General Two-Party Oblivious Circuit Evaluation

Sufyan T. Faraj
College of IT, Nahrain University
Al-Jaderiya, Baghdad, Iraq
Email: sufyantaih@yahoo.com

***Abstract*—A Two-Party Oblivious Circuit Evaluation (2P-OCE) Protocol is a way for a party Alice owns a secret x and another party Bob owns a secret y to compute the value of an agreed upon function f(x,y), where f can be computed by a polynomial sized Boolean circuit. This is done in such a way that Alice learns nothing about y and Bob learns nothing about x, except for what can be inferred from one's private input and the public value of f(x,y). This paper presents a general, correct, fair, honest, and maximum privacy 2P-OCE protocol that is based on a set of reductions to more simple cryptographic primitives. The protocol uses the primitives of one-out-of-two Oblivious Transfer (1-2-OT) and Bit Commitment (BC) as black boxes. Consequently, the protocol may be implemented with or without computational assumptions, depending on the type of 1-2-OT and BC used by participants.**

Index Terms— **Bit Commitment, Circuit Evaluation, Multi-Party Computation, Oblivious Transfer.**

I. Introduction

THE main purpose of many cryptographic protocols is to allow parties to collaborate towards some common goal, while maintaining the maximum possible privacy of their secrets. Typically, the common goal is to compute some function of the local inputs (secrets) held by the different parties. This general cryptographic task is usually referred to as a Secure Multi-Party Computation (SMPC). Secure two-party computation is a special case of SMPC where the number of participants equals two.

An analogy can be made between the study of oblivious circuit evaluation and ordinary circuit evaluation. One of the very trivial but extremely important properties of Boolean circuits is that they can be broken up into extremely simple computational entities, such as NAND gates. A non-trivial property of secure two-party protocols is that they can be broken up into simple secure protocols [1].

The first protocol for the problem of Two-Party Oblivious Circuit Evaluation (2P-OCE) -also referred to as secure two-party computation- was presented by Yao in [2], where he obtained a general result for generating maximum privacy two-party protocols for any two argument function. Since then, many works had appeared aiming towards finding better constructions for solving the SMPC problem. One important related work is that by Goldreich and Vainish [3] who discovered a reduction of 2P-OCE to one-out-of-two Oblivious Transfer (1-2-OT) for the case where *both parties are honest*.

It can be noted that, in general, all of the SMPC problems can be solved in theory by first representing the computational problem as a combinatorial circuit, and the using a technique for circuit evaluation. However, using this general solution for special cases of multi-party computation can be impractical. Hence, dedicated solutions can be developed for special cases for efficiency reasons. Accordingly, some researchers have proposed special solutions for each specific SMPC problem such as private information retrieval, privacy-preserving statistical analysis, scientific computation, and several other problems [4] and [5].

However, it seems that even those special solutions are still not efficient enough for practical uses. This remark had stimulated a new research paradigm aiming at developing practical solutions to SMPC problems based on an "acceptable" security model instead of the ideal security model (focusing on zero information disclosure) that is expensive to achieve [6].

In this paper, we extend the work of [3] to the more general case of *dishonest* parties. This is done by using techniques from [7] and [8] to build a Committed Oblivious Transfer (COT) protocol (will be defined later). This protocol is constructed using the Bit Commitment (BC) and 1-2-OT primitives as black boxes. The COT protocol is then used as a building block for the construction of the 2P-OCE protocol. Our protocol is relatively simple, general, and neither biased toward a dedicated application nor toward a special implementation. It is also based on the harder requirement of the ideal security model.

II. Preliminaries

In this section, some important cryptographic primitives, which would be used to build the 2P-OCE protocol, are described briefly. In 1-2-OT Bob has to choose between learning bit a_0 or a_1 prepared by Alice, who does not learn his choice b. Bob learns a_b and obtain no information about $a_{\bar{b}}$ [7].

The All-or-Nothing Disclosure Of Secrets (ANDOS) protocol is a way for Alice to reveal an element from a set of

Manuscript received May 21, 2006.

m strings of length *t* to Bob, in a way that Alice does not learn which string Bob gets. Bob should not get information about more than one Alice's strings. The 1-2-OT protocol is a special case of this general protocol where *m* = 2 and *t* = 1. It was shown in [9] how to achieve ANDOS from 1-2-OT. Another approach is to Build ANDOS schemes from basic techniques directly [10], [11]. This could achieve more efficient oblivious transfer protocols in terms of computations and/or communication overhead.

