x < N, \gcd(x, N) = 1\}. \]

\(\mathbb{Z}_N^*\) is a multiplicative group modulo \(N\).

2.4 Number Theory

Greatest Common Divisor

If two numbers share no factors in common other than 1, i.e., the greatest common divisor is equal to 1, we say the numbers are relatively prime. The famous method to compute the greatest common divisor is Euclid’s algorithm.

The Euler phi function \(\phi(N)\)

The Euler phi function is written as \(\phi(N)\), it is the number of positive integers less than \(N\) that are relatively prime to \(N\).

Suppose

\[ N = \prod_{i=1}^{n} p_i^{e_i}. \]

where the \(p_i\) are distinct primes and \(e_i > 0, 1 \leq i \leq N\) then

\[ \phi(N) = \prod_{i=1}^{n} (p_i^{e_i} - p_i^{e_i-1}). \]

Inverses Modulo Number

The definition of the inverse modulo number is as follows:

Definition 2.6 Suppose \(a \in \mathbb{Z}_N\), The multiplicative inverse of \(a\) is an element \(a^{-1} \in \mathbb{Z}_N\) such that \(aa^{-1} \equiv a^{-1}a \equiv 1 (\text{mod} N)\)
Any elements in $\mathbb{Z}_n^*$ will have a multiplicative inverse. We can use the extended Euclidean algorithm to compute the inverse of a number modulo $n$.

**Fermat's little Theorem**

If $b \in \mathbb{Z}_N^*$ then $b^{\phi(N)} \equiv 1 \mod N$

**Quadratic Residue**

If there exists an integer $a \in \mathbb{Z}_N^*$ such that $x \equiv a^2 \mod N$, we say the number $x$ is a quadratic residue denoted as $QR_N$, otherwise we say it is a quadratic non-residue denoted as $QNR_N$.

## 2.5 Number Theoretic Problems

Whitfield Diffie noted that most public-key algorithms are based on one of three hard problems [6].

- **Knapsack**: Given a set of unique numbers, find a subset whose sum is $N$.

- **Discrete logarithm**: if $p$ is a prime and $g$ and $M$ are integers, find $x$ such that $g^x = M \mod p$.

- **Factoring**: if $N$ is the product of two primes, either
  - factor $N$.
  - given integers $M$ and $C$, to find $d$ such that $M^d \equiv C \mod N$.
  - given integers $e$ and $C$, to find $M$ such that $M^e \equiv C \mod N$.
  - given and integer $x$, decide whether there exists and integer $y$ such that $x \equiv y^2 \mod N$.

The computing discrete logarithm is related to the factoring. If you can solve the discrete logarithm, then you can factor. The reverse has never been proven to be true.
2.5. **NUMBER THEORETIC PROBLEMS**

2.5.1 **Knapsack Problem**

The Knapsack problem also is called the NP-complete problem.

**Definition 2.7 (The knapsack problem)** Given a set of values \(m_1, m_2, \ldots, m_n\) and the sum \(S\), compute the values of \(b_i\) such that

\[ S = b_1m_1 + b_2m_2 + \cdots + b_nm_n. \]

The values of \(b_i\) can be either zero or one, one means that the item is in the Knapsack and zero means that it is not in Knapsack.

2.5.2 **The Discrete Logarithm Problem**

The discrete logarithm problem and some of its variants definitions are listed as following:

The discrete logarithm problem

The discrete logarithm problem is the basic intractable problem which is the underlying focus of this thesis. Formally the discrete logarithm problem (DLP) is defined as follows:

**Definition 2.8 (The discrete logarithm problem)** is the following: given a cyclic group \(G\) of order \(N\), generator \(g \in G\) and an element \(h \in G_N\), find the integer \(x \in \mathbb{Z}_N\) such that \(g^x \equiv h\).

So far the discrete logarithm problem is assumed to be intractable for finite group \(G_N\) where \(N\) has a large prime factor. In the case that \(N\) is a prime, the best known method to solve the discrete logarithm problem is the number field sieve. It uses \(O(\exp((1.92+o(1))(\ln N)^{1/3}(\ln \ln N)^{2/3}))\) group operations to compute a discrete logarithm, where \(o(1)\) denotes a function of \(N\) with \(\lim_{N \to \infty} f(N) = 0\).

In current practice, it is usually recommended to choose \(N\) such that \(|N| \geq 160\).
The representation problem

The representation problem is a generalisation of the discrete logarithm problem with the respect to the number of bases. The notion of representation is defined as follows:

**Definition 2.9** A representation of an element \( h \) in a cyclic group \( G \) of order \( N \) with respect to the distinct generators \( g_1, \ldots, g_n \) of \( G \) is an \( n \)-tuple \( x_1, \ldots, x_n \in \mathbb{Z}_N \) such that

\[
    h = \prod_{i=1}^{n} g_i^{x_i}.
\]

The value \( x_i \) is called the discrete logarithm of \( h \) with the respect to \( g_i \) in the representation.

**Definition 2.10** (The representation problem) is the following: given a finite cyclic group \( G \) of order \( N \) and a generator tuple \( g_1, \ldots, g_n \) and an element \( h \), find integers \( x_1, \ldots, x_n \in \mathbb{Z}_N \) such that

\[
    h = \prod_{i=1}^{n} g_i^{x_i}.
\]

The representation problem is introduced by Brands [12] and is proven to be equivalent to the DLP. Particularly if the generators are randomly chosen, finding two different representations of an element is equivalent to solving the DLP.

The Diffie-Hellman problem

**Definition 2.11** (The Diffie-Hellman problem (DHP)) is the following: given a finite cyclic group \( G \), a generator \( g \) of \( G \) and the two elements \( g^u \) and \( g^v \) in \( G \), find the element \( g^{uv} \in G \).

Clearly the DHP is solved once the DLP is solved. However whether the DLP and the DHP are computationally equivalent remains unknown. A related computational problem is the decision Diffie-Hellman problem.
The decision Diffie-Hellman problem (DDHP) is the following: given cyclic group $G$, a generator $g$ of $G$ and three elements $g^w, g^v$ and $g^w$ in $G$ whether $g^w$ and $g^v$ are equivalent.

solution for the DHP problem implies one for the DDHP. The DDHP is assumed to be intractable. Though, its hardness compared with other problems is not much known.

The Factorisation Problem

The logarithm problem and some of its variant's definitions are listed as

\textbf{n 2.13} (The integer factorisation problem (FACTORIZING)) is the following: given a positive integer $n$, finds its prime factorisation.

\textbf{n 2.14} (The RSA problem (RSAP)) is the following: given a positive integer $n$ is the product of two distinct odd primes $p$ and $q$, a positive $e$ that is prime to $\phi(n)$ and an integer $c$, find an integer $m$ such that $m^e \equiv c \mod n$.

It is believed that the RSA and the integer factorisation problems are computationally intractable, although no formal proof of this is known. A special case (case) of the RSA problem is the quadratic residuosity problem.
The quadratic residuosity problem

Definition 2.15 (The quadratic residuosity problem (QRP)) is the following: given integers \( n \) and \( a \), \( 0 \leq a < n \), decide whether there exists an integer \( w \), \( 0 \leq w < n \), such that

\[ w^2 \equiv a \mod n. \]

2.6 Miscellaneous Notations

Here we list some of the symbols we use in this dissertation;

\begin{tabular}{|l|l|}
\hline
Symbol & Description \\
\hline
\( h, \mathcal{H}() \) & hash function \\
\hline
\( r \in_R X \) & \( r \) is randomly chosen from the set \( X \) using the uniform distribution \\
\hline
\( \mathbb{Z}_N \) & a cyclic additive group with the order \( N \) \\
\hline
\( \mathbb{Z}_N^* \) & a multiplicative group modulo \( N \) \\
\hline
\( \mathbb{R} \) & the set of real numbers \\
\hline
\( \mathbb{R}^* \) & the set of positive real numbers \\
\hline
\( U \) & the set of unknown public keys \\
\hline
\( K \) & the set of known public keys \\
\hline
\end{tabular}
Chapter 3

Basic Protocol

In this chapter, we give some basic protocols in cryptography which are used in this thesis. Readers who are familiar with this content can skip this chapter. Some materials in this chapter are borrowed from [5, 6, 13, 14, 15]

3.1 Hash Functions and the Random Oracle Model

The hash function is one of the most important building blocks in cryptographic protocol. The hash function was originally developed to sign the long message with a single short message, and later acted as an essential ingredient for the security of signature scheme. Now, it is widely used in cryptographic applications such as digital signature, electronic cash, and conventional message authentication. Cryptographic hash function can take a message of arbitrary length and produce a special size message, typically 128 or 160 bit. There are many functions which can be used as a hash function.

If given a message $x$, it is computationally infeasible to find a message $x' \neq x$ such that $h(x') = h(x)$, we say the hash function is weekly collision free.

19
If it is computationally infeasible to find two messages $x, x'$ such that $h(x') = h(x)$, we say the hash function is strongly collision free.

If the input message is given and it is hard to find another input message to produce the same output, we call the hash function a one-way hash function.

There are two kinds hash function that are widely used. One is the message digest family MD4, MD5, and the other is derived functions SHA-1, HAVAL, RIPEMD, RIPEMD-160.

If a hash function is one-way, collision free, and its output are random and unpredictable, we call it the random oracle model. It is regarding the security of the hash function. In this model, the hash function can be seen as an oracle, for each new query it always gives a random answer and the same query will give exactly the same answer. If a cryptographic system uses the oracle model, we will say that the system operates under the oracle model.

3.2 Public Key Encryption

There are two fundamental approaches in cryptography. One is conventional encryption, also known as symmetric encryption, or private key encryption, or single key encryption. Another one is public key encryption, also known as asymmetric encryption.

In the communication network, the sender sends a message to the receiver through an insecure channel, where it may be intercepted by an eavesdropper. The interception becomes possible, because we cannot secure the channel.

We use encryption to secure the message in order to communication securely through the insecure network. The sender encrypts the plain message into a cipher text using the encryption key before transmission. Then the cipher text is sent to the receiver. The receiver decrypts the cipher text and gets the plain text after receiving
3.2. PUBLIC KEY ENCRYPTION

the message. In this system, the enemy can only get the ciphertext message and cannot get the original message because he does not know the decryption key.

In the above cryptosystem scheme, if the encryption key $e_k$ and the decryption key $d_k$ are the same or easily derived, it is known as conventional encryption or private key encryption. The most widely used private encryption system is AES (Advanced Encryption Standard).

The drawback of the conventional system is that it requires the prior communication of the encryption key using a secure communication channel. In some cases, this may be very difficult. Another drawback of the conventional system is that there will be too many keys if it is extended to $n$ parties.

Public key encryption system can solve the problem. In public key encryption system, the encryption key $e_k$ (also referred to as public key) and the decryption key $d_k$ (also referred to as private key) are different. It is computationally infeasible to determine decryption $d_k$ if the encryption key $e_k$ is given. The public key can be published in some directory and the sender can send the encryption message to the receiver without prior communication of the private key. The receiver will be the only person to decrypt the message using his private key.

Formally a public key encryption scheme can be defined as follows:

**Definition 3.1** A public key encryption scheme consists of a triple of algorithms:

- A probabilistic expected polynomial-time algorithm $\text{GEN}$ which on input of the security parameter $1^k$, produces a secret key $x_a$ and a corresponding public key $y_a$ for each entity $A$.

- A (probabilistic) polynomial-time algorithm $\text{ENC}$ which on input of a public key $y_a$ and a message $m$, produces a ciphertext $c$.

- A deterministic polynomial-time algorithm $\text{DEC}$ which on input of a secret key $x_a$ and a ciphertext $c$, outputs the original message $m$. 
Here $\text{DEC}(x_a, c) = m$ only if $\text{Prob}(c = \text{ENC}(y_a, m)) > 0$. GEN, ENC and DEC are called the key generation algorithm, the encryption algorithm and the decryption algorithm respectively.

Whitfield Diffie noted that most public-key algorithms are based on one of three hard problems [6].

- Knapsack.
- Discrete logarithm.
- Factoring.

RSA public key encryption scheme was introduced in [16]. It is the easiest to understand, implement and also the most popular. Its security comes from the difficulty of factoring large numbers.

- The key generation algorithm GEN which on input of $1^k$, outputs a public key $(e, n)$ and a private key $(d, n)$ such that
  
  - For a randomly chosen $c$, $(c, e, n)$ is an instance of the RSA problem.
  - $cd = 1 \mod \phi(n)$.

- The encryption algorithm ENC that on input $(e, n)$ and a message $m \in \mathbb{Z}_n^*$, outputs
  
  $$\text{ENC}((e, n), m) = m^e \mod n.$$  

- The decryption algorithm DEC that on input of $(d, n)$ and a ciphertext $c$, outputs
  
  $$\text{DEC}((d, n), c) = c^d \mod n.$$
3.2. PUBLIC KEY ENCRYPTION

The ElGamal scheme was introduced in [17]. Its security comes from the difficulty of calculating the discrete logarithms in a finite field.

The encryption scheme consists of the three following three algorithms:

- The key generation algorithm $\text{GEN}$ that on input of $1^k$, outputs a public key $e = (p, g, y)$ and a private key $d = (p, g, x)$, where
  - $(p, g)$ is an instance of the DLP,
  - $x$ is a uniformly chosen element of $\mathbb{Z}_{p-1}$, and
  - $y = g^x \mod p$.

- The encryption algorithm $\text{ENC}$ that on input $(p, g, y)$ and a message $m \in \mathbb{Z}_p$, uniformly selects an element $r$ in $\mathbb{Z}_{p-1}$ and outputs
  $$\text{ENC}((p, g, y), m) = (g^r, my^r).$$

- The decryption algorithm $\text{DEC}$ that on input of $(p, g, x)$ and a ciphertext $(y_1, y_2)$, outputs
  $$\text{DEC}((p, g, x), (y_1, y_2)) = y_2(y_1^x)^{-1} \mod p.$$ 

The Merkle-Hellman Knapsack Cryptosystem scheme was introduced in [18]. Its security comes from the Knapsack problem, an NP-complete problem.

- The key generation algorithm $\text{GEN}$ that on input of $1^k$, outputs a public key $e = (t)$ and a private key $d = (p, a, s)$, where
  - $s = (s_1, \cdots, s_n)$ be a super increasing list of integers
  - $p > \sum_{i=1}^{n} s_i$ be prime
  - $1 \leq a \leq p - 1$. 

CHAPTER 3. BASIC PROTOCOL

- The encryption algorithm \( \text{ENC} \) that on input \((t = (t_1, \cdots, t_n))\) and a message \(m = (x_1, \cdots, x_n)\), and outputs

\[
\text{ENC}(t = (t_1, \cdots, t_n), m = (x_1, \cdots, x_n)) = \sum_{i=1}^{n} x_i t_i.
\]

- The decryption algorithm \( \text{DEC} \) that on input of \((s_1, \cdots, s_n, z = a^{-1}y), 0 \leq y \leq n(p - 1)\) and a cipher text \((y_1, \cdots, y_n)\), outputs

\[
\text{DEC}((s_1, \cdots, s_n, z), (y_1, \cdots, y_n)) = (x_1, \cdots, x_n).
\]

3.3 Digital Signature Schemes

Digital signature is one of the most fundamental achievements of modern cryptography. A digital signature on a message \(m\) is a special encryption of that message using the signer’s private key, which can be easily verified by anyone else using the user’s public key.

**Definition 3.2** A digital signature scheme consists of three following algorithms:

- A probabilistic polynomial-time algorithm \( \text{GEN} \) which on input a security parameter \(1^k\), produces a pair \((x_s, y_s)\) for a signer where \(x_s\) is the signer private key and \(y_s\) is the corresponding public key.

- A (probabilistic) polynomial-time algorithm \( \text{SIGN} \) which given a security parameter \(1^k\), a secret key \(x_s\) and a message \(m \in \{0, 1\}^k\), outputs a string \(s\) which is called the signature of \(m\).

