Efficient reduction of 1 out of n oblivious transfers in random oracle model ⋆

Bao Li[1], Hongda Li[1],Guangwu Xu[2] and Haixia Xu[1]

[1]*State Key Laboratory of Information Security, Graduate School Chinese Academy of Sciences, 100039, Beijing, China*
lb@is.ac.cn
[2]*Department of Electrical Engineering and Computer Science University of Wisconsin-Milwaukee, Milwaukee, WI 53211, USA*
gxu@comm.utoronto.ca

Abstract

We first present a protocol which reduces 1-out-of-n oblivious transfer OT_l^m to 1-out-of-n oblivious transfer OT_m^k for $n > 2$ in random oracle model, and show that the protocol is secure against malicious sender and semi-honest receiver. Then, by employing a cut-and-choose technique, we obtain a variant of the basic protocol which is secure against a malicious receiver.

Key words: oblivious transfer, secure reduction

1 Introduction

Oblivious transfer (OT) is a fundamental cryptographic primitive that may be used in secure multi-party computations and a wide variety of other cryptographic protocols. It is still an open problem if OT can be based on one-way functions. Partial answers to this problem were given [1]. Impagliazzo and Rudich [1] showed that a black-box reduction from oblivious transfer to a one-way function (or a one-way permutation) would imply P≠NP. It is even not known whether a non-black-box reduction from oblivious transfer to one-way functions exists.

⋆ Supported by the National Natural Science Foundation of China under Grant No. 90304013 and the National High-Tech Research and Development Plan of China under Grant No. 2003AA144151.

However, a result of Beaver [2] shows that one-way functions are sufficient to extend a few oblivious transfers into many. By this result, one could implement a few oblivious transfers using, for example, public-key primitives[3–7], physical methods or multi-party computations, and then use one way functions for the rest. Unfortunately, Beaver's reduction is inherently non-black-box with respect to the one-way function it uses and appears to be inefficient in practice.

In [8], an efficient and simpler protocol of extending oblivious transfers was proposed in the random oracle model in black-box way. However, the reduction in [8] is only applied to the type of 1-out-of-2 OT. It is natural to ask whether similar things can be done to the type of 1-out-of-n OT and even more to the more general type of k-out-of-n OT?

In this paper, we answer the above questions by constructing a new protocol which is a modification and generalization of the protocol in [8]. The new protocol is shown to be secure for arbitrary sender and semi-honest receiver. The protocol may be modified so that it is secure against a malicious receiver by applying a cut-and-choose technique similar to that used in [8].

Convention: We use capital letters to denote matrices and the corresponding small letters to denote the entries and the boldfaced letters to denote the rows and columns. E.g., if we use U to denote a matrix, we use u_{ij} to denote the entry located in ith row and jth column, and $\mathbf{u}_i$ the ith row and $\mathbf{u}^j$ the jth column of matrix U.

This paper is organized as follows. In section 2, we briefly introduce oblivious transfers within the framework of secure two party computation, random oracle, and black-box reduction. In section 3, by using random oracle, we propose a protocol which securely reduces 1 out of $n\,(n > 2)$ oblivious transfers with parameters l and m to 1 out of n oblivious transfers with parameters m and k efficiently in the semi-honest model. In section 4, we modify the protocol presented in section 3 and obtain a new protocol which is secure against a malicious receiver. We conclude our paper in section 5 by summarizing what we have done.

2 Oblivious Transfer Protocols

We view oblivious transfer protocols within the more general framework of secure two-party computation. We assume readers' familiarity with standard simulation-based definitions of secure computation from the literature. For more details, please refer to [9,10].

Secure two-party computation. A secure two-party computation task is specified by a two-party functionality, i.e., a function mapping a pair of inputs to a pair of outputs. A protocol is said to securely realize the given functionality if an adversary attacking a party in a reallife execution of the protocol can achieve no more than it could have achieved by attacking the same party in an ideal implementation which makes use of a trusted party. In particular, the adversary should not learn more about the input and output of the uncorrupted party than it must inevitably be able to learn. This is formalized by defining a real process and an ideal process, and requiring that the interaction of the adversary with the real process can be simulated in the ideal process.

