Round Efficient Unconditionally Secure Multiparty Computation Protocol

Arpita Patra [*] **Ashish Choudhary** [†] **C. Pandu Rangan** [‡]
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai India 600036
Email:{ arpita,ashishc }@cse.iitm.ernet.in, rangan@iitm.ernet.in

Abstract

In this paper, we propose a round efficient *unconditionally secure multiparty computation* (UMPC) protocol in *information theoretic* model with $n > 2t$ players, in the absence of any physical broadcast channel, which communicates $\mathcal{O}(n^4)$ field elements per multiplication and requires $\mathcal{O}(n \log(n) + \mathcal{D})$ rounds, even if up to t players are under the control of an active adversary having *unbounded computing power*. In the absence of a physical broadcast channel and with $n > 2t$ players, the best known UMPC protocol with minimum number of rounds, requires $\mathcal{O}(n^2\mathcal{D})$ rounds and communicates $\mathcal{O}(n^6)$ field elements per multiplication, where $\mathcal{D}$ denotes the multiplicative depth of the circuit representing the function to be computed securely. On the other hand, the best known UMPC protocol with minimum communication complexity requires communication overhead of $\mathcal{O}(n^2)$ field elements per multiplication, but has a round complexity of $\mathcal{O}(n^3 + \mathcal{D})$ rounds. Hence our UMPC protocol is the most round efficient protocol so far and ranks second according to communication complexity. To design our protocol, we use certain new techniques which are of independent interest.

Keywords: Multiparty Computation, Information Theoretic Security, Error Probability.

1 Introduction

Secure Multiparty Computation (MPC): Secure multiparty computation (MPC) allows a set of n players to securely compute an agreed function, even if up to t players are under the control of a centralized adversary. More specifically, assume that the desired functionality can be specified by a function $f : (\{0,1\}^*)^n \rightarrow (\{0,1\}^*)^n$ and player P_i has input $x_i \in \{0,1\}^*$. At the end of the computation of f, P_i gets $y_i \in \{0,1\}^*$, where $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$. The function f has to be computed securely using a protocol where at the end of the protocol all players (honest) receive correct outputs and the messages seen by the adversary during the protocol contain no *additional* information about the inputs and outputs of the honest players, other than what can be computed from the inputs and outputs of the corrupted players. In the *information theoretic model*, the adversary who *actively* controls at most t players, is adaptive, rushing [11] and has *unbounded computing power*. The function to be computed is represented as an arithmetic circuit over a finite field $\mathbb{F}$ consisting of five type of gates, namely addition, multiplication, random, input and output. A MPC protocol securely evaluates the circuit gate-by-gate [5, 22, 2, 22, 17, 19, 4].

The MPC problem was first defined and solved by Yao [23] in his seminal work in two-party scenario. The first generic solutions presented in [16, 9, 14] were based on cryptographic intractibility assumtions. Later, the research on MPC in information theoretic model was initiated by Ben-Or

[*]Financial Support from Microsoft Research India Acknowledged
[†]Financial Support from Infosys Technology India Acknowledged
[‡]Work Supported by Project No. CSE/05-06/076/DITX/CPAN on Protocols for Secure Communication and Computation Sponsored by Department of Information Technology, Govt. of India.

et. al. [5] and Chaum et. al. [8] in two different independent work and carried forward by the works of [22, 2]. Information theoretic security can be achieved by MPC protocols in two flavors –(a) Perfect: The outcome of the protocol is **perfect** in the sense that no probability of error is involved in the computation of the function (b) Unconditional: The outcome of the protocol is correct except with negligible error probability. While Perfect MPC can be achieved iff $t < n/3$ [5], unconditional MPC (UMPC) requires only honest majority i.e $t < n/2$ [22]. In the recent years, lot of research concentrated on designing communication efficient protocols for both perfect and unconditional MPC. Perfect MPC protocols with optimal resilience i.e $t < n/3$ are presented in [17, 19, 4]. UMPC protocols with non-optimal resilience i.e $t < n/3$ are presented in [18, 12]. Finally, UMPC protocols with optimal resilience i.e $t < n/2$ are presented in [11, 3].

Broadcast: Broadcast is a very important primitive and is heavily used in all MPC and UMPC protocols. Broadcast allows a sender to distribute a value x, such that all the players identically receive the same value x (even if the sender is faulty). If a physical broadcast channel is available in the network, then achieving broadcast is very trivial. In such a case, broadcasting ℓ bits requires single round and exactly ℓ bits of communication. But if the broadcast channel is not physically available in the network, then broadcasting an ℓ bit(s) message can be simulated by executing some protocol. In particular, for perfectly (without any error probability) broadcasting ℓ bits, the protocol presented in [6, 7] communicates $\Omega(n^2\ell)$ bits and requires $\Omega(n)$ rounds with $t < n/3$. For unconditionally (with negligible error probability) broadcasting ℓ bits, the protocol presented in [21] communicates $\Omega(n^2\ell + n^6\kappa)$ bits and requires $\Omega(n)$ rounds with $t < n/2$ on the availability of information theoretic PKI setup (information theoretic pseudo-signature), where κ is the error parameter. Recently Fitzi et. al. [13] have proposed multi-valued broadcast where broadcast of ℓ bits requires communication of $\mathcal{O}(\ell n + n\mathcal{B}(n + \kappa))$ bits, provided there exists a broadcast protocol which communicates $\mathcal{B}(b)$ bits for broadcasting a b bit message where $b < \ell$. Thus using the broadcast protocol of [21] as black-box, broadcast protocol of [13] communicates $O(n\ell + n^7\kappa)$ for broadcasting an ℓ bit message and requires $\Omega(n)$ rounds of communication, where $t < \frac{n}{2}$.

Our Motivation: Two important parameters of multiparty computation protocols are *communication complexity* and *round complexity.* These have been the subject of intense study over the past two decades. Establishing bounds on communication and round complexity of secure multiparty computation protocols are of fundamental theoretical interest. Moreover, reducing the communication and round complexity of multiparty computation protocols is crucial, if we ever hope to use these protocols in practice. But looking at the most recent advancements in the arena of MPC, we find that round complexity of MPC protocols has been increased to an unacceptable level in order to reduce communication complexity. For example, the perfect MPC protocol of [4] celebrated for its best known communication complexity, requires round complexity of $\mathcal{O}(n^2 + \mathcal{D})$ where $\mathcal{D}$ denotes the multiplicative depth of the circuit. On the other hand, the perfect MPC protocols achieving best known round complexities such as $\mathcal{O}(n\mathcal{D})$ [5] and $\mathcal{O}(n + \mathcal{D})$ [1] are far from being truly communication efficient. In the sequel, we present a table which gives an overview of the communication complexities and round complexities of perfect and unconditional MPC protocols. The complexity figures are provided assuming that *physical broadcast* channel is not available and hence broadcast is simulated by some protocol (as mentioned earlier). The communication complexities are given in terms of bits where κ represents the bit length of a field element in the case of perfect MPC and error parameter in the case of unconditional MPC. For simplicity, we assume that the computed function takes n inputs (one from each player) and gives n outputs. c_M and $\mathcal{D}$ denote the number of multiplication gates and multiplicative depth of the circuit, respectively.

