A note on “Selling multiple secrets to a single buyer”

Nasrollah Pakniat a, Ziba Eslami a,⇑, Ali Miri b

a Department of Computer Science, Shahid Beheshti University, Tehran, Iran
b Department of Computer Science, Ryerson University, Toronto, Canada

A R T I C L E   I N F O
Article history:
Received 11 June 2013
Received in revised form 20 January 2014
Accepted 1 February 2014
Available online 12 February 2014

Keywords:
Cryptography
Jacobi symbols
Oblivious transfer
Secret selling

A B S T R A C T
In 2009, Rey and Sanchez proposed a protocol to solve the oblivious transfer problem using quadratic residues and Jacobi symbols. The goal of this paper is to prove that the scheme of Rey and Sanchez does not achieve the security requirements of this problem in the sense that the sender can learn the secrets retrieved by the receiver and thereby violate his privacy.

1. Introduction

Consider the following problem: Alice (A) has k secrets and is willing to sell any of them to Bob (B) subject to two conditions: (1) if B pays for only t (t < k) secrets, he cannot obtain any information about the other secrets and (2) A cannot obtain any information about which of the secrets B has bought. This problem is known as “secret selling of secrets”. In cryptography, it is also referred to as the “oblivious transfer (OT)” problem, where a sender transfers some of potentially many pieces of information to a receiver, but remains oblivious to which pieces have been transferred. The importance of the applications that can be built based on oblivious transfer makes this problem a critical problem in the field. In particular, it is one of the fundamental building blocks for secure multi-party computation applications[5,15]. Throughout this paper, we consider the two problems, secret selling of secrets, and oblivious transfer, interchangeably.

The oblivious transfer problem was first introduced in 1981 by Rabin[11]. He proposed an OT protocol and used it to design a secret exchange protocol, hoping that the two parties could exchange their secrets fairly. In Rabin’s OT protocol, A sends one secret bit to B. Now, B either gets this bit with probability 1/2 or he gets nothing at all. In this protocol, A is not able to determine whether B gets the secret bit or not. Even et al. in [3] extended Rabin’s OT concept to “1-out-of-2 oblivious transfer” (OT2 1). In an OT2 1 protocol, B can retrieve only one of the two secret bits which have been sent by A and A cannot determine which one is B’s choice. In [1], Brassard and Crépeau introduced a further generalization in the form of OTk 1 (also known as all-or-nothing disclosure of secrets). Here, A has k secret messages and B can retrieve exactly one of them. As in the previous concepts, A should not be able to determine the position of the retrieved message.

A more flexible extension of OTk 1 is OTt 1 in which B is able to retrieve exactly t-out-of-k secret messages and A cannot position the retrieved messages. Running OTt 1 protocols t times is a straightforward and non-efficient solution to OTk 1. Some papers, including [10,14], proposed OTt 1 protocols with universally usable parameters which makes them more efficient.

⇑ Corresponding author. Tel.: +98 21 29903007; fax: +98 21 22431655.
E-mail addresses: n_pakniat@sbu.ac.ir (N. Pakniat), z_eslami@sbu.ac.ir (Z. Eslami), Ali.Miri@ryerson.ca (A. Miri).
for multiple runs. The first non-trivial solution for $OT^k$ was proposed in [9] by Naor and Pinkas. Later, Mu et al. proposed three non-interactive $OT^k$ protocols [8]. The first protocol, as the authors themselves said, was not efficient. The second and third protocols were shown to be insecure in [4]. Chang and Lee [2] used the Chinese reminder theorem and blind signatures to propose another $OT^k$ protocol in 2009. In the same year, Rey and Sanchez [12], used Jacobi symbols and hash functions and proposed a new solution for the $t$-out-of-$k$ oblivious transfer problem.