Oblivious transfers. An OT protocol can be defined as a secure two-party protocol between a sender $\boldsymbol{S}$ and a receiver $\boldsymbol{R}$ realizing the OT functionality. The OT functionality discussed in this paper is defined in section 3.

The use of a random oracle. There is an augmentation to the standard model of secure two-party computation with a random oracle H in the construction of protocol. The random oracle is assumed to be a function with the property that if a value in its domain is not queried before then the corresponding function value is a random value, i.e., can not be predicted. Accordingly, the simulation in the ideal process is required to be indistinguishable from the real execution even if the distinguisher is allowed to adaptively make polynomially many queries to the same H that was used in the process (real or ideal) generating its input.

Reductions. A functionality f is said to be securely reduced to a functionality g, if a protocol for securely computing f can be designed by using the functionality g. This protocol may be viewed as a secure reduction from f to g. Composition theorems, e.g. from [9], guarantee that the call to g can be replaced by any secure protocol realizing g, without violating the security of the high level protocol. Moreover, these theorems relativize to the random oracle model. Thus, it suffices to formulate and prove our reductions using the hybrid model where the parties in the high level protocol are allowed to invoke g, i.e., a trusted party to which they can securely send inputs and receive the corresponding outputs.

Black-box reductions. In our construction of protocol, the reduction we use is black-box reduction, i.e., the protocol and its security proof do not depend on the implementation of low level primitive OT_m^k. For comparison, Beavers reduction [1] is clearly non-black-box with respect to the one-way function (or PRG) on which it relies. We refer the reader to [11,12] for a more thorough and general exposition to black-box reductions in cryptography.

3 Reduction from $\mathbf{OT}_l^m$ to $\mathbf{OT}_m^k$: 1 out of n, $n > 2$

The OT primitive we consider, denoted OT_l^m, realizes m 1-out-of-n oblivious transfers of $l-$bit strings. The OT_l^m functionality is as follows.

Inputs: $\boldsymbol{S}$ holds m $n-$tuples $(x_{11}, \cdots, x_{1n}), \cdots, (x_{m1}, \cdots, x_{mn})$ of $l-$bit strings. $\boldsymbol{R}$ holds m selection numbers $\mathbf{r} = (r_1, \cdots, r_m)$, $1 \le r_j \le n$.

Outputs: $\boldsymbol{R}$ outputs x_{jr_j} for $1 \le j \le m$. $\boldsymbol{S}$ has no outputs.

As indicated in [8], the task of extending oblivious transfers can be defined as that of reducing OT_l^m to OT_k^k, due to the possibility for a more efficient direct implementation of the primitive OT_k^k than via k separate applications of OT_k^1 [3], where k is a security parameter and $m > k$. And the critical reduction is from OT_l^m to OT_m^k. Hence, we will only present a protocol of reducing OT_l^m to OT_m^k and arrive a general conclusion for the reduction of OT.?

We express the number r_j by a bit vector $\mathbf{p}_j$ where the r_jth element is 1 and the others are all 0. In this way, the selection vector $\mathbf{r}$ corresponds to a $m \times n$ matrix P. In the following construction we suppose that the receiver hold the selection matrix P instead of the selection numbers $\mathbf{r}$.

Now we give the protocol of reduction for 1 out of n OT.

Protocol 3.1 (Reduction from OT_l^m to OT_m^k)

INPUT OF $\boldsymbol{S}$: m $n-$tuples $(x_{11}, \cdots, x_{1n}), \cdots, (x_{m1}, \cdots, x_{mn})$ of $l-$bit strings.

