More robust security specifications for 1 out of 2 Oblivious Transfer

Abdul Hannan Ahsan

Master of Science

School of Computer Science

McGill University

Montreal,Quebec

February 2006

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

ABSTRACT

An 1 out of 2 Oblivious Transfer ($\binom{2}{1}$-OT) is a two-party computation in which a sender $\mathcal{A}$ owning two secret bits b_0, b_1 can transfer one to them, b_c, to receiver Bob who chooses c. This is done in such a way that Alice does not learn anything about c and Bob does not learn anything other than bit b_c. Security specifications for $\binom{2}{1}$-OT are defined in terms of privacy and correctness. Privacy constraint enforces that $\mathcal{B}$ gets the bit of his choice and gains no knowledge about the other input bit of $\mathcal{A}$, and $\mathcal{A}$ does not gain any knowledge of c. Correctness constraint enforces that the output received by $\mathcal{B}$ is not corrupted by a dishonest $\mathcal{A}$. Traditionally privacy and correctness have been defined as disjoint security specifications, and various attempts have been made to merge these two constraints.

We present a new set of security specifications for $\binom{2}{1}$-OT in which these two constraints are enforced concurrently. In addition, unlike the previous specifications, our new correctness constraint deals with the correct view of the protocol instead of output only. We also extend these security specifications to another variant of $\binom{2}{1}$-OT called 1 out of 2 XOR Oblivious Transfer.

SOMMAIRE

Le transfert inconscient ($\binom{2}{1}$-OT) est une primitive cryptographique permettant à l'expéditeur $\mathcal{A}$ possédant deux bits secrets (b_0, b_1) d'en transmettre un au récepteur $\mathcal{B}$ qui obtient, au choix, b_c. La primitive garantit d'une part que $\mathcal{B}$ n'apprend rien de plus sur les deux bits, et d'autre part que son choix c reste secret. Les spécifications de la sécurité de la primitive $\binom{2}{1}$-OT comprennent deux volets: celui de la protection du secret (privacy), et celui de la protection de la correction (correctness). Le premier garantit que $\mathcal{B}$ n'obtient qu'un seul des deux bits, au choix, et que $\mathcal{A}$ n'apprend rien de ce choix. Le deuxième garantit qu'un expéditeur malhonnête admissible $\tilde{\mathcal{A}}$ ne pourrait pas corrompre la sortie de $\mathcal{B}$. Ces deux volets sont d'habitude traités séparément, et plusieurs tentatives de les joindre dans une seule définition de sécurité ont été faites dans le passé.

Dans ce mémoire, nous présentons de nouvelles spécifications de sécurité pour $\binom{2}{1}$-OT où les deux volets sont traités conjointement. À la différence des spécifications antérieures, nos spécifications traitent de la vue (view) des deux parties et non seulement de leurs sorties. Nos spécifications sont ensuite adaptées à une variante du transfert inconscient, le transfert inconscient XOR, permettant à $\mathcal{B}$ d'apprendre soit l'un des deux bits soit leur ou-exclusif, tout en gardant son choix secret.

DEDICATION

This thesis would be incomplete without a mention of the support given by my wife and my best friend, Shaheen, to whom this thesis is dedicated. She kept my spirits up when this thesis seemed interminable. Without her I doubt it should ever have been completed.

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to my thesis supervisor, Claude Crépeau for his stimulating guidance and constant encouragement during this thesis work. I was especially impressed by not only his commitment to high standards of research, but also his compassionate approach towards the personal needs of the students.

I am immensely thankful to George Savvides for his generous help in proof-reading and layout of this thesis. The thesis in the current shape would not have been possible without his help. Using his own words: "It has been a pleasure and a privilege" to work with him.

TABLE OF CONTENTS

LIST OF NOTATIONS

$\overline{\mathcal{A}}$	Honest player Alice
$\widetilde{\mathcal{B}}$	Dishonest player Bob
$[\mathcal{A}, \mathcal{B}](\alpha, \beta, \kappa)$	A protocol with two players, $\mathcal{A}$ and $\mathcal{B}$. α is $\mathcal{A}$'s private input, and β is $\mathcal{B}$'s private input. κ is their common input
$[\mathcal{A}, \mathcal{B}]^*(\alpha, \beta, \kappa)$	The view of a protocol
$[\mathcal{A}, \mathcal{B}]^*_{\mathcal{A}}(\alpha, \beta, \kappa)$	A's view of a protocol
$\mathbf{H}(X)$	Entropy of a random variable X
$\mathbf{H}(X, Y)$	Joint Entropy of two random variables X and Y
$\mathbf{I}(X; Y)$	Mutual Information between two random variables X and Y
$\binom{2}{1}$–OT	1-out-of-2 Oblivious Transfer
TOX-$\binom{2}{1}$	1-out-of-2 Reverse XOR Oblivious Transfer
TO–$\binom{2}{1}$	1 out of 2 Reverse Oblivious Transfer
$\binom{2}{1}$-XOT	1-out-of-2 XOR Oblivious Transfer

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

1.1 Multi-party Computation

Cryptography, the science of encrypting and decrypting information, dates as far back as 1900 BC [17] when a scribe in Egypt first used a derivation of the standard hieroglyphics of the day for secret communication. Cryptography has come a long way since then and has rapidly evolved over the last decade to deal with the security of more complex multi-party scenarios. Such scenarios range from such seemingly simple tasks as fair coin flips when the participants are far apart, to complex scenarios such as electronic voting, on-line auction and anonymous transactions. In cryptographic terms, these scenarios are categorized as *multi-party computations*. In such settings two or more parties with private inputs jointly compute some predetermined function. The computation is done in such a way that each honest party's input and output is protected from the malicious intents of dishonest participants.

A number of security definitions have been proposed for multi-party computation [8, 20, 1, 2, 4, 13] . These definitions aim to formalize important security aspects that must be enforced while designing protocols for such computations. The most central of these security aspects are:

- ***Privacy:*** No party should learn anything more than its prescribed output. In particular, the only information that should be learned about other parties' inputs is what can be derived from the output itself.
- ***Correctness:*** Each party is guaranteed that the output it receives is valid according to a predefined set of protocol specifications.
- ***Independence of Inputs:*** Dishonest parties must choose their inputs independently of honest parties' inputs.
- ***Corresponding Input/Output pairs:*** The output of an honest party must be consistent with its input.

Complex multi-party protocols are often built by using much simpler and fundamental protocols as subroutines. These simple protocols are well studied and their security has been rigorously scrutinized by the research community. We aim to construct complex protocols so that their security directly relies upon the security of simpler subroutines. The reasoning behind this is very simple: we can provide a proof of security for the complex protocol based on the well-tested security of those subroutines. In cryptographic terms, these relatively simple protocols are generally referred to as *cryptographic primitives*. Given a complex protocol α which uses a cryptographic primitive β as a subroutine, we say that the security of α is *equivalent* to the security of β, if any security violation of α results in a security violation of β. In scientific literature the term *reduction* is also used in a similar context to show equivalence of security.

The equivalence between cryptographic protocols is a major research area [3, 6, 9, 18, 7, 8, 19, 10] and a large number of cryptographic protocols have been shown to be equivalent to one another.

One very important and well studied cryptographic primitive is called *Oblivious Transfer*.

1.2 Oblivious Transfer

The notion of Oblivious Transfer was first introduced by Rabin[21] and takes place between two parties, a sender Alice and a receiver Bob. Alice owns a message m that Bob does not know. After a successful run of the protocol, with probability $\frac{1}{2}$ Bob learns the value of m. This is done in such a way that Bob knows whether he got m or not. On the other hand Alice does not know whether m has been learned. Rabin also suggested a an implementation based on the difficulty of factoring. This implementation was *correct* provided both the parties acted honestly.

Oblivious Transfer (OT) is a cornerstone in the foundation of cryptography and has become the basis for realizing a broad class of interactive protocols such as bit commitment, zero-knowledge proofs, and more general secure multi-party computations [22, 14, 15, 18]. The importance and extreme generality of this protocol was evidenced by the work of Brassard, Crépeau, and Robert [3], Crépeau [6], and Kilian [18], who basically showed that every two-party protocol can be securely realized using only Oblivious Transfer as cryptographic primitive. The results of [3] and [6] show that a very general disclosure problem (all-or-nothing disclosure of secrets (ANDOS)) can be solved through a set of reductions from an

Oblivious Transfer protocol. Kilian showed how to use the ANDOS primitive to implement the very general oblivious transfer circuit evaluation protocol.

1.2.1 1-out-of-2 Oblivious Transfer

A *1-out-of-2 Oblivious Transfer* ($\binom{2}{1}$–OT) is a primitive in which a sender Alice owning two secret bits b_0, b_1 can transfer one to them, b_c, to receiver Bob who chooses c. This is done in such a way that Alice does not learn anything about c and Bob does not learn anything other than bit b_c. This primitive was first introduced by Even, Goldreich and Lempel [11] with applications to contract signing protocols.

1.3 Motivation of this research

The security of two-party computation like $\binom{2}{1}$–OT has been traditionally defined in terms of *privacy* and *correctness*. In recent literature, various attempts have been made [20, 1, 5, 13] to merge these two fundamental properties into a unified definition of security. Micali [20] argued that though privacy and correctness are the fundamental aspects of secure computation, the logical connective *"and"* does not combine them adequately. Correctness and privacy may seem to be conflicting requirements, and capturing in the most general sense what it means to simultaneously enforce them is quite difficult. To obtain a satisfactory notion of security, privacy and correctness should not be handled independently but need to be *blended* in a proper manner.

The main motivation for this research was to investigate a new and more robust set of security specifications for $\binom{2}{1}$–OT in which privacy and correctness

constraints were defined concurrently and treated as a single unified security constraint. We present our main results in chapter 4.

In the latter part of the thesis, we present the security specifications of another variant of $\binom{2}{1}$–OT called 1-out-of-2 Reverse XOR Oblivious Transfer (TOX-$\binom{2}{1}$). We also present an efficient reduction showing that an attacker to TOX-$\binom{2}{1}$ privacy can be reduced to an attacker to $\binom{2}{1}$–OT privacy.

Due to the breadth of material and for the purpose of clarity, we have expanded the background material to span the next two chapters, each dealing with individual topics. The breakup of chapters is as follows:

- ***Chapter 2*** presents the notations and definitions used in this thesis. This chapter also discusses in detail what we mean by protocol and protocol security, and provides formal definitions. Mathematical tools used in this thesis are also presented in this chapter.
- ***Chapter 3*** presents the current security specifications of $\binom{2}{1}$–OT dealing formally with *correctness* and *privacy*. This definition was first presented in [2, 4] and constitutes the starting point of this research work. We present certain reservations about the completeness of these security specifications due to the disjoint nature of the privacy and correctness constraints.
- ***Chapter 4*** presents our main results and provide new security specifications for $\binom{2}{1}$–OT.
- ***Chapter 5*** presents new security specifications for TOX-$\binom{2}{1}$. We also present an efficient reduction showing that an attacker to TOX-$\binom{2}{1}$ privacy can be reduced to an attacker to $\binom{2}{1}$–OT privacy.

- ***Chapter 6*** presents the security specifications and a construction of TO–$(\binom{2}{1})$.
- ***Chapter 7*** presents the conclusions.

CHAPTER 2
PRELIMINARIES

We present the notation and terminology used in this research work. This chapter also provides the necessary mathematical tools used in this research work.

2.1 Notation and Terminology

2.1.1 Parties, Honest and otherwise

Throughout this research work, the interaction will take place between two parties, *Alice* and *Bob*. We will denote Alice by $\mathcal{A}$ and Bob by $\mathcal{B}$. If we want to indicate that a party is acting honestly (i.e. the party is participating in accordance to a legitimate set of steps as described by the protocol specifications), we will place a *bar* over that party's symbol. For example $\overline{\mathcal{A}}$ denotes an honest Alice. We define *dishonest* behavior to be any action taken by a party which is not valid according to protocol specifications. This includes a party not following a legitimate set of steps as prescribed by the protocol specifications. A dishonest behavior will be denoted by a placing a *tilde* on the party's symbol. $\widetilde{\mathcal{B}}$ denotes a dishonest Bob.

We denote as $\wp$ any program that a party may run to execute the protocol. The ownership of the program is denoted by putting the owner's symbol as a subscript to $\wp$. Similar to the previous set of notations, the symbol on the top of $\wp$ describes the behavior of the program. For example, $\widetilde{\wp_a}$ represents a dishonest program run by Alice for the execution of the protocol.

2.1.2 Deterministic and Probabilistic Oracle Machines

An Oracle machine is a Turing machine that is augmented so that it can ask questions to an outside oracle. An oracle machine has an additional tape, called the *oracle tape*, and two special states called *oracle invocation* and *oracle appeared* [12]. The output distribution of the oracle machine $\mathcal{M}$, on input x and with access to oracle f is denoted by $\mathcal{M}^f(x)$. The computation of a deterministic oracle machine $\mathcal{M}$ on input x, with access to oracle $f : \{0,1\}^* \rightarrow \{0,1\}^*$ is defined by the *successive-configuration* relation. For configurations with states other than *oracle invocation*, the next configuration is defined as usual. Let γ be a configuration in *oracle invocation* state having q as the contents of the oracle tape. Then the configuration following γ is identical to γ, except that the state will be *oracle appeared*, and the contents of the tape will be $f(q)$. The string q is $\mathcal{M}$'s query and $f(q)$ is called *oracle reply*. The computation of a probabilistic oracle machine is defined analogously.

2.1.3 Interactive Turing Machines

We represent the behavior of a party by an Interactive Turing Machine (ITM). An ITM $\mathcal{M}$ is a deterministic multi-tape Turing machine [12] consisting of the following tapes.

- ***A read-only input tape*** which holds $\mathcal{M}$'s private input.
- ***A read-only random tape*** which holds $\mathcal{M}$'s random coin flips.
- ***A read-and-write work tape*** used by $\mathcal{M}$ for private internal computations.
- ***A write-only output tape*** whose contents at termination are called the *output* of $\mathcal{M}$.

- ***A read-and-write switch tape*** which indicates whether $\mathcal{M}$ is active or not. Each ITM is associated with a single bit $\sigma \in \{0,1\}$ called its *identity*. An ITM is said to be *active* in a configuration if the content of its switch tape equals the machine's identity, Otherwise the machine is said to be *idle*. While idle, the state of the machine, the location of its head on the various tapes, and the contents of the writable tapes of the ITM are not modified.
- ***A pair of communication tapes*** which are used to communicate with the other ITM. One of the tapes is *read-only* and the other one is *write-only*. The content written on the *write-only* communication tape when $\mathcal{M}$ is active is called the *message sent* at that period. Likewise, the content read from the read-only communication tape during an active period is called the *message received*.

In addition to these tapes, we allow ITM to have an additional read-only tape called ***the auxiliary-input tape***. We use auxiliary tapes to model individual local inputs of each machine. These tapes may contain the results of any local computation done on the inputs, or the history of previous interactions.