- A deterministic polynomial-time algorithm \( \text{VER} \) which given the public key \(y_s\), a digital signature \(s\) and a message \(m\), returns true or false to indicate
3.3. DIGITAL SIGNATURE SCHEMES

whether or not $s$ is valid signature on $m$ generated using the private key $x$, 
corresponding to the given public key $y$.

GEN, SIGN and VER are called the key generation algorithm, the signing algorithm 
and the verification algorithm respectively.

Now we give three digital signature schemes.

The RSA Signature scheme:

- The key generation algorithm GEN that on input of $1^k$, outputs a large com-
  posite number $N = pq$, a public key $e$ and a private key $d$, where
  
  $ed = 1 \mod \varphi(N)$

- The signing algorithm SIGN that on input $d$ and a message $m$, and outputs

  $\operatorname{SIG}_d(m) = m^d \mod N$.

- The Verification algorithm VER that on input of $e$ and a signature text $(e, y)$, 
  outputs

  $\operatorname{VER}_e(m, y) = true \iff m \equiv y^b \mod N$.

The ElGamal Signature Signature scheme:

- The key generation algorithm GEN that on input of $1^k$, outputs three public 
  key $p, \alpha, \beta$ and a private key $a$, where

  - $p$ is a prime such that the discrete log problem in $\mathbb{Z}_p$ is intractible
  - $\beta \equiv \alpha^a \mod p$
  - $\alpha \in \mathbb{Z}_p^*$ be a primitive element
CHAPTER 3. BASIC PROTOCOL

- The signing algorithm \( \text{SIGN} \) that on input \( a \), a random number \( k \in \mathbb{Z}_{p-1} \) and a message \( m \), and outputs

\[
\text{SIG}_a(m, k) = (\gamma, \delta).
\]

where

- \( \gamma = \alpha^k \mod p \)
- \( \delta = (m - a \gamma)k^{-1} \mod p - 1 \)

- The Verification algorithm \( \text{VER} \) that on input of \( p, \alpha, \beta \) and a signature text \( (m, \gamma, \delta) \), outputs

\[
\text{VER}_a(m, \gamma, \delta) = \text{true} \iff \beta^\gamma \gamma^\delta \equiv \alpha^m \pmod{p}.
\]

The Schnorr signature Signature scheme:

- The key generation algorithm \( \text{GEN} \) that on input of \( 1^k \), outputs four public key \( p, q, \alpha, \nu \) and a private key \( a \), where

  - \( p \) is a 512-bit prime such that the discrete log problem in \( \mathbb{Z}_p \) is intractible
  - \( q \) is a 160-bit prime \( q \mid (p - 1) \) i.e \( q \) divides \( p - 1 \)
  - \( \alpha \in \mathbb{Z}_p^* \) is a \( q \)th root of \( 1 \) modulo \( p \)
  - \( h \) is a hash function with range \( \mathbb{Z}_q \)

- The signing algorithm \( \text{SIGN} \) that on input \( a \), a random number \( k \in \mathbb{Z}_{p-1} \) and a message \( m \), and outputs

\[
\text{SIG}_a(m, k) = (\gamma, y).
\]

where
3.3. DIGITAL SIGNATURE SCHEMES

- \( \gamma = \alpha^k \mod p \)
- \( y = k + ah(m, \gamma) \mod q \)

- The Verification algorithm \( \text{VER} \) that on input of \( p, \alpha, \beta \) and a signature text \( (m, \gamma, h, v) \), outputs

\[
\text{VER}_r(m, \gamma, y) = \text{true} \iff \gamma \equiv \alpha^y v^{h(m, \gamma)} \pmod{p}.
\]

There are four basic types of attack on the digital signature:

key-only attack, known message attack, chosen message attack and adaptive chosen message attack. In the key-only attack, the adversary knows only the signer's public key. In the known message attack model, the adversary is assumed to only see signatures for a set of messages chosen by the signer. In the chosen message attack model, the adversary obtains valid signatures from a chosen list of messages before attempting to break the signature scheme. In the adaptive chosen message attack model which is the most severe, the adversary is allowed to use the signer as a signing oracle, the adversary might request signatures of messages which depend on previously obtained signatures or messages.
3.4 Blind Digital Signature Schemes

Blind Digital Signature is another one of the most important cryptographic primitive. It was invented in 1982 [19]. It is the variant of the digital signature schemes. This scheme has two parties, one is called signer, and another party called provider. It allows the provider to get the signer’s signature without giving the signer any information about the message and the resulting signature. Blind signatures have many applications such as anonymous electronic cash, electronic voting, and so on.

**Definition 3.3** A blind signature scheme consists of three following algorithms:

- A probabilistic polynomial-time algorithm **GEN** that on input a security parameter $1^k$, produces a pair $(x_s, y_s)$ for a signer $S$ where $x_s$ is the signer private key and $y_s$ is the corresponding public key.

- An interactive polynomial-time protocol $(R, S)$ which takes a security parameter $1^k$ as common input, $(x_s, y_s)$ as the signer input and $(y_s, m)$ as the recipient input and produces a string $s$ which is called the signature of $m$ as the private output of the recipient. The information that the signer can gather in the protocol is called the signer’s view or transcript of the protocol.

- A deterministic polynomial-time algorithm **VER** that given the public key $y_s$, a digital signature $s$ and a message $m$, returns **true** or **false** to indicate whether or not $s$ is valid signature on $m$ generated using the private key $x_s$ corresponding to the given public key $y_s$.

**GEN**, $(R, S)$, and **VER** are called the key generation algorithm, the blind signing protocol, and the signature verification algorithm respectively.

The blind RSA Signature Scheme
3.4. **BLIND DIGITAL SIGNATURE SCHEMES**

- The key generation algorithm `GEN` that on input of `1^k`, outputs a large composite number `N = pq`, a public key `e` and a private key `d`, and a public hash function `H` where
  
  \[ ed = 1 \mod \varphi(N) \]

- The Interactive protocol `RS` (Figure 3.1)

\[
\begin{align*}
\text{Receiver} & \quad \text{Signer} \\
(m, e, N) & \quad (N, p, q, d, H) \\
\hline
m' = H(m)^r \mod N & \\
\hline
\overset{m'}{\downarrow} & \\
\hline
\hat{s} = m'^d = (H(m)r^e)^d \mod N & \\
\hline
s = \hat{s}r^{-1} \mod N
\end{align*}
\]

Figure 3.1: The blind RSA signature generation protocol

- the receiver blinds the message `m` with a random value \( r^e \mod N \)
- the signer returns a signature \( \hat{s} = s'd = (H(m)r^e)^d \mod N \)
- the receiver un-blinds this signature by computing \( s = \hat{s}r^{-1} \mod N \)

- The Verification algorithm `VER` that on input of `e` and a signature text `(e, y)`, outputs

\[ \text{VER}_k(m, s) = \text{true} \iff m^e \equiv H(m)(\text{mod}N). \]

**The blind Schnorr Signature Scheme**

Blind schnorr digital signature scheme [20] is another example of blind signature. We give briefly description here.
• The key generation algorithm \( \text{GEN} \) that on input of \( 1^k \), outputs four public key \( p, q, \alpha, v \) and a private key \( a \), where
  
  - \( p \) be a 512-bit prime such that the discrete log problem in \( \mathbb{Z}_p \) is intractible
  
  - \( q \) be a 160-bit prime \( q \mid (p - 1) \) i.e., \( q \) divides \( p - 1 \)
  
  - \( \alpha \in \mathbb{Z}_p^* \) be a \( q \)th root of 1 modulo \( p \)
  
  - \( \mathcal{H} \) be a hash function with range \( \mathbb{Z}_q \)

• The signing algorithm \( \text{SIGN} \) (Figure 3.2) that on input \( a \), a random number \( \gamma \in \mathbb{Z}_{p-1} \) and a message \( m \), and outputs

\[
\text{SIG}_a(m, \gamma) = (c, s).
\]

where

- \( c = \mathcal{H}(m||w) \mod p \)

- \( s = \tilde{s} + \lambda \mod q \)

• The Verification algorithm \( \text{VER} \) that on input of a signature text \( (m, c, s) \), outputs

\[
\text{VER}_a(m, c, s) = \text{true} \iff c \equiv \mathcal{H}(m||g^a y^c) \pmod{p}.
\]

**The Security**

There are two kinds of attacks:

• The sequential attack

• The parallel attack
3.5. **Identification Protocols and Zero Knowledge Proof**  

![Figure 3.2: The blind Schnorr signature generation protocol](image)

3.5 **Identification Protocols and Zero Knowledge Proof**

3.5.1 **Identification Protocols**

The identification scheme involves two parties: the prover and the verifier. The prover can prove that he is himself to the verifier. The Feige-Fiat-Shamir identification scheme [21] is the best known zero-knowledge proof of identity. The Ong-Shnorr identification scheme [22] is a variant of the Fiat-Shamir scheme which provides shorter communication and signatures than the former. The Schnorr Identification scheme is one of the most practical identification schemes.
Formally, a definition of identification protocols is as following:

**Definition 3.4** An identification protocol consists of two algorithms:

- A probabilistic polynomial-time algorithm $\text{GEN}$ that on input of the security parameter $1^k$, outputs a secret key $x_p$ and a corresponding public key $y_p$ for the prover.

- An interactive protocol $(\text{Pro}, \text{Ver})$ that takes $(x_p, y_p)$ as the prover’s private inputs and $(y_p)$ as the verifier input. At the completion of the protocol, the verifier’s output is true if the verifier is convinced of the prover identity and false otherwise.

$\text{GEN}$ and $(\text{Pro}, \text{Ver})$ are called the key generation algorithm and the identification algorithm respectively.

The Fiat-Shamir identification scheme’s security is based on the difficulty of factoring large integer numbers. The Ong-Shnorr identification scheme’s security is based on both discrete logarithm and factoring the large integer number.

There is a standard method of converting an identification scheme to a signature scheme due to Fiat-Shamir and someone called it the Fiat-Shamir technique [5].

But the zero knowledge identification cannot be converted into a signature scheme using this technique [23] because the signature converted from the zero knowledge identification protocol can be forged by using the same algorithm as the simulation for proving the zero-knowledge of the identification protocol.
3.5. IDENTIFICATION PROTOCOLS AND ZERO KNOWLEDGE PROOF

3.5.2 Zero Knowledge Proofs of Knowledge

Zero knowledge proof was introduced in [24] in 1985. Loosely speaking, a zero-knowledge proof is an interactive protocol that allows one party (the prover) to prove to the other party (the verifier) the validity of an assertion without yielding any information beyond the validity of the assertion.

To define zero-knowledge, we need some notions of indistinguishability of random variables introduced in [24, 25].

**Definition 3.5** Let \( L \in \{0,1\}^* \) be a language and let \( A = \{A(x)\}_{x \in L} \) and \( B = \{B(x)\}_{x \in L} \) be two ensembles of random variables indexed by string \( x \in L \). We say that the ensembles \( A \) and \( B \) are

- perfectly indistinguishable if for all \( x \in L \), \( A(x) \) and \( B(x) \) are identically distributed.

- statistically indistinguishable if for every polynomial \( p(.) \) and for all sufficient long \( x \in L \), it holds that

\[
\Sigma_{\alpha \in \{0,1\}} \left| \text{Prob}(A(x) = \alpha) - \text{Prob}(B(x) = \alpha) \right| < \frac{1}{p(x)}. \tag{3.1}
\]

- computationally indistinguishable if there exists no efficient algorithm that can distinguish \( A(x) \) and \( B(x) \).

Using the above definition, we can define the notion of zero knowledge as follows:

**Definition 3.6** (Zero-knowledge[24]) An interactive protocol \((P, \bar{V})\) is perfect, statistical, or computational zero-knowledge respectively if for every probabilistic polynomial-time verifier \( \bar{V} \) who interacts with \( P \) properly but might behave arbitrarily, there exists a probabilistic expected polynomial-time simulator \( \mathcal{S} \) so that the two ensembles
\{[\bar{V}, P]\}_{x \in L} \text{ and } \{S(x)\}_{x \in L}

are perfectly, statistically, or computationally indistinguishable.

A special class of zero-knowledge proof is zero-knowledge proof of knowledge which allows a prover to assert the knowledge of some secret object, not merely its existence without revealing any information about the object.

To formalise a proof of knowledge system, we need the following notion of the message specification function:

**Definition 3.7** (message specification function[26]) Denote by $P_{x,y,r}(\bar{m})$ the message sent by machine $P$ on common-input $x$, private input $y$ and random input $r$, after receiving message $\bar{m}$. The function $P_{x,y,r}$ is called the message specification function of machine $P$ on common input $x$, private input $y$ and random input $r$.

Now, we are ready to formally define a proof of knowledge system.

**Definition 3.8** (Proof of knowledge[26]) Let $R \subseteq \{0,1\}^* \times \{0,1\}^*$ be a binary relation. Let $R(x) = \{y : (x,y) \in R\}$ and

\[
L_R = \{x : \exists y \text{ such that } (x,y) \in R\}.
\]

Further let $\kappa : \{0,1\}^* \to [0,1]$. We say that an interactive Turing machine $V$ is a knowledge verifier for the relation $R$ with knowledge error $\kappa$ if non-triviality and validity, defined in the following, hold:

- **Non-triviality:** There exists an interactive Turing machine $P$ so that for every $(x,y) \in R$, all possible interactive protocol $(V,P)$ on common-input $x$ and $P$'s secret input $y$, $V$ accepts the proof by outputting true.
3.5. IDENTIFICATION PROTOCOLS AND ZERO KNOWLEDGE PROOF

- Validity (with error $\kappa$): There exists a polynomial $q(.)$ and a probabilistic oracle machine $K$ such that for every interactive Turing machine $P$, every $x \in L_R$ and every $y$, given access to the oracle $P_{x,y,r}$, the machine $K$ outputs a solution $s \in R(x)$ within an expected number of steps bounded by

\[
\frac{q(|x|)}{p(x) - \kappa(|x|)},
\]

where $p(x)$ is the probability that $V$ accepts the proof and $p(x) > \kappa(|x|)$. The oracle machine $K$ is called a universal knowledge extractor.

A protocol is said to be a zero-knowledge proof of knowledge if it is zero-knowledge and it constitutes a proof of knowledge.
3.6 Secret Sharing and Condition Disclosure of Secrets

Both the secret sharing scheme and the condition disclosure of secrets are cryptographic primitives. We introduce the basic concepts and the relationship between them in this section.

The condition disclosure setting includes two parties and a secret. One party is $t$ players, each holding some input, another one is an external party. The secret $s$ is known to at least one player, and the external party does not know the secret. The goal is that the players disclose the secret to the external party under some conditions input by the players. If the condition is met, the external party can learn the secret $s$. If the condition is not met, the external party learn nothing about the secret $s$. The formal definition refers to [14].

The threshold schemes independently invented in [27][28]. It is a special type of secret sharing scheme, usually referred to as the $t$ out of $m$ secret sharing. In this scheme, a secret is shared by a set of players $m$ and only an authorized subset of player $t$ ($t < m$) can recover the secret.

The generalized secret sharing scheme is an extension of the threshold scheme. It is a randomized protocol in that a secret is divide into $m$ shares and any qualified set of the shares can recover the secret, whereas any unqualified set of the shares can not recover the secret.

The condition disclosure of the secret can be reduced to generalized secret sharing, reader can refer to [14].
3.7 Oblivious Transfer Protocol

The concept of the Oblivious Transfer (OT) was introduced in [2]. Roughly speaking, oblivious transfer means the sender has one or more messages, the receiver can get nothing, one or more messages and the sender has no idea what the receiver gets. There are three types: the sender has one message, the receiver's possibility is $\frac{1}{2}$ to get the message, and the sender does not know whether the receiver receives it or not. Another is one out of two OT, it means the sender has two messages, the receiver gets one of them but he does not know which one he will get, and the sender also does not know which one the receiver gets. Third is the one chosen out of two OT, this means the sender has two messages, receiver chooses one of them, the sender still does not know which message the receiver gets.

