# MOBILE AGENT SECURITY WITH EFFICIENT OBLIVIOUS TRANSFER

Wataru Hasegawa, Masakazu Soshi and Atsuko Miyaji
*School of Information Science, Japan Advanced Institute of Science and Technology*

Keywords: Mobile Agent, Security, Oblivious Transfer, Encrypted Circuit, Secure Function Evaluation.

Abstract: Cachin et al. and Algesheimer et al. proposed schemes using secure function evaluation for protecting mobile agents in untrusted environments. One of essential ingredients of their protocols is *oblivious transfer* (although not all of them require it). Unfortunately, naive application of oblivious transfer is inefficient because it must be performed for each bit of encrypted circuit inputs. Therefore, in this paper we propose secure mobile agent protocols with emphasis on efficient oblivious transfer suitable for secure function evaluation.

## 1 INTRODUCTION

Mobile agents are migratable autonomous software program and mobile agent technology has drawn much attention as a fundamental technology in next generation computing (Rothermel and Popescu-Zeletin, 1997). However, realization of mobile agents is confronted by a serious security problem: *an attack on mobile agents by malicious execution hosts* such as tampering or eavesdropping agents' secret during their execution. So the way of executing an 'encrypted' agent without decrypting it has been studied so far. Among such approaches, Cachin et al. (Cachin et al., 2000) and Algesheimer et al. (Algesheimer et al., 2001) proposed promising methods based on *secure function evaluation*. In particular, Algesheimer et al. introduced Trusted Third Party (TTP) to their protocols and succeeded in enhancing security of them.

One of essential ingredients of their protocols is *oblivious transfer* (although not all of them require it[1]). Unfortunately, from a viewpoint of communication cost, naive application of oblivious transfer is inefficient because it must be performed for each bit of encrypted circuit inputs. Hence Mori et al. (Mori et al., 2005) proposed a mobile agent security scheme using a new efficient oblivious transfer. However it turns out that their oblivious transfer protocol is insecure in a special situation (Hasegawa et al., 2007).

In this paper we propose two secure mobile agent protocols with emphasis on efficient oblivious transfer suitable for secure function evaluation. We show that one is secure in *honest-but-curios model* and the other is secure even in the *malicious model*. Furthermore, we shall show that our proposed oblivious transfer protocols are more efficient than a naive application of 1-out-of-2 oblivious transfer in mobile agent security schemes. Besides, our proposed oblivious transfer protocols are interesting in their own right and they are also important because they can be building blocks for other security protocols, especially, more sophisticated type of oblivious transfer, i.e., Naor's *k*-out-of-*n* oblivious transfer (Naor and Pinkas, 1999).

## 2 PRELIMINARY

### 2.1 Assumptions and Definitions

In this section, we present some preliminaries for our work. Let $\mathbb{G}_1$ be an additive group of a large prime order $p$ and $\mathbb{G}_2$ be a multiplicative group of the same

[1]For example, oblivious transfer is not needed in the basic scheme by Algesheimer et al. The scheme is discussed in Section 4.2.



Hasegawa W., Soshi M. and Miyaji A. (2007).
MOBILE AGENT SECURITY WITH EFFICIENT OBLIVIOUS TRANSFER.
In *Proceedings of the Second International Conference on Security and Cryptography*, pages 299-304
DOI: 10.5220/0002129002990304

order. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a function that satisfies the following properties: (1) Bilinearity: for any $P$, $Q \in \mathbb{G}_1$ and $a$, $b \in \mathbb{Z}$, $e(aP, bQ) = e(P,Q)^{ab}$; (2) Non-degeneration: $e(P,P) \neq 1$ where $P$ is a generator of $\mathbb{G}_1$.

Now we give the definition of NT-CDH problem below.

