ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.4, April 2007, pp.1015−1025
DOI: 10.1360/jos181015

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

独立的不经意传输*

黄 琼+, 赵一鸣

(复旦大学 计算机科学与工程系,上海 200433)

Independent Oblivious Transfer

HUANG Qiong+, ZHAO Yi-Ming

(Department of Computer Science and Engineering, Fudan University, Shanghai 200433, China)

+ Corresponding author: Phn: +86-21-55078212, E-mail: 032021139@fudan.edu.cn

Huang Q, Zhao YM. Independent oblivious transfer. *Journal of Software*, 2007,18(4):1015−1025. http://www.jos.org.cn/1000-9825/18/1015.htm

Abstract: This paper presents a flavor of OT named Independent Oblivious Transfer in the Public-Key Public-Randomness Model (PKPR IOT, in short), with respect to the open problem given by De Santis. First it gives a non-interactive implementation of IOT, which can independently and obliviously transfer polynomial messages. The implementation is based on Quadratic Residuosity Assumption. Since it is limited to pre-fixed times, then another non-interactive implementation of IOT is presented, which can independently transfer messages for any times. The second implementation requires the sender to be honest and the receiver couldn't make his choice independently. So, the third interactive implementation is given. It is based on the BBCS oblivious transfer scheme of Rivest, and is rather more efficient than the above two non-interactive counterparts. All the three implementations presented are secure against receivers with unlimited computational power.

Key words: oblivious transfer; zero-knowledge proof; quadratic residuosity assumption; PKPR model; GM encryption

摘 要: 针对 De Santis 给出的开放问题,提出了公开密钥公开随机性模型下的独立的不经意传输(简称 PKPR IOT).首先,给出了 IOT 的一个非交互式实现,它能够独立地不经意传输多项式条消息.该实现是基于二次剩余假设.由于它受限于多项式次,所以又给出了 IOT 的另一个非交互式实现,它能够独立地传输任意次消息.但是这个实现要求发送者是诚实的,并且接收者不能独立地选择接收哪条消息.因此给出了第三个交互式的实现.它基于 Rivest 的 BBCS 不经意传输,并且比上面两个非交互式实现的效率高得多.这 3 种实现对于具有无限计算能力的接收者都是安全的.

关键词: 不经意传输;零知识证明;二次剩余假设;公开密钥公开随机性模型;GM 加密
中图法分类号: TP309 文献标识码: A

* Supported by the National Natural Science Foundation of China under Grant No.60573054 (国家自然科学基金)
Received 2005-09-14; Accepted 2005-12-01

1 Introduction

Since Rabin[1] first introduced oblivious transfer (OT) into cryptography in 1981, OT has been widely used in cryptography and becomes an important cryptographic primitive. It has already become the basis for realizing interactive protocols, such as bit commitments, zero-knowledge proofs, and general secure multiparty computation, and etc.[2,3]

OT refers to several types of two-party protocols. One party is the *sender* named Alice, who has a string *s* to be sent, and the other party is the *receiver* named Bob. At the end of an execution of OT protocol, Bob either learns the string or gets nothing about *s*, and each of the two events happens with the same probability. However, Alice cannot figure out better than randomly guessing that whether Bob learns the string or not. In 1989, Bellare and Micali[4] presented a non-interactive implementation of Oblivious Transfer based on Discrete Log Assumption that is secure with respect to an all-powerful sender and a probabilistic polynomial-time receiver.

There are great deals of variants of OT, such as 1-*out-of*-2 *OT*, *k*-*out*-*of*-*n OT*, committed OT, concurrent OT and etc. They have been studied extensively[4−9] and have been used in lots of cryptographic protocols[3,10]. Among all the variants, we are most concerned with the 1-*out-of*-2 *OT*. The notion of 1-*out-of*-2 *Oblivious Transfer* was devised by Even, Goldreich, and Lempel[11]. A 1-*out-of*-2 *OT* is from Alice whose input consists of two strings, s_0 and s_1, to Bob who has a bit meaning which message to receive. At the end of an execution of the 1-*out-of*-2 *OT*, Bob only knows exactly one of the two strings, s_0 or s_1, and Alice can only guess with probability 1/2 which string of the two Bob gets. De Santis *et al.* presented a non-interactive implementation of 1-*out-of*-2 *Oblivious Transfer* in the *PKPR Model* in 1995, which relies upon the quadratic residuosity assumption and is secure against receivers with unlimited computational power[12].

The *PKPR Model* is a certain kind of model in which all the users share a random common string, choose and validate by themselves their own public and private keys without interacting with the other users. It doesn't need any trusted center to protest against possible "cheating" by users.

Zero-knowledge proof is used in their implementation, which is a cryptographic primitive as well. It was formalized by Goldwasser, Micali and Rackoff[13]. The proof system De Santis *et al*. used in their implementation is called non-interactive Zero-knowledge Proof System[14].

