Efficient Information Theoretic Multi-party Computation from Oblivious Linear Evaluation

Louis Cianciullo, Hossein Ghodosi

Efficient Information Theoretic Multi-Party Computation from Oblivious Linear Evaluation

Louis Cianciullo* (✉) and Hossein Ghodosi

James Cook University, Townsville 4811, Australia
louis.cianciullo@jcu.edu.au, hossein.ghodosi@jcu.edu.au

Abstract. Oblivious linear evaluation (OLE) is a two party protocol that allows a receiver to compute an evaluation of a sender's private, degree 1 polynomial, without letting the sender learn the evaluation point. OLE is a special case of oblivious polynomial evaluation (OPE) which was first introduced by Naor and Pinkas in 1999. In this article we utilise OLE for the purpose of computing multiplication in multi-party computation (MPC).
MPC allows a set of n mutually distrustful parties to privately compute any given function across their private inputs, even if up to $t < n$ of these participants are corrupted and controlled by an external adversary. In terms of efficiency and communication complexity, multiplication in MPC has always been a large bottleneck. The typical method employed by most current protocols has been to utilise Beaver's method, which relies on some precomputed information. In this paper we introduce an OLE-based MPC protocol which also relies on some precomputed information.
Our proposed protocol has a more efficient communication complexity than Beaver's protocol by a multiplicative factor of t. Furthermore, to compute a share to a multiplication, a participant in our protocol need only communicate with one other participant; unlike Beaver's protocol which requires a participant to contact at least t other participants.

1 Introduction

Oblivious polynomial evaluation (OPE) was first introduced by Naor and Pinkas [10] in 1999. An OPE protocol consists of two participants, a sender, S who holds a polynomial $f(x)$ and a receiver, R who has a value α. OPE allows R to learn $f(\alpha)$ without having S learn α and also keeping $f(x)$ private. A more formal definition, originally given in [5] is presented below:

Definition 1. *[5] An OPE protocol is composed of two parties, S who has a polynomial $f(x)$ over a finite field $\mathbb{F}$ and R who has an input value $\alpha \in \mathbb{F}$. Correctness is achieved if, at the end of the protocol, R learns $f(\alpha)$. Security is guaranteed if the following two conditions are met after the protocol has been executed:*

* This research is supported by an Australian Government Research Training Program (RTP) Scholarship.

1. *S cannot reduce his uncertainty of α.*
2. *R does not learn any information relating to $f(x)$, other than $f(\alpha)$.*

In this article we focus on a special case of OPE, wherein $f(x)$ is of degree at most one, known as oblivious linear evaluation (OLE). Specifically, we utilise OLE for the purpose of performing multiplication in multi-party computation (MPC).

MPC allows a set of n mutually distrustful participants to compute any given function across their private inputs, without revealing any information relating to their private inputs. We focus on the threshold setting, where an MPC protocol is considered secure if a set of t or less participants, where $t < n$, cannot gain any information relating to another participant's private input, other than what the output of the protocol gives them. More formally:

Definition 2. *A (t, n) threshold MPC protocol allows a set of n participants, $P_1, \cdots, P_n$ with respective private inputs, $x_1, \cdots, x_n$ to compute a given function, $f(x_1, \cdots, x_n)$.*

Privacy is maintained if, after completion of the protocol, an adversary controlling any subset of up to t participants ($t < n$), cannot learn more information (about other participant's private inputs) than what could be derived from each participant's individual, private input and the output of the protocol.

Traditionally, the adversary is classified as either passive or malicious. Participants under control of a passive adversary may share information with one another but do not deviate from the MPC protocol. Participants under control of a malicious adversary also share information but may act arbitrarily, i.e. they do not necessarily follow the protocol. Another aspect of the adversary considered in an MPC protocol is the resources it has at its command. Specifically, an unconditionally (information theoretic) secure MPC protocol is secure against a computationally unbounded adversary. Whilst a conditionally (computationally) secure MPC protocol is secure against a computationally bounded adversary.

