Multi-Party Computation with Conversion of Secret Sharing

Hossein Ghodosi[1], Josef Pieprzyk[2], and Ron Steinfeld[2]

[1] Department of Information Technology, School of Business
James Cook University, Townsville, Qld 4811, Australia
[2] Department of Computing
Center for Advanced Computing – Algorithms and Cryptography
Macquarie University, Sydney, NSW 2109 Australia

Abstract. Classical results in unconditionally secure multi-party computation (MPC) protocols with a passive adversary indicate that every n-variate function can be computed by n participants, such that no set of size $t < n/2$ participants learns any additional information other than what they could derive from their private inputs and the output of the protocol.
We study unconditionally secure MPC protocols in the presence of a passive adversary in the trusted setup ('semi-ideal') model, in which the participants are supplied with some auxiliary information (which is random and independent from the participant inputs) ahead of the protocol execution (such information can be purchased as a "commodity" well before a run of the protocol). We present a new MPC protocol in the trusted setup model, which allows the adversary to corrupt an arbitrary number $t < n$ of participants. Our protocol makes use of a novel subprotocol for converting an additive secret sharing over a field to a multiplicative secret sharing, and can be used to securely evaluate any n-variate polynomial G over a field F, with inputs restricted to non-zero elements of F. The communication complexity of our protocol is $O(\ell \cdot n^2)$ field elements, where ℓ is the number of non-linear monomials in G. Previous protocols in the trusted setup model require communication proportional to the number of multiplications in an arithmetic circuit for G; thus, our protocol may offer savings over previous protocols for functions with a small number of monomials but a large number of multiplications.

Keywords: Multi-Party Computation; Hybrid Secret Sharing Schemes; Unconditional Security.

1 Introduction

Secure multi-party computation (MPC) protocols enable a set of n mutually distrusting participants $P_1, \ldots, P_n$, each with their own private input x_i, to compute a function $Y = F(x_1, \ldots, x_n)$, such that at the end of the protocol, all participants learn the correct value of Y, while their inputs remain private. Assuming that a single trusted party exists, a simple solution to the MPC problem can be achieved as follows. All participants will forward their inputs to the trusted party, who would compute the function value and announce the result to each participant. However, this solution, also known as the *ideal process*, is not acceptable, when the participants are not able to agree on a trusted party.

The design of secure MPC protocols has been the subject of investigation by a great number of researchers, and many protocols have been presented in the literature. In terms of security, these protocols can be classified into two broad categories: (i) *computationally secure*, and (ii) *unconditionally secure* MPC protocols. Irrespective of the classification, the security model used includes the adversary, who may corrupt a subset of participants. For computationally secure MPC protocols, the assumption is that the adversary is polynomially bounded (i.e. breaking the security of the protocol implies that the adversary, in polynomial time, is able to solve a problem that is believed to be intractable). Unconditionally secure MPC protocols are intrinsically secure; that is, no matter how much time and computing power is available to the adversary, they cannot break the system by any method better than by guessing private inputs.

Two common types of adversaries, namely *active* and *passive*, have been studied in the literature. Participants who are corrupted by an active adversary may behave arbitrarily or maliciously, and may deviate from the protocol at any time. That is, an active adversary threatens the correctness of the protocol. On the other hand, participants who are corrupted by a passive adversary behave honestly and properly follow the protocol; however, they are curious in learning additional information. That is, a passive adversary threatens the privacy of the participants.

1.1 Background

The concept of secure MPC was introduced by Yao [36] in the context of two parties, and then generalized by Goldreich, Micali, and Wigderson [22] for any number of participants. Although they have provided a solution to every general MPC problem, from the efficiency point of view, their solutions are not practical for real-life applications. Moreover, the security achieved in [36] and [22] is computational.

In 1988, Ben-Or, Goldwasser, and Wigderson [11] and Chaum, Crépeau, and Damgård [12], independently studied unconditionally secure MPC protocols. Their model of computation is a complete synchronous network of n participants. The private communication channels between participants are secure (i.e. they cannot be read or tampered with by other participants). In their solution, a generic real-life MPC protocol consists of the following three phases [11].

Initialization – Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n participants who wish to compute a function $Y = F(x_1, \ldots, x_n)$, where P_i holds private input x_i. Without loss of generality, we assume that all input variables are elements of a finite field $GF(p)$, where p is a prime number, and the function F can be computed by a circuit over $GF(p)$ using the field operations $+$, $\times$, and the inverse operations and constants from $GF(p)$.

Each participant P_i $(i = 1, \ldots, n)$ distributes his private input x_i amongst all participants in a *t-private* manner. More precisely, using Shamir's secret sharing scheme [34], P_i chooses a random t-degree polynomial $f_i(x) = x_i + a_1 x + \ldots, +a_t x^t$, and for all $j = 1, \ldots, n$, P_i gives $s_{i,j} = f_i(j)$ to participant P_j.

Distributed Computation – Calculation of every linear function is straightforward, since a linear combination of t-degree polynomials is a t-degree polynomial. Computation of non-linear functions, however, is not so straightforward. Firstly, the multiplication of two t-degree polynomials is a $2t$-degree polynomial. The second problem is that the resulting $2t$-degree polynomial is not a random polynomial. Current solution to these problems implies $n \geq 2t + 1$, and the computation protocols are highly interactive.

Reconstruction of the function value – The function $F(x_1, \ldots, x_n)$ can be represented as a polynomial containing the sum of products, and the participants can collectively evaluate first products (product gates), and then sums (sum gates), finally obtaining the shares of the function value Y. In order to reconstruct Y, a sufficiently large set of participants can pool their shares and recover the value Y.

1.2 Security Consideration

Classical results in unconditionally secure MPC protocols (in the standard model, i.e. without any *trusted setup* assumptions) can be found in [11, 12] and are summarized below.

1. In the presence of a passive adversary, no set of size $t < n/2$ of participants learns any additional information, other than what they could derive from their private inputs and the output of the protocol.
2. In the presence of an active adversary, no set of size $t < n/3$ of participants can learn any additional information or disrupt the protocol.

Although there have been some developments in improving the bound $t < n/3$ in MPC protocols with an active adversary [33, 2, 7, 3, 23, 9, 18, 21, 17, 19], it has been confirmed that $t < n/2$ is a tight bound (even under assumption that broadcast channels are available, and allowing a small error probability).