These are extended into more general notations. The first one extended to $p$-OT, it means the receiver's possibility is $p$ to get the message, and the sender does not know whether the receiver gets it or not. The second and third are extended to one out of $n$ and one chosen out of $n$ respectively.

The OT can be divided two-fold from the communication perspective, it is the interactive and non-interactive.

Some other OT notations or protocols includes reversing OT [29], quantum OT [30, 31, 32], priced OT [4], fair OT [33], Pre-computing OT [34], committed OT [35], etc.

The OT protocols are used in many areas [36, 37] [38], it also includes blind signature [33], electronic commerce [4], mobile phone security [39] and so on.

In this section, we give some basic concepts and protocols.
3.7.1 Rabin’s Oblivious Transfer Protocol

The notion of oblivious transfer (OT) was introduced in 1982 and it became a very important tool in designing cryptographic protocol. The security of the implementation of oblivious transfer is based on the difficulty of the factoring.

The definition of the Oblivious transfer protocol is as follows, and the protocol is shown in Figure 3.3:

**Definition 3.9** Oblivious Transfer

- *Sender knows one bit* $b$
- *Receiver gets* $b$ *from sender with probability* $1/2$
- *Receiver knows if he got* $b$ *or not*
- *Sender does not know whether receiver got* $b$ *or not*

Rabin’s Oblivious Transfer Protocol

<table>
<thead>
<tr>
<th>sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(p, q, N)$</td>
<td>$(N)$</td>
</tr>
<tr>
<td>$N = pq$</td>
<td></td>
</tr>
</tbody>
</table>

\[ N \rightarrow y = x^2 \mod N \]

\[ u = \sqrt{y} \mod N \]

\[ u \rightarrow \]

Figure 3.3: The Rabin’s Oblivious Transfer protocol
3.7. OBLIVIOUS TRANSFER PROTOCOL

3.7.2 One out of Two Oblivious Transfer Protocol

One out of two OT can be seen as another type of OT, its definition is as follows:

Definition 3.10 One out of two oblivious Transfer protocol

- Sender knows two bits \( b_0 \) and \( b_1 \)
- Receiver gets \( b_0 \) or \( b_1 \) from sender with probability \( 1/2 \)
- Receiver knows if he got \( b_0 \) or \( b_1 \)
- Sender does not know receiver got \( b_0 \) or \( nb_1 \)

3.7.3 Non-Interactive Oblivious Transfer Protocol

The non-interactive protocol was introduced in 1989 [3]. The definition and the implementation protocol is as follows (Figure 3.4)

Definition 3.11 One out of two oblivious transfer protocol

The sender has two string \( s_a, s_b \). Sender computes a message \( m \) using a function and the receiver's public key and sends it to the receiver. The receiver extracts one of the string \( s_a, s_b \) from the message \( m \). The sender will not know which of the two the receiver got.

Key Generation

Let \( p \) be a prime, generator \( g \) of \( \mathbb{Z}_p^* \), and some element \( C \) of \( \mathbb{Z}_p^* \). The receiver randomly picks \( i \in \{0, 1\} \), and \( x_i \in \{0, \cdots, p - 2\} \) and sets

- \( \beta_i = g^{x_i} \mod p \)
- \( \beta_{1-i} = \frac{C}{g^{x_i}} \mod p \)
Figure 3.4: The Non-interactive Oblivious Transfer protocol

Then the public key is \((\beta_0, \beta_1)\) and the secret key is \((i, x_i)\)

**Non-interactive Oblivious Transfer Protocol**

The non-interactive OT protocol is shown in Figure 3.4, we explain it as follows:

- The sender randomly picks \(y_0, y_1 \in \{0, \cdots, p-2\}\) and sends \(\alpha_0 = g^{y_0}, \alpha_1 = g^{y_1}\) to receiver

- The sender then calculates \(\gamma_0 = \beta_0^{y_0}, \gamma_1 = \beta_1^{y_1}\) and sends \(R_0 = s_a \oplus \gamma_0, R_1 = s_b \oplus \gamma_1\) to receiver

- The receiver calculates \(\gamma_0 = \alpha_0^{x_0}, \gamma_1 = \alpha_1^{x_1}\) using his secret key.

- The receiver calculates \(s_a = \gamma_0 \oplus R_0, s_b = \gamma_1 \oplus R_1\).
Chapter 4

Oblivious Transfer and Blind Signature

4.1 Introduction

The notion of Oblivious Transfer (OT) was developed with an implementation based on the factoring problem. The related concept of one out of two oblivious transfer was proposed in [40] together with some applications. The other related concept of one chosen out of two oblivious transfer was introduced in [41]. The $p$-O.T. Protocol and $\alpha$-slightly Oblivious Transfer were introduced in [42] and [43] separately. The first and last one refer to one bit, and the second and the third refer to two bits. The two flavors of oblivious transfer (oblivious transfer and one out of two oblivious transfer) was proved computationally equivalent in [44].

The one out of two strings oblivious transfer was presented in [45], in which the bits are extended by two strings. A method to extend one out of two Oblivious Transfer to one out of two string oblivious transfer protocol was given in [46]. Another technique to reduce one out of two strings oblivious transfer to one out of two
oblivious transfer protocol was introduced in [47].

The notion of one out of $n$ strings oblivious transfer or ANDOS (All or Nothing Disclosure of Secrets protocol) was given in [48]. And that the one out of two OT is a special case of one out of $n$ was mentioned in [48]. The other paper [49] gave a set of reduction showing that the one of two oblivious bit transfer problem leads to an ANDOS.

In this chapter, we introduce the notion of $m$ out of $n$ strings oblivious transfer and $m$ chosen out of $n$ strings oblivious transfer and an implementation of this oblivious transfer protocol. We then explain that one out of $n$ OT and one out of two OT are a special case of $m$ out of $n$ OT. After this we give a more efficient OT and one application of $m$ out of $n$ OT.

This chapter is organized as follows. First, section 2 introduces the notion of $m$ out of $n$ oblivious transfer. In section 3 we present the implementation of $m$ out of $n$ oblivious transfer and then discuss the relation of three concepts and implementations. Section 4 gives a more efficient $m$ out of $n$ OT Protocol. The application for $m$ out of $n$ oblivious transfer is given in section 5, and the last section is the conclusion.

4.2 Concept of $m$ out of $n$ Oblivious Transfer

The concept of oblivious transfer was introduced in [40]. Formally its definition is as follows:

- Sender knows one bit $b$
- Receiver gets bit $b$ from sender with probability $1/2$
- Receiver knows whether he got $b$ or not
4.2. CONCEPT OF M OUT OF N OBLIVIOUS TRANSFER

- Sender does not know whether Receiver got $b$ or not

The notion of the one out of two oblivious transfer defined in [40] is as follows:

- Sender knows two bit $b_0$ and $b_1$.

- Receiver gets bit $b_k$ and not $b_{\overline{k}}$ from sender with probability $1/2$ ($k = 0$ or $1$).

- Receiver knows whether he got it or not.

- Sender does not know whether Receiver got $b_0$ or $b_1$.

The difference between the one out of two oblivious transfer and one chosen out of two oblivious transfer is that the receiver can choose one of the bit and the sender does not know which one the receiver got.

The one out of two strings oblivious transfer protocol is that the sender has two messages, the receiver can get one of them, and the receive cannot find which one the receiver got.

Similarly, we define the $m$ out of $n$ strings oblivious transfer as following:

- Sender knows $n$ messages

- Receiver gets $m$ different messages from sender

- Receiver knows whether he got it or not

- Sender does not know which $m$ messages Receiver got

In the $m$ chosen out of $n$ oblivious transfer, the receiver can choose any of the $m$ message from $n$ messages, and the sender has no idea which one the receiver chooses.
4.3 Implementation of \( m \) out of \( n \) Oblivious Transfer

In this section, we present the \( m \) out of \( n \) messages oblivious transfer protocol, and then give the \( m \) chosen out of \( n \) message oblivious transfer protocol.

4.3.1 \( m \) out of \( n \) Oblivious Transfer

The first interactive \( m \) out of \( n \) oblivious transfer protocol scheme is hybrid. We described it briefly as follows.

The sender generates \( n \) pairs of public-private keys based on the key generator \( \mathcal{G}(1^t) \). The information theoretic security level for this scheme is \( 1 - \mathcal{O}(2^t) \). Then he publishes all the public keys. The receiver generates \( m (m < n) \) secret keys, and encrypts these keys using \( m \) of the sender's public keys, then sends the encrypted keys to the sender.

The sender has no idea which public keys have been used and just decrypts each encrypted key using all the \( n \) public keys. Obviously only one of them is correct, and the sender does not know which one is correct. Then the sender encrypts \( n \) messages using the decrypted keys. The sender repeats the process for all \( m \) encrypted symmetric keys and then sends them to receiver.

The receiver gets \( m \times n \) encrypted messages and then he decrypts all of them, finally he can get \( m \) correct messages.

Suppose that the sender has \( n \) messages \( M_i, i = 1, \ldots, n \). The receiver wants to get \( m \) different messages out of \( n \) messages, at same time, cannot get anything about the other \( n - m \) messages.

The \( m \) out of \( n \) oblivious transfer protocol is shown in Figure 4.1:

Let us explain the above protocol in detail:
### 4.3. IMPLEMENTATION OF M OUT OF N OBLIVIOUS TRANSFER

<table>
<thead>
<tr>
<th>Sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomly choose $(E_i, D_i)$ [i = 1, \ldots, n]</td>
<td>Randomly choose $K_j$ - key, $j = 1, \ldots, m$</td>
</tr>
<tr>
<td>$E_{E_i, i=1,\ldots,n}$</td>
<td>$F_{r_j, j=1,\ldots,n}$</td>
</tr>
<tr>
<td>$k_{ij} = D_i(E_{r_j}(k_j))$</td>
<td>$r_j \in 1, \ldots, n$</td>
</tr>
<tr>
<td>$i \in 1, \ldots, n$</td>
<td></td>
</tr>
<tr>
<td>$j = 1, \ldots, m$</td>
<td></td>
</tr>
<tr>
<td>$C_{ij} = F_{k_{ij}}(M'_i)$, $i = 1, \ldots, n$, $s = 1, \ldots, m$</td>
<td>$C_{ij, i=1,\ldots,n,j=1,\ldots,m}$</td>
</tr>
<tr>
<td>$M_j = D_{K_j}(C_{ij})$</td>
<td>$M_j = D_{K_j}(C_{ij})$</td>
</tr>
<tr>
<td>$j = 1, \ldots, m$</td>
<td>$j = 1, \ldots, m$</td>
</tr>
</tbody>
</table>

**Figure 4.1: m out of n Oblivious Transfer**
1. Sender randomly chooses $n$ encryption and decryption algorithms of the public key system, $(E_i, D_i), i = 1, \cdots, n$. And sends all of the encryption algorithms to Receiver.

2. Receiver Randomly chooses $m$ keys $K_j, j = 1, \cdots, m$, and a cryptosystem. Then, chooses $m$ out of $n$ encryption algorithms and uses it to encrypt the randomly chosen keys. Finally he sends it to the sender.

3. Sender uses the decryption algorithm to decrypt all the keys which he got from the receiver. Then sender can get $m \times n$ keys.

4. Sender randomly chooses $s \in 1, \cdots, n$, and reorders the message $M'_i = M_{n+1-s}, S \in (1, \cdots, n)$. For $j = 1$ to $m$, he uses the cryptosystem to encrypt the message $M_i, (i = 1, \cdots, n)$ and sends it to the receiver in order.

5. Receiver knows the encryption number chosen and chooses them and gets the messages. After $m$ times, the receiver gets $m$ messages.

Note that the sender cannot mess up the order, otherwise the receiver will find it hard to get the messages.

### 4.3.2 Security Analysis

Let us analyze the security of this scheme.

- **Completeness** If the sender correctly follows the process, the receiver will receive exactly $m$ out of $n$ messages sent by sender.

The receiver gets $n$ public keys from the sender, then he uses $m$ of them to encrypt the key $k_j, j = 1, \cdots, m$, and keep the chosen public indices secret. The sender gets $m$ encrypted keys and does not know which one is the corresponding secret key, but if he decrypts one in all his secret keys, there must be
one that is correct. Next the sender uses all of them to encrypt \( n \) messages, obviously the receiver can only decrypt one correct message. In this way, the sender repeats \( m \) times and the receiver can get exactly \( m \) messages.

- **Soundness** *The receiver cannot get more than \( m \) messages*

The receiver cannot get more than \( m \) messages because receiver can only choose \( m \) keys to encrypt. If receiver chooses any more than \( m \), the sender will obviously know.

One the other hand, the sender cannot cheat, because the receiver will know if he gets less than \( m \) messages or if one more messages are same. Sender does not know which one the receiver gets because he does not know which are the real keys after he decrypts the receiver keys.

So, if both of them follow the protocol, the receiver will get exactly \( m \) messages, and the sender has no idea which messages the receiver gets. In conclusion, the proposed scheme is secure.

In the above scheme, the sender has to do \( m \times n \) encryptions. If the \( m \) and \( n \) are large, obviously this scheme will have the drawback of computational overhead. We will give another \( m \) out of \( n \) oblivious transfer scheme which is more efficient and only requires \( n \) encryption.

### 4.3.3 \( m \) Chosen out of \( n \) Oblivious Transfer

We can get the \( m \) chosen out of \( n \) oblivious transfer from the above protocol. In the fourth step, if the sender does not change the message order, the receiver can get exactly the messages corresponding to the encryption algorithm. This is because the receiver knows the messages order the sender lists. The protocol is shown in Figure 4.2:
Sender

Randomly choose
\((E_i, D_i)\)
\(i = 1, \cdots, n\)

\[ E_{i,i=1,\cdots,n} \]

Receiver

Randomly choose
\(K_j\) - key, \(j = 1, \cdots, m\)
\(F\) - Cryptosystem

\[ r_j \in 1, \cdots, n \]

\(k_{ij} = D_i(E_{r_j}(k_j))\)
\(i \in 1, \cdots, n\)
\(j = 1, \cdots, m\)

\[ C_{ij} = F'_{k_{ij}}(M_i^s), \]
\(i = 1, \cdots, n, s = 1, \cdots, m\)

\[ C_{ij,i=1,\cdots,n,j=1,\cdots,m} \]

\[ M_j = D_{K_j}(C_{ij}) \]
\(j = 1, \cdots, m\)

Figure 4.2: \(m\) Chosen out of \(n\) Oblivious Transfer

### 4.3.4 Oblivious Transfer

The one out of \(n\) string oblivious transfer and the oblivious transfer can be regarded as a special case of the \(m\) out of \(n\) oblivious transfer protocol. We can get the one out of \(n\) OT protocol and the oblivious transfer from the above \(m\) out of \(n\) protocol.
One out of $n$ Oblivious Transfer

In the above protocol, if we choose $m = 1$, we can get the one out of $n$ oblivious transfer protocol. In the same way, we can get the one chosen out of $n$ string protocol.

One out of two Oblivious Transfer

We can get the one out of two oblivious transfer protocol, if we let $n = 2$. Also, in the same way we can get the one chosen out of two oblivious transfer.

4.4 Efficient $m$ out of $n$ OT Scheme

In this section we develop the efficient $m$ out of $n$ OT protocol and discuss the security of the scheme.

4.4.1 System Set up

We develop an efficient $m$ out of $n$ OT scheme using shared public keys which has $m$ real keys in $n$ public keys. The method to set up the keys will be given in next chapter.