**Assumption 1.** New Target Computational Diffie-Hellman (NT-CDH) Problem

Let $P$ be a generator of $\mathbb{G}_1$, $s_0, s_1 \in_R \mathbb{Z}_p^*, P_0 = s_0P, P_1 = s_1P$, $b_i \in \{0,1\}$ for $i = 1, 2, \cdots, n$. Furthermore, let $T_{\mathbb{G}_1}(\cdot)$ be a *target oracle* that returns $Q_i \in \mathbb{G}_1$, and $\mathcal{H}_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ be a cryptographic hash function. The attacker $A$ is given $(p, P_0, P_1, \mathcal{H}_1, s_{b_1}Q_1, \ldots, s_{b_n}Q_n)$ and the access to $T_{\mathbb{G}_1}$. Then the advantage $Adv_G^{NT-CDH}(A)$ of $A$ in attacking NT-CDH problem is defined as the probability that $A$ outputs $s_bQ_j \notin \{s_{b_1}Q_1, \ldots, s_{b_n}Q_n\}$, where $1 \leq j \leq n$, and $b \in \{0,1\}$. There is no probabilistic polynomial-time adversary $A$ with non-negligible $Adv_G^{NT-CDH}(A)$.

Note that NT-CDH Problem is defined for the first time in this paper and we believe that it is reasonable to consider that the problem is computationally difficult to solve.

Finally, we define the attack model (Algesheimer et al., 2001; Cachin et al., 2000; Chu and Tzeng, 2005) supposed in this paper.

**Definition 1.** Attack Model

Attack models for a mobile agent security protocol are classified into two types: *honest-but-curious* (semi-honest) model and *malicious* model. Honest-but-curious hosts follow the protocol, but seek to steal some useful information about secrets of agents. On the other hand, malicious hosts can do whatever they want in order to obtain secret information.

## 2.2 Oblivious Transfer

One of key tools in security protocols is *oblivious transfer* (Naor and Pinkas, 1999), which often means 1-out-of-2 oblivious transfer. A (1-out-of-2) oblivious transfer is an interactive protocol between a sender (Alice) with two secret messages $m_0$ and $m_1$ and a receiver (Bob) with a bit $b$. By oblivious transfer, Bob gets $m_b$, but learns nothing about $m_{b\oplus 1}$. Furthermore, Alice does not learn anything about $b$.

More general form of oblivious transfer, namely, *k-out-of-n oblivious transfer* (Chu and Tzeng, 2005; Naor and Pinkas, 1999) is also useful. As the name implies, in $k$-out-of-$n$ oblivious transfer, Alice has $n$ secrets $m_1, m_2, \ldots, m_n$ and Bob has $k$ choices $i_1, \ldots, i_k$.

As we will see later, in mobile agent security schemes, we basically need to repeat 1-out-of-2 oblivious transfer $n$ times for some $n$. That is, Alice has $n$ pairs of secret messages $(m_{1,0}, m_{1,1})$, $(m_{2,0}, m_{2,1})$, $\ldots$, $(m_{n,0}, m_{n,1})$ and Bob has $n$ choices $(1, b_1)$, $(2, b_2)$, $\cdots$, $(n, b_n)$, where $b_i \in \{0,1\}$ $(1 \leq i \leq n)$. After completion of $n$ times 1-out-of-2 oblivious transfer, Bob receives $m_{1,b_1}, m_{2,b_2}, \cdots, m_{n,b_n}$. Hence an oblivious transfer scheme for mobile agent security must satisfy the following three requirements[2].

**Definition 2.** Correctness

An scheme is *correct* if the receiver $R$ (Bob) obtains the chosen messages when both of the sender $S$ (Alice) and $R$ do not deviate from the steps of the scheme.

**Definition 3.** The Receiver's privacy - indistinguishability

For any two choice sets of $R$, say, $C = \{(1,b_1), (2,b_2), \cdots, (n,b_n)\}$ and $C' = \{(1,b'_1), (2,b'_2), \cdots, (n,b'_n)\}$, the transcripts of the protocol execution corresponding to $C$ and $C'$, which $S$ sees, are indistinguishable. Furthermore, if the received messages of $S$ for $C$ and $C'$ are identically distributed, then the choices of $R$ are said to be *unconditionally secure*.

