1-out-of-2 Oblivious Transfer based on Diffie-Hellman Assumption
基于迪菲-赫尔曼假设的二选一不经意传输

A Final Report

submitted to School of Mathematics & Physics

of Xi'an Jiaotong-Liverpool University

in partial fulfilment for the award of the degree of

BSc Financial Mathematics

By

Hanlong Li 1929768

Supervisor: Dr. Neng Zeng

November 23, 2023

Abstract

As one of the fundamental scheme in cryptography. This report will give a comprehensive inroduction about oblivious transfer and its application, security definition and variants. Focusing on 1-out-of-2 oblivious transfer with static semi-honest/ malicious adversary under DDH assumption, this report present six classic feasible structure of this type. Realated background and definition are well presented in chapter 2&3 along with the protocol structures.Rigorous security proof is also given in chapter4.

Key words:Cryptography, SMC, 1-out-of-2 OT, DDH

摘要: 作为密码学中的基本主题之一，本报告将全面介绍不经意传输及其应用、安全定义和变体。重点介绍了在 DDH 假设下，对于 1-out-of-2 不经意传输的静态半诚实/恶意对手，本报告提出了六种经典的可行结构。相关背景和定义在第 2 和第 3 章中得到了充分的阐述，同时介绍了协议结构。第 4 章中给出了严格的安全证明.

关键词: 密码学，多方安全计算，二选一不经意传输，迪菲-赫尔曼假设

Chapter 1

Introduction

As a primitive protocol, oblivious transfer (OT) is the foundation of many cryptography topics, including private information retrieval, private set intersection and so on [1]. This oblivious transfer concept was firstly introduced by Rabin in 1981 as a method to solve exchange of secrete (EOS) problem [5]. Generally speaking, it requires the sender to send a group of encrypted messages without knowing which one (or more) is predetermined by the receiver, while the receiver can decrypt nothing but the predetermined message(s) among what the sender offers.
To be specific, for an oblivious transfer protocol that is called k-out-of-n OT, there is a sender A who can offer n pieces of messages, and a receiver B who wants k out of the n messages, which are determined in advance. Meanwhile, A wants to make sure that B can only get the intended k messages and nothing extra from what A have sent, while B wants to make sure A knows nothing about the specific k choices that B have made from the n potential messages.
Over the last 4 decades, its great potential and intriguing nature has raised a lot of scholars to further develop the structure and apply in many scenarios. Especially in the field of secure multi-party computation (SMC), oblivious transfer turned out to be one of the bases in forming important protocols like gamble circuit, zero-knowledge proof [1,10]. Therefore, it is of great importance to research this primitive protocol and illustrate its development and related security analysis.
In terms of application of OT, so far there are three crucial applications of OT, private information retrieval (PIR), location based services (LBS), private set intersection (PSI). PIR is the protocol to help a database user securely extract his or her user information from the data base holder (administrator), while the holder is ignorant about exact content required by the user. In PIR protocol, the database user takes the role of the receiver in OT, requiring 1 or more messages. Database holder is the sender in OT, sending a group of messages without knowing which is (are) required by receiver. In such way, the user information is kept private from the holder and the user can not steal data of other users. Real life examples can be online-health database or any database with sensitive information[18,19]. LBS protocol is to help the LBS user to get his or her location from the LBS service provider. In this protocol, the user is the receiver of OT, trying to get the exact location a_r of herself which is one message. The provider divides the general location A of that user into n parts: $\{a_1, a_2, ...a_r, ..., a_n\}$ and sends this set to the user. In such way, the user can get his exact location but the service provider knows noth-

ing about the exact location. This protocol may be seen on any map application like Google map, Baidu map[16,17]. PSI is to find the intersection of two groups of messages; This protocol involve two parties sender and receiver, and both of them will execute PSI with the help of OT. The intersection set will be the output sent to the receiver. Real life examples include password monitoring that checks whether the saved password of users is divulged to other third parties, illegal material identification that checks whether some online pictures are illegal, which means they have intersection with "illegal picture set"[14,15].

In terms of security settings of OT, the security of oblivious transfer aligns with security definition of SMC, since the all applications mentioned above are sub-topics of SMC. In secure multi-party computation, two major conditions of security, correctness and privacy, were firstly introduced [2]. Later, this was expanded to 5 security definitions: correctness, privacy, independent of inputs, guaranteed delivery and fairness [3]. However, under some situations we need to loose those requirements. As a result, researchers formalized the security definition of any SMC protocol by comparing an "ideal case" with a trusted authority [4]. More details are included in chapter 2&3.

Literature review: This report basically review 3 parts, including cryptography basics, secure multi-party computation and 1-out-of-2 oblivious transfer. Since the reporter is in financial mathematics, few cryptography basic knowledge are clear to him. Thus the topics like module computation, the concept of group, hard problem, polynomial time machine are well reviewed. As the core of this report is oblivious transfer, learning the background of OT is necessary– secure multi-party computation is essential in constructing the OT protocols and defining the security of OT. Therefore, the content and history of SMC is well reviewed[1,2,3,4,6,13]. The first two properties: correctness and privacy is put forward under the context of SMC[2]. Then the five standards of SMC is detailed defined by Donald Beaver and Shaft Goldwasser in 1989[3]. However this 5 standards are far to complexy and inconvenient in defining the security of various protocols. Therefore, Donald Beaver further put forward the important concept of real/ideal paradigm.[4]. See more in chapter 2
In terms of 1-out-of-2 oblivious transfer, in 1981, Rabin firstly put forward the concept of oblivious transfer as a sub-protocol in solving EOS problem[5]. The security of that protocol depends on the factorization of a composite number, which is not suitable for application. Afterwards, Shimon Even and his fellows construct the first feasible 1-out-of-2 OT protocol based on public key technique and applied it to contract signing under electronic business[9]. However, the above OT requires a lot of interaction between the sender and chooser. Therefore, Mihir Bellare and Silvio Micali invented the non-interactive OT with 3 steps, based on the Computational version of **Diffie-Hellman assumption**(see chapter 2.1.1) [10]. Later, Moni Naor and Benney Pinkas amended the 1-out-of-2 OT with random oracle model and further improved the efficiency [11]. At the same time, they also constructed an oblivious transfer which is solely based on Decisional Diffie-Hellman assumption. However as it requires high computation cost, Gilad Ashrov et al.(2013) gave an efficient version of that based on Key Derivation Function[12]. In 2010, the first 1-out-of-2 OT with the presence of malicious adversary was invented, with the introduction

of zero-knowledge proof[13]. In their work, they specify the three levels of security definitions under malicious adversary and put forward three protocols respectively.

Methodology: my methodology of FYP is to understand this unfamiliar field, cryptography, by every possible mean, including reading academic articles of oblivious transfer, covering the knowledge gaps in abstract algebra and cryptography, having discussions with my supervisor and so on. Using digital library of XJTLU, I firstly searched the development of oblivious transfer. After getting a basic knowledge of this area. I started to search in details of how those variants lift their performance by changing the specific scheme and introducing new underlying hard prob-lems. Meanwhile, realizing that oblivious transfer is a protocol in the area of secure multi-party computation(SMC), I start to reach more knowledge in SMC. Most importantly, understanding the modeling of the adversaries and security definitions under the context of SMC and how they related to OT. As my project went on, I realized that my knowledge gap of basic cryptography is obvious. Therefore, I choose to use coursera as an online platform for filling the gaps in some pre-requisite courses like abstract algebra and introduction to cryptography. Moreover, having a active and regular discussion with my supervisors is of great importance. My supervisor helped me a lot in the direction of research, and improved my methodology and gave me timely academic guidance in cryptography.

Coverage: In this project, some related backgrounds will be given first in chapter 2. The first part of chapter 2 is a brief introduction of secure multi-party computation(SMC), focusing on its security definitions and special properties, as those discussion will be our cornerstone in defining the security of 1-out-of-2 oblivious transfer. Meanwhile, as we can see in later chapter, the construction of an 1-out-of-2 OT must rely on some hard problem(or assumptions) to achieve computational security. Thus, the main assumption of this project, Diffie-Hellman assumption, along with necessary techniques that are used in different protocols, will be introduced as the second part of the chapter2. In chapter 3, the formal definition of 1-out-of-2 OT which is based on SMC security setting will be given. Afterwards, certain specific constructions of OT with different security standards and computation efficiency will be explained in detail. Those protocols can be roughly divided into 3 groups, and within each group the protocols share the similar structures. Therefore, chapter 4 will offer some necessary proofs about the security.

Chapter 2

Background

In this chapter, intended secure properties and security definition of SMC will be presented first. The so called ideal/real simulation paradigm will be introduced as well. After that adversary modeling under SMC will be categorized depend on 3 factors: corruption approach, allowed adversary behavior, and computation capacity. Then, we will give a brief description about the Diffie-Hellman assumption and several cryptography techniques that are used in chapter 3 in the construction of protocols. It should be noted that although the secure multi-party computation part is not directly related to the main content in chapter 3, it is the basis in defining the security of oblivious transfer and gives us a bigger picture to evaluate different oblivious transfer protocols.

