数理解析研究所講究録
第 1599 巻 2008 年 65-72

Symmetricity of the Protocols Related to Oblivious Transfer

井上 大輔 田中 圭介
Daisuke Inoue Keisuke Tanaka

東京工業大学 情報理工学研究科
Department of Mathematical and Computing Sciences
Tokyo Institute of Technology
W8-55, 2-12-1 Ookayama Meguro-ku, Tokyo 152-8552, Japan
{inoue1, keisuke}@is.titech.ac.jp.

Abstract— In this paper, we show that the special case of strong conditional oblivious transfer from S to R can be obtained from only one instance of itself from R to S. Our reduction protocol is simple and efficient, and preserves the security of the inversion.

Keywords: oblivious transfer, conditional, reduction, symmetricity.

1 Introduction

In modern cryptography, secure *multi-party computation* (MPC) has been actively studied, where two or more mutually distrusting parties share computational tasks. *Oblivious transfer* is one of the most interesting primitives which can construct a secure MPC protocol [5][4]. However, it is known that OT cannot be achieved from scratch. Therefore various assumptions are proposed, such as computational assumptions, cryptographic tools, weaker variation of OT, or *inversion* of itself.

OT is first introduced by Rabin [5], which involves two parties, the sender and the receiver. The sender sends a bit to the receiver and the receiver obtains it with probability $1/2$, however the sender never learns about the output to the receiver. OT was developed to various types, such as *chosen 1-out-of-2 OT* ($\binom{2}{1}$-OT) [3] and *strong conditional OT* (SCOT) [1]. In $\binom{2}{1}$-OT, the sender sends two bits b_0 and b_1 and the receiver obtains exactly one of them b_c dependently on receiver's choosing bit c. c is kept secret from the sender, and b_{1-c} from the receiver. SCOT is similar to $\binom{2}{1}$-OT, except the point that the sender also inputs the choosing bit. SCOT requires a predicate Q, since the output to the receiver depends on the resulting value $c = Q(x, y)$, where x is the choosing bit of the sender and y of the receiver. The sender obtains no information about y, and the receiver does about x and b_{1-c}.

In [2], Crépeau and Santha proposed a reduction obtaining $\binom{2}{1}$-OT from S to R invoking some instances of $\binom{2}{1}$-OT from R to S. Their reduction needs n instances to achieve $2^{\Theta(n)}$ security. They raised the question whether it is possible to implement oblivious transfer in one direction using fewer instance of oblivious transfer in the other. Wolf and Wullschleger found an affirmative answer in [6]. They also proposed $\binom{2}{1}$-OT via only one instance of the inversion of $\binom{2}{1}$-OT. The reduction protocol achieves equivalent security to invoked $\binom{2}{1}$-OT protocol. Their construction bases on the fact that $\binom{2}{1}$-OT has information-theoretical symmetry.

Contribution. In this paper, we construct the special cases of SCOT from only one instance of their inversion. We assume XOR-, AND- and OR-OT, which are SCOT for predicates XOR, AND and OR, respectively, and whose inputs are restricted to a bit. Our reduction is simple, efficient and tight. We can apply the reduction protocol to a SCOT protocol with arbitrary security, perfect, statistical and perfect, and the resulting protocol achieves the equivalent security to invoked SCOT protocol.

2 Preliminaries

We use the standard notations and conventions for writing probabilistic algorithms and experiments. Let a and b be bits. Then $a \oplus b$, ab and $a + b$ denotes the bit obtained as the bitwise logical XOR, AND and OR of a and b, respectively. We say a function $f : \mathbb{N} \to \mathbb{R}$ is *negligible in n* if for every positive polynomial p there exists an N, such that for all $n > N$, $f(n) < 1/p(n)$. Let $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ be distribution ensembles. We say $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are *computationally indistinguishable* if, for any distinguisher algorithm D, $|\Pr_D(X_n) - \Pr_D(Y_n)| < \epsilon(n)$ is negligible in n where $\Pr_D(X_n)$ is the probability that D accepts x chosen according to the distribution X_n. We call $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are *statistically indistinguishable* if $\sum_\alpha |\Pr[X_n = \alpha] - \Pr[Y_n = \alpha]|$ is negligible. We call $\{X_n\}_{n\in\mathbb{N}}$ and $\{Y_n\}_{n\in\mathbb{N}}$ are *perfectly indistinguishable* if for all $\alpha \in \mathbb{N}$, $\Pr[X_n = \alpha] = \Pr[Y_n = \alpha]$. $X \overset{*}{\equiv} Y$ denotes that distributions X and Y are indistinguishable, where $*$ is p, s or c if that indistinguishability is perfect, statistical or computational, respectively.

An algorithm is a Turing machine. An *efficient* algorithm is an algorithm running in probabilistic polynomial time. An interactive Turing machine is a probabilistic algorithm with an additional communication tape. A set of interactive Turing machines is a *protocol*. Let $A = (A_1, A_2)$ and $\overline{A} = (\overline{A_1}, \overline{A_2})$ be the pairs of algorithm. Then we say $\overline{A}$ is *admissible for* A if $\overline{A_1} = A_1$ or $\overline{A_2} = A_2$ holds.