Reference	Type?	Resilience	Brodcast Protocol	Communication Complexity	Round Complexity
[11]	Unconditional	$t < n/2$	[13]	$\mathcal{O}((c_M n^6 + n^7)\kappa)$	$\mathcal{O}(n^2\mathcal{D})$
[17]	Perfect	$t < n/3$	[6, 7]	$\mathcal{O}((c_M n^3 + n^4)\kappa)$	$\mathcal{O}(n^2 + \mathcal{D})$
[3]	Unconditional	$t < n/2$	[13]	$\mathcal{O}((c_M n^2 + n^7)\kappa)$	$\mathcal{O}(n^3 + \mathcal{D})$
[12]	Unconditional	$t < n/3$	[6, 7]	$\mathcal{O}((c_M n + \mathcal{D}n^2 + n^4)\kappa)$	$\mathcal{O}(n^2 + \mathcal{D})$
[4]	Perfect	$t < n/3$	[6, 7]	$\mathcal{O}((c_M n + \mathcal{D}n^2 + n^3)\kappa)$	$\mathcal{O}(n^2 + \mathcal{D})$

If the practical applicability of multiparty protocols are of primary focus, then it is always desirable not to sacrifice one parameter for the other. So it is very essential to design protocol which balances both the parameters appropriately. Motivated by this, in this work we design an UMPC protocol which achieves efficiency in both the parameters simultaneously.

Our Network Model: We denote the set of $n = 2t + 1$ players (parties) involved in the secure computation by $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ where player P_i possesses c_i input values. We assume that all the n players are connected with each other by pairwise secure channels, as assumed in generic UMPC protocols [3, 11, 22]. Moreover, the system is synchronous and the protocols proceed in rounds, where in each round a player performs some computations, sends (broadcasts) values to its neighbors (everybody), receives values from neighbors and may again perform some more computation, in that order. The function to be computed is specified as an arithmetic circuit over a finite field $\mathbb{F}$ with input, addition, multiplication, random and output gates. We denote the number of gates of each type by c_I, c_A, c_M, c_R and c_O, respectively. Note that $c_I = \sum_{i=1}^{n} c_i$.

We model the distrust in the system by a centralized adversary $\mathcal{A}_t$, who has *unbounded computing* power and can actively control at most t players during the protocol execution, where $t < \frac{n}{2}$. To actively control a player means to take full control over it and making it behave arbitrarily. The adversary is *adaptive* [11] and hence can corrupt players dynamically during the protocol execution. Moreover, the choice of the adversary to corrupt a player may depend upon the data seen so far from the corrupted players. Moreover, the adversary is a *rushing adversary* [15], who in a particular round, first collects all the messages addressed to the corrupted players and exploits this information to decide on what the corrupted players send during the same round. If a player comes under the control of $\mathcal{A}_t$, then it remains so throughout the protocol. A player which is not under the control of $\mathcal{A}_t$ is called *honest* or *uncorrupted*. We define two sets $\mathcal{C}$ and $\mathcal{P}'$ where at any point of time $\mathcal{C}$ denotes the set of corrupted players identified so far and $\mathcal{P}' = \mathcal{P} \setminus \mathcal{C}$. Initially, $\mathcal{P}' = \mathcal{P}$ and $\mathcal{C} = \emptyset$. As the protocol proceeds, some players will be detected as corrupted and will be added to $\mathcal{C}$ and removed from $\mathcal{P}'$. We denote the number of players in $\mathcal{P}'$ by n' which is initially equal to n. The number of players which can be still corrupted from $\mathcal{P}'$ is denoted by t' where $t' = t - |\mathcal{C}|$. Note that n' will always maintain the following: $n' \geq t + 1 + t' \geq 2t' + 1$ since $t \geq t'$. Also at any point of time $\mathcal{P} = \mathcal{P}' \cup \mathcal{C}$.

Our protocol provides unconditional security i.e. information theoretic security with a negligible error probability of $2^{-\mathcal{O}(\kappa)}$ for some security parameter κ. To achieve this error probability, all our computation are done over a finite field $\mathbb{F} = GF(2^{\kappa})$. Thus each field element can be represented by κ bits. Notice that, we also assume that n is polynomial in κ. For the ease of exposition, we always assume that the messages sent through the channels are from the specified domain. Thus if a player receives a message which is not from the specified domain (or no message at all), he replaces it with some pre-defined default message from the specified domain.

Our Contribution: In this paper, we propose a new UMPC protocol which communicates $\mathcal{O}(n^4)$ field elements per multiplication and requires $\mathcal{O}(n\log(n) + \mathcal{D})$ rounds over a point-to-point network (in the absence of physical broadcast channel) with $n = 2t+1$ players. This result is to be compared with the UMPC protocol of [11] which provides so far best known round complexity of $\mathcal{O}(n^2 D)$ (and communicates $\mathcal{O}(n^6)$ field elements per multiplication) and with UMPC protocol of [3] which achieves the best known communication complexity of $\mathcal{O}(n^2)$ field elements per multiplication (but consumes $\mathcal{O}(n^3 + \mathcal{D})$ rounds). Hence our protocol is the most round efficient protocol so far and ranks second according to communication complexity. We introduce a new technique called *Rapid Player Elimination* (RPE) which is used in the preprocessing stage of our proposed UMPC protocol. Loosely speaking, RPE works as follows: The preprocessing stage of our UMPC protocol may fail several times due to the (mis)behavior of certain number of corrupted players whose corruptions are identified. RPE creates a win-win situation, where the adversary must reveal the identities of 2^i new corrupted players at the i^{th} step. Otherwise, the preprocessing stage will not fail. Thus RPE ensures that preprocessing stage may fail at most $\lceil \log(t) \rceil$ times.

2 Unconditionally Secure MPC Protocol with $n = 2t + 1$

In this section, we present an UMPC protocol with $n = 2t + 1$. Prior to that we present a number of sub-protocols each solving a specific task. Some of the sub-protocols are based on few existing techniques while some are proposed by us for the first time. We describe all our sub-protocols in the following settings: $\mathcal{P}'$ denotes the set of players involved in the execution of the sub-protocols where $|\mathcal{P}'| = n'$, the number of corrupted players present in $\mathcal{P}'$ is t' and $n' = t+1+t'$. During the execution of the sub-protocols, some more corrupted players may be detected as faulty and wil be removed from $\mathcal{P}'$. Accordingly n' and t' will change (without affecting the equality $n' = t+1+t'$). Thus it is clear that at any stage $\mathcal{P}'$ will always contain all the $t+1$ honest players. For simplicity, we assume that $\mathcal{P}'$ always contains the first n' players $(P_1, \ldots, P_{n'})$ from the set $\mathcal{P}$. For convenience, we provide analysis of the communication and round complexities of our sub-protocols and protocols assuming that physical broadcast channel is available in the system. At the end we give the complete communication and round complexity figure for our general multiparty computation protocol, assuming that the broadcast has to be simulated by protocol of [13]. Notice that a physical broadcast channel enables all the players from $\mathcal{P}$ to receive the broadcasted message. Since our sub-protocols are executed among the players in $\mathcal{P}'$ and $\mathcal{P}' \subseteq \mathcal{P}$, a broadcast step in our sub-protocol should allow only the players of $\mathcal{P}'$ to get the broadcasted information. But notice that all the broadcast steps will be finally replaced by protocols (say from [13]) where only players from $\mathcal{P}'$ are allowed to participate. *Thus without loss of generality, we may interpret all the broadcasts steps in our sub-protocols as the broadcast to the restricted set $\mathcal{P}'$.*

2.1 Information Checking

Information Checking (IC) and IC Signatures [11, 22]: IC is an information theoretically secure method for authenticating data and is used to generate IC signatures. When a player $INT \in \mathcal{P}'$ receives an IC signature from a dealer $\mathbf{D} \in \mathcal{P}'$ on some secret value(s) S, then INT can later produce the signature and have the players in $\mathcal{P}'$ verify that it is in fact a valid signature of $\mathbf{D}$ on S. An IC scheme consists of a sequence of three protocols:

1. **Distr($\mathbf{D}, INT, \mathcal{P}', S$)** is initiated by the dealer $\mathbf{D}$, who hands secret $S = [s^{(1)} \ \ldots \ s^{(\ell)}] \in \mathbb{F}^{\ell}$, where $\ell \geq 1$ to intermediary INT. In addition, $\mathbf{D}$ hands some **authentication information** to INT and **verification information** to individual players in $\mathcal{P}'$, also called as receivers.