We also note that there is the concept of generalized oblivious transfer (GOT) in the literature. In a GOT protocol, $A$ has a set of secret messages $M = \{m_1, \ldots, m_t\}$ and $B$ can retrieve a subset of these messages under some conditions. The interested reader can find more information in [6,13].

The aim of this paper is to prove that the protocol of Rey and Sanchez [12] does not satisfy the security requirements of an $OT^k$ protocol. We show that in this protocol, it is possible for $A$ to determine the secrets retrieved by $B$ and violate his privacy.

The rest of the paper is organized as follows. Section 2 gives a brief overview of the necessary mathematical background as well as a review of the protocol proposed in [12]. The details of the security flaw are addressed in Section 3 and finally, conclusions are drawn in Section 4.

2. Related works

In this section, we first briefly review the required mathematical background and then proceed to outline the protocol proposed in [12].

2.1. Mathematical background

Let $n$ be an integer. An element $X \in Z_n$ is said to be a quadratic residue modulo $n$ if there exists another element $x \in Z_n$ such that $x^2 \equiv X (\text{mod } n)$. The element $x$ is called a square root of $X$. The set of all quadratic residues modulo $n$ is denoted by $Q_n$. If $n$ is an odd prime integer and $X \in Q_n$, then $X$ has exactly two square roots modulo $n$. Moreover, if $n = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$, with $p_i$’s primes and $e_i \geq 1$, then $X$ has $2^k$ different square roots modulo $n$. It is a well known fact with interesting cryptographic applications, that for odd primes $p$ and $q$ with $n = p \cdot q$, the problem of computing the square roots of $X \in Q_n$, is computationally equivalent to the problem of factoring $n$.

A useful tool to know whether or not an integer number $x$ is a quadratic residue modulo an odd prime $p$ is given by the Legendre symbol $\left( \frac{x}{p} \right)$ which is defined as follows:

$$\left( \frac{x}{p} \right) = \begin{cases} 
0 & \text{if } p \text{ divides } x, \\
1 & \text{if } x \in Q_p, \\
-1 & \text{otherwise}.
\end{cases}$$

The Jacobi symbol is a generalization of the Legendre symbol to odd integers $n \geq 3$, which are not necessarily prime. The Jacobi symbol, $\left( \frac{x}{n} \right)$, when $n = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k}$ is:

$$\left( \frac{x}{n} \right) = \left( \frac{x}{p_1} \right)^{e_1} \cdot \left( \frac{x}{p_2} \right)^{e_2} \cdots \left( \frac{x}{p_k} \right)^{e_k}.$$  

The following result relates both to square roots modulo $n$ and Jacobi symbols [7]:

**Proposition 1.** Let $n = p \cdot q$, where $p$ and $q$ are odd primes such that $p \cdot q \equiv 3 \pmod{4}$. Then for $X \in Q_n$, the two square roots of $X$ modulo $n$ have distinct Jacobi symbols.

2.2. Review of the Rey and Sanchez’s protocol [12]

In this section, we provide a brief review of the $OT^k$ protocol proposed in [12]. This protocol needs a trusted third party ($TTP$). We denote the secret messages by $s_1, \ldots, s_k$. For simplicity, we consider the case where $B$ wants to buy only one secret. The details are as follows:

1. $A$ computes $k$ pairs of RSA keys: $\{(n_i, e_i), d_i\}, 1 \leq i \leq k$, where $n_i = p_i \cdot q_i$ and $p_i, q_i$ are odd primes congruent to $3 \pmod{4}$. Then $A$ encrypts each secret $s_i, 1 \leq i \leq k$ with the ith public key to obtain the corresponding cryptogram: $c_i \equiv s_i^{e_i} \pmod{n_i}$.
2. $A$ sends encrypted secrets $(c_1, \ldots, c_k)$ to $B$ with corresponding public keys $(n_1, e_1), \ldots, (n_k, e_k)$.
3. $A$ sends the $k$ public keys to the $TTP$.
4. The $TTP$ chooses $k$ random numbers, $x_1, \ldots, x_k$, with $\gcd(x_i, n_i) = 1, 1 \leq i \leq k$, and computes the corresponding Jacobi symbols $\left( \frac{x_1}{n_1} \right), \ldots, \left( \frac{x_k}{n_k} \right)$.
5. The $TTP$ computes the hashes of the following sequences of Jacobi symbols:
hash\left(-\left(\frac{x_1}{n_1}\right), \ldots, \left(\frac{x_i}{n_i}\right), \ldots, \left(\frac{x_k}{n_k}\right)\right) \rightarrow h_1
\]