In this work, we study the design of MPC protocols secure against a passive adversary, but we aim at achieving security even against a majority of $t > n/2$ colluding participants; in fact we aim at achieving security even against $t = n - 1$ colluding participants. To circumvent the above upper bound $t < n/2$ shown to hold in the standard model, we assume a modified MPC model with a *trusted setup assumption*. In this trusted setup model, there is a preliminary setup phase executed before the participants receive their protocol inputs, in which a trusted party supplies each party (privately) with some correlated *auxiliary information*. Since the auxiliary information is independent of the participants' private inputs and the function to be computed, it can be distributed to the participants 'offline', before the participants know their protocol inputs. Thus, the trusted setup MPC model is can be viewed as an intermediate model between the 'ideal process' and the standard MPC model. Previous work has shown that, in the trusted setup model, it is possible to design MPC protocols secure against a dishonest majority: the work of Kilian [29] first showed how to build such protocols using a black-box subprotocol for *Oblivious Transfer* (OT) (see also [14]), and in turn, Beaver [5] showed how build an unconditionally secure OT protocol in the trusted setup model, using his reduction from '$\binom{2}{1}$'-OT to precomputed '$\frac{1}{2}$'-OT.

1.3 Efficiency Consideration

Since the invention of MPC protocols, their efficiency has been a major issue. Indeed, the earlier protocols [36, 22, 11, 12] are *polynomial time*, however, they are far from being *practical*, when the group of participants is large. This is because MPC protocols are highly interactive, even if all participants behave honestly and follow the protocol properly (they can still be curious in learning additional information).

There are two factors, namely *round complexity* and *communication complexity*, relevant to the efficiency of MPC protocols. The communication complexity of MPC protocols is the maximum number of bits transmitted during the execution of the protocol. Similarly, the round complexity is the maximum number of rounds necessary to execute the protocol. During the last two decades, considerable efforts have been made to improve the efficiency of MPC protocols, either by reducing the round complexity (e.g. [31, 4, 32, 16, 30]), or improving the communication complexity (e.g. [21, 25, 24, 26]) of MPC protocols. It is also worth mentioning that MPC protocols with constant-round has been discussed by some researchers (e.g., [1, 8]).

To our knowledge, previous MPC protocols in the literature that are secure against a dishonest majority in the trusted setup model (see [20] for classical results and [27, 28] for more recent work) make black box use of an Oblivious Transfer (OT) subprotocol, which is then implemented in the trusted setup model using Beaver's 'pre-computation' construction [5], where the precomputation is done during the setup phase. In these protocols, the protocol inputs are shared additively among the n parties, and the function to be computed F is encoded as an arithmetic circuit over some ring R (typically $R = GF(2)$) consisting of addition and multiplication gates. Then, for each multiplication gate, the protocol runs $O(n^2)$ OT instances (for strings of length $O(\log |R|)$ bits) between every pair

of parties to compute an additive sharing of the gate output (this holds for $R = GF(2)$ [20]; for larger fields or general rings, the unconditionally secure protocols from [27] require $O((\log |R|+k)\cdot n^2)$ OTs, where k is a statistical security parameter). As a consequence, the total communication complexity of these protocols is $O(m\cdot n^2)$ field elements over $R = GF(2)$ (or $O((\log p+k)\cdot m\cdot n^2)$ over $R = GF(p)$ for general p), where m is the number of multiplication gates in the circuit for the function F. In contrast, our protocol works over $GF(p)$, and is not based on OT but on converting multiplicative to additive secret sharing. It requires a total communication of $O(\ell\cdot n^2)$ field elements, where ℓ denotes the number of *monomials* in the polynomial representation of F. Consequently, our protocol may have lower complexity than previous protocols for functions F with a small number of monomials ℓ but a large number of multiplications m. Note that it is possible to have $\ell = O(1)$ whereas $m = \Omega(n)$; for instance, consider the function $F(x_1, \ldots, x_n) = x_1x_2 \cdots x_{n/2} + x_{n/2}x_{n/2+1} \cdots x_n$ (on the other hand, in general a function that can be evaluated with m multiplications and $O(m)$ additions can have $\ell = \Omega(2^m)$ monomials, in which case previous protocols would have lower complexity).

Other relevant works to ours are [6, 15], which utilize some auxiliary information to achieve efficiency. Similar to the commodity-based computation of Beaver [6], the auxiliary information in our protocols can be purchased as commodity, or can be generated by some random generator functions (as in [15]). Finally, we note that the notion of converting shares between different secret sharing schemes is also studied in [13], although for different secret sharing schemes than ours.

1.4 Our Contribution

We study the MPC problem in an unconditionally secure environment. Let $F : (GF(p)^*)^n \rightarrow GF(p)$ denote a n-variate polynomial over $GF(p)$ (with inputs restricted to $GF(p)^*$) having ℓ non-linear monomials. That is, we represent the function F as $F(x_1, \ldots, x_n) = F_L(.) + F_{C_1}(.) + \ldots + F_{C_\ell}(.)$, where $F_L(.)$ denotes the linear component, and $F_{C_i}(.)$ $(i = 1, \ldots, \ell)$ denotes monomials. We demonstrate the followings:

1. The linear component, and every monomial (regardless of its depth), can be computed with no interaction. This is achieved by introducing a *hybrid secret sharing scheme*, which consists of two (namely an additive and a multiplicative) secret sharing schemes.
2. Calculation of the function value, however, requires the combination of results obtained from the additive and multiplicative secret sharing schemes. The conversion of multiplicative secret sharing into the additive secret sharing is done with the help of some auxiliary information distributed to the participants in a trusted setup phase. The auxiliary information is completely independent from the secret input of participants, and can be purchased as "commodity" well ahead of a run of the protocol.
3. Our MPC protocol uses allows the adversary to corrupt up to $n - 1$ participants. This is a significant improvement over the security of unconditionally secure MPC protocols without a trusted setup phase, and, unlike previous protocols in the trusted setup model, is achieved without using Oblivious Transfer (OT) as a subroutine, but with a share conversion subprotocol. How our contribution fits in the MPC scenarios is illustrated in Figure 1. As shown, our trusted setup model may be viewed as a 'semi-ideal' model.