While in a Bit Commitment (BC) protocol, Alice has a bit *a* in mind, to which she would like to be committed toward Bob. That is, Alice wishes to provide Bob with a piece of evidence that she has a bit *a* in mind and that she cannot change it. Meanwhile, Bob should not be able to tell from that evidence what *a* is. At a later time, Alice can open her commitment by revealing the value of *a* and prove to Bob that the piece of evidence sent earlier really corresponded to that bit [12].

It is possible to prove that some BCs satisfy an XOR-relation without giving away their values. For this purpose, a protocol known as Bit Commitment with XOR (BCX) is used [7]. If Alice is committed to several BCXs to bits b^1, b^2, …, b^k she can prove to Bob that: $\bigoplus_{j=1}^{k} BCX(b^j) = c$, for some public value *c* without disclosing the b^j's [13], [14]. Note that this technique can be used to prove that two BCXs to *a* and *b* are equal or different simply by proving $BCX(a) \oplus BCX(b) = 0$ or $BCX(a) \oplus BCX(b) = 1$, respectively. Since each such proof destroys the BCX involved, they must be copied first [7].

Because of the complexity of these constructions, for the remaining of this paper we consider as a *unitary* BCX operation: creation of a BCX, opening of a BCX, or proof that a constant number of BCXs satisfy a given linear relation.

III. Committed Oblivious Transfer

In the ANDOS protocol, although Alice cannot find out which string Bob is getting, she may nevertheless use some "garbage" secrets that are of no use to him. Because of this possibility, if the outcome of the protocol must be used by Bob in some further interaction, Alice may, by offering "good" and "bad" secrets, determine from which set he chooses, depending on his ability to continue the protocol or not. Thus, it might be necessary for Bob to verify some properties of the secrets before getting one of them. This argument is hold to other versions of OT. For this reason, a Committed Oblivious Transfer (COT) protocol is required.

The COT protocol is the natural fusion of 1-2-OT and BC protocols. At the start of the COT protocol, Alice is committed to bits a_0 and a_1 using $BCX(a_0)$ and $BCX(a_1)$. Also, Bob is committed to bit *b* using $BCX(b)$. After running the COT protocol, $COT(BCX(a_0), BCX(a_1))\,(BCX(b))$, Bob will be committed to $BCX(a) = a_b$. Alice, whatever she does cannot use the protocol to learn information on *b*. Bob, whatever he does, cannot use the protocol to learn information on $a_{\bar{b}}$ [7]. Committed ANDOS (CANDOS) can be achieved from COT exactly like ANDOS can be obtained from 1-2-OT [8].

First an elementary version of COT that we call E-COT is constructed. The E-COT protocol enables Bob to verify the correctness of Alice's actions in the COT protocol. Let a_0 and a_1 be the two secret bits of Alice. The E-COT protocol can proceed as described in [7] and [8]. Then, Bob should be committed to his choice *b* and what he reads a_b. Hence, we use the E-COT as a primitive to build the COT protocol [7], [8].

IV. The Proposed Method

In this section, the proposed method for achieving the general cryptographic task of oblivious circuit evaluation is described. The computational model required to design the oblivious evaluation protocol is explained at first. Then, the Two-Party Oblivious Circuit Evaluation (2P-OCE) protocol is presented. We assume the general case of dishonest parties involved in the cryptographic protocol.

The Boolean circuit to be evaluated is assumed to be consisting of a number *g* of gates. Thus, the main step is to manage to compute the output of a Boolean gate without revealing the input values. The computational model for evaluating the output of a Boolean gate is shown in Fig. 1. For each bit *c* in the computation, Alice owns *x* and Bob owns *y* such that $c = x \oplus y$. It is also assumed that each gate has two inputs, represented by bits c_0 and c_1. The value of c_0 and c_1 should not be revealed in the protocol. Each gate has an output represented by bit c_2.

Consider the evaluation of the output of the NAND gate on two bits c_0 and c_1, for example. Alice owns x_0, x_1 and Bob owns y_0, y_1 such that they wish to come up with secret values x_2 and y_2 such that $x_2 \oplus y_2 = c_2 = c_0\ NAND\ c_1 = \overline{c_0 \cdot c_1}$. Thus, we have:

$$c_2 = x_2 \oplus y_2 \tag{1}$$

and

$$c_2 = \overline{c_0 \cdot c_1} \tag{2}$$

Substituting for the values of c_0 and c_1 in equation (2) and solving, then substituting for the value of c_2 from equation (1) we obtain:

$$y_2 = x_0 \cdot y_0 + \bar{x}_0 \cdot \bar{y}_0 + x_1 \cdot y_1 + \bar{x}_1 \cdot \bar{y}_1 \oplus x_2 \tag{3}$$

It is well known that any Boolean function can be computed using either NAND or NOR gates only. In fact, it is straightforward to convert Boolean circuits to pure NAND or pure NOR form. Hence, this conversion procedure can be done before using the 2P-OCE protocol to evaluate the circuit. However, although this procedure shows that a solution

always feasible, it does not lead to the optimal circuits in the sense of the minimum number of gates. Hence, we use the computational model of Fig. 1 for the other types of Boolean gates.