**Definition 4.** The Sender's privacy

This property is defined according to the type of the attack model.

- The Sender's privacy in the honest-but-curious model - indistinguishability:
  For any choices of $R$, the unchosen secret messages of $S$ are indistinguishable from random ones.
- The Sender's privacy in the malicious model - compared with the Ideal Model:
  In *the Ideal model*, first $S$ sends all secret messages to TTP[3]. Next $R$ sends his choices to TTP and then TTP sends the chosen secret messages of $S$ to $R$. The Ideal model, as its name implies, is the most secure scheme. We achieve the sender's privacy if for any $R$ in the real world, there exists another probabilistic polynomial-time Turing Machine (PPTM) $R^*$ (called *simulator*) in the Ideal model such that the outputs of $R$ and $R^*$ are indistinguishable.

## 2.3 Mobile Agent Computation based on Secure Function Evaluation

Secure function evaluation (Yao, 1986) is closely related to the model of mobile agent computation in this

[2]The requirements are adopted from (Chu and Tzeng, 2005), but with slight modification.

[3]TTP in the Ideal model is a different entity from TTP involved in secure mobile agent protocols in the following sections.

paper. This section formalizes the basic idea. For more details on secure function evaluation, refer to (Yao, 1986).

In this paper, we suppose that agents travel only one-hop away and back. That is, an agent which works on behalf of a user is generated on the site where the user resides. In particular the site is called the *originator O* of the agent. Next the agent moves to a host $H$ to perform a task on behalf of the user. Then the agent runs on $H$ and returns to $O$ along with the result. This is the scenario for one-hop agents, but it is straightforward to extend it into multi-hop cases (Algesheimer et al., 2001; Cachin et al., 2000). Therefore in the subsequent sections, for simplicity we consider one-hop agents only.

Now we give a formalization of mobile agent computations based on secure function evaluation. Suppose that the task that the agent carries out on behalf of the user is represented by function $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{Z}$ for some sets $\mathcal{X}$ and $\mathcal{Y}$. Furthermore, let $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ be two inputs of $O$ and $H$ into $f$, respectively. Let $n_x$, $n_y$, and $n_z$ be the lengths of $x$, $y$, and $z$, respectively. Furthermore, $(x_1, \ldots, x_{n_x})$, $(y_1, \ldots, y_{n_y})$, and $(z_1, \ldots, z_{n_z})$ denote the binary representations of $x$, $y$, and $z$, respectively. Let $C$ be a polynomial-size circuit to compute $f$.

Mobile agent computation proceeds as follows. First $O$ executes construct to obtain a tuple $(\mathcal{C}, \mathcal{K}, \mathcal{L}, \mathcal{U})$, where $\mathcal{C}$ is an encrypted circuit for $C$, and $\mathcal{K}$, $\mathcal{L}$, and $\mathcal{U}$ are "key pairs" for $x$, $y$, and $z$, respectively: $\mathcal{K} = ((K_{1,0}, K_{1,1}), \ldots, (K_{n_x,0}, K_{n_x,1}))$, $\mathcal{L} = ((L_{1,0}, L_{1,1}), \ldots, (L_{n_y,0}, L_{n_y,1}))$, and $\mathcal{U} = ((U_{1,0}, U_{1,1}), \ldots, (U_{n_z,0}, U_{n_z,1}))$. Inputs $x$ and $y$ and output $z$ of $\mathcal{C}$ are represented in an 'encrypted' form in terms of $\mathcal{K}$, $\mathcal{L}$, and $\mathcal{U}$. Namely, $x$, $y$, and $z$ are expressed as $(K_{1,x_1}, \ldots, K_{n_x,x_{n_x}})$, $(L_{1,y_1}, \ldots, L_{n_y,y_{n_y}})$, and $(U_{1,z_1}, \ldots, U_{n_z,z_{n_z}})$, respectively.