2.1 Secure Multi-party Computation

Distributed computation is to achieve some computation by multiple distant parties. Secure Multi-party computation is to securely conduct such distributed computation. In order to achieve security, we need to introduce multiple protocols related with security, and OT is one of the most commonly used basic protocol in SMC. Therefore, as a primitive tool in secure multiparty computation, Oblivious transfer serves for the general security requirement of SMC. In other words, The specific security definition of different variants of OT will be derived based on the following security settings and adversary modeling:

Security Setting of SMC

In SMC, we mainly focusing the potential adversaries and their harm induced. Therefore, security definition of SMC can be divided into five security standards[3]:

Definition 2.1 (five standards of SMC) *A secure multi-party computation should have the following:*
Privacy*: Every party should learn only the output of the protocol, nothing else.*
Correctness*: After the execution of the protocol, every party should get the correct output.*
Independent of Inputs*: Corrupted parties that follow the adversaries' instruction can choose*

their input independently with respect to the honest parties' inputs.
Guaranteed output delivery*: Under any circumstances, the corrupted party cannot prevent honest parties to receive their output.*
Fairness*: if the corrupted entities receive their output, then honest entities also receive their output.*

Any protocol that satisfied the five standard can make sure all the honest parties get their intended output with privacy, while the adversaries can not interrupt or intercept the output from them. However, given the complexity of environment in SMC, we would like to loose some of the standards in various real applications. Moreover the five standards is not simple. Thus, researches propose a formal security definition with comparison with an "ideal case"[4]: Since the SMC is actually a kind of decentralized computation, we can always find a counterpart that is a centralized computation. That is to say, we can always simulate any protocol of SMC with the replacement of a trusted authority to better locate the source of insecure. In other words, there will always exist a ideal case where a trusted authority will take the input from the original parties in the multiparty computation protocol, compute and deliver the output correctly.

Definition 2.2 *Suppose there exists an adversary that can successfully break the real-case protocol. In that situation, it can also break the ideal-case protocol such that the input/output distributions of the participating entities and adversary in the real and ideal-case execution are completely indistinguishable.*

Following the ideal/real simulation paradigm, all five properties can be obtained. In an ideal execution, the adversary learns nothing beyond the outputs of the corrupted party, and the same holds for a real execution. The correctness is ensured because the honest parties' outputs are identical in both the real and ideal executions. Additionally, in an ideal execution, the trusted party computes the correct outputs for all honest parties. As for the independence of inputs, note that in an ideal execution, all inputs are sent to the trusted party before any output is received, and the corrupted parties have no knowledge of the honest parties' inputs at the time of sending their inputs. Therefore, the inputs of the corrupted parties are chosen independently of the inputs of the honest parties, as required. Finally, the ideal world ensures guaranteed output delivery and fairness since the trusted party always returns all outputs. The fact that the same holds in the real world is again guaranteed by the identical outputs of the honest parties in both the real and ideal executions.

Adversary Modeling under SMC

This leads us to model the adversaries in 3 aspects:
Corruption Approach:
Depending on how and when parties get corrupted, the approach of corruption by adversaries can be divided into 3 types:
Static Corruption Approach: In static corruption, the adversary have corrupted some parties in advance. During execution of the protocol, the honest parties and the corrupted parties will remain their status.

Adaptive Corruption Approach: In adaptive corruption, the adversary can corrupt parties as the protocol be executed. The adversary have the flexibility to choose any parties as the execution of protocol goes on. However, if one party gets corrupted by adversary, it will continue to be corrupted during the process afterwards.
Proactive Corruption Approach: In this approach, the corrupted parties can only remain corrupted in a limited time. Thus, honest or corrupted parties both may turn to the other side during the execution.
Allowed Adversarial Behavior
When constructing our protocol, 2 scenarios could happen concerning different allowed adversarial behavior:
Semi-honest Adversaries: Also known as passive and honest-but-curious adversary, the semi-honest adversary will properly follow the protocol scheme. As a result, the adversary can only learn the secret information from the protocol by analysing the output and input from the corrupted party. Obviously, this is a relatively weak adversary.
Malicious Adversaries: malicious adversaries, or so called active adversaries do not follow the rule of the attacked protocol. In fact, they can let the controlled parties to change the protocol scheme and learn secrete by such behavior. This kind of adversary is relatively a strong type. Ideally, we prefer the protocol to be secure against such adversary. However, this goal often requires OT to spend high costs in communication and computation.
Computationally Dependent or Not:
the computational capacity can also divides adversaries into 2 types:
Polynomial time adversaries: This type of adversary can break the protocol' s security in probabilistic polynomial time (finite time,PPT).
Computationally unbounded adversaries: This type of adversary have unlimited time to break the protocol.
Computational or Information-theoretical Model

2.2 Hard Problems and Encryption Techniques

The above security modeling leads us to two different models in secure computation: one is *computational models* that are based on mathematical hard problems like computational Diffie-Hellman problems. In this case we consider the adversaries tries to break the protocol in polynomial time; The other is *information-theoretic models* which dose not depend on the any hard problems and consider the adversary in independent of any time complexity. It is worth noting that the information-theoretic models can be achieved as long as the number of honest parties exceed the that of dishonest parties. However, oblivious transfer only involves two parties: Sender A and receiver B. Therefore, we need to consider certain underlying hard problems to build any oblivious transfer protocol in computational models. The DDH hard probelem is our focus in this project. In addition, some encryption techniques will be used in the construction of certain protocols, we briefly present those techniques in this section as well.

Diffie-Hellman Assumption and Discrete Log Problem

It is firstly shown in the article of Whitfield Diffie and Martin Hellman in 1976[20]. To create a so called one-way function or trap-door function in cryptography, they need a kind of mathematical operation that is relatively easy to conduct but hard to reverse. Therefore this assumption is put forward. The underlying mathematics is the fact that it is usually easy to do exponentiation but hard to do logarithm, which is also known as discrete log problem. The assumption was implicit in early works since 1976. Until 1993, the Brands, S. formal defined it under the context of undeniable signatures[21]. To be specific, this assumption contains two variants the *Computational Diffie-Hellman Assumption* (CDH) and the *Decisional Diffie-Hellman Assumption* (DDH). The later is actually a wilder assumption than the former, enabling a wider real application like ElGamal encryption, efficient pseudo random function and so on [22].
Assume there is a multiplicative group G_p^* with order p and operation: $*$. Then for any $g \in G$ and positive integer a we have the denotation $g^a = \underbrace{g * g * ... * g}_{a}$. The *discrete log problem* (DLP) is to find the value of a given information $\left\{G_p^*, g, g^a\right\}$. For some simple cases this a is easy to find, but with careful choose of g and G_p^*, the probability of finding a is negligible.
The *computation Diffie-Hellman problem* (CDH) is to find the value of g^{ab} given information $\left\{G_p^*, g, g^a, g^b\right\}$. As we can see if we can successfully solve the DLP, then CDH can be easily solved as well, by getting the value of a and b and compute $g^{ab} = ((g)^a)^b$. In other word, **the hardness of DLP is greater or equal to CDH.** Thus, if we wisely choose g and G, the p, the probability of solving CDH can be negligible as well. Therefore, the computational Diffie-Hellman assumption can hold.
The *decisional Diffie-Hellman problem*(DDH) is to judge whether a number $C \in G_p^*$ satisfying $C = g^{ab}$, given information $\left\{G_p^*, g, g^a, g^b\right\}$. If we can compute g^{ab} then we can differentiate C and g^{ab}. Therefore, **the hardness of CDH is greater or equal to DDH.** If computational Diffie-Hellman assumption holds, decisional Diffie-Hellman assumption holds, too.
In constructing the specific scheme of OT, some other protocols might be used:

Public Key Encryption

Public key encryption is an encryption method that use a public-secrete key pair, firstly put forward by hellman in 1976[20]. The receiver Bob of this protocol will use some methods to generate a key pair. The public key as a way of encryption will made public for everyone who wants to send messages to A, while the secrete key is kept by Bob himself as way to decrypt every message he received. In such way, every can get the encrypted messages through public communicate channel, but only Bob can correctly decrypt the message.
Formally speaking, the PKE scheme consists of three algorithms–key generation G , encryption E, and decryption D. After input the security parameter λ , receiver B will have the public-secret key pair (pk, sk) by using G. And the public key pk will be sent to A. Here I give one simple example of an RSA public key with receiver B and sender A[8]:

RSA public key encryption

Input of B: primes p, q
Input of A: message m
Output of B: message m
Output of A: nothing
(1)B: by choosing two large random primes p, q, he can get the number $n = p * q$, and d satisfying $gcd\,(d, (p-1)*(q-1)) = 1$, and e satisfying $e * d = 1\ mod\ ((p-1)*(q-1))$. Then, he sends the pair (e, n) as a public key to A.
(2)A: After receiving public key (e, n), A encrypt message m: $E_e(m) = m^e$ and send it to B. (3)B: using secrete key, B can get the message: $D_d(E_e(m)) = (m^e)^d = m$

Random Oracle Model

The concept of the random oracle model was first proposed by Bellare and Rogaway [23]. The basic idea is to provide all parties involved in a cryptographic protocol with access to a function h, and then prove the correctness of the protocol based on the assumption that h maps each input to a truly random output, similar to a truly random oracle. In practice, h is typically set to a specific function derived from a standard cryptographic hash function. In cryptography, using a specific function may not provide truly random outputs as it is deterministic and will always return the same output when given the same input. The random oracle model is a way to provide efficiency and security guarantees by assuming that a function behaves like a truly random oracle, even though it is not truly random. Although the security guarantees of the random oracle model are not as strong as those provided by the provable security approach, they are still considered better than those provided by an ad hoc protocol design.

Definition 2.3 (Hash Function) *A hash function h is a well-defined deterministic algorithm that takes as input data of arbitrary length and produces a short fixed-length digital representation of the data, or a digest, as its output.*

There are various real applicable hash functions, like MD5[24] and SHA-1 [25]. Here is a formal definition:

Definition 2.4 *1.One-way (or preimage resistance): given a hash value y, it is infeasible to find an input x that hashes to y, i.e., $h(x) = y$.*
2.Weak collision resistance (or second preimage resistance): given an input x and the corresponding digest of that input $h(x)$, it is infeasible to find another input x'; that matches the digest, i.e., $h(x) = h(x')$.
3.Strong collision resistance (or simply collision resistance): it is infeasible to find two different inputs x and x'; that hash to the same value, i.e., $h(x) = h(x')$.