hash\left(\left(\frac{x_1}{n_1}\right), \ldots, \left(\frac{x_i}{n_i}\right), \ldots, \left(\frac{x_k}{n_k}\right)\right) \rightarrow h_j
\]

hash\left(\left(\frac{x_1}{n_1}\right), \ldots, -\left(\frac{x_i}{n_i}\right), \ldots, -\left(\frac{x_k}{n_k}\right)\right) \rightarrow h_k
\]

Thus, the sequence of hashes \{h_1, \ldots, h_k\} is obtained.

(6) The TTP sends \(x_1, \ldots, x_k\) to \(B\).

(7) The TTP sends the set \(\text{PermHashSeq} = \{h_{II(1)}, \ldots, h_{II(k)}\}\) to \(A\), where \(II\) is a secret permutation of \(k\) elements.

(8) For each \(x_i, 1 \leq i \leq k\), \(B\) computes \(X_i \equiv x_i^2 \pmod{n_i}\) and its Jacobi symbol \(\left(\frac{x_i}{n_i}\right)\).

(9) In order to buy the \(j\)th secret, \(B\) sends the following pairs to \(A\):
\[
\left(X_1, \left(\frac{x_1}{n_1}\right)\right), \ldots, \left(X_i, -\left(\frac{x_i}{n_i}\right)\right), \ldots, \left(X_k, \left(\frac{x_k}{n_k}\right)\right).
\]

(10) \(A\), upon receiving the sequence of Jacobi symbols sent by \(B\), computes its hash \(\tilde{h}\). She then checks if \(\tilde{h}\) is in \(\text{PermHashSeq}\). If \(\tilde{h} \in \text{PermHashSeq}\), then \(A\) is sure that \(B\) is honest and has asked for just one secret, otherwise, \(B\) is found to be a cheater.

(11) For each \(i\), \(A\) computes the square root of \(X_i\) modulo \(n_i\), whose Jacobi symbol is equal to that sent by \(B\). Consequently, \(A\) obtains the sequence:
\[
x_1, \ldots, x_{j_1-1}, y_j, x_{j_1+1}, \ldots, x_k
\]

and sends these values to \(B\).

(12) \(B\) obtains two different square roots of \(X_j\) modulo \(n_j\), i.e. \(x_j\) and \(y_j\). Then, \(B\) can factorize \(n_j\), decrypt \(c_j\) and get the secret message \(s_j\).

3. Violation of receiver’s privacy

In this section, we show that the protocol reviewed in Section 2.2 does not achieve the security requirements for the receiver. We show that this protocol suffers from an important security flaw so that \(A\) can easily position the secrets that \(B\) has bought. For simplicity, we consider (as in Section 2.2) the case where \(B\) wants to buy only one secret. It is then straightforward to extend the attack to the case of multiple secrets.

**Theorem 1.** Assume that \(A, B\) and TTP perform the protocol of Section 2.2 for the secrets \(s_1, \ldots, s_k\). Let \(s_l\) be the secret bought by \(B\) at the end of the protocol. Then \(A\) can determine \(l\) as well.