Next $O$ performs transfer procedure. That is, it repeats 1-out-of-2 oblivious transfer with $H$ $n_y$ times to securely send the encrypted input for $y$, i.e., $(L_{1,y_1}, \ldots, L_{n_y,y_{n_y}})$, to $H$. Then $O$ also transfers $\mathcal{C}$ and $(K_{1,x_1}, \ldots, K_{n_x,x_{n_x}})$ to $H$. Essentially speaking, it can be considered that the mobile agent consists of $\mathcal{C}$, $(K_{1,x_1}, \ldots, K_{n_x,x_{n_x}})$, and $(L_{1,y_1}, \ldots, L_{n_y,y_{n_y}})$.

Finally the mobile agent runs on $H$. This means that evaluate($\mathcal{C}$, $(K_{1,x_1}, \ldots, K_{n_x,x_{n_x}})$, $(L_{1,y_1}, \ldots, L_{n_y,y_{n_y}})$) is executed on $H$ and the output $(U_{1,z_1}, \ldots, U_{n_z,z_{n_z}})$ is obtained. Then the agent returns to $O$ along with the output. From this, $O$ can recover the final result $z$.

# 3 OUR SCHEMES

In this section we propose two secure mobile agent protocols with emphasis on efficient oblivious transfer suitable for secure function evaluation. Actually in the two protocols, two novel oblivious transfer protocols, each of which is based on its own security assumption, are devised.

Our model of mobile agent computation is basically the same as presented in Section 2.3, but with one additional participant involved, i.e., a trusted third party (TTP). The reason for the introduction of TTP in our mobile agent computation is that no secure mobile computing schemes exist without TTP as shown in (Algesheimer et al., 2001). We call TTP $T$. In this paper we suppose that $T$ utilizes a secure public key cryptosystem. $E_T$ and $D_T$ denote the corresponding encryption and decryption operations, respectively. The symbols in the subsequent sections follow the definitions given in Section 2.

## 3.1 Our Scheme 1

In this section we propose a secure mobile agent scheme in the honest-but-curious model.

**Protocol 1**

**Step 1.** 1. $O$ chooses a unique string *id* for the mobile computation.
2. $O$ executes construct$(C)$ and has the output $(\mathcal{C}, \mathcal{L}, \mathcal{K}, \mathcal{U})$.
3. Using encryption with $T$'s public key, $O$ generates $\bar{L} = E_T(id \| 1 \| (L_{1,0}, L_{1,1}) \| 2 \| \ \ldots \ \| n_y \| (L_{n_y,0}, L_{n_y,1}))$, where '$\|$' means concatenation operation.
4. Let $K'_i$ be $K_{i,x_i}$ for $i = 1, 2, \ldots, n_x$ and $x = (x_1, x_2, \ldots, x_{n_x})$.
5. $O$ sends *id*, $\mathcal{C}$, $K'_1, K'_2, \ldots, K'_{n_x}$, $\bar{L}$ to the host $H$.

**Step 2.** $H$ forwards *id* and $\bar{L}$ to $T$.

**Step 3.** 1. $T$ decrypts $\bar{L}$ with its own private key and checks whether or not the decrypted message includes *id*. If it does not, $T$ quits the protocol. Otherwise, if *id* is used in some previous computation, then $T$ also aborts.
2. $T$ chooses $s_0, s_1 \in_R \mathbb{Z}_p^*$ and computes $s_0P$, $s_1P$. Then it sends $s_0P, s_1P$ to $H$.
3. $H$ chooses $a_i \in_R \mathbb{Z}_p^*$ and compute $A_i = \mathcal{H}_1(i) + s_{y_i}P + a_iP$ $(i = 1, 2, \ldots, n_y)$.
4. $H$ sends $A_i$ to $T$ $(i = 1, 2, \ldots, n_y)$.
5. For $i = 1, 2, \ldots, n_y$, $T$ calculates $D_{i,0} = s_0(A_i - s_0P)$, $D_{i,1} = s_1(A_i - s_1P)$. Next it also chooses $r_{i,0}$, $r_{i,1} \in_R \mathbb{Z}_p^*$ and computes $C_{i,0} = (r_{i,0}P,$

$L_{i,0} \times e(\mathcal{H}_1(i), s_0P)^{r_{i,0}})$, $C_{i,1} = (r_{i,1}P, L_{i,1} \times e(\mathcal{H}_1(i), s_1P)^{r_{i,1}})$.