INPUT OF $\boldsymbol{R}$: $m \times n$ selection matrix P satisfying that there is one and only one entry equal to 1 in each row, where $p_{ji} = 1$ means $\boldsymbol{R}$ receives ith string out of jth $n-$tuples $(x_{j1}, \cdots, x_{jn})$.

COMMON INPUT: a security parameter k.

ORACLE: a random oracle $H : [m] \times [n] \times \{0,1\}^k \to \{0,1\}^l$.

CRYPTOGRAPHIC PRIMITIVE: an ideal 1 out of n OT_m^k primitive.

(1) $\boldsymbol{S}$ initializes a random $k \times n$ matrix U satisfying that there is one and only one entry equal to 1 in each row, and $\boldsymbol{R}$ initializes a random $m \times k$ bit matrix T.
(2) The parties invoke the 1 out of n OT_m^k primitive: $\boldsymbol{S}$ acts as a receiver with input $k \times n$ matrix U and $\boldsymbol{R}$ as a sender with inputs $(\mathbf{p}^1 \oplus \mathbf{t}^i, \cdots, \mathbf{p}^n \oplus \mathbf{t}^i), 1 \le i \le k$. Let Q be the $m \times k$ matrix of values received by $\boldsymbol{S}$. (Note

that $\mathbf{q}^i = u_{i1} \cdot \mathbf{p}^1 \oplus \cdots \oplus u_{in} \cdot \mathbf{p}^n \oplus \mathbf{t}^i, \mathbf{q}_j = p_{j1} \cdot \mathbf{u}^{1T} \oplus \cdots \oplus p_{jn} \cdot \mathbf{u}^{nT} \oplus \mathbf{t}_j$, where $\mathbf{u}^{iT}$ denote the transpose of the column vector $\mathbf{u}^i$.)

(3) ***S*** queries the random oracle H by $(j, i, \mathbf{q}_j \oplus \mathbf{u}^{iT}), 1 \leq i \leq n, 1 \leq j \leq m$. Let $H(j, i, \mathbf{q}_j \oplus \mathbf{u}^{iT}), 1 \leq i \leq n, 1 \leq j \leq m$, be the reply from the random oracle H.

(4) ***S*** sends $y_{ji} = x_{ji} \oplus H(j, i, \mathbf{q}_j \oplus \mathbf{u}^{iT}), 1 \leq i \leq n, 1 \leq j \leq m$.

(5) ***R*** outputs $z_j = y_{ji} \oplus H(j, i, \mathbf{t}_j)$ whenever $p_{ji} = 1$.

CORRECTNESS. It is easy to verify that the outputs of Protocol 3.1 are correct, i.e., $z_j = x_{ji}$ when both parties follow the protocol.

SECURITY. We will show that Protocol 3.1 is secure against a malicious sender and a semi-honest receiver. Specifically, we will show a perfect simulator for any malicious sender $\boldsymbol{S}^*$ and a statistical simulator for ***R***. In the latter case, the output of the ideal process involving the simulator is indistinguishable from that of the real process even if the distinguisher is allowed to adaptively make $e^{\frac{k}{n}}/k^{\omega(1)}$ additional calls to H.

All we need to do in proving the security of Protocol 3.1 is construct two simulators ***S'*** and ***R'*** according to the definition of security for protocols given in [10].

Simulator *S'*. We construct simulator ***S'*** as follows.

- ***S'*** randomly choose a string λ' as its random input.
- ***S'*** randomly generate a $k \times n$ matrix U' satisfying that there is one and only one entry equal to 1 in each row.
- ***S'*** generate a random $m \times k$ matrix Q' and take the columns of Q' as the reply from the OT_m^k primitive.
- ***S'*** queries the random oracle H by $(j, i, \mathbf{q}'_j \oplus \mathbf{u}'^i), 1 \leq i \leq n, 1 \leq j \leq m$. Let $H(j, i, \mathbf{q}'_j \oplus \mathbf{u}'^i), 1 \leq i \leq n, 1 \leq j \leq m$, be the reply from the random oracle H.
- ***S'*** output all the above values.