Their implementation can execute efficiently, but there is still a disadvantage, that is, for every transfer, the choice of Bob to receive a message is always the same. We cannot use this implementation to independently and obliviously transfer many pairs of messages with the same public information. If we need to obliviously transfer multi-pairs of messages, a naive method is to simply use their implementation of 1-*out-of*-2 *OT* repeatedly for multi-times. The result is that Bob will receive all the messages from the same channel. Then Alice can guess the channel from which Bob receives the messages in an all-or-nothing fashion, which is the same as that of a single OT. Here we want the channel Bob receives messages from to be independent and unpredictable to Alice at each time. If Alice learns Bob's choice during some *OT* without knowing Bob's secret, it doesn't make any good for Alice to know his counterpart's choices during other former or later transfers. Another way to implement this is that before every 1-*out-of*-2 *OT*, Bob publishes a new public file which consists of all the public information, and generates new corresponding secret information. Then from which channel Bob receives a message will be different every time. However, the frequent changes of public information will be costly and inconvenient.

In this paper, we present a flavor of OT named Independent Oblivious Transfer in the Public-Key Public-Randomness Model (PKPR IOT). And to our knowledge, there has been no previous work on independent OT. With a PKPR IOT, the two parties can independently and obliviously transfer multi pairs of messages efficiently, without worrying about that information leakage for some or other OT will threaten the security of other

OT's, supposed that Bob's secret keys are still unrevealed to Alice. First we give a non-interactive implementation, within which Alice and Bob can independently transfer polynomial messages. Due to its limitation to polynomial times, we then present another non-interactive implementation within which the two parties could independently transfer messages for any times. But the sender is required to be honest and the receiver's choice is not made totally on his own. Thus, we give a third interactive implementation. It needs an initialization step which is done once for all. This implementation is much more efficient than the other two. Compared with the OT scheme in Ref.[12], our first two implementations are at least the same efficient, and the third scheme is much more efficient. All the three schemes own the additional property of independence.

In the next section, we mainly give some preliminary knowledge which will be used in IOT and introduce PKPR OT presented by De Santis *et al*. In Section 3, we present the Independent Oblivious Transfer in the PKPR Model, and give three implementations of it, two non-interactive and one interactive, along with the security and efficiency analysis of them. Conclusion about our work is given in Section 4.

2 Preliminaries

2.1 Basic notations

We denote by $|x|$ the length of x if x is a string, the length of the binary string representing x if x is an integer, or the absolute value of x if x is a real number. By the expression $x||y$ we denote the concatenation of x and y if they are both strings.

We say that a function $f()$ is negligible if for all constants c, there exists an integer n_c such that for all integers $n>n_c$ it holds that $f(x)<n^{-c}$.

If $A(x)$ is a probabilistic algorithm with one input parameter, then for any input x, the notation $A(x)$ refers to the probability space that assigns to the string σ the probability that A, on input x, outputs σ. If S is a probability space, then "$x\leftarrow S$" denotes the algorithm which assigns to x an element randomly selected according to S.

If $p(\cdot,\cdot,\ldots)$ is a predicate, the notation $\Pr[x\leftarrow S;y\leftarrow T;\ldots;p(x,y,\ldots)]$ denotes the probability that $p(x,y,\ldots)$ will be true after the ordered execution of the algorithms $\{x\leftarrow S; y\leftarrow T;\ldots\}$.

2.2 Facts

A multitude of concepts and definitions in this paper are the same as those in Ref.[12]. Here we only list some facts.

Fact 1: y is a quadratic residue modulo x if and only if y is a quadratic residue modulo for each of the prime divisors of x.

Fact 2: Let p_1,p_2 are distinct primes and $x=p_1*p_2$, and $y\in Z_x^*$, then we have $(y|x)=(y|p_1)*(y|p_2)$.

Fact 3: Let x be a Blum integer and let $y_1, y_2\in Z_x^*$. Then $QR_x(y_1*y_2)=QR_x(y_1)\oplus QR_x(y_2)$.

Note that this fact can be easily extended to general case. That is, let x be a Blum integer and let $y_1,y_2,\ldots, y_n\in Z_x^*$, then $QR_x(y_1*y_2*\ldots*y_n)=QR_x(y_1)\oplus QR_x(y_2)\oplus\ldots\oplus QR_x(y_n)$.

2.3 Oblivious transfer

OT was first introduced by Rabin[1], who gave an implementation based on the difficulty of factoring. OT is a protocol for two parties: the *sender* named Alice who has a string s to send, and the *receiver* named Bob. Both of the following two events are equally likely to occur at the end of an execution of the protocol: either Bob learns the string s, or Bob gets nothing about s. Moreover, at the end of the execution, Alice cannot tell with a non-negligible advantage whether Bob gets s or not.

Bellare and Micali[4] presented in 1989 a non-interactive implementation of OT based on Discrete Log

Assumption that is secure with respect to an all-powerful sender and a probabilistic polynomial-time receiver.