6. $T$ sends $D_{i,0}, D_{i,1}, C_{i,0}, C_{i,1}$ to $H$ $(1 \leq i \leq n_y)$.
7. $H$ computes $D'_{i,y_i} = D_{i,y_i} - a_i s_{y_i} P$ and obtains $L_{i,y_i} = C_{i,y_i}[2]/e(D'_{i,y_i}, C_{i,y_i}[1]) = e(\mathcal{H}_1(i), s_{y_i}P)^{r_{i,y_i}} / e(D'_{i,y_i}, r_{i,y_i}P)$ $(i = 1, 2, \ldots, n_y)$. Here $C_{i,y_i}[1]$ and $C_{i,y_i}[2]$ denote the first and the second part of the tuple $C_{i,y_i}$ respectively.

**Step 4.** Let $L'_i$ be $L_{i,y_i}$ $(i = 1, 2, \ldots, n_y)$. $H$ executes evaluate$(C, K'_1, K'_2, \ldots, K'_{n_x}, L'_1, L'_2, \ldots, L'_{n_y})$, which yields the output $U'_1, U'_2, \ldots, U'_{n_z}$. $H$ sends them to $O$.

**Step 5.** $O$ obtains the final result $z = (z_1, z_2, \ldots, z_{n_z})$ by comparing $U'_1, U'_2, \ldots, U'_{n_z}$ with $\mathcal{U}$.

Our protocol 1 is almost the same as (Algesheimer et al., 2001; Cachin et al., 2000; Mori et al., 2005) except for Step 3, which is the large difference between ours and the previous work.

## 3.2 Our Scheme 2

In our scheme 1, if $H$ is malicious and sends some queries $A_i$ in some special form in Step 3.4, it would be able to get extra information. Therefore we improve our scheme 1 and propose scheme 2 which is secure even in the malicious model.

**Protocol 2** The difference between Protocol 1 and 2 lies in Step 3. Other steps of Protocol 2 are the same as those of Protocol 1. Below $\mathcal{H}_2$ is a cryptographic hash function over $\mathbb{G}_1 \times \{0,1\}^* \times \{0,1\}$.

**Step 3.** 1. $T$ decrypts $\bar{L}$ with its own private key and checks whether or not the decrypted message includes $id$. If it does not, $T$ quits the protocol. Otherwise, if $id$ is used in some previous computation, then $T$ also aborts.
2. $T$ chooses $s_0, s_1 \in_R \mathbb{Z}_p^*$ and computes $s_0P$, $s_1P$. It then sends $s_0P$, $s_1P$ to $H$.
3. $H$ chooses $a_i \in_R \mathbb{Z}_p^*$ and computes $A_i = \mathcal{H}_1(i) + s_{y_i}P + a_iP$ $(i = 1, 2, \ldots, n_y)$.
4. $H$ sends $A_i$ to $T$ $(i = 1, 2, \ldots, n_y)$.
5. For $i = 1, 2, \ldots, n_y$, $T$ computes $D_{i,0} = s_0(A_i - s_0P)$, $D_{i,1} = s_1(A_i - s_1P)$ and computes $C_{i,0} = L_{i,0} \oplus \mathcal{H}_2(s_0\mathcal{H}_1(i), i, 0)$, $C_{i,1} = L_{i,1} \oplus \mathcal{H}_2(s_1\mathcal{H}_1(i), i, 1)$.
6. $T$ sends $D_{i,0}, D_{i,1}, C_{i,0}, C_{i,1}$ to $H$ $(i = 1, 2, \ldots, n_y)$.
7. $H$ obtains $L_{i,y_i} = C_{i,y_i} \oplus \mathcal{H}_2(D_{i,y_i} - a_i(s_{y_i}P), i, y_i)$ $(i = 1, 2, \ldots, n_y)$.