Key Derivation Function

A Key Derivation Function (KDF) is an important part of cryptographic systems. Its main purpose is to take a source of initial keying material, which may have some randomness but

not uniformly distributed, and derive one or more cryptographically strong secret keys from it. Cryptographically strong keys are considered to be pseudorandom keys, which means that they are computationally indistinguishable from a truly random string of the same length. This is important to ensure the security of the cryptographic system.[26]
The general definition of KDF is not well defined, since the role that KDF plays may vary a lot. Here we only give the definition of KDF that will be used in chapter3 in the protocol 5[12].
Let $\mathbf{F}(1^k)$ be a function that produces a group (Z, q, g) for which the DDH problem is believed to be hard. We define:

Definition 2.5 (Key-Derivation Function) *A key derivation function KDF with l-bit output is said to be secure with respect to DDH if for any PPT attacker A there exists a negligible function $\mu(\cdot)$ such that:*

$$|Pr[A(Z, q, g, g^r, h, KDF(h^r)) = 1] - Pr[A(Z, q, g, g^r, h, z) = 1]| \leq \mu(k)$$

where $(G, q, g) = \boldsymbol{F}(1^k)$, r is distributed uniformly in Z_q and z is distributed uniformly in $\{0,1\}^l$.

Similar to ROM, real KDF may also based on hash functions like SHA-2[26].

Zero-Knowledge Proof

The concept of zero-knowledge is a fundamental property associated with interactive proofs, interactive arguments, and non-interactive proofs. In cryptography, the soundness property ensures that a verifier can trust the information provided by a prover. However, the zero-knowledge property is essential for protecting the privacy of the prover. With a zero-knowledge proof, a prover can demonstrate the validity of a given statement to a verifier without revealing any additional information beyond the statement's validity.
In other words, a zero-knowledge protocol ensures that the verifier learns only the validity of the statement from an honest prover, and anything the verifier "sees" can be simulated efficiently. This condition must be met even if the verifier deviates from the protocol in arbitrary ways.[27] provides more details.

Chapter 3

In chapter 3, we first present the formal secure definition of 1-out-of-2 oblivious transfer under different types of adversaries. Based on the ideal/real simulation paradigm, the security of malicious adversary can be categorized into 3 level, privacy only, one-sided simulation and full simulation. Secondly, we will detail some of the most important 1-out-of-2 OT, most of which are based on DDH assumption. The specific structures will be presented and their properties will be discussed and compared, including its security level, computation cost and pros and cons.

3.1 Security Definition under SMC

In general, the security of OT is a relaxed version of SMC security. This is due to the fact that oblivious transfer is a **two party computation** (see Definition A.2) protocol, a protocol in SMC that only involves two party, a sender and a receiver. In other words, the five standards of SMC should be relaxed or modified to some extent. To be specific,n situations where one party may be corrupt, it is not possible for two parties to toss an unbiased coin [13]. To account for this, the definition of fairness is relaxed, allowing the adversary to terminate the computation prematurely and prevent guaranteed output delivery. Meanwhile, the adversary can cause the abortion in advance after obtaining their output but before all honest parties receive theirs. Thus "fairness" is not possible. To put it simple, this modified definition is the definition where the ideal execution is altered to allow the adversary to instruct the corrupted party not to send outputs to some of the honest parties. This modification does not affect the other properties of the original definition.

To make the definition formal, we need to consider the various conditions of the adversary, and define them separately. Owing to space and complexity limitation, we assume the corruption approach to be static and computational dependent. Therefore, the allowed adversarial behavior only can be semi-honest adversary or malicious adversary. In addition, as discussed in chapter 2, the information-theoretical model can only be achieved when there exist honest majority, which is not relevant to a two party protocol. Therefore, we only here consider computational model. To sum up, we consider two cases: 1)*Security Definition under static Semi-honest Adversary with probabilistic polynomial time* and 2)*Security Definition under static malicious Adversary with probabilistic polynomial time.*

Security Definition under Semi-honest Adversary

First of all, we need to discuss the exact meaning of the semi-honest adversary under the context of 1-out-of-2 oblivious transfer. Since the adversary is semi-honest, they can not lie about their input and can not revise the protocol itself. Thus, the only method left where they can "get more" is to get extra information from what they are supposed have. In another word, a secure OT is a protocol under semi-honest adversary is a protocol which is able to prevent the such method to work out. We introduce the concept of "view" to describe the set of messages that are supposed have by each party. Given its input and output, a party can only get its "view":

Definition 3.1 *The view of the ith party ($i \in \{1,2\}$) during an execution of π on (x,y) and security parameter n is denoted by*

$$VIEW_i^{\pi}(x,y,n) = (w, r^i, m_1^i, m_2^i, m_2^i, ...)$$

, where $w \in \{x,y\}$ (depending the value of i), r^i equals the contents of the ith party' s internal random tape, and m_j^i represents the jth message that it received.

For example, the view of the receiver in 1-out-of-2 OT is (σ, r, m_σ), where the σ is the choice the receiver made from0,1, and r^i to be the random number(s) it received during the process of executing OT, and m_σ is the wanted message.
Therefore, the definition of security under semi-honest adversary can be briefly described as the view of each party can be simulated by some simulator, and the function output is correct:

Definition 3.2 *Let $f = (f1, f2)$ be a functionality. We say that securely computes f in the presence of static semi-honest adversaries if*
$\{output^{\pi}(x,y,n)\}_{x,y\in\{0,1\}^*;\ n\in N} \overset{c}{\equiv} \{f(x,y)\}_{x,y\in\{0,1\}^*}$
and there exist probabilistic polynomial time algorithms S_1, S_2 such that:
$\{(S_1(1^n, f_1(x,y))\}_{x,y\in\{0,1\}^*;\ n\in N} \overset{c}{\equiv} \{view_1^{\pi}(x,y,n)_{x,y\in\{0,1\}^*;\ n\in N}$,
$\{(S_2(1^n, f_2(x,y))\}_{x,y\in\{0,1\}^*;\ n\in N} \overset{c}{\equiv} \{view_2^{\pi}(x,y,n)_{x,y\in\{0,1\}^*;\ n\in N}$

Security Definition under Malicious Adversary

In the presence of a malicious adversary, it is not enough to construct simulators that only generate the view of the corrupted party. This is because a malicious adversary can use any efficient attack strategy and may arbitrarily deviate from the protocol specification. Therefore, a simulator for the semi-honest case, where parties are assumed to follow the protocol but may try to learn more than they should, cannot correctly generate the actual view of the two parties in the presence of a malicious adversary. Moreover, in addition to ensuring that a corrupted party cannot learn more than it should, we also require correctness (i.e., a corrupted party cannot cause the output to be incorrectly distributed) and independence of inputs (i.e., a corrupted party cannot make its input depend on the other party's input).
In order to capture these properties, the security should be defined under the ideal/real world model. See 4.1 for more notation regarding the execution in real world and execution in ideal world where a trusted authentic third party is involved.

In other words, a secure protocol under this security level means that an attacker can not do more damage in real paradigm than if there was a trusted third party involved in the computation to form an ideal paradigm.

Definition 3.3 (Full Simulatability) *Let f and π be as above. Protocol π is said to securely compute f with abort in the presence of malicious adversaries if for every non-uniform probabilistic polynomial-time adversary A for the real model, there exists a non-uniform probabilistic polynomial-time adversary S for the ideal model, such that for every $i \in \{1,2\}$,*
$\{IDEAL_{f,S(Z),i}(x,y,n)\}_{x,y,z,n} \stackrel{c}{\equiv} \{REAL_{\pi,A(Z),i}(x,y,n)\}_{x,y,z,n}$,
where $x,y \in \{0,1\}^*$ under the constraint that $|x| = |y|$, $z \in \{0,1\}^*$ and $n \in N$.

The above definition following ideal/real paradigm holds a relatively strong security, guaranteeing privacy, correctness, independence of inputs. However, in real application we may need some relaxation of this definition to achieve a lower cost with sufficient security. The most simple case is that sometimes we only wish to hold privacy:

Definition 3.4 (Privacy Only) *A two-message two-party probabilistic polynomial-time protocol (S, R) is said to be a* **private oblivious transfer** *if the following holds:*
[Non-triviality] *If S and R follow the protocol then after an execution in which S has for input any pair of strings $x0, x1 \in \{0,1\}^*$, and R has for input any bit $\sigma \in \{0,1\}$, the output of R is x_σ.*
[Privacy in the case of a malicious S^*] *For every non-uniform probabilistic polynomial-time S^* and every auxiliary input $z \in \{0,1\}^*$, it holds that*

$$\{VIEW_{S^*}(S^*(1^n,z),R(1^n,0))\}_{n\in N} \stackrel{c}{\equiv} \{VIEW_{S^*}(S^*(1^n,z),R(1^n,1))\}_{n\in N}.$$

[Privacy in the case of a malicious R^*] *For every non-uniform deterministic polynomial-time receiver R^*, every auxiliary input $z \in \{0,1\}^*$, and every triple of inputs $x_0, x_1, x \in \{0,1\}^*$ such that $|x_0| = |x_1| = |x|$ it holds that either:*

$$\{VIEW_{R^*}(S^*(1^n,(x_0,x_1));R^*(1^n,z))\}_{n\in N} \stackrel{c}{\equiv} \{VIEW_{R^*}(S^*(1^n,(x_0,x));R^*(1^n,z))\}_{n\in N}.$$

or

$$\{VIEW_{R^*}(S^*(1^n,(x_0,x_1));R^*(1^n,z))\}_{n\in N} \stackrel{c}{\equiv} \{VIEW_{R^*}(S^*(1^n,(x,x_1));R^*(1^n,z))\}_{n\in N}.$$

In addition to privacy only definition, a stronger relaxation is half simulation:

Definition 3.5 (One-Sided Simulatability) *Let f be a functionality where only P_2 receives output. We say that a protocol π* **securely computes** *f* **with one-sided simulation** *if the following holds:*
1. For every non-uniform PPT adversary A controlling P_2 in the real model, there exists a non-uniform PPT adversary S for the ideal model, such that

$$\{REAL_{\pi,A(Z),2}(x,y,n)\}_{x,y,z,n} \stackrel{c}{\equiv} \{IDEAL_{f,S(Z),2}(x,y,n)\}_{x,y,z,n}$$

where $x,y,z \in \{0,1\}^*$, $n \in N$ and $|x| = |y|$.
2. For every non-uniform PPT adversary A controlling P_1;

$$\{VIEW^A_{\pi,A(Z),1}(x,y,n)\}_{x,y,y',z,n} \stackrel{c}{\equiv} \{VIEW^A_{\pi,A(Z),1}(x,y',n)\}_{x,y,y',z,n}$$

where $x,y,y',z \in \{0,1\}^*$, $n \in N$ and $|x| = |y| = |y'|$.