In this article we focus on information theoretic (t, n) threshold MPC secure against a passive adversary. We show the construction of an efficient MPC scheme based on OLE. In the next section we give some background and motivation on this topic, following this we then discuss our contribution in depth.

1.1 Background

MPC is an extremely powerful tool that can be used to solve practically any given problem involving a set of distrustful parties. In classical, unconditionally secure protocols [2,6,11] each participant, P_i $(i = 1, \cdots n)$ shares their private input, x_i by utilising Shamir's secret sharing scheme [12] to distribute shares to all participants. To compute a given function, $f(x_1, \cdots x_n)$, participants need simply perform all computations on the shares of each input value. For instance, if a participant wants to compute a share relating to the sum of two distributed input values he simply adds his two corresponding shares together. At the end

of the protocol, a set of $t+1$ or more participants then pool their information to reconstruct the output.

Due to the homomorphic nature of Shamir's scheme [3] participants can easily compute any linear operation by privately computing on their shares. However, since the inception of MPC [8] the largest limiting factor has been the high amount of resources required to compute a multiplication. Perhaps the most widely known and efficient method of computing a multiplication in an MPC protocol is known as Beaver's method (A.K.A Beaver's triples) [1]. For completeness we review this protocol below.

Beaver's Method Beaver's method [1] for computing a multiplication in MPC relies on some pre-shared information known as a triple. Specifically, a triple is composed of three values, a, b and c where $a \cdot b = c$ and $a, b, c \in \mathbb{F}_q$ such that $q > n$ and q is a prime number. Each participant has a share of these triples, such that participant P_k $(k = 1, \cdots, n)$ receives the shares a_k, b_k and c_k relating to (respectively) a, b and c.

Suppose we have participants P_i with input x_i and P_j with input x_j for $i, j = 1, \cdots, n$ and $i \neq j$. To compute shares of the multiplication $\gamma = x_i \cdot x_j$ we first have both P_i and P_j distribute shares of their private values among the other participants, where P_k gets x_{i_k} relating to x_i and x_{j_k} relating to x_j. To compute a share, γ_k relating to the product γ, a set of at least $t+1$ participants execute the following steps:

1. Each participant, P_k, computes $z_k = x_{i_k} - a_k$ and $v_k = x_{j_k} - b_k$, where z_k is a share of the value $z = x_i - a$ and v_k is a share of $v = x_j - b$.
2. A set of at least $t+1$ participants broadcast their shares, z_k and v_k amongst themselves.
3. Participants publicly reconstruct the values of z and v using the shares z_k and v_k, respectively.
4. P_k computes his share of γ as $\gamma_k = zv + zb_k + va_k + c_k$.
5. $t+1$ or more participants can reconstruct $\gamma = x_i \cdot x_j$ by pooling their shares.

In order to construct z and v a set of $t+1$ participants is required to cooperate. If all participants in this set wish to compute these values (and consequently, the multiplication) then each participant must both receive and send t messages. Since each message would consist of 2 elements from the field $\mathbb{F}_q$ (i.e. z_k and v_k) the communication complexity of this protocol can be given as $\mathcal{O}(qt^2)$.

Many recent MPC protocols utilise a resource intensive computationally secure offline phase to compute these multiplication triples. The actual MPC is then carried out in a faster information theoretic online phase. For our purposes, we focus solely on the information theoretic online phase. It suffices to assume that participants gain the shares of the triples via an external party known as an initialiser, who (after computing and distributing the shares of the triples) does not take part in the actual MPC protocol. In the next section we review the OLE based two-party protocol given by Döttling et al. [7].