2. **AuthVal($\mathbf{D}, INT, \mathcal{P}', S$)** is initiated by INT to ensure that in protocol **RevealVal**, secret S held by INT will be accepted by all the (honest) players (receivers) in $\mathcal{P}'$.

3. **RevealVal($\mathbf{D}, INT, \mathcal{P}', S$)** is carried out by INT and the receivers in $\mathcal{P}'$, where INT produces S, along with **authentication information** and the individual receivers in $\mathcal{P}'$ produce **verification information**. Depending upon the values produced by INT and the receivers, either S is accepted or rejected by all the players/receivers.

The **authentication information**, along with S, which is held by INT at the end of **AuthVal** is called $\mathbf{D}$'s IC signature on S, obtained by INT. The IC signature must satisfy the following properties:

1. If $\mathbf{D}$ and INT are uncorrupted, then S will be accepted in **RevealVal**.

2. If INT is uncorrupted, then at the end of **AuthVal**, INT possesses secret(s), say S', which will be accepted in **RevealVal**, except with probability at most $2^{-\mathcal{O}(\kappa)}$.

3. If $\mathbf{D}$ is uncorrupted, then during **RevealVal**, with probability at least $1 - 2^{-\mathcal{O}(\kappa)}$, every $S' \neq S$ produced by a corrupted INT will be rejected.

4. If $\mathbf{D}$ and INT are uncorrupted, then at the end of **AuthVal**, S is information theoretically secure from $\mathcal{A}_t$.

We now describe an IC protocol which is a slight modification of the IC protocol described in [11].

The protocol allows **D** to sign on a single field element $s \in \mathbb{F}$ (i.e. $\ell = 1$; $S = s$). In the protocol, **Distr** takes single round (**Round 1**), **AuthVal** takes threes rounds (**Round 2, 3** and **Round 4**), while **RevealVal** takes two rounds (**Round 1** and **Round 2**). The protocol is given in Table 1. Before describing the protocol, we recall the following definition from [11].

Definition 1 (1_α-consistent [11]) *A vector $(x, y, z) \in \mathbb{F}^3$ is 1_α-consistent if there exists a degree one polynomial w over $\mathbb{F}$ such that $w(0) = x$, $w(1) = y$ and $w(\alpha) = z$.*

Lemma 1 *Protocol* **IC** *correctly generates IC signatures on a single secret (i.e. $\ell = 1$) by communicating $\mathcal{O}(n\kappa)$ bits and broadcasting $\mathcal{O}(n\kappa)$ bits. The protocol satisfies all the properties of IC signature with an error probability of at most $2^{-\mathcal{O}(\kappa)}$.*

PROOF: The proof is similar to the proof of the properties of generalized IC protocol of [11] (see Lemma 1, Page 318-319). □

Protocol IC(D, $INT, \mathcal{P}', s$)

Distr(D, $INT, \mathcal{P}', s$) **Round 1:** Corresponding to each receiver $P_i \in \mathcal{P}'$, **D** chooses a random value $\alpha_i \in \mathbb{F} - \{0, 1\}$ and additional random values $y_i, z_i \in \mathbb{F}$, such that the three tuple (s, y_i, z_i) is 1_{α_i}-consistent. In addition, corresponding to each $P_i \in \mathcal{P}'$, **D** chooses another random 1_{α_i}-consistent vector (s'_i, y'_i, z'_i). **D** sends s, y_i, s'_i and y'_i to INT and α_i, z_i, z'_i to the receiver P_i. The n' tuple $[y_1\ y_2\ \ldots\ y_{n'}]$ held by INT is called **authentication information**. The two n' tuples $[y'_1\ y'_2\ \ldots\ y'_{n'}]$ and $[s'_1\ s'_2\ \ldots, s'_{n'}]$ held by INT are called as **auxiliary information**, where as the tuple (α_i, z_i) held by receiver P_i is called **verification information**.

AuthVal(D, $INT, \mathcal{P}', s$): **Round 2:** INT randomly selects n' random elements $d_i, 1 \leq i \leq n'$ from $\mathbb{F} - \{0\}$ and broadcasts the tuples $(d_i, s'_i + d_i s, y'_i + d_i y_i)$.

Round 3: In response to INT's broadcast in **Round 2**, **D** checks the correctness of the broadcasted information and also checks whether $(s'_i + d_i s, y'_i + d_i y_i, z'_i + d_i z_i)$ is 1_{α_i}-consistent for $1 \leq i \leq n'$. **D** broadcasts s, along with the n' tuple $[y_1\ y_2\ \ldots\ y_{n'}]$ if he finds any inconsistency. Each $P_i \in \mathcal{P}'$ accordingly adjusts his **verification information** (α_i, z_i), such that (s, y_i, z_i) is 1_{α_i}-consistent and the protocol **ends** here.
Parallely, $P_i \in \mathcal{P}'$ checks if $(s'_i + d_i s, y'_i + d_i y_i, z'_i + d_i z_i)$ is 1_{α_i}-consistent and broadcasts "Accept" or "Reject", depending upon whether $(s'_i + d_i s, y'_i + d_i y_i, z'_i + d_i z_i)$ is 1_{α_i}-consistent or not.

Now the value s and the n' tuple $[y_1\ y_2\ \ldots\ y_{n'}]$ possessed by INT is called the **IC-Signature** on s given by **D** to INT, which is denoted by $ICSig_s(\mathbf{D}, INT)$.

Round 4: If **D** has not broadcasted s, along with the tuple $[y_1\ y_2\ \ldots\ y_{n'}]$ in the previous round, then **D** broadcasts (α_i, z_i) corresponding to all receivers $P_i \in \mathcal{P}'$, whose response was "Reject" in the previous round. Accordingly INT will adjust his y_i so that (s, y_i, z_i) becomes 1_{α_i}-consistent.

RevealVal(D, $INT, \mathcal{P}', s$): **Round 1:** INT broadcasts s and $[y_1\ y_2\ \ldots\ y_{n'}]$; **Round 2:** Each $P_i \in \mathcal{P}'$ broadcasts (α_i, z_i). If there exists at least $t + 1$ distinct j's such that (s, y_j, z_j) is 1_{α_j}-consistent then **D**'s signature on s is "valid". Otherwise the signature is "invalid".

Table 1: An IC Protocol to Sign on a Single Field Element

We now present an IC protocol, called **EfficientIC**, which allows **D** to sign on an ℓ length secret $S \in \mathbb{F}^\ell$ simultaneously, with $\ell \geq 1$, by communicating $\mathcal{O}((\ell + n)\kappa)$ bits and broadcasting $\mathcal{O}((\ell + n)\kappa)$ bits, where $n' = t + 1 + t'$. Let $S = (s^{(1)}, \ldots, s^{(\ell)}) \in \mathbb{F}^\ell$. The idea of this protocol is taken from [20]. The protocol **EfficientIC** is given in Table 2.