3.2 1-out-of-2 OT Structures

Here a few important OT structures will be presented, which may vary in security level, computation efficiency, or underlying technique. As we mentioned earlier, the inputs of S will include two messages: $x_0, x_1 \in \{0,1\}^l$ with length l and the inputs of R will contain the choice index $\sigma \in \{0,1\}$, while the output for S in nothing and x_σ for R(see protocol 1). For simplicity, we use these notations from protocol 2 to protocol 5 without mentioning. In addition, we will have some auxiliary inputs related with assumptions and techniques to facilitate the execution of protocols.

The first practical scheme of 1-out-of-2 OT is based on the public technique[9]. S is the sender in this oblivious protocol. Meanwhile, he take the role of the receiver in public key encryption(PKE), who generate the public-secret key pair and use secret key for decryption. R is the receiver in this oblivious protocol. Meanwhile, he is the sender in PKE who only has the public key for encryption. The encryption system here could be any types of public key encryption, such as the RSA we mentioned in chapter 2.

Protocol 1. PKE based 1-out-of-2 OT for semi-honest adversary【9】

Main Inputs:S has $x_0, x_1 \in \{0,1\}^l$; R has $\sigma \in \{0,1\}$;
Auxiliary Inputs: Security parameter λ for PKE
Outputs: nothing for S and x_σ for R
1) S randomly chooses two messages $r_0, r_1 \in \{0,1\}^l$, and sends them with encryption algorithm-public key E, pk to the R .
2) R randomly chooses a message $K \in \{0,1\}^l$ to compute $q = E_{pk}(K) \bigoplus m_\sigma$. Then, R sends q to S.
3) S uses his secret key sk to computes the following $K_i = D_{sk}(q \bigoplus m_i),\ i = 0,1$ and encrypt his message by $C_i = x_i \bigoplus K_i,\ i = 0,1$. Then S sends C_0, C_1 to R.
4) R decrypts the desired message by $x_\sigma = C_\sigma \bigoplus K$.

As it shows, the above protocol is based on the public encryption as a sub-protocol and it involves two many steps to finish the task of oblivious transfer. Therefore, researchers put forward a non-interactive protocol with only fewer rounds[10]. Later others amended that with the introduction of random oracle model(ROM) to make sure the encrypted messages for receiver to decrypt are fully randomized[11].

Protocol 2. CDH & ROM based 1-out-of-2 OT for semi-honest adversary 【11】

Auxiliary Inputs: Security parameter λ, a multiplicative group Z_q of order prime q with generator g, ROM with Hash function $H : Z_q \to \{0,1\}^l$, $C \in Z_q$,where its discrete log is unknown to S

1)R randomly chooses $k \in Z_q$ and computes ${}_{\beta_\sigma} = g^k$ and $\beta_{1-\sigma} = C/\beta_\sigma$. He will keep σ and k private and sent β_0, β_1 to S.

2) S firstly checks $C \stackrel{?}{=} \beta_0\beta_1$. If it is not true she will aborts the protocol, otherwise she continues. After that, she randomly chooses $r_1, r_2 \in Z_q$ and sends (e_0, e_1) to R, where $e_i = (g^{r_i}, H(\beta_i^{r_i}) \bigoplus m_i),\ i = 0,1.$

3) R can use the value σ and k to compute desired message:

$$m_\sigma = H((g^{r_\sigma})^k) \bigoplus\ e_\sigma = H((g^{r_\sigma})^k) \bigoplus\ (H(\beta_\sigma^{r_\sigma}) \bigoplus m_\sigma),$$

since $H((g^{r_\sigma})^k) = H(\beta_\sigma^{r_\sigma})$

This protocol is a one-out-of-two oblivious transfer (OT) protocol that uses the computational Diffie-Hellman (CDH) assumption for security. Both the sender and receiver share common inputs, such as k, q, g, Z_q, and C. The value of C is generated by the sender in such a way that the receiver cannot determine its discrete log. The receiver chooses a random element k and computes a pair (β_0, β_1) based on their input bit σ in a way that the sender cannot guess the secret bit. The sender then checks if β_0 and β_1 are equal; if so, the sender aborts the protocol. If not, the sender randomly chooses two numbers r_0 and r_1 from Z_q and computes e_0 and e_1 using a formula. The sender sends these values to the receiver, who then decrypts the required message using their r_σ and k. The protocol uses ROM to ensure security and reduce computation cost.

Protocol 3. efficient DDH & ROM based 1-out-of-2 OT for semi-honest adversary 【11】

Auxiliary Inputs: Security parameter λ, a multiplicative group Z_q of prime order q with generator g, ROM with Hash function $H : Z_q \to \{0,1\}^l$, $C \in Z_q$,where its discrete log is unknown to S

1) R randomly chooses $k \in Z_q$ and computes ${}_{\beta_\sigma} = g^k$ and $\beta_{1-\sigma} = C/\beta_\sigma$. He will keep σ and k private and sent β_0, β_1 to S.

2) S firstly checks $C \stackrel{?}{=} \beta_0\beta_1$. If it is not true she will aborts the protocol, otherwise she continues. After that, she randomly chooses $r \in Z_q$ and sends (e_0, e_1) to R, where $e_i = (g^r, H(\beta_i^r, i) \bigoplus m_i),\ i = 0,1.$

3) R can use the value σ and k to compute desired message:

$$m_\sigma = H((g^{r_\sigma})^k, \sigma) \bigoplus\ e_\sigma = H((g^{r_\sigma})^k, \sigma) \bigoplus\ (H(\beta_\sigma^{r_\sigma}, \sigma) \bigoplus m_\sigma),$$

since $H((g^{r_\sigma})^k) = H(\beta_\sigma^{r_\sigma})$

To achieve a higher efficiency,Moni Naor and Benny Pinkas also put forward a similiar construction with higher efficency: As we can see, the computation cost is reduced by one exponential operation, while the general structure is similar to the previous one. Note that the hash function here use two input: $H(\beta_i^r, i)$. This is due to the fact that in order to fully randomize the β_0^r and β_1^r, we need to consider their index $i = 0,1$, since they are computational indistinguishable.

However, in Protocol 2. & 3., we may need the help of a trusted authority to generate a $C \in Z_q$

to make sure the discrete log of it is unknown to sender–the generation of C is very inconvenient. Moreover, the use of the random oracle is of great complexity. Therefore, scholars wish to find a protocol only relies on the Diffie-Hellman assumption. In the same article, they put forward the following protocl:

Protocol 4. DDH based 1-out-of-2 OT for malicious adversary with privacy【11, 13】

Auxiliary inputs: Security parameter λ, a multiplicative group Z_q of order prime q with generator g.

1) R randomly chooses $a, b, r \in Z_q$, computes $\bar{E}$ as follows and sends $\bar{E}$ to S:

a. If $\sigma = 0$ then $\bar{E} = (g^a, g^b, g^{ab}, g^r)$.

b. If $\sigma = 1$ then $\bar{E} = (g^a, g^b, g^r, g^{ab})$.

2)S receives the tuple $\bar{E}$, denoted by $(\alpha, \beta, k_0, k_1)$, then checks that $\alpha, \beta, k_0, k_1 \in Z_q$ and that $k_0 \neq k_1$. If not, it will stop the protocol. It it is true, she randomly chooses $x_0, x_1, y_0, y_1 \in Z_q$ and computes the following four values:

$$w_0 = \alpha^{x_0} * g^{y_0}, z_0 = (k_0)^{x_0} * \beta^{y_0}, \quad w_1 = \alpha^{x_1} * g^{y_1}, z_1 = (k_1)^{x_1} * \beta^{y_1}.$$

then encrypts m_0, m_1 by $c_i = m_i * z_i, \ i = 0, 1$. After that, S sends R the pairs (w_0, c_0) and (w_1, c_1).

3) R computes $k_\sigma = (w_\sigma)^b$ and outputs $m_\sigma = c_\sigma * (z_\sigma)^{-1}$.

Protocol 4 is a secure protocol based on the decisional Diffie-Hellman (DDH) assumption. In the first step, the receiver chooses two random numbers a, b, and r from the group Z_q, and uses their private input σ to create Diffie-Hellman tuples $(\alpha, \beta, k_0, k_1)$. This construction ensures that the sender cannot determine the receiver's choice bit σ. In step 2, the sender checks if $k_0 = k_1$ and aborts the protocol if it is true. Otherwise, the sender encrypts the messages m_0 and m_1 using z_0 and z_1, respectively, and sends the two pairs (w_i, c_i) to the receiver. Finally, the receiver decrypts the message using their private inputs (σ, b).