A variant of OT is the 1-*out-of*-2 *Oblivious Transfer*, which was introduced by Even, Goldreich, and Lempel[11]. In this protocol, Alice has two strings s_0 and s_1 to send. The following two events are equally likely to occur at the end of an execution of the protocol: either Bob learns the string s_0, and doesn't get any information about s_1, or Bob learns the string s_1, and doesn't get any information about s_0. Moreover, at the end of the execution, Alice cannot tell with a non-negligible advantage which string Bob gets.

De Santis *et al*. presented a flavor of 1-*out-of*-2 OT, Public-Key Public-Randomness Oblivious Transfer (PKPR OT), which is based on quadratic residuosity assumption. In a *PKPR OT*, Alice and Bob share a random string σ and there are two channels used to transfer messages. A PKPR OT is a quadruple of algorithms (*Key_Generator*, *Verify*, *Send*, *Receive*), where *Key_Generator* and *Send* are efficient, and *Verify* and *Receive* are deterministic polynomial time. They should satisfy the following conditions:

1. Meaningfulness: If Alice and Bob act according to the protocol, Bob will get the message he wants to receive with a quite high probability.
2. Verifiability: If Bob generates the public file and private key correctly, the test whether the public file is valid can be passed with very high probability.
3. 1-*out-of*-2: There is no way for Bob to construct a public file that has a non-negligible probability of being declared valid by *Verify* and that allows Bob to obtain information on both strings sent by Alice.
4. Obliviousness: Alice has no way to guess better than at random that which of the two channels is open (in another word, Alice cannot know which of the two strings is received by Bob).

The detail of the implementation given by De Santis *et al*. is in Ref.[12]. They put forward an open problem in that paper, that is, how to use the same public file to transfer multi-messages at the end of each transfer, and from which channel a message *Bob* gets is independent. With respect to the problem, we present Independent Oblivious Transfer, whose detail is given in the next section.

3 Independent Oblivious Transfer

3.1 Independent oblivious transfer

We say an oblivious transfer is independent, if which of these two channels is open at each transfer is different and independent. Even if the adversary (usually Alice) knows Bob's choice during some or other transfer, without the knowledge of Bob's secret keys, he still cannot figure out Bob's choices during other transfers better than randomly guessing. Hence, which of the two channels is open to Bob at each OT is unpredictable for Alice. A PKPR IOT should satisfy not only the above four conditions which a PKPR OT should satisfy, but also the property of independence.

If we simply repeat the 1-*out-of*-2 *Oblivious Transfer* using the same public file, this method is only partially content to the requirements of the above open problem. However, the channel Bob gets messages from will be the same during each transfer. That is, Bob will either get all the messages from Ch_0 or learn all the messages from Ch_1. If Alice knows which channel Bob gets the message from during a certain 1-*out-of*-2 *OT*, it means that Alice will know all the choices of Bob during all the transfers, no matter before or after that. The probability that Alice can figure out successfully all the channels open to Bob during the whole multi OT's is 1/2, exactly the same with that in a single OT. That is, Alice can succeed in an all-or-nothing fashion. In our implementation which is presented in the following, the probability that Alice guesses successfully for all OT's that which of the two channels is open will be reduced to $1/2^k$, where k is the number of OT's. Even if Alice knows which channel Bob gets the message from in some or other 1-*out-of*-2 *OT*, he will have no better way than randomly guessing to know Bob's choices in

the 1-*out-of*-2 *OT*'s before or after that, supposed that Bob's private key is still secret to Alice. Of course, if Alice knows Bob's private key, he will know all Bob's choices with probability 1.

Another method to independently and obliviously transfer multi-messages is to use a single complete 1-*out-of*-2 *OT* for each pair of messages. Before transferring a pair of messages, Bob generates a new public file including new channels, and a new corresponding secret key. Alice then uses the new channels for oblivious transfer messages. Obviously, at each time which of the two channels is open will be different and independent, but the frequent changes of public files and secret keys will be costly and inconvenient. It will waste much bandwidth.

3.2 First non-interactive implementation

Here, we present a scheme within which we can independently and obliviously transfer polynomial pairs of messages using the same public file.

This implementation of independent oblivious transfer is non-interactive in the PKPR model, which is based on that of De Santis *et al*.[12]. It needs a non-interactive perfect zero-knowledge proof system (***A***,***B***) for the language of pairs (x,y), where x is a Blum integer and y is a quadratic non residue modulo x. (***A***,***B***) for moduli of length n needs a reference string σ of n^3 bits. Moreover, the program of the prover A can be performed in probabilistic polynomial time provided that the factorization of x is available. In our implementation, the GM encryption system[15] is used as well, which was also used in Ref.[12]. And we still follow the denotation GM_E, GM_D to denote the encryption and decryption of GM system.