TinyOLE Recently Döttling et al. [7] proposed a two-party protocol ($n = 2$) in which the two participants, P_1 and P_2 use OLE to compute shares to a multiplication. Specifically, they use OLE to compute multiplication triples in an offline phase. Their scheme utilises a simple additive secret sharing scheme wherein a given value, a (for example), is represented as $a = a_1 + a_2$, across a finite field $\mathbb{F}$; where P_2 has the share a_2 and P_1 gets a_1. Addition in their scheme consists of simply adding shares together. Multiplication is achieved by utilising OLE in a black-box fashion.

To compute a multiplication of two distributed (and not necessarily known) values, a and b, they rely on the fact that: $ab = a_1b_1 + a_1b_2 + a_2b_1 + a_2b_2$. To compute the "troublesome" terms of the form $c = a_1b_2$ they utilise a black-box OLE. Essentially, P_1 acts as a sender and submits the polynomial $f(x) = a_1x - c_1$ where c_1 is a randomly chosen value. The second participant, P_2 acts as receiver and submits $\alpha = b_2$. Both participants send their values to a black-box OLE, with P_2 receiving back $f(\alpha) = a_1b_2 - c_1$. If we set $c_2 = f(\alpha)$ then each participant now holds a share of c as $c = c_1 + c_2$. To compute shares to the entire multiplication it is easy to see that at least 2 OLEs are needed.

Döttling et al. specifically use this method in a computationally secure offline phase to compute random multiplication triples, where the values of a and b are not actually known to either participant. Our proposed scheme differs to theirs in that we wish to utilise OLE in an information theoretic, online phase to compute the multiplication of known input values for a given MPC function.

1.2 Our Contribution

In this section we summarise our proposed MPC scheme which utilises OLE to compute shares to a given multiplication. In contrast to the methods discussed above our protocol obtains the following desirable properties:

1. Unlike Beaver's scheme [1] our proposed protocol only requires communication between two participants to compute a given share to a multiplication i.e. a participant may compute his share without the assistance of t other participants. We achieve this result by having one designated participant who acts as a sender in an OLE. The other participants need simply privately compute an OLE with this sender participant to compute a share to a multiplication. As a result of this, the communication complexity of our protocol is $\mathcal{O}(qt)$, which is more efficient than Beaver's (at $\mathcal{O}(qt^2)$) by a multiplicative factor of t.
2. We do not rely on a black-box method of OLE and instead provide a specific construction. Our OLE multiplication scheme is based on the information theoretic protocol given in [9,13]. This scheme, like Beaver's scheme, relies on some precomputed information which can be produced via an offline phase or an initialiser. Since we wish to focus solely on the information theoretic OLE-based MPC scheme itself we will assume that the information is provided via an initialiser.

3. Our scheme only utilises one OLE per participant to compute a multiplication. In a two party protocol we would only need one OLE. So, for an individual participant to compute his share (in either a multi-party or two-party protocol) the complexity cost is just $\mathcal{O}(q)$. Using Beaver's scheme, this would be $\mathcal{O}(qt)$.
4. Lastly, unlike the TinyOLE scheme [7], our scheme is scalable, in that it extends to the multi-party case with n participants. In fact, computing n shares (one for each participant) to a single multiplication requires only $n-1$ OLEs, one for each individual participant to compute his share. We note that utilising all n participants is not actually necessary. We only really require a set of $t+1$ participants, enough to compute the output of a given multiplication at the end of the protocol.

1.3 Outline

The rest of the paper is organised as follows. In section 2 we go over some of the sub protocols and tools used in our proposed MPC protocol. Section 3 gives a high level overview of our protocol as well as a model for security. The actual construction for our protocol is given in section 4, along with an evaluation and proof of correctness and security.

2 Preliminaries

In this section we review Shamir's secret sharing scheme [12] and the the information theoretic OPE originally given by Hanaoka et al. in [9]. Both of these protocols are fundamental building blocks of our proposed MPC protocol.