The described OT protocol operates under the standard model, which incurs high computation costs. Protocol 5 introduces a new OT protocol under the DDH assumption that reduces the computation costs by three times compared to the previous OT protocol. The input and output of the protocol are the same, but it requires an additional key distribution function (KDF). More details can be found in chapter 2.

In step 1), the receiver randomly generates a, r from the multiplicative group Z_q and uses his private input and creates Diffie-Hellman tuples(α, β) and sends them to the S so that she is unable to differentiate the two numbers. In step 2), the sender randomly generates $b \in Z_q$, using the pair α, β and computing the key pair z_0 and z_1. Having the keys computed, the sender encrypts the value of two messages m_0 and m_1 by using the key distribution (or key derivation function): $e_i = m_i \bigoplus KDF(z_i)$ for $i = 0, 1$. Finally, R uses (a, σ) to computes the required message m_σ.

Protocol 5. Efficient DDH based 1-out-of-2 OT for malicious adversary with privacy【12】

Auxiliary inputs: Security parameter λ , a multiplicative group, Z_q of prime order q with generator g, and key distribution function KDF defined in chapter 2.

1) R randomly chooses $a, r \in Z_q$, computes (α, β) as follows and sends it to S:

a. If $\sigma = 0$ then $(\alpha, \beta) = (g^a, g^r)$.

b. If $\sigma = 1$ then $(\alpha, \beta) = (g^r, g^a)$.

2) S receives (α, β) then checks that $\alpha, \beta \in Z_q$. If not, it aborts. Otherwise, she randomly generates an element $b \in Z_q$ and computes the keys as $(z_0, z_1) = (\alpha^b, \beta^b)$. After that, the sender uses the keys to encrypt the messages as follows:

$$e_i = m_i \bigoplus KDF(z_i), \quad i = 0, 1.$$

The sender then sends the ciphertexts (e_0, e_1) along with $u = g^\beta$ to the receiver.

3) R uses the values of (σ, a) to compute $m_\sigma = e_\sigma \bigoplus KDF(z_\sigma)$, where $z_\sigma = (u)^\alpha$.

In the end, Protocol 6 is a new 1-out-of-2 OT protocol based on the DDH assumption that provides full simulatability security against a malicious receiver. By incorporating a zero-knowledge proof (ZKP) protocol into their OT protocol, this protocol can protect against a malicious receiver. The ZKP sub-protocol enforces the sender and receiver to follow the prescribed execution, making it impossible for a malicious adversary to alter the protocol or provide malicious inputs. More information can be found in Chapter 4.

Protocol 6. DDH based 1-out-of-2 OT for malicious adversary with full Simulatability【13】

Auxiliary inputs: Security parameter λ, a multiplicative group Z_q of order prime q with generator g.

1) R randomly chooses $\alpha_0, \alpha_1, \gamma \in Z_q$, and computes $H_i = g^{a_i} \quad (i = 0, 1)$, $A = g^r$, and $B_i = H_i^r * g^\sigma$. After that, he sends (H_0, H_1, B_0, B_1, A) to S.

2)S receives (H_0, H_1, B_0, B_1, A), and checks that $(H_0, H_1, B_0, B_1, A) \in Z_q$. If not, it aborts.

3) R need to show S that tuples (H, B, A) are Diffie-Hellman tuples under group Z_q with generator g by using zero-knowledge proof, where $H = \frac{H_0}{H_1}, B = \frac{B_0}{B_1}$. In other words:

$$R_{DH} = \{((Z_q, q, g, H, A, B), r) \mid A = g^r \ \& \ B = H^r\}$$

4) S randomly chooses $u_0, u_1, v_0, v_1 \in Z_q$, and sends (e_0, e_1) where: a. $e_0 = (w_0, k_0)$ with $w_0 = A^{u_0} * g^{v_0}$ and $z_0 = B_0^{x_0} * H_0^{y_0} * m_0$ a. $e_1 = (w_1, k_1)$ with $w_1 = A^{u_1} * g^{v_1}$ and $k_1 = (\frac{B_1}{g})^{u_1} * H_1^{v_1} * m_1$ 5) R compute $m_\sigma = \frac{k_\sigma}{(w_\sigma)^{\alpha_\sigma}}$

Chapter 4

Security Proof

Here we prove the security under the three definition we mentioned above based on malicious adversary. The construction among protocols are similar, except when they have a different security level. Thus, we first present some necessary notations, then give security proof of under two level of security definition, security under malicious receiver with privacy and security under malicious receiver with full simulation.

4.1 Notations and Basic Definitions

A function $\mu(\cdot)$ is said to be negligible in n(or simply negligible), if the condition $\mu(n) < 1/p(n)$ holds for every positive polynomial function $p(\cdot)$ and every n that are large enough.
A Probability Ensemble $A = \{A(m,\lambda)\}_{a\in\{0,1\}^l;\lambda\in N}$ is an set containing infinite random variables where each random variable is a function of m and $\lambda \in N$. In cryptography, m stands for the inputs of parties and λ represents the security parameter.
Computationally Indistinguishable: Given two distribution ensembles we mentioned above $A = \{A(m,\lambda)\}_{a\in\{0,1\}^l;\lambda\in N}$ and $B = \{B(m,\lambda)\}_{m\in\{0,1\}^l;\lambda\in N}$, the two ensembles are said to be computationally indistinguishable (or $X \stackrel{c}{\equiv} Y$) , if for every polynomial-time algorithm S there exists a negligible function $\mu(\cdot)$ such that for every $m \in \{0,1\}^l$ and every $\lambda \in N$,

Two-Party Computation: A two-party protocol is a process that turn the input of two party into two output for the two party repectively. To put it formal, we refer to such a process as a ***functionality*** and denote it as $f : \{0,1\}^l \times \{0,1\}^l \to \{0,1\}^l \times \{0,1\}^l$, where $f = (f_1, f_2)$. TO be specific, for every pair of inputs $a, b \in \{0,1\}^l$, the output is actually two: $f_1(a,b), f_2(a,b)$ and party one P_1 with input a will obtain $f_1(a,b)$ and the party two P_2 with b will get obtain $f_2(a,b)$. In 1-out-of-2 oblivious transfer this can be more simple: $((m_0,m_1),\sigma)(/,m_\sigma)$, where m_0, m_1 are two message of the sender and σ is the choice bit made by receiver.

4.2 Proof of privacy security

Theorem 4.1 *Assuming the DDH hard problem holds true under the group Z with generator g. Then, Protocol 4. is a private oblivious transfer, as in Definition3.3 in chapter 2.*

Proof. We firstly prove this non-triviality that are required by definition 3.4. Let m_0, m_1 be S' s two message inputs and σ be the choice input of R. By the protocol 4, the pair (w_σ, c_σ) sent by S to R is defined as $w_\sigma = \alpha^{x_\sigma} g^{y_\sigma}$ and $c_\sigma = \alpha_\sigma \cdot z_\sigma$ where $z_\sigma = (k_\sigma)^{x_\sigma} \beta^{y_\sigma}$. Non-triviality follows from the fact that

$$(w_\sigma)^b = \alpha^{x_\sigma \cdot b} \cdot g^{y_\sigma \cdot b} = g^{(a \cdot b) \cdot x_\sigma} \cdot g^{b \cdot y_\sigma} = (k_\sigma)^{x_\sigma} \cdot \beta^{y_\sigma} = z_\sigma$$

Since $\alpha = g^a$ and $k_\sigma = g^{ab}$. The above quality holds. Therefore, it is indeed that R can compute z_σ and can further get $(z_\sigma)^1$. Therefore, R can get m_σ with correctness $m_\sigma = c_\sigma (z_\sigma)^{-1}$.
After prove the non-triviality, we prove the privacy. For the case that S is an malicious adversary S^*, the privacy property requires that the view of S^* is indistinguishable no matter R has input 0 or input 1. Meanwhile, the view of the malicious sender in the above protocol only contains the message $\bar{E}$.
As the DDH assumption reveals:

$$\{(g^a, g^b, g^{ab})\}_{a,b \leftarrow Z_q} \stackrel{c}{\equiv} \{(g^a, g^b, g^r)\}_{a,\ b\, r \leftarrow Z_q}.$$

We prove by contradiction. Assuming that we have a PPT distinguisher D and a non-negligible function ϵ such that for every n

$$|Pr[D(g^a, g^b, g^{ab}, g^r) = 1] - Pr[D(g^a, g^b, g^r, g^{ab}) = 1]| \geq \epsilon(n)$$

where $a, b, r \leftarrow Z_q$. Then, by subtracting and adding

$$Pr[D(g^a, g^b, g^r, g^\delta = 1)]$$

we have,

$$\begin{aligned} &|Pr[D(g^a, g^b, g^{ab}, g^r) = 1] - Pr[D(g^a, g^b, g^r, g^{ab}) = 1]| \\ \leq\ &|Pr[D(g^a, g^b, g^{ab}, g^r) = 1] - Pr[D(g^a, g^b, g^r, g^\delta) = 1]| \\ +\ &|Pr[D(g^a, g^b, g^r, g^\delta) = 1] - Pr[D(g^a, g^b, g^r, g^{ab}) = 1]| \end{aligned}$$

where $a, b\, r, \delta \leftarrow Z_q$. Therefore, by the contradicting assumption,

$$|Pr[D(g^a, g^b, g^{ab}, g^r) = 1] - Pr[D(g^a, g^b, g^r, g^\delta) = 1]| \geq \tfrac{\epsilon(n)}{2}$$

or

$$|Pr[D(g^a, g^b, g^r, g^\delta) = 1] - Pr[D(g^a, g^b, g^r, g^{ab}) = 1]| \geq \tfrac{\epsilon(n)}{2}$$