The implementation of PKPR IOT consists of *Key_Generator*, *Verify*, *Send* and *Receive*. *Key_Generator* and *Verify* algorithms initialize the public file and *Send* and *Receive* algorithms actually perform the IOT. Algorithm *Receive* is the same with that in Ref.[12].

We denote by Alice the *Sender* and Bob the *Receiver* respectively. They share a reference string σ. Alice and Bob first agree on an integer k in advance, which denotes the number of messages to be transferred and is polynomial in n (otherwise, the whole process cannot be done in polynomial time). Bob then constructs two channels Ch_0, Ch_1, a validation *Val* and a secret key *Key* using the algorithm *Key_Generator* and publishes Ch_0, Ch_1, *Val* and $z_1,z_2,\ldots,z_k$. Bob can compute $z=z_1*z_2*\ldots*z_k \bmod x$ once and for all. We'll refer to a pair of channels along with its validation and $z_1,z_2,\ldots,z_k$ as the public file for PKPR IOT. In our implementation of PKPR IOT, algorithms *Verify*, *Send* and *Receive* can access the public file.

Before the first time to obliviously transfer a pair of messages to Bob, Alice verifies Bob's public file by checking that $Verify(\sigma,Ch_0,Ch_1,Val)=VALID$. This verification only needs to be done for once. Then Alice can directly send a pair of messages to Bob without validity check of Bob's public file. If he wants to send a pair of messages, s_0 and s_1, Alice randomly chooses a distinct j from $\{1,2,\ldots,k\}$ and computes and sends two processed messages $msg_0=Send(Ch_0,s_0,j)||j$ and $msg_1=Send(Ch_1,s_1,j)||j$. To retrieve one of the two messages, Bob first extracts j from the two processed messages, and checks that whether j is used before. If j was ever used, Bob rejects this transfer, and asks Alice to re-transfer the pair of messages. Else, he then computes $z'=z/z_j \bmod x$ and $b=QR_x(z')$. Then Bob can remove the last $|k|$ bits of msg_b, denote by msg'_b, and compute $Receive(Key, msg'_b)$. Followed are the formal descriptions of these algorithms.

Algorithm 1. *Key_Generator*(σ,k).

Input: An n^3-bit reference string σ and an integer k;

1. Construct channels and secret keys.

   Randomly select two n-bit primes p, $q\equiv 3 \bmod 4$, and set $x=p*q$.

   Randomly select $y\in NQR_x$ and $z_1,z_2,\ldots,z_k\in Z_x^{+1}$.

2. Construct *validation*.

Run the algorithm ***A*** on input (x,y) using the reference string σ and obtaining *Proof* as output.

3. Set $z=z_1*z_2*\ldots*z_k \bmod x$.

Set $Ch_0=(x,z*y \bmod x)$, $Ch_1=(x,z)$, $Val=(Proof,y)$, $Key=(p,q)$

Output: $(Ch_0, Ch_1, Key, Val, z_1,z_2,\ldots,z_k)$.

Algorithm 2. *Verify*(σ,Ch_0,Ch_1,Val).

Input: An n^3-bit reference string σ and a public key consisting of two channels $Ch_0=(x_0,v_0)$, $Ch_1=(x_1,v_1)$ and a validation $Val=(Proof, y)$;

1. Verify that $v_1=z_1*z_2*\ldots*z_k \bmod x_0$, $x_0=x_1$, $v_0=v_1*y \bmod x_0$ and $y,v_0\in Z_x^{+1}$;

2. Run algorithm ***B*** on input (x_0, y), the reference string σ, and the string Proof.

Output: If all checks are successfully passed, then output VALID, else INVALID.

Algorithm 3. *Send*(Ch,s,j).

Input: A channel $Ch=(x,v)$, a binary string s and an integer $j(1\le j\le k)$;

Output: $msg=GM\_E(x,v/z_j \bmod x,s)||j$.

Algorithm 4. *Receive*(Key,msg).

Input: A key $Key=(p,q)$ and a message *msg*;

Output: $s=GM\_D(msg,p,q)$.

Now, we are going to analyze the security and efficiency of this non-interactive implementation of independent oblivious transfer.

3.2.1 Security

Theorem 1. Under the Quadratic Residuosity Assumption, the above quadruple of algorithms (*Key_Generator*, *Verify*,*Send*,*Receive*) is a PKPR Oblivious Transfer.

Note: The proof of Theorem 1 is similar to that in Ref.[12]. For the sake of clarity, we give a whole proof here, including the parts that are the same with those in Ref.[12].

Proof: First of all, the above algorithms all run in polynomial time. Notice that, in the algorithms *Key_Generator* and *Receive*, the factorization of x is known.

Meaningfulness: Now, as y is a quadratic non-residue, then by Fact 2, exactly one of z and $zy \bmod x$ is a quadratic non-residue and thus, by the properties of the encryption scheme, the corresponding string s_i will be read by the receiver.