In an algorithm we use the following notations. If A is a probabilistic algorithm, then $y \leftarrow A(x_1, x_2, \ldots,)$ is the probabilistic experiment of obtaining y by running A on inputs $(x_1, x_2, \ldots)$, where the probability space is given by the random coins of algorithm A. If S is a finite set, then $x \leftarrow S$ is the operation of picking an element uniformly from S. If $\Pi = (P_1, \ldots, P_n)$ is a protocol, then $y \leftarrow \Pi_{P_i}(x_1, \ldots, x_n)$ denotes running a protocol with inputs $x_1, \ldots, x_n$ and receiving y resulting output of participant P_i. If α is neither an algorithm nor a set nor a protocol, then $x \leftarrow \alpha$ is a simple assignment statement. Let $\perp$ be a special symbol which indicates that an algorithm outputs nothing.

3 Definition

3.1 Security

We employ the security model of protocol [6], *the real* and *the ideal model*. The ideal model means the situation in which every player can access a *functionality* by defined inputs and outputs, and have no other way of communication each other. Since functionality always computes and outputs correctly with inputs, any cheating is impossible. However, there does not exist such a perfect black box practically. Hence it should be enabled by a protocol, namely the real model. In the real model, a player can cheat by not following the protocol. We say a protocol *securely computes* a functionality, if for any player in the real model, there exists a player in the ideal model which has indistinguishable outputs from the real player with the same information. Common input z represents some additional auxiliary input for both parties. An honest party does not need it, but a malicious party can use it, for instance, to record and carry down information about previous executions of the protocol. We assume at least one of the parties is honest.

Let $\mathcal{Z}$ be a domain of a common input z. We formalize the secure implementation of $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{U} \times \mathcal{V}$ as follows.

The Real Model.

In the *real* model, f has to be computed by a protocol $\Pi = (A_1, A_2)$ without any help of a trusted party. Let $\overline{A} = (\overline{A_1}, \overline{A_2})$ be an admissible pair for Π. Then the *joint execution of* Π *under* $\overline{A}$ *in the real model*,

$$\mathrm{real}_{\Pi,\overline{A}(z)}(x, y),$$

is the resulting output pair, given the inputs $x \in \mathcal{X}, y \in \mathcal{Y}$ and auxiliary input $z \in \mathcal{Z}$. $\mathrm{real}_{\Pi,\overline{A}(z)}(x, y)$ is also a random variable with the coin of $\overline{A}$.

The Ideal Model.

In the *ideal* model, the two parties can make use of a trusted party to calculate the function. The algorithms $\overline{B_1}$ and $\overline{B_2}$ of the protocol $\overline{B} = (\overline{B_1}, \overline{B_2})$ receive the inputs x and y, respectively, and auxiliary input z. They send values x' and y' to the trusted party, who sends back the values u' and v' satisfying $(u', v') = f(x', y')$. Finally, $\overline{B_1}$ and $\overline{B_2}$

output the value u and v. The two *honest* algorithms B_1 and B_2 always send $x' = x$ and $y' = y$ to the trusted party, and always output $u = u'$ and $v = v'$. Now, if $\overline{B} = (\overline{B_1}, \overline{B_2})$ is admissible pair of algorithm for protocol $B = (B_1, B_2)$, the *joint execution of f under $\overline{B}$ in the ideal model,*

$$\text{ideal}_{f,\overline{B}(z)}(x, y),$$

is the resulting output pair, given the inputs $x \in \mathcal{X}, y \in \mathcal{Y}$ and auxiliary input $z \in \mathcal{Z}$. $\text{ideal}_{f,\overline{B}(z)}(x, y)$ is also a random variable with the coin of $\overline{B}$.

Security: Real ≡ Ideal.

A protocol Π *computes* a functionality f *computationally* (or *statistically, perfectly*) securely if for any real adversary, there exists an ideal adversary which is as efficient as, and has computationally (or statistically, perfectly) indistinguishable output from the real.

This formulation can also apply to *reduction* protocol of functionality f to functionality g, such that the "real" participants access a protocol securely computing g.

Definition 1. *A protocol Π computes f computationally (or statistically, perfectly) securely if for every admissible $\overline{A} = (\overline{A_1}, \overline{A_2})$ there exists an admissible $\overline{B} = (\overline{B_1}, \overline{B_2})$, as efficient as, and with exactly the same set of honest players as $\overline{A}$, such that for all $x \in \mathcal{X}, y \in \mathcal{Y}$ and $z \in \mathcal{Z}$,*

$$\text{real}_{\Pi,\overline{A}(z)}(x, y) \equiv \text{ideal}_{f,\overline{B}(z)}(x, y)$$

holds, where $\equiv$ means that the distributions are computationally (or statistically, perfectly) indistinguishable.

Definition 2. *A protocol Π securely reduces f to a functionality g if Π computationally (or statistically, perfectly) securely computes f where every player in the real model have access to the protocol which computationally (or statistically, perfectly) securely computes g.*

3.2 Strong Conditional Oblivious Transfer

We formalize SCOT as a functionality.