Lemma 2 *Protocol* **EfficientIC** *correctly generates IC signature on ℓ field elements (each of size κ bits) at once by communicating $\mathcal{O}((\ell + n)\kappa)$ bits and broadcasting $\mathcal{O}((\ell + n)\kappa)$ bits. The protocol satisfies the properties of IC signature with an error probability of at most $2^{-\mathcal{O}(\kappa)}$.*

PROOF: The proof is similar to the proof of the IC protocol given in [20] and hence is omitted. □

Linearity of Protocol IC and EfficientIC: Protocol **IC** and **EfficientIC** satisfies the *linearity* property as specified by the following lemmas:

Lemma 3 (Linearity of Protocol IC [11]) *Let $ICSig_{s_1}(\mathbf{D}, INT)$ and $ICSig_{s_2}(\mathbf{D}, INT)$ denotes the IC signature on two different secrets s_1 and s_2, generated by* **D** *using protocol* **IC**.

Moreover, let $\mathbf{D}$ *uses the same set of* $\alpha_1, \alpha_2, \ldots, \alpha_{n'}$ *to give his IC signature on* s_1 *and* s_2*. Let* r_1, r_2 *be two random public constants from* $\mathbb{F}$*. Then* INT *can generate* $ICSig_{(r_1s_1+r_2s_2)}(\mathbf{D}, INT)$ *from* $ICSig_{s_1}(\mathbf{D}, INT)$ *and* $ICSig_{s_2}(\mathbf{D}, INT)$ *without any further communication. Similarly, the receivers in* $\mathcal{P}'$ *can obtain the verification information corresponding to* $ICSig_{(r_1s_1+r_2s_2)}(\mathbf{D}, INT)$ *from the verification information corresponding to* $ICSig_{s_1}(\mathbf{D}, INT)$ *and* $ICSig_{s_2}(\mathbf{D}, INT)$ *without doing any communication.*

Extending the above lemma for an arbitrary length secret, we can state the following lemma:

Lemma 4 (Linearity of Protocol EfficientIC) *The IC signature generated by* **EfficientIC** *satisfies* **linearity** *property. In particular,* INT *can compute* $ICSig_{((r_1s^{(1,1)}+r_2s^{(2,1)}),\ldots,(r_1s^{(1,\ell)}+r_2s^{(2,\ell)}))}(\mathbf{D}, INT)$ *from* $ICSig_{(s^{(1,1)},s^{(1,2)}\ldots,s^{(1,\ell)})}(\mathbf{D}, INT)$ *and* $ICSig_{(s^{(2,1)},s^{(2,2)}\ldots,s^{(2,\ell)})}(\mathbf{D}, INT)$ *and receivers can compute verification information corresponding to* $ICSig_{((r_1s^{(1,1)}+r_2s^{(2,1)}),\ldots,(r_1s^{(1,\ell)}+r_2s^{(2,\ell)}))}(\mathbf{D}, INT)$*.*

EfficientIC$(\mathbf{D}, INT, \mathcal{P}', \ell, s^{(1)}, \ldots, s^{(\ell)})$

EfficientDistr$(\mathbf{D}, INT, \mathcal{P}', \ell, s^{(1)}, \ldots, s^{(\ell)})$**: Round 1**: $\mathbf{D}$ selects a random $\ell + t' - 1$ degree polynomial $F(x)$ over $\mathbb{F}$, whose lower order ℓ coefficients are $s^{(1)}, \ldots, s^{(\ell)}$. In addition, $\mathbf{D}$ selects another random $\ell + t' - 1$ degree polynomial $R(x)$, over $\mathbb{F}$, which is independent of $F(x)$. $\mathbf{D}$ selects n' distinct random elements $\alpha_1, \alpha_2, \ldots, \alpha_{n'}$ from $\mathbb{F}$ such that each $\alpha_i \in \mathbb{F} - \{0, 1, \ldots, n' - 1\}$. $\mathbf{D}$ privately gives $F(x)$ and $R(x)$ to INT. To receiver $P_i \in \mathcal{P}'$, $\mathbf{D}$ privately gives α_i, v_i and r_i, where $v_i = F(\alpha_i)$ and $r_i = R(\alpha_i)$. The polynomial $R(x)$ is called **authentication information**, while for $1 \leq i \leq n'$, the values α_i, v_i and r_i are called **verification information**.

EfficientAuthVal$(\mathbf{D}, INT, \mathcal{P}', \ell, s^{(1)}, \ldots, s^{(\ell)})$**: Round 2**: INT chooses a random $d \in \mathbb{F} \setminus \{0\}$ and broadcasts $d, B(x) = dF(x) + R(x)$.

Round 3: For $1 \leq j \leq n'$, $\mathbf{D}$ checks $dv_j + r_j \stackrel{?}{=} B(\alpha_j)$. If $\mathbf{D}$ finds any inconsistency, he broadcasts $F(x)$. Parallely, receiver P_i broadcasts "Accept" or "Reject", depending upon whether $dv_i + r_i = B(\alpha_i)$ or not.

Local Computation (by each player): IF $F(x)$ is broadcasted in **Round 3** then accept the lower order ℓ coefficients of $F(x)$ as $\mathbf{D}$'s secret and terminate. ELSE construct an n' length bit vector V^{Sh}, where the $j^{th}, 1 \leq j \leq n'$ bit is 1(0), if $P_j \in \mathcal{P}'$ has broadcasted "Accept" ("Reject") during **Round 3**. The vector V^{Sh} is public, as it is constructed using broadcasted information. If V^{Sh} does not contain $n' - t'$ 1's, then $\mathbf{D}$ fails to give any signature to INT and IC protocol terminates here.

If $F(x)$ is not broadcasted during **Round 3**, then $(F(x), R(x))$ is called $\mathbf{D}$'s IC signature on $S = (s^{(1)}, \ldots, s^{(\ell)})$ given to INT, which is denoted by $ICSig_{(s^{(1)},\ldots,s^{(\ell)})}(\mathbf{D}, INT)$.

EfficientRevealVal$(\mathbf{D}, INT, \mathcal{P}', \ell, s^{(1)}, \ldots, s^{(\ell)})$: (a) **Round 1**: INT broadcasts $F(x), R(x)$; (b) **Round 2**: P_i broadcasts α_i, v_i and r_i.

Local Computation (by each player): For the polynomial $F(x)$ broadcasted by INT, construct an n' length vector $V^{Rec}_{F(x)}$ whose j^{th} bit contains 1 if $v_j = F(\alpha_j)$, else 0. Similarly, construct the vector $V^{Rec}_{R(x)}$ corresponding to $R(x)$. Finally compute $V^{Rec}_{FR} = V^{Rec}_{F(x)} \otimes V^{Rec}_{R(x)}$, where $\otimes$ denotes bit wise AND. Since broadcasted information is public, each player (honest) will compute the same vectors $V^{Rec}_{F(x)}$ and $V^{Rec}_{R(x)}$ and hence V^{Rec}_{FR}. If V^{Rec}_{FR} and V^{Sh} matches at least at $t + 1$ locations (irrespective of bit value at these locations), then accept the lower order ℓ coefficients of $F(x)$ as $S = (s^{(1)}, \ldots, s^{(\ell)})$. In this case, we say that $\mathbf{D}$'s signature on S is correct. Else reject $F(x)$ broadcasted by INT and we say that INT has failed to produce $\mathbf{D}$'s signature.