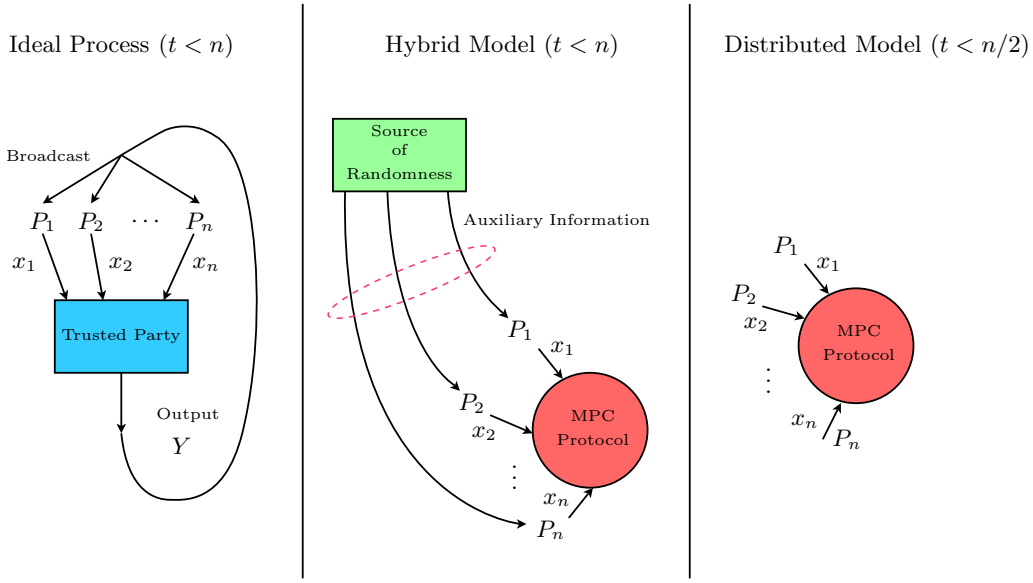

Fig. 1. Different models of MPC

4. The communication complexity of our protocol is $O(\ell \cdot n^2)$ field elements, where ℓ is the number of monomials in F, as defined above. Although we may not achieve efficiency for functions with a large number of monomials, for functions with a small number of monomials, regardless of their depths, our protocol is very efficient. More precisely, for each monomial, we need one round of interaction amongst the participants (note that computation of functions that consist of only one monomial can be achieved non-interactively, and without the help of any auxiliary information).

In the next section, we will provide basic definitions. In Section 3, we will introduce the concept of hybrid secret sharing schemes, and demonstrate an implementation of a hybrid secret sharing containing an additive and a multiplicative (n, n)-threshold scheme. In Section 4, we will utilize the proposed hybrid secret sharing to compute every component of a function in a non-interactive manner. Section 5 discusses a possible protocol for converting the shares of a secret associated with a multiplicative (n, n)-threshold scheme to its corresponding additive (n, n)-threshold scheme. In Section 6, we will demonstrate an implementation of MPC protocols, using the proposed hybrid secret sharing. Section 7 concludes the work.

2 Preliminaries

Our model of computation is a complete synchronous network with private channels available between every pairs of n collaborating participants. Participants have unlimited computing power and properly follow the protocol, but they are curious in learning more information regarding the other participant private inputs (i.e. the adversary is passive with unlimited computing capabilities).

Definition 1. *A multi-party protocol is t-private if after completion of the protocol no subset of t participants learns any information (about uncorrupted participant private inputs) more than what they could derive from their private inputs and the output of the protocol.*

2.1 Sharing a Secret

Secret sharing allows a set of n participants $\mathcal{P} = \{P_1, \ldots, P_n\}$ to share a piece of information in such a way that only authorized subsets of the participants can recover the secret. The partial information s_i, given to participant P_i, is called the *share* of participant P_i from the secret. The set of all authorized subsets that are able to recover the secret is called the *access structure*. The set $\mathcal{K}$ of all possible values of the secret is called the *secret domain*, and the set $\mathcal{S}$ of all possible values of the shares is called the *share domain*. We shall assume throughout that $\mathcal{P}$, $\mathcal{K}$ and $\mathcal{S}$ are all finite sets.

A secret sharing scheme in which every subset, $\mathcal{A} \subseteq \mathcal{P}, (|\mathcal{A}| = t)$, is an authorized set is called a (t, n)-threshold scheme. Hence, a (t, n)-threshold scheme determines a set of functions $F_A : \mathcal{S}^t \to \mathcal{K}$ defined for every $A \subseteq \{1, 2, \ldots, n\}$ with $|A| = t$, such that for any given set of t shares the function value is the secret, i.e. $K = F_A(s_{i_1}, \ldots, s_{i_t})$.

Definition 2. [10] *Let $\top$ and $\perp$ be binary functions on elements of the secret domain $\mathcal{K}$ and the share domain $\mathcal{S}$, respectively. A (t, n)-threshold scheme has the ($\top$, $\perp$)-homomorphism property if for every subset A, whenever $K = F_A(s_{i_1}, \ldots, s_{i_t})$ and $K' = F_A(s'_{i_1}, \ldots, s'_{i_t})$, then*

$$K \top K' = F_A\left((s_{i_1} \perp s'_{i_1}), \ldots, (s_{i_t} \perp s'_{i_t})\right).$$

That is, the composition of the shares of the secrets K and K' are the shares of the composition of the secrets K and K'. One can check that Shamir's (t, n)-threshold scheme is $(+, +)$-homomorphic, but it is not $(\times, \times)$-homomorphic.

3 A Hybrid Secret Sharing Scheme

Ordinary secret sharing schemes utilize a functionality for setting the system. For example, in Shamir threshold scheme, the shares of participants are points that satisfy a polynomial, where the secret is the constant term of the polynomial. The secret can be reconstructed by interpolating the associated polynomial. Now, we introduce the concept of a hybrid secret sharing scheme, where a secret can be shared using two sets of functions.