Definition 3 (Q-SCOT). *By Q-SCOT or strong conditional oblivious transfer for predicate Q, we denote the following primitive between a sender S and a receiver R (see Figure 1). S has three inputs x, m_0 and m_1 and no output, so does R with input y and output u such that $u = m_{Q(x,y)}$.*

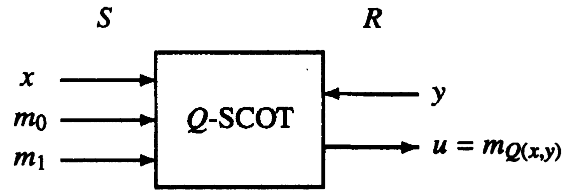

Figure 1: Strong conditional oblivious transfer

In this paper, we consider three predicates, XOR, AND and OR. We define XOR-OT, AND-OT and OR-OT as a special case of Q-SCOT where all inputs are 1-bit. In other words, $(x, m_0, m_1) \in \{0, 1\}^3$ and $y \in \{0, 1\}$ and each output of XOR-OT, AND-OT and OR-OT is as follows:

XOR-OT: $m_{x \oplus y} = m_0(1 \oplus x \oplus y) \oplus m_1(x \oplus y)$,

AND-OT: $m_{xy} = m_0(1 \oplus xy) \oplus m_1(xy)$,

OR-OT: $m_{x+y} = m_0(1 \oplus x)(1 \oplus y) \oplus m_1(1 \oplus (1 \oplus x)(1 \oplus y))$.

4 Construction

In this section, we construct three restricted SCOT protocols, XOR-OT, AND-OT and OR-OT via protocols which securely compute inversions of themselves. An inversion means the replacement of player's role, i.e., S can access the inversion as R, and vice versa. Let Q-TO denote a protocol which securely computes the inversion of a functionality Q-OT, where Q = {XOR, AND, OR}. We use the notion $\overset{*}{\equiv}$ in our security proofs. $X \overset{*}{\equiv} Y$ denotes that distributions X and Y are indistinguishable, where $*$ is p, s or c if invoked Q-TO is perfect, statistical or computational secure, respectively.

4.1 XOR-OT

We present a simple protocol, INV-XOR = (S, R), securely computing XOR-OT via an inversion protocol of XOR-OT, namely XOR-TO. The protocol is defined as Figure 2.

Theorem 4. INV-XOR *securely computes* XOR-OT *reducing to one instance of* XOR-TO.

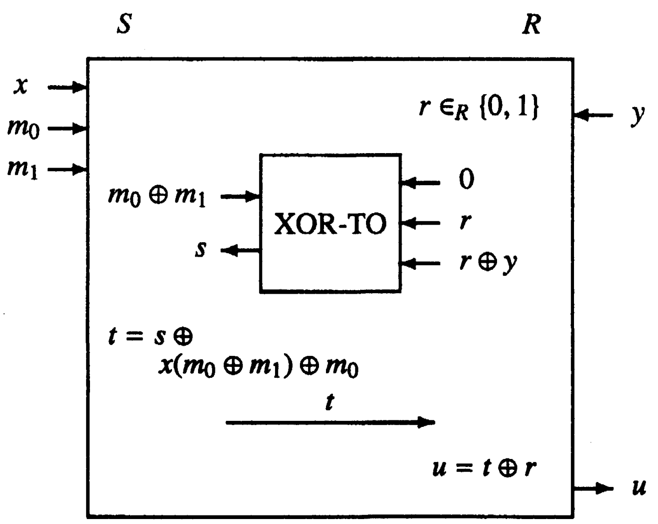

Figure 2: INV-XOR

Proof. We have to consider three cases, both parties are honest, the sender is honest, and the receiver is honest.

Both parties are honest. Let first both parties be honest, i.e., $\overline{A}$ = INV-XOR. In this case, we define the adversary $\overline{B}$ in the ideal model as B. For all $(x, m_0, m_1) \in \{0,1\}^3, y \in \{0,1\}$ and $z \in \mathcal{Z}$, since we have $s \stackrel{*}{\equiv} r \oplus y(m_0 \oplus m_1)$,

$$
\begin{aligned}
&\mathrm{real}_{\text{XOR-OT},\overline{A}(z)}((x, m_0, m_1), y)\\
&\quad = (\perp, u)\\
&\quad = (\perp, t \oplus r)\\
&\quad = (\perp, s \oplus x(m_0 \oplus m_1) \oplus m_0 \oplus r)\\
&\quad \stackrel{*}{\equiv} (\perp, (x \oplus y)(m_0 \oplus m_1) \oplus m_0)\\
&\quad = (\perp, (1 \oplus x \oplus y)m_0 \oplus (x \oplus y)m_1)\\
&\quad = (\perp, m_{x \oplus y})\\
&\quad = \mathrm{ideal}_{\text{XOR-OT},\overline{B}(z)}((x, m_0, m_1), y).
\end{aligned}
$$