Table 2: An IC Protocol to Sign ℓ Length Secret where $n' = t + 1 + t'$

2.2 Unconditional Verifiable Secret Sharing and Reconstruction

Definition 2 t'**-1D-Sharing:** *We say that a value* s *is correctly* t'*-1D-shared among the players in* $\mathcal{P}'$ *if every honest player* $P_i \in \mathcal{P}'$ *is holding a share* s_i *of* s*, such that there exists a degree* t' *polynomial* $f(x)$ *over* $\mathbb{F}$ *with* $f(0) = s$ *and* $f(j) = s_j$ *for every* $P_j \in \mathcal{P}'$*. The vector* $(s_1, s_2, \ldots, s_{n'})$ *of shares is called a* t'*-sharing of* s *and is denoted by* $[s]_{t'}$*. We may skip the subscript* t' *when it is clear from the context. A set of shares (possibly incomplete) is called* t'*-consistent if these shares lie on a* t' *degree polynomial.*

Definition 3 **t'-2D-sharing:** *[3] We say that a value s is correctly t'-2D-shared among the players in $\mathcal{P}'$ if there exists t' degree polynomials $f, f^1, f^2 \ldots, f^{n'}$ with $f(0) = s$ and for $i = 1, \ldots, n'$, $f^i(0) = f(i)$. Moreover, every player $P_i \in \mathcal{P}'$ holds a share $s_i = f(i)$ of s, the polynomial $f^i(x)$ for sharing s_i and a share-share $s_{ji} = f^j(i)$ of the share s_j of every player $P_j \in \mathcal{P}'$. We denote t'-2D-sharing of s as $[[s]]_{t'}$.*

Definition 4 **t'-$2D^{(+)}$-sharing:** *We say that a value s is correctly t'-$2D^{(+)}$-shared among the players in $\mathcal{P}'$ if there exists t' degree polynomials $f, f^1, f^2 \ldots, f^{n'}$ with $f(0) = s$ and for $i = 1, \ldots, n'$, $f^i(0) = f(i)$. Moreover, every player $P_i \in \mathcal{P}'$ holds a share $s_i = f(i)$ of s, the polynomial $f^i(x)$ for sharing s_i and P_j's IC Signature on share-share $s_{ji} = f^j(i)$ of P_j's share s_j, i.e. $ICSig_{s_{ji}}(P_j, P_i)$ for every player $P_j \in \mathcal{P}'$. We denote the t'-$2D^{(+)}$-sharing of s as $\langle\langle s \rangle\rangle_{t'}$. Note that in [3], the authors called this sharing as $2D^*$-sharing.*

Definition 5 **t'-$2D^{(+,\ell)}$-sharing:** *We say that a set of values $s^{(1)}, \ldots, s^{(\ell)}$ are correctly t'-$2D^{(+,\ell)}$-shared among the players in $\mathcal{P}'$ if every secret $s^{(l)}$ is individually t'-$2D^{(+)}$-shared. But now instead of P_i holding separate IC-signatures for each of the share-shares $s_{ji}^{(l)}$ for $l = 1 \ldots, \ell$ from P_j (i.e. $ICSig_{s_{ji}^{(l)}}(P_j, P_i)$ for $l = 1 \ldots, \ell$), a single IC-signature on $s_{ji}^{(1)}, \ldots, s_{ji}^{(\ell)}$ is given by P_j to P_i (i.e. $ICSig_{(s_{ji}^{(1)}, \ldots, s_{ji}^{(\ell)})}(P_j, P_i)$). We denote the t'-$2D^{(+,\ell)}$-sharing of ℓ values as $\langle\langle s^1, \ldots, s^\ell \rangle\rangle_{t'}$.*

If a secret s is t'-1D-shared/t'-2D-shared/t'-$2D^{(+)}$-shared by a dealer $\mathbf{D} \in \mathcal{P}'$ (any player from $\mathcal{P}'$ may perform the role of a dealer), then we denote the sharing by $[s]_{t'}^{\mathbf{D}}/[[s]]_{t'}^{\mathbf{D}}/\langle\langle s \rangle\rangle_{t'}^{\mathbf{D}}$. Similarly if a set of ℓ secrets $s^{(1)}, \ldots, s^{(\ell)}$ are t'-$2D^{(+,\ell)}$-shared by player $\mathbf{D}$, we denote it by $\langle\langle s^1, \ldots, s^\ell \rangle\rangle_{t'}^{\mathbf{D}}$. Notice that when a secret s is t'-$2D^{(+)}$-shared, then s is also t'-1D-Shared and t'-2D-shared by default. Hence t'-$2D^{(+)}$-sharing is the strongest sharings among t'-1D-sharing, t'-2D-sharing and t'-$2D^{(+)}$-sharing. In some sense, t'-$2D^{(+,\ell)}$-sharing is the extension of t'-$2D^{(+)}$-sharing for ℓ secrets. If a dealer $\mathbf{D} \in \mathcal{P}'$ is honest, then he will always correctly t'-1D-share/t'-2D-share/t'-$2D^{(+)}$-share a secret s. Among these three types of sharings, t'-$2D^{(+)}$-sharing of a secret s allows efficient reconstruction of the secret with n' players. However, a corrupted $\mathbf{D}$ may perform sharing in an incorrect way. To achieve parallelism, in the sequel, we describe a protocol called $2D^{(+,\ell)}$**Share** which allows a dealer $\mathbf{D} \in \mathcal{P}'$ to *verifiably* t'-$2D^{(+,\ell)}$-share $\ell \geq 1$ length secret $[s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}]$. Verifiably t'-$2D^{(+,\ell)}$-sharing ensures correct t'-$2D^{(+,\ell)}$-sharing even for a corrupted $\mathbf{D}$. The protocol $2D^{(+,\ell)}$**Share** is given in Table 3. The goal of the protocol is as follows: (a) If $\mathbf{D}$ is honest then he correctly generates t'-$2D^{(+,\ell)}$-sharing of the secret $[s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}]$, such that all the honest players publicly verify that $\mathbf{D}$ has correctly generated the sharing. Also when $\mathbf{D}$ is honest, then the secret will be information theoretically secure from the adversary $\mathcal{A}_t$. (b) If $\mathbf{D}$ is corrupted and has not generated correct t'-$2D^{(+,\ell)}$-sharing, then with very high probability, everybody will detect it and protocol will terminate. The idea of the protocol is taken from [11], but instead of using the IC protocol of [11], we employ the **EfficientIC** protocol proposed in this paper, which provides us with higher efficiency.

Lemma 5 *In protocol $2D^{(+,\ell)}$**Share**, $\mathbf{D}$ generates correct t'-$2D^{(+,\ell)}$-sharing of ℓ field elements (each of size κ bits), with overwhelming probability. $2D^{(+,\ell)}$**Share** is a ten round protocol which communicates $\mathcal{O}((\ell n^2 + n^3)\kappa)$ bits and broadcasts $\mathcal{O}((\ell n^2 + n^3)\kappa)$ bits.*

Proof (Sketch): For every $l \in \{1, \ldots, \ell\}$, secret $s^{(l)}$ is t'-$1D$-shared by t' degree polynomial $g_0^{(l)}(y)$ i.e $g_0^{(l)}(0) = s^{(l)}$. Also t' degree polynomials $f_1^{(l)}(x), \ldots, f_{n'}^{(l)}(x)$ are such that $g_0^{(l)}(i) = f_i^{(l)}(0)$ for $i = 1, \ldots, n'$. Hence, every secret $s^{(l)}$ is t'-$2D$-shared by t' degree polynomials $g_0^{(l)}(y)$ and $f_1^{(l)}(x), \ldots, f_{n'}^{(l)}(x)$. Hence player P_i (implicitly) holds i^{th} share of $g_0^{(l)}(y)$ and polynomial $f_i^{(l)}(x)$ for all $l \in \{1, \ldots, \ell\}$. In addition to that, player P_i possesses correct $ICSig_{(f_j^{(1)}(i), f_j^{(2)}(i), \ldots, f_j^{(\ell)}(i))}(P_j, P_i)$ for every player $P_j \in \mathcal{P}'$, with very high probability (from the properties of our **EfficientIC** protocol). Hence it is clear that $\mathbf{D}$ has generated correct t'-$2D^{(+,\ell)}$-sharing of ℓ length secret with very probability. Round complexity of $2D^{(+,\ell)}$**Share** is easy to verify. Since in sum at most $4(n')^2$ instances of **EfficientDistr** and **EfficientAuthVal** of Protocol **EfficientIC** are executed, Protocol $2D^{(+,\ell)}$**Share** communicates and broadcasts $\mathcal{O}((\ell n^2 + n^3)\kappa)$ bits. $\square$