Verifiability: If the two channels are constructed by the *Key_Generator* algorithm, then x is a Blum Integer, each of $z_1,z_2,\ldots,z_k$ has Jacobi Symbol +1 modulo x and y is a quadratic non-residue modulo x. Since $z=z_1*z_2*\ldots* z_k \bmod x$, according to Fact 2, z has Jacobi Symbol +1 as well. Thus it follows from the completeness of (***A***,***B***) that *Verify* will output VALID with overwhelming probability.

1-*out-of*-2: Suppose that there exists an algorithm that violates the 1-*out-of*-2 condition. Two cases need to be considered. If y is a quadratic non-residue modulo x and x is a Blum integer, then, by Fact 2, one of z and zy is a quadratic residue modulo x. Thus, for the properties of the encryption scheme GM_E, no information about the string encrypted with a quadratic residue can be obtained from its encryption.

Suppose now that y is a quadratic residue modulo x or that x is not a Blum integer. In this case, for the soundness of the proof system (***A***,***B***), it follows that with a high probability, the algorithm *Verify* will output INVALID.

Obliviousness: Though Alice knows $z_1,z_2,\ldots,z_k$, by Quadratic Residuosity Assumption and Fact 3, he cannot know the quadratic residuosities of $z_1,z_2,\ldots,z_k$ and $z'=z/z_j \bmod x$. Therefore Alice cannot know which of the two channels is open in a PKPR IOT.

Now we prove that if there exists an efficient algorithm $Adv(\sigma,Ch_0,Ch_1,Val,k)$ which, for some $d>0$ and infinitely many n, violates the obliviousness condition, we can construct an algorithm which contradicts the QRA.

As (A,B) is a non-interactive perfect zero-knowledge proof system, there exists an efficient simulator algorithm S for (A,B). On inputing a pair (x,y), where x is a Blum integer and y is a quadratic non-residue modulo x, S generates a pair $(\sigma,Proof)$ where σ is a random string of n^3 bits and $Proof$ has the same distribution of $\boldsymbol{A}$'s output on the inputs σ and (x,y).

We exhibit an algorithm $Q(\cdot,\cdot)$ that decides quadratic residuosity using Adv and S as subroutines. The following is the formal description of $Q(\cdot,\cdot)$.

Algorithm 5. $Q(x,z)$.

Input: $x\in BL(n)$, $z\in Z_x^{+1}$;

Construct Channels and Validations.

Randomly choose y in NQR_x;

Randomly choose $k=poly(n)$;

Randomly choose $z_1,z_2,\ldots,z_{k-1}$ in Adv;

Run algorithm S on input (x,y) obtaining a pair $(\sigma,Proof)$ as output;

Set $z=z_1*z_2*\ldots*z_{k-1} \bmod x$;

Set $Ch_0=(x,z'*y \bmod x)$, $Ch_1=(x,z')$ and $Val=(Proof,y)$;

Set $b'=Adv(\sigma,Ch_0, h_1,Val,k)$;

If k is odd, set $b=b'$; else set $b=1-b'$.

Output: b.

Next we are going to compute $Pr[x\leftarrow BL(n);k\leftarrow poly(n);z\leftarrow Z_x^{+1}:Q(x,z,k)=QR_x(z)]$. Suppose that $QR_x(z)=1$. Thus any string sent using channel Ch_1 will be received by Bob. As Adv guesses b with probability at least $1/2+n^{-d}$, in this case, since $z_1,z_2,\ldots,z_{k-1}$ are quadratic non residue modulo x, if k is odd, then the quadratic residue of z modulo x is the same as that of z'; otherwise, the quadratic residue of z modulo x is just opposite to that of z'. So the algorithm Q will return the correct quadratic residue of z modulo x with a probability the same as that of Adv. The same reasoning applies to the case $QR_x(z)=0$. Therefore, for some $d>0$ and infinitely many n,

$$Pr[x\leftarrow BL(n);k\leftarrow poly(n);z\leftarrow Z_x^{+1}:Q(x,z,k)=QR_x(z)]\geq 1/2+n^{-d},$$

which contradicts the Quadratic Residuosity Assumption. □

Next, we prove that this implementation satisfies the independence property.

Theorem 2. The above quadruple of algorithms (*Key_Generator*,*Verify*,*Send*,*Receive*) is an Independent Oblivious Transfer.

Proof: When Alice sends a pair of messages, he randomly selects a distinct j from the set $\{1,2,\ldots,k\}$, and then computes $z'=z/z_j \bmod x$. Since j is different at each time, z' is different as well. Therefore, the channel from which Bob gets message is different. Because which of the two channels is open depends on the quadratic residuosity of z' and $z_1,z_2,\ldots,z_k$ are independently chosen, from which channel Bob gets a message at each time will be independent as well. □

Combined the above two theorems, we can say our implementation is really a PKPR IOT.