The first party is honest. Let now the first party be honest, i.e., $\overline{A} = (S, \overline{A_2})$. To describe the cheating method of $\overline{A_2}$, we divide $\overline{A_2}$ into two algorithms $\overline{A_{21}}$ and $\overline{A_{22}}$. $\overline{A_{21}}$ receives (y, z) and sends $((a, b_0, b_1), i) = \overline{A_{21}}(y, z)$ to XOR-TO and $\overline{A_{22}}$, where (a, b_0, b_1) is an input to XOR-TO and i is status information. Then $\overline{A_{22}}$ receives $t = b_{(m_0 \oplus m_1) \oplus a} \oplus x(m_0 \oplus m_1) \oplus m_0$, and outputs $\overline{A_{22}}(y, z, a, b_0, b_1, t, i)$. For any $\overline{A_2}$, we define the ideal adversary $\overline{B_2}$ as follows:

$$
\begin{aligned}
&\texttt{Algorithm } \overline{B_2}(y, z)\\
&\quad ((a', b'_0, b'_1), i') \leftarrow \overline{A_{21}}(y, z)\\
&\quad u \leftarrow \text{XOR-OT}_R(b'_0 \oplus b'_1)\\
&\quad t' \leftarrow u \oplus (1 \oplus a')b'_0 \oplus a'b'_1\\
&\quad \texttt{output } \overline{A_{22}}(y, z, a', b'_0, b'_1, t', i').
\end{aligned}
$$

Note that $((a', b'_0, b'_1), i')$ is identically distributed with $((a, b_0, b_1), i)$ and

$$
\begin{aligned}
t' &= u \oplus (1 \oplus a')b'_0 \oplus a'b'_1\\
&= m_{x \oplus (b'_0 \oplus b'_1)} \oplus (1 \oplus a')b'_0 \oplus a'b'_1\\
&= m_0(1 \oplus x \oplus (b'_0 \oplus b'_1)) \oplus m_1(x \oplus (b'_0 \oplus b'_1))\\
&\qquad \oplus (1 \oplus a')b'_0 \oplus a'b'_1\\
&= b'_0(1 \oplus m_0 \oplus m_1 \oplus a') \oplus b'_1(m_0 \oplus m_1 \oplus a')\\
&\qquad \oplus x(m_0 \oplus m_1) \oplus m_0\\
&= b'_{(m_0 \oplus m_1) \oplus a'} \oplus x(m_0 \oplus m_1) \oplus m_0.
\end{aligned}
$$

Thus we have, for all $(x, m_0, m_1) \in \{0,1\}^3, y \in \{0,1\}$ and $z \in \mathcal{Z}$,

$$
\begin{aligned}
&\mathrm{real}_{\text{XOR-OT},\overline{A}(z)}((x, m_0, m_1), y)\\
&\quad = (\perp, \overline{A_{22}}(y, z, a, b_0, b_1, t, i))\\
&\quad \stackrel{*}{\equiv} (\perp, \overline{A_{22}}(y, z, a, b_0, b_1,\\
&\qquad\qquad b_{(m_0 \oplus m_1) \oplus a} \oplus x(m_0 \oplus m_1) \oplus m_0, i))\\
&\quad \stackrel{p}{\equiv} (\perp, \overline{A_{22}}(y, z, a', b'_0, b'_1,\\
&\qquad\qquad b'_{(m_0 \oplus m_1) \oplus a'} \oplus x(m_0 \oplus m_1) \oplus m_0, i'))\\
&\quad = (\perp, \overline{A_{22}}(y, z, a', b'_0, b'_1, t', i'))\\
&\quad = \mathrm{ideal}_{\text{XOR-OT},\overline{B}(z)}((x, m_0, m_1), y).
\end{aligned}
$$

The second party is honest. Assume now that the second party is honest, i.e., $\overline{A} = (\overline{A_1}, R)$. Similarly to the previous case, we divide $\overline{A_1}$ into three algorithms, $\overline{A_{11}}, \overline{A_{12}}$ and $\overline{A_{13}}$. $\overline{A_{11}}$ receives (x, m_0, m_1) and sends $(c, j) = \overline{A_{11}}(x, m_0, m_1)$ to XOR-TO and $\overline{A_{12}}$, where c is an input to XOR-TO which returns $s = r \oplus yc$ and j is a status information. Then $\overline{A_{12}}$ sends $t = \overline{A_{12}}((x, m_0, m_1), z, c, s, j)$ to R and outputs $\overline{A_{13}}((x, m_0, m_1), z, c, s, j, t)$. For any real adversary $\overline{A_1}$, we define the ideal adversary $\overline{B_1}$ as follows:

$$
\begin{aligned}
&\texttt{Algorithm } \overline{B_1}((x, m_0, m_1), z) \\
&\quad (c', j') \leftarrow \overline{A_{11}}((x, m_0, m_1), z) \\
&\quad s' \leftarrow_R \{0, 1\} \\
&\quad t' \leftarrow \overline{A_{12}}((x, m_0, m_1), z, c', s', j') \\
&\quad \perp \leftarrow \text{XOR-OT}_S(0, s' \oplus t', s' \oplus t' \oplus c') \\
&\quad \texttt{output } \overline{A_{13}}((x, m_0, m_1), z, c', s', j', t').
\end{aligned}
$$

For the sender's inputs $(0, s' \oplus t', s' \oplus t' \oplus c')$ and the receiver's input y, the output of XOR-OT to the receiver u is as follows:

$$
\begin{aligned}
u &= (s' \oplus t')(1 \oplus 0 \oplus y) \oplus (s' \oplus t' \oplus c')(0 \oplus y) \\
&= t' \oplus s' \oplus yc'.
\end{aligned}
$$