Remark: When an ℓ length secret $[s^{(1)}, \ldots, s^{(\ell)}]$ is t'-$2D^{(+,\ell)}$-shared, then implicitly the individual secrets are t'-$1D$-shared by polynomials $g_0^{(1)}(y), \ldots, g_0^{(\ell)}(y)$. Also note that given $\langle\langle a^{(1)}, \ldots, a^{(\ell)}\rangle\rangle_{t'}$ and $\langle\langle b^{(1)}, \ldots, b^{(\ell)}\rangle\rangle_{t'}$, the players in $\mathcal{P}'$ can compute $\langle\langle c^{(1)}, \ldots, c^{(\ell)}\rangle\rangle_{t'}$ where for $l = 1, \ldots, \ell$, $c^{(l)} = \mathcal{F}(a^{(l)}, b^{(l)})$ and $\mathcal{F}$ denotes any linear combination. This is due to the linearity property of our **EfficientIC** protocol presented in subsection 2.1.

$\langle\langle s^{(1)}, \ldots, s^{(\ell)}\rangle\rangle_{t'}^{\mathbf{D}} = 2D^{(+,\ell)}\mathbf{Share}(\mathbf{D}, \mathcal{P}', t', \ell, s^{(1)}, \ldots, s^{(\ell)})$

1. For every $l = 1, \ldots, \ell$, **D** picks a random bivariate polynomial $H^{(l)}(x,y)$ of degree t' in both the variables, with $H^{(l)}(0,0) = s^{(l)}$. Let $f_i^{(l)}(x) = H^{(l)}(x,i)$ and $g_i^{(l)}(y) = H^{(l)}(i,y)$. Now **D** wants to hand over n' values on $f_i^{(l)}(x)$ and $g_i^{(l)}(y)$ for $l = 1, \ldots, \ell$ to P_i with his IC signature on them. For that **D** executes **EfficientDistr** and **EfficientAuthVal** of **EfficientIC**$(\mathbf{D}, P_i, \mathcal{P}', \ell, f_i^{(1)}(j), f_i^{(2)}(j), \ldots, f_i^{(\ell)}(j))$ for for all $j \in \{1, \ldots, n'\}$. Similarly **D** executes **EfficientDistr** and **EfficientAuthVal** of **EfficientIC**$(\mathbf{D}, P_i, \mathcal{P}', \ell, g_i^{(1)}(j), g_i^{(2)}(j), \ldots, g_i^{(\ell)}(j))$.

2. For $l = 1, \ldots, \ell$ player P_i checks whether the two sets $f_i^{(l)}(1), \ldots, f_i^{(l)}(n')$ and $g_i^{(l)}(1), \ldots, g_i^{(l)}(n')$ are t'-consistent. If the values are not t'-consistent, for some $l \in \{1, \ldots, \ell\}$ then P_i along with all players in $\mathcal{P}'$ invoke **EfficientRevealVal**$(\mathbf{D}, P_i, \mathcal{P}', \ell, f_i^{(1)}(j), f_i^{(2)}(j), \ldots, f_i^{(\ell)}(j))$ and **EfficientRevealVal**$(\mathbf{D}, P_i, \mathcal{P}', \ell, g_i^{(1)}(j), g_i^{(2)}(j), \ldots, g_i^{(\ell)}(j))$ for all $j \in \{1, \ldots, n'\}$. If the signatures produced by P_i are valid and for some $l \in \{1, \ldots, \ell\}$ either one of the two sets $f_i^{(l)}(1), \ldots, f_i^{(l)}(n')$ or $g_i^{(l)}(1), \ldots, g_i^{(l)}(n')$ is not t'-consistent, then the protocol terminates here without generating the desired output.

3. For every pair of players P_i and P_j from $\mathcal{P}'$ the following will be executed:
   (a) Ideally for P_i and P_j the following should hold: $f_i^{(l)}(j) = g_j^{(l)}(i)$ and $g_i^{(l)}(j) = f_j^{(l)}(i)$ for $l = 1, \ldots, \ell$. P_i as a dealer executes **EfficientDistr** and **EfficientAuthVal** of **EfficientIC**$(P_i, P_j, \mathcal{P}', \ell, f_i^{(1)}(j), \ldots, f_i^{(\ell)}(j))$ to give his IC signature on $f_i^{(1)}(j), \ldots, f_i^{(\ell)}(j)$ to P_j. Upon receiving the signature, P_j checks whether $f_i^{(l)}(j) \stackrel{?}{=} g_j^{(l)}(i)$ for $l = 1, \ldots, \ell$. If there is an inconsistency then P_j along with all players in $\mathcal{P}'$ invoke **EfficientRevealVal**$(\mathbf{D}, P_j, \mathcal{P}', \ell, g_j^{(1)}(i), g_j^{(2)}(i), \ldots, g_j^{(\ell)}(i))$.
   (b) If P_j fails to produce valid signature in the previous step, then all the players from $\mathcal{P}'$ ignore the IC signatures received from P_j in previous step. Otherwise, if P_j is able to produce valid signature then $g_j^{(1)}(i), g_j^{(2)}(i), \ldots, g_j^{(\ell)}(i)$ become public. Using the public values P_i checks whether $f_i^{(l)}(j) \stackrel{?}{=} g_j^{(l)}(i)$ for $l = 1, \ldots, \ell$. If he finds any inconsistency, then P_i along with all players in $\mathcal{P}'$ invoke **EfficientRevealVal**$(\mathbf{D}, P_i, \mathcal{P}', \ell, f_i^{(1)}(j), f_i^{(2)}(j), \ldots, f_i^{(\ell)}(j))$.
   (c) If P_i fails to produce valid signature in the previous step, then all the players from $\mathcal{P}'$ ignore the IC signatures received from P_i in step 3(a). Otherwise, if P_i is able to produce valid signature, then all the values $f_i^{(1)}(j), f_i^{(2)}(j), \ldots, f_i^{(\ell)}(j)$ become public. Every player then verifies whether $f_i^{(l)}(j) \stackrel{?}{=} g_j^{(l)}(i)$ for $l = 1, \ldots, \ell$. If $f_i^{(l)}(j) \neq g_j^{(l)}(i)$ for some $l \in \{1, \ldots, \ell\}$ then the protocol terminates here without generating the desired output.

Table 3: A Ten Round Protocol for Verifiably t'-$2D^{(+,\ell)}$-share ℓ Length Secret.

Conversion From a t'-$2D^{(+,\ell)}$-sharing to ℓ t'-$2D^+$-sharings: Given t'-$2D^{(+,\ell)}$-sharing of ℓ secrets, we present a protocol **Convert**$2D^{(+,\ell)}$**to**$2D^+$ which converts the t'-$2D^{(+,\ell)}$-sharing of the ℓ secrets to ℓ t'-$2D^+$-sharings of the individual ℓ secrets. Thus given $\langle\langle s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}\rangle\rangle_{t'}$, **Convert**$2D^{(+,\ell)}$**to**$2D^+$ produces $\langle\langle s^{(l)}\rangle\rangle_{t'}$ for $l = 1, \ldots, \ell$. Note that $\langle\langle s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}\rangle\rangle_{t'}$ could have been generated directly by Protocol $2D^{(+,\ell)}$**Share** or it could be linear combination of a number of t'-$2D^{(+,\ell)}$-sharings generated by different instances of Protocol $2D^{(+,\ell)}$**Share**.