2.1.4 Joint Computation of Two ITMs

We represent the protocol as a joint computation of Two ITMs. Two ITMs are said to be *linked* if they have opposite identities, their input tapes coincide, their switch tapes coincide, and the read-only communication tape of one machine coincides with the write-only communication tape of the other machine and vice versa. The other tapes of both machines (i.e. random tape, work tape, output tape, and auxiliary input tape) are distinct and inaccessible by the other machine.

The *joint computation* of a *linked pair* of ITM, on a common input x is a sequence of pairs representing the local configurations of both machines. That is, each pair consists of two strings, each representing the local configuration of one of the machines. In each pair one machine is active while the other is idle. The first pair in the sequence consists of initial configurations corresponding to the common input x, with the contents of the switch tape set to zero.

If one machine halts while the switch tape still holds its identity, then we say that both machines have halted. The *outputs* of the protocol can be determined by looking at the output tapes of each party. Given two parties $\mathcal{A}$ and $\mathcal{B}$, denote as ($\mathcal{A}$,$\mathcal{B}$) a connecting pair of parties (CPP) doing joint computations.

2.2 View of a Protocol

During the execution of a protocol, both $\mathcal{A}$ and $\mathcal{B}$ have access to some information. This information is completely determined by the internal coin tosses and the messages received from the other party. This is a random variable and is called the *view* of the protocol. We formalize the notation as follows:

$[\wp_a, \wp_b](\alpha)(\beta)$ denotes the random variable that describes the outputs obtained by $\mathcal{A}$ and $\mathcal{B}$ when they execute together the programs $\wp_a$ and $\wp_b$ on inputs α and β respectively. Note that $[\wp_a, \wp_b](\alpha)(\beta)$ is a random variable since $\widetilde{\wp_a}$ and $\widetilde{\wp_b}$ may be probabilistic programs.

Let $[\wp_a, \wp_b]^*(\alpha)(\beta)$ be a random variable that describes the total information (including not only the messages received and issued by the parties but also the result of any local random sampling they may have performed) acquired during the execution of the protocol $[\wp_a, \wp_b]$ on inputs (α, β). Let $[\wp_a, \wp_b]_P(\alpha)(\beta)$ and

$[\wp_a, \wp_b]^*_P(\alpha)(\beta)$ be the marginal random variables obtained by restricting the above to only one party P. The latter is called the *view* of P [16].

2.3 Simulation Paradigm

Simulation is one of the most important paradigms used in providing proofs of security of cryptographic protocols. This paradigm is used in a setting when two parties $\mathcal{A}$ and $\mathcal{B}$ interact with each other and one party (say $\mathcal{B}$) has a secret which he wants to keep private. To ensure that $\mathcal{B}$'s secret is kept private during the execution of the protocol, we simulate the entire protocol with $\mathcal{A}$. This is done such that the simulator has no or very controlled access to $\mathcal{B}$'s input. If we are able to simulate a *view* for $\mathcal{A}$ that is indistinguishable from the one created during an actual interaction, it implies that the steps of the protocol do not provide any knowledge of $\mathcal{B}$'s secret to $\mathcal{A}$, beyond what she can discover on her own by looking at her output. A crucial point is that we do not want $\mathcal{A}$ to gain knowledge even if she arbitrarily deviates from the protocol when interacting with $\mathcal{B}$. This approach is reminiscent of Goldwasser, Micali and Rackoff's definition of *zero-knowledge* [16]. In the case of *zero-knowledge* proofs, one problem that a simulator faces is that it is impossible for it to generate a convincing proof without having access to $\mathcal{B}$'s input. The question that arises is how can the simulator generate a *view* that is *indistinguishable* from that of an actual interaction with $\mathcal{B}$? The answer is, the simulator has two advantages over the $\mathcal{B}$ which compensate for the serious disadvantage of not knowing the inputs. The first advantage is that the simulator has access to $\mathcal{A}$'s random tape. This means that it can actually determine her next question. The second advantage is, unlike in the actual interaction, the simulator

has many attempts to answer the questions. During an interaction if it fails, it can try again and output only the attempt in which it succeeds. This is in contrast to an actual interaction where if the party fails even once to answer the question, the proof is rejected. This technique is called *rewinding* because when the simulator fails to answer a question posed by the verifier, it simply rewinds the verifier back to the last successful state and tries again.

In cases where $\mathcal{A}$ gets some output related to $\mathcal{B}$'s input, we extend the definition of the *simulator* and allow it to access an oracle running the protocol. The oracle has access to $\mathcal{B}$'s input and behaves according to the legitimate steps of the protocol. The simulator runs in two stages. In the first stage, it determines the *effective* input of $\mathcal{A}$ used in the protocol. This allows us to take into account the fact that $\widetilde{\mathcal{A}}$ on input α can choose to do some local computation and use the results (effective input $\tilde{\alpha}$) in the protocol. In the second stage, the simulator acting as $\mathcal{A}$ at an appropriate step invokes the oracle with $\tilde{\alpha}$. It gets the output from the oracle and then provides it to $\mathcal{A}$.

Note that failure to provide a simulation of a protocol does not necessarily mean that this interaction results in some gain in illegal information. What matters is that any real gain cannot occur whenever we are able to present a simulation.

2.4 Protocol and Protocol Specifications

A *protocol* is a multi-party synchronous program that describes the computations to be performed by each party, and the messages to be sent to some other

party at every point in time. The protocol terminates when no party has any message to send or information to compute. A set of *protocol specifications* not only defines what a protocol must achieve at the end of a successful run, but also provides bounds on the dishonest behavior of the participants.

Definition 2.1 (Protocol) *Let $(\mathcal{A}, \mathcal{B})$ be a CPP. A protocol is a pair of programs $\wp = (\wp_a, \wp_b)$ where $\wp_a$ and $\wp_b$ are probabilistic programs run by $\mathcal{A}$ and $\mathcal{B}$ respectively.*

Let α and β be $\mathcal{A}$'s and $\mathcal{B}$'s private inputs respectively and let κ be their common input. The output after the execution of the protocol is a pair of strings (γ, δ), where γ and δ are the contents of the output tapes of $\mathcal{A}$ and $\mathcal{B}$ respectively. We can view the protocol as the description of a process that, for any fixed contents of the random tape, transforms a triplet of inputs (α, β, κ) into a pair of outputs (γ, δ). As the contents of the random tapes of $\mathcal{A}$ and $\mathcal{B}$ are uniformly distributed, for all inputs $\alpha, \beta, \kappa \in \Sigma^*$, a protocol $\wp$ specifies a probability distribution over $\Sigma^* \times \Sigma^*$ (the output domain) namely $\wp(\alpha, \beta, \kappa)$. These probabilities are taken over all possible contents of the random tapes.

Definition 2.2 (Protocol Specifications) *Let $\mathcal{D} = \{D(\alpha, \beta, \kappa)\}_{\alpha,\beta,\kappa \in \Sigma^*}$ be a family of probability distributions over $\Sigma^* \times \Sigma^*$. We call $\mathcal{D}$ a protocol specification. $\mathcal{D}_a(\alpha, \beta, \kappa)$ and $\mathcal{D}_b(\alpha, \beta, \kappa)$ are marginal distributions that are obtained by restricting $\mathcal{D}(\alpha, \beta, \kappa)$ to $\mathcal{A}$ or $\mathcal{B}$ respectively.*

$\mathcal{D}_a(\alpha, \beta, \kappa)$ and $\mathcal{D}_b(\alpha, \beta, \kappa)$ correspond to the output specifications of $\mathcal{A}$ and $\mathcal{B}$ respectively after a successful run of the protocol. We say that a protocol *implements* specifications $\mathcal{D}$ if $\forall \alpha, \beta, \kappa \in \Sigma^*$

$$[\wp(\alpha, \beta, \kappa) = \mathcal{D}(\alpha, \beta, \kappa)]$$

In general given a protocol specification, our task is to construct a protocol to implement it.

2.5 Correct and Private Protocols

Two-party protocol security specifications consist of two fundamental notions of security: *correctness* and *privacy*. By *correctness* we mean two things: Firstly, when the parties are acting honestly, the protocol must accomplish the task it was designed for. Secondly, a dishonest party must not be able to induce an illegitimate output distribution on honest participants[1] . The notion of *privacy* ensures that the protocol does not give *any information* about the inputs other than what each output inherently reveals. These security specifications were first presented in [8] and have been extended to include dishonest behavior of the participants.

Two-party protocols can be categorized as being *symmetric* or *asymmetric*. In *symmetric* protocols both parties get an output at the end, while in *asymmetric* protocols only one party receives an output. In the following specifications, the equality sign (=) means that the distribution on the left-hand side and the right-hand

[1] An illegitimate output distribution is any output distribution which does not conform to the protocol specifications.

side are indistinguishable.[2] Assume without any loss of generality that $\mathcal{A}$ does not receive any output in the *asymmetric* case.

2.5.1 Correctness

Definition 2.3 (Protocol Correctness) A Protocol is ***correct*** if it is both $\mathcal{A}$***-correct*** and $\mathcal{B}$***-correct***

- $\mathcal{A}$***-Correctness***
  - $\forall \alpha, \beta, \kappa$

$$[\overline{\wp_a}, \overline{\wp_b}]_{\mathcal{A}}(\alpha, \beta, \kappa) = \mathcal{D}_{\mathcal{A}}(\alpha, \beta, \kappa) \tag{2.1}$$

  - $\forall \widetilde{\wp_a}, \alpha, \beta, \kappa, \ \exists \ \widetilde{\wp_a}' \ s.t$

$$\left([\widetilde{\wp_a}, \overline{\wp_b}]_{\mathcal{B}}(\alpha, \beta, \kappa), \mathcal{B} \ accepts\right) = \left(\mathcal{D}_{\mathcal{B}}(\widetilde{\wp_a}'(\alpha, \kappa), \beta, \kappa), \mathcal{B} \ accepts\right) \tag{2.2}$$

- $\mathcal{B}$***-Correctness***
  - $\forall \alpha, \beta, \kappa$

$$[\overline{\wp_a}, \overline{\wp_b}]_{\mathcal{B}}(\alpha, \beta, \kappa) = \mathcal{D}_{\mathcal{B}}(\alpha, \beta, \kappa) \tag{2.3}$$

  - $\forall \widetilde{\wp_b}, \ \alpha, \beta, \kappa, \ \exists \widetilde{\wp_b}' \ s.t$

$$\left([\overline{\wp_a}, \widetilde{\wp_b}]_{\mathcal{A}}(\alpha, \beta, \kappa), \mathcal{A} \ accepts\right) = \left(\mathcal{D}_{\mathcal{A}}(\alpha, \widetilde{\wp_b}'(\beta, \kappa), \kappa), \mathcal{A} \ accepts\right) \tag{2.4}$$

[2] Discussion about various forms of indistinguishability will be presented later in this chapter.

Condition (2.1) defines what the protocol must achieve when both parties are acting honestly. $\mathcal{A}$ must get the output it is entitled to in accordance with the protocol specifications. Condition (2.2) defines the bounds on the dishonest behavior of $\mathcal{A}$. Any distribution that $\widetilde{\mathcal{A}}$ may try to induce on $\overline{\mathcal{B}}$'s view during the execution of the protocol, either $\overline{\mathcal{B}}$ rejects, or there must exist a valid input which is efficiently computable (effective input) that $\widetilde{\mathcal{A}}$ can use in an otherwise honest execution of the protocol to achieve that same results. This is an equivalent way of saying that her dishonest behavior is restricted to doing efficient local computations on her input bits and then using the results of these computations in the protocol and acting honestly.

We define $\mathcal{B}$-Correctness specifications analogously. $\mathcal{B}$ must get the output he is entitled to in accordance with the protocol specifications (equation (2.3)). Similarly for any output distribution that $\widetilde{\mathcal{B}}$ may try to induce on $\overline{\mathcal{A}}$'s view, either $\overline{\mathcal{A}}$ rejects, or there must exist a valid input that is efficiently computable (effective input) that $\widetilde{\mathcal{B}}$ can use in an otherwise honest execution of the protocol to achieve that same result.

2.5.2 Privacy

We demonstrate that a protocol is private by showing that a party can *simulate* its own view of the protocol having no or controlled access[3] to the other party's inputs.

Let $View_{\mathcal{A}} \stackrel{def}{=} [\widetilde{\wp_a}, \overline{\wp_b}]^*_{\mathcal{A}}(\alpha, \beta, \kappa)$ and $View_{\mathcal{B}} \stackrel{def}{=} [\overline{\wp_a}, \widetilde{\wp_b}]^*_{\mathcal{B}}(\alpha, \beta, \kappa)$. Let $\mathcal{M}$ denote the simulator. We now provide individual definitions of privacy for both parties for asymmetric and symmetric cases.

Symmetric Case

- **$\mathcal{A}$-Privacy**

  $\exists \mathcal{M}\ \forall \widetilde{\wp_a}, \alpha, \beta, \kappa$

  $$\mathcal{M}^{\wp(*,\beta,\kappa)}(\widetilde{\wp_a}, \alpha, \kappa) = View_{\mathcal{A}} \tag{2.5}$$

- **$\mathcal{B}$-Privacy**

  $\exists \mathcal{M}\ \forall \widetilde{\wp_b}, \alpha, \beta, \kappa$

  $$\mathcal{M}^{\wp(\alpha,*,\kappa)}(\widetilde{\wp_b}, \beta, \kappa) = View_{\mathcal{B}} \tag{2.6}$$

For the symmetric case, a simulator $\mathcal{M}$ interacting with either participant ($\widetilde{\wp_a}$ or $\widetilde{\wp_b}$) must be able to create a view that is indistinguishable from that of an actual instance of the protocol. As both parties get an output at the end of the protocol, we allow $\mathcal{M}$ to access an outside oracle running the protocol.

[3] In cases where a party gets an output at the end of the protocol, the simulator must make an oracle call to get that output. The oracle has access to other party's inputs.

Asymmetric Case

- $\mathcal{A}$**-Privacy**

  $\exists \mathcal{M}\ \forall \widetilde{\wp_a}, \alpha, \beta, \kappa$

$$\mathcal{M}(\widetilde{\wp_a}, \alpha, \kappa) = View_{\mathcal{A}} \tag{2.7}$$

- $\mathcal{B}$**-Privacy**

  $\exists \mathcal{M}\ \forall \widetilde{\wp_b}, \alpha, \beta, \kappa$

$$\mathcal{M}^{\wp(\alpha,*,\kappa)}(\widetilde{\wp_b}, \beta, \kappa) = View_{\mathcal{B}} \tag{2.8}$$

Equation (2.7) defines the bounds on the behavior of any program $\widetilde{\wp_a}$ run by $\widetilde{\mathcal{A}}$. For any $\widetilde{\wp_a}$ interacting with $\overline{\mathcal{B}}$, a simulator $\mathcal{M}$ interacting with $\widetilde{\wp_a}$ must be able to create an indistinguishable *view* from that of an actual instance of the protocol.