(c', j') has the identical distribution with (c, j). Since $s = r \oplus yc$ and r is uniform and independent of all other variables, s is uniform and independent as well, which means that it has exactly the same distribution as s'. Therefore t' is identically distributed with t. We have from above, for all $(x, m_0, m_1) \in \{0, 1\}^3, y \in \{0, 1\}$ and $z \in \mathcal{Z}$,

$$
\begin{aligned}
&\text{real}_{\text{XOR-OT}, \overline{A}(z)}((x, m_0, m_1), y) \\
&\quad \overset{*}{\equiv} (\overline{A_{13}}((x, m_0, m_1), z, c, s, j, t), t \oplus r) \\
&\quad = (\overline{A_{13}}((x, m_0, m_1), z, c, s, j, t), t \oplus s \oplus yc) \\
&\quad \overset{p}{\equiv} (\overline{A_{13}}((x, m_0, m_1), z, c', s', j', t'), t' \oplus s' \oplus yc') \\
&\quad = \text{ideal}_{\text{XOR-OT}, \overline{B}(z)}((x, m_0, m_1), y).
\end{aligned}
$$

Obliviously, the simulated adversaries are as efficient as the real adversary. □

4.2 AND-OT

We present a simple protocol, INV-AND $= (S, R)$, securely computing AND-OT via an inversion protocol of AND-OT, namely AND-TO. The protocol is defined as Figure 3.

Theorem 5. INV-AND *securely computes* AND-OT *reducing to one instance of* AND-TO.

Proof. As in the case of XOR-OT, we have to consider three cases, both parties are honest, the sender is honest and the receiver is honest.

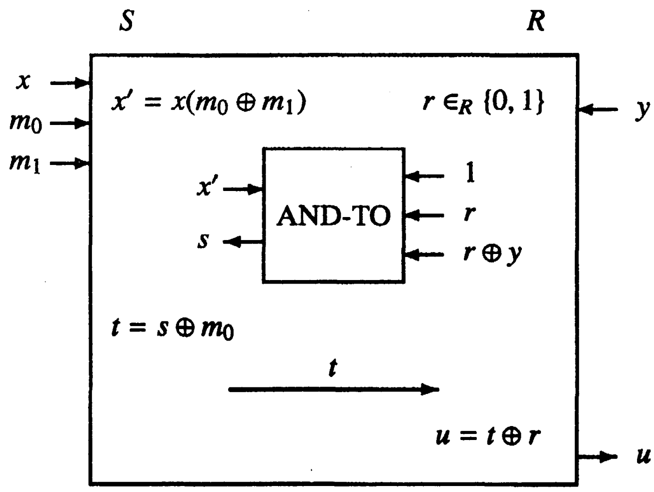

Figure 3: INV-AND

Both parties are honest. Let first both parties be honest, i.e., $\overline{A}$ = INV-AND. In this case, we define the adversary $\overline{B}$ in the ideal model as B. For all $(x, m_0, m_1) \in \{0, 1\}^3, y \in \{0, 1\}$ and $z \in \mathcal{Z}$, since we have $s \overset{*}{\equiv} r \oplus xy(m_0 \oplus m_1)$,

$$
\begin{aligned}
&\text{real}_{\text{AND-OT}, \overline{A}(z)}((x, m_0, m_1), y) \\
&\quad = (\perp, u) \\
&\quad = (\perp, t \oplus r) \\
&\quad = (\perp, s \oplus m_0 \oplus r) \\
&\quad \overset{*}{\equiv} (\perp, xy(m_0 \oplus m_1) \oplus m_0) \\
&\quad = (\perp, (1 \oplus xy)m_0 \oplus xym_1) \\
&\quad = (\perp, m_{xy}) \\
&\quad = \text{ideal}_{\text{AND-OT}, \overline{B}(z)}((x, m_0, m_1), y).
\end{aligned}
$$

The first party is honest. Let now the first party be honest, i.e., $\overline{A} = (S, \overline{A_2})$. To describe the cheating method of $\overline{A_2}$, we divide $\overline{A_2}$ into two algorithms, $\overline{A_{21}}$ and $\overline{A_{22}}$. $\overline{A_{21}}$ receives (y, z) and sends $((a, b_0, b_1), i) = \overline{A_{21}}(y, z)$ to AND-TO and $\overline{A_{22}}$, where (a, b_0, b_1) is an input to AND-TO and i is status information. Then $\overline{A_{22}}$ receives $t = b_{xa(m_0 \oplus m_1)} \oplus m_0$, and outputs $\overline{A_{22}}(y, z, a, b_0, b_1, t, i)$. For any $\overline{A_2}$, we define the ideal adversary $\overline{B_2}$ as follows:

Algorithm $\overline{B_2}(y,z)$

$((a',b'_0,b'_1),i') \leftarrow \overline{A_{21}}(y,z)$

$u \leftarrow \text{AND-OT}_R(a(b'_0 \oplus b'_1))$

$t' \leftarrow u \oplus b'_0$

output $\overline{A_{22}}(y,z,a',b'_0,b'_1,t',i')$.