Let $p$ be a large prime number, $q$ be an integer for $q|p - 1$, $\mathbb{Z}_p^*$ be a multiplicative group of order $q$. Suppose that the receiver sets up his system with the private keys $s_j, j = 1, \cdots, m$, the public keys $y_i, i = 1, \cdots, n(m < n)$ and $y = g^s \mod p$. Then the efficient $m$ out of $n$ OT protocol is as shown in Figure 4.3:

We explain the efficient $m$ out of $n$ Oblivious Transfer protocol as follows:

- Receiver

  - Chooses $m$ symmetric keys $k_j$
Figure 4.3: The efficient $m$ out of $n$ protocol
4.4. EFFICIENT M OUT OF N OT SCHEME

- Encrypts \( m \) symmetric keys \( k_j \)
  
  * Randomly selects numbers \( w_j \in \mathbb{Z}_q^m, j \in \mathbb{K} \)
  
  * Compute \( r_j = k_j g^{w_j} \) and \( k_j' = s_j r_j + w_j \mod q \). Note that the symmetric keys can be decrypted using the public key \( y_i, j \in \mathbb{K} \).

- Extends \( k_j', j = 1, \cdots, m \) to \( k_i', i = 1, \cdots, n \) using the method which will be given in the next chapter (System Set up section)

- Similarly, extends \( r_j, j = 1, \cdots, m \) to \( r_i', i = 1, \cdots, n \)

- Sends \( k_i' \) and \( r_i' \) to sender

- Sender

  - Decrypts \( k_i'' \) using the public keys \( y_i \) in order. Note that only \( m \) out of \( n \) are correct.

\[
k_i = g^{-k_i''} y_i^{r_i' r_i} = g^{s_i r_i' k_i g^{w_i}} = g^{s_i r_i' g^{w_i} k_i} = k_i
\]

  - Encrypts \( n \) messages using the symmetric keys \( k_i \) in order, \( C_i = E_{k_i}(M_i) \).

  - Sends \( C_i \) to the receiver

- Receiver

Receiver decrypts \( m \) of \( C_i \)

\[
M_j = D_{k_j}(C_j), j \in \mathbb{K}
\]

This method is more efficient than the first one, because the sender only needs to do \( n \) encryption instead of \( m \times n \) encryption.
4.4.2 Security Analysis

Let us analyze the security of the scheme:

- **Completeness**

  *The completeness of the protocol is two fold:*
  
  - The sender can correctly decrypt $m$ out of $n$ keys, if the receiver has encrypted the $m$ keys using his correct private keys.
  
  - The receiver can correctly decrypt $m$ messages, if the sender correctly follows the process.

The first one is due to the correctness of the encryption method that is based on the Nyberg-Rueppel digital signature scheme [50]. The second one is obvious.

- **Soundness**

  *The receiver can obtain at most $m$ correct messages.*

We can know the sender has $n$ public keys but only $m$ of them has the corresponding private key from the public key set up system (which will be introduced in next chapter). In other words, the receiver only has $m$ private keys and can only choose $m$ symmetric keys. Otherwise, the sender can find out that the receiver is cheating. On the other hand, because the receiver uses the variant of the Nyberg-Rueppel digital signature scheme [50] to encrypt the symmetric keys $k_i$, the security is assured. This is because the public keys $y_i$ are actually known to the receiver and sender only, and no one else can decrypt the "signature". The receiver cannot cheat by using his private key more than once, since his public keys on the sender side and are placed in order and used only once.
4.5 Blind Signature

The notion of blind signature was introduced in [19]. The blind signature scheme involves two parties, the sender and the signer. It is a protocol in which the sender can get a message signed by the signer without revealing any information to the signer. The blind signature protocol using the oblivious transfer protocol was developed [51]. It is not efficient in practice, as it needs a lot of communications.

In this section, we develop the blind signature using the $m$ out $n$ oblivious transfer. We can see it only needs two rounds of communication. It is more efficient than the one in [51].

The blind signature protocol using the $m$ out of $n$ oblivious transfer is shown in Figure 4.4.

Then the pair $(\tilde{s}, \tilde{r})$ is a valid signature of $m$ ($C_i$ is the $i$-th bit of $\mathcal{H}(\tilde{r} || m)$)

$$\tilde{s}^3 = \tilde{r} \prod_{i=1}^{k} y_i^{C_i} (mod)n$$

We based the above protocol on a variation of the Fiat-Shamir signature scheme [52] and on the concept on $m$ out of $n$ oblivious transfer protocol.

Let us analyze the blindness of the scheme: The signer cannot get any more information than the number of the bits, so the signer has no idea about the $c_i$. This means that the sender gets the message signed, and the signer does not know the message he signed.

4.6 Concluding Remarks

The $m$ out of $n$ oblivious transfer protocol can solve the problem that the receiver can get $m$ different messages from the sender and the sender has no idea which messages the receiver takes. Other protocols cannot realize this. This protocol still
\textbf{Sender()} \\
\hspace{1cm} \text{choose } r_1, \ldots, r_k \in \mathbb{Z}_n^* \\
\hspace{1cm} t = \prod_{i=1}^k r_i^3 \mod n \\

Randomly choose } \alpha \in \mathbb{Z}_n^* \leftarrow t \\
\hspace{1cm} t = ta^3 \mod n \\
\hspace{1cm} c = H(t || m) \\
\hspace{1cm} c_i \text{ is the } i\text{-th bit of } c \\
\hspace{1cm} \text{for } c_i = 0 \text{ do} \\
\hspace{2cm} s_j = m \cdot c_i \\
\hspace{2cm} \text{for } c_i = 1 \text{ do} \\
\hspace{2cm} s_i = m \cdot c_i \\
\hspace{2cm} m_0 = r_j \\
\hspace{2cm} m_1 = r_i \cdot x_i \\
\hspace{2cm} s_i = s_i \cup s_j \\
\hspace{2cm} \tilde{s} = \alpha \prod_{i=1}^k s_i \mod n \\

\textbf{Figure 4.4: The blind signature protocol}
needs to be made more efficient and practical. As other oblivious protocols, the non-interactive oblivious transfer has also been developed. And this new developed protocol can be used in more applications.
Chapter 5

Non-interactive Oblivious Transfer

5.1 Introduction

The first Oblivious Transfer (OT) was proposed in [53], with an implementation based on the factoring problem. The related concept of one out of two oblivious transfer was proposed by in [40] along with some applications. The other related concept of one chosen of two oblivious transfer was introduced in [41].

The $m$ out of $n$ oblivious transfer protocol was introduced in the last chapter. Roughly speaking, the $m$ out of $n$ oblivious transfer means that the sender has $n$ messages, the receiver can get $m$ ($m < n$) messages from the sender and the sender has no idea which one the receiver got.

Non-Interactive Oblivious Transfer was first proposed in [3]. It can be used for one out of two, two out of three and any $n - 1$ out of $n$ oblivious transfer protocol. Another non-interactive oblivious transfer protocol was developed in [54].

In this chapter, we develop a non-interactive $m$ out of $n$ oblivious transfer protocol. It can be used as any $m \geq 1$ $m$ out of $n$ oblivious transfer protocol.

This chapter is organized as follows. Section 2 introduces the notion of non-
interactive $m$ out of $n$ oblivious transfer. Section 3 discusses the general way to generate the keys. In section 4 we present an simple example of how to generate the keys. The implementation of the non-interactive of $m$ out of $n$ oblivious transfer is given in section 5. Section 6 discusses the security of this scheme. Section 7 gives a modified non-interactive $m$ out of $n$ OT protocol which can be used to prevent fraud. The last section is the conclusion.

5.2 Non-Interactive Oblivious Transfer

The non-interactive oblivious transfer was used in [3] and in [54].

In [3], Non-Interactive Oblivious Transfer is defined as follow: A has two strings $s_0$ and $s_1$. As a function of these and B's public key $P_b$ she computes a message $m$ and sends it to $B$. Using his secret key $S_B$, $B$ can extract from $m$ exactly one of the strings $s_0$ or $s_1$. $A$ will not know which of the two $B$ got.

In [54], Non-Interactive Oblivious Transfer is defined as the following:

A Non-Interactive Oblivious Transfer is a quadruple of algorithm ($Key$-$Generator$; $Verify$-$Public$ $Key$, $Sender$, $Receiver$), where $Key$-$Generator$ and $Sender$ are probabilistic polynomial time and $Verify$-$public$ $Key$ and $Receiver$ are deterministic polynomial time, such that:

- Meaningfullness: the receiver gets one of the two strings
- Verifiability: the validity of the construction can be efficiently verified
- 1 out-of 2: The receiver gets only one string and not even a bit of the other.
- Obliviousness: The sender cannot predict which string is going to be received.

Similarly, we define the non-interactive $m$ out of $n$ oblivious transfer as the following:
5.2. NON-INTERACTIVE OBLIVIOUS TRANSFER

Definition 5.1: A non-interactive \( m \) out of \( n \) oblivious transfer is a quadruple of algorithm \((\text{Key Generator}, \text{Verify - PublicKey}, \text{Sender}, \text{Receiver})\) such that:

- **Meaningfulness:** the receiver gets \( m \) of \( n \) messages
- **Verifiability:** the validity of the construction can be efficiently verified
- \( m \) out-of \( n \): The receiver gets only \( m \) messages and not even a bit of the other.
- **Obliviousness:** The sender can not predict which messages is going to be received.

The non-interactive \( m \) out of \( n \) model we developed above is briefly described as follows.

Firstly, the receiver generates \( m \) pairs of private-public keys based on the key generator \( G(1^k) \). Then he generates \( n - m \) other public keys using the public key setup algorithm. The public key setup algorithm can generate more public keys based on the public keys developed before, and no one can develop the corresponding private keys, anyone can check \( n - m \) public keys are produced by the other \( m \) public keys and no one else can find which \( m \) public keys have associated private key and which public keys do not have associated private key. Next, the receiver publishes the \( n \) public keys. Finally, the sender can implement the non-interactive OT protocol by using these public keys with the receiver.

Let us suppose that the sender has \( m \) out of \( n \) messages to be sent to the receiver. To implement the non-interactive OT, the sender changes the message order using a random number generator and then encrypts the messages using the receiver’s public keys in order respectively. When the receiver gets the encrypt messages, he can decrypt \( m \) messages using his private keys. Because the receiver knows the public keys order, he can decide which one he should decrypt.
The security is assured in this scheme. The sender cannot cheat and he has to use the receiver's public keys to encrypt the messages because the receiver will decrypt the encrypted messages using his corresponding private keys after he gets it. The receiver can not cheat to gain more than \( m \) messages, as the sender can check if only \( m \) of \( n \) public keys are useful and the other \( n - m \) keys do not have the corresponding private keys. This feature eliminates the need for a trusted third party.

5.3 System Set up

Assume that the sender has \( n \) messages, \( M_1, M_2, \ldots, M_n \in \mathbb{Z}_p^n \). He intends to let the receiver get \( m \) of them and has no idea which \( m \) of them the receiver gets.

In this section, we describe the public key generation algorithm and the public key verifying algorithm in general, In the next section we will give an example to explain it. We will use it to build the non-interactive \( m \) out of \( n \) OT protocol.

Let us suppose:

- \( p \) is a large prime number
- \( q \) is an integer for \( q | p - 1 \)
- \( \mathbb{Z}_p^* \) is a multiplicative group of order \( q \)
- \( g \) is a primitive element
- \( x_i \in \mathbb{Z}_p, i = 1, \ldots, n \) is a set of integers

All this data is pre-agreed and made public. For simplicity, we omit modulus \( p \) in the rest of the presentation.

5.3.1 Public Key Generation Algorithm

The receiver selects \( m \) private keys \( s_i \in \mathbb{Z}_q, i = 1, \ldots, m \) and then computes \( y_j = g^{x_i}, i = 1, \ldots, m, (m < n) \). Given \( x_i, i = 1, \ldots, m \). We can use a group of \( m \) linear equations
5.3. **SYSTEM SET UP**

To construct other \((n > m)\) public keys.

In general, we have

\[
y_{ij} = a(x_{ij})
\]

1 \(\leq j \leq m\), where

\[
a(x_j) = a_1x_j^1 + a_2x_j^2 + \cdots + a_mx_j^m
\]  \(\text{(5.1)}\)

The system of linear equations is shown as follows:

\[
a_1x_{i1} + a_2x_{i1}^2 + \cdots + a_mx_{i1}^m = y_{i1}
\]

\[
a_1x_{i2} + a_2x_{i2}^2 + \cdots + a_mx_{i2}^m = y_{i2}
\]

\[\vdots\]

\[
a_1x_{im} + a_2x_{im}^2 + \cdots + a_mx_{im}^m = y_{im}
\]

This linear equations can be written in matrix form as follows:

\[
\begin{pmatrix}
1 & x_{i1} & x_{i1}^2 & \cdots & x_{i1}^{m-1} \\
1 & x_{i2} & x_{i2}^2 & \cdots & x_{i2}^{m-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_{im} & x_{im}^2 & \cdots & x_{im}^{m-1}
\end{pmatrix}
\begin{pmatrix}
a_1' \\
a_2' \\
\vdots \\
a_m'
\end{pmatrix}
= 
\begin{pmatrix}
y_{i1} \\
y_{i2} \\
\vdots \\
y_{im}
\end{pmatrix}
\]  \(\text{(5.2)}\)

Note that

\[
a_j' = a_ix_{ij}
\]

The coefficient matrix \(A\) is a so-called Vandermonde matrix or non-singular matrix. The determinant of a Vandermonde matrix is not equal to zero:
\[ \text{det} A = \prod_{1 \leq j < k \leq m} (x_{ik} - x_{ij}) \neq 0 \]

This is because the \( x_{ij} \) the receiver chooses are distinct, no item \( x_{ij} - x_{ik} \) in this product is equal zero. The product is computed in \( \mathbb{Z}_p \). The product of non-zero element in \( \mathbb{Z}_p \) is always non-zero, so \( \text{det} A \) is not equal to zero. Because the determinant of the coefficient matrix is non-zero, the system has a unique solution over the field \( \mathbb{Z}_p \).

After we have the unique solution, we can calculate the other \( m - n \) public keys using the following formula:

\[ y_{ij} = a(x_{ij}) \]

i.e.

\[ a(x_j) = a_1 x_j + a_2 x_j^2 + \cdots + a_m x_j^m \]

Note that their discrete logs are unknown to anyone.

As a result, the receiver gets \( n \) public keys. Let us say it is \( y_i, i = 1, \ldots, n \) and he then shuffles his public keys such that the order of the public keys is known only to himself. Finally, these shuffled public keys are published.

For convenience, let us assume:

\( \mathbb{U} \) is the subset of public key indices whose corresponding public key discrete logs are unknown to the receiver

\( \mathbb{K} \) is the subset of public key indices whose corresponding public key discrete logs are known to the receiver

\( \{x_i, y_i\}_{i=1}^n \) is the shuffled public key set
5.3. SYSTEM SET UP

5.3.2 Public Key Verification

The public keys can be easily verified without knowing the corresponding private
keys. The user can use the following method.

Because the public key set \( \{x_i, y_i\}_{i=1}^{n} \) is given, we can choose any \( m \) of them from
the public key set, and then calculate \( a_i, i = 1, \ldots, m \) with respect to the \( m \) public
keys.

With the result \( a_i, i = 1, \ldots, m \), we verify the rest of \( (n - m) \) public keys.

\[
y_j = a_1 x_{ij} + a_2 x_{ij}^2 + \cdots + a_m x_{ij}^m, m < j \leq n
\]

5.3.3 Security Considerations

Claim 1 Given \( x_i \), receiver can not cheat in poly-time.

The explanation is as follows: After the receiver found the unique coefficient set
\( \{a'_i\}_{i=1}^{m} \), he can compute \( y_i \) for \( i \in U \) in terms of the given \( \{x_i\}_{i=m+1}^{n} \). However, it is
infeasible to compute the discrete logs of these values.