Equation (2.8) defines the bounds on the behavior of $\widetilde{\mathcal{B}}$. As $\widetilde{\mathcal{B}}$ gets an output in the end, we provide $\mathcal{M}$ with access to an oracle. This oracle is running a copy of the protocol and has access to $\overline{\mathcal{A}}$'s inputs. $\mathcal{M}$ interacting with $\widetilde{\wp_b}$ must be able to create an indistinguishable view from that of an actual instance of the protocol.

2.6 Reduction Among Protocols

In order to formalize how our parties can use existing protocols as *subroutines* we modify our notion of CPP to allow execution of *abstract protocols*. In addition to the description of CPP given before, each party has an extra read/write protocol tape. These tapes are to be used in connection with a special purpose device known as *abstract protocol*. The abstract protocol is a trusted device that implements a protocol specification. At each step of its computation, a machine can read or write a single character on the protocol tape. The parties would first

write the input on their protocol tape, then request the execution of the abstract protocol, and then finally extract their output from the tape. After execution, the abstract protocol loses its inputs and both protocol tapes contain only the output or result of the execution.

We say that a protocol $\wp$ reduces to $\hat{\wp}$ if $\wp$ uses a set $\hat{\wp}$ of abstract protocols.

2.7 Mathematical Tools

2.7.1 Notation

Let U be a set. The expression $u \in_R U$ means that an element is chosen from set U randomly i.e each $u \in U$ has an equal probability of being chosen.

2.7.2 Entropy

The *entropy* of a random variable X is defined as

$$\mathbf{H}(X) = \sum_{x \in X} p(x) \log \left(\frac{1}{p(x)} \right)$$

where $p(x)$ is the probability that X is in state x.

2.7.3 Conditional Entropy

The *conditional entropy* of a pair of discrete random variables (X, Y) with joint distribution $p(x, y)$ is defined as

$$\mathbf{H}(Y \mid X) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(y|x)$$

This can also be written in the following equivalent way

$$\begin{aligned}\mathbf{H}(Y \mid X) &= -\sum_{x \in X} p(x) \sum_{y \in Y} p(y|x) \log(p(y|x) \\ &= -\sum_{x \in X} p(x) H(Y|X = x)\end{aligned}$$

2.7.4 Joint Entropy

The *joint entropy* of a pair of discrete random variables (X, Y) with joint distribution $p(x, y)$ is defined as

$$\mathbf{H}(X, Y) = -\sum_{x \in X} \sum_{y \in Y} p(x, y) \log p(x, y)$$

Joint entropy can be expressed in terms of conditional entropy as

$$\mathbf{H}(X, Y) = \mathbf{H}(X) + \mathbf{H}(Y \mid X)$$

2.7.5 Mutual Information

Intuitively *mutual information* is a measure of how much information one random variable contains about another. The mutual information between two variables X and Y denoted by $I(X; Y)$ is given by

$$\mathbf{I}(X; Y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

The mutual information can be expressed in terms of conditional entropy as

$$\mathbf{I}(X; Y) = \mathbf{H}(X) - \mathbf{H}(X \mid Y)$$

The mutual information between two random variables X and Y conditioned upon a third random variable Z is given by

$$\mathbf{I}(X;Y|Z) = \mathbf{H}(X \mid Z) - \mathbf{H}(X \mid Y, Z)$$

2.7.6 Probability Ensembles

For $w \in \{0,1\}^*$, let X be a *probability distribution* ranging over binary strings w. Let I be a countable index set. A *probability ensemble* $X = \{X_i\}_{i \in I}$ is a sequence of random variables each ranging over all values of w, and indexed by I. (Sometimes we omit $i \in I$ from the notation.)

2.8 Indistinguishability

Let $X = \{X_n\}_{n \in \mathbb{N}}$ and $Y = \{Y_n\}_{n \in \mathbb{N}}$ be two probability ensembles. Consider the following framework. A random sample is selected from either X or Y and is handed to a judge. After studying the sample, the judge makes a decision about the origin of the sample. We say that X becomes *replaceable* by Y when the verdict of any judge becomes uncorrelated with the distribution from which the sample came. There are two relevant parameters in this framework: the *size* of the sample and the amount of *time* the judge is given to produce his verdict. By bounding these two parameters in different ways we obtain different notions of *indistinguishability*.

2.8.1 Equality

We say that $X = \{X_n\}$ and $Y = \{Y_n\}$ are *indistinguishable* or equal if for any judge, the judge's verdict is independent from the origin of the sample even if he

is given samples of arbitrary size and he can study them for an arbitrary length of time.

Formally, two ensembles X and Y are said to be *perfectly indistinguishable* or *equal* if $\forall \alpha, n$

$$\mathbf{Pr}(X_n = \alpha) = \mathbf{Pr}(Y_n = \alpha)$$

2.8.2 Statistical Indistinguishability

We define the two ensembles to be *statistically indistinguishable* if they are replaceable by each other, with respect to a judge that may have unbounded computational power but is given polynomial size samples to work on.

Formally, two ensembles $X = \{X_n\}$ and $Y = \{Y_n\}$ are said to be *statistically indistinguishable* if for every positive polynomial $p(.)$, $\exists N_p, \forall n > N_p$

$$\sum_{\alpha} |\mathbf{Pr}(X_n = \alpha) - \mathbf{Pr}(Y_n = \alpha)| < \frac{1}{p(n)}$$

2.8.3 Polynomial-Time Indistinguishability

Two ensembles $X = \{X_n\}_{n \in \mathbb{N}}$ and $Y = \{Y_n\}_{n \in \mathbb{N}}$ are *indistinguishable in polynomial time* if for every probabilistic polynomial-time algorithm D, every positive polynomial $p(.)$, and all sufficiently large n,

$$|\mathbf{Pr}(D(X_n, 1^n) = 1) - \mathbf{Pr}(D(Y_n, 1^n) = 1)| < \frac{1}{p(n)}$$

The term *computational indistinguishability* is also used to represent indistinguishability in polynomial time.

CHAPTER 3
CURRENT $\binom{2}{1}$–OT SECURITY SPECIFICATIONS

We present the current security specifications of $\binom{2}{1}$–OT based on privacy and correctness.

3.1 $\binom{2}{1}$–OT Specifications

$\binom{2}{1}$–OT is a cryptographic protocol between two parties, a sender $\mathcal{A}$ and a receiver $\mathcal{B}$. This protocol enables $\mathcal{A}$ to transfer one of her two input bits (b_0, b_1) to $\mathcal{B}$ who secretly chooses which bit (b_c) to get. The choice of $\mathcal{B}$ is represented by c. This is done in such a way that $\mathcal{A}$ does not gain any knowledge about c, while $\mathcal{B}$ gets only one of $\mathcal{A}$'s input bits. We use the $\binom{2}{1}$–OT security specifications first introduced by Brassard, Crépeau, and Wolf [2, 4].

Let (B_0, B_1) and C be random variables (RV) representing the inputs of $\mathcal{A}$ and $\mathcal{B}$ respectively. Let C' be a random variable representing $\widetilde{\mathcal{B}}$'s effective input. We assume that both $\mathcal{A}$ and $\mathcal{B}$ are aware of the arbitrary joint probability distribution of these random variables $P_{B_0,B_1,C}$. A sample (b_0, b_1, c) is generated from the distribution and (b_0, b_1) is provided as $\mathcal{A}$'s secret input while c is provided as $\mathcal{B}$'s secret input. As mentioned previously, the equality sign(=) in the following specifications means that the distributions on the left-hand side and the right-hand side are indistinguishable.

3.1.1 $\binom{2}{1}$–OT Correctness

Definition 3.1 (OT-Correctness) Protocol $[\mathcal{A}, \mathcal{B}]$ is *correct* for $\binom{2}{1}$–OT if

- $\forall b_0, b_1 \in \mathcal{F}_2, c \in \mathcal{F}_2$,

$$\left[\overline{\mathcal{A}}, \overline{\mathcal{B}}\right](b_0, b_1)(c) = (\varepsilon, b_c) \tag{3.1}$$

- $\forall \widetilde{\mathcal{A}},\ \exists \widetilde{\mathcal{A}}'$ s.t $\forall b_0, b_1 \in \mathcal{F}_2, c \in \mathcal{F}_2$,

$$\left([\widetilde{\mathcal{A}}, \overline{\mathcal{B}}]_{\mathcal{B}}(b_0, b_1)(c), \mathcal{B}\ accepts\right) = \left([\overline{\mathcal{A}}, \overline{\mathcal{B}}]_{\mathcal{B}}(\widetilde{\mathcal{A}}'(b_0, b_1))(c)\right) \tag{3.2}$$

Equation (3.1) defines what a $\binom{2}{1}$–OT protocol must achieve when both parties are acting honestly. $\mathcal{B}$ should only get the bit corresponding to his choice c (i.e b_c), and $\mathcal{A}$ does not get any output from the protocol. We use ε to denote that $\mathcal{A}$'s output is an empty string.

Equation (3.2) defines $\mathcal{A}$-*Correctness*. It defines the bounds on the dishonest behavior of $\mathcal{A}$. Any distribution that $\widetilde{\mathcal{A}}$ may try to induce on $\overline{\mathcal{B}}$'s output, either $\overline{\mathcal{B}}$ is given the knowledge that $\widetilde{\mathcal{A}}$ is cheating and he aborts, or there must exist a pair of input bits (b'_0, b'_1) efficiently computable from (b_0, b_1) that $\widetilde{\mathcal{A}}$ can use in the protocol while otherwise acting honestly, to achieve that same result. This is an equivalent way of saying that her dishonest behavior is restricted to doing local computation on her input bits and then using the results of these computations (her effective input) and acting honestly. (b'_0, b'_1) denote $\widetilde{\mathcal{A}}$'s effective input i.e. $(b'_0, b'_1) \leftarrow \widetilde{\mathcal{A}}'(b_0, b_1)$.

3.1.2 $\binom{2}{1}$–OT Privacy

Definition 3.2 (OT-Privacy) Protocol $[\mathcal{A}, \mathcal{B}]$ is *private* for $\binom{2}{1}$–OT if

$\forall (B_0, B_1) \in RV(\mathcal{F}_2)^2, C \in RV(\mathcal{F}_2)$

- $\forall b_0, b_1 \in \mathcal{F}_2, \forall \widetilde{\mathcal{A}}$,

$$\mathbf{I}\Big(C; [\widetilde{\mathcal{A}}, \overline{\mathcal{B}}]^*_{\mathcal{A}}\ (B_0, B_1)(C) \mid (B_0, B_1) = (b_0, b_1)\Big) = 0 \tag{3.3}$$

- $\forall c \in \mathcal{F}_2,\ \forall \widetilde{\mathcal{B}},\ \exists \widetilde{C} \in RV(\mathcal{F}_2)$ s.t

$$\mathbf{I}\Big(B_{\neg \widetilde{C}}; [\overline{\mathcal{A}}, \widetilde{\mathcal{B}}]^*_{\mathcal{B}}(B_0, B_1)(C) \mid (B_{\widetilde{C}}, C = c)\Big) = 0 \tag{3.4}$$

These two *privacy* constraints guarantee that each party is limited to the information he or she should get at the end of a successful run of the protocol.

Equation (3.3) defines $\mathcal{A}$*-privacy*. $\widetilde{\mathcal{A}}$'s *view* must not provide any knowledge of $\mathcal{B}$'s input bit c.

Equation (3.4) defines $\mathcal{B}$*-privacy*. $\widetilde{\mathcal{B}}$ may acquire complete knowledge of either b_0 or b_1 from the protocol but not both. This includes information about any joint function these two input bits. Note that we cannot make any assumptions that $\widetilde{\mathcal{B}}$ will use the provided input bit c in the protocol. $\widetilde{\mathcal{B}}$ may do some local computation on his input bit and use the result (effective input c') as his input. Similarly, during the execution of the protocol, he may try to gain knowledge of both bits by trying to get any joint information about input bits of $\overline{\mathcal{A}}$ Due to these reasons, these constraints are conditioned on providing *one* of $\mathcal{A}$'s input bits to $\widetilde{\mathcal{B}}$. We do not require that $\widetilde{\mathcal{B}}$ be given b_c because there is no way to prevent him from obtaining any other $b_{c'}$ through otherwise honest use of the protocol.

3.2 Motivation for new $\binom{2}{1}$–OT Specifications

The security of multi-party computation has been traditionally defined in terms of *privacy* and *correctness*. In recent literature, various attempts have been

made [20, 1, 5, 13] to merge these two fundamental properties into a unified definition of security. Micali [20] argued that though privacy and correctness are the fundamental aspect of secure computation, the logical connective *"and"* does not combine them adequately to provide complete protocol security, and capturing in the most general sense what simultaneously enforcing them means is quite difficult. To obtain a satisfactory notion of security, privacy and correctness should not be handled independently but need to be *blended* in a proper manner. In certain cases correctness and privacy may seem to be conflicting requirements. The negative effect of this definitional approach is thus considering secure some protocols that may not be such. In a secure protocol, privacy is taken to mean that a protocol admits a certain type of *simulator*, and correctness is a concept defined through the same simulator proving the protocol private.

The main motivation of this research was to investigate and present a new and more robust set of security specifications for $\binom{2}{1}$–OT in which privacy and correctness are defined concurrently and treated as interlinked security constraints. We present new security specifications of $\binom{2}{1}$–OT in chapter 4.

3.3 Conclusion

We have presented a set of current security specifications for $\binom{2}{1}$–OT which deal with privacy and correctness as separate and disjoint constraints. In the next chapter we present a new set of security specifications for $\binom{2}{1}$–OT.

CHAPTER 4
NEW $\binom{2}{1}$–OT SECURITY SPECIFICATIONS

We present a new set of security specifications for $\binom{2}{1}$–OT in this chapter.

4.1 $\binom{2}{1}$–OT Specifications

We start by looking at what a protocol implementing $\binom{2}{1}$–OT must achieve. In accordance with the naming conventions used in the previous chapters, assume that $\mathcal{A}$ is the sender and $\mathcal{B}$ is the receiver.

Definition 4.1 ($\binom{2}{1}$–OT) Protocol $[\mathcal{A}, \mathcal{B}]$ *implements* $\binom{2}{1}$–OT if
$\forall b_0, b_1, c \in \mathcal{F}_2$

$$[\overline{\mathcal{A}}, \overline{\mathcal{B}}](b_0, b_1)(c) = (\varepsilon, b_c) \tag{4.1}$$

In a $\binom{2}{1}$–OT protocol, $\mathcal{A}$ does not get any output, and the protocol must not provide any information about $\mathcal{B}$'s input to $\mathcal{A}$ (denoted by ε). $\mathcal{B}$ must get the bit of his choice c. The following definition was presented as one of the *correctness* constraints (see section 3.1.1), and it helped us conceptualize what $\binom{2}{1}$–OT security specifications must achieve. This equation defines the input/output relation in an ideal execution of $\binom{2}{1}$–OT protocol and provides bounds for the behavior of both parties.

4.2 New $\binom{2}{1}$–OT Security Specifications

Let (B_0, B_1) and C be the random variables taking values over $\mathcal{F}_2$ that describe the inputs of $\mathcal{A}$ and $\mathcal{B}$ respectively. Let $\mathcal{T}$ (transcript) denote the random variable that describes the view of the protocol.

$$\mathcal{T} = [\mathcal{A}, \mathcal{B}]^*(B_0, B_1)(C)$$

$\mathcal{T}$ contains all details including the messages received, the result of any local random sampling done and the internal coin tosses of both parties.