Definition 3. *Let $\mathcal{K}$ be the domain of possible secrets, and let $\mathcal{S}$ be the domain of possible shares. A hybrid (t, n)-threshold scheme determines two sets of functions $F_A : \mathcal{S}^t \to \mathcal{K}$ and $G_A : \mathcal{S}^t \to \mathcal{K}$ defined for every $A \subseteq \{1, \ldots, n\}$ with $|A| = t$, such that for any given set of t shareholders each function defines the value of the secret, i.e., $K = F_A(s_{i_1}, \ldots, s_{i_t}) = G_A(s'_{i_1}, \ldots, s'_{i_t})$.*

We refer to such secret sharing scheme as a (F, G)-hybrid (t, n)-threshold scheme.

Definition 4. *A (F, G)-hybrid (t, n)-threshold scheme is perfect, if every subset of $t - 1$ or fewer shareholders each knowing two shares (associated with functions F and G) cannot learn any useful information about the secret.*

Definition 5. *A (F, G)-hybrid (t, n)-threshold scheme is F-convertible (respectively, G convertible), if given the shares associated to function F (respectively, function G) enables every authorized set to compute their shares associated to function G (respectively, F), without revealing any useful information about the secret.*

For the purpose of MPC protocols, in the following, we will devise a (F, G)-hybrid (n, n)-threshold scheme over $GF(p)$, where the functions F and G are simply the modular additions and multiplications over $GF(p)$. By abuse of language, we may refer to it as a $(+, \times)$-hybrid threshold scheme.

3.1 An Additive (n, n)-Threshold Scheme

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n participants, and let $K \in GF(p)$ be the secret, where p is a prime integer.

Definition 6. *For every secret $K \in GF(p)$, an additive (n, n)-threshold scheme determines a set of functions $F_{\mathcal{P}} : GF(p)^n \to GF(p)$ defined for the set of n shareholders $\mathcal{P} = \{P_1, \ldots, P_n\}$ such that $K = F_{\mathcal{P}}(s_1, \ldots, s_n) = s_1 + \ldots + s_n \pmod p$.*

An implementation of this secret sharing scheme is shown in Figure 2. This scheme is perfect [35], as the shares of every set of $n-1$ participants are completely independent from the secret.

Share Distribution – The dealer chooses $n-1$ shares $s_1, \ldots, s_{n-1}$ at random from all possible values in $GF(p)$, and computes $K = s_n + \Sigma_{i=1}^{n-1} s_i \pmod p$. The dealer sends (privately) share s_i to participant P_i $(i = 1, \ldots, n)$.

Secret Reconstruction – All participants pool their shares and reconstruct the secret $K = \Sigma_{i=1}^{n} s_i \pmod p$.

Fig. 2. An additive (n, n)-threshold scheme.

Theorem 1. *Given the proposed additive (n, n)-threshold secret sharing. If $s_1, \ldots, s_n$ and $s'_1, \ldots, s'_n$ are shares of participants $P_1, \ldots, P_n$ associated with secrets K and K', respectively, then $(s_1 + s'_1), \ldots, (s_n + s'_n)$ are the shares of participants $P_1, \ldots, P_n$ associated with secret $K + K'$. That is, the additive (n, n)-threshold secret sharing has the $(+, +)$-homomorphism property.*

Proof. The proof is simple and skipped.

3.2 A Multiplicative (n, n)-Threshold Scheme

Let $GF(p)^*$ denote the set of non-zero elements of $GF(p)$, which form a group under multiplication. By replacing the additive group $GF(p)$ with the multiplicative group $GF(p)^*$, we define a multiplicative (n, n)-threshold scheme as follows.

Definition 7. *For every secret $K \in GF(p)^*$, a multiplicative (n, n)-threshold scheme determines a set of functions $G_{\mathcal{P}} : (GF(p)^*)^n \to GF(p)^*$ defined for the set of shareholders, $\mathcal{P} = \{P_1, \ldots, P_n\}$ such that $K = G_{\mathcal{P}}(s_1, \ldots, s_n) = s_1 \times \ldots \times s_n \pmod p$.*

An implementation of this secret sharing scheme is shown in Figure 3.

Share Distribution – The dealer chooses $n-1$ independent and uniformly random shares $s_1, \ldots, s_{n-1}$ from $GF(p)^*$, and computes $s_n = K \times (\Pi_{i=1}^{n-1} s_i)^{-1} \pmod p$. For $i = 1, \ldots, n$, the dealer privately sends share s_i to participant P_i.

Secret Reconstruction – All participants pool their shares and reconstruct the secret $K = \Pi_{i=1}^{n} s_i \pmod p$.

Fig. 3. A multiplicative (n, n)-threshold scheme.

Theorem 2. *Given the proposed multiplicative (n, n)-threshold secret sharing. If $s_1, \ldots, s_n$ and $s'_1, \ldots, s'_n$ are shares of participants $P_1, \ldots, P_n$ associated with secrets K and K', respectively, then $(s_1 \times s'_1), \ldots, (s_n \times s'_n)$ are the shares of participants $P_1, \ldots, P_n$ associated with secret $K \times K'$. That is, the multiplicative (n, n)-threshold secret sharing has $(\times, \times)$-homomorphism property.*

Proof. The proof is simple and skipped.

Theorem 3. *The $(+, \times)$-hybrid threshold secret sharing consisting of an additive (n, n)-threshold scheme defined in Figure 2 and a multiplicative (n, n)-threshold scheme defined in Figure 3, is perfect.*

Proof. Let K be a secret, and let $s_1, \ldots, s_n$ and $m_1, \ldots, m_n$ be the shares of participants $P_1, \ldots, P_n$, associated with the additive and multiplicative schemes, respectively. Also, w.l.o.g. let $P_1, \ldots, P_{n-1}$ be a group of collaborating participants who wish to learn some information about K. The most that they can compute is that $s_n + \Sigma_{i=1}^{n-1} s_i = m_n \times \Pi_{i=1}^{n-1} m_i$, where s_n and m_n are unknown to the collaborating participants. However, for every possible value of s_n (respectively, m_n), there exists one value for m_n (respectively, s_n), and thus one value for the secret. Therefore, the set of $n-1$ collaborating participants cannot decrease their uncertainty about the secret.

4 Function Computation using Hybrid Secret Sharing

In this section, we demonstrate how to perform the computation of linear functions and monomials.