The property, security, and efficiency of our protocol 2 are extensively discussed in Section 4.

# 4 EVALUATION

## 4.1 General Discussion on Our Protocols

Our scheme 1 and 2 are almost the same as (Algesheimer et al., 2001; Cachin et al., 2000; Mori et al., 2005). However, each Step 3. of them deviates far from the previous work. As stated in Section 2, in the previous mobile agent security schemes we need to repeat 1-out-of-2 oblivious transfer $n_y$ times between Alice and Bob. On the other hand, in this paper in each Step 3. of our protocol 1 and 2 we have proposed two novel oblivious transfer protocol suitable for mobile agent security. This is one of the reasons why our protocols are efficient compared with the previous work. More detailed analysis on performance is given in Section 4.4.

Our oblivious transfer protocols modify $k$-out-of-$n$ oblivious transfer proposed in (Chu and Tzeng, 2005). Note that we cannot use $k$-out-of-$n$ oblivious transfer in mobile agent security schemes in a naive manner because in $k$-out-of-$n$ oblivious transfer, it is possible to choose $k$ indices arbitrarily.

Another important point to note is that in our protocols $T$ can be *less trusted*. For example, in the basic scheme in (Algesheimer et al., 2001), if $T$ and $O$ collude, then the secret $y$ of $H$ is revealed. However, even in such a case, our two protocols are secure.

## 4.2 Security Analysis of Scheme 1

In this section we conduct security analysis of our protocol 1. It is obvious that our protocol 1 is as secure as the original secure function evaluation except for Step 3. Therefore in order to prove the security of our protocol 1, what we have to do is only to show that Step 3. actually satisfies the requirements given in Section 2.2.

**1. Correctness**

The proof is omitted because the readers should easily verify the correctness of the protocol.

**2. The Receiver's privacy**

For privacy of the receiver, $H$, we can prove Theorem 1:

**Theorem 1.** *For our scheme1, the choices made by H are unconditionally secure.*

The proof is omitted. Theorem 1 can be proved in a similar way as in (Chu and Tzeng, 2005).

**3. The Sender's privacy**

**Theorem 2.** *Our scheme1 meets the Sender's privacy requirement. That is, by the DBDH assumption, if*

*H has honest-but-curious behavior (semi-honest), he gets no information about unchosen messages.*

The proof is also omitted due to space limitation. It would be straightforward to prove Theorem 2 by consulting (Chu and Tzeng, 2005).

## 4.3 Security Analysis of Scheme 2

In this section we consider the security of our protocol 2. As stated in Section 4.2, here we consider the Step 3. of our protocol 2.

**1. Correctness**

It is easily proved and the proof is omitted.

**2. The Receiver's privacy**

We introduce Theorem 3 without proof. It can be proved almost in the same way as in (Chu and Tzeng, 2005).

**Theorem 3.** *For our scheme2, the choices made by H are unconditionally secure.*

**3. The Sender's privacy**

**Theorem 4.** *In the malicious model our scheme2 satisfies The Sender's privacy under the assumption of NT-CDH and the random oracle model.*

*Proof.* First remember that $\mathcal{H}_2$ is considered as a random oracle. So in order to query the oracle to obtain $\mathcal{H}_2(s_{y_i}\mathcal{H}_1(i), i, y_i)$, the malicious $H$ must have $s_{y_i}\mathcal{H}_1(i)$ beforehand. Now given any malicious $H$, we construct a simulator $H^*$ in the Ideal model, whose output is indistinguishable from that of $H$. $H^*$ works in the following way:

**Step 1.** $H^*$ simulates $H$ to obtain its output $A_i^*$ ($i = 1, 2, \ldots, n_y$). If $H$ submits query with index $i$ to $\mathcal{H}_1$, then $H^*$ feeds into $H$ a random $Q_i^*$, which should be consistent with the previous queries.