Theorem 3.1 *Protocol 3.1 is perfectly secure with respect to an arbitrary sender.*

PROOF. In a real execution of Protocol 3.1, the view of the sender $\boldsymbol{S}^*$ is composed of the input $(x_{11}, \cdots, x_{1n}), \cdots, (x_{m1}, \cdots, x_{mn})$, the random input, the matrix Q, i.e., the message received from the OT_m^k and the message $H(j, i, \mathbf{q}_j \oplus \mathbf{u}^{iT}), 1 \leq i \leq n, 1 \leq j \leq m$, received from the random oracle H. Comparing the view of the sender $\boldsymbol{S}^*$ in the real execution and the output of the simulator ***S'*** term by term we can conclude that the two distributions are identical. In the real execution the outputs of the receiver ***R*** are

$z_j = y_{ji} \oplus H(j, i, \mathbf{t}_j)$ where the indices satisfying $p_{ji} = 1$, and the output of the 1 out of n OT_l^m functionality are x_{ji} whenever $p_{ji} = 1$. From the correctness of Protocol 3.1 these two outputs are equal. According to the definition for protocol's security in [10], we have proven the security of Protocol 3.1, and the simulator ***S'*** is perfect. Note that the above simulation remains perfect even if the distinguisher makes an arbitrary number of calls to H.

In the sense that even if the sender $\boldsymbol{S}^*$ violates the protocol maliciously he still can not get any information about the receiver's inputs yet, Protocol 3.1 is said to be secure with respect to a malicious sender. □

Simulator *R'*. Simulator ***R'*** is constructed as follows.

- ***R'*** invokes Protocol 3.1, where the inputs of ***S*** is given by substituting the values z_j for the known inputs x_{ji} and 0^l for the unknown inputs of ***S***, and the inputs of ***R*** is given the selection matrix P and the random input of ***R*** in the real execution.
- ***R'*** outputs the entire view of ***R***.

Theorem 3.2 *As long as $n = o(k)$ and $n > 2$, Protocol 3.1 is secure with respect to a semi-honest receiver and a polynomial-time distinguisher having access to the random oracle.*

PROOF. According to the definition for protocol's security given in [10] we need to show that for fixed inputs of ***S*** and ***R*** the view of ***R*** in the real execution is indistinguishable from the output of the simulator ***R'***. If we check the description of Protocol 3.1 step by step, we will see that the most problematic issue is the values of the random oracle H used for masking the values x_{ji} which are unknown to the receiver. But the values of H used in Step (4) of Protocol 3.1 are uniformly random and independent of the receiver's view and independent of each other. Thus, it is clear that the output of the simulator ***R'*** is identically distributed to the output of ***R*** in the real execution.

However, to make a meaningful security statement in the random oracle model, we must allow the distinguisher to make additional calls to H.

Specifically, after given the view of ***R*** in the real execution and the output of the simulator ***R'***, the distinguisher can query the random oracle H many times and apply the reply from H to distinguish the above two random variables.

If ***R'*** can know definitely that some $x_{js} \neq 0$ with $p_{js} \neq 1$, then the distinguisher can tell that the view is from the real process, and therefore can distinguish the real process from the ideal process.

Since we suppose H to be a random oracle, it seems that there is no problem in obtaining any information about x_{js} with $p_{js} \neq 1$. The problem arises only when the distinguisher can "guess" the oracle query used in the real process to mask some secret x_{js}, $p_{js} \neq 1$. Since $\boldsymbol{S}$ queries the random oracle H by $(j, s, \mathbf{q}_j \oplus \mathbf{u}^s)$, and $\mathbf{q}_j \oplus \mathbf{u}^s = \mathbf{u}^i \oplus \mathbf{u}^s \oplus \mathbf{t}_j$ with $p_{ji} = 1$ and $\boldsymbol{R}$ knows $\mathbf{t}_j$, the problem is reduced to the probability that the distinguisher guesses $\mathbf{u}^i \oplus \mathbf{u}^s$ for some $1 \leq s \leq n$, $s \neq i$. For $n > 2$, this probability is almost the same as that the distinguisher guesses $\mathbf{u}^s$ for some $1 \leq s \leq n$. Since