There is one thing we need to say more. This implementation is secure against receivers with unlimited computational power. Since y is a quadratic non-residue modulo x, by Fact 3, we have that either $z*y$ or z is a quadratic residue modulo x. It also holds for $z*y/z_j$ and z/z_j. In a GM bit encryption $GM\_E(x,v,c)$, where c is a bit to be encrypted, the encryption of c is $r^2v^c \bmod x$, where r is a randomly selected number in Z_x^*. If v is a quadratic residue modulo x, no matter what c is, the encryption of c is a quadratic residue modulo x. The encryption of bit 0

and the encryption of bit 1 are in the same distribution, no one can distinguish them, no matter how powerful he is.

3.2.2 Efficiency

Here we are going to discuss the efficiency of our implementation of IOT. We compare our implementation with that of PKPR OT in Ref.[12].

The *Key_Generator* step requires additional $k-1$ chooses of z_i and the computation of z, which can be done in polynomial time. The *Verify* step requires additional verification of the validity of z. The other verifications are the same with those of Ref.[12]. Both the two steps are a little more costly than those of Ref.[12], but they are in a once-and-for-all fashion. During the following k transfers, both parties can directly use the public file and secret key which are generated during this step, and Alice doesn't need to verify Bob's public file before every transfer.

While sending a pair of messages, what Alice needs to do additionally is to choose a distinct j from $\{1,2,\ldots,k\}$. When receiving a message, what Bob needs to do additionally is to extract a j from the processed messages and check that whether this j is ever used before. There are efficient algorithms to do these. The additional workload is much lower than that of regenerating a new public file and corresponding secret key. Our implementation saves a certain bandwidth of communication.

With the above discussion, we can say that our implementation of PKPR IOT is more efficient than simply using a single complete 1-*out-of*-2 *OT* for each pair of messages.

This implementation of PKPR IOT can be used to oblivious transfer multi-messages independently, but it is limited to polynomial times and it's only partially content to our need. Hence, we present another non-interactive implementation, within which we can obliviously transfer messages for any times independently. In the implementation, the sender is required to be honest and behave according to the protocol.

3.3 Second non-interactive implementation—Honest-Sender PKPR IOT

This implementation implements to obliviously transfer any number of messages independently between the two parties. In this independent oblivious transfer, we require that the sender is honest and behaves totally in accordance with the protocol, so we also call it *honest-sender* IOT.

Key Generation and algorithm *Verify* are exact the same with those in Ref.[12]. The public key of Bob is z, whose quadratic residuosity b is unknown to Alice. Bob's private key is the factorization of x, p and q. In order to send message s_0 and s_1 to Bob through channels (x,v_0) and (x,v_1), Alice first randomly selects r from Z_x^{+1}, and sets $v_0'=v_0*r \bmod x$, $v_1'=v_1*r \bmod x$. Then She uses GM encryption procedure with input x, v_0', s_0 and x, v_1', s_1 for each channel. After that, Alice sends the encryptions and r through the channels. In order to receive a message, Bob first extracts r from the messages he received and computes $c=b\oplus QR_x(r)$. Then he can get the message s_c by using GM decryption procedure to decrypt the cipher text he received from channel Ch_c. The detail of each algorithm is as follows.

Algorithm 6. *Key_Generator*(σ,k).

Input: An n^3-bit reference string σ and an integer k;

1. Construct channels and secret keys.

   Randomly select two n-bit primes $p,q \equiv 3 \bmod 4$, and set $x=p*q$.

   Randomly select $y \in NQR_x$ and $z \in Z_x^{+1}$.

2. Construct validation.

   Run the algorithm $\boldsymbol{A}$ on input (x,y) using the reference string σ and obtaining Proof as output.

3. Set $Ch_0=(x,z*y \bmod x)$, $Ch_1=(x,z)$, $Val=(Proof,y)$, $Key=(p,q)$, $b=QR_x(z)$

Output: (Ch_0,Ch_1,Key,Val,b).

Algorithm 7. *Verify*(σ,Ch_0,Ch_1,Val).

Input: An n^3-bit reference string σ and a public key consisting of two channels $Ch_0=(x_0,v_0)$, $Ch_1=(x_1,v_1)$ and a validation $Val=(Proof,y)$;

1. Verify that $x_0=x_1$, $v_0=v_1*y \bmod x_0$ and $y,v_0\in Z_x^{+1}$,

2. Run algorithm B on input (x_0,y), the reference string σ, and the string Proof.

Output: If all checks are successfully passed, then output VALID, else INVALID.

Algorithm 8. *Send*(*Ch*,*s*).

Input: A channel $Ch=(x,v)$, a binary string s;

Randomly select $r\in Z_x^{+1}$, and set $v'=v*r \bmod x$

Output: $msg=GM\_E(x,v',s)\|r$.

Algorithm 9. *Receive*(*Key*,*msg*).

