2003

Robust non-interactive oblivious transfer

Y. Mu
University of Wollongong, ymu@uow.edu.au

J. Zhang
Macquarie University

V. Varadharajan
Macquarie University

Y. X. Lin
University of Wollongong, yanxia@uow.edu.au

Publication Details
Robust non-interactive oblivious transfer

Abstract
We present a novel scheme of noninteractive m out of n oblivious transfer, which demonstrates significant improvement over the existing schemes in terms of completeness, robustness and flexibility. This scheme is useful for protection of user privacy in the Internet.

Keywords
Internet data privacy security of data telecommunication security

Publication Details
Robust Non-Interactive Oblivious Transfer

Yi Mu, Member, IEEE, Junqi Zhang, Vijay Varadharajan, Senior Member, IEEE, and Yan-Xia Lin

Abstract—We present a novel scheme of noninteractive \( \gamma \) out of \( n \) oblivious transfer, which demonstrates significant improvement over the existing schemes in terms of completeness, robustness and flexibility. This scheme is useful for protection of user privacy in the Internet.

Index Terms—Data security, oblivious transfer.

I. INTRODUCTION

The concept of Oblivious Transfer (OT) was introduced by Rabin [1]. Rabin’s OT can be considered as a game between two polynomial time parties, Alice and Bob. Alice sends a bit to Bob in such a way that with 1/2 probability Bob will receive the same bit and with 1/2 probability Bob will receive nothing. Alice does not know which event has happened. Rabin’s initiative has attracted a lot of attentions. Various OT methods have been subsequently proposed (e.g., [2]–[7]), where most notable ones are one out of two OT and chosen one out of two OT. In a one out of two OT \((2/2)\)-OT), Alice sends two bits to Bob who receives one of these bits with equal probability and knows which bit he has received, while Alice does not know which bit Bob received. A chosen one out of two OT is similar to a normal one out of two OT; the different between them is that, in the former, Bob can choose an index \( c \) and receives bit \( b_c \). Alice does not learn \( c \).

One direct extension to \((2/2)\)-OT is 1 out of \( n \) oblivious transfer \([8],[9]\). However, there has been little study in \( m \) out of \( n \) oblivious transfer, \((n/m)\)-OT. The closest scheme is the \( n - 1 \) out of \( n \) OT proposed by Bellare and Micalli [10]. Roughly speaking, in an \( m \) out of \( n \) oblivious transfer, Bob can receive only \( m \) messages out of \( n \) messages \( n > m \) sent by Alice; and Alice has no idea about which ones have been received. The OT proposed by Bellare and Micalli [10] is noninteractive. By noninteractive we mean that Bob does not need to communicate with Alice during an OT process. Santis and Persiano [8] also proposed a noninteractive OT protocol. Their scheme falls within the case of 1 out of \( n \).

In this paper, we go one step further by giving a new noninteractive OT scheme that covers the complete OT spectrum. We called them \( m \) out of \( n \) OT, \((n/m)\)-OT. Here, \( m \) is an arbitrary number in \( 1 \leq m < n \). The original OT scheme by Rabin can be considered as one of cases in our scheme. One important feature in our schemes is that the sender and the recipient can securely implement an OT process without the involvement of a trusted third party, because the security can be proved by both the sender and the recipient.

II. SYSTEM SETUP

\( m \) of \( n \) oblivious transfer is defined as follows. Alice knows \( n \) messages and wants to send \( m \) of them to Bob. Bob gets \( m \) of them with probability \( m!/(n-m)!/n! \) and knows which ones he has got, but Alice has no idea about which \( m \) messages Bob has received.

Assume that Alice intends to send \( n \) messages, \( M_1,\ldots,M_n \in \mathbb{Z}_p \) to Bob and knows for sure that Bob can receive \( m \) of them. Which ones will be received by Bob is unknown to Alice. We now describe the Bob’s public key generation algorithm that will be used to our \((n/m)\)-OTs.