Additive (n, n)-threshold scheme –
P_i chooses $n-1$ shares $s_{i,j}$ $(j = 1, \ldots, n,\ j \neq i)$ at random from all possible values in $GF(p)$, and computes his share, $s_{i,i}$, using

$$x_i = s_{i,i} + \Sigma_{\substack{j=1 \\ j \neq i}}^{n} s_{i,j} \pmod{p}$$

Multiplicative (n, n)-threshold scheme –
P_i chooses $n-1$ non-zero shares $m_{i,j}$ $(j = 1, \ldots, n,\ j \neq i)$ at random from all possible values in $GF(p)$, and computes his share, $m_{i,i}$, using

$$x_i = m_{i,i} \times (\Pi_{\substack{j=1 \\ j \neq i}}^{n} m_{i,j}) \pmod{p}$$

Fig. 4. Sharing input x_i by participant P_i $(i = 1, \ldots, n)$

4.1 Computation of Linear Functions

Assume that the secret inputs $x_1, \ldots, x_n$ of participants $P_1, \ldots, P_n$ are shared, using the additive (n, n)-threshold scheme of Figure 4. Let $s_{i,j}$ and $s_{k,j}$ be P_j's shares from the secret inputs x_i and x_k, respectively. Computation of every linear function can be done in the following way:

- In order to compute $x_i + x_k$, each participant P_j computes $s_j^{i+k} = s_{i,j} + s_{k,j}$, which is the share of P_j associated with $x_i + x_k$. This is because the additive (n, n)-threshold scheme is $(+, +)$-homomorphic.
- For every known scalar $c \in GF(p)$ and each secret input x_i, computation of $c \times x_i$ requires that each participant P_j $(j = 1, \ldots, n)$ calculates $c \times s_{i,j}$ as his share of $c \times x_i$.
- For every scalar $c \in GF(p)$ and each secret input x_i, computation of $c + x_i$ can be done in any of the following ways:
  (i) Share the value c amongst all participants, using the additive (n, n)-threshold scheme. Then each participant P_j $(j = 1, \ldots, n)$ computes $c_j + s_{i,j}$ as his share of $c + x_i$, where c_j is the share of P_j from c.
  (ii) A more efficient way is that only a designated participant, P_ℓ, $\ell \in \{1, \ldots, n\}$ (who is chosen by all participants) adds c to his share from x_i, i.e., computes $c + s_{i,\ell}$.
- Computation of an additive inverse is easy. Every participant P_j computes the additive inverse of his share.

Thus, every linear function with n inputs can be computed with no interaction.

4.2 Computation of Monomials

Assume that the secret inputs $x_1, \ldots, x_n$, of participants $P_1, \ldots, P_n$ are shared, using the multiplicative (n, n)-threshold scheme of Figure 4. Let $m_{i,j}$ and $m_{k,j}$ be P_j's shares from the secret inputs x_i and x_k, respectively. Computation of every multiplication gate can be done in the following way:

- In order to compute $x_i \times x_k$, each participant P_j computes $m_j^{i \times k} = m_{i,j} \times m_{k,j}$, which is the share of P_j associated with $x_i \times x_k$. This is because the multiplicative (n, n)-threshold scheme is $(\times, \times)$-homomorphic.
- For every scalar $c \in GF(p)$ and each secret input x_i, computation of $c \times x_i$ can be done in any of the following ways:
  (i) Share the value c amongst all participants, using the multiplicative (n, n)-threshold scheme. Then each participant P_j $(j = 1, \ldots, n)$ computes $c_j \times m_{i,j}$ as his share of $c \times x_i$, where c_j is the share of P_j from c.
  (ii) A more efficient way is that only a designated participant, P_ℓ, $\ell \in \{1, \ldots, n\}$ (who is chosen by all participants) multiplies c on his share of x_i, i.e., computes $c \times m_{i,\ell}$.
- Computation of a multiplicative inverse is easy. Every participant P_j computes the multiplicative inverse of his share.

Thus, every multiplication gate, regardless of its depth, can be computed with no interaction.

5 Conversion of Multiplicative Shares to Additive Shares

Up to this point, we have shown that for every function $F(x_1, \ldots, x_n) = F_L(.) + F_{C_1}(.) + \ldots + F_{C_\ell}(.)$, the linear component $F_L(.)$, and all monomials $F_{C_i}(.)$ $(i = 1, \ldots, \ell)$ can be computed non-interactively. However, value of the linear component is shared in an additive (n, n)-threshold scheme, while the value of each monomial is shared in a multiplicative (n, n)-threshold scheme. Computing the function value, without revealing any information about the value of each component, requires converting the multiplicative (n, n)-threshold sharing associated with each monomial $F_{C_i}(.)$ $(i = 1, \ldots, \ell)$ to a corresponding additive (n, n)-threshold sharing.

Let $m_1, \ldots, m_n$ be the shares of participants $P_1, \ldots, P_n$, such that $K = \Pi_{i=1}^n m_i \pmod p \in GF(p)^*$. Also assume that the shareholders have been provided with some auxiliary information, which could be completely independent from the secret and from their shares. Our solution is given by the protocol depicted in Figures 5.

Correctness – Each participant P_i $(i = 1, \ldots, n)$ receives $n - 1$ values $\alpha_{i,j} m_j$ from participants P_j $(j = 1, \ldots, n,\ j \neq i)$. Knowing $\alpha_{i,i}$, m_i, and the received information, P_i computes

$$s_i = \Pi_{j=1}^n \alpha_{i,j} m_j = \Pi_{j=1}^n \alpha_{i,j} K,$$

as his share corresponding to an additive (n, n)-threshold scheme. The protocol of Figure 5 is correct, because at the end of the protocol, the sum of the computed shares of all participants is:

$$\Sigma_{i=1}^n s_i = \Sigma_{i=1}^n \left(\Pi_{j=1}^n \alpha_{i,j} K \right) = \left(\Sigma_{i=1}^n \left(\Pi_{j=1}^n \alpha_{i,j} \right) \right) K = K \pmod p.$$