Input: A key $Key=(p,q)$ and a message *msg*;

Output: $s=GM\_D(msg,p,q)$.

3.3.1 Security

The security analysis is similar to that of the first non-interactive implementation. *Meanfulness* and *Verification* can easily be checked. If Alice is honest, she randomly selects r from Z_x^{+1}. By Fact 3, we know that only one of $z*r$ and $z*y*r$ is quadratic non-residue modulo x. Hence, Bob can only get one of the two messages, for he knows p and q, the factorization of x. 1-*out-of*-2 property is satisfied. Since the factorization of x is unknown to her, according to quadratic residuosity assumption, Alice could not know the quadratic residuosity of z better than randomly guessing, as well as the quadratic residuosity of r. So, she could not know which channel is open to Bob with non-negligible advantage. This satisfies the *obliviousness* condition. During each transfer, r is independently and randomly chosen by Alice, therefore, $QR_x(r)\oplus b$ differs each time. *Independence* property is also met. As the same with the first implementation, this scheme is also secure against receivers with unbounded power.

3.3.2 Efficiency

The efficiency is almost the same with that of Ref.[12], except one random selection, one multiplication and one quadratic residuosity computation plus for each transfer. Hence, we can say, this implementation is almost as efficient as that of Ref.[12].

3.4 Interactive implementation

The above second non-interactive implementation has its own advantage than the first one, but still, there is some drawbacks with it. First, the sender Alice needs to be honest and behave totally according to the protocol. Otherwise, Alice may suit herself to choose r. Similarly, she selects a random number w from Z_x^{+1} and sets $r=w^2$ or $r=-w^2$. In this case, Alice knows the quadratic residuosity of r. And in each transfer, she chooses such a r with the same quadratic residuosity, thus, if she knows which channel is open to Bob in some or other transfer, it means that Alice could know all Bob's choices, which is the same with the general oblivious transfer. Another shortcoming is that, Bob could not choose which message to receive totally according to his will. His choice depends on not only z whose quadratic residuosity Bob knows, but also r Alice selects. Alice's selection imposes on Bob's choice. With respect to the above considerations, we present the third implementation, which is interactive, and is based on the BBCS oblivious transfer protocol of Ref.[16].

The BBCS oblivious transfer protocol relies on a third party, Ted, who is trusted by both Alice and Bob. It includes three steps: SETUP, REQUEST and REPLY. In SETUP step, Ted privately gives Alice two random k-bit strings r_0, r_1, provided that the length of a message is k-bit. Then he flips a bit d, and privately gives Bob d and r_d. Now Ted's work is all done. In the REQUEST step, Bob determines a bit c, which means that he wants to obtain m_c.

He privately sends Alice the bit $e=c\oplus d$. Then in the REPLY step, assumed that the messages Alice wants to transfer are m_0 and m_1, Alice replies Bob by privately sending Bob the values $f_0=m_0\oplus r_e$ and $f_1=m_1\oplus r_{1-e}$. At this time, Bob gets m_c by computing $m_c=f_c\oplus r_d$.

It is clear that Alice has no information about c, and Bob knows no information about m_{1-c}. But the main disadvantage of this protocol is that a trusted third party is involved. Usually, a third party trusted by both parties is not easy to find. It is better if the SETUP step can be done only between Alice and Bob. Alice and Bob communicate, without any other party, to setup their common and private parameters.

In order to make the protocol work in public-key public-randomness model, we substitute the original SETUP step with a PKPR oblivious transfer between Alice and Bob. In the new SETUP step, Alice randomly selects two strings r_0, r_1, Bob flips a bit d. Then, Alice and Bob start a PKPR oblivious transfer. Any PKPR OT can be used here. In our implementation, we prefer the PKPR OT of Ref.[12]. Alice uses the two channels to transfer r_0, r_1 to Bob, and Bob receives r_d. At this time, Alice knows r_0, r_1, and Bob owns d and r_d. Formal description is in Fig.1.

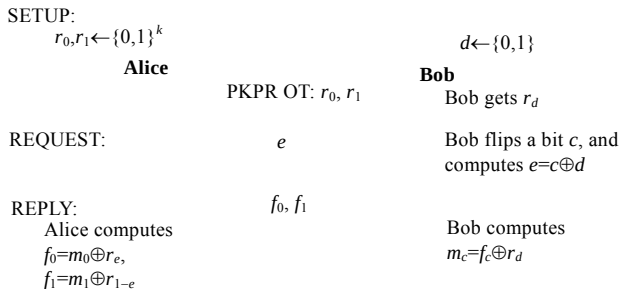

Fig.1 PKPR IOT based on BBCS OT of Ref.[16]

3.4.1 Security

Given that the PKPR oblivious transfer used in this scheme is secure, we can say, this scheme is also secure. After the PKPK OT, Bob knows r_d, but he doesn't have any information about r_{1-d}; Alice does not get any information about d. After receiving e from Bob, she could not know c with any advantage, and c is theoretically secure to her. Also, e doesn't leak any information concerning d to Alice. As Bob knows only one of r_0, r_1, he can merely get m_c from f_c. With regard to m_{1-c}, he could get no information about it, for he doesn't know r_{1-d}. In each transfer, c is randomly chosen by Bob on his own, so Bob's choice is independent of that in any other transfer. Supposed that Bob's secret is kept private, even if Alice knows the c in some transfer, all the c's in other transfers are still safe for Bob. Obliviously, this interactive scheme is secure against receivers with unbounded power as well.