Note that $((a',b'_0,b'_1),i')$ is identically distributed with $((a,b_0,b_1),i)$ and

$$\begin{aligned} t' &= u \oplus b'_0 \\ &= m_{xa'(b'_0 \oplus b'_1)} \oplus b'_0 \\ &= m_0(1 \oplus xa'(b'_0 \oplus b'_1)) \oplus m_1(xa'(b'_0 \oplus b'_1)) \oplus b'_0 \\ &= b'_0(1 \oplus xa'(m_0 \oplus m_1)) \oplus b'_1(xa'(m_0 \oplus m_1)) \oplus m_0 \\ &= b'_{xa'(m_0 \oplus m_1)} \oplus m_0. \end{aligned}$$

Thus we have, for all $(x,m_0,m_1) \in \{0,1\}^3, y \in \{0,1\}$ and $z \in \mathcal{Z}$,

$$\begin{aligned} &\text{real}_{\text{AND-OT},\overline{A}(z)}((x,m_0,m_1),y) \\ &\quad = (\bot, \overline{A_{22}}(y,z,a,b_0,b_1,t,i)) \\ &\quad \overset{*}{\equiv} (\bot, \overline{A_{22}}(y,z,a,b_0,b_1, \\ &\qquad\qquad b_{xa(m_0 \oplus m_1)} \oplus m_0, i)) \\ &\quad \overset{p}{\equiv} (\bot, \overline{A_{22}}(y,z,a',b'_0,b'_1, \\ &\qquad\qquad b'_{xa'(m_0 \oplus m_1)} \oplus m_0, i')) \\ &\quad = (\bot, \overline{A_{22}}(y,z,a',b'_0,b'_1,t',i')) \\ &\quad = \text{ideal}_{\text{AND-OT},\overline{B}(z)}((x,m_0,m_1),y). \end{aligned}$$

The second party is honest. Assume now that the second party is honest, i.e., $\overline{A} = (\overline{A_1}, R)$. Similarly to the previous case, we divide $\overline{A_1}$ into three algorithms, $\overline{A_{11}}, \overline{A_{12}}$ and $\overline{A_{13}}$. $\overline{A_{11}}$ receives (x,m_0,m_1) and sends $(c,j) = \overline{A_{11}}(x,m_0,m_1)$ to AND-TO and $\overline{A_{12}}$, where c is an input to AND-TO which returns $s = r \oplus \underline{y}(1 \oplus c)$ and $\underline{j}$ is status information. Then $\overline{A_{12}}$ sends $\underline{t} = \overline{A_{12}}((x,m_0,m_1),z,c,s,j)$ to R and outputs $\overline{A_{13}}((x,m_0,m_1),z,c,s,j,t)$. For any real adversary $\overline{A_1}$, we define the ideal adversary $\overline{B_1}$ as follows:

Algorithm $\overline{B_1}((x,m_0,m_1),z)$

$(c',j') \leftarrow \overline{A_{11}}((x,m_0,m_1),z)$

$s' \leftarrow_R \{0,1\}$

$t' \leftarrow \overline{A_{12}}((x,m_0,m_1),z,c',s',j')$

$\bot \leftarrow \text{AND-OT}_S(1, s' \oplus t', s' \oplus t' \oplus 1 \oplus c')$

output $\overline{A_{13}}((x,m_0,m_1),z,c',s',j',t')$.

For the sender's inputs $(0, s' \oplus t', s' \oplus t' \oplus c')$ and the receiver's input y, the output of AND-OT to the receiver u is as follows:

$$\begin{aligned} u &= (s' \oplus t')(1 \oplus y) \oplus (s' \oplus t' \oplus 1 \oplus c')y \\ &= t' \oplus s' \oplus y(1 \oplus c'). \end{aligned}$$

(c',j') has the identical distribution with (c,j). Since $s = r \oplus y(1 \oplus c)$ and r is uniform and independent of all other variables, s is uniform and independent as well, which means that it has exactly the same distribution as s'. Therefore t' is identically distributed with t. We have from above, for all $(x,m_0,m_1) \in \{0,1\}^3, y \in \{0,1\}$ and $z \in \mathcal{Z}$,

$$\begin{aligned} &\text{real}_{\text{XOR-OT},\overline{A}(z)}((x,m_0,m_1),y) \\ &\quad \overset{*}{\equiv} (\overline{A_{13}}((x,m_0,m_1),z,c,s,j,t), t \oplus r) \\ &\quad = (\overline{A_{13}}((x,m_0,m_1),z,c,s,j,t), \\ &\qquad\qquad t \oplus s \oplus y(1 \oplus c)) \\ &\quad \overset{p}{\equiv} (\overline{A_{13}}((x,m_0,m_1),z,c',s',j',t'), \\ &\qquad\qquad t' \oplus s' \oplus y(1 \oplus c')) \\ &\quad = \text{ideal}_{\text{XOR-OT},\overline{B}(z)}((x,m_0,m_1),y). \end{aligned}$$

The second "≡" means perfect indistinguishable, hence the bottleneck of this evaluation is the first "≡" deriving from the security of the AND-TO protocol.

Obliviously, the simulated adversaries are as efficient as the real adversary. □

4.3 OR-OT

We present a simple protocol, INV-OR $= (S,R)$, securely computing OR-OT via an inversion protocol of OR-OT, namely OR-TO. The protocol is defined as Figure 4.

Theorem 6. INV-OR *securely computes* OR-OT *reducing to one instance of* OR-TO.

Proof. We have to consider three cases, both parties are honest, the sender is honest and the receiver is honest.