**Step 2.** $H^*$ simulates $T$. First it generates $s_0^*$ and $s_1^*$. Then for $i = 1, 2, \ldots, n_y$, with input $A_i^*$, $H^*$ obtains $D_{i,0}^* = s_0^*(A_i^* - s_0^*P)$ and $D_{i,1}^* = s_1^*(A_i^* - s_1^*P)$.

**Step 3.** $H^*$ outputs $(C_{i,0}^*, C_{i,1}^*)$ at random ($i = 1, 2, \ldots, n_y$).

**Step 4.** $H^*$ simulates $H$ with inputs $s_0^*P$, $s_1^*P$, $\{D_{i,0}^*, D_{i,1}^*, C_{i,0}^*, C_{i,1}^*\}$ ($i = 1, 2, \ldots, n_y$). If $H$ issues a query with $(x, i, b)$ to $\mathcal{H}_2$ ($b \in \{0, 1\}$), then $H^*$ verifies $x \stackrel{?}{=} s_b^*Q_i^*$. If it holds, then $H^*$ obtains $L_{i,b}$ from the TTP in the Ideal model and returns $C_{i,b}^* \oplus L_{i,b}$ to $H$ as the hash value (consistent with the previous queries).

**Step 5.** Outputs $s_0^*P$, $s_1^*P$, $\{A_i^*, D_{i,0}^*, D_{i,1}^*, C_{i,0}^*, C_{i,1}^*\}$ ($i = 1, 2, \ldots, n_y$).

First note that if for some $i$, $H$ can obtain both of decryption keys for the $i$-th key pair $L_{i,0}$ and $L_{i,1}$, then $H^*$ cannot exactly know the indices chosen by $H$ and the simulation would not succeed. This situation could arise if $H$ sends to $\mathcal{H}_2$ two queries $(x_0, i, 0)$ and $(x_1, i, 1)$ such that $x_0 = s_0^*Q_i^*$ and $x_1 = s_1^*Q_i^*$. However, it contradicts to the assumption of the hardness of NT-CDH problem and hence the situation above cannot occur.

For $i = 1, 2, \ldots, n_y$, if $(x, i, y_i)$ is queried and legal at the same time, then $C_{i,0}$ and $C_{i,1}$ are consistent with the returned hash values. Since no other $(s_b^*Q_j^*, j, b)$ where $s_b^*Q_j^* \notin \{s_{y_1}^*Q_1^*, s_{y_2}^*Q_2^*, \cdots, s_{y_{n_y}}^*Q_{n_y}^*\}$ can be queried to the $\mathcal{H}_2$ hash oracle, $C_{j,0}$ and $C_{j,1}$ have the right distribution due to the random oracle model. Thus, the output distribution is indistinguishable from that of $H$. □

## 4.4 Performance Evaluation

In this section we compare our schemes with Cachin's scheme (Cachin et al., 2000) and Mori's scheme (Mori et al., 2005) in terms of communication cost and the computational complexity[4]. Table 1 and Table 2 depict the communication cost between sender $S$ and receiver $R$ and the computational complexity of $S$ and $R$, respectively. Note that in Cachin's scheme, $S$ and $R$ correspond to $O$ and $H$. On the other hand, in our schemes $S$ and $R$ correspond to $T$ and $H$.

The communication cost is estimated by the number of required messages, each of which is in $\mathbb{G}_1$. For the computational complexity, first we interpret operations of the protocol in (Cachin et al., 2000) as those in $\mathbb{G}_1$. Then we take into consideration the number of the most expensive operations, that is, the scalar multiplication in $\mathbb{G}_1$ and bilinear map (pairing) $e$ over $\mathbb{G}_1 \times \mathbb{G}_1$. Note that for good legibility $n_y$ is written as $n$ in Table 1 and 2.