Let $\mathcal{T}_{\mathcal{A}}$ and $\mathcal{T}_{\mathcal{B}}$ denote the *view* restricted to $\mathcal{A}$ and $\mathcal{B}$ respectively. Let $\overline{\mathcal{T}}$ and $\widetilde{\mathcal{T}}$ denote the transcripts produced by honest and dishonest parties respectively.

Let X be a discrete random variable. The function $f_X(x) = \mathbf{Pr}[X = x]$ is called the *discrete probability function* of X. Discrete probability function gives the probability that a discrete random variable is exactly equal to some value. Given two random variables X and Y taking values over the same sample space S and having discrete probability functions $f_X(x)$ and $f_Y(x)$ respectively, we use the definition of L_1 *norm* to denote the difference of their discrete probability functions as follows.

$$\mathbf{L_1}(\mathbf{X}; \mathbf{Y}) = \frac{1}{2} \times \sum_{x \in S} |f_X(x) - f_Y(x)|$$

4.2.1 Sender Specifications

Equation (4.1) defines what information the two parties are restricted to after a successful run of the $\binom{2}{1}$–OT protocol. $\mathcal{A}$ does not get any information about $\mathcal{B}$'s

input bit, while $\mathcal{B}$ gets one of $\mathcal{A}$'s input bits of his choosing. We look at what a $\widetilde{\mathcal{A}}$ may try to gain from the protocol.

- *$\mathcal{A}$-Privacy:* $\widetilde{\mathcal{A}}$ may try to gain knowledge of $\overline{\mathcal{B}}$'s input bit c from the protocol. Our security specifications must ensure that $\widetilde{\mathcal{A}}$ does not learn about c. As $\mathcal{T}$ is the view of the protocol and encapsulate all possible behaviors, $\widetilde{\mathcal{T}}_{\mathcal{A}}$ must not provide any information about c. This condition must hold for all scenarios including an abort of the protocol by $\widetilde{\mathcal{A}}$.
- *$\mathcal{A}$-Correctness:* Ideally we would like $\widetilde{\mathcal{A}}$ to use her provided input bits during the protocol. As this would be unenforceable in practice because $\widetilde{\mathcal{A}}$ can precompute another set of inputs and then act honestly without raising suspicion, $\widetilde{\mathcal{A}}$ is allowed to choose from the four possible input pairs with a certain probability distribution. For any distribution that $\widetilde{\mathcal{A}}$ can induce on $\overline{\mathcal{B}}$'s transcript, there must exist (B'_0, B'_1) representing $\mathcal{A}$'s input pair (b'_0, b'_1) which she can use and act honestly to achieve the same distribution on $\mathcal{T}_{\mathcal{B}}$. i.e for any $\widetilde{\mathcal{A}}$ there must exist an associated probability distribution on her new input pair such that she can pick an input pair according to this distribution and act honestly in the protocol to achieve the same results.

  Note that $\widetilde{\mathcal{A}}$'s dishonest behavior is dependent upon the contents of her random tape, the messages received from $\overline{\mathcal{B}}$, what she sees as the output of the intermediate steps of the protocol, and her own bias towards her input pair. Our definition is therefore conditioned on the existence of a function f which takes an instance of $\widetilde{\mathcal{T}}_{\mathcal{A}}$ as input and outputs a pair of output bits (b'_0, b'_1). If f is run on all the possible outcome instances of $\widetilde{\mathcal{T}}_{\mathcal{A}}$, and is able to output

a bit pair (b'_0, b'_1) each time, we get a specific probability distribution on the four possible input pairs of $\widetilde{\mathcal{A}}$. This function f validates any such dishonest behavior as acceptable or unacceptable for the protocol, and provides bounds on $\widetilde{\mathcal{A}}$ by mapping her dishonest behavior to a specific probability distribution on her input pair (b'_0, b'_1).

This condition must hold for all scenarios when the protocol finishes successfully and $\overline{\mathcal{B}}$ does not reject. Note that we do not require this function to be efficiently computable, and mere existence of such function is sufficient for the definition to hold.

We now formalize these two constraints.

Definition 4.2 ($\binom{2}{1}$–OT $\mathcal{A}$-Security) Protocol $[\mathcal{A}, \mathcal{B}]$ is $\mathcal{A}$-*secure* for $\binom{2}{1}$–OT if

$\forall C \in RV(\mathcal{F}_2),\ b_0, b_1 \in \mathcal{F}_2,$ adv. $\widetilde{\mathcal{A}}, \exists$ function f,

Let $\widetilde{\mathcal{T}} = \left[\widetilde{\mathcal{A}}, \overline{\mathcal{B}}\right]^*(b_0, b_1)(C)$

$(B'_0, B'_1) = f(\widetilde{\mathcal{T}}_{\mathcal{A}})$

Let $\overline{\mathcal{T}} = \left[\overline{\mathcal{A}}, \overline{\mathcal{B}}\right]^*(B'_0, B'_1)(C)$

$$\max \left\{ \begin{array}{l} \mathbf{I}\left(C; \widetilde{\mathcal{T}}_{\mathcal{A}}\right), \\ \mathbf{Pr}(\overline{\mathcal{B}} \text{ accepts}) \times \mathbf{L_1}\left(\overline{\mathcal{T}}_{\mathcal{B}}; \widetilde{\mathcal{T}}_{\mathcal{B}}\right) \end{array} \right\} = \mathbf{0}$$

4.2.2 Receiver Specifications

We look at what $\widetilde{\mathcal{B}}$ may try to gain from the protocol.

- $\mathcal{B}$-*Privacy:* Equation (4.1) enforces that $\mathcal{B}$ should only get the input bit corresponding to his provided input c and nothing else. We relax this restriction

and allow $\widetilde{\mathcal{B}}$ to get any one of the input bits of $\overline{\mathcal{A}}$. The reason is that we cannot assume that $\widetilde{\mathcal{B}}$ will use the provided input bit c in the protocol. $\widetilde{\mathcal{B}}$ may do some local computation on c, and use the result (effective input c') in the protocol. The security specifications must ensure that when $\widetilde{\mathcal{B}}$ is given $b_{c'}$, $\widetilde{\mathcal{T}}_{\mathcal{B}}$ must not provide any additional information about $\overline{\mathcal{A}}$'s other input bit $b_{\neg c'}$. This includes any joint information about the two input bits of $\overline{\mathcal{A}}$.

- *$\mathcal{B}$-Correctness:* Ideally we would like $\widetilde{\mathcal{B}}$ to use the provided input c in the $\binom{2}{1}$–OT protocol. As this would be enforceable because $\widetilde{\mathcal{B}}$ can pre-compute another input c' (effective input) and act honestly in the protocol. $\widetilde{\mathcal{B}}$ is allowed to choose from the two possible input choices with a certain probability distribution. For any distribution that $\widetilde{\mathcal{B}}$ can induce on $\overline{\mathcal{A}}$'s transcript $\mathcal{T}_{\mathcal{A}}$, there must exist C' representing $\mathcal{B}$'s input bit c' which he can use and act honestly to achieve the same distribution on $\mathcal{T}_{\mathcal{A}}$. i.e for any $\widetilde{\mathcal{B}}$, there must exist an associated probability distribution on his new input bit such that he can pick an input according to this distribution and act honestly in the protocol to achieve the same results.

  Similar to $\mathcal{A}$-correctness, our specifications for $\mathcal{B}$-correctness are conditioned on the existence of a function f which takes an instance of $\widetilde{\mathcal{T}}_{\mathcal{B}}$ as input and outputs a bit c'. If f is run on all the possible outcome instances of $\widetilde{\mathcal{T}}_{\mathcal{B}}$, and is able to output a bit c' each time, we get a specific probability distribution on the two possible input bits of $\widetilde{\mathcal{B}}$. This function f validates any such dishonest behavior as acceptable or unacceptable for the protocol, and provides

bounds on $\widetilde{\mathcal{B}}$ by mapping his dishonest behavior to a specific probability distribution on his input c'.

This condition must hold for all scenarios when the protocol finishes successfully and $\overline{\mathcal{A}}$ does not reject. Note that we do not require this function to be efficiently computable, and mere existence of such function is sufficient for the definition to hold.

We now formalize $\mathcal{B}$'s security specifications.

Definition 4.3 ($\binom{2}{1}$–OT $\mathcal{B}$-Security) Protocol $[\mathcal{A}, \mathcal{B}]$ is *$\mathcal{B}$-secure* for $\binom{2}{1}$–OT if

$\forall (B_0, B_1) \in RV(\mathcal{F}_2)^2,\ c \in \mathcal{F}_2,$ adv. $\widetilde{\mathcal{B}}, \exists$ function f,

Let $\widetilde{\mathcal{T}} = \left[\overline{\mathcal{A}}, \widetilde{\mathcal{B}}\right]^*(B_0, B_1)(c)$,

$C' = f(\widetilde{\mathcal{T}}_{\mathcal{B}})$ and $\mathbf{I}(C'; B_0, B_1 \mid c) = 0$

Let $\overline{\mathcal{T}} = \left[\overline{\mathcal{A}}, \overline{\mathcal{B}}\right]^*(B_0, B_1)(C')$

$$\mathbf{max}\left\{\begin{array}{l}\mathbf{I}\Big(B_{\neg C'}; \widetilde{\mathcal{T}}_{\mathcal{B}} \;\Big|\; C', B_{C'}\Big) \\ \mathbf{Pr}\big(\overline{\mathcal{A}} \text{ accepts}\big) \times \mathbf{L_1}\left(\overline{\mathcal{T}}_{\mathcal{A}}; \widetilde{\mathcal{T}}_{\mathcal{A}}\right)\end{array}\right\} = \mathbf{0}$$

Combining both sender and receiver security specifications, we define $\binom{2}{1}$–OT security as follows.

Definition 4.4 ($\binom{2}{1}$–OT Security) Protocol $[\mathcal{A}, \mathcal{B}]$ is *secure* for $\binom{2}{1}$–OT if it is both *$\mathcal{A}$-Secure* and *$\mathcal{B}$-Secure*.

4.3 Function f is Computable

Our definition uses the notion of a global view of the protocol $\mathcal{T}$, and enforces security by defining privacy and correctness on the marginal views created by both parties. For $\mathcal{A}$-correctness, we require that f produce a pair (b'_0, b'_1) for every

instance $\tilde{\tau}_a$ of $\widetilde{\mathcal{T}}_{\mathcal{A}}$. However there is no reason to assume that each τ_a is compatible with only one such pair. If we are provided with a specific τ_a for which such a pair (b'_0, b'_1) exist, we can provide an algorithm $\mathcal{Z}$ which can find two bits $\hat{b}_0$ and $\hat{b}_1$ that are compatible with $\tilde{\tau}_a$. $\hat{b}_0$ and $\hat{b}_1$ are found independently of each other and have the same marginal probability distribution as b'_0 and b'_1 respectively. The reason is that if there exist a certain probability distribution P_1 according to which a pair (b'_0, b'_1) is selected, there exists another probability distribution P_2 such that bit $\hat{b}_0$ (corresponding to the first bit in the pair) and $\hat{b}_1$ (corresponding to the second bit in the pair) have the same marginal probability distributions. Note that as a pair these bits may not have the same probability distribution as (b'_0, b'_1). If $\mathcal{B}$-privacy is enforced, $\overline{\mathcal{B}}$ cannot differentiate if his transcript was created with $\widetilde{\mathcal{A}}$ selecting input from P_1 or P_2, and $\overline{\mathcal{B}}$'s transcript from his viewpoint would be indistinguishable. $\mathcal{Z}$ has access to the programs for $\widetilde{\mathcal{A}}$ and $\overline{\mathcal{B}}$, and is provided a specific transcript $\tilde{\tau}_a$ of $\widetilde{\mathcal{A}}$ as input.

***A*lgorithm 4.1 ($\mathcal{Z}(\tilde{\tau}_a)$)**

1. Extract the random tape from $\tilde{\tau}_a$ and set it as the random tape of $\widetilde{\mathcal{A}}$.
2. Set $\overline{\mathcal{B}}$'s input $c = 0$.
3. Select a random tape for $\overline{\mathcal{B}}$.
4. Let $\overline{\mathcal{B}}$ run an instance of $\binom{2}{1}$–OT with $\widetilde{\mathcal{A}}$. $\overline{\mathcal{B}}$ receives output b_0.
5. Compare $\widetilde{\mathcal{A}}$'s transcript with $\tilde{\tau}_a$. If both transcripts are identical, set $\hat{b}_0 = b_0$, and go to step 6. Else, rewind $\widetilde{\mathcal{A}}$ and go to step 2.
6. Set $\overline{\mathcal{B}}$'s input $c = 1$.

7. Select a random tape for $\overline{\mathcal{B}}$.
8. Let $\overline{\mathcal{B}}$ run an instance of $\binom{2}{1}$–OT with $\widetilde{\mathcal{A}}$. $\overline{\mathcal{B}}$ receives output b_1.
9. Compare $\widetilde{\mathcal{A}}$'s transcript with $\tilde{\tau}_a$. If both transcripts are identical, set $\hat{b}_1 = b_1$, and stop. Else, rewind $\widetilde{\mathcal{A}}$ and go to step 6.

4.4 Conclusion

We have presented a new set of security specifications for $\binom{2}{1}$–OT which enforces privacy and correctness concurrently. Unlike the previous specifications presented in section 3.1, the new specifications enforce correctness on the transcript created by the honest party. This is a more stronger and robust notion of security as it enforces correctness for not only the output obtained by the honest party, but also the output of any intermediate steps of the protocol. In the next chapter we provide a new set of security specifications for TOX-$\binom{2}{1}$.

CHAPTER 5
NEW TOX-$\binom{2}{1}$ SECURITY SPECIFICATIONS

We extend the new $\binom{2}{1}$–OT security specifications presented in chapter 4 to another variant of $\binom{2}{1}$–OT called TOX-$\binom{2}{1}$. We present a reduction of TOX-$\binom{2}{1}$ protocol to $\binom{2}{1}$–OT, and show that if we are provided with a privacy attacking algorithm to TOX-$\binom{2}{1}$, it can be efficiently reduced to attack $\binom{2}{1}$–OT privacy.

5.1 $\binom{2}{1}$-XOT and TOX-$\binom{2}{1}$

1-out-of-2 XOR Oblivious Transfer ($\binom{2}{1}$-XOT) is a variant of $\binom{2}{1}$–OT and is defined as follows: Sender Alice has two bits b_0, b_1 to offer and receiver Bob can choose one of b_0, b_1 or $b_\oplus$ where $b_\oplus = b_0 \oplus b_1$. At the end of the protocol Bob gets b_c, while Alice does not gain any knowledge of c.