Assume the first inequality holds. We construct a distinguisher D for the DDH problem that works as follows. Upon input $\bar{E} = (\alpha, \beta, k)$, the distinguisher D' chooses a random $\delta \leftarrow Z_q$ and hands D the tuple $\bar{E}' = (\alpha, \beta, k, g^\delta)$. The key observation is that on the one hand, if $\bar{E} = (g^a, g^b, g^r)$ then $E' = (g^a, g^b, g^r, g^\delta)$. On the other hand, if $\bar{E} = (g^a, g^b, g^{ab})$ then $\bar{E}' = (g^a, g^b, g^{ab}, g^\delta)$. Noting that in this last tuple r does not appear, and r and δ are distributed identically, we have that $\bar{E}' = (g^a, g^b, g^{ab}, g^r)$. Thus,

$$
\begin{aligned}
&|Pr[D^{'}(g^a, g^b, g^{ab}) = 1] - Pr[D^{'}(g^a, g^b, g^r) = 1]| \\
&= |Pr[D(g^a, g^b, g^{ab}, g^r) = 1] - Pr[D(g^a, g^b, g^r, g^\delta) = 1] \\
&\geq \tfrac{\epsilon(n)}{2}
\end{aligned}
$$

Without doubt this violate the DDH assumption. A similar analysis can be done for the second inequality. Therefore, ϵ must be a negligible function. This further implies that (g^a, g^b, g^{ab}, g^r) must be the same distribution over R' s messages when $\sigma = 0$ while (g^a, g^b, g^r, g^{ab}) is exactly the distribution over R' s message when $\sigma = 1$. Thus, the proof of R' s privacy is concluded. Finally, we prove the privacy under a adversary receiver with malicious power. Here we assume that he has an auxiliary input k. We should note that this proof will be true, even if the R^* is not restricted by PPT. Let $\bar{E} = (\alpha, \beta, k_0, k_1)$ denote $R^*(k)$' s first message, and let a and b be such that $\alpha = g^a$ and $\beta = g^b$. If $k_0 = k_1$ then S aborts. Under this case the privacy is satisfied. Otherwise, let $\tau \in \{0, 1\}$ be such that $k_\tau \neq g^{ab}$ (note that since $k_0 = k_1$ it cannot be that both $k_0 = g^{ab}$ and $k_1 = g^{ab}$). The security of sender relies on the following Theorem:

Theorem *Let $\alpha = g^a$, $\beta = g^b$ and $k_\tau = g^r = g^{ab}$. Then, given a, b and r the pair of values (w_τ, z_τ), where $w_\tau = \alpha^{x_\tau} g^{y_\tau}$ and $z_\tau = (k)^{x_\tau} \cdot \beta^{y_\tau}$, is uniformly distributed when x_τ, y_τ are chosen uniformly in Z_q.*

Proof. *We prove that for every $(\mu_0, \mu_1) \in Z_q \times Z_q$,*

$$Pr[w_\tau = \mu_0 \cap z_\tau = \mu_1] = \tfrac{1}{|Z_q|^2},$$

where the probability is taken over random choices of $x_\tau, y_\tau, w_\tau, z_\tau$ are computed according to the honest sender' s instruction based on the message a from R (this is equivalent to the stated claim). Recall first that

$$w_\tau = \alpha^{x_\tau} \cdot g_\tau^y = g^{a x_\tau} \cdot g^{y_\tau} = g^{a x_\tau + y_\tau},$$

meanwhile

$$z_\tau = (k_\tau)^{x_\tau} \cdot \beta^{y_\tau} = g^{r x_\tau} \cdot g^{b y_\tau} = g^{r x_\tau + b y_\tau}$$

where the above holds because $\alpha = g^a$, $\beta = g^b$ and $k_\tau = g^r$. If we let ϵ_0 and ϵ_1 be the value which satisfy $\mu_0 = g^{\epsilon_0}$ and $\mu_1 = g^{\epsilon_0}$,as a result we then have that the statement holds if and only if there is a single solution to the equations $a x_\tau + y_\tau = \epsilon_0$ and $r \cdot x_\tau + b \cdot y_\tau = \epsilon_1$. since x_τ and y_τ are uniformly distributed in Z_q, Now, there exists a single solution to these equations if and only if the matrix

$\begin{pmatrix} a & 1 \\ r & b \end{pmatrix}$ *is invertible, which is the case here because its determinant is $\alpha \cdot \beta - \gamma$ and by the assumption $a \cdot b \neq r$ and $\alpha \cdot \beta - \gamma \neq 0$. This completes the proof. Proving this claim, it follows that k_τ is uniformly distributed, even given w_τ (and $k_{1-\tau}$, $w_{1-\tau}$). Thus, for every $x, x_\tau \in G$, $k_\tau x_\tau$ is distributed identi-cally to $k_\tau x$. This completes the proof of the sender' s privacy.*

4.3 Proof of full Simulatability

Firstly, we show the correctness given S and R are honest. There are two possible situations: 1) For $\sigma = 0$, it is relatively direct:

$$\frac{z_0}{w_0^{\alpha_0}} = \frac{b_0^{u_0} \cdot h_0^{v_0} \cdot x_0}{a^{u_0 \alpha_0} \cdot g^{v_0 \alpha_0}} = \frac{g^{u_0 r \alpha_0} \cdot g^{v_0 \alpha_0} \cdot x_0}{g^{u_0 \alpha_0 r} \cdot g^{v_0 \alpha_0}} = x_0.$$

2) For $\sigma = 1$, we observe the fact that $b_1 = h_1^r \cdot g$, which means $\left(\frac{b_1}{g}\right)^{u_1} = h_1^{u_1 r} = g^{u_1 r \alpha_1}$. Therefore, we can derive:

$$\frac{z_1}{w_1^{\alpha}} = \frac{\left(\frac{b_1}{g}\right)^{u_1} \cdot h_1^{v_1} \cdot x_1}{a^{u_1 \alpha_1} \cdot g^{v_1 \alpha_1}} = \frac{g^{u_1 r \alpha_1} \cdot g^{v_1 \alpha_1} \cdot x_1}{g^{u_1 \alpha_1 r} \cdot g^{v_1 \alpha_1}} = x_1$$

The above shows that this protocol can function under normal situations. Next, we proof the security:

Theorem 4.2 *Assume that the DDH problem is hard in Z_q with generator g. Then, Protocol 6. securely computes $\mathcal{F}_{\text{OT}}$ in the presence of malicious adversaries.*

Proof. *The security proof for the protocol is carried out in two separate cases: one where the sender S is corrupted, and the other where the receiver R is corrupted. The proof is conducted in a hybrid model, where a trusted third party is introduced to compute an ideal functionality for the zero-knowledge proof of knowledge for the $\mathcal{R}_{\text{DH}}$ assumption.*
The sender S is corrupted. *Let $\mathcal{A}$ be an adversary controlling S. We construct a simulator $\mathcal{S}_{\text{SEND}}$ as follows:*

1. *$\mathcal{S}_{SEND}$ invokes $\mathcal{A}$ upon its input and computes h_0 and h_1 as the honest R would. Then it computes $a = g^r, b_0 = h_0^r$ and $b_1 = h_1^r \cdot g$. (Note that this message is computed differently than an honest R.)*

$\mathcal{S}_{\text{SEND}}$ hands (h_0, h_1, a, b_0, b_1) to $\mathcal{A}$.

2. *$\mathcal{S}_{SEND}$ receives from $\mathcal{A}$ its input to the trusted party computing the zeroknowledge functionality $\mathcal{F}_{\text{ZK}}^{\text{DH}}$; this input is just $(\mathbb{G}, q, g, h_0/h_1, a, b_0/b_1)$. If the input is the values handed by $\mathcal{S}_{\text{SEND}}$ to $\mathcal{A}$, it simulates the trusted party returning 1 ; otherwise it simulates the trusted party returning 0 .*

3. *$\mathcal{S}_{SEND}$ receives from $\mathcal{A}$ two encryptions e_0 and e_1. It then computes $x_0 = \frac{z_0}{w_0^{\alpha_0}}$ and $x_1 = \frac{z_1}{w_1^{\alpha_1}}$ and sends (x_0, x_1) to the trusted party computing the oblivious transfer functionality.*

4. *$\mathcal{S}_{\text{SEND}}$ outputs whatever $\mathcal{A}$ outputs and halts.*

In this part of the proof, we aim to show that the output distribution of the real protocol execution involving $\mathcal{A}$ and R is indistinguishable from the output distribution of the ideal-world simulation involving $\mathcal{S}_{\text{SEND}}$ and R, where $\mathcal{S}_{\text{SEND}}$ generates the values b_0 and b_1 incorrectly due to the multiplication with g. To do this, we will separately consider the case where R's input is $\sigma = 0$ and the case where it is $\sigma = 1$.Denoting Protocol 6 by π_{OT}, we begin by proving that

$$\left\{\text{IDEAL}_{\mathcal{F}_{\text{OT}}, \mathcal{S}_{\text{SEND}}(z), S}\left((x_0, x_1), \sigma, n\right)\right\} \stackrel{c}{=} \left\{\text{HYBRID}^{\text{ZK}}_{\pi_{\text{OT}}, \mathcal{A}(z), S}\left((x_0, x_1), \sigma, n\right)\right\}$$

To distinguish a Diffie-Hellman tuple from a non-Diffie-Hellman tuple with the same probability that it is possible to distinguish the IDEAL and HYBRID executions, we construct a distinguisher D_{DDH} for the DDH *problem. It's worth noting that if the* DDH *problem is hard, it's also difficult to distinguish between specific non-Diffie-Hellman tuples and Diffie-Hellman tuples of*

the form $(\mathbb{G}, q, g, h, g^r, h^r)$. *The full proof is provided in [13].*

D_{DDH} *takes a tuple* $(\mathbb{G}, q, g, h, s, t)$ *and aims to determine if there exists an* r *such that* $s = g^r$ *and* $t = h^r$. *First,* D_{DDH} *chooses* α_0 *and computes* $h_0 = g^{\alpha_0}$ *and* $b_0 = s^{\alpha_0}$. *It then sets* $h_1 = h$ *and* $b_1 = t$. *If* $(\mathbb{G}, q, g, h, s, t)$ *is a Diffie-Hellman tuple, then* D_{DDH} *generates* (h_0, h_1, a, b_0, b_1) *exactly as an honest* R *with input* $\sigma = 0$ *would. This is easily proven by considering* r *such that* $s = g^r$, *which implies that* $b_1 = t = h^r = h_1^r$ *and* $b_0 = s^{\alpha_0} = h_0^r$.