2.1 Shamir's Secret Sharing Scheme

Like all MPC schemes our proposed protocol utilises secret sharing to ensure privacy. In a secret sharing scheme a set of n participants are each privately sent a share of a given secret. An authorised subset of these participants can pool their shares to recover the secret, whilst an unauthorised subset should get no information. We note that our proposed OLE-based MPC scheme can potentially work with any linear secret sharing scheme. However, in order to show a specific implementation we will demonstrate our proposed protocol using Shamir's secret sharing scheme [12].

Shamir's scheme is a (t, n) threshold scheme, meaning that an authorised subset is any set of $t+1$ or more participants where $t < n$. This scheme operates over a finite field $\mathbb{F}_q$ where $q > n$ and q is a prime number. To demonstrate, suppose we have participant P_i $(i = 1, \cdots, n)$ who wishes to distribute his input value, x_i among the other participants. P_i computes a random polynomial, $g(x)$ of degree at most t and sets $g(0) = x_i$. He then privately sends to each participant, P_k $(k = 1, \cdots n)$ the share $x_{i_k} = g(k)$. A set of $t+1$ or more participants can reconstruct x_i by performing Lagrange interpolation across their shares to compute $g(x)$.

2.2 Information Theoretic OPE

Hanoka et al. [9] introduced an unconditionally secure OPE protocol that utilises some pre-distributed information to achieve information theoretic security. Our proposed OLE-based MPC protocol utilises a modified variant of their OPE protocol to compute a multiplication. Therefore, for completeness, we display their full protocol in figure 1.

In the initial, setup phase of their protocol a third party, known as an initialiser assigns some random information to both $\mathcal{R}$ and $\mathcal{S}$. Following this the initialiser takes no further part in the protocol and $\mathcal{R}$ and $\mathcal{S}$ utilise the assigned information to compute an OPE in the computation phase of the protocol.

Input: $\mathcal{R}$ has a value α and $\mathcal{S}$ the polynomial $f(x)$ of degree at most t.
Output: $\mathcal{R}$ obtains $f(\alpha)$ and S gets nothing.

Setup The initialiser privately sends:

1. A random polynomial, $S(x)$, of degree at most t to $\mathcal{S}$.
2. A random value, d and the value $g = S(d)$ to $\mathcal{R}$.

Computation

1. $\mathcal{R}$ sends the value $l = \alpha - d$ to $\mathcal{S}$.
2. $\mathcal{S}$ then computes and sends to $\mathcal{R}$ the polynomial $V(x) = f(l + x) + S(x)$.
3. $\mathcal{R}$ computes $f(\alpha) = V(d) - g$.

Fig. 1. Information Theoretic OPE [9,13]

3 Model

This section presents a high level overview of our protocol and a set of criteria for evaluating the security of our scheme. We use the traditional setting of MPC protocols. That is, each party P_j $(1 \leq j \leq n)$ distributes its private input, x_j amongst all participants, using a Shamir (t, n) threshold scheme. Linear functions can be computed by each participant privately. In order to perform multiplication, however, we must utilise OLE.

3.1 Overview

Suppose we have a set of n participants who wish to compute shares to the value $\gamma = x_i \cdot x_j$, where x_i and x_j are the respective private input values of

participants P_i and P_j, for $1 \leq i, j \leq n$ and $i \neq j$. Further suppose, that P_j utilises the polynomial $f_j(x)$ to share x_j among all participants (via Shamir's secret sharing scheme), such that a given participant P_k $(k = 1, \cdots, n)$ receives the share $x_{j_k} = f_j(k)$ of x_j.

A simple method for computing shares of γ is to have each participant, P_k, simply send his share, x_{j_k}, to P_i who can then send back the value $\gamma_k = x_i x_{j_k}$. Due to the homomorphic nature of Shamir's secret sharing scheme the value γ_k is a share corresponding to the polynomial $\Gamma(x) = x_i f_j(x)$ with free term $x_i \cdot x_j$. The obvious problem with this simple protocol is that neither P_i's nor P_j's privacy is maintained.