Both parties are honest. Let first both parties be honest, i.e., $\overline{A}$ = INV-OR. In this case, we define the adversary $\overline{B}$ in the ideal model as B. For all $(x,m_0,m_1) \in \{0,1\}^3, y \in \{0,1\}$ and $z \in \mathcal{Z}$,

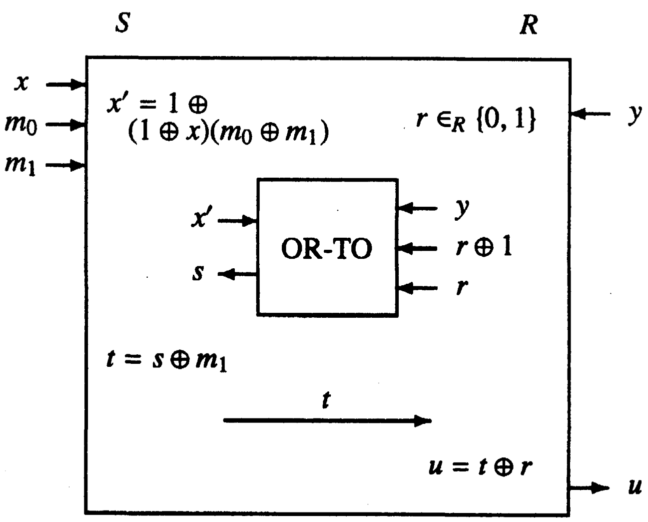

Figure 4: INV-OR

since we have $s \stackrel{*}{\equiv} r \oplus (1 \oplus x)(1 \oplus y)(m_0 \oplus m_1)$,

$$
\begin{aligned}
&\mathrm{real}_{\text{XOR-OT},\overline{A}(z)}((x, m_0, m_1), y)\\
&\quad = (\perp, u)\\
&\quad = (\perp, t \oplus r)\\
&\quad = (\perp, s \oplus m_1 \oplus r)\\
&\quad \stackrel{*}{\equiv} (\perp, (1 \oplus x)(1 \oplus y)(m_0 \oplus m_1) \oplus m_1)\\
&\quad = (\perp, (1 \oplus x)(1 \oplus y)m_0 \oplus (1 \oplus (1 \oplus x)(1 \oplus y))m_1)\\
&\quad = (\perp, m_{1 \oplus (1 \oplus x)(1 \oplus y)})\\
&\quad = (\perp, m_{x+y})\\
&\quad = \mathrm{ideal}_{\text{XOR-OT},\overline{B}(z)}((x, m_0, m_1), y).
\end{aligned}
$$

The first party is honest. Let then the first party be honest, i.e., $\overline{A} = (S, \overline{A_2})$. To describe the cheating method of $\overline{A_2}$, we divide $\overline{A_2}$ into two algorithms, $\overline{A_{21}}$ and $\overline{A_{22}}$. $\overline{A_{21}}$ receives (y, z) and sends $((a, b_0, b_1), i) = \overline{A_{21}}(y, z)$ to OR-TO and $\overline{A_{22}}$, where (a, b_0, b_1) is an input to OR-TO and i is status information. Then $\overline{A_{22}}$ receives $t = b_{1 \oplus (1 \oplus x)(1 \oplus a)(m_0 \oplus m_1)} \oplus m_1$ and outputs $\overline{A_{22}}(y, z, a, b_0, b_1, t, i)$. For any $\overline{A_2}$, we define the ideal adversary $\overline{B_2}$ as follows:

Algorithm $\overline{B_2}(y, z)$
  $((a', b'_0, b'_1), i') \leftarrow \overline{A_{21}}(y, z)$
  $u \leftarrow \text{OR-OT}_R(1 \oplus (1 \oplus a')(b'_0 \oplus b'_1))$
  $t' \leftarrow u \oplus b'_1$
  output $\overline{A_{22}}(y, z, a', b'_0, b'_1, t', i')$.

Note that $((a', b'_0, b'_1), i)$ is identically distributed with $((a, b_0, b_1), i)$ and

$$
\begin{aligned}
t' &= u \oplus b'_1\\
&= m_{1 \oplus (1 \oplus x)(1 \oplus a')(b'_0 \oplus b'_1)} \oplus b'_1\\
&= m_0(1 \oplus x)(1 \oplus a')(b'_0 \oplus b'_1)\\
&\qquad \oplus m_1(1 \oplus (1 \oplus x)(1 \oplus a')(b'_0 \oplus b'_1)) \oplus b'_1\\
&= b'_0(1 \oplus x)(1 \oplus a')(m_0 \oplus m_1)\\
&\qquad \oplus b'_1(1 \oplus (1 \oplus x)(1 \oplus y)(m_0 \oplus m_1)) \oplus m_1\\
&= b'_{1 \oplus (1 \oplus x)(1 \oplus a')(m_0 \oplus m_1)} \oplus m_1.
\end{aligned}
$$