1-out-of-2 Reverse XOR Oblivious Transfer (TOX-$\binom{2}{1}$) is $\binom{2}{1}$-XOT in the reverse direction. In this protocol, Bob acting as a sender offers two bits b_0, b_1 to Alice, and Alice can choose one of b_0, b_1 or $b_\oplus$ where $b_\oplus = b_0 \oplus b_1$. At the end of the protocol Alice gets b_c, while Bob on the other hand does not gain any knowledge about c.

5.2 TOX-$\binom{2}{1}$ Specifications

We start by looking at what a TOX-$\binom{2}{1}$ protocol should achieve at the end of a successful instance. Assume that $\mathcal{A}$ is the receiver and $\mathcal{B}$ is the sender.

Definition 5.1 (TOX-$\binom{2}{1}$) Protocol $[\mathcal{A}, \mathcal{B}]$ *implements* TOX-$\binom{2}{1}$ if

$\forall b_0, b_1 \in \mathcal{F}_2, c \in \{0, 1, \oplus\}$

$$[\overline{\mathcal{A}}, \overline{\mathcal{B}}](c)(b_0, b_1) = (b_c, \varepsilon) \tag{5.1}$$

The protocol must not provide any information about $\mathcal{A}$'s input bit c to $\mathcal{B}$(denoted by ε), and $\mathcal{A}$ must get the bit of her choice and nothing else i.e $b_c \in \{b_0, b_1, b_\oplus\}$. The protocol must not provide any additional information about $\mathcal{B}$'s input bits to $\mathcal{A}$ other than what she can infer from the output alone.

5.2.1 New TOX-$\binom{2}{1}$ Security Specifications

Let B_0, B_1 denote the random variables representing the input of sender $\mathcal{B}$. Let $B_\oplus = B_0 \oplus B_1$. Let C' denote the effective choice of $\mathcal{A}$, and let $B_{C'}$ denote the random variable representing the bit corresponding to $\mathcal{A}$'s choice. Let $B_{-C'}, B_{+C'}$ denote the random variables representing the other two choices.

Definition 5.2 (TOX-$\binom{2}{1}$-Security) Protocol $[\mathcal{A}, \mathcal{B}]$ is *secure* for TOX-$\binom{2}{1}$ if it is both $\mathcal{A}$-*secure* and $\mathcal{B}$-*secure*.

- ***$\mathcal{A}$-Security***

  Protocol $[\mathcal{A}, \mathcal{B}]$ is $\mathcal{A}$-*secure* for TOX-$\binom{2}{1}$ if

  $\forall (B_0, B_1) \in RV(\mathcal{F}_2)^2,\ c \in \{0, 1, \oplus\},$ adv. $\widetilde{\mathcal{A}}, \exists$ function f

  Let $\widetilde{\mathcal{T}} = \left[\widetilde{\mathcal{A}}, \overline{\mathcal{B}}\right]^*(c)(B_0, B_1)$

  $C' = f(\widetilde{\mathcal{T}}_{\mathcal{A}})$ and $\mathbf{I}(C'; B_0, B_1 \mid c) = 0$

  Let $\overline{\mathcal{T}} = \left[\widetilde{\mathcal{A}}, \overline{\mathcal{B}}\right]^*(C')(B_0, B_1)$

$$\mathbf{max} \left\{ \begin{array}{l} \mathbf{I}\Big(B_{-C'}; \widetilde{\mathcal{T}}_{\mathcal{A}} \;\Big|\; C', B_{C'}\Big) + \mathbf{I}\Big(B_{+C'}; \widetilde{\mathcal{T}}_{\mathcal{A}} \;\Big|\; C', B_{C'}\Big) \\ \mathbf{Pr}(\overline{\mathcal{B}} \text{ accepts}) \times \mathbf{L_1}\left(\overline{\mathcal{T}}_{\mathcal{B}}; \widetilde{\mathcal{T}}_{\mathcal{B}}\right) \end{array} \right\} = \mathbf{0} \tag{5.2}$$

- ***$\mathcal{B}$-Security***

  Protocol $[\mathcal{A}, \mathcal{B}]$ is *$\mathcal{B}$-secure* for TOX-$\binom{2}{1}$ if

  $\forall C \in RV(0, 1, \oplus), b_0, b_1 \in \mathcal{F}_2,$ adv. $\widetilde{\mathcal{B}}, \exists$ function f

  Let $\widetilde{\mathcal{T}} = \left[\overline{\mathcal{A}}, \widetilde{\mathcal{B}}\right]^*(C)(b_0, b_1)$

  $(B_0', B_1') = f(\widetilde{\mathcal{T}}_{\mathcal{B}})$

  Let $\overline{\mathcal{T}} = \left[\overline{\mathcal{A}}, \overline{\mathcal{B}}\right]^*(C)(B_0', B_1')$

$$\max \left\{ \begin{array}{l} \mathbf{I}\left(C; \widetilde{\mathcal{T}}_{\mathcal{B}}\right), \\ \mathbf{Pr}(\overline{\mathcal{A}} \text{ accepts}) \times \mathbf{L_1}\left(\overline{\mathcal{T}}_{\mathcal{A}}; \widetilde{\mathcal{T}}_{\mathcal{A}}\right) \end{array} \right\} = \mathbf{0} \tag{5.3}$$

Equation (5.2) defines $\mathcal{A}$-security. $\widetilde{\mathcal{A}}$ may acquire only one of b_0, b_1 or $b_\oplus$ from the protocol. $\widetilde{\mathcal{T}}_{\mathcal{A}}$ must not provide any information about the other two values to $\widetilde{\mathcal{A}}$ other than what she can infer from the output alone. Secondly, for any distribution that $\widetilde{\mathcal{A}}$ can induce on $\overline{\mathcal{B}}$'s transcript, there must exist C' representing $\mathcal{A}$'s input bit c' which she can use and act honestly to achieve the same distribution on $\mathcal{T}_{\mathcal{B}}$. i.e for any $\widetilde{\mathcal{A}}$ there must exist an associated probability distribution on her input such that she can pick an input according to this distribution and act honestly in the protocol to achieve the same results.

Equation (5.3) defines $\mathcal{B}$-security. $\widetilde{\mathcal{T}}_{\mathcal{B}}$ must not provide any information about $\mathcal{A}$'s choice c to $\widetilde{\mathcal{B}}$. The second part of equation (5.3) enforces that at the same time, for any distribution that $\widetilde{\mathcal{B}}$ can induce on $\overline{\mathcal{A}}$'s transcript, there must exist (B_0', B_1') representing $\mathcal{B}$'s input pair (b_0', b_1') which he can use and act honestly to achieve

the same distribution on $\mathcal{T}_{\mathcal{A}}$. i.e for any $\widetilde{\mathcal{B}}$ there must exist an associated probability distribution on his input such that he can pick an input according to this distribution and act honestly in the protocol to achieve the same results.

5.3 A reduction from TOX-$\binom{2}{1}$ to $\binom{2}{1}$–OT

We now present a TOX-$\binom{2}{1}$ reduction which uses two runs of $\binom{2}{1}$–OT. This construction was first presented by Crépeau and Sántha [10], and formal proof of correctness was presented for honest players only. In the later section, we will use this construction to show that under certain conditions, if a dishonest party is given access to an ϵ-attacker to this specific TOX-$\binom{2}{1}$ protocol, this attacker can be can be used as an ϵ'-attacker to a $\binom{2}{1}$–OT instance.

Assume $\mathcal{A}$ is the receiver, and $\mathcal{B}$ is the sender in the TOX-$\binom{2}{1}$ protocol.

Protocol 5.1 ($(c)(b_0, b_1)$**TOX-**$\binom{2}{1}$ **)**

1: *$\mathcal{A}$ has a choice trit $c \in \{0, 1, \oplus\}$ and $\mathcal{B}$ has two bits (b_0, b_1).*

2: *$\mathcal{A}$ constructs a random bit-matrix* $C = \begin{pmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{pmatrix}$

such that $c_{00} \oplus c_{01} = 1$ if and only if $c = 0$ or $c = \oplus$

$c_{10} \oplus c_{11} = 1$ if and only if $c = 1$ or $c = \oplus$

3: *$\mathcal{A}$ runs $\binom{2}{1}$–$OT(c_{00}, c_{01})(b_0)$ with $\mathcal{B}$. $\mathcal{B}$ gets c_{0b_0}.*

4: *$\mathcal{A}$ runs $\binom{2}{1}$–$OT(c_{10}, c_{11})(b_1)$ with $\mathcal{B}$. $\mathcal{B}$ gets c_{1b_1}.*

5: *$\mathcal{B}$ computes $b' = c_{0b_0} \oplus c_{1b_1}$ and sends b' to $\mathcal{A}$.*

6: *$\mathcal{A}$ computes $b_c = c_{00} \oplus c_{10} \oplus b'$*

5.3.1 Proof of Correctness for honest players

Theorem 5.1 *If $\mathcal{A}$ and $\mathcal{B}$ follow the protocol honestly, then $\mathcal{A}$'s output will be b_c*

Proof. We make use of the following trivial identity.

Lemma 5.2

$$\forall b, c_0, c_1[c_0 \oplus c_b = b \wedge (c_0 \oplus c_1)]$$

We have the following equalities.

$$\begin{aligned} output &= c_{00} \oplus c_{10} \oplus b' \\ &= c_{00} \oplus c_{10} \oplus \left(c_{0b_0} \oplus c_{1b_1}\right) \\ &= (c_{00} \oplus c_{0b_0}) \oplus (c_{10} \oplus c_{1b_1}) \\ &= b_0 \wedge (c_{00} \oplus c_{01}) \oplus b_1 \wedge (c_{10} \oplus c_{11}) \qquad \text{by Lemma 5.2} \\ output &= b_c \end{aligned} \tag{5.4}$$

Equation (5.4) is correct because,

if $c = 0$, then we have $c_{00} \oplus c_{01} = 1$ and $c_{10} \oplus c_{11} = 0$. Equation (5.4) becomes

$$\begin{aligned} output &= (b_0 \wedge 1) \oplus (b_1 \wedge 0) \\ &= b_0 \end{aligned}$$

If $c = 1$ then we have $c_{00} \oplus c_{01} = 0$ and $c_{10} \oplus c_{11} = 1$. Equation (5.4) becomes

$$\begin{aligned} output &= (b_0 \wedge 0) \oplus (b_1 \wedge 1) \\ &= b_1 \end{aligned}$$

If $c = \oplus$ then we have $c_{00} \oplus c_{01} = 1$ and $c_{10} \oplus c_{11} = 1$. Equation (5.4) becomes

$$\begin{aligned} output &= (b_0 \wedge 1) \oplus (b_1 \wedge 1) \\ &= b_{\oplus} \end{aligned}$$

5.3.2 Proof of Security for dishonest players

For honest players, theorem (5.1) provides a correctness proof for the TOX-$\binom{2}{1}$ protocol. We cannot provide a proof of *correctness* of our construction for dishonest players, because in step 5 of the protocol, $\widetilde{\mathcal{B}}$ can set b' to any value and send it to $\overline{\mathcal{A}}$. This is a violation of the *correctness* constraint of TOX-$\binom{2}{1}$, as for a fixed set of $\widetilde{\mathcal{B}}$'s effective input bits (b'_0, b'_1), and for any choice c used by $\overline{\mathcal{A}}$ the output b'_0, b'_1 or $b_{\oplus}$ $\overline{\mathcal{A}}$ gets from the protocol may not be consistent. For instance when $b_0 = b_1 = 0$, $b_{\oplus} = 1$. Note that this is not equivalent to $\widetilde{\mathcal{B}}$ doing local computation on its initial input bits and then using the results in the protocol and acting honestly.

Let's denote this faulty construction as $\overline{\text{TOX-}\binom{2}{1}}$. In the next chapter, we will present a construction of Reverse Oblivious Transfer (TO–$\binom{2}{1}$) which uses $\overline{\text{TOX-}\binom{2}{1}}$ as a subroutine.

5.4 A privacy attack on $\binom{2}{1}$–OT

Assume that a dishonest party is given access to a cheating algorithm which can successfully attack the privacy of $\overline{\text{TOX-}\binom{2}{1}}$ protocol. Given such an algorithm, a dishonest party can use it to create a privacy compromising transcript of a $\binom{2}{1}$–OT instance. This is an efficient reduction without any assumptions or bounds on the power of the attacker. Note that we do not claim that the information can be retrieved (efficiently or otherwise) from the transcript.

5.4.1 A $\binom{2}{1}$–OT privacy attack by a Dishonest Sender

Given access to a $\overline{\text{TOX-}\binom{2}{1}}$ attacking program $\mathcal{H}$, we show that $\widetilde{\mathcal{A}}$ acting as a sender, can successfully attack the privacy of an $\binom{2}{1}$–OT instance. Let (B_0, B_1) and C be random variables (RV) representing the inputs of $\mathcal{A}$ and $\mathcal{B}$ respectively. We assume that both $\mathcal{A}$ and $\mathcal{B}$ are aware of the arbitrary joint probability distribution of these random variables $P_{B_0,B_1,C}$. A sample (b_0, b_1, c) is generated from the distribution and (b_0, b_1) is provided as $\mathcal{A}$'s $\binom{2}{1}$–OT input while c is provided as $\mathcal{B}$'s $\binom{2}{1}$–OT input. $\widetilde{\mathcal{A}}$ embeds this $\binom{2}{1}$–OT instance in a run of $\overline{\text{TOX-}\binom{2}{1}}$. During the execution of $\overline{\text{TOX-}\binom{2}{1}}$, she randomly selects one of the two instances of the underlying $\binom{2}{1}$–OT and allows $\mathcal{H}$ to run that instance with $\overline{\mathcal{B}}$. $\widetilde{\mathcal{A}}$ acting as an honest receiver runs the other instance of $\binom{2}{1}$–OT with $\mathcal{H}$ using a dummy input bit. Note that during the execution of the protocol, $\mathcal{H}$ is oblivious to the fact that it is running the two instances of $\binom{2}{1}$–OT of $\overline{\text{TOX-}\binom{2}{1}}$ with two different parties. Also note that $\widetilde{\mathcal{A}}$ does not provide any initial $\binom{2}{1}$–OT input to $\mathcal{H}$. We show with non-zero probability, that $\mathcal{H}$ produces a transcript (for $\binom{2}{1}$–OT) that contains illegal information about Bob's choice bit, thereby violating the privacy of $\binom{2}{1}$–OT. Note that our only claim is that a transcript containing illegal information can be created with non-zero probability.

Complete description of the attack is provided as follows. For the $\binom{2}{1}$–OT instance which $\widetilde{\mathcal{A}}$ wants to initiate with $\overline{\mathcal{B}}$, (x_0, x_1) is provided as her input while c is provided as $\overline{\mathcal{B}}$'s input. Figure 5.4.1 provides a visualization of the attack.