The receiver should not be able to cheat by pre-selecting \( \{y_i\}_{i=m+1}^{n} \) and then
trying to find \( \{a'_i\}_{i=1}^{m} \) that satisfies all equations. To fix the potential problem, we
give the following lemma.

Lemma 5.1 To prevent receiver from cheating by pre-selecting all \( \{y_i\}_{i=m+1}^{n} \), the
rank of matrix \( A \) must be \( m + 1 \) where:

\[
A' = \begin{pmatrix}
1 & x_{i1} & x_{i1}^2 & \cdots & x_{i1}^{m-1} & y_1 \\
1 & x_{i2} & x_{i2}^2 & \cdots & x_{i2}^{m-1} & y_2 \\
& \cdots & \cdots & \cdots & \cdots & \cdots \\
1 & x_{im} & x_{im}^2 & \cdots & x_{im}^{m-1} & y_n
\end{pmatrix}
\]

Which is an \( n \times (m + 1) \) matrix.
Proof: Rewrite the equation as

\[
A' \begin{pmatrix} a_1 \\ a_2 \\ \cdots \\ a_m \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \cdots \\ 0 \end{pmatrix}
\]

Because the rank of $A'$ is $m + 1$, it can be rewritten as

\[
A' = \begin{pmatrix} B \\ 0 \\ 0 \end{pmatrix}
\]

The triangle matrix is an $(m + 1) \times (m + 1)$ non-singular matrix, where $B$ represents the non-zero part. Assume that the triangle matrix is $T$ and then we have

\[
\begin{pmatrix} a_1 \\ a_2 \\ \cdots \\ a_m \\ -1 \end{pmatrix} = T^{-1} \begin{pmatrix} 0 \\ 0 \\ \cdots \\ 0 \end{pmatrix}
\]

Because $T^{-1}$ is non-zero, it implies that the equations with respect to $a_i$ have no solution at all. In other words, the receiver can not find a solution if he wants to cheat by pre-computing $y_i$.

Therefore, in the verification of public keys, we also need to check if the rank a $A'$ is equal to $m + 1$.

Claim 2 The set up is correct for $1 \leq m < n$. 

Now we know how to set up the public key. The correctness for \( m > 1 \) is obvious. Let us have a look at the special case when \( m = 1 \). In this case, the Equation (1) is reduced as one equation,

\[ a_1 x_1 = y_1 \]

where the receiver has one public key \( y_1 \) and corresponding private key \( s_1 \). We can see the set up system works in the same way.

Given \( x_1 \) and \( y_1 \), \( a_1 \) can be determined. Using \( a_1 \) and \( \{x_i\}_{i=2}^{n} \) we can form

\[ y_i = a_1 x_i, \quad i = 2, \ldots, n \]

### 5.4 A Simple Example

Before we go into the details, let us see a simple example to show how to generate the public keys and verify the public key.

#### 5.4.1 Key Generator Algorithm - A Simple Example

Let us suppose the receiver(B) has three secret keys \( s_j \) \( j = 1, 2, 3 \), \( p = 17 \). B calculates \( y_j = g^{x_j} \). Let us assume

\[ y_1 = 8 \]

\[ y_3 = 10 \]

\[ y_5 = 11 \]

Writing the polynomial \( a(x) \) as

\[ a(x) = a_0 + a_1 x + a_2 x^2 \]
and computing $a(1), a(3), a(5)$, the following three linear equations in $\mathbb{Z}_{17}$ are obtained:

$$a_0 + a_1 + a_2 = 8$$

$$a_0 + 3a_1 + 9a_2 = 10$$

$$a_0 + 5a_1 + 8a_2 = 11$$

This system does have a unique solution in $\mathbb{Z}_{17}$: $a_0 = 13$, $a_1 = 10$, and $a_2 = 2$.

So we have the following equation:

$$a(x) = 13 + 10x + 2x^2$$

In order to get another $n - m = 5 - 3 = 2$ public keys, then calculating $a(2) = 7$, and $a(4) = 0$. Let $y_2 = 7$ and $y_4 = 0$. Then we can get the public keys

$$y_1 = 8$$

$$y_2 = 7$$

$$y_3 = 10$$

$$y_4 = 0$$

$$y_5 = 11$$

Because any 3 out of 5 can be used to calculate the value, we know 3 of them are valid keys and 2 of them are invalid. But no one else know which of them are valid.

### 5.4.2 Public Keys Verify Algorithm

The sender can check the public keys in two ways:
5.5. NON-INTERACTIVE OBLIVIOUS TRANSFER PROTOCOL

First, if the sender knows the public key \( y_i, i = 1, 2, 3, 4, 5 \) and knows the \( x_i, i = 1, 2, 3, 4, 5 \), the sender can choose any three of them and calculate \( a_0, a_1, a_2 \). No matter which three of them he chooses, \( a_0, a_1, a_2 \) always have the same values.

Second, if the sender knows the \( a_0, a_1, a_2, x_i, i = 1, 2, 3, 4, 5 \) and the \( y_i, i = 1, 2, 3, 4, 5 \), he can check it more directly. He just proves if it satisfies the linear equations.

5.5 Non-Interactive Oblivious Transfer Protocol

Using the public key setup system in the previous section, the receiver obtains the public keys pairs \( \{x_i, y_i\}, i = 1, \cdots, n \) where the discrete logs of \( y_i \) for \( i \in U \) are not known to anyone, and his private keys \( s_j, j = 1, \cdots, m \).

In this section, we introduce the implementation of non-interactive oblivious transfer protocol using the public keys setup system.

The non interactive oblivious transfer protocol is show in Figure 5.1

Now we explain the non-interactive OT protocol in Figure 5.1 as follows:

- **Sender**
  - Obtains the receiver’s public key \( \{x_i, y_i\}, i = 1, \cdots, n \)
  - Randomly chooses \( t_1, \cdots, t_n \in \mathbb{Z}_p^* \), calculates \( \alpha_i = g^{t_i} \mod p, i = 1, \cdots, n \), and sends \( \alpha_1, \cdots, \alpha_n \) to the receiver.
  - Calculates \( z_i = M_i y_i^{t_i}, i = 1, \cdots, n \), and then sends \( z_1, \cdots, z_n \) to the receiver.

- **Receiver**
  - Calculates the public key \( y_j, j = 1, \cdots, m \) according to the private keys \( x_j, j = 1, \cdots, m \), \( y_j = g^{x_j} \). Then the receiver generates \( n \) public keys \( y_i, i = 1, \cdots, n \)
### Chapter 5. Non-Interactive Oblivious Transfer

<table>
<thead>
<tr>
<th>Sender (A)</th>
<th>Receiver (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public key</td>
<td>Public key $y_1, \cdots, y_n$</td>
</tr>
<tr>
<td>$y_1, \cdots, y_n$</td>
<td>Public key $x_1, \cdots, x_n$</td>
</tr>
<tr>
<td>$x_1, \cdots, x_n$</td>
<td>Secret key $j, s_j$</td>
</tr>
<tr>
<td></td>
<td>$j \in 1, \cdots, n$</td>
</tr>
<tr>
<td></td>
<td>$s_i \in s_1, \cdots, s_m$</td>
</tr>
</tbody>
</table>

Randomly choose

$t_1, \cdots, t_n \in_R \mathbb{Z}_q^*$

$\alpha_i = g^{t_i} \mod p$

$i = 1, \cdots, n$

\[ z_i = M_i y_i^{t_i} \mod p \]

$i = 1, \cdots, n$

\[ M_i = z_i / \alpha_i^{s_i} \mod p \]

$i \in 1, \cdots, n$

---

**Figure 5.1:** The Non-interactive OT protocol
5.6. SECURITY ANALYSIS

according to the method introduced last section.

- Upon receiving $\alpha_1, \cdots, \alpha_n$ and $z_1, \cdots, z_n$, the receiver can get the messages by calculating the $M_i = z_i/\alpha_i^{z_i}, i \in 1, \cdots, n$.

Thus B receives the $M_i$. He can not get the other $M_i$ for which he does not knows the secret key. A has succeeded in obliviously transferring the strings $M_i$. The transfer is indeed non-interactive transfer, because B sends nothing to A.

Let us analyze the Meaningfullness, Verifiability, $m$ out-of $n$ and Obliviousness of the protocol.

- Meaningfullness: from the protocol we can know if the receiver follows the protocol, he can get $m$ out of $n$ messages

- Verifiability: in the protocol, the sender can verify the public keys and the receiver knows which $m$ messages he got.

- $m$ out-of $n$: because only $m$ keys are given, the other $m - n$ keys can not do anything. The receiver gets only $m$ messages and not even a bit of the other.

- Obliviousness: because the receiver does not know which of the $m$ out of $n$ keys are valid, the sender can not predict which messages is going to be received.

5.6 Security Analysis

In this section, we analyze the security of the scheme. This scheme is based on the discrete logarithm problem. The protocol involves two parties, the sender and the receiver. Let us check the security of this protocol.

Claim 3 (Completeness) If the Receiver follows the procedure, the receiver can recover $m$ out of $n$ messages
CHAPTER 5. NON-INTERACTIVE OBLIVIOUS TRANSFER

This is obvious. Note that the sender can change the messages order randomly so that the receiver can not get the messages he wants. The order of the public keys are not changed and the receiver knows the indices. He has \( m \) corresponding private keys \( s_i, i \in \mathbb{K} \), and he knows which ones he should use to decrypt.

**Claim 4 (Soundness)** Both the sender and receiver can not cheat.

First, let us check the sender. The sender can cheat in two ways. One is that he gives less than \( m \) messages to receiver, another one is to find out which messages the receiver will receive.

If the sender gives less than \( m \) messages to the sender, the sender must receive more than two of the same messages so that the sender can find the the problem. On the other hand, because the sender does not know which public keys are associated with the receiver's private keys and which public keys are not associated with the receiver's private keys, the sender can not find which messages the receiver gets. So it is impossible for the sender to cheat in this scheme.

Next, let us analyze the receiver. After the receiver publishes the public key, everyone can check only \( m \) out of \( n \) are associated with his private keys, the other \( n - m \) are not associated with his private keys. If the receiver wants to use one calculated public key to generate the private key, this means he knows \( g \) and \( y = g^x \), and wants to calculate \( x \). This is a discrete logarithm problem.

The security of the protocol is also based on the Diffie-Hellman assumption: given \( g^x \) and \( g^y \), but neither \( x \) nor \( y \), it is hard to compute \( g^{xy} \). The receiver can only compute \( M_i \) for \( i \in \mathbb{K} \), but not the messages whose public keys \( y_i, i \in \mathbb{U} \). Given the public keys \( y_i \) and \( \alpha_i \) for \( i \in \mathbb{U} \), the receiver can not recover \( M_i \).

Note that we only consider both the sender and the receiver follow the protocol. Some of the cheats like the garbage the sender gives is beyond this protocol.
5.7 Check for Fraud

In the scheme above, the sender was assumed to be honest in that he always used the receiver's public keys in the encryption, and the transmission is secured so no one else can change it.

However, if the order of public keys are changed by the sender, or the order of the cipher text is changed by the adversary (or by accident) during the transmission, the receiver will not be able to find the fraud in the case that the messages are unrecognized strings.

One way to solve this problem is by using hashed messages when the sender is honest. Since the hash function is computed by the sender, the receiver can not find the fraud if the sender attempt to cheat.

5.7.1 Verifiable Encrypted Message Using the Double Encryption

In order to make the encryption verifiable, we use double encryption. Now we modify the scheme so that the receiver can check if the sender correctly constructed the encrypted messages.

The modified protocol is listed in Figure 5.2

The protocol is described as follows:

- The Sender

  - Randomly chooses $t_1, \cdots, t_n \in_R \mathbb{Z}_q^*$ and $t_1', \cdots, t_n' \in_R \mathbb{Z}_q^*$
  - Calculates $\alpha_i = g^{t_i} \mod p$ and $\alpha_i' = g^{t_i'} \mod p$
  - Calculates $z_i = M_i y_i^{t_i}$ and $z_i' = M_i y_i^{t_i'}$ for $i = 1, \cdots, n$
<table>
<thead>
<tr>
<th>Sender (A)</th>
<th>Receiver (B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Public key</td>
<td>Public key $y_1, \cdots, y_n$</td>
</tr>
<tr>
<td>$y_1, \cdots, y_n$</td>
<td>Public key $x_1, \cdots, x_n$</td>
</tr>
<tr>
<td>$x_1, \cdots, x_n$</td>
<td>Secret key $j, s_j$</td>
</tr>
<tr>
<td>$t_1, \cdots, t_n \in_R \mathbb{Z}_q^*$</td>
<td>$j \in 1, \cdots, n$</td>
</tr>
<tr>
<td>$t'_1, \cdots, t'_n \in_R \mathbb{Z}_q^*$</td>
<td>$s_i \in s_1, \cdots, s_m$</td>
</tr>
<tr>
<td>$\alpha_i = g^{t_i} \mod p$</td>
<td></td>
</tr>
<tr>
<td>$\alpha'_i = g^{t'_i} \mod p$</td>
<td></td>
</tr>
<tr>
<td>$i = 1, \cdots, n$</td>
<td></td>
</tr>
</tbody>
</table>

$$\alpha_1, \cdots, \alpha_n, \alpha'_1, \cdots, \alpha'_n$$

$$z_i = M_i y_i^{t_i} \mod p$$

$i = 1, \cdots, n$

$$z'_i = M_i y'_i^{t'_i} \mod p$$

$i = 1, \cdots, n$

$$\alpha_1, \cdots, \alpha_n, \alpha'_1, \cdots, \alpha'_n$$

$$M_i = z_i / \alpha_i^{s_i}$$

$$M'_i = z'_i / \alpha'_i^{s_i}$$

$i \in 1, \cdots, n$

Figure 5.2: The Modified Non-Interactive OT protocol
5.7. **CHECK FOR FRAUD**

- Sends to Receiver $i = \alpha_1, \cdots, \alpha_n, \tilde{i} = \alpha'_1, \cdots, \alpha'_n, \tilde{i} = z_1, \cdots, z_n$, and $i = z'_1, \cdots, z'_n$

- **Receiver**

- Decryps $M_i = \frac{z_i}{a_i}$ and $M'_i = \frac{z'_i}{a'_i}$ to recover the $n$ messages.
- Checks the messages obtained from two separate encryption are equal.

Let us analyze the security of the scheme:

- **Completeness** The completeness of the protocol is obvious: if the sender has used the receiver’s public keys in encryption, the receiver can find the messages from two different encryption are equal.

- **Soundness** If the different public keys $y' = g^t$ are used in the encryption, the messages from two different decryption will not be equal because $t \neq t'$

5.7.2 **Verifiable Encryption by Reconstructing the Private Keys**

The drawback of this scheme is the compromise of the computational efficiency. We now develop a verifiable encryption by reconstructing the private keys:

Select private keys, $s_i \in \mathbb{Z}_q$, and some integers $s'_i \in \mathbb{Z}_q$, such that they can satisfy:

$$s_i s'_i (mod) q = s_i$$

It is not hard to find that we can select any $s_i$ and $s'_i$ that satisfy $s_i (s'_i - 1) = q$ and $s'_i \neq 1$. The receiver needs to keep $s_i$ and $s'_i$ secret. The public keys are still published.
The correctness of the encryption can be verified during the decryption. The receiver can decrypt the oblivious transferred messages using the following two methods:

- **Method 1**: Compute \( \frac{s'_i}{a_i^{s_i}} = M'_i \) and then remove \( s'_i \).

- **Method 2**: Compute \( \frac{s_i}{a_i^{s_i}} = M_i \)

The receiver can then check if the two messages are same. Let us analyze the security of the scheme:

- **Completeness**: The completeness is obvious.

- **Soundness**: If the sender has not correctly used the receiver's public keys in the encryption, let's say he uses \( g^{s_i} \), then the receiver can immediately find the fraud.