To keep P_i's input, x_i, private we can have P_i introduce a random, private masking polynomial, $h_i(x)$, of degree at most t, with free term $h_i(0) = 0$. Now, when he receives a given share, x_{j_k}, we require P_i to send back $\gamma_k = x_i x_{j_k} + h_i(k)$. Each P_k now holds shares to the polynomial $\Gamma(x) = x_i f_j(x) + h_i(x)$. Intrinsically we can see that, due to Shamir's secret sharing scheme, the protocol is now t-private with respect to P_i, as a set of t participants with t shares cannot compute any information relating to the effectively random polynomial $\Gamma(x)$. It remains to ensure the privacy of P_j.

Surprisingly, ensuring that P_j's privacy is maintained is actually quite simple. Rather than having each P_k simply hand his share to P_i we instead have P_k and P_i utilise an OLE protocol, where P_k acts as the receiver and P_i as the sender. First P_i computes two polynomials, $f_i(x) = x_i \cdot x$ and $h_i(x)$ (the masking polynomial, as before). Each P_k $(1 \leq k \leq n)$ then acts as the receiver and executes an OPE protocol with P_i (who acts as the sender) to privately evaluate P_i's polynomial, $f_i(x)$ at the point x_{j_k}, as before P_i adds the masking polynomial to his computation.

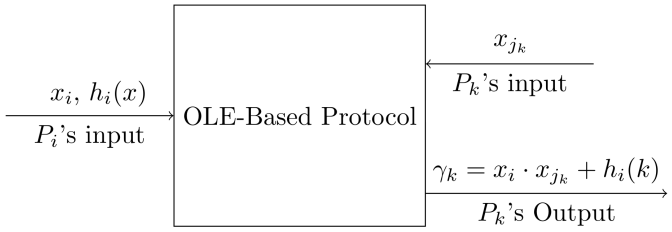

Fig. 2. Overview of the protocol

Since the OLE protocol does not allow P_i to learn the evaluation point then the protocol can now be considered t-private for both P_i and P_j. Specifically, P_i's privacy is maintained via the masking polynomial and P_j's privacy is maintained via Shamir's secret sharing scheme and the OLE protocol. An overview of this is given in figure 2. Note that P_i will also use his share from x_j and compute

his own share of $\gamma = x_i \cdot x_j$ (of course, there is no need to perform OLE, as he plays the role of the sender and receiver at the same time).

3.2 Security and Correctness

In order to prove the security and correctness of our proposed scheme we will evaluate it against the following criteria specified below:

1. **Correctness** – Upon completion of the protocol each participant, P_k holds a share, γ_k of the polynomial $\Gamma(x)$, of degree at most t with free term $\Gamma(0) = x_i \cdot x_j$.
2. **Privacy** – A set of t or less participants, not including either P_i or P_j, cannot reduce their uncertainty of x_i or x_j.
3. **Privacy with respect to** P_i – A set of t or less participants, including P_j, cannot reduce their uncertainty of x_i.
4. **Privacy with respect to** P_j – A set of t or less participants, including P_i, cannot reduce their uncertainty of x_j.

We note that the last three criterion presented here simply encapsulate the notion of privacy given in definition 2.

4 Proposed OLE-Based MPC Protocol

Similar to the OPE protocol given in figure 1, our proposed multiplication protocol consists of two phases:

1. **The Setup Phase:** Where the initialiser privately sends some (essentially random) information to each participant involved in the protocol.
2. **The Computation Phase:** Where participants are able to compute shares to the multiplication.

Where our scheme differs, however, is in the addition of a masking polynomial ($h_i(x)$) and the limiting of the receivers polynomial to a degree no greater than 1 (OLE).