Protocol 5.2 (An attack on $\binom{2}{1}$–OT $\mathcal{A}$-privacy)

1: *$\widetilde{\mathcal{A}}$ chooses $s \in_u \{0,1\}$ and selects bit $b_{\bar{s}} \in_u \{0,1\}$.*

2: *$\widetilde{\mathcal{A}}$ initializes the cheating algorithm $\mathcal{H}$ and lets it run until it is ready to run $\overline{TOX\text{-}\binom{2}{1}}$.*

3: *$\widetilde{\mathcal{A}}$ initiates an instance of $\overline{TOX\text{-}\binom{2}{1}}$ with $\mathcal{H}$.*

4:

- *If s = 0, $\widetilde{\mathcal{A}}$ uses $\mathcal{H}$ (sender) to run $\binom{2}{1}$–$OT(c_{00}, c_{01})(b_0)$ with $\overline{\mathcal{B}}$. $\widetilde{\mathcal{A}}$ (receiver) runs $\binom{2}{1}$–$OT(c_{10}, c_{11})(b_1)$ with $\mathcal{H}$.*
- *If s = 1, $\widetilde{\mathcal{A}}$ (receiver) runs $\binom{2}{1}$–$OT(c_{00}, c_{01})(b_0)$ with $\mathcal{H}$. $\widetilde{\mathcal{A}}$ uses $\mathcal{H}$ (sender) to run $\binom{2}{1}$–$OT(c_{10}, c_{11})(b_1)$ with $\overline{\mathcal{B}}$.*

5: *$\widetilde{\mathcal{A}}$ stops $\mathcal{H}$ and gets the effective inputs (c_{s0}, c_{s1}) used by $\mathcal{H}$ in the $\binom{2}{1}$–$OT(c_{s0}, c_{s1})(b_s)$ instance from its transcript. If $(c_{s0}, c_{s1}) = (x_0, x_1)$, $\widetilde{\mathcal{A}}$ outputs $\mathcal{H}$'s transcript of the $\binom{2}{1}$–$OT(c_{s0}, c_{s1})(b_s)$ instance, else she aborts the attack.*

Note that in step 5 of the protocol, $\widetilde{\mathcal{A}}$ extracts the effective input bits used by $\mathcal{H}$. By the definition of $\binom{2}{1}$–OT $\mathcal{A}$-security (section 4.2.1), we are guaranteed that there exist a function f which when given the transcript as input, maps the given transcript to a pair of effective input bits.

Also note that $\widetilde{\mathcal{A}}$ discards the transcript if $\mathcal{H}$ effective input bits are not the same as the ones (x_0, x_1) provided to $\widetilde{\mathcal{A}}$ at the start of the protocol. The reason is that $\mathcal{H}$ chooses his effective input bits independently of $\widetilde{\mathcal{A}}$'s inputs. Even if $\widetilde{\mathcal{A}}$ only wishes to learn $\overline{\mathcal{B}}$'s choice bit, she can't use information based on the different set of inputs chosen by $\mathcal{H}$ as both parties' inputs are correlated through a joint probability distribution. It may be the case that the bit pair chosen by $\mathcal{H}$ has a zero probability of occurring. Provided that all four possible pairs of input of $\widetilde{\mathcal{A}}$ have a

non-zero probability of occurring (according to the joint probability distribution), then we have a non-zero probability of producing an illegal transcript. For example, lets say that all four input pairs have non-zero probabilities (a, b, c, d). Then however $\mathcal{H}$ chooses its own effective input bits, the probability they match is at least $\min(a, b, c, d)$. As $\mathcal{H}$ chooses independently of what was handed to $\widetilde{\mathcal{A}}$, so in the worst case, it always chooses the least frequent pair.

Another important point to note is that $\widetilde{\mathcal{A}}$ stopped $\mathcal{H}$ before getting bit b' which is a required step of $\overline{\text{TOX-}\binom{2}{1}}$ protocol. By the construction of $\overline{\text{TOX-}\binom{2}{1}}$ protocol, any illegal information gained about the choice bit (c) of $\overline{\mathcal{B}}$ in $\binom{2}{1}$–OT has to be obtained prior to receiving bit b'. If both $\binom{2}{1}$–OT are privacy preserving, then the knowledge of b' alone does not suffice to compromise privacy. We also provide a bound on the information carried by b' to show that it does not help in compromising TOX-$\binom{2}{1}$ $\mathcal{A}$-privacy. Let $C_{\bar{s}}$ and C_s denote the outputs of the two $\binom{2}{1}$–OT instances. The outcome values of $C_{\bar{s}}$ and C_s can be fixed, or a function of $\overline{\mathcal{B}}$'s input bits $b_{\bar{s}}$ and b_s respectively, i.e. $C_{\bar{s}} \in \{\bar{0}, \bar{1}, b_{\bar{s}}, \neg b_{\bar{s}}\}$[1] [2] , and $C_s \in \{\bar{0}, \bar{1}, b_s, \neg b_s\}^2$. Therefore $b' \in \{\bar{0}, \bar{1}, b_{\bar{s}} \oplus b_s, \neg b_{\bar{s}} \oplus b_s, b_{\bar{s}} \oplus \neg b_s, \neg b_{\bar{s}} \oplus \neg b_s\}$. This set only provides the knowledge of only one of $b_{\bar{s}}$, b_s or $(b_{\bar{s}} \oplus b_s)$ to $\mathcal{H}$ which is not a violation of privacy.

We now present an information-theoretic proof that TOX-$\binom{2}{1}$ privacy with dishonest sender reduces to $\binom{2}{1}$–OT privacy with dishonest sender.

---

[1] $\neg b_{\bar{s}} = 1 \oplus b_{\bar{s}}$

[2] $\bar{0}$ and $\bar{1}$ are functions which constantly output 0 and 1 respectively.

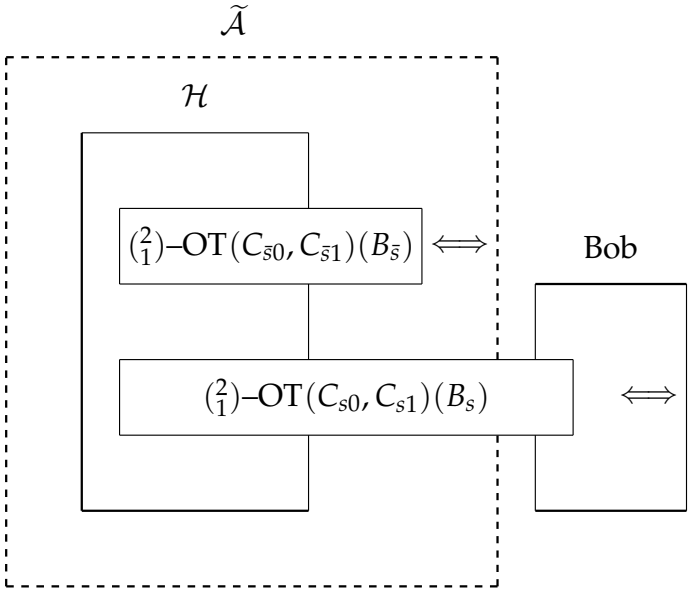

Figure 5–1: An $\mathcal{A}$-*attack* on $\binom{2}{1}$–OT

5.4.2 Information Theoretic Proof of Privacy Reduction

Let C' denote the effective choice computed as $C' = C_{\bar{s}0} \oplus C_{\bar{s}1} | C_{s0} \oplus C_{s1}$. $C' \in \{0, 1, \oplus\}$. Let $-C'$ and $+C'$ denote the other two choices.

Given:

$$\mathbf{I}\Big(B_{-C'}; \widetilde{\mathcal{T}}_{\mathcal{A}} \;\Big|\; C', B_{C'}\Big) + \mathbf{I}\Big(B_{+C'}; \widetilde{\mathcal{T}}_{\mathcal{A}} \;\Big|\; C', B_{C'}\Big) > \epsilon \tag{5.5}$$

Equation (5.5) can be written as:

$$\begin{aligned} &\mathbf{H}\big(B_{-C'} \mid C', B_{C'}\big) - \mathbf{H}\Big(B_{-C'} \mid C', \widetilde{\mathcal{T}}_{\mathcal{A}}, B_{C'}\Big) + \\ &\mathbf{H}\big(B_{+C'} \mid C', B_{C'}\big) - \mathbf{H}\Big(B_{+C'} \mid C', \widetilde{\mathcal{T}}_{\mathcal{A}}, B_{C'}\Big) > \epsilon \end{aligned} \tag{5.6}$$

Note that

$$\begin{aligned}\mathbf{H}\big(B_{-C'}, B_{+C'} \mid C', B_{C'}\big) &= \mathbf{H}\big(B_{-C'} \mid C', B_{C'}\big) + \mathbf{H}\big(B_{+C'} \mid C', B_{-C'}, B_{C'}\big) \\ &= \mathbf{H}\big(B_{-C'} \mid C', B_{C'}\big) \end{aligned} \tag{5.7}$$

and

$$\begin{aligned}\mathbf{H}\Big(B_{-C'}, B_{+C'} \mid C', \widetilde{\mathcal{T}}_{\mathcal{A}}, B_{C'}\Big) &= \mathbf{H}\Big(B_{-C'} \mid C', \widetilde{\mathcal{T}}_{\mathcal{A}}, B_{C'}\Big) + \mathbf{H}\Big(B_{+C'} \mid C', \widetilde{\mathcal{T}}_{\mathcal{A}}, B_{-C'}, B_{C'}\Big) \\ &= \mathbf{H}\Big(B_{-C'} \mid C', \widetilde{\mathcal{T}}_{\mathcal{A}}, B_{C'}\Big) \end{aligned} \tag{5.8}$$

Similarly,

$$\mathbf{H}\big(B_{-C'}, B_{+C'} \mid C', B_{C'}\big) = \mathbf{H}\big(B_{+C'} \mid C', B_{C'}\big) \tag{5.9}$$

and

$$\mathbf{H}\Big(B_{-C'}, B_{+C'} \mid C', \widetilde{\mathcal{T}}_{\mathcal{A}}, B_{C'}\Big) = \mathbf{H}\Big(B_{+C'} \mid C', \widetilde{\mathcal{T}}_{\mathcal{A}}, B_{C'}\Big) \tag{5.10}$$

Comparing equation (5.6) with equation (5.7) and equation (5.8), we have

$$\mathbf{I}\Big(B_{-C'}; \widetilde{\mathcal{T}}_{\mathcal{A}} \;\Big|\; C', B_{C'}\Big) = \mathbf{H}\big(B_{-C'}, B_{+C'} \mid C', B_{C'}\big) - \mathbf{H}\Big(B_{-C'}, B_{+C'} \mid C', \widetilde{\mathcal{T}}_{\mathcal{A}}, B_{C'}\Big) \tag{5.11}$$

Comparing equation (5.6) with equation (5.9) and equation (5.10), we have

$$\mathbf{I}\Big(B_{+C'}; \widetilde{\mathcal{T}}_{\mathcal{A}} \;\Big|\; C', B_{C'}\Big) = \mathbf{H}\big(B_{-C'}, B_{+C'} \mid C', B_{C'}\big) - \mathbf{H}\Big(B_{-C'}, B_{+C'} \mid C', \widetilde{\mathcal{T}}_{\mathcal{A}}, B_{C'}\Big) \tag{5.12}$$

Equation (5.11) and equation (5.12) show that both terms on the left hand side of equation (5.6) contribute equally towards the final outcome value ϵ.

We have

$$\mathbf{I}\Big(B_{+C'}; \widetilde{\mathcal{T}}_{\mathcal{A}} \;\Big|\; C', B_{C'}\Big) > \frac{\epsilon}{2} \tag{5.13}$$

and

$$\mathbf{I}\Big(B_{-C'}; \widetilde{\mathcal{T}}_{\mathcal{A}} \;\Big|\; C', B_{C'}\Big) > \frac{\epsilon}{2} \tag{5.14}$$

$\widetilde{\mathcal{A}}$ is concerned about the knowledge of $\overline{\mathcal{B}}$'s input B_s. Note that it is possible that $\mathcal{H}$ can set $c' = s$ as its effective choice. As s is randomly chosen, $\overline{\mathcal{B}}$ either participates in the first or the second instance of $\binom{2}{1}$–OT with probability exactly $\frac{1}{2}$. Therefore in at least half of the cases, either $-c'$ or $+c'$ will correspond to $\overline{\mathcal{B}}$'s input bit. In these cases,

$$\begin{aligned}\mathbf{I}\Big(B_s; \widetilde{\mathcal{T}}_{\mathcal{A}_s} | C', B_{C'}\Big) &> \frac{\epsilon}{2} \\ &> \epsilon' \qquad \because \epsilon' = \frac{1}{2} \times \epsilon\end{aligned} \tag{5.15}$$

if $\mathcal{H}$ is able to gain knowledge about $B_{-C'}$ and $B_{+C'}$, at least half of the time this knowledge is gained through a compromise of the $\mathcal{A}$-privacy of the $\binom{2}{1}$–OT$_s$.

5.4.3 A $\binom{2}{1}$–OT privacy attack by a Dishonest Receiver

Given access to a $\overline{\text{TOX-}\binom{2}{1}}$ attacking program $\mathcal{K}$, we show that $\widetilde{\mathcal{B}}$ acting as a receiver, can successfully attack the receiver-privacy of an $\binom{2}{1}$–OT instance. Let (B_0, B_1) and C be random variables (RV) representing the inputs of $\mathcal{A}$ and $\mathcal{B}$ respectively. We assume that both $\mathcal{A}$ and $\mathcal{B}$ are aware of the arbitrary joint probability distribution of these random variables $P_{B_0,B_1,C}$. A sample (b_0, b_1, c) is generated

from the distribution and (b_0, b_1) is provided as $\mathcal{A}$'s $\binom{2}{1}$–OT input while c is provided as $\mathcal{B}$'s $\binom{2}{1}$–OT input. $\widetilde{\mathcal{B}}$ embeds this $\binom{2}{1}$–OT instance in a run of $\overline{\text{TOX-}\binom{2}{1}}$. During the execution of $\overline{\text{TOX-}\binom{2}{1}}$, he randomly selects one of the two instances of the underlying $\binom{2}{1}$–OT and allows $\mathcal{K}$ to run that instance with $\overline{\mathcal{A}}$. $\widetilde{\mathcal{B}}$ acting as an honest sender runs the other instance of $\binom{2}{1}$–OT with $\mathcal{K}$. For this instance, $\widetilde{\mathcal{B}}$ selects two input bits $(c_{\bar{s}0}, c_{\bar{s}1})$ such that $c_{\bar{s}0} \oplus c_{\bar{s}1} = 1$. This is to imitate a valid choice c in the $\overline{\text{TOX-}\binom{2}{1}}$ protocol as $\mathcal{K}$ may only be able to successfully compromise the $\mathcal{B}$-privacy if the other party is honest.

Note that during the execution of the protocol, $\mathcal{K}$ is oblivious to the fact that it is running the two instances of $\binom{2}{1}$–OT of $\overline{\text{TOX-}\binom{2}{1}}$ with two different parties. Also note that $\widetilde{\mathcal{B}}$ does not provide any initial $\binom{2}{1}$–OT input to $\mathcal{H}$. We show with non-zero probability, that $\mathcal{K}$ produces a transcript (for $\binom{2}{1}$–OT) that contains illegal information about $\overline{\mathcal{A}}$'s choice bits, thereby violating the privacy of $\binom{2}{1}$–OT. Note that our only claim is that a transcript containing illegal information can be created with non-zero probability.