  - **Method 1**: Calculates \( \frac{(M_i g^{s_i})^{s_i}}{a_i^{s_i}} = M'_i g^{t_i(s_i - s'_i)} \) and then removes \( s'_i \) from the message. Receiver gets \( M_i g^{t_i(s_i - s'_i)} \).

  - **Method 2**: Calculates \( \frac{M_i g^{s_i}}{a_i^{s_i}} = M_i g^{t_i(s_i - s_i)} \)

The receiver can find it is not equal.

### 5.8 Concluding Remarks

We have proposed the non-interactive \( ^m_n \) - OT schemes in this chapter. This scheme also covers all types of non interactive OT, i.e., \( n \) can be arbitrary number that is greater than one and \( 0 < m < n \). If \( m = 1 \), it is the \( ^1_n \) - OT schemes. If \( n = 2, m = 1 \), it is the one out of two non interactive OT.
5.8. CONCLUDING REMARKS

The \( m \) out of \( n \) oblivious transfer can also be used in many applications in electronic commerce. It also can be used in other applications like secrets distribution, key distribution and so on.
Chapter 6

Maintaining the Privacy in Buying Digital Goods

6.1 Introduction

In E-commerce, we have two sides (the buyer and the vendor) and two activities (money transfer i.e., payment and good transfer). Traditionally, we use the on-line payment system in which the buyer can make the payment to the vendor on-line immediately based on the account the buyer has. In order to protect the buyer’s privacy, some other payment systems, i.e., the token-based system was developed [12, 55, 56, 57]. In this anonymous payment system, it is computationally feasible to identify the buyer in a given transaction. An anonymous token-based system is traditionally referred to as an electronic cash (e-cash) system. But, in such a system we can not hide what goods are being bought and when. Another approach such as [4] were developed to provide a alternative way to meet such needs.

In this chapter, we use another approach to meet such needs, in which we maintain the buyer privacy from the goods transfer activity, not like the e-cash from the
money transfer activity.

Let us consider such a scenario: You purchase some digital goods on the Internet, and do not want the vendor know what you are buying and when you buy. There are many digital photos and pictures, and you purchase some of them. Or there are a lot of film channels for you choose. In both cases you want to choose some of them and you do not want the vendor know what you buy or subscribe to.

In this chapter, we develop the $m$ out of $n$ priced oblivious transfer protocol and discuss that one out of $n$ priced oblivious transfer protocol can be seen as its special case. We combine the $m$ out of $n$ priced OT with the sell digital goods protocol developed in [4] to realize maintaining the privacy in buying digital goods. Also, we give the buying $m$ items or subscribe $m$ channels protocol.

This chapter is organized as follows. In the section 2, we give the problem specification and security requirement and the tools we will use in the protocol. In section 3 we describe the details of setting up the protocol. In section 4, we discuss the protocol for subscription. In section 5, we introduce how to avoid the buyer increasing the balance and buying more expensive items than his current balance using the zero knowledge proofs and the condition disclosure technology. In section 6, we give the protocol details to realize the $m$ out of $n$ priced oblivious protocol and its special case one out of $n$ priced oblivious transfer protocol. Finally, the last section is the conclusion.

6.2 Preliminaries

6.2.1 Problem Specification

Our goal is to construct a protocol between a buyer and a vendor which allows the buyer and the vendor to engage in multiple transactions where the buyer can
purchase several items of his choice. We describe the functionality as follows:

Initialization: at time 0, the buyer makes a pre-payment to the vendor to initializes his balance.

Main Protocol: At time \( t, t = 1, 2, 3, \cdots \)

- The vendor can choose a database \( X = (x^0, x^1, \cdots, x^{n-1}) \) of \( n \) items for sale. Public information \( P \) contains a price list \( P = (p^0, p^1, \cdots, p^{n-1}) \) which can be used to identify the items. The item \( x^i \) is priced \( p^i \).

- The buyer can make the following decision:

  - \textit{Buy} \( m \) items from \( n \ (m < n) \) items for sale. The total price of the \( m \) items is \( p_m = p_0 + p_1 + \cdots + p_{m-1} \). If the buyer's remaining balance is sufficiently large than the total price \( p_m \), the buyer can receive the \( m \) items he chooses from the \( n \) items in the vendor sales.

  - \textit{Subscribe} to \( m \) out of \( n \) channels: by subscribing, the buyer indicates that he wishes to continue buying \( m \) channels until overriding the subscription with a new request. As with buying items, if the remaining balance of the buyer is larger than the total price of the subscribing channels, the buyer can successfully subscribe to the channels. We suppose that through the subscription, the buyer is charged the total price of the channels he subscribed when initiating the subscription.

  - \textit{Unsubscribe} the buyer terminates the previously subscribed channels.

  - \textit{Do nothing} just maintain the subscribed channels if it exists, otherwise stay idle
6.2.2 Security Requirement

The two party protocol is considered secure if it satisfies correctness, buyer security and vendor security.

- **correctness** If both the buyer and vendor are honest, the buyer can output the $m$ items he choose from $n$ items at the end of the transaction.

- **buyer’s security** A malicious vendor should not know what items the honest buyer chooses.

- **Vendor’s Security** A malicious buyer should not buy more than what the buyer’s initial deposit allows.

6.2.3 Tools

- **Homomorphic** Informally, an encryption scheme can be said to be homomorphic encryption if:
  
  - The plain texts are taken from a group $(H, +)$, $H$ is a group of large prime order $Q$.
  
  - We can efficiently compute a random encryption of $h_1 + h_2$ from encryption of group elements $h_1, h_2$.

- **PIR** A Private Information Retrieval Protocol allows a user to retrieve items from a database while keeping the item private.

- **$m$ out of $n$ chosen Oblivious Transfer** Oblivious transfer was introduced in [53], with an implementation based on the factoring problem. Roughly speaking, the $m$ out of $n$ chosen oblivious transfer means that the sender has $n$ messages, the receiver choose $m$ messages from the $n$ messages, the
sender does not know which message the receiver gets, the receiver only gets \( m \) messages and knows nothing about other messages.

- **Zero knowledge proof** Zero knowledge proof is a protocol which allows one party to convince another party of some fact without revealing any information about the proof. There are two forms of zero-knowledge proof system, one is an interactive proof system and another one is a non-interactive proof system.

- **Conditional disclosure and secrets sharing** Condition disclosure of secrets introduced in [14], also is reduced to generalized secret-sharing. Informally, an input string \( y \) is partitioned among \( k \) players. No player knows the entire string \( y \), and at least one player hold the secret \( s \). The goal is that the player discloses the secret \( s \) to an external party, subject to the condition of the players input.

### 6.3 System Set up

In this section we give the two party protocol for the oblivious transfer protocol and then we explain the protocol in detail.

First, we describe the protocol briefly. In the initialization phase, the vendor creates the item database and every item is marked with a distinct price. The buyer generates his public key system using the key generator algorithm \( G \), and deposits some money to the vendor. The vendor encrypts the initial balance using the buyer’s public key.

During the transaction, the buyer selects some items using the PIR technique, and then encrypts the selected item’s price using probabilistic encryption and sends to vendor. The vendor does not know which items they are because of the probabilistic encryption. Then the vendor reduces the buyer’s balance using the homomorphic
encryption technology, at the same time the vendor checks if the total selected item’s price is between zero and the balance using the zero knowledge proof or the condition disclosure method. Next the vendor encrypts all the item price using the $m$ out of $n$ priced OT protocol. Finally, the buyer decrypts the information and gets the corresponding items he paid for.

### 6.3.1 The Solution Protocol

The solution protocol is shown in Figure 6.1

The protocol listed is described as following:

- **The Buyer Initialization** The buyer applies the Key Generator $G$ to sample the keys: the private key $k_{sk}$ and the public key $k$, saves some money $b^0$ to the vendor’s account, sets the random receipt $u_i^t$ as zero, which is used for checking if the buyer’s behavior is correct, and finally sends the public key to the vendor.

- **The Vendor Initialization** The vendor encrypts the balance $b^0$ according to the amount the buyer sends to him using the public key the buyer sent, and sets the random receipt $u_i^t$ as zero.

- **Transaction**

  - **Buyer Setup** Firstly, the buyer sets the receipt $u = u_i^{t-1}$. Secondly, he encrypts the current balance as $E_k(b_{i-1}), \ldots, E_k(b_1)$, $b_{i-1}, \ldots, b_0$ as the binary representation of current balance. Then the buyer chooses $m$ items and calculates the total price $p_i^t = \sum p_{si}$, $si = 1, \ldots, m$. $E_k(p_{i-1}), \ldots, E_k(p_{i-1}), p_{i-1}, \ldots, p_0$ as the binary representation of the total price $p_i^t$. Finally, the buyer sends the encryption of the total price and the encryption of the item price he selected to the vendor.
6.3. SYSTEM SET UP

The buyer(B)  The Vendor(V)

Key Generator $G \rightarrow (k, sk)$  Encrypt $b^0, E_k(b^0)$

Initialize the balance $b_0$  Set $u^0$ as zero string

$k, b^0$  

For $t = 1, 2, \ldots, t$ Do

$E_k(u) = E_k(u^{t-1})$

$E_k(b_{t-1}), \ldots, E_k(b_0)$

$b_{t-1}, \ldots, b_0$ as the binary representation of current balance

$p^t_{st} = \sum p_{si}, si = 1, \ldots, m$

$E_k(p_{t-1}), \ldots, E_k(p_0)$

$p_{t-1}, \ldots, p_0$ as the binary representation of total price $p^t_{st}$

$p^t_{st}, E_k(p_{st}), (si=1, \ldots, m)$  $E_k(p) = E_k(\sum_j (p^j_{si}))$

$E_k(b^j) = E_k(b^{j(t-1)}) - E_k(p)$  update the current balance under condition

$(b = \sum_j (b^j_{si})) \land (0 \leq p \leq b)$

$\land (u = u^{t-1})$

$m^j = x^j + v^j, j = 1, \ldots, n$

$v^j = \prod_{si=1}^m \alpha^j (E_k(p^j) - E_k(p^j_{si}))$

disclose under $p^j_{si} = p_{si}$,

$si = 1, \ldots, m$

compute $m^1, \ldots, m^n$

compute $x^j_{si} = D_{sk}(m^j_{si})$,  

$si = 1, \ldots, m, j = 1, \ldots, n$

Next $t$

Figure 6.1: Maintain privacy in buying digital good OT protocol
- **Vendor Updates balance** The vendor calculates the total price of the current balance according to the binary representation. Then he updates the balance $E_k(b^{(t)}) = E_k(b^{(t-1)}) - E_k(p)$, under the conditions of following; $(b = \sum_j(b_j^2)) \land (0 \leq p \leq b) \land (u = u^{t-1})$.

- **Vendor Encryption** The vendor encrypts the all the massage $m^j = x^j + v^j$ which can be decrypted under the condition $p_{si}^j = p_{si}$, $si = 1, \ldots, m$, because $v^j = \prod_{s=1}^{m} \alpha^j(E_k(p^j) - E_k(p_{si}^j))$. Then he sends the $m_j, j = 1, \ldots, n$ to the buyer.

- **Buyer Decryption** The buyer can decrypt the message he paid for, $x_{si}^j = D_{sk}(m_{si}^j)$, $si = 1, \ldots, m$, $j = 1, \ldots, n$

- **Do next transaction**

### 6.3.2 Some Consideration

Let us analyze the security of the above protocol.

- **Correctness** From the protocol above, we can see that if both the buyer and the vendor follow the protocol, the buyer can get the $m$ items he buys, and the vendor can get the money the buyer paid for the items.

- **Buyer’s Security** The vendor only can see the encryption of the item price and the total price. And he can not know the exact item price. The encryption being used is probabilistic, as it applies some randomized computation to the message and the public key so that each message can have exponentially many possible encryptions, one of which is chosen at random. This property is called semantic security. It is this semantic property that makes it impossible to distinguish between the encryption of the different messages. So even the
malicious vendor still can find the real price in each transaction. Because the
item price is distinct, the buyer will get exactly the \( m \) items he paid for.

- **Vendor's Security** From the protocol, we can see the buyer can not buy
more than he initially deposits. This is because, firstly the buyer gets the \( m \)
items he paid for and the the balance is reduced by exactly the total price he
chooses. Secondly, if the buyer’s balance is less than the total price of the \( m \)
items, the vendor will not update the current balance, and the transaction will
be cancelled. At the same time, the recorder \( u \) will not be updated and the
buyer can not do any more transactions. Next, the buyer can not increase his
account as the protocol will check the price \( (p > 0) \) at every transaction. In
conclusion, the above protocol is security for the vendor.

### 6.4 Subscription

#### 6.4.1 Set up the Protocol

We gave the buying items protocol last section. In this section we present the
subscribing and un-subscribing protocol, and then we explain the protocol in detail.

We describe the protocol briefly. The difference between buying and subscribing
is that in subscribing, the buyer will pay the money or the vendor will deduct the
balance after a regular billing period, and the buyer can un-subscribe. The protocol
is mostly the same except that: the buyer should give the subscribing period, the
vendor deducts the balance from the buyer’s account regularly, and the vendor can
cancel the subscription after he receives the request from the buyer. The other parts
of the protocol are same as the buying protocol.

The subscribing protocol is shown in Figure 6.2

The protocol initialization part is the same as buying items. The transaction
The buyer (B)

Key Generator $G \rightarrow (k, sk)$
Initialize the balance $b^0$

For $t = 1, 2, \ldots, T$ Do
$E_k(u) = E_k(u^{t-1})$
$E_k(b_{t-1}), \ldots, E_k(b_0)$
$b_{t-1}, \ldots, b_0$ as the binary representation of current balance

$p^t_{st} = \sum p_{si}, si = 1, \ldots, m$
$E_k(p_{t-1}), \ldots, E_k(p_0)$
$p_{t-1}, \ldots, p_0$ as the binary representation of total price $p^t_{st}$
$\tau$ the length buyer subscribes

$E_k(p) = E_k(\sum_{j=1}^{m} p^j_{si})$
$m^j = x^j + v^j, j = 1, \ldots, n$
$v^j = \prod_{si=1}^{m} \alpha^j (E_k(p^j_{si}) - E_k(p))$
disclose under $p^j_{si} = p_{si}$,
$si = 1, \ldots, m$
compute $m^1, \ldots, m^n$

compute $x^j_{st} = D_{sk}(m^j_{st})$, $si = 1, \ldots, m, j = 1, \ldots, n$

$mt$ for $T = 0, 1, \ldots, \tau$
If $(0 \geq \tau p) \lor (\tau p > b) \lor (u \neq u^{t-1})$

If $(mt = 0) \lor (T = \tau)$
$E_k(b^{(t)}) = E_k(b^{(t-1)}) - TE_k(p^t_{st})$
Unsubscribing

Next $T$

The Vendor (V)

Encrypt $b^0, E_k(b^0)$
Set $u^0$ as zero string

Figure 6.2: Maintain privacy in subscribing digital good OT protocol
6.4. SUBSCRIPTION

part is described as follows:

- **Buyer set up** Firstly, same as buying items, the buyer sets the receipt \( u = u^{t-1} \). Secondly, he encrypts the current balance as \( E_k(b_{i-1}), \ldots, E_k(b_0), b_{i-1}, \ldots, b_0 \) as the binary representation of current balance. Then the buyer chooses \( m \) channels and calculates the total price \( p_i = \sum p_{si}, s_i = 1, \ldots, m. \ E_k(p_{i-1}), \ldots, E_k(p_0), p_{i-1}, \ldots, p_0 \) as the binary represent of the total price \( p_i \). Finally, the buyer sends the encryption of the total price and the encryption of the item price he selected to the vendor.

- **buyer Subscribing** The buyer chooses the length he is going to subscribe to the channel \( \tau \) and sends it to the vendor. The honest buyer should let \( \tau \leq (b/p_d) \) in order not to spend more money than his balance. Of course, usually, it is the period he wants to use the channels.