Security – Without loss of generality, let $P_1, \ldots, P_{n-1}$ be the set of $n-1$ participants who collude in order to breach the security of the proposed conversion protocol via learning some information about the secret, K. They collectively know $n - 1$ shares $m_1, \ldots, m_{n-1}$ associated with a multiplicative (n, n)-threshold scheme; auxiliary information $\alpha_{i,j}$ $(i = 1, \ldots, n$ and $j = 1, \ldots, n - 1)$; and $n - 1$ values $v_{i,n} \equiv m_n \alpha_{i,n} \pmod p$ $(i = 1, \ldots, n - 1)$ received from the honest participant P_n. To

Inputs:

- **Shares** – Each participant P_j $(j = 1, \ldots, n)$ owns a share m_j associated to a multiplicative (n, n)-threshold scheme over $GF(p)$, such that $m_1 \times \ldots \times m_n = K \pmod p$, where $K \in GF(p)^*$ is the secret.
- **Auxiliary information** – Each participant P_j $(j = 1, \ldots, n)$ is given a set of n elements $\alpha_{1,j}, \ldots, \alpha_{n,j}$, such that

$$\Sigma_{i=1}^{n} u_i \equiv 1 \pmod p, \text{ where } u_i \equiv \Pi_{j=1}^{n} \alpha_{i,j} \pmod p. \quad (1)$$

  The $\alpha_{i,j}$'s are generated as follows:
  - Pick $u_1, \ldots, u_n$ in $GF(p)$ as shares for an additve (n, n)-threshold sharing of 1, i.e. pick $u_1, \ldots, u_{n-1}$ independently and uniformly at random from $GF(p)$ and compute $u_n \equiv 1 - \sum_{i=1}^{n-1} u_i \pmod p \in GF(p)$.
  - For $i = 1, \ldots, n$, pick $n - 1$ independent and uniformly random elements $\{\alpha_{i,j}\}_{j \neq i}$ from $GF(p)^*$ and compute $\alpha_{i,i} \equiv u_i \cdot (\prod_{j \neq i} \alpha_{i,j})^{-1} \pmod p \in GF(p)$ (note that $\alpha_{i,i} = 0$ if and only if $u_i = 0$).

Conversion:

- Each participant P_j $(j = 1, \ldots, n)$ sends $v_{i,j} = \alpha_{i,j} m_j \pmod p$ (for $i = 1, \ldots, n$) to participant P_i.

Outputs:

- Participant P_i $(i = 1, \ldots, n)$ computes

$$s_i = \prod_{j=1}^{n} v_{i,j} = \Pi_{j=1}^{n} \alpha_{i,j} m_j = u_i K \pmod p$$

  as his share of K, associated to an additive (n, n)-threshold scheme.

Fig. 5. Protocol for converting the shares of a multiplicative (n, n) scheme to an additive (n, n) scheme.

demonstrate the security, we show that all these known values can be perfectly simulated by the collusion $P_1, \ldots, P_{n-1}$ by itself, independently of the secret K.

Clearly, the collusion cannot learn anything about K from their shares m_i. This is because of the perfectness of the associated multiplicative (n, n)-threshold scheme: the $m_1, \ldots, m_{n-1}$ can be simulated by independent and uniformly random elements in $GF(p)^*$. By definition, the $\alpha_{i,j}$ for $i = 1, \ldots, n$ and $j \neq i$ are independent and uniformly random in $GF(p)^*$. Therefore, since $m_n \in GF(p)^*$, the values $v_{i,n} \equiv m_n \alpha_{i,n} \pmod p$ observed by the collusion are independent of m_n and uniform in $GF(p)^*$ – hence, the collusion can simulate $\alpha_{i,j}$ (for $i = 1, \ldots, n$ and $j \notin \{i, n\}$) and $v_{i,n}$ (for $i = 1, \ldots, n-1$) by independent uniformly random elements in $GF(p)^*$. Finally, it remains to simulate the values $\alpha_{i,i} \equiv u_i \cdot (\prod_{j \neq i} \alpha_{i,j})^{-1} \pmod p$ for $i = 1, \ldots, n-1$. Here we use the perfectness of the additive (n, n)-threshold scheme to conclude that the shares $u_1, \ldots, u_{n-1}$ of 1 are independent of each other and of $\alpha_{i,j}$ for $i \neq j$, and uniformly random in $GF(p)$. Therefore, since $(\prod_{j \neq i} \alpha_{i,j})^{-1} \in GF(p)^*$, the collusion can simulate the values $\alpha_{i,i}$ (for $i = 1, \ldots, n-1$) by independent uniformly random elements in $GF(p)$. It follows that the set of $n - 1$ colluding participants cannot learn any information about the secret K, as claimed.

6 MPC Protocols With Hybrid Secret Sharing

Now we are going to show a potential application of hybrid secret sharing to MPC problem.

Similar to [11, 12], our model of computation is a complete synchronous network with private channels available between every pairs of n collaborating participants. Participants have unlimited computing power and properly follow the protocol, but they are curious in learning more information regarding the other participant private inputs (i.e. the adversary is passive with unlimited computing capabilities).

We note that the security of our multiplicative to additive sharing conversion protocol from Sec. 5 relies on the shared secret being non-zero. Therefore, in the following MPC protocol, we restrict the private inputs of all participants to be non-zero (i.e. elements of $GF(p)^*$). This implies that the value of all monomials in the function F (when evaluated at the private inputs) is in $GF(p)^*$, as required. Although this restriction prevents our protocol from being applied non-trivially over $GF(2)$, it is still possible to use our protocol to compute arbitrary functions over $GF(2)$ by encoding them as polynomials over a larger field such as $GF(5)$, see Remark 1 below.

Initialization – Each participant P_i $(i = 1, \ldots, n)$ distributes his private input $x_i \in GF(p)^*$ amongst all participants, using the additive and multiplicative (n, n)-threshold schemes – refer to Figures 4.

Computation – In order to compute the function $F(x_1, \ldots, x_n) = F_L(.) + F_{C_1}(.) + \ldots, F_{C_\ell}(.)$, each participant P_i $(i = 1, \ldots, n)$ privately computes $F_L(.)$ and all monomials $F_{C_j}(.)$ $(j = 1, \ldots, \ell)$.

Reconstruction of the function value – All participants collectively have the values of each component of the function. The linear component is shared in additive (n, n)-threshold form, while each monomial is shared in multiplicative (n, n)-threshold form. For every monomials (if any), participants convert their multiplicative shares to additive shares.