In contrast, if $(\mathbb{G}, q, g, h, s, t)$ *is such that for some* $r, s = g^r$ *and* $t = gh^r$, *then* D_{DDH} *generates* (h_0, h_1, a, b_0, b_1) *exactly as the simulator* $\mathcal{S}_{\text{SEND}}$ *would. This can be shown by considering* $b1 = gh_1^r$ *and* $b_0 = s^{\alpha_0} = h_0^r$. *Next,* D_{DDH} *simulates the rest of the computation as*$\mathcal{S}_{\text{SEND}}$ *by providing* $\mathcal{A}$ *with the values* $h0, h_1, a, b_0, b_1$ *and receiving back the encryptions* e_0, e_1. *Finally, it computes* $x_0 = \frac{z_0}{w_0^{\alpha_0}}$ *and sets this as* R*'s output, which it combines with* $\mathcal{A}$*'s output to generate a joint output distribution of* $\mathcal{A}$ *and* R.
If D_{DDH} *receives a non-Diffie-Hellman tuple with the fourth element being* gh^r, *then the output of* D_{DDH} *is identical to the output distribution of the ideal-world execution with* S.
The receiver R ***is corrupted.*** *Let* $\mathcal{A}$ *be an adversary controlling* R. *We construct a simulator* $\mathcal{S}_{\text{REC}}$ *as follows:*

1. $\mathcal{S}_{\text{REC}}$ *invokes* $\mathcal{A}$ *upon its input and receives the vector* (h_0, h_1, a, b_0, b_1).
2. $\mathcal{S}_{\text{REC}}$ *checks that all* $h_0, h_1, a, b_0, b_1 \in \mathbb{G}$*; if not, it sends* $\perp$ *to the trusted party for* $\mathcal{F}_{\text{OT}}$, *simulates* S *aborting, outputs whatever* $\mathcal{A}$ *outputs and halts.*
3. *Otherwise,* $\mathcal{S}_{\text{REC}}$ *obtains from* $\mathcal{A}$ *its input* $((\mathbb{G}, q, g, h, a, b), r)$ *for the trusted party computing the zero-knowledge functionality* $\mathcal{F}_{\text{ZK}}^{\text{DH}}$. *If the conditions for outputting* $(1, \lambda)$ *are not met then* $\mathcal{S}_{\text{REC}}$ *sends* $\perp$ *to the trusted party for* $\mathcal{F}_{\text{OT}}$ *and aborts. Otherwise,* $\mathcal{S}_{\text{REC}}$ *proceeds.*
4. $\mathcal{S}_{\text{REC}}$ *computes* $\ell = b_0/h_0^r$. *If* $\ell = 1, \mathcal{S}_{\text{REC}}$ *sets* $\sigma = 0$*; if* $\ell \neq 1$ *(including the case where* $\ell = g), \mathcal{S}_{\text{REC}}$ *sets* $\sigma = 1$. *Then,* $\mathcal{S}_{\text{REC}}$ *sends* σ *to the trusted party computing the oblivious transfer functionality and receives back the string* x_σ.
5. $\mathcal{S}_{\text{REC}}$ *computes* $e_\sigma = (w_\sigma, z_\sigma)$ *as an honest* S *would, using the value* x_σ. *In contrast, it computes* $e_{1-\sigma}$ *using* $x_{1-\sigma} = 1$.
6. $\mathcal{S}_{\text{REC}}$ *outputs whatever* $\mathcal{A}$ *outputs and halts.*

We continue by proving that

$$\left\{\text{IDEAL}_{\mathcal{F}_{\text{OT}}, \mathcal{S}_{\text{REC}}(z), R}\left((x_0, x_1), \sigma, n\right)\right\} \stackrel{c}{\equiv} \left\{\text{HYBRID}_{\pi_{\text{OT}}, \mathcal{A}(z), R}^{\text{ZK}}\left((x_0, x_1), \sigma, n\right)\right\}.$$

To prove that the distribution over $\mathcal{A}$*'s output is computationally indistinguishable between a hybrid execution with* S *and a simulation with* $\mathcal{S}$REC, *it is sufficient to demonstrate that the distributions are identical. Since* S *has no output, it is clear that* $\mathcal{A}$*'s view is identical until the last message, and the only difference lies in the construction of* $e1 - \sigma$. *Therefore, it is only necessary to show that for every* $x_{1-\sigma}$, *the distribution over* $e_{1-\sigma}$ *is the same when generated by an honest* S *or by* $\mathcal{S}$REC. *We perform this analysis separately for* $\ell = 1$, $\ell = g$, *and* $\ell \notin 1, g$,

where ℓ is the value computed by $\mathcal{S}$REC by evaluating b_0/h_0^r). To distinguish between values generated by the simulator $\mathcal{S}_{\text{REC}}$ and those generated by a real S, we add a tilde to the former.
1) For $\ell = 1$: In this case we know that for some r it holds that $a = g^r$ and $b_0/b_1 = (h_0/h_1)^r$ (this is given due to the zero-knowledge protocol). Combining this with the fact that $b_0/h_0^r = 1$ we have that

$$b_1 = b_0 \cdot \left(\frac{h_1}{h_0}\right)^r = (h_1)^r$$

Letting α be such that $h_1 = g^\alpha$, we have that $b_1 = (h_1)^r = (g^\alpha)^r$. (Such an α is guaranteed to exist because $h_1 \in \mathbb{G}$ as checked by $\mathcal{S}_{\text{REC}}$ and the honest S.) This implies that $\frac{b_1}{g} = g^{r\alpha} \cdot g^{-1}$. Finally, for every $x_1 \in \mathbb{G}$ it holds that there exists a value s such that $x_1 = g^s$; this holds because g is a generator and x_1 is in $\mathbb{G}$). Recalling that $w_1 = b^{u_1} \cdot g^{v_1}$ and $z_1 = \left(\frac{b_1}{g}\right)^{u_1} \cdot h_1^{v_1} \cdot x_1$ and that $\mathcal{S}_{\text{REC}}$ uses $x_1 = 1$, we have that the distributions generated over e_1 are
generated by $\mathcal{S}_{\text{REC}}$:

$$\tilde{w}_1 = a^{\tilde{u}_1} \cdot g^{\tilde{v}_1} = g^{r\tilde{u}_1 + \tilde{v}_1}$$

$$\tilde{z}_1 = \left(\frac{b_1}{g}\right)^{\tilde{u}_1} \cdot h_1^{\tilde{v}_1} = g^{r\alpha\tilde{u}_1 + \tilde{v}_1\alpha} \cdot g^{-\tilde{u}_1}$$

generated by S :

$$w_1 = a^{u_1} \cdot g^{v_1} = g^{ru_1 + v_1}$$

$$z_1 = \left(\frac{b_1}{g}\right)^{u_1} \cdot h_1^{v_1} \cdot x_1 = g^{r\alpha u_1 + v_1\alpha} \cdot g^s \cdot g^{-u_1}$$

We claim that the above distributions are identical. In order to see this, let $\tilde{u}_1 = u_1 - s$ and $\tilde{v}_1 = v_1 + rs$. For a fixed r and s and a uniformly distributed u_1 and v_1, the values $\tilde{u}_1$ and $\tilde{v}_1$ chosen in this way are also uniformly distributed. Plugging these values in, we have that

$$\tilde{w}_1 = g^{r\tilde{u}_1 + \tilde{v}_1} = g^{r(u_1 - s) + v_1 + rs} = g^{ru_1 + v_1}$$

$$\begin{aligned}\tilde{z}_1 &= g^{r\alpha\tilde{u}_1 + \tilde{v}_1\alpha - \tilde{u}_1} = g^{r\alpha(u_1 - s) + \alpha(v_1 + rs) - (u_1 - s)} \\ &= g^{r\alpha u_1 + \alpha v_1 - u_1 + s} = g^{r\alpha u_1 + \alpha v_1} \cdot g^s \cdot g^{-u_1},\end{aligned}$$

which are the exact values w_1 and z_1 generated by an honest S. Thus, the distribution viewed by $\mathcal{A}$ in this case is the same in the hybrid and simulated executions.
2) For $\ell = g$: In this case we know that for some r it holds that $a = g^r$ and $b_0 = gh_0^r$ (this is given due to the zero-knowledge protocol and the fact that $b_0/h_0^r = g$, as in the previous case). Now, as above there exists a value α such that $h_0 = g^\alpha$ (because $h_0 \in \mathbb{G}$) and we know that $a = g^r$, implying that $b_0 = gh_0^r = ga^\alpha = g \cdot g^{r\alpha}$. Furthermore, for every $x_0 \in \mathbb{G}$ there exists a value s such that $x_0 = g^s$.
Recalling that $w_0 = a^{u_0} \cdot g^{v_0}$ and $z_0 = b_0^{u_0} \cdot h_0^{v_0} \cdot x_0$ and that $\mathcal{S}_{\text{REC}}$ uses $x_0 = 1$, we have that the distributions generated over e_0 are
generated by $\mathcal{S}_{\text{REC}}$:

$$\tilde{w}_0 = a^{\tilde{u}_0} \cdot g^{\tilde{v}_0} = g^{r\tilde{u}_0 + \tilde{v}_0}$$

$$\tilde{z}_0 = b_0^{\tilde{u}_0} \cdot h_0^{\tilde{v}_0} = g^{r\alpha\tilde{u}_0 + \tilde{v}_0\alpha} \cdot g^{\tilde{u}_0}$$

generated by S :

$$w_0 = a^{u_0} \cdot g^{v_0} = g^{ru_0+v_0}$$

$$z_0 = b_0^{u_0} \cdot h_0^{v_0} \cdot x_0 = g^{r\alpha u_0 + v_0\alpha} \cdot g^s \cdot g^{u_0}$$