Thus we have, for all $(x, m_0, m_1) \in \{0, 1\}^3, y \in \{0, 1\}$ and $z \in \mathcal{Z}$,

$$
\begin{aligned}
&\mathrm{real}_{\text{OR-OT},\overline{A}(z)}((x, m_0, m_1), y)\\
&\quad = (\perp, \overline{A_{22}}(y, z, a, b_0, b_1, t, i))\\
&\quad \stackrel{*}{\equiv} (\perp, \overline{A_{22}}(y, z, a, b_0, b_1,\\
&\qquad\qquad b_{1 \oplus (1 \oplus x)(1 \oplus a)(m_0 \oplus m_1)} \oplus m_1, i))\\
&\quad \stackrel{p}{\equiv} (\perp, \overline{A_{22}}(y, z, a', b'_0, b'_1,\\
&\qquad\qquad b'_{1 \oplus (1 \oplus x)(1 \oplus a')(m_0 \oplus m_1)} \oplus m_1, i'))\\
&\quad = (\perp, \overline{A_{22}}(y, z, a', b'_0, b'_1, t', i'))\\
&\quad = \mathrm{ideal}_{\text{OR-OT},\overline{B}(z)}((x, m_0, m_1), y).
\end{aligned}
$$

The second party is honest. Assume now that the second party is honest, i.e., $\overline{A} = (\overline{A_1}, R)$. Similarly to the previous case, we divide $\overline{A_1}$ into three algorithms, $\overline{A_{11}}, \overline{A_{12}}$ and $\overline{A_{13}}$. $\overline{A_{11}}$ receives (x, m_0, m_1) and sends $(c, j) = \overline{A_{11}}(x, m_0, m_1)$ to OR-TO and $\overline{A_{12}}$, where c is an input to OR-TO which returns $s = r \oplus (1 \oplus y)(1 \oplus c)$ and j is status information. Then $\overline{A_{12}}$ sends $t = \overline{A_{12}}((x, m_0, m_1), z, c, s, j)$ to R and outputs $\overline{A_{13}}((x, m_0, m_1), z, c, s, j, t)$. For any real adversary $\overline{A_1}$, we define the ideal adversary $\overline{B_1}$ as follows:

Algorithm $\overline{B_1}((x, m_0, m_1), z)$
  $(c', j') \leftarrow \overline{A_{11}}((x, m_0, m_1), z)$
  $s' \leftarrow_R \{0, 1\}$
  $t' \leftarrow \overline{A_{12}}((x, m_0, m_1), z, c', s', j')$
  $\perp \leftarrow \text{OR-OT}_S(0, s' \oplus t', s' \oplus t' \oplus c')$
  output $\overline{A_{13}}((x, m_0, m_1), z, c', s', j', t')$.

For the sender's inputs $(c', s' \oplus t' \oplus 1, s' \oplus t')$ and the receiver's input y, the output of OR-OT to the receiver u is as follows:

$$u = (s' \oplus t' \oplus 1)(1 \oplus c')(1 \oplus y) \oplus (s' \oplus t')(1 \oplus (1 \oplus c')(1 \oplus y)) = t' \oplus s' \oplus (1 \oplus y)(1 \oplus c').$$

(c', j') has the identical distribution with (c, j). Since $s = r \oplus (1 \oplus y)(1 \oplus c)$ and r is uniform and independent of all other variables, s is uniform and independent as well, which means that it has exactly the same distribution as s'. Therefore t' is identically distributed with t. We have from above, for all $(x, m_0, m_1) \in \{0,1\}^3, y \in \{0,1\}$ and $z \in \mathcal{Z}$,

$$\begin{aligned}
&\mathrm{real}_{\text{OR-OT},\overline{A}(z)}((x, m_0, m_1), y) \\
&\quad \stackrel{*}{\equiv} (\overline{A_{13}}((x, m_0, m_1), z, c, s, j, t), t \oplus r) \\
&\quad = (\overline{A_{13}}((x, m_0, m_1), z, c, s, j, t), \\
&\qquad\qquad t \oplus s \oplus (1 \oplus y)(1 \oplus c)) \\
&\quad \stackrel{p}{\equiv} (\overline{A_{13}}((x, m_0, m_1), z, c', s', j', t'), \\
&\qquad\qquad t' \oplus s' \oplus (1 \oplus y)(1 \oplus c')) \\
&\quad = \mathrm{ideal}_{\text{OR-OT},\overline{B}(z)}((x, m_0, m_1), y).
\end{aligned}$$

The second "≡" means perfect indistinguishable, hence the bottleneck of this evaluation is the first "≡" deriving from the security of the OR-TO protocol.

Obliviously, the simulated adversaries are as efficient as the real adversary. □

References

[1] Blake, I. F., and Kolesnikov, V. Strong conditional oblivious transfer and computing on intervals. In *ASIACRYPT* (2004), P. J. Lee, Ed., vol. 3329 of *Lecture Notes in Computer Science*, Springer, pp. 515–529.

[2] Crépeau, C., and Santha, M. On the reversibility of oblivious transfer. In *EUROCRYPT* (1991), pp. 106–113.

[3] Even, S., Goldreich, O., and Lempel, A. A randomized protocol for signing contracts. *Commun. ACM 28*, 6 (1985), 637–647.

[4] Goldreich, O., Micali, S., and Wigderson, A. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC* (1987), ACM, pp. 218–229.

[5] O.Rabin, M. How to exchange secrets by oblivious transfer. Tech. Rep. TR-81, Aiken computation laboratory, Harvard University, 1981.

[6] Wolf, S., and Wullschleger, J. Oblivious transfer is symmetric. In *EUROCRYPT* (2006), S. Vaudenay, Ed., vol. 4004 of *Lecture Notes in Computer Science*, Springer, pp. 222–232.