From Table 1 and 2, it should be clear that our schemes are more efficient than Cachin's scheme with respect to the communication cost and the computational complexity. Furthermore, our schemes are almost as efficient as Mori's scheme, but note that the latter is insecure. On the other hand, as we showed, our protocol 1 is secure in the semi-honest model and our protocol 2 is secure in the malicious model.

[4]From the viewpoint of the communication cost and the computational complexity, (Cachin et al., 2000) and (Algesheimer et al., 2001) are almost the same. Therefore due to space constraints here we compare ours with the former only.

Table 1: Comparison on the communication cost.

| | Communication cost | | | |
|---|---|---|---|---|
| | $S \rightarrow R$ | $R \rightarrow S$ | $S \rightarrow R$ | Total |
| C. Cachin, etc. (Cachin et al., 2000) | $n$ | $2n$ | $4n$ | $7n$ |
| Mori, etc. (Mori et al., 2005) | $n$ | $n$ | $3n+2$ | $5n+2$ |
| Our scheme 1 | 2 | $n$ | $4n$ | $5n+2$ |
| Our scheme 2 | 2 | $n$ | $4n$ | $5n+2$ |

Table 2: Comparison on the computational complexity.

| | Computational complexity | | |
|---|---|---|---|
| | $S$ | $R$ | Total |
| C. Cachin, etc. (Cachin et al., 2000) | $5nM$ | $2nM$ | $7nM$ |
| Mori, etc. (Mori et al., 2005) | $4n+2M$ | $2nM$ | $(6n+2)M$ |
| Our scheme 1 | $(4n+2)M+2nE$ | $2nM+nE$ | $(6n+2)M+3nE$ |
| Our scheme 2 | $(4n+2)M$ | $2nM$ | $(6n+2)M$ |

$M$: one scalar multiplication, $E$: one paring

## 5 CONCLUSION

In this paper we proposed two secure mobile agent protocols with emphasis on efficient oblivious transfer suitable for secure function evaluation in untrusted environments. Actually in the two protocols, two novel oblivious transfer protocols were devised. We showed that one is secure in honest-but-curios model and the other is secure even in the malicious model. Furthermore, we showed that our proposed oblivious transfer protocols are more efficient than the previous work.

## ACKNOWLEDGEMENTS

We would like to thank anonymous referees for their valuable comments.

## REFERENCES

Algesheimer, J., Cachin, C., Camenisch, J., and Karjoth, G. (2001). Cryptographic security for mobile code. In *Proc. IEEE Symp. Security and Privacy*, pages 2–11.

Cachin, C., Camenisch, J., Kilian, J., and Müller, J. (2000). One-round secure computation and secure autonomous mobile agents. In *Proc. ICALP*, vol. 1853 of *LNCS*, pages 512–523. Springer-Verlag.

Chu, C.-K. and Tzeng, W.-G. (2005). Efficient *k*-out-of-*n* oblivious transfer schemes with adaptive and non-adaptive queries. In *PKC 2005*, vol. 3386 of *LNCS*, pages 172–183. Springer-Verlag.

Hasegawa, W., Soshi, M., and Miyaji, A. (2007). Efficient communication on mobile agent security. In *Symposium on Cryptography and Information Security (SCIS2007), 4F1-5*.

Mori, M., Soshi, M., and Miyaji, A. (2005). Consideration for mobile agent security. IPSJ SIG Technical Reports 2005-CSEC-28. pp. 123–128.

Naor, M. and Pinkas, B. (1999). Oblivious transfer and polynomial evaluation. In *Proc. STOC '99*, pages 245–254.

Rothermel, K. and Popescu-Zeletin, R., editors (1997). *Mobile Agents: First International Workshop (MA '97)*, vol. 1219 of *LNCS*. Springer-Verlag.

Yao, A. C. (1986). How to generate and exchange secrets. In *Proc. FOCS*, pages 162–167.