Let $A_{i,j}$ be the share of participant P_i associated with monomial $F_{C_j}(.)$, in an additive (n, n)-threshold format. Now, P_i computes $Y_i = A_{i,0} + A_{i,1} + \ldots, A_{i,\ell}$, where $A_{i,0}$ is the share of P_i associated with the linear component $F_L(.)$ (if it exists). Since each participant has a share of the function value associated to an additive (n, n)-threshold scheme, they can pool their shares and compute the function value, using

$$Y = \Sigma_{i=1}^{n} Y_i \pmod{p}.$$

The above considerations lead us to the following statement.

Corollary 1. *Let $F : (GF(p)^*)^n \to GF(p)$ denote a n-variate polynomial over $GF(p)$ (with inputs restricted to $GF(p)^*$) having ℓ non-linear monomials. Then, assuming a setup phase in which an auxiliary information (which is independent of the function inputs and consists of $O(\ell \cdot n^2)$ elements of $GF(p)$) is privately distributed among the n participants, the function F can be computed by the n participants such that no subset of $n-1$ participants can learn any additional information, other than what they can learn from their inputs and the protocol's output. The protocol has a total communication complexity of $O(\ell \cdot n^2)$ elements of $GF(p)$.*

Remark 1 – Despite the restriction of our protocol to non-zero inputs, it can still be used to compute arbitrary functions over $GF(2)$ by using a suitable encoding over a larger field (although the resulting number of monomials over the larger field will be in general exponential in the depth of the original $GF(2)$ circuit). To see that this is possible, it suffices to show that a two-input NAND gate can be encoded into a polynomial over non-zero inputs over the larger field. Indeed, consider, for example, the polynomial $h(x_1, x_2) = 2x_1^2x_2^2 + 3x_1x_2 + 2$ over the field $GF(5)$. It is easy to verify that $h(2, 2) = h(1, 2) = h(2, 1) = 1$ and $h(1, 1) = 2$, so h computes an encoding of the $GF(2)$ NAND function over $GF(5)$, where we encode the $GF(2)$ values 0 (respectively 1) as the $GF(5)$ non-zero values 2 (respectively 1).

Remark 2 – For security reasons, any set of auxiliary information should be used only once. That is, for computing a function containing ℓ monomials, ℓ sets of auxiliary information should be provided to the participants.

7 Conclusions

We have introduced a new cryptographic tool called hybrid secret sharing that allows to convert multiplicative secret sharing into its additive version. The conversion uses some auxiliary information which is independent from the secret. We have demonstrated that, in an unconditionally secure setting, if the participants are given access to the auxiliary information distributed during a trusted setup phase, we achieve maximum level of security in MPC protocols, demonstrating a new way to obtain maximal security in the trusted setup model.

For future research, one could ask the following questions:

- Can our hybrid secret sharing approach be adapted to achieve security against *active* adversaries in the trusted setup model?
- Hybrid secret sharing is an interesting tool, and its properties need more investigation. In particular, it is interesting to investigate how the conversion depends on the access structure and required homomorphic properties.
- The complexity of our hybrid secret sharing based protocol is proportional to the number of monomials in the generic sum-of-product representation. However, for general arithmetic circuits, the number of monomials grows exponentially with the multiplicative depth of the circuit. If we would like to efficiently extend our approach to arithmetic circuits of an arbitrary depth, then we need a conversion of additive secret sharing into its multiplicative version. So far, we do not know how to do this.

Acknowledgements
Josef Pieprzyk and Ron Steinfeld were supported by the Australian Research Council grant DP0987734.

References

1. Bar-Ilan, J., Beaver, D.: Non-Cryptographic Fault-Tolerant Computing in a Constant Number of Rounds of Interaction. In: 8th Annual ACM Symposium on Principles of Distributed Computing, Edmonton, Alberta, Canada (1989) 201–209
2. Beaver, D.: Multiparty Protocols Tolerating Half Faulty Processors. In Brassard, G., ed.: Advances in Cryptology - Proceedings of CRYPTO'89. Volume 435 of Lecture Notes in Computer Science. Springer-Verlag (1990) 560–572
3. Beaver, D.: Secure Multiparty Protocols and Zero-Knowledge Proof Systems Tolerating a Faulty Minority. Journal of Cryptology **4** (1991) 75–122
4. Beaver, D.: Efficient Multiparty Protocols Using Circuit Randomization. In Feigenbaum, J., ed.: Advances in Cryptology - Proceedings of CRYPTO'91. Volume 576 of Lecture Notes in Computer Science. Springer-Verlag (1992) 420–432
5. Beaver, D.: Precomputing Oblivious Transfer. In Dwork, C., ed.: Advances in Cryptology - Proceedings of CRYPTO 1995. Volume 963 of Lecture Notes in Computer Science. Springer-Verlag (1995) 97–109
6. Beaver, D.: Commodity-Based Cryptography. In: Proceedings of the 29th ACM Annual Symposium on the Theory of Computing (STOC'97). (1997) 446–455
7. Beaver, D., Goldwasser, S.: Multiparty Computation with Faulty Majority. In: the 30th IEEE Symposium on the Foundations of Computer Science (FOCS89). (1989) 468–473
8. Beaver, D., Micali, S., Rogaway, P.: The Round Complexity of Secure Protocols. In: Proceedings of the 22nd ACM Annual Symposium on the Theory of Computing (STOC'90). (1990) 503–513
9. Beaver, D., Wool, A.: Quorum-Based Secure Multi-party Computation. In Nyberg, K., ed.: Advances in Cryptology - Proceedings of EUROCRYPT'98. Volume 1403 of Lecture Notes in Computer Science. Springer-Verlag (1998) 375–390
10. Benaloh, J.: Secret Sharing Homomorphisms: Keeping Shares of a Secret Secret. In Odlyzko, A., ed.: Advances in Cryptology - Proceedings of CRYPTO'86. Volume 263 of Lecture Notes in Computer Science. Springer-Verlag (1987) 251–260
11. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorem for Non-Cryptographic Fault-Tolerant Distributed Computation. In: Proceedings of the 20th ACM Annual Symposium on the Theory of Computing (STOC'88). (1988) 1–10