Lemma 6 *Protocol* **Convert**$2D^{(+,\ell)}$**to**$2D^+$ *takes five rounds and communicates* $\mathcal{O}((\ell n^3 + n^4)\kappa)$ *bits and broadcasts* $\mathcal{O}((\ell n^3 + n^4)\kappa)$ *bits.*

Reconstruction of a t'-$2D^+$-shared secret: Let $\langle\langle s\rangle\rangle_{t'}$ be a t'-$2D^+$-sharing, which is shared using the polynomials $H(x,y), f_i(x), g_i(y), 1 \leq i \leq n'$ among the players in $\mathcal{P}'$. We may assume $\langle\langle s\rangle\rangle_{t'}$ as one of the ℓ outcomes of Protocol **Convert**$2D^{(+,\ell)}$**to**$2D^+$. We now present a protocol $2D^+$**Recons** which allows the (honest) players to correctly recover s with very high probability.

$(\langle\langle s^{(1)}\rangle\rangle_{t'}, \ldots, \langle\langle s^{(\ell)}\rangle\rangle_{t'}) =$ **Convert**$2D^{(+,\ell)}$**to**$2D^{+}$**(**$\mathcal{P}', t', \ell, \langle\langle s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}\rangle\rangle_{t'}$**)**

Let $f_i^{(l)}(x), 1 \le i \le n', 1 \le l \le \ell$ be the polynomials used for generating $\langle\langle s^{(1)} \ldots, s^{(\ell)}\rangle\rangle_{t'}$. For every pair of players P_i and P_j from $\mathcal{P}'$, the following is done:

1. Player P_i as a dealer executes **Distr** and **AuthVal** of **IC(**$P_i, P_j, \mathcal{P}, f_i^{(l)}(j)$**)** for all $l \in \{1, \ldots, \ell\}$ to give $ICSig_{f_i^{(l)}(j)}(P_i, P_j)$ to P_j for all $l \in \{1, \ldots, \ell\}$. Since $\langle\langle s^{(1)}, \ldots, s^{(\ell)}\rangle\rangle_{t'}$ is generated using $2D^{(+,\ell)}$**Share**, it implies that either P_j already holds a combined signature on $f_i^{(1)}(j), \ldots, f_i^{(\ell)}(j)$ from P_i, i.e. $ICSig_{(f_i^{(1)}(j), f_i^{(2)}(j), \ldots, f_i^{(\ell)}(j))}(P_i, P_j)$ or it may also happen that every player from $\mathcal{P}'$ has ignored P_i's signature. In the later case, players in $\mathcal{P}'$ will again ignore P_i's signature. Otherwise P_j can now check if P_i has given signature on the same individual values.

2. Upon receiving the signatures on say $\bar{f}_i^{(1)}(j), \ldots, \bar{f}_i^{(\ell)}(j)$ (i.e $ICSig_{\bar{f}_i^{(l)}(j)}(P_i, P_j)$ for $l = 1, \ldots, \ell$) , P_j checks $f_i^{(l)}(j) \stackrel{?}{=} \bar{f}_i^{(l)}(i)$. If there is inconsistency for some $l \in \{1, \ldots, \ell\}$ then P_j along with all players in $\mathcal{P}'$ invoke **EfficientRevealVal(**$P_i, P_j, \mathcal{P}', \ell, f_i^{(1)}(j), f_i^{(2)}(j), \ldots, f_i^{(\ell)}(j)$**)** and **RevealVal(**$P_i, P_j, \mathcal{P}', \bar{f}_i^{(l)}(j)$**)** for all $l \in \{1, \ldots, \ell\}$.

3. In the previous step, if P_j is not able to produce the signature that he received from P_i, then all the players from $\mathcal{P}'$ ignore the IC signatures received from P_j during step 1. Otherwise if the signatures are valid then $f_i^{(1)}(j), \ldots, f_i^{(\ell)}(j)$ and $\bar{f}_i^{(1)}(j), \ldots, \bar{f}_i^{(\ell)}(j)$ are public. All players in $\mathcal{P}'$ check $f_i^{(l)}(j) \stackrel{?}{=} \bar{f}_i^{(l)}(j)$ for $l = 1, \ldots, \ell$. If the test fails for some l, then all the players in $\mathcal{P}'$ ignore the values received from P_i during first step. Otherwise the signature produced by P_j will be ignored by all the players in $\mathcal{P}'$.

$s = 2D^{+}$**Recons(**$\mathcal{P}', t', \langle\langle s\rangle\rangle_{t'}$**)**

For all $P_j \in \mathcal{P}'$ such that P_j's IC signatures are not ignored by the players in $\mathcal{P}'$, player P_i sends $ICSig_{f_j(i)}(P_j, P_i)$ to every player P_k in $\mathcal{P}'$. Player $P_k \in \mathcal{P}'$ checks the validity of $ICSig_{f_j(i)}(P_j, P_i)$ with respect to his own *verification information.* If the verification passes then P_k accepts $ICSig_{f_j(i)}(P_j, P_i)$. Now if for all $P_j \in \mathcal{P}'$, P_k accepts $ICSig_{f_j(i)}(P_j, P_i)$ (which he receives from P_i) then P_k checks whether $f_j(i)$s are t'-consistent (ideally $f_j(i) = g_i(j)$ for all $P_j \in \mathcal{P}'$; so $f_j(i)$s will lie on t' degree polynomial $g_i(y)$). If yes then P_k adds P_i to his $CORE$ set and let the t' degree polynomial (on which $f_j(i)$s lie on) be $g_i(y)$. Player P_k takes all the $g_i(y)$ polynomials corresponding to the players in his $CORE$ and interpolates the bivariate polynomial $H(x, y)$ and finally sets the secret $s = H(0, 0)$. It is easy to check that all honest players from $\mathcal{P}'$ recovers the same secret s.

Lemma 7 *Protocol* $2D^{+}$**Recons** *takes one round and privately communicates* $\mathcal{O}(n^3\kappa)$ *bits.*

2.3 Generating Random t'-$2D^{(+,\ell)}$-Sharing

We now present protocol **Random(**$\mathcal{P}', t', \ell$**)** which allows the players in $\mathcal{P}'$ to jointly generate a random t'-$2D^{(+,\ell)}$-sharing, $\langle\langle r^{(1)}, \ldots, r^{(\ell)}\rangle\rangle_{t'}$, where each $r^{(l)}$ is a random element from $\mathbb{F}$.

$\langle\langle r^{(1)}, \ldots, r^{(\ell)}\rangle\rangle_{t'} =$ **Random(**$\mathcal{P}', t', \ell$**)**

Every player $P_i \in \mathcal{P}'$ invokes $2D^{(+,\ell)}$**Share(**$P_i, \mathcal{P}', t', \ell, r^{(1,P_i)}, \ldots, r^{(\ell,P_i)}$**)** to generate $\langle\langle r^{(1,P_i)}, \ldots, r^{(\ell,P_i)}\rangle\rangle_{t'}^{P_i}$, where $r^{(1,P_i)}, \ldots, r^{(\ell,P_i)}$ are randomly selected from $\mathbb{F}$. Let $Pass$ denotes the set of players P_i in $\mathcal{P}'$ such that t'-$2D^{(+,\ell)}$**Share(**$P_i, \mathcal{P}', t', \ell, r^{(1,P_i)}, \ldots, r^{(\ell,P_i)}$**)** is executed successfully. Now all the players in $\mathcal{P}'$ jointly computes $\langle\langle r^{(1)}, \ldots, r^{(\ell)}\rangle\rangle_{t'} = \sum_{P_i \in Pass} \langle\langle r^{(1,P_i)}, \ldots, r^{(\ell,P_i)}\rangle\rangle_{t'}^{P_i}$.