**Proof.** Since \(B\) is after \(s_k\), she sends \(\left(X_1, \left(\frac{x_1}{n_1}\right)\right), \ldots, \left(X_i, -\left(\frac{x_i}{n_i}\right)\right), \ldots, \left(X_k, \left(\frac{x_k}{n_k}\right)\right)\) to \(A\). Now, by receiving \((X_1, js_1), \ldots, (X_k, js_k)\), \(A\) can perform the following steps to determine \(l\). Note that all computations on indices are done modulo \(k\).

- For \(i = 1, \ldots, k-1:\)
  1. Compute \(h = \text{hash}\{js_1, \ldots, js_{i-1}, -js_i, -js_{i+1}, js_{i+2}, \ldots, js_k\}\).
  2. If \(h \in \text{PermHashSeq}\) then
     a. Compute \(h' = \text{hash}\{js_1, \ldots, js_i, -js_{i+1}, -js_{i+2}, js_{i+3}, \ldots, js_k\}\).
     b. If \(h' \in \text{PermHashSeq}\), then set \(l = i + 1\) and stop.
     c. Otherwise set \(l = i\) and stop.

We now show that the above procedure determines \(l\). Suppose first that \(l > 1\). Hence, for \(i = 1, 2, \ldots, (l-2)\), the hash value \(h = \text{hash}\{js_1, \ldots, -js_i, -js_{i+1}, \ldots, js_k\}\) does not belong to \(\text{PermHashSeq}\) and therefore the \(\text{if}\) statement in the procedure will be skipped. When \(i\) reaches \(l-1\), then we have
\[
h = \text{hash}\{js_1, \ldots, -js_{l-1}, -js_l, \ldots, js_k\}\.
\]
However, \(B\) has already negated the \(l\)th Jacobi symbol \(\left(\frac{x_l}{n_l}\right)\) (or equivalently \(js_l = -\left(\frac{x_l}{n_l}\right)\)), therefore in \(h\), only one Jacobi symbol is actually negated which means that \(h \in \text{PermHashSeq}\). Now \(h'\) is computed as \(\text{hash}\{js_1, \ldots, js_{l-1}, -js_l, -js_{l+1}, \ldots, js_k\}\), where
\[ j_{i-1} = \frac{x_{i-1}}{n_{i-1}}, \quad j_i = -\left(\frac{x_i}{n_i}\right), \quad j_{i+1} = \frac{x_{i+1}}{n_{i+1}}. \]

Therefore \( h' \) is also in \( \text{PermHashSeq} \) which means that \( l \) is computed correctly as \( i + 1 \) and the procedure stops.

Now, suppose that \( l = 1 \). In this case, for \( i = 1 \), we have \( h \in \text{PermHashSeq} \) while \( h' \) is not in \( \text{PermHashSeq} \). Therefore, \( l \) is correctly determined as \( 1 \) and the procedure stops.

A possible solution for this problem would be to remove the exchange of hash values and employ techniques to prevent \( B \) from obtaining more than \( t \) secrets. For example, instead of sending the vector \( Y = [y_1, \ldots, y_k] \) to \( B \), the seller \( A \) might force \( B \) to solve some equations to get the secrets. \( A \) can send the augmented matrix \( [C|b] \) corresponding to a linear system of \( t \) equations in \( k \) unknowns to \( B \) where \( C \) is an arbitrary \( t \times k \) matrix such that every \( t \) columns of it are linearly independent and \( b = C \cdot Y \). Now, by the property of the coefficient matrix \( C \), honest buyers would be able to solve a \( t \times t \) linear system and obtain \( t \) secrets. It is easy to see that the proposed attack does not apply anymore and at the same time the security requirements of \( (OT^A_t) \) problem are achieved.

4. Conclusions

In this paper, we consider the security of the \( t \)-out-of-\( k \) oblivious transfer protocol proposed in [12]. We prove that this protocol does not protect privacy of the receiver and show that the sender can determine the secrets retrieved by the receiver.

Acknowledgments

The authors would like to thank Prof. Witold Pedrycz and anonymous reviewers for their valuable comments.

References