$$\Pr[\mathbf{u}^s = \alpha] = \frac{(n-1)^{k-w}}{n^k}, \qquad \text{for each } \alpha \text{ including } w \text{ } 1,$$

the probability that the distinguisher guesses $\mathbf{u}^s$ to be α is $\frac{(n-1)^{k-w}}{n^k} \leq (1 - \frac{1}{n})^k$, and is less than $e^{-\frac{k}{n}}$.

So, even if the distinguisher makes $\frac{e^{\frac{k}{n}}}{k^{\omega(1)}}$ additional calls to H, the probability that the distinguisher guesses the value of $\mathbf{u}^s$ is less than $\frac{e^{\frac{k}{n}}}{k^{\omega(1)}} \times e^{-\frac{k}{n}} \leq \frac{1}{k^{\omega(1)}}$. Whereas $\frac{e^{\frac{k}{n}}}{k^{\omega(1)}}$ is greater than any polynomial in k when $k \to \infty$ under the condition $n = o(k)$.

On the other hand, as long as the distinguisher does not guess such a critical query, the masks remain random and independent given its view, and so indistinguishability is remained. It follows that the probability that the distinguisher wins is negligible. □

4 A Secure Protocol against Malicious Receiver

By employing a cut-and-choose technique similar to that used in [8], we may modify Protocol 3.1 to obtain a protocol which is secure against a malicious receiver. The scheme can be described informally as follows.

Let σ denote a statistical security parameter. The players engage in σ (parallel) executions of the previous protocol, where all inputs to these executions are picked randomly and independently of the actual inputs. Next, the sender challenges the receiver to reveal its private values for a random subset of $\sigma/2$ executions, and aborts if an inconsistency is found. This ensures $\boldsymbol{S}$ that except with $2^{-\Omega(\sigma)}$ probability, the remaining $\sigma/2$ executions contain at least one good execution where the receiver was well-behaved in the above sense. Finally, the remaining executions are combined as follows. Based on its actual selection numbers (matrices), the receiver sends a correction number for each of its $m\sigma/2$ random selections in the remaining executions, telling $\boldsymbol{S}$ to shift the choice of the input strings. For each of its actual secrets x_{ji}, the sender sends the exclusive-or of this secret with the $\sigma/2$ (random) inputs of the remaining executions which correspond to x_{ji} after performing the shifts indicated by the receiver. Having aligned all of the selected masks with the selected secrets, the receiver can now easily recover each selected secret x_{ji} where (j,i) satisfies $p_{ji}=1$.

This protocol is formally described as follows.

Protocol 4.1

Input of the sender $\boldsymbol{S}$: m $n-$tuples $(x_{11},\cdots,x_{1n}),\cdots,(x_{m1},\cdots,x_{mn})$ of $l-$bit strings.

Input of the receiver $\boldsymbol{R}$: $m\times n$ selection matrix P satisfying that there is one and only one entry equal to 1 in each row, where $p_{ji}=1$ means $\boldsymbol{R}$ receives ith string out of jth $n-$tuples $(x_{j1},\cdots,x_{jn})$.

Common Input: security parameters k, σ

Oracle: a random oracle $H:[\sigma]\times[m]\times[n]\times\{0,1\}^k\rightarrow\{0,1\}^l$.

Cryptographic Primitive: an ideal 1 out of n $\mathrm{OT}_m^{\sigma k}$ primitive.