Complete description of the attack is provided as follows. For the $\binom{2}{1}$–OT instance which $\widetilde{\mathcal{B}}$ wants to initiate with $\overline{\mathcal{A}}$, (x_0, x_1) is provided as $\overline{\mathcal{A}}$'s input while c is provided as $\widetilde{\mathcal{B}}$'s input.

Protocol 5.3 (Attack on $\binom{2}{1}$–OT using an adversary to TOX-$\binom{2}{1}$)

1: *$\widetilde{\mathcal{B}}$ chooses $s \in_u \{0,1\}$. $\widetilde{\mathcal{B}}$ selects bits $(c_{\bar{s}0}, c_{\bar{s}1})$ such that $c_{\bar{s}0} \oplus c_{\bar{s}1} = 1$.*

2: *$\widetilde{\mathcal{B}}$ initializes the cheating algorithm $\mathcal{K}$, and lets it run until it is ready to run $\overline{TOX\text{-}\binom{2}{1}}$.*

3: *$\widetilde{\mathcal{B}}$ initiates an instance of TOX-$\binom{2}{1}$ with $\mathcal{K}$.*

4:
- *If s = 0, $\widetilde{\mathcal{B}}$ uses $\mathcal{K}$ (receiver) to run $\binom{2}{1}$–$OT(c_{00}, c_{01})(b_0)$ with $\overline{\mathcal{A}}$. $\widetilde{\mathcal{B}}$ (sender) runs the second instance $\binom{2}{1}$–$OT(c_{10}, c_{11})(b_1)$ with $\mathcal{K}$.*
- *If s = 1, $\widetilde{\mathcal{B}}$ (sender) runs the first instance $\binom{2}{1}$–$OT(c_{00}, c_{01})(b_0)$ with $\mathcal{K}$. $\widetilde{\mathcal{B}}$ uses $\mathcal{K}$ (receiver) to run $\binom{2}{1}$–$OT(c_{10}, c_{11})(b_1)$ with $\overline{\mathcal{A}}$.*

5: *$\widetilde{\mathcal{B}}$ runs $\mathcal{K}$ until it provides b'.*

6: *$\widetilde{\mathcal{B}}$ lets $\mathcal{K}$ run to completion.*

7: *$\widetilde{\mathcal{B}}$ gets $\mathcal{K}$'s input (b_s) used in $\binom{2}{1}$–$OT(c_{s0}, c_{s1})(b_s)$ from its transcript. If $b_s = c$, $\widetilde{\mathcal{B}}$ outputs $\mathcal{K}$'s $\binom{2}{1}$–$OT(c_{s0}, c_{s1})(b_s)$ transcript, else he aborts the attack.*

Note that in step 7 of the protocol, $\widetilde{\mathcal{B}}$ extracts the effective input bit used by $\mathcal{K}$. By the definition of $\binom{2}{1}$–OT $\mathcal{B}$-security (section 4.2.2), we are guaranteed that there exist a function f which when given the transcript as input, maps the transcript to an effective input bit.

Also note that $\widetilde{\mathcal{B}}$ discards the transcript if $b_s \neq c$. The reason is that $\mathcal{K}$ chooses its effective input bit independently of $\widetilde{\mathcal{B}}$'s input. Even if $\widetilde{\mathcal{B}}$ only wishes to learn $\overline{\mathcal{A}}$'s input bits, he can't use information based on a different input chosen by $\mathcal{K}$ as both parties' inputs are correlated through a joint probability distribution. It may be the case that the bit chosen by $\mathcal{K}$ has a zero probability of occurring.

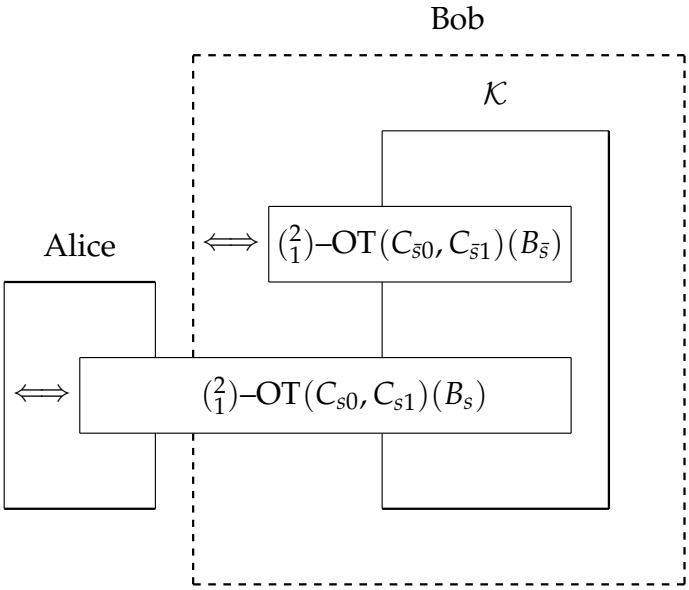

Figure 5–2: A $\mathcal{B}$-*attack* on $\binom{2}{1}$–OT

Another important point to note is that if $\widetilde{\mathcal{B}}$'s two possible inputs $(c = \{0, 1\})$ have a non-zero probability of occurring (according to the joint probability distribution), then we have a non-zero probability of producing an illegal transcript. The reason is that however $\mathcal{K}$ chooses its own effective input bit, the probability they match is the minimum of the probabilities of the two possible choices. As $\mathcal{K}$ chooses independently of what was handed to $\widetilde{\mathcal{B}}$, so in the worst case, it always chooses the least frequent bit.

5.4.4 Information Theoretic Proof of Privacy Reduction

Let C denote receiver's choice in $\overline{\text{TOX-}\binom{2}{1}}$. C is determined by the two pairs of inputs used in the two $\binom{2}{1}$–OT instances i.e $C = C_{s0} \oplus C_{s1} | C_{\bar{s}0}, C_{\bar{s}1}$. If $\mathcal{K}$ is able to successfully gain knowledge of C from $\overline{\text{TOX-}\binom{2}{1}}$, we claim that this is only possible

if $\mathcal{K}$ was able to successfully compromise the $\mathcal{B}$-privacy of the underlying $\binom{2}{1}$–OT instances.

Given:

$$\begin{aligned} &\mathbf{I}\Big(C; \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) > \epsilon \\ \implies &\mathbf{H}(C) - \mathbf{H}\Big(C|\widetilde{\mathcal{T}}_{\mathcal{B}}\Big) > \epsilon \end{aligned} \tag{5.16}$$

Note that C is a random variable which defines and is defined by $C_{s0} \oplus C_{s1}$ and $C_{\bar{s}0} \oplus C_{\bar{s}1}$. Hence $\mathbf{H}(C)$ can be written as as a joint entropy of these random variables.

$$\mathbf{H}(C) = \mathbf{H}(C_{s0} \oplus C_{s1}, C_{\bar{s}0} \oplus C_{\bar{s}1}) \tag{5.17}$$

and

$$\mathbf{H}\Big(C \mid \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) = \mathbf{H}\Big(C_{s0} \oplus C_{s1}, C_{\bar{s}0} \oplus C_{\bar{s}1} \mid \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) \tag{5.18}$$

Substituting the values in equation (5.16).

$$\mathbf{H}(C_{s0} \oplus C_{s1}, C_{\bar{s}0} \oplus C_{\bar{s}1}) - \mathbf{H}\Big(C_{s0} \oplus C_{s1}, C_{\bar{s}0} \oplus C_{\bar{s}1} \mid \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) > \epsilon \tag{5.19}$$

For sake of reading clarity denote:

$X = C_{s0} \oplus C_{s1}$

$Y = C_{\bar{s}0} \oplus C_{\bar{s}1}$

$\binom{2}{1}$–$\mathrm{OT}_s = \binom{2}{1}$–$\mathrm{OT}(C_{s0}, C_{s1})(B_s)$

$\binom{2}{1}$–$\mathrm{OT}_{\bar{s}} = \binom{2}{1}$–$\mathrm{OT}(C_{\bar{s}0}, C_{\bar{s}1})(B_{\bar{s}})$

Equation (5.19) is can now be written as:

$$\mathbf{H}(X, Y) - \mathbf{H}\Big(X, Y \mid \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) > \epsilon \tag{5.20}$$

Now

$$\mathbf{H}(X, Y) = \mathbf{H}(X) + \mathbf{H}(Y \mid X) \tag{5.21}$$

and

$$\mathbf{H}\Big(X, Y \mid \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) = \mathbf{H}\Big(X \mid \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) + \mathbf{H}\Big(Y \mid X, \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) \tag{5.22}$$

Substituting the values from equations (5.21) and (5.22) in equation (5.20)

$$\begin{aligned}
&\mathbf{H}(X) + \mathbf{H}(Y \mid X) - \mathbf{H}\Big(X \mid \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) - \mathbf{H}\Big(Y \mid X, \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) > \epsilon \\
&\mathbf{H}(X) - \mathbf{H}\Big(X \mid \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) + \mathbf{H}(Y \mid X) - \mathbf{H}\Big(Y \mid X, \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) > \epsilon \\
&\mathbf{I}\Big(X; \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) + \mathbf{I}\Big(Y; \widetilde{\mathcal{T}}_{\mathcal{B}} \mid X\Big) > \epsilon
\end{aligned} \tag{5.23}$$

Note that depending upon the outcome value of X and $\mathcal{K}$'s apriori knowledge of Y, $\mathbf{I}\Big(Y; \widetilde{\mathcal{T}}_{\mathcal{B}} \mid X\Big)$ can take a range of values. The two extreme cases are:

$$\mathbf{I}\Big(Y; \widetilde{\mathcal{T}}_{\mathcal{B}} \mid X\Big) = \begin{cases} 0 & \text{When X provides complete knowledge of Y,} \\ \mathbf{I}\Big(Y; \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) & \text{When X provides no knowledge of Y.} \end{cases}$$

When $\mathbf{I}\Big(Y; \widetilde{\mathcal{T}}_{\mathcal{B}} \mid X\Big) = 0$, equation (5.23) becomes

$$\mathbf{I}\Big(X; \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) > \epsilon \tag{5.24}$$

When $\mathbf{I}\Big(Y; \widetilde{\mathcal{T}}_{\mathcal{B}} \mid X\Big) = \mathbf{I}\Big(Y; \widetilde{\mathcal{T}}_{\mathcal{B}}\Big)$, equation (5.23) becomes

$$\mathbf{I}\Big(X; \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) + \mathbf{I}\Big(Y; \widetilde{\mathcal{T}}_{\mathcal{B}}\Big) > \epsilon \tag{5.25}$$

The probability of $\overline{\mathcal{A}}$ participating in either the first or the second instance of the $\binom{2}{1}$–OT is exactly $\frac{1}{2}$ as s is chosen randomly. Therefore,

$$\begin{aligned}\mathbf{I}\Big(X;\widetilde{\mathcal{T}}_{\mathcal{B}}\Big) =& \frac{1}{2}\times\Big[\mathbf{H}(C_{00}\oplus C_{01}\,)+\mathbf{H}\Big(C_{00}\oplus C_{01}|\widetilde{\mathcal{T}}_{\mathcal{B}}\Big)\Big]- \\ & \frac{1}{2}\times\Big[\mathbf{H}(C_{10}\oplus C_{11})+\mathbf{H}\Big(C_{10}\oplus C_{11}\mid\widetilde{\mathcal{T}}_{\mathcal{B}}\Big)\Big]\end{aligned} \tag{5.26}$$

Similarly,

$$\begin{aligned}\mathbf{I}\Big(Y;\widetilde{\mathcal{T}}_{\mathcal{B}}\Big) =& \frac{1}{2}\times\Big[\mathbf{H}(C_{00}\oplus C_{01}\,)+\mathbf{H}\Big(C_{00}\oplus C_{01}|\widetilde{\mathcal{T}}_{\mathcal{B}}\Big)\Big]- \\ & \frac{1}{2}\times\Big[\mathbf{H}(C_{10}\oplus C_{11})+\mathbf{H}\Big(C_{10}\oplus C_{11}\mid\widetilde{\mathcal{T}}_{\mathcal{B}}\Big)\Big]\end{aligned} \tag{5.27}$$

Comparing equations (5.26) and (5.27), it is clear that

$$\mathbf{I}\Big(X;\widetilde{\mathcal{T}}_{\mathcal{B}}\Big)=\mathbf{I}\Big(Y;\widetilde{\mathcal{T}}_{\mathcal{B}}\Big) \tag{5.28}$$

Both these terms contribute equally in equation (5.23). Therefore,

$$\begin{aligned}\mathbf{H}(X)-\mathbf{H}\Big(X\mid\widetilde{\mathcal{T}}_{\mathcal{B}}\Big) &> \frac{1}{2}\times\epsilon \\ &> \epsilon' \qquad \text{where } \epsilon'=\frac{1}{2}\times\epsilon\end{aligned} \tag{5.29}$$

From equation (5.24) and equation (5.28) it is clear that if $\mathcal{K}$ is able to gain knowledge about C, at least half of the time, this knowledge is gained through a compromise of the receiver-privacy of the desired $\binom{2}{1}$–OT$_s$ instance.

5.5 Conclusion

We have presented new security specifications for TOX-$\binom{2}{1}$ protocol. We have presented a reduction from TOX-$\binom{2}{1}$ to $\binom{2}{1}$–OT and provided a partial proof of security for privacy attacking adversaries. In the next chapter we present a construction of TO–$\binom{2}{1}$ based on $\overline{\text{TOX-}\binom{2}{1}}$.

CHAPTER 6
NEW TO–$\binom{2}{1}$ SECURITY SPECIFICATIONS

We present the security specifications for 1 out of 2 reverse Oblivious Transfer (TO–$\binom{2}{1}$) in this chapter. We also present a TO–$\binom{2}{1}$ protocol which is based on $\overline{\text{TOX-}\binom{2}{1}}$ protocol first presented in chapter 5.

6.1 TO–$\binom{2}{1}$ Specifications

When we visualize protocols such as $\binom{2}{1}$–OT, traditionally we assume the sender $\mathcal{A}$ to be on the left and the receiver $\mathcal{B}$ on the right and the flow of information is from left to right. TO–$\binom{2}{1}$ is an $\binom{2}{1}$–OT in reverse direction. We denote receiver as $\mathcal{A}$, and sender as $\mathcal{B}$.

We start by first presenting what TO–$\binom{2}{1}$ must achieve after a successful run. This protocol takes place between a receiver $\mathcal{A}$ and a sender $\mathcal{B}$. This protocol allows $\mathcal{B}$ to transfer one of his two input bits (b_0, b_1) to $\mathcal{A}$ who secretly chooses which bit (b_c) to get. The choice of $\mathcal{A}$ is represented by c. This is done in such a way that $\mathcal{B}$ does not gain any knowledge about c, while $\mathcal{A}$ gets only one of $\mathcal{B}$'s input bits as an output.