As per section 3.1 suppose we have a set of n participants $P_1, \cdots, P_n$, with respective private inputs $x_1, \cdots, x_n$, who wish to compute shares of the value $\gamma = x_i \times x_j$ where $i, j \in [1, n]$ and $i \neq j$. Participant P_j first privately distributes shares for x_j amongst all participants, using the polynomial $f_j(x)$, such that P_k $(1 \leq k \leq n)$ gets the share $x_{j_k} = f_j(k)$. To compute a share γ_k, of γ each P_k cooperates with P_i to execute our modified OLE protocol, with P_k essentially acting as the receiver and P_i acting as the sender for each P_k. Note that all computations are performed in the field $\mathbb{F}_q$ where q is a prime number such that $q > n$. The full protocol is given in Figure 3.

In order to compute a share P_k and P_i exchange exactly 3 field elements (l and $V(x)$). This gives a communication complexity of $\mathcal{O}(q)$. Therefore, the overall communication complexity, required for each of the n participants to compute

his share can be given as $\mathcal{O}(qn)$. However, since the protocol is based on Shamir's (t, n) secret sharing scheme [12] we actually only require $t + 1$ participants to ensure the output can be constructed. This gives a communication complexity of $\mathcal{O}(qt)$.

Input: P_i has x_i and P_k has the share x_{j_k}, of x_j.
Output: P_k obtains the share γ_k, of $\Gamma(x)$ where $\Gamma(0) = x_i \cdot x_j$.

Setup The initialiser privately sends:

1. A set of $n - 1$ random polynomials, $S_{i_k}(x)$, of degree at most 1 to P_i where $k = 1, \cdots, n$ and $k \neq i$.
2. A random value, d_k and the value $g_k = S_{i_k}(d_k)$ to every participant P_k.

Computation
P_i privately computes:

- A masking polynomial, $h_i(x)$ of degree at most t with $f_i(0) = 0$.
- The multiplication polynomial, $f_i(x) = x_i \cdot x$.

Each P_k then privately executes the following steps with P_i:

1. P_k sends the value $l_k = x_{j_k} - d_k$ to P_i.
2. P_i then computes and sends to P_k the polynomial $V_k(x)$ which is computed as:
$$V_k(x) = h_i(k) + f_i(x + l_k) + S_{i_k}(x)$$
3. P_k computes his share of γ as $\gamma_k = V_k(d_k) - g_k$.

Fig. 3. An information theoretic, OLE-based multiplication protocol for MPC

4.1 Evaluation

In this section we evaluate the proposed protocol against the set of security criteria given in section 3.2. We note that all four of these criterion evaluate the specific multiplication protocol and not the actual MPC itself. That is, we evaluate the multiplication protocol only and assume that participants have not yet reconstructed the actual output of the MPC.

Correctness At the end of our protocol each participant, P_k, will now have a share of the polynomial: $\Gamma(x) = h_i(x) + x_i f_j(x)$. Since the free term of $h_i(x)$ is equal to zero, we can say that $\Gamma(0) = x_i f_j(0)$. Now, $f_j(x)$ has the free term

x_j and both $h_i(x)$ and $f_j(x)$ are of degree at most t. As a result of this we can conclude that correctness is achieved, as each P_k has a share to a polynomial, $\Gamma(x)$, of degree at most t, with free term equal to $x_i \cdot x_j$.

Privacy

Theorem 1. *A set of t participants, not including P_i or P_j, cannot compute any information relating to x_i or x_j.*

In order to prove this we must first show that the modified OLE protocol is secure. Following this, we need to prove that a set of t shares relating to the multiplication reveals no information.