- **Vendor Section** Same as buying Items, the vendor will use the priced \( m \) out of \( n \) oblivious transfer protocol to send the channels key to the buyer. At the same time, he must make sure the data (includes balance, total price and the length the buyer subscribe to) he receives from buyer is valid, and the money the buyer wants to spend is less than his deposit.

- **Maintain a subscription** After subscribing, the vendor knows the buyer is honest, and the buyer can use the channels he subscribed. For every billing period, the vendor can check if the buyer un-subscribed from the channels, and he can decide to cancel the valid keys for the buyer.

- **Un-subscribing**

  If the buyer decides to unsubscribe the channels he subscribed to, he can send the un-subscribing message to the vendor. If the vendor receives the un-
CHAPTER 6. MAINTAINING THE PRIVACY IN BUYING DIGITAL GOODS

subscribing order, he will cancel all the keys for the buyer, and deducts the
money the buyer spent from his account.

6.4.2 Security Analysis

In this section, we analyze the security of above protocol.

- **correctness** From the protocol above, it is easy to see that if both the buyer
  and vendor follow the protocol, the buyer can get the $m$ channels he subscribes
to and he can unsubscribe at any time he wants. Also the vendor can get the
money the buyer paid for the channels.

- **Buyer's Security** The buyer only sends the vendor the encrypted channels
  price. The vendor does not know which channels the buyer subscribed.

- **Vendor's Security** The vendor can get the money for the buyer's subscribed
  channels. The buyer can not spend more money than his deposit. He can not
  add more money to his account.

6.5 Zero-Knowledge Proofs and Condition Disclosures

In the above protocol, we have to prove the price $p_i$ is not negative and less than
the current balance $b_c$ because we know if the price $p_i$ is not negative, the buyer can
increase his or her balance. If the price $p_i$ is less than the current balance $b_c$, the
buyer can get more expensive items and pay less. There are two ways to avoid this
problem, one way is to use zero knowledge proofs and another is to use condition
disclosure technology. In this section, we introduce zero knowledge proof and the
condition disclosures.
6.5. ZERO-KNOWLEDGE PROOFS AND CONDITION DISCLOSURES

6.5.1 Zero-Knowledge Proofs

Zero knowledge proof is a protocol which allows one party to convince another party of some fact without revealing any information about the proof. There are two forms of zero-knowledge proof systems, one is the interactive proof system and another one is the non-interactive proof system. In our case, the buyer should prove the item price $p^i$ he chooses is less than the current balance $b_c$ and is greater than zero.

This means we need to prove that a committed number lies in a specific interval. There are three methods nowadays to prove a committed integer is in a specific interval. These methods are given in [58], here we give a different presentation.

- the first method allows you to prove that the bit-length of the committed number is less or equal to a fixed value $k$, and then this committed number belongs to $[0, 2^k - 1]$

- the second method only can prove the committed number belong to a much larger interval, even though it is efficient.

- the third method developed by Fabrice Boudot can prove a committed number $x \in I$ belongs to $I$.

Classical Proof

This protocol can prove a committed number $x \in I = [0, b]$, belongs to $[0, 2^k - 1]$, where the binary length of $b$ is $k$.

Let us suppose:

\[ p \text{ is a large prime number} \]
\[ q \text{ is such that } q|p - 1 \]
g and h are elements of order q in \( \mathbb{Z}_p^* \).

The committed number \( x \) is denoted by \( E(x, r) = g^x h^r \mod p \). Where \( r \) is randomly selected over \( \mathbb{Z}_p^* \).

Let binary representation \( x = x_02^0 + x_12^1 + \cdots + x_{k-1}2^{k-1} \) where \( x \in \{0, 1\} \) and \( i = 0, 1, \cdots, k - 1 \).

The protocol is shown in figure 6.3

<table>
<thead>
<tr>
<th>The Prover(P)</th>
<th>The Verifier(V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>set ( E(x_i, r_i) )</td>
<td></td>
</tr>
<tr>
<td>( i = 0, 1, \cdots, k - 1 )</td>
<td></td>
</tr>
<tr>
<td>( r = \sum_{i=0,1,\ldots,k-1} r_i )</td>
<td></td>
</tr>
<tr>
<td>( E(x_i, r_i) )</td>
<td></td>
</tr>
<tr>
<td>( \prod_{i=0,1,\ldots,k-1} E(x_i, r_i) )</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.3: Classical Proof protocol

The prover sets \( E(x_i, r_i) \) and proves for all \( i \) that the number hidden by \( E(x_i, r_i) \) is either 0 or 1 by proving that she knows either a discrete logarithm of \( E(x_i, r_i) \) in base \( h \), using proofs of knowledge of a discrete logarithm.

**BCDG Proof**

This protocol proves that a committed number \( x \in I \) belong to \( J \), the expansion rate is equal to 3.
Let us suppose:

\[ t \quad \text{is a security parameter} \]
\[ p \quad \text{is a large prime number} \]
\[ q \quad \text{is such that } q \mid p - 1 \]
\[ g \text{ and } h \quad \text{are elements of order } q \text{ in } \mathbb{Z}_p^* \]

The committed number \( x \) is denoted by \( E(x, r) = g^x h^r \mod p \), where \( r \) is randomly selected over \( \mathbb{Z}_p^* \).

The protocol is shown in figure 6.4

- The prover randomly selects \( \omega_1 \in_R [0, b] \) and \( \omega_2 = \omega_1 - b \), then randomly selects \( \eta_1 \in_R [0, q - 1] \) and \( \eta_2 \in_R [0, q - 1] \) and then calculates \( W_1 = g^{\omega_1} h^{\eta_1} \mod p \), \( W_2 = g^{\omega_2} h^{\eta_2} \mod p \) and finally send \( w_1, w_2 \) to verifier.

- Verifier sends challenge \( c \in_R 0, 1 \) to prover

- if challenge \( c = 0 \), prover sends \( \omega_1, \omega_2, \eta_1, \eta_2 \) to verifier

- Verifier checks \( W_1 = g^{\omega_1} h^{\eta_1} \mod p \) and \( W_2 = g^{\omega_2} h^{\eta_2} \mod p \)

- if challenge \( c = 1 \), prover sends \( x + \omega_j, r + \eta_j, j \in 1, 2 \) to verifier

- Verifier checks \( W_j = g^{\omega_j} h^{\eta_j} \mod p \) and \( x + \omega_j \in [a, b] \)

**Fabrice Boudot proof**

The Fabrice Boudot proof scheme is much more efficient than the other two. In this scheme the expansion rates is equal to 1.

In this scheme, we suppose:
The Prover (P)  The Verifier (V)

\[ \omega_1 \in_R [0, b] \]
\[ \omega_2 = \omega_1 - b \]
\[ \eta_1 \in_R [0, q - 1] \]
\[ \eta_2 \in_R [0, q - 1] \]
\[ W_1 = g^{\omega_1 \eta_1} \mod p \]
\[ W_2 = g^{\omega_2 \eta_2} \mod p \]
\[ c \in_R 0, 1 \]

if \( c = 0 \)

\[ \omega_1, \omega_2, \eta_1, \eta_2 \]

check \( W_1 = g^{\omega_1 \eta_1} \mod p \)
\[ W_2 = g^{\omega_2 \eta_2} \mod p \]

if \( c = 1 \)

\[ x + \omega_j, r + \eta_j, j \in 1, 2 \]

check \( W_j = g^{\omega_1 \eta_j} \mod p \)
\[ x + \omega_j \in [a, b] \]

Figure 6.4: BCDG Proof Protocol
6.5. ZERO-KNOWLEDGE PROOFS AND CONDITION DISCLOSURES

\[ s \text{ is a security parameter} \]
\[ n \text{ is a composite number} \]
\[ g \text{ and } h \text{ are elements of order } q \text{ in } \mathbb{Z}_p^* \]
\[ r \text{ is randomly selected in } [-2^s n + 1, 2^s n - 1] \]

The committed number \( x \) is denoted by \( E(x, r) = g^x h^r \mod p \). Then to prove \( x \in [a, b] \).

Firstly, from the CFT proof we know that if a commitment \( x \in [0, b] \) then it is easy to prove \( x \in [-2^{t+l} b, 2^{t+l} b] \).

Secondly, we can prove \( x \in [a - 2^{t+l+1} \sqrt{b-a}, b + 2^{t+l+1} \sqrt{b-a}] \)

- Let \( x_1^2 = x - a \), so \( x_1 < 2 \sqrt{x - a} \)

- \( x_1 < 2 \sqrt{b-a} \), because \( b > a \)

- Randomly select \( r_1, r_2 \in [0, 2^s n - 1] \) and \( r = r_1 + r_2 \)

- Compute \( E_1 = E(x_1^2, r_1) \) and \( E_2 = E(r, r_2) \)

- Then we can prove \( x_1^2 \in [-2^{t+l} b, 2^{t+l} b] \), by the CFT proof

- Then we have \( x \in [a - 2^{t+l+1} \sqrt{b-a}, b + 2^{t+l+1} \sqrt{b-a}] \), because \( x_1^2 = x - a \)

Thirdly, we can prove \( x \in [a, b] \)

- Let \( x' = 2^T x \)

- From the above scheme, we can prove that \( x' \in [2^T a - 2^{t+l+T/2+1} \sqrt{b-a}, 2^T b + 2^{t+l+T/2+1} \sqrt{b-a}] \)

- Let \( T \) big enough so that \( 2^{t+l+T/2+1} \sqrt{b-a} < 2^T \)
CHAPTER 6. MAINTAINING THE PRIVACY IN BUYING DIGITAL GOODS

- Then \( x' \in [2^T a - 2^T + 1, 2^T b + 2^T - 1] \)
- Let \( \varepsilon = 1 - 1/2^T \) then \( x \in [a - \varepsilon, b + \varepsilon] \)
- Then verifier can know \( x \in [a, b] \)

About CFT Proof

Last section we mention the CFT proof. We introduce it here. This protocol was introduced by Chan, Frankel and Tsounis. It proves that a committed number \( x \in I \) belong to \( J \), and the expansion rate is equal to \( 2^{t+l+1} \)

In this scheme, it is supposed:

- \( s, t, l \) is a security parameter
- \( n \) is a composite number
- \( g \) and \( h \) are elements of order \( q \) in \( \mathbb{Z}_p^* \)
- \( H \) is a hash function which outputs 2t-bit strings
- \( r \) is randomly selected in \([-2^s n + 1, 2^s n - 1]\)

The committed number \( x \) is denoted by \( E(x, r) = g^x h^r \mod p \). Then to prove \( x \in [0, b] \).

The proof protocol is shown in Figure 6.5:

6.5.2 Condition Disclosures of Secrets

Condition disclosure of secrets introduced in [14], is also reduced to generalized secret-sharing. The secrets \( s \) is known to at least one player, and the external party does not know the secret. The goal is that the players disclose the secrets to the
The Prover (P) \hspace{2cm} \text{The Verifier (V)}

Randomly $\omega \in_R [0, 2^{t+1}b - 1]$

$\eta \in_R [-2^{t+1}+s_n + 1, 2^{t+1}+s_n - 1]$

Compute $W = g^\omega h^n \mod n$

$C = H(W)$

$c = C \mod 2^t$

$D_1 = \omega + xc$

$D_2 = \eta + rc$

\[c, D_1, D_2\rightarrow\]

check $D_1 \in [cb, 2^{t+1}b - 1]$

check $C = h(g^{D_1} h^{D_2} E^{-c})$

Figure 6.5: CFT proof protocol

external party under some conditions input by the players. If the condition is met, the external party can learn the secret $s$, otherwise he learn nothing.

The protocol we used needs to prove $0 \leq p \leq b$. We can use the conditional disclosures and chains technique. At the $t$-th transaction, the vendor samples random mask $v^t$ and random receipt $u^t$. The vendor will disclose the $u$ and $v$ under the condition $0 \leq p \leq b$.

The $v^t$ is used to mask the secrets $x^i$ in our case. The vendor lets the buyer retrieve $x^i + v^t$ instead of $x^i$. The buyer can not get the $x^i$ without disclosing the $v^t$.

The $u^t$ is used as a receipt for future interaction. If the buyer does not act correctly, he will not do any more transactions.
6.6 Oblivious Transfer

We introduce the details of the priced oblivious transfer protocol in this section. Firstly, we give the \( m \) out of \( n \) priced oblivious transfer protocol and explain it in detail. Then we give the one out of \( n \) priced oblivious transfer as one of the special cases of \( m \) out of \( m \) oblivious transfer protocol.

6.6.1 \( m \) out of \( n \) Oblivious Transfer

We present the \( m \) out of \( n \) priced oblivious transfer protocol in this section. We give the protocol first and then explain it in detail.

The protocol is shown in Figure 6.6

The protocol listed is described as follows:

- **Set up the Protocol by Buyer**

  Let \( q \) be the prime such that the discrete log problem in \( \mathbb{Z}_q \) is intractable, and let \( \alpha, g \in \mathbb{Z}_q^* \) be a primitive element, then we define

  \[
  \beta = \alpha^a (mod \ q)
  \]

  The value \( q, g, \alpha, \beta \) are public, and keep \( a \) secret.

  The buyer chooses a secret random number \( r_i, i = 1, \cdots, m \) and encrypts the item prices which he or she wants to buy.

  \[
  E(p_i) = g^{p_i\beta^{r_i}} (mod \ q)
  \]

  \((i = 1, \cdots, m) \) and \( P_i \in P_j \). \( P_j \) are the item prices which the vendor sells. \( j = 1, \cdots, n \)

  Finally, the buyer sends the random encrypted item prices to the vendor.
6.6. OBLIVIOUS TRANSFER

The buyer (B)                                      The Vendor (V)

\[ \beta = \alpha^k \mod q \]
\[ E(p_i) = g^{p_i \beta^{r_i}} \]
\[ y_i = \alpha^{r_i} \]
\[ r_i \text{ Random number} \]
\[ p_i \in p_j \]
\[ i = 1, \cdots, m \]
\[ j = 1, \cdots, n \]

\[ E(p_j) = g^{p_j \beta^{r_j}} \]
\[ y_j = \alpha^{k r_j} \]
\[ j = 1, \cdots, n \]

\[ E(p_i), \beta \]

\[ E(b_{ji}) = g^{k(p_j - p_i) + r_i + r_j} \beta^{kr_{jb}} \]
\[ = \left( \frac{g^{r_j}}{g^{p_j}} \right)^k g^{x_i} \beta^{kr_{jb}} \]
\[ = \left( \frac{E(p_j)}{E(p_i)} \right)^k \left( \frac{g^{r_j}}{g^{p_j}} \right)^k g^{x_i} \beta^{kr_{jb}} \]
\[ = \left( \frac{E(p_j)}{E(p_i)} \right)^k \left( \beta^{r_v} \right)^k g^{x_i} \]
\[ = \left( \frac{E(p_j)}{E(p_i)} \right)^k \left( \beta^{r_v} \right)^k g^{x_i} \]
\[ r_b = r_i + r_j + r_{jb} \]

Random number
\[ y_{jb} = \alpha^{r_{jb}} \]
\[ j = 1, \cdots, n \]
\[ i = 1, \cdots, m \]
\[ y_k = \alpha^k \]

\[ D(b_{ji}) \]
\[ = \left( \frac{g^{kr_{kb}}}{g^{p_i \beta^{r_{kb}}}} \right)^k \]
\[ = \left( \frac{g^{r_k}}{g^{p_i \beta^{r_{kb}}}} \right)^k \]
\[ = \left( \frac{g^{r_k}}{g^{p_i \beta^{r_{kb}}}} \right)^k \]
\[ i = 1, \cdots, m \]
\[ j = 1, \cdots, n \]
\[ (p = p_j) \]

Figure 6.6: m out of n Priced OT protocol
• Set up the Protocol by Vendor

The vendor encrypts the price with the buyer’s public key $\beta$, and $g$. The vendor chooses a secret random number $r_j$, $j = 1, \cdots, n$). Then calculates $E(p_j)$ and $y_j$ as follows:

$$\begin{align*}
E(p_j) &= g^{p_j} \beta^{r_j} \mod q \\
y_j &= \alpha^{kr_j} \mod q
\end{align*}$$

$y_j$ is used for the buyer to decrypt the message.