(1) For each p, $1\le p\le\sigma$, $\boldsymbol{S}$ prepares randomly a $k\times n$ matrix $U^{(p)}$ satisfying that there is one and only one entry equal to 1 in each row, and m $n-$tuples $(x_{11}^{(p)},\cdots,x_{1n}^{(p)}),\cdots,(x_{m1}^{(p)},\cdots,x_{mn}^{(p)})$ of $l-$bit strings.
For each p, $1\le p\le\sigma$, $\boldsymbol{R}$ prepares a random $m\times n$ matrix $V^{(p)}$ satisfying that there is one and only one entry equal to 1 in each row, and a random $m\times k$ bit matrix $T^{(p)}$.
(2) $\boldsymbol{S}$ and $\boldsymbol{R}$ invoke $\mathrm{OT}_m^{\sigma k}$ with $\boldsymbol{S}$ as a receiver and $\boldsymbol{R}$ as a sender, where the inputs of $\boldsymbol{S}$ are $U^{(p)}$, $1\le p\le\sigma$, and the inputs of $\boldsymbol{R}$ are $(\mathbf{v}^{(p),1}\oplus\mathbf{t}^{(p),i},\cdots,\mathbf{v}^{(p),n}\oplus\mathbf{t}^{(p),i})$, $1\le p\le\sigma$, $1\le i\le k$. Let $Q^{(p)}$ denote the pth $m\times k$ matrix of the values received by $\boldsymbol{S}$.
(3) $\boldsymbol{S}$ picks a random subset $S\subset[\sigma]$ of size $\sigma/2$, and challenges $\boldsymbol{R}$ to reveal all matrices $V^{(p)}$ and $T^{(p)}$ with $p\in S$. If the reply of $\boldsymbol{R}$ is not fully consistent with the values received in Step 2, $\boldsymbol{S}$ aborts.
(4) For each $p\notin S$ and $1\le j\le m$, $\boldsymbol{S}$ sends

$$y_{ji}^{(p)}=x_{ji}^{(p)}\oplus H(p,j,i,\mathbf{q}_j^{(p)}\oplus\mathbf{u}^{(p),iT}).$$

(5) For each $p\notin S$ and $1\le j\le m$, $\boldsymbol{R}$ sends a correction number $c_j^{(p)}$ with $c_j^{(p)}=i'-i\pmod n$, in which i and i' satisfy $\mathbf{p}_{ji}=1,\mathbf{v}_{ji'}^{(p)}=1$.
(6) For $1\le j\le m$ and $1\le i\le n$, $\boldsymbol{S}$ sends

$$w_{ji}=x_{ji}\oplus\bigoplus_{p\notin S}x^{(p)}_{j,i\uplus c_j^{(p)}},$$

where the operation '$\uplus$' is the same as the addition module n except that replacing the result by n when it is zero.

(7) For $1 \leq j \leq m$, $\boldsymbol{R}$ outputs

$$z_j = w_{ji} \oplus \bigoplus_{p \notin S, v_{ji'}^{(p)}=1} x_{ji'}^{(p)}$$

whenever $p_{ji} = 1$.

Note that the above protocol does not give a malicious $\boldsymbol{S}^*$ any advantage in guessing the inputs of $\boldsymbol{R}$. Moreover, except with $2^{-\Omega(\sigma)}$ failure probability, security against a malicious $\boldsymbol{R}^*$ reduces to security against a well-behaved ***R'***.

Efficiency. The modification described above increases the communication and time complexity of the original protocol by a factor of σ. The probability of $\boldsymbol{R}^*$ getting away with cheating is $2^{-\Omega(\sigma)}$[7]. In terms of round complexity, the protocol described above adds a constant number of rounds to the original protocol.

5 Conclusion

We first present a protocol which reduces 1-out-of-n oblivious transfer OT_l^m to 1-out-of-n oblivious transfer OT_m^k for $n > 2$ in random oracle model, and show that the protocol is secure against malicious sender and semi-honest receiver. Then, by employing a cut-and-choose technique, we obtain a variant of the basic protocol which is secure against a malicious receiver.