Proof. Suppose that a given participant, P_k executes the multiplication protocol with P_i. After sending l_k to P_i he receivers the polynomial $V_k(x) = h_i(k)+f_i(x+l_k) + S_{i_k}(x)$ which we can simplify as $V_k(x) = v_k + z_k x$. Let $S_{i_k}(x) = \kappa_k + \omega_k x$ and recall that $f_i(x) = x_i \cdot x$, then we can rewrite the equation as $V(x) = h_i(k) + x_i l_k + \kappa_k(x_i + \omega_k)x$. This gives P_k the following information:

$$\begin{aligned} v_k &= h_i(k) + x_i l_k + \kappa_k \\ z_k &= x_i + \omega_k \end{aligned}$$

Since the values ω_k and κ_k (as well as the coefficients of $h_i(x)$) are chosen at random, P_k cannot gain any information from the above equations. The next step in the protocol is for P_k to compute $\gamma_k = V(d_k) - g_k$, which can be written as $\gamma_k = h_i(k) + x_i x_{i_j}$. Individually, this gives no information to P_k as he does not know the value of either $h_i(k)$ or x_i, it remains to be seen if a coalition of participants can compute any information.

Without loss of generality suppose that the first set of t participants, $P_1, \cdots, P_t$ pool their information together. Let $h_i(x) = m_1 x + m_2 x^2 + \cdots + m_t x^t$, then the coalition can compute the following system:

$$\begin{aligned} \gamma_1 &= x_i \cdot x_{i_1} + m_1 + m_2 + \cdots + m_t \\ \gamma_2 &= x_i \cdot x_{i_2} + 2m_1 + 4m_2 + \cdots + 2^t m_t \\ &\vdots \\ \gamma_t &= x_i \cdot x_{i_t} + tm_1 + t^2 m_2 + \cdots + t^t m_t \end{aligned}$$

Due to the perfectness of of Shamir's secret sharing scheme [4,12] the above system does not reveal any information to the participants as they effectively have a set of t shares relating to a degree t polynomial. This becomes even more evident when we take into account that $x_{i_k} = f_j(k)$ meaning that each P_k has a share of the polynomial $\Gamma = x_i f_j(x) + h_i(x)$.

The end result being that a coalition of t participants cannot reduce their uncertainty of x_i. The same is also true for x_j, as collectively the coalition only has t shares of $f_j(x)$.

Privacy with respect to P_i

Theorem 2. *A set of t participants, including P_j, cannot compute any information relating to x_i.*

Proof. The proof of this is similar to the proof of theorem 1 along with some extra information. Namely, we now assume that the coalition of participants has the values of both $f_j(x)$ and, consequently x_j. The first, obvious ramification of this is that the coalition now know the shares of every other participant relating to x_j. This actually gives them no advantage, in regards to the OLE, as they do not know (and cannot compute) the values given to the other participants by the initialiser (namely d_k and g_k). We therefore only need to prove that knowing $f_j(x)$ reveals no information relating to x_i.

As before, at the end of the protocol each participant has a share to the polynomial $\Gamma(x) = x_i f_j(x) + h_i(x)$. It is easy to see that if the coalition can compute $\Gamma(x)$ or even $h_i(x)$ then they can easily compute x_i. However, the coalition do not hold direct shares to $h_i(x)$, so even knowing $h_i(0) = 0$ gives them nothing. Furthermore, to compute any information relating to $\Gamma(x)$ would require the coalition to compute a solution to the system given in the proof of theorem 1.

Computing a solution to this system is analogous to solving a system of equations composed of $t + 1$ unknowns (x_i and the coefficients of $h_i(x)$) and t equations. We can therefore conclude that a set of participants, including P_j cannot reduce their uncertainty of x_i.