Definition 6.1 (TO–$\binom{2}{1}$) Protocol $[\mathcal{A}, \mathcal{B}]$ *implements* TO–$\binom{2}{1}$ if
$\forall b_0, b_1, c \in \mathcal{F}_2$

$$[\overline{\mathcal{A}}, \overline{\mathcal{B}}](b_0, b_1)(c) = (b_c, \varepsilon) \tag{6.1}$$

6.2 New TO–$\binom{2}{1}$ Security Specifications

The security specifications of TO–$\binom{2}{1}$ are exactly the same for $\binom{2}{1}$–OT, only the roles of $\mathcal{A}$ and $\mathcal{B}$ are changed.

Definition 6.2 (TO–$\binom{2}{1}$-Security) Protocol $[\mathcal{A}, \mathcal{B}]$ is secure for TO–$\binom{2}{1}$ if it is both $\mathcal{A}$-*Secure* and $\mathcal{B}$-*Secure*.

- ***$\mathcal{A}$-Security***

  Protocol $[\mathcal{A}, \mathcal{B}]$ is $\mathcal{A}$-secure for TO–$\binom{2}{1}$ if

  $\forall c \in \mathcal{F}_2,\ (B_0, B_1) \in RV(\mathcal{F}_2)^2,$ adv. $\widetilde{\mathcal{A}}, \exists$ function f,

  Let $\widetilde{\mathcal{T}} = \left[\widetilde{\mathcal{A}}, \overline{\mathcal{B}}\right]^*(c)(B_0, B_1)$

  $C' = f(\widetilde{\mathcal{T}}_{\mathcal{A}})$ and $\mathbf{I}(C'; B_0, B_1 \mid c) = 0$

  Let $\overline{\mathcal{T}} = \left[\overline{\mathcal{A}}, \overline{\mathcal{B}}\right]^*(C')(B_0, B_1)$

$$\max \left\{ \begin{array}{l} \mathbf{I}\Big(B_{\neg C'}; \widetilde{\mathcal{T}}_{\mathcal{A}} \;\Big|\; C', B_{C'}\Big) \\ \mathbf{Pr}\big(\overline{\mathcal{B}} \text{ accepts}\big) \times \mathbf{L_1}\left(\overline{\mathcal{T}}_{\mathcal{B}}; \widetilde{\mathcal{T}}_{\mathcal{B}}\right) \end{array} \right\} = \mathbf{0} \tag{6.2}$$

- ***$\mathcal{B}$-Security***

  Protocol $[\mathcal{A}, \mathcal{B}]$ is $\mathcal{B}$-secure for TO–$\binom{2}{1}$ if

  $\forall C \in RV(\mathcal{F}_2), b_0, b_1 \in \mathcal{F}_2,$ adv. $\widetilde{\mathcal{B}}, \exists$ function f,

  Let $\widetilde{\mathcal{T}} = \left[\overline{\mathcal{A}}, \widetilde{\mathcal{B}}\right]^*(C)(b_0, b_1)$

  $(B_0', B_1') = f(\widetilde{\mathcal{T}}_{\mathcal{B}})$

  Let $\overline{\mathcal{T}} = \left[\overline{\mathcal{A}}, \overline{\mathcal{B}}\right]^*(C)(B_0', B_1')$

$$\max \left\{ \begin{array}{l} \mathbf{I}\left(C; \widetilde{\mathcal{T}}_{\mathcal{B}}\right) \\ \mathbf{Pr}(\overline{\mathcal{A}} \text{ accepts}) \times \mathbf{L_1}\left(\overline{\mathcal{T}}_{\mathcal{A}}; \widetilde{\mathcal{T}}_{\mathcal{A}}\right) \end{array} \right\} = \mathbf{0} \tag{6.3}$$

We now present the construction of a TO–$\binom{2}{1}$ protocol. Such constructions are useful when we have a secure implementation of a cryptographic primitive in one direction only and we need to achieve the flow of output in other direction. This construction is based on an implementation of $\overline{\text{TOX-}\binom{2}{1}}$. In chapter 5, we argued that $\overline{\text{TOX-}\binom{2}{1}}$ construction had an inherent design fault which led to a compromise of sender-correctness without the knowledge of the receiver. In $\overline{\text{TOX-}\binom{2}{1}}$, privacy is enforced if we assume that the the underlying $\binom{2}{1}$–OT instances are secure. Figure 6–1 shows a visualization of the construction.

6.2.1 Construction of a TO–$\binom{2}{1}$ Protocol

Protocol 6.1 (TO–$\binom{2}{1}(c)(b_0, b_1)$)

1: *$\mathcal{A}$ has a choice bit c and $\mathcal{B}$ has two bits (b_0, b_1).*

2: *$\mathcal{B}$ picks two random n-bit strings x_0 and x_1.*

3: **DO**$_{i=1}^{n}$ $\overline{TOX\text{-}\binom{2}{1}}(c)(x_0^i, x_1^i)$. *$\mathcal{A}$ gets t_i.*

4: *$\mathcal{B}$ selects two $1 \times n$ matrices M_0, M_1 over $\mathcal{F}_2$, such that $b_0 = M_0 x_0$ and $b_1 = M_1 x_1$. $\mathcal{B}$ announces M_0, M_1 to $\mathcal{A}$.*

5: *$\mathcal{A}$ calculates $b_c = M_c t$.*

From the construction, it is clear that the security of TO–$\binom{2}{1}$ directly depends upon a privacy enforcing construction of $\overline{\text{TOX-}\binom{2}{1}}$. $\mathcal{A}$ and $\mathcal{B}$ only interact with each other through the $\overline{\text{TOX-}\binom{2}{1}}$ instances in the protocol. In the case of $\widetilde{\mathcal{A}}$, she may try

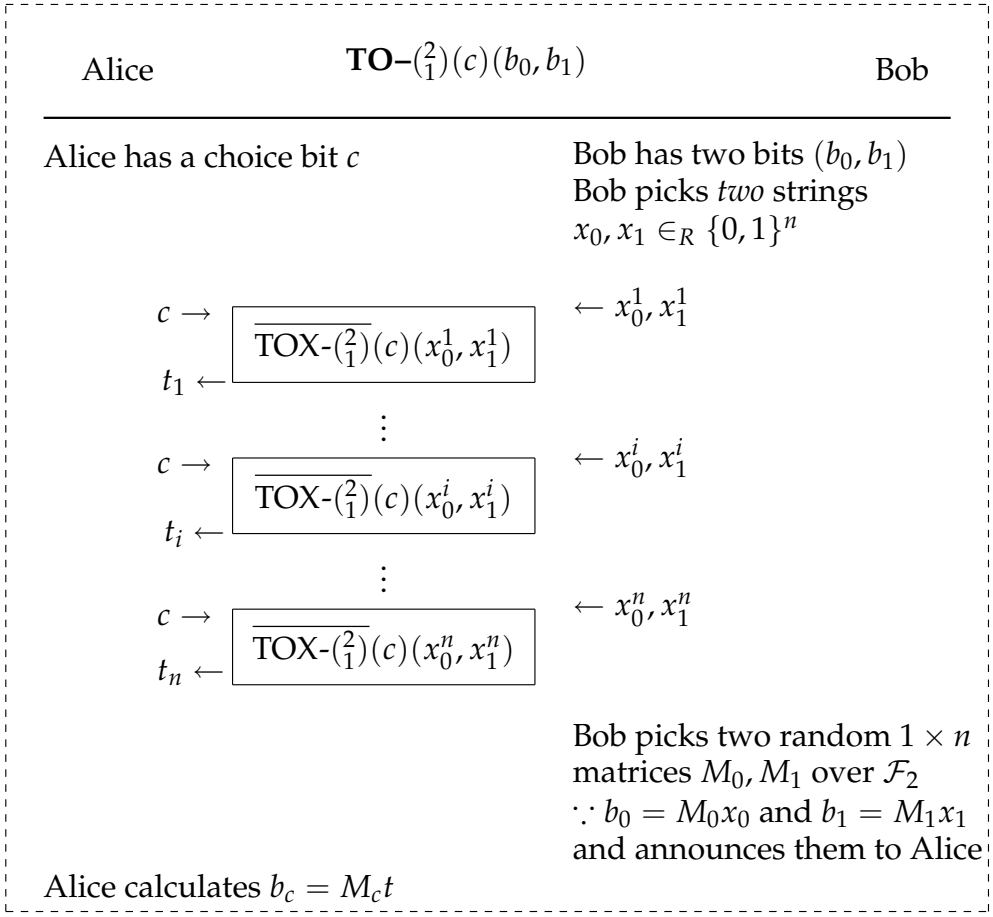

Figure 6–1: TO–$\binom{2}{1}(c)(b_0, b_1)$

to gain information about both x_0^i and x_1^i through the $\overline{\text{TOX-}\binom{2}{1}}$ instances. She may put $c = \oplus$ and use it in the $\overline{\text{TOX-}\binom{2}{1}}$ instances to get $x_0^i \oplus x_1^i$, but this is useless to her as at the end of the protocol she cannot recover b_c from this information. Similarly in the case of $\widetilde{\mathcal{B}}$, he may try to gain information about c. Therefore if the privacy of $\overline{\text{TOX-}\binom{2}{1}}$ is enforced, the privacy of TO–$\binom{2}{1}$ is also enforced. A formal proof of security for this specific construction is beyond the scope of this research and we leave it as an exercise for the esteemed reader.

6.3 Conclusion

We have presented a construction of TO–$\binom{2}{1}$ based on $\overline{\text{TOX-}\binom{2}{1}}$. We have also presented a new set of security specifications for TO–$\binom{2}{1}$.

CHAPTER 7
CONCLUSION

This thesis has sought to provide new and more robust security specifications for $\binom{2}{1}$–OT. Unlike the previous security specifications presented in chapter 3, the new correctness constraint now deals with the view of honest party instead of output only. This is a more robust and stronger notion of correctness as we provide bounds on all the information seen by the honest party during the protocol. This includes outputs of any intermediate steps.

In addition we have also shown an efficient reduction of TOX-$\binom{2}{1}$ privacy to $\binom{2}{1}$–OT privacy. Given access to privacy attacker to TOX-$\binom{2}{1}$, we have shown that it can be efficiently used to compromise the privacy of $\binom{2}{1}$–OT. This is done without making any assumptions about the setup of $\binom{2}{1}$–OT, or the power of the attacker. We have also provided a TO–$\binom{2}{1}$ protocol using a faulty TOX-$\binom{2}{1}$ protocol first presented in chapter 5.

This research work can be extended to answer at least two important open questions. Intuitively it seems that our new security specifications for $\binom{2}{1}$–OT are composable under sequential settings. It would be nice to have a formal proof of composability under such settings. Goldreich in [13] presented a new set of security specifications for two-party computation. He defined security in the real model as an emulation of the ideal model in which a trusted party computes the function and provides outputs to the players. In defining the behavior of an ideal

malicious adversary, he argued that three types of adversarial behaviors cannot be avoided. These are, *refusal to participate*, *substitution of its local input with any one of its choosing*, and *premature abortion of the protocol*. A real protocol is secure, if it emulates the ideal model in which the behavior of ideal adversary is restricted to these three actions. He also presented a formal proof to show that these specifications are composable under sequential settings. Our new security specifications also allow such adversarial behavior and enforce security under such conditions. Another extension to our work would be to present a formal proof of equivalence between our new security specifications and the ones presented by Goldreich.

Another possible venue to explore would be to present a formal proof of security of the TO–$\binom{2}{1}$ protocol presented in chapter 6.

Two-party security is a very important and ever evolving field, and in the end we hope that this research work contributes toward the proper understanding of the topics we have tackled.

REFERENCES

[1] BEAVER, D. Foundations of secure interactive computing. *Advances in Cryptology: Proceedings of Crypto '92* (1992), 377–391.

[2] BRASSARD, G., AND CRÉPEAU, C. Oblivious transfers and privacy amplification. *Lecture Notes in Computer Science 1233* (1997), 334–345.

[3] BRASSARD, G., CRÉPEAU, C., AND ROBERT, J. Information theoretic reductions among disclosure problems. *Proceedings of 27th Annual IEEE Symposium on Foundations of Computer Science* (1986), 168–173.

[4] BRASSARD, G., CRÉPEAU, C., AND WOLF, S. Oblivious transfers and privacy amplification. *Journal of Cryptology 16* (2003), 219–237.

[5] CANETTI, R. Universally composable security: a new paradigm for cryptographic protocols. In *42nd IEEE Symposium on Foundations of Computer Science* (2001), IEEE Computer Society Press, pp. 136–145.

[6] CRÉPEAU, C. Equivalence between two flavours of oblivious transfers. In *CRYPTO* (1987), pp. 350–354.

[7] CRÉPEAU, C. Verifiable disclosure of secrets and applications (abstract). In *Advances in Cryptology — EUROCRYPT '89* (1989), vol. 434 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 150–154.

[8] CRÉPEAU, C. *Correct and Private Reductions Among Oblivious Transfers*. PhD thesis, Massachusetts Institute of Technology, 1990.

[9] CRÉPEAU, C., AND KILIAN, J. Achieving oblivious transfer using weakened security assumptions (extended abstract). In *IEEE Symposium on Foundations of Computer Science* (1988), pp. 42–52.

[10] CRÉPEAU, C., AND SÁNTHA, M. On the reversibility of oblivious transfer. *Advances in Cryptology: Proceedings of Eurocrypt* (1991), 106–113.

[11] EVEN, S., GOLDREICH, O., AND LEMPEL, A. A randomized protocol for signing contracts. *Commun. ACM 28*, 6 (1985), 637–647.

[12] GOLDREICH, O. *Foundations of Cryptography*, vol. 1. Cambridge University Press, 2001.

[13] GOLDREICH, O. *Foundations of Cryptography*, vol. 2. Cambridge University Press, 2004.

[14] GOLDREICH, O., MICALI, S., AND WIGDERSON, A. How to play any mental game. In *Proceedings of the nineteenth annual ACM conference on Theory of computing* (1987), ACM Press, pp. 218–229.

[15] GOLDREICH, O., AND VAINISH, R. How to solve any protocol problem- an efficieny improvement. *Advances in Cryptology CRYPTO '87 293* (1988), 73–86.

[16] GOLDWASSER, S., MICALI, S., AND RACKOFF, C. The knowledge complexity of interactive proof-systems. *SIAM Journal on Computing 18* (1989), 186–208.

[17] KAHN, D. *The Codebreakers: The Story of Secret Writing*. Scribner, New York, NY, 1986.

[18] KILIAN, J. Founding cryptography on oblivious transfer. *Proceedings of 20th Annual ACM Symposium on Theory of Computing* (1988), 20–31.

[19] KILIAN, J. More general completeness theorems for secure two-party computation. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing* (2000), ACM Press, pp. 316–324.

[20] MICALI, S., AND ROGAWAY, P. Secure computation. *Advances in Cryptology CRYPTO '91 Proceedings* (1991), 392–404.

[21] RABIN, M. O. How to exchange secrets by oblivious transfer. Tech. Rep. TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[22] YAO, A. C.-C. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science (FOCS)* (1986), IEEE Computer Society, pp. 162–167.