Similar to the comments in [8], we may partition the task of extending oblivious transfers into several parts. The first part is what has been done in this paper and [8]: reducing OT_l^m to OT_m^k, where k is a security parameter and $m > k$. The second part is reducing OT_m^k to OT_k^k, which can be easily done in random oracle model by generating $n \cdot m$ pseudo-random bits as masks of the strings to be sent, similar to that pointed out in [8]. The third part is to reduce OT_k^k to OT_k^1 by invoking k OT_k^1 in parallel, while OT_k^1 can be realized by direct implementations or reduced to OT_1^1[13]. Thus, we may take OT_1^1 or OT_k^k to be the cryptographic primitive and reach any type of OT protocols. Note that in all the above, we always mean 1-out-of-n OT whereas the parameter n does not occur apparently in the notation.

In general, the results obtained in this paper can be summarized as follows.

Theorem 5.1 *Let k denote a computational security parameter. Let $n > 2$. For any constant $c > 1$ it is possible to reduce k^c 1-out-of-n oblivious transfers* *to k 1-out-of-n oblivious transfers in random oracle model under the condition $n = o(k)$.*

Combining this conclusion with that in [8], we can have

Theorem 5.2 *Let k denote a computational security parameter. For any constant $c > 1$ and any $n \geq 1$ it is possible to reduce k^c 1-out-of-n oblivious transfers to k 1-out-of-n oblivious transfers in random oracle model under the condition $n = o(k)$.*

References

[1] R. Impagliazzo and S. Rudich, Limits on the provable consequences of one-way permutations, Proceedings of 21st Annual ACM Symposium on the Theory of Computing, 1989, pp. 44 - 61.

[2] D. Beaver, Correlated pseudorandomness and the complexity of private computations, STOC 1996: 479 - 488.

[3] M. Naor, B. Pinkas, Efficient oblivious transfer protocols, In Proceedings of SODA 01, 2001.

[4] V. Niemi, A. Renvall, Cryptographic protocols and voting, In Result and Trends in Theoretical Computer Science, Lecture Notes in Computer Science 812, pp. 307 - 316, 1994.

[5] A. Salomaa, L. Santean, Secret selling of secrets with several buyers, In the 42nd EATCS Bulletin, pp. 178 - 186, 1990.

[6] J. P. Stern, A new and efficient all-or-nothing disclosure of secrets protocol, In Proceedings of Advances in Cryptology - Asiacrypt 98, Lecture Notes in Computer Science 1514, pp.357 - 371, Springer-Verlag, 1998.

[7] W. -G. Tzeng, Efficient oblivious transfer schemes, In Proceedings of 2002 International Workshop on Practice and Theory in Public-Key Cryptography (PKC 02), Lecture Notes in Computer Science 2274, Springer-Verlag, 2002.

[8] Y. Ishai, J. Kilian, K. Nissim and E. Petrank, Extending oblivious transfers efficiently, Crypto'03, 2003. pp. 145 - 161.

[9] R. Canetti, Security and composition of multiparty cryptographic protocols, J. of Cryptology, 13(1), 2000.

[10] O. Goldreich, Secure multi-party computation, Available at http://philby.ucsb.edu/cryptolib/BOOKS, February 1999.

[11] Y. Gertner, S. Kannan, T. Malkin, O. Reingold and M. Viswanathan, The Relationship between Public Key Encryption and Oblivious Transfer, Proc. of

the 41st Annual Symposium on Foundations of Computer Science (FOCS 00), 2000.

[12] Y. Gertner, T. Malkin and O. Reingold, On the Impossibility of Basing Trapdoor Functions on Trapdoor Predicates Proc. of the 42st Annual Symposium on Foundations of Computer Science (FOCS 01) 2001.

[13] G. Brassard, C. Crépeau, and J. M. Robert, All-or-nothing disclosure of secrets, Crypto'86, pp. 234 C 238, 1987.