As above, let $\tilde{u}_0 = u_0 + s$ and $\tilde{v}_0 = v_0 - rs$; again these are uniformly distributed if u_0 and v_0 are uniformly distributed. Then,

$$\tilde{w}_0 = g^{r\tilde{u}_0+\tilde{v}_0} = g^{r(u_0+s)+v_0-rs} = g^{ru_0+v_0}$$

$$\tilde{z}_0 = g^{r\alpha\tilde{u}_0+\tilde{v}_0\alpha} \cdot g^{\tilde{u}_0} = g^{r\alpha(u_0+s)+\alpha(v_0-rs)+(u_0+s)} = g^{r\alpha u_0+\alpha v_0} \cdot g^s \cdot g^{u_0},$$

which are the exact values generated by an honest S. Thus, the distribution viewed by $\mathcal{A}$ in this case is the same in the hybrid and simulated executions.

3) For $\ell \notin \{1, g\}$: In this case, $\mathcal{S}_{\text{REC}}$ sets $\sigma = 1$ and so generates the distribution for e_1 exactly as an honest S would. In contrast to above, here we only know that $a = g^r$ and $b_0 = \ell \cdot h_0^r$ for some ℓ. This means that for α such that $h_0 = g^\alpha$ we have that $b_0 = \ell \cdot g^{r\alpha}$. However, since $b_0 \in \mathbb{G}$ (as checked by $\mathcal{S}_{\text{REC}}$) and $h_0 = g^{r\alpha}$ is also in $\mathbb{G}$, it follows that $\ell \in \mathbb{G}$; let t be such that $\ell = g^t$ and let s be such that $x_0 = g^s$. As above, we write the distributions generated by $\mathcal{S}_{\text{REC}}$ and an honest S :
Generated by $\mathcal{S}_{\text{REC}}$:

$$\tilde{w}_0 = g^{r\tilde{u}_0+\tilde{v}_0}$$

$$\tilde{z}_0 = g^{r\alpha\tilde{u}_0+\tilde{v}_0\alpha} \cdot \ell^{\tilde{u}_0} = g^{r\alpha\tilde{u}_0+\tilde{v}_0\alpha} \cdot g^{t\tilde{u}_0}$$

Generated by S :

$$w_0 = g^{ru_0+v_0}$$

$$z_0 = g^{r\alpha u_0+v_0\alpha} \cdot g^b \cdot \ell^{u_0} = g^{r\alpha u_0+v_0\alpha} \cdot g^s \cdot g^{tu_0}.$$

Let $\tilde{u}_0 = u_0 + s/t$ and $\tilde{v}_0 = v_0 - rs/t$. The crucial point here is that $\mathbb{G}$ is a group of prime order, and thus t has an inverse modulo q, meaning that s/t is well defined (recall that we can work modulo q in the exponent). Thus, we can define $\tilde{u}_0$ and $\tilde{v}_0$ in this way, and as above, if v_0 and u_0 are uniformly distributed then so are $\tilde{v}_0$ and $\tilde{u}_0$. Plugging these values in as above, we have that $\tilde{w}_0, \tilde{z}_0$ are distributed identically to w_0, z_0, as required.
This completes the proof of the theorem.

Chapter 5

Conclusion

under the context of secure multi-party computation, the potential of oblivious transfer is significant. As a fundamental tools in this field, OT is of great important, leading some of the crucial application such as personal information retrieval and so on. Although it is relatively new concept in cryptography, its intriguing nature has made many researchers to work on it. With the help of the mature concept in security definition of SMC, OT has clrealy define its security facing different types of adversary. Under the hard work of numerous scholars, different types of structures has been put forward, in order to achieve a higher efficiency and better performance in security. The rigorous proof has also been made as the same time. Under the assumption of Diffie-Hellman, especially under the DDH assumption, we can construct very efficient 1-out-of-2 oblivious transfer with a few computation while keep the security under the malicious adversary with full simulation. However, under new tools like quantum computer and new types of adversary like covert adversary, current model may be no loner feasible. Therefore, more research should be conducted under these new topics.

Bibliography

[1]YADAV, V. K., ANDOLA, N., VERMA, S., & VENKATESAN, S. (2022). A Survey of Oblivious Transfer Protocol. ACM Computing Surveys, 54, 1–37. https://doi-org.ez.xjtlu.edu.cn/10.1145/3503045

[2]Oded Goldreich, Silvio Micali, and Avi Wigderson. 2019. How to play any mental game, or a completeness theorem for protocols with honest majority. In Providing Sound Foundations for Cryptography: On the Work of Shai Goldwasser and Silvio Micali. 307-328.

[3]Donald Beaver and Shaft Goldwasser. 1989. Multiparty computation with faulty majority. In Conference on the Theory and Application of Cryptology. Springer, 589-590.

[4]. 1991. Foundations of secure interactive computing. In Annual International Cryptology Conference. Springer, 377-391.

[5]Michael O Rabin. 2005. How To Exchange Secrets with Oblivious Transfer. IACR Cryptol. ePrint Arch. 2005, 187 (2005)

[6]Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In 27th Annual Symposium on Foundations of Computer Science (sfcs 1986). IEEE, 162-167.

[7]Diffie W, Hellman M. New directions in cryptography. IEEE Transactions on Information Theory, Information Theory, IEEE Transactions on, IEEE Trans Inform Theory. 1976;22(6):644-654. doi:10.1109/TIT.1976.1055638

[8]Rivest RL, Shamir A, Adleman L. A Method for Obtaining Digital Signatures and Public-Key Components. Communications of the ACM. 1978;21(2):120-126. doi:10.1145/359340.359342

[9]Shimon Even, Oded Goldreich, and Abraham Lempel. 1985. A randomized protocol for signing contracts. Commun. ACM 28, 6 (1985), 637-647.

[10]Mihir Bellare and Silvio Micali. 1989. Non-interactive oblivious transfer and applications. In Conference on the Theory and Application of Cryptology. Springer, 547-557.

[11]Moni Naor and Benny Pinkas. 2001. Eicient oblivious transfer protocols.. In SODA, Vol.

1. 448-457.

[12] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. 2013. More eicient oblivious transfer and extensions for faster secure computation. In Proceedings of the 2013 ACM SIGSAC conference on Computer communications security. 535-548.

[13] Carmit Hazay and Yehuda Lindell. 2010. Eicient secure two-party protocols: Techniques and constructions. Springer Science Business Media.

[14] Michele Ciampi and Claudio Orlandi. 2018. Combining private set-intersection with secure two-party computation. In International Conference on Security and Cryptography for Networks. Springer, 464-482.

[15] Emiliano De Cristofaro and Gene Tsudik. 2010. Practical private set intersection protocols with linear complexity. In International Conference on Financial Cryptography and Data Security. Springer, 143-159.

[16] Bo Bi, Darong Huang, Bo Mi, Zhenping Deng, and Hongyang Pan. 2019. Eicient LBS Security-Preserving Based on NTRU Oblivious Transfer. Wireless Personal Communications 108, 4 (2019), 2663-2674.

[17] Hoda Jannati and Behnam Bahrak. 2017. An oblivious transfer protocol based on elgamal encryption for preserving location privacy. Wireless Personal Communications 97, 2 (2017), 3113-3123.

[18] Claude Crépeau. 1987. Equivalence between two lavours of oblivious transfers. In Conference on the Theory and Application of Cryptographic Techniques. Springer, 350-354.

[19] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. 2000. Single database private information retrieval implies oblivious transfer. In International Conference on the Theory and Applications of Cryptographic Techniques. Springer, 122-138.

[20] Diffie, W. and M. Hellman (1976). "New directions in cryptography." IEEE Trans. Info. Theory, IT-22, 644–654.

[21]Brands, S. (1993). "An efficient off-line electronic cash system based on the representation problem." CWI TR CS-R9323.

[22]Canetti, R. (2005). Decisional Diffie-Hellman Assumption. In: van Tilborg, H.C.A. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA . https://doi.org/10.1007/0-

387-23483-7-99

[23]Bellare, Mihir and Philip Rogaway (1993). "Random oracles are practical: A paradigm for designing efficient protocols." 1st ACM Conference on Computer and Communications Security, Proceedings, Fairfax, November 1993. ACM Press, New York, 62–73.

[24]Rivest, Ronald (1992). The MD5 Message-Digest Algorithm; RFC1321. http://www.faqs.org/rfcs/rfc1321.html

[25]National Institute of Standards and Technology (NIST) (1995). Secure Hash Standard. Federal Information Processing Standards Publication (FIPS PUB 180-1).

[26]Krawczyk, H. (2010). Cryptographic Extraction and Key Derivation: The HKDF Scheme. In: Rabin, T. (eds) Advances in Cryptology –CRYPTO 2010. CRYPTO 2010. Lecture Notes in Computer Science, vol 6223. Springer, Berlin, Heidelberg. https://doiorg.ez.xjtlu.edu.cn/10.1007/978-3-642-14623-7-34

[27]Schoenmakers, B. (2005). Zero-knowledge. In: van Tilborg, H.C.A. (eds) Encyclopedia of Cryptography and Security. Springer, Boston, MA . https://doi.org/10.1007/0-387-23483-7-463