3.4.2 Efficiency

The SETUP step could be done in a once-for-all fashion. It is the initialization of the whole independent oblivious transfer scheme. Once it is done, the parameters can be used for all the later message transferring. In each transfer, there are only four exclusive or operations. Thus, we say, this scheme is rather more efficient than the above two schemes.

4 Conclusion

In this paper, we define an independent oblivious transfer and present three implementations of IOT in the PKPR Model, two non-interactive and one interactive. The first non-interactive implementation can be used to independently and obliviously transfer messages for pre-fixed polynomial times. It only satisfies our need partially. The second non-interactive implementation can satisfy our need of oblivious transferring for any number of times,

whereas, the sender is required to be honest and the receiver could not make his choice totally on his own. Then, we present the third implementation which is interactive. The two parties cooperate to initialize the independent oblivious transfer scheme first, and then they can interactively and obliviously transfer messages for any times they wish. Compared with the OT scheme in Ref.[12], our first two implementations are at least almost the same efficient, and the third scheme is much more efficient than that in Ref.[12]. Besides, all the three schemes own the additional property of independence.

References:

[1] Rabin MO. How to exchange secrets by oblivious transfer. Technical Report, TR-81, Harvard, 1981.

[2] Goldreich O, Micali S, Wigderson A. How to play any mental game or a completeness theorem for protocols with honest majority. In: Proc. of the 19th Annual ACM Symp. on Theory of Computing (STOC). New York: ACM Press, 1987. 218−229.

[3] Killian J. Founding cryptography on oblivious transfer. In: Proc. of the 20th Annual ACM Symp. on Theory of Computing (STOC). 1988. 20−31.

[4] Bellare M, Micali S. Non-Interactive oblivious transfer and applications. In: Brassard G, ed. Advances in Cryptology—CRYPTO'89. LNCS 435, Springer-Verlag, 1989. 547−557.

[5] Brassard G, Crépeau C, Robert JM. Information theoretic reductions among disclosure problems. In: Proc. of the 27th Annual Symp. on Foundations of Computer Science. IEEE, 1986. 168−173.

[6] Brassard G, Crépeau C, Robert JM. All-or-Nothing disclosure of secrets. In: Advances in Cryptology—CRYPTO'86. LNCS 263, Springer-Verlag, 1987. 234−238.

[7] Cachin C. On the foundations of oblivious transfer. In: Advances in Cryptology—EUROCRYPT'98. LNCS 1403, Springer-Verlag, 1998. 361−374.

[8] Garay J, Mackenzie P. Concurrent oblivious transfer. In: Proc. of the FOCS 2000. 2000. 314−324.

[9] Crépeau C. Verifiable disclosure of secrets and applications. In: Proc. of the Eurocrypt'89. LNCS 434, 1990. 181−191.

[10] Naor M, Pinkas B. Oblivious transfer and polynomial evaluation. In: Proc. of the 31st Annual ACM Symp. on Theory of Computing. 1999. 245−254.

[11] Even S, Goldreich O, Lempel A. A randomized protocol for signing contracts. Communications of the ACM, 1985, 28: 637−647.

[12] De Santis A, Di Crescenzo G, Persiano G. Zero-Knowledge arguments and public-key cryptography. Information and Computation, 1995,121(1):23−40.

[13] Goldwasser S, Micali S, Rackoff C. The knowledge complexity of interactive proof-systems. In: Proc. of the 17th Annual ACM Symp. of Theory of Computing. New York: ACM Press, 1985. 291−304.

[14] Blum M, De Santis A, Micali S, Persiano G. Non-Interactive zero knowledge. SIAM Journal on Computing, 1991,20(6): 1084−1118.

[15] Goldwasser S, Micali S. Probabilistic encryption. Journal of Computer and System Sciences, 1984,28(2):270−299.

[16] Rivest RL. Unconditionally secure commitment and oblivious transfer schemes using private channels and a trusted initializer. Manuscript, 1999. http://theory.lcs.mit.edu/~/rivest/publications.html

HUANG Qiong was born in 1982. He is a graduate student at the Department of Computer Science and Engineering, Fudan University. His current research areas are cryptology and information security.

ZHAO Yi-Ming is an associate professor at the Computer Science and Engineering Department, Fudan University and a CCF senior member. His research areas are cryptology and information security.