Privacy with respect to P_j

Theorem 3. *A set of t participants, including P_i, cannot compute any information relating to x_j.*

Proof. In the proof of theorem 1 it was shown that the modified OLE is secure, therefore to prove the above theorem we need to show that a coalition of t participants, including P_i, with t shares relating to $\Gamma(x) = x_i f_j(x) + h_i(x)$ and t shares of $f_j(x)$ cannot compute any information relating to x_j. First, let $f_j(x) = x_j + W_1 x + \cdots + W_t x^t$ and assume, as before, that a coalition composed of the first t participants (which includes P_i) pool their knowledge. They can construct the following system from their shares of $\Gamma(x)$:

$$\begin{aligned}
\gamma_1 &= x_i \cdot (x_j + W_1 + \cdots + W_t) + h_i(1) \\
\gamma_2 &= x_i \cdot (x_j + 2W_1 + \cdots + 2^t W_t) + h_i(2) \\
&\vdots \\
\gamma_t &= x_i \cdot (x_j + tW_1 + \cdots + t^t W_t) + h_i(t)
\end{aligned}$$

From the shares of $f_j(x)$ we get:

$$
\begin{aligned}
x_{j_1} &= x_j + W_1 + \cdots + W_t \\
x_{j_2} &= x_j + 2W_1 + \cdots + 2^t W_t \\
&\vdots \\
x_{j_t} &= x_j + tW_1 + \cdots + t^t W_t
\end{aligned}
$$

It is easy to see that the two systems are actually linearly dependent. Since the values of x_i and $h_i(x)$ are known to the coalition, this results in a system composed of $t+1$ unknowns (the coefficients of $f_j(x)$) and only t linearly independent equations. The net result of this is that each value of x_j is, from the point of view of the coalition, equally likely. Meaning that they cannot compute any information relating to x_j.

References

1. Beaver, D.: Efficient Multiparty Protocols Using Circuit Randomization. In: Feigenbaum, J. (ed.) Advances in Cryptology — CRYPTO '91. pp. 420–432. Springer Berlin Heidelberg (1992)
2. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. STOC '88, ACM, New York, NY, USA (1988)
3. Benaloh, J.C.: Secret sharing homomorphisms: Keeping shares of a secret secret (extended abstract). In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO' 86. pp. 251–260. Springer Berlin Heidelberg, Berlin, Heidelberg (1987)
4. C. L. F. Corniaux, H. Ghodosi: An entropy-based demonstration of the security of Shamir's secret sharing scheme. In: 2014 International Conference on Information Science, Electronics and Electrical Engineering. vol. 1, pp. 46–48 (Apr 2014)
5. Chang, Y.C., Lu, C.J.: Oblivious polynomial evaluation and oblivious neural learning. In: Boyd, C. (ed.) Advances in Cryptology — ASIACRYPT 2001. pp. 369–384. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
6. Chaum, D., Crépeau, C., Damgård, I.: Multiparty Unconditionally Secure Protocols. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. pp. 11–19. STOC '88, ACM, New York, NY, USA (1988)
7. Döttling, N., Ghosh, S., Nielsen, J.B., Nilges, T., Trifiletti, R.: Tinyole: Efficient actively secure two-party computation from oblivious linear function evaluation. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 2263–2276. CCS '17, ACM, New York, NY, USA (2017)
8. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing. pp. 218–229. STOC '87, ACM, New York, NY, USA (1987), http://doi.acm.org/10.1145/28395.28420
9. Hanaoka, G., Imai, H., Mueller-Quade, J., Nascimento, A.C.A., Otsuka, A., Winter, A.: Information Theoretically Secure Oblivious Polynomial Evaluation: Model, Bounds, and Constructions. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) Information Security and Privacy. pp. 62–73. Springer Berlin Heidelberg (2004)

10. Naor, M., Pinkas, B.: Oblivious Transfer and Polynomial Evaluation. In: Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing. pp. 245–254. STOC '99, ACM, New York, NY, USA (1999)
11. Rabin, T., Ben-Or, M.: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In: Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing. STOC '89, ACM, New York, NY, USA (1989)
12. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)
13. Tonicelli, R., Nascimento, A.C.A., Dowsley, R., Müller-Quade, J., Imai, H., Hanaoka, G., Otsuka, A.: Information-theoretically secure oblivious polynomial evaluation in the commodity-based model. International Journal of Information Security 14(1), 73–84 (Feb 2015)