Let \( p \) be a large prime number, \( \mathbb{Z}_p^* \) be a multiplicative group, \( g \in \mathbb{Z}_p^* \) be the generator of order \( q \) and \( q + 1 = n \). Given \( x_i \), the \( n \) public keys are constructed by using a set of \( m \) linear equations with respect to \( a_1,\ldots,a_m \).

\[
\begin{align*}
& a_1 x_1 + a_2 x_2^2 + \cdots + a_m x_m^m = y_i, \quad i = 1,\ldots,m. 
\end{align*}
\]

The corresponding linear equations in a matrix form are as follows:

\[
\begin{pmatrix}
1 & x_1 & x_1^2 & \cdots & x_1^{m-1} \\
1 & x_2 & x_2^2 & \cdots & x_2^{m-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_m & x_m^2 & \cdots & x_m^{m-1}
\end{pmatrix}
\begin{pmatrix}
\alpha_1 \\
\alpha_2 \\
\vdots \\
\alpha_m
\end{pmatrix}
= \begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_m
\end{pmatrix}.
\]

Here \( y_i = y_i/x_i \). The coefficient matrix \( A \) is a so-called Vandermonde matrix. The determinant of \( A \) is not equal to zero

\[
\det A = \prod_{1 \leq i < k \leq m} (x_i - x_k) \neq 0,
\]

i.e., \( A \) is a nonsingular matrix, because \( x_i \) Bob chose are distinct and no element \( (x_i - x_k) \) in this product equals zero. Since the determinant of the coefficient matrix is nonzero, the equations have a unique solution over the field \( \mathbb{Z}_p \).

After Bob has got the unique solution \( a_1,\ldots,a_m \), he can calculate other \( n - m \) “public keys” (their discrete logs are unknown), using the following formula:

\[
y_j = a_1 x_j + \cdots + a_m x_j^m, \quad m < j \leq n.
\]

As a result, he has \( n \) public keys \( \{x_i,y_i\}_{i=1}^n \). Bob shuffles his public keys such that the order is known to himself only. The shuffled public keys are then made public. For
convenience, we denote by $U$ the subset of public key indices whose associated public key discrete logs are unknown to Bob and by $K$ those known. Since the public keys will always come with a shuffled form, we still denote by $\{x_i,y_i\}_{i=1}^m$ the shuffled public key set.

The public keys can be easily verified without knowing the corresponding private keys. Given the public key set $\{x_i,y_i\}_{i=1}^m$, we can choose any $m$ of public keys from the public key set, and then calculate $\hat{a}_i$ for $i = 1, \ldots, m$ with respect to the $m$ public keys, where $\hat{a}_i \equiv a_i$ if the public keys are genuine. With the resultant $\hat{a}_i$, we can verify the rest of $(n-m)$ public keys,

$$\hat{y}_j = \hat{a}_1 x_j + \cdots + \hat{a}_m x_j^m, \quad m < j \leq n.$$  

Here, $\hat{y}_j \in \{y_i\}_{i=1}^n$ have not been used in computation of $\hat{a}_i$.

### A. Claim 1

**Given $x_i$. Bob cannot cheat by pre-selecting $y_i$.**

The explanation is as follows. After Bob found the unique coefficient set $\{a_i\}_{i=1}^m$, he can compute $y_i$ for $i \in U$ in terms of the given $\{x_i\}_{i=m+1}^n$. However, it is infeasible for him to compute the discrete logs of these values in poly-time. Bob should not be able to cheat by pre-selecting $\{y_i\}_{i=m+1}^n$ and then try to find $\{a_i\}_{i=1}^m$ that satisfies all $n$ equations. To fix this potential problem, we give the following lemma.

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

$$A' = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{m-1} & y_1 \\ 1 & x_2 & x_2^2 & \cdots & x_2^{m-1} & y_2 \\ \vdots & & & & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{m-1} & y_n \end{pmatrix},$$

which is an $n \times (m+1)$ matrix.

**Proof:** Consider equation:

$$A' \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$  

Because the rank of $A'$ is $m+1$, by making row transformations, we have a nonsingular matrix $C$ such that

$$CA' = \begin{pmatrix} B \\ 0 \end{pmatrix}$$

where $B$ is an $(m+1) \times (m+1)$ nonsingular upper triangle matrix. Therefore

$$CA' = \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$  

This is equivalent to

$$B \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$  

Thus, we have

$$\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \\ -1 \end{pmatrix} = B^{-1} \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$  

Because $B^{-1}$ is nonsingular, it implies that the equations with respect to $a_i$ have no nonzero solution at all. In other words, Bob cannot find a nonzero solution if he wants to cheat by precomputing $y_i$. Therefore, in the verification of public keys, we also need to check if or not the rank of $A'$ is equal to $m+1$.