Lemma 8 *With overwhelming probability, protocol* **Random** *generates a random* t'*-*$2D^{(+,\ell)}$*-sharing* $\langle\langle r^{(1)}, \ldots, r^{(\ell)}\rangle\rangle_{t'}$ *in eight rounds and privately communicates and broadcasts* $\mathcal{O}((\ell n^3 + n^4)\kappa)$ *bits.*

2.4 Proving $c = ab$

Definition 6 t'**-**$1D^{(+)}$**-sharing:** *We say that a value* s *is correctly* t'*-*$1D^{(+)}$*-shared among the players in* $\mathcal{P}'$ *if there exists* t' *degree polynomial* $f(x)$ *held by* $\mathbf{D}$ *with* $f(0) = s$*. Every player* $P_i \in \mathcal{P}'$ *holds a share* $s_i = f(i)$ *of* s *with an IC signature on it from the dealer* $\mathbf{D}$ *(i.e.* $ICSig_{s_i}(\mathbf{D}, P_i)$*). We denote the* t'*-*$1D^{(+)}$*-sharing of a secret* s *by* $\langle s\rangle_{t'}$*.*

Definition 7 t'**-**$1D^{(+,\ell)}$**-sharing:** *We say that a set of secrets* $s^{(1)}, \ldots, s^{(\ell)}$ *are correctly* t'*-*$1D^{(+,\ell)}$*-shared among the players in* $\mathcal{P}'$ *if there exists* t' *degree polynomials* $f^{(1)}, \ldots, f^{(\ell)}$ *held by* $\mathbf{D}$*, with* $f^{(l)}(0) = s^{(l)}$ *for* $l = 1, \ldots, \ell$*. Every player* $P_i \in \mathcal{P}'$ *holds shares* $s_i^{(1)} = f^{(1)}(i), \ldots, s_i^{(\ell)} = f^{(\ell)}(i)$ *of* $s^{(1)}, \ldots, s^{(\ell)}$ *along with a single IC signature on them from the dealer* $\mathbf{D}$ *(i.e.* $ICSig_{(s_i^{(1)}, \ldots, s_i^{(\ell)})}(\mathbf{D}, P_i)$*)). We denote the* t'*-*$1D^{(+,\ell)}$*-sharing of* ℓ *length secret by* $\langle s^1, \ldots, s^\ell\rangle_{t'}$*.*

If a secret s is t'-$1D^{(+)}$-shared by a player $\mathbf{D} \in \mathcal{P}'$, then we denote it as $\langle s \rangle_{t'}^{\mathbf{D}}$. Similarly if ℓ secrets $s^{(1)}, \ldots, s^{(\ell)}$ are t'-$1D^{(+,\ell)}$-shared by player $\mathbf{D}$, we denote it by $\langle s^1, \ldots, s^\ell \rangle_{t'}^{\mathbf{D}}$. Notice that if $s^{(1)}, \ldots, s^{(\ell)}$ is t'-$2D^{(+,\ell)}$-shared, i.e. $\langle\langle s^{(1)}, \ldots, s^{(\ell)} \rangle\rangle_{t'}$, then the i^{th} shares of the secrets, namely $s_i^{(1)}, \ldots, s_i^{(\ell)}$ will be automatically t'-$1D^{(+,\ell)}$-shared by player P_i, i.e $\langle s_i^{(1)}, \ldots, s_i^{(\ell)} \rangle_{t'}^{P_i}$.

Now let $\mathbf{D} \in \mathcal{P}'$ holds ℓ pairs of values $(a^{(1)}, b^{(1)}), \ldots, (a^{(\ell)}, b^{(\ell)})$ such that $\mathbf{D}$ has already correctly t'-$1D^{(+,\ell)}$-shared $a^{(1)}, \ldots, a^{(\ell)}$ and $b^{(1)}, \ldots, b^{(\ell)}$ among the players in $\mathcal{P}'$. Now $\mathbf{D}$ wants to correctly t'-$2D^{(+,\ell)}$-share $c^{(1)}, \ldots, c^{(\ell)}$ without leaking any *additional* information about $a^{(l)}$, $b^{(l)}$ and $c^{(l)}$, such that every (honest) player in $\mathcal{P}'$ knows that $c^{(l)} = a^{(l)} b^{(l)}$ for $l = 1, \ldots, \ell$. We propose a protocol **ProveCeqAB** to achieve this task. The idea of the protocol is inspired from [11] with the following modification: we make use of our protocol $2D^{(+,\ell)}$-**Share**, which provides us with high efficiency.

We try to explain the idea of the protocol with a single pair (a, b). Thus $\mathbf{D}$ has already t'-$1D^{(+)}$-shared a and b using polynomials, say $f_a(x)$ and $f_b(x)$. Now he wants to generate t'-$2D^{(+)}$-sharing of c, where $c = ab$, without leaking any *additional* information about a, b and c. For this, he first selects a random non-zero $\beta \in \mathbb{F}$ and generates t'-$2D^{(+)}$-sharing of c, β and βb. Let $f_c(x), f_\beta(x)$ and $f_{\beta b}(x)$ are polynomials implicitly used for sharing c, β and βb. All the players in $\mathcal{P}'$ then jointly generate a random value r. $\mathbf{D}$ then broadcasts the polynomial $F_1(x) = r f_a(x) + f_\beta(x)$. Every player locally checks whether the appropriate linear combination of his shares lies on the broadcasted polynomial $F_1(x)$. If it does not then the player broadcasts $\mathbf{D}$'s signature on the shares of a and β. If the signature is valid and indeed the player's value does not lie on $F_1(x)$, then all the players will conclude that $\mathbf{D}$ fails to prove $c = ab$.

Otherwise, $\mathbf{D}$ again broadcasts $F_2(x) = F_1(0) f_b(x) - f_{\beta b}(x) - r f_c(x)$. As before every players locally checks whether the appropriate linear combination of his shares lies on the broadcasted polynomial $F_2(x)$. If it does not then the player broadcasts $\mathbf{D}$'s signature on the shares of b and βb and c. If the signature is valid and indeed the player's value does not lie on $F_2(x)$, then all players will conclude that $\mathbf{D}$ fails to prove $c = ab$. Otherwise every player checks whether $F_2(0) \overset{?}{=} 0$. If so the everybody accepts the t'-$2D^{(+)}$-sharing of c as valid t'-$2D^{(+)}$-sharing of ab.

The error probability of the protocol is negligible because of the random r which is jointly generated by all the players. Specifically, a corrupted $\mathbf{D}$ might have shared $\overline{\beta} \neq \beta, \overline{\beta b} \neq \beta b$ or $\overline{c} \neq c$ but still $F_2(0)$ can be zero and this will happen iff $f_{\overline{\beta b}}(x) + r f_{\overline{c}}(x) = f_{\beta b}(x) + r f_c(x)$. However this equation is satisfied by only one value of r. Since r is randomly generated, independent of $\mathbf{D}$, the probability that the equality will hold is $\frac{1}{|\mathbb{F}|}$ which is negligibly small. Now we can extend the above idea parallely for each of the ℓ pairs $(a^{(l)}, b^{(l)})$. The secrecy follows from the fact that the broadcasted polynomials $F_1(x)$ and $F_2(x)$ are randomly distributed with the constant coefficient of $F_2(x)$ as zero.

$\langle\langle c^{(1)}, \ldots, c^{(\ell)}\rangle\rangle_{t'}^