Following a similar analysis to that used for the case of the NAND gate above, we obtain for the case of the AND, OR, NOR, and XOR gates. Indeed, our computational model can be easily relaxed to the case of the NOT gate with a single input c_0 and an output c_2. It can be noted that this procedure may be easily extended for the case of any type of a Boolean gate, with different number of inputs.

Now, we propose a protocol for the evaluation of a single Boolean gate, which we call the Two-Party Oblivious Gate Evaluation (2P-OGE) protocol. Then, this protocol is used as a basic building block for implementing the 2P-OCE protocol. The 2P-OGE protocol is as follows (For clearness and simplicity, the protocol is described for the case of the NAND gate):

1. Alice commits to x_0, x_1 using BCXs.
2. Bob commits to y_0, y_1 using BCXs.
3. Alice randomly chooses x_2 and commits to it using BCX.
4. Alice prepares the four following secrets (corresponding to the cases (y_0, y_1) = (0, 0); (0, 1);(1, 0); and (1, 1), respectively):
   $i-\quad \overline{(x_0 \cdot x_1)} \oplus x_2$
   $ii-\quad \overline{(x_0 \cdot \bar{x}_1)} \oplus x_2$
   $iii-\quad \overline{(\bar{x}_0 \cdot x_1)} \oplus x_2$
   $iv-\quad \overline{(\bar{x}_0 \cdot \bar{x}_1)} \oplus x_2$
5. Bob uses the XOR property of BCX to prove to Alice whether $y_0 = y_1$ or not.
6. If $y_0 = y_1$, then Alice performs a COT protocol with Bob using the *first* and the *fourth* of the above secrets as bits a_0 and a_1, respectively. Otherwise, if $y_0 \neq y_1$, Alice uses the *second* and *third* secrets instead, respectively. In both cases, Bob uses $b = y_0$ in the COT protocol. It is obvious that the outcome of this COT protocol is y_2.
7. Bob commits to y_2 using BCX.

For evaluating the AND, OR, NOR, and XOR gates, the above 2P-OGE protocol is performed with different secrets in step (4). The required values for these secrets for each case are shown in Table 1. These secret values are obtained from the respective equations for calculating y_2. The 2P-OGE protocol can also be easily modified to evaluate the NOT gate, with an input c_0 and an output c_2, by ignoring any operation that includes x_1 and y_1. Thus, step (5) is not needed and only two secrets are required to be prepared by Alice in step (4), which correspond to the cases $y_0 = 0$ or $y_0 = 1$, respectively. Therefore, in the NOT gate case, there is only one possibility for performing the COT protocol in step (6), where Alice offers two secrets.

Now, the above 2P-OGE protocol is used for building the required 2P-OCE protocol, which consists of the following three steps:

1. *Initialization*: In this step, Alice and Bob agree on a Boolean circuit *F* with *g* gates to be evaluated and on a security parameter *n*. They also commits to their respective input bits using BCXs.
2. *Computation*: In the computation step, Alice and Bob evaluate the circuit *F*, one gate after the other, using the 2P-OGE protocol described previously. They use the BCX protocol to commit to all input and output bits. Sine the output bit of a gate can be an input bit to several other gates, committed bit can be copied as needed using the technique for copying BCXs describe earlier.
3. *Revelation*: At the end of the computation step, each output bit of the circuit is hidden in a BCX. In order that Alice and Bob learn an output bit of *F*, each one of them should open his (her) related BCX. The fairness of this step can be further enhanced by using a technique for the *fair disclosure of secrets*.

V. Conclusion

It is worth to compare our protocol with the fair secure two-party computation protocol due to B. Pinkas who demonstrated a transformation to the original Yao's protocol by adding more steps based on gradual release time commitments and blind signatures [15]. However, we believe that our protocol is more robust in the sense that it is based on more fundamental primitives and has no cryptographic assumptions at the high level of its design. Another advantage of our protocol is that it is not biased toward any specific implementation. Indeed, our two-party protocol can be easily modified to produce the generator of the multi-party protocol of. Hence, the extension of this OCE protocol to the more general case of multi-party protocols, where the number of engaged parties is more than two, is feasible.