• Vendor Encrypts the item message the buyer pays

After the vendor receives the randomly encrypted item priced from the buyer, he can encrypt the message according to the homomorphic encryption.

Because the vendor knows the encryption of the item prices that the buyer chose and because he encrypted all the item prices with the buyer’s public key, vendor can now calculate the message as follows:

$$E(b_{ji}) = \left( \frac{E(p_j)}{E(p_i)} \right)^k \beta^{\tau_j} g^{x_j} (\mod q)$$

$r_b$ is the secret random number.

Because

$$\begin{align*}
E(p_i) &= g^{p_i} \beta^{r_i} \mod q \\
E(p_j) &= g^{p_j} \beta^{r_j} \mod q
\end{align*}$$

Then we have the following:

$$E(b_{ji}) = \left( \frac{g^{p_j}}{g^{p_i}} \right)^k g^{x_j} \left( \frac{\beta^{r_j}}{\beta^{r_i}} \right)^k g^{x_j} \beta^{r_b}$$

$$= \left( \frac{g^{p_j}}{g^{p_i}} \right)^k (\beta^{r_j-r_i+r_b})^k g^{x_j}$$
6.7. CONCLUSION

Now we let \( r_{jb} = r_j - r_i + r_b \)

Then \( E(b_{ji}) = g^{k(p_j - p_i) + x_j b^{kr_{jb}}} \)

Next, the vendor calculates:

\[ y_{jb} = \alpha^{r_{jb}} \]

\[ y_k = \alpha^k \]

\( j = 1, \ldots, n, \ i = 1, \ldots, m \)

Finally, the vendor sends \( E(b_{ji}), y_j, y_{jb}, y_k \) to the buyer.

- The buyer decrypts the messages

The buyer can get the message \( g^x \) when the item price he paid is equal to the item price the vendor gets, i.e. \( p_i = p_j \). Then we have

\[ D(b_{ji}) = \]

\[ \frac{(g^{b_{ji}})^{y_{jb}}}{(y_j)^{\alpha r_{jb} b^{kr_{jb}}}} = \frac{\alpha^{r_{jb} + r_j - r_i} g^{x_j}}{\alpha^{r_b + r_j - r_i} y_k} = g^{x_j} \]

\( j = 1, \ldots, n, \ i = 1, \ldots, m \)

6.6.2 One out of \( n \) Oblivious Transfer

In this section, we give the one out of \( n \) priced oblivious transfer protocol. This protocol can be seen as a special case of the \( m \) out of \( n \) priced oblivious transfer protocol, because when \( m = 1 \), we can get the protocol shown in Figure 6.7.

6.7 Conclusion

In this paper, we give the protocol to maintain people’s privacy when they buy items such as photos, songs, etc. or subscribe to some channels such as movies, news and
\[ \beta = \alpha^a \mod q \]
\[ E(p) = g^\epsilon \beta^r \]
\[ y = \alpha^r \]
\[ r \text{ Random number} \]
\[ p \in p_j \]
\[ j = 1, \cdots, n \]

\[ E(p_j) = g^{p_j} \beta^{r_j} \]
\[ y_j = \alpha^{k r_j} \]
\[ j = 1, \cdots, n \]
\[ E(b_j) = g^{k(p_j - p) + r_j} \beta^{kr_jb} \]
\[ = (g^{p_j}/g^p)^k g^{r_j} \beta^{kr_jb} \]
\[ = (E(p_j)/E(p))^k (\beta^r/\beta^{r_j})^k g^{r_j} \beta^{kr_jb} \]
\[ = (E(p_j)/E(p))^k (\beta^{r-r_j+r_jb})^k g^{r_j} \]
\[ = (E(p_j)/E(p))^k (\beta^r)^k g^{r_j} \]
\[ r_b = r - r_j + r_{jb} \text{ Random number} \]
\[ y_{jb} = \alpha^{r_b k} \]
\[ j = 1, \cdots, n \]
\[ y_k = \alpha^k \]

\[ D(b_j) = (\beta^{kr_jb}) g^{r_j}/(y_j)^a y_{jb}^a \beta^{r_a} \]
\[ = \alpha^{ak(r_a + r_j - r)} g^{r_j}/\alpha^{(r_b + r_j - r)k_a} \]
\[ = g^{r_j} \]
\[ j = 1, \cdots, n \]
\[ (p = p_j) \]

Figure 6.7: one out of n Priced OT protocol
so on. There is still some research to do to make it more flexible, or to develop new technology to protect personal privacy.
Chapter 7

Conclusion

The main goal of this dissertation was to develop the $m$ out of $n$ oblivious transfer protocol and its application in secure electronic commerce. We have provided several $m$ out of $n$ OT protocols and application schemes for E-commerce in this dissertation.

We gave the new notation of the $m$ out of $n$ OT protocol. The general scheme of $m$ out of $n$ interactive OT was implemented using the symmetry and asymmetry key system. The relationship of this scheme with the previously developed OT schemes were compared. All the previous schemes were seen as a special case of our new $m$ out of $n$ OT scheme. The drawback of this scheme was analyzed, and a more efficient $m$ out of $n$ OT scheme was developed. An application of this interactive OT scheme was given in the blind signature scheme.

The new non-interactive $m$ out of $n$ OT protocol was developed. This scheme includes a new mixed public key generation algorithm, which was developed by us based on the discrete log problem. The new non interactive $m$ out of $n$ OT protocol is based on the Diffie-Hellman problem. The limitation of this scheme was discussed. The security of the new scheme was analyzed. Two improved schemes were given,
which can be used for protection from fraud.

The application of the OT in buying digital goods was analyzed. We gave an improved scheme which use the $m$ out of $n$ priced OT protocol. We discussed the tools in this scheme, which includes homomorphic cryptography, the secret share and the zero knowledge proof, etc. We gave a concrete homomorphic cryptographic protocol, and the $m$ out of $n$ priced OT protocol was listed in detail.

The Oblivious Transfer is referred to as a subset of PIR. The reader can also read related materials.
List of Figures

2.1 The Model of interactive protocol ........................................... 11

3.1 The blind RSA signature generation protocol ............................ 29
3.2 The blind Schnorr signature generation protocol ....................... 31
3.3 The Rabin's Oblivious Transfer protocol .................................. 38
3.4 The Non-interactive Oblivious Transfer protocol ...................... 40

4.1 $m$ out of $n$ Oblivious Transfer ........................................ 45
4.2 $m$ Chosen out of $n$ Oblivious Transfer ................................ 48
4.3 The efficient $m$ out of $n$ protocol .................................... 50
4.4 The blind signature protocol .............................................. 54

5.1 The Non-interactive OT protocol ......................................... 68
5.2 The Modified Non-Interactive OT protocol .............................. 72

6.1 Maintain privacy in buying digital good OT protocol .................. 83
6.2 Maintain privacy in subscribing digital good OT protocol ............ 86
6.3 Classical Proof protocol .................................................. 90
6.4 BCDG Proof Protocol ...................................................... 92
6.5 CFT proof protocol ....................................................... 95
6.6 $m$ out of $n$ Priced OT protocol ....................................... 97
6.7 one out of $n$ Priced OT protocol .......................... 100
Bibliography


[35] C. Crepeau, J. van de Graaf, and A. Tapp, “Committed oblivious transfer and private multiparty computation,” in Advances in Cryptology - CRYPTO’95,


BIBLIOGRAPHY


Oblivious Transfer Protocols for Securing Electronic Commerce

Jun Qi Zhang

A thesis submitted for the degree of Master of Science at University of Western Sydney

February 2002
PLEASE NOTE

The greatest amount of care has been taken while scanning this thesis,

and the best possible result has been obtained.
Except where otherwise indicated, this thesis is solely my own original work. I certify that no part of this thesis has been submitted as a part of any other degree.

Jun Qi Zhang
October 29, 2002
To my Family.
Acknowledgements

This dissertation would not become the fruit without other people's help. First and foremost, thanks to my supervisors Prof. Vijay Varadharajan and Dr. Yi Mu. Prof. Vijay Varadharajan gave me the chance to undertake a research degree, and I know he is always behind me. Dr. Yi Mu guided me step by step, gave me time whenever I needed, and he helped me not only in research but also in life.

Thanks to other staff at CIT for providing me with a supportive and cooperative environment. A special thanks to Rajan Shankaran and Trish Saladine for their kind help in my study environment.

Thanks to Lin Smith, Warren Hosey and Dr Vijay Khandelwal for offering me part time work during my study.

Finally, I would like to devote this dissertation to my family. I am ever grateful for their support and sacrifice.
Abstract

Electronic commerce is based on the open computer network for which security is a major issue. Cryptography is the foundation of the security and Oblivious Transfer (OT) protocols are one primitive of modern cryptography. The main goal of this dissertation is to develop new and more efficient oblivious transfer protocols and its applications in electronic commerce.

In this dissertation, we propose a new $m$ out of $n$ OT scheme and its implementation and discuss its security and efficiency. We also propose a new more efficient $m$ out of $n$ OT protocol and compared it with the previous works. The analysis shows that the previous oblivious protocol can be regarded as a special case of the new proposed OT scheme. We also explore its applicability in blind signatures.

We propose new non-interactive $m$ out of $n$ OT scheme that includes a newly developed public keys generation algorithm based on the discrete log problem and an OT protocol based on the Diffie-Hellman problem. The security of the scheme has been discussed.

We give a new buying digital goods scheme using the new $m$ out of $n$ priced OT which is based on the priced oblivious transfer protocol developed by Bill Aiello, Yuval Isahai, and Omer Reingold. Security of this scheme is analyzed and tools used in this scheme are discussed. A concrete homomorphic protocol is given.
Contents

Acknowledgements vii

Abstract ix

1 Introduction 1
  1.1 Electronic Commerce ........................................... 1
  1.2 The Cornerstone of Security .................................... 2
  1.3 The Scope of the Research ...................................... 3
  1.4 Outline of Dissertation ........................................ 5

2 Basic Notations and Background 7
  2.1 Algorithms and Complexity Theory ............................ 7
    2.1.1 Algorithms .................................................. 7
    2.1.2 Complexity of the Algorithm ............................... 8
    2.1.3 Complexity of the Problem ................................ 9
  2.2 Interactive and Non-Interactive Protocols ................... 10
  2.3 Basic Abstract Algebra ........................................ 11
    2.3.1 Group ...................................................... 12
    2.3.2 Integer Group .............................................. 12
  2.4 Number Theory .................................................. 13
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>2.5.1</td>
<td>Knapsack Problem</td>
<td>15</td>
</tr>
<tr>
<td>2.5</td>
<td>2.5.2</td>
<td>The Discrete Logarithm Problem</td>
<td>15</td>
</tr>
<tr>
<td>2.5</td>
<td>2.5.3</td>
<td>The Factorisation Problem</td>
<td>17</td>
</tr>
<tr>
<td>2.6</td>
<td></td>
<td>Miscellaneous Notations</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>3.1</td>
<td>Hash Functions and the Random Oracle Model</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>3.2</td>
<td>Public Key Encryption</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>3.3</td>
<td>Digital Signature Schemes</td>
<td>24</td>
</tr>
<tr>
<td>3</td>
<td>3.4</td>
<td>Blind Digital Signature Schemes</td>
<td>28</td>
</tr>
<tr>
<td>3</td>
<td>3.5</td>
<td>Identification Protocols and Zero Knowledge Proof</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>3.5.1</td>
<td>Identification Protocols</td>
<td>31</td>
</tr>
<tr>
<td>3</td>
<td>3.5.2</td>
<td>Zero Knowledge Proofs of Knowledge</td>
<td>33</td>
</tr>
<tr>
<td>3</td>
<td>3.6</td>
<td>Secret Sharing and Condition Disclosure of Secrets</td>
<td>36</td>
</tr>
<tr>
<td>3</td>
<td>3.7</td>
<td>Oblivious Transfer Protocol</td>
<td>37</td>
</tr>
<tr>
<td>3</td>
<td>3.7.1</td>
<td>Rabin's Oblivious Transfer Protocol</td>
<td>38</td>
</tr>
<tr>
<td>3</td>
<td>3.7.2</td>
<td>One out of Two Oblivious Transfer Protocol</td>
<td>39</td>
</tr>
<tr>
<td>3</td>
<td>3.7.3</td>
<td>Non-Interactive Oblivious Transfer Protocol</td>
<td>39</td>
</tr>
<tr>
<td>4</td>
<td>4.1</td>
<td>Introduction</td>
<td>41</td>
</tr>
<tr>
<td>4</td>
<td>4.2</td>
<td>Concept of $m$ out of $n$ Oblivious Transfer</td>
<td>42</td>
</tr>
<tr>
<td>4</td>
<td>4.3</td>
<td>Implementation of $m$ out of $n$ Oblivious Transfer</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>4.3.1</td>
<td>$m$ out of $n$ Oblivious Transfer</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>4.3.2</td>
<td>Security Analysis</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>4.3.3</td>
<td>$m$ Chosen out of $n$ Oblivious Transfer</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>4.3.4</td>
<td>Oblivious Transfer</td>
<td>48</td>
</tr>
</tbody>
</table>
CONTENTS

4.4 Efficient m out of n OT Scheme ........................................ 49
  4.4.1 System Set up ......................................................... 49
  4.4.2 Security Analysis ..................................................... 52
4.5 Blind Signature .......................................................... 53
4.6 Concluding Remarks ...................................................... 53

5 Non-interactive Oblivious Transfer ........................................ 57
  5.1 Introduction ............................................................... 57
  5.2 Non-Interactive Oblivious Transfer .................................... 58
  5.3 System Set up ............................................................. 60
    5.3.1 Public Key Generation Algorithm .............................. 60
    5.3.2 Public Key Verification ........................................... 63
    5.3.3 Security Considerations ........................................... 63
  5.4 A Simple Example ....................................................... 65
    5.4.1 Key Generator Algorithm - A Simple Example ............... 65
    5.4.2 Public Keys Verify Algorithm .................................... 66
  5.5 Non-Interactive Oblivious Transfer Protocol ......................... 67
  5.6 Security Analysis ....................................................... 69
  5.7 Check for Fraud .......................................................... 71
    5.7.1 Verifiable Encrypted Message Using the Double Encryption . 71
    5.7.2 Verifiable Encryption by Reconstructing the Private Keys . 73
  5.8 Concluding Remarks ..................................................... 74

6 Maintaining the Privacy in Buying Digital Goods ......................... 77
  6.1 Introduction .............................................................. 77
  6.2 Preliminaries ............................................................. 78
    6.2.1 Problem Specification .............................................. 78
    6.2.2 Security Requirement .............................................. 80
6.2.3 Tools .................................................. 80
6.3 System Set up .......................................... 81
  6.3.1 The Solution Protocol ............................ 82
  6.3.2 Some Consideration ............................... 84
6.4 Subscription ............................................ 85
  6.4.1 Set up the Protocol ............................... 85
  6.4.2 Security Analysis .................................. 88
6.5 Zero-Knowledge Proofs and Condition Disclosures ........................................ 88
  6.5.1 Zero-Knowledge Proofs ............................ 89
  6.5.2 Condition Disclosures of Secrets ............... 94
6.6 Oblivious Transfer ..................................... 96
  6.6.1 $m$ out of $n$ Oblivious Transfer ............... 96
  6.6.2 One out of $n$ Oblivious Transfer ............... 99
6.7 Conclusion ............................................. 99

7 Conclusion .............................................. 103