12. Chaum, D., Crépeau, C., Damgård, I.: Multiparty Unconditionally Secure Protocols. In: Proceedings of the 20th ACM Annual Symposium on the Theory of Computing (STOC'88). (1988) 11–19
13. Cramer, R., Damgård, I., Ishai, Y.: Share Conversion, Pseudorandom Secret-Sharing, and Applications to Secure Computation. In Kilian, J., ed.: 2nd Theory of Cryptography Conference TCC 2005. Volume 3378 of Lecture Notes in Computer Science. Springer-Verlag (2005) 342–362
14. Crépeau, C., van de Graaf, J., Tapp, A.: Committed Oblivious Transfer and Private Multi-Party Computation. In Dwork, C., ed.: Advances in Cryptology - Proceedings of CRYPTO 1995. Volume 963 of Lecture Notes in Computer Science. Springer-Verlag (1995) 110–123
15. Damgård, I., Ishai, Y.: Scalable Secure Multiparty Computation. In Dwork, C., ed.: Advances in Cryptology - Proceedings of CRYPTO 2006. Volume 4117 of Lecture Notes in Computer Science. Springer-Verlag (2006) 501–520
16. Damgård, I., Nielsen, J.: Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption. In Boneh, D., ed.: Advances in Cryptology - Proceedings of CRYPTO 2003. Volume 2729 of Lecture Notes in Computer Science. Springer-Verlag (2003) 247–264
17. Fitzi, M., Garay, J., Maurer, U., Ostrovsky, R.: Minimal Complete Primitives for Secure Multi-party Computation. In Kilian, J., ed.: Advances in Cryptology - Proceedings of CRYPTO 2001. Volume 2139 of Lecture Notes in Computer Science. Springer-Verlag (2001) 80–100
18. Fitzi, M., Hirt, M., Maurer, U.: Trading Correctness for Privacy in Unconditional Multi-Party Computation. In Krawczyk, H., ed.: Advances in Cryptology - Proceedings of CRYPTO'98. Volume 1462 of Lecture Notes in Computer Science. Springer-Verlag (1998) 121–136
19. Fitzi, M., Holenstein, T., Wullschleger, J.: Multi-party Computation with Hybrid Security. In Cachin, C., Camenisch, J., eds.: Advances in Cryptology - Proceedings of EUROCRYPT 2004. Volume 3027 of Lecture Notes in Computer Science. Springer-Verlag (2004) 419–438
20. Goldreich, O.: Foundations of Cryptography, Volume II. Cambridge University Press, 2004.
21. Gennaro, R., Rabin, M., Rabin, T.: Simplified VSS and Fast-track Multiparty Computations with Applications to Threshold Cryptography. In: 17th Annual ACM Symposium on Principles of Distributed Computing. (1998) 101–111
22. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game. In: Proceedings of the 19th ACM Annual Symposium on the Theory of Computing (STOC'87). (1987) 218–229, May 25–27
23. Hirt, M., Maurer, U.: Complete Characterization of Adversaries Tolerable in Secure Multi-Party Computations. In: 16th Annual ACM Symposium on Principles of Distributed Computing (PODC). (1997) 25–34
24. Hirt, M., Maurer, U.: Robustness for Free in Unconditional Multi-party Computation. In Kilian, J., ed.: Advances in Cryptology - Proceedings of CRYPTO 2001. Volume 2139 of Lecture Notes in Computer Science. Springer-Verlag (2001) 101–118
25. Hirt, M., Maurer, U., Przydatek, B.: Efficient Secure Multi-party Computation. In Okamoto, T., ed.: Advances in Cryptology - Proceedings of ASIACRYPT 2000. Volume 1976 of Lecture Notes in Computer Science. Springer-Verlag (2000) 143–161
26. Hirt, M., Nielsen, J.: Upper Bounds on the Communication Complexity of Optimally Resilient Cryptographic Multiparty Computation. In Roy, B., ed.: Advances in Cryptology - Proceedings of ASIACRYPT 2005. Volume 3788 of Lecture Notes in Computer Science. Springer-Verlag (2005) 79–99
27. Ishai, Y., Prabhakaran, M., Sahai, A.: Secure Arithmetic Computation with No Honest Majority. In Reingold, O., ed.: 6th Theory of Cryptography Conference TCC 2009. Volume 5444 of Lecture Notes in Computer Science. Springer-Verlag (2009) 294–314
28. Ishai, Y., Prabhakaran, M., Sahai, A.: Founding Cryptography on Oblivious Transfer – Efficiently. In Wagner, D., ed.: Advances in Cryptology - Proceedings of CRYPTO 2008. Volume 5157 of Lecture Notes in Computer Science. Springer-Verlag (2008) 572–591
29. Kilian, J.: Founding Cryptography on Oblivious Transfer. In: Proceedings of the 20th ACM Annual Symposium on the Theory of Computing (STOC'88). (1988) 20–31
30. Katz, J., Ostrovsky, R., Smith, A.: Round Efficiency of Multi-party Computation with a Dishonest Majority. In Biham, E., ed.: Advances in Cryptology - Proceedings of EUROCRYPT 2003. Springer-Verlag (2003) 578–595
31. Kushilevitz, E.: Privacy and Communication Complexity. In: the 30th IEEE Symposium on the Foundations of Computer Science (FOCS89). (1989) 416–421
32. Kushilevitz, E., Roscèn, A.: A Randomness-Rounds Tradeoff in Private Computation. In Desmedt, Y., ed.: Advances in Cryptology - Proceedings of CRYPTO'94. Volume 839 of Lecture Notes in Computer Science. Springer-Verlag (1994) 397–409
33. Rabin, T., Ben-Or, M.: Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In: Proceedings of the 21th ACM Annual Symposium on the Theory of Computing (STOC'89). (1989) 73–85
34. Shamir, A.: How to Share a Secret. Communications of the ACM **22** (1979) 612–613
35. Stinson, D.: An Explication of Secret Sharing Schemes. Designs, Codes and Cryptography **2** (1992) 357–390

36. Yao, A.: Protocols for Secure Computations. In: the 23rd IEEE Symposium on the Foundations of Computer Science. (1982) 160–164