References

[1] C. Crépeau and J. Kilian, "Achieving oblivious transfer using weakened security assumptions", Proceedings of IEEE 29th FOCS, 1988, pp. 42 - 52.
[2] A. Yao, "How to generate and exchange secrets", Proceedings of IEEE 27th FOCS, 1986.
[3] O. Goldreich and R. Vainish, "How to solve any protocol problem - An efficiency improvement", Proceedings of CRYPTO'87, Lecture Notes in Computer Science, Springer-Verlag, 1988, pp. 73-86.
[4] W. Du and M. Atallah, "Secure multi-party computation problems and their applications: A review and open problems", New Security Paradigms Workshop, New Mexico, USA, 2001, pp. 11-20.
[5] W. Du, "A study of several specifc secure two-party computation problems", Ph.D. Thesis, Purdue University, Indiana, 2001.
[6] W. Du and Z. Zhan, "A practical approach to solve secure multiparty computation problems", Department of Electrical Engineering and Computer Science, Syracuse University, 2002.

[7] C. Crépeau, J. Graaf, and A. Tapp, "Committed oblivious transfer and private multi-party computations", Proceedings of CRYPTO'95, Lecture Notes in Computer Science, Springer-Verlag, 1995, pp. 110-123.
[8] C. Crépeau, "Verifiable disclosure of secrets and applications", In Advances in Cryptology: Proceedings of EUROCRYPT'89, Lecture Notes in Computer Science, Springer-Verlag, Vol. 434, 1990, pp. 181-191.
[9] G. Brassard, C. Crépeau, and J-M Robert, "Information theoretic reductions among disclosure problems", Proceedings of IEEE 27th FOCS, 1986, pp. 168-173.
[10] W-G Tzeng, "Efficient oblivious transfer schemes", Proceedings of PKC'02, Lecture Notes in Computer Science, Springer-Verlag, Vol. 2274, 2002, pp.159-171.
[11] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols", Proceedings of SODA'01, 2001, pp.448-457.
[12] G. Brassard, C. Crépeau, R. Jozsa, and D. Langlois, "A quantum bit commitment scheme provably unbreakable by both parties", Proceedings of IEEE 34th FOCS, 1993, pp. 362 – 371.
[13] J. Kilian, "A note on efficient zero-knowledge proofs and arguments", Proceedings of ACM 24th STOC, 1992, pp. 723-732.
[14] J. Kilian, "On the complexity of bounded-interaction and noninteractive zero-knowledge proofs", Proceedings IEEE 35th FOCS, 1994, pp. 466-477.
[15] B. Pinkas, "Fair secure two-party computation", Proceedings of EUROCRYPT'03, Lecture Notes in Computer Science, Springer-Verlag, Vol. 2656, 2002, pp. 87-105.

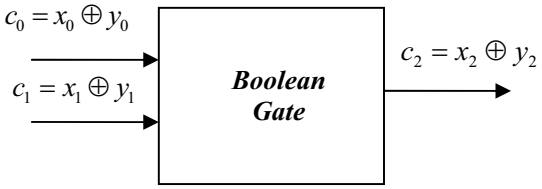

Fig. 1. The Computational model for a Boolean gate.

TABLE 1
ALICE'S SECRET VALUES REQUIRED FOR DIFFERNT TYPES OF GATES

y_0	y_1	*NAND*	*AND*	*OR*	*NOR*	*XOR*
0	0	$\overline{(x_0 \cdot x_1)} \oplus x_2$	$(x_0 \cdot x_1) \oplus x_2$	$(x_0 + x_1) \oplus x_2$	$\overline{(x_0 + x_1)} \oplus x_2$	$(x_0 \oplus x_1) \oplus x_2$
0	1	$\overline{(x_0 \cdot \bar{x}_1)} \oplus x_2$	$(x_0 \cdot \bar{x}_1) \oplus x_2$	$(x_0 + \bar{x}_1) \oplus x_2$	$\overline{(x_0 + \bar{x}_1)} \oplus x_2$	$(x_0 \oplus \bar{x}_1) \oplus x_2$
1	0	$\overline{(\bar{x}_0 \cdot x_1)} \oplus x_2$	$(\bar{x}_0 \cdot x_1) \oplus x_2$	$(\bar{x}_0 + x_1) \oplus x_2$	$\overline{(\bar{x}_0 + x_1)} \oplus x_2$	$(\bar{x}_0 \oplus x_1) \oplus x_2$
1	1	$\overline{(\bar{x}_0 \cdot \bar{x}_1)} \oplus x_2$	$(\bar{x}_0 \cdot \bar{x}_1) \oplus x_2$	$(\bar{x}_0 + \bar{x}_1) \oplus x_2$	$\overline{(\bar{x}_0 + \bar{x}_1)} \oplus x_2$	$(\bar{x}_0 \oplus \bar{x}_1) \oplus x_2$

Top