### III. Non-Interactive $m$ Out of $n$ OT

Using the setup phase given in Section II-A, Bob obtains his private keys $s_i$ for $i = 1, \ldots, m$ and his public keys $\{x_i,y_i\}_{i=1}^n$, where the discrete logs of $y_i$ for $i \in U$ are not known. The protocol is described as follows.

**Alice:**

- randomly chooses $t_1, \ldots, t_n \in R \mathbb{Z}_q$;
- calculates $z_i = M_t^{t_i}$;
- sends to Bob $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ and $\begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$.

**Bob:**

- decrypts $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ to recover $\begin{pmatrix} s_1 \\ \vdots \\ s_n \end{pmatrix}$;
- calculates $\alpha_i = g^{s_i}$;
- generates the order of messages at random;
- based on the order, calculates $z_i = M_t^{t_i}$ for $i = 1, \ldots, m$;
- then sends to Alice $\gamma_i$, $\alpha_i t_i$ and $z_i$;
- Bob: decrypts $z_i$ to recover $m$ messages, $z_i / \alpha_i t_i = M_i$, $i \in K$.

### A. Claim 2

**Completeness** If Alice correctly follows the procedure, Bob can recover $m$ out of $n$ messages, $1 \leq m < n$.

This is obvious. Note the facts that the order of the public keys are not changed and Bob knows their indices. Bob has $m$ private decryption keys $s_i$, $i \in K$, and knows which ones to decrypt. The encryptions done by Alice are based on the standard ElGamal encryption scheme. [11]
B. Claim 3

(Soundness) Both Alice and Bob cannot cheat. Alice does not know which public keys are associated with Bob’s private keys, so she cannot know which messages Bob can decrypt and has no control over which messages Bob will receive. Bob cannot cheat by manipulating his public keys. This is because Alice can check the correctness of Bob’s public key using the method described earlier in this paper. The security is, however, based on the assumption that our system is poly-time-bounded. Bob cannot solve the discrete log problem in poly-time.

C. Non-Malleable Encryption

In the scheme presented in the preceding section, Alice was assumed to be honest in that she always uses Bob’s public keys in encryption. The assumption is reasonable, since Alice wants Bob to receive $\eta$ out of $\eta$ messages she sent. However, if the order of the public keys or the order of the ciphertext is changed by accident (or by an adversary) during the transmission, Bob will not be able to find the fraud in the case that the messages consist of unrecognizable strings. We now modify the scheme so that he can check if or not the encrypted messages sent by Alice are correctly constructed. We now construct a nonmalleable encryption by reconstructing the private keys: select private keys, $s_i \in \mathbb{Z}_q$, and some integers, $s'_i \in \mathbb{Z}_q$, such that they satisfy $s_i s'_i \mod q = s_i$. It is not hard to find that we can select $s_i$ and $s'_i$ that satisfy $s_i(s'_i - 1) \equiv q$ and $s'_i \neq 1$. Bob needs to keep $s_i$ and $s'_i$ secret. His public keys are still the same. The correctness of the encryptions can then be verified during the decryption. Bob now decrypts the obliviously transferred messages using two different methods: for message $M_i$:

Method 1: Compute $(M_i g^{(\sigma_i)} s'_i)^{s_i} / \alpha_i^{s_i} = M_i^{g^{(n)}} g^{(s'_i - s_i)}$.

Remove $s'_i$ from the message, Bob then gets $M_i^{g^{x}} (s'_i - s_i)(s'_i - 1)$.

Method 2: Compute $M_i g^{(\sigma_i)} / \alpha_i^{s_i} = M_i g^{x} (\sigma_i - s_i)$.

Bob then checks the equality of two messages. The completeness is straightforward. To prove the soundness, we assume that Alice has not correctly used Bob’s public keys in her encryptions, but uses $g^{x_i}$. Bob can immediately find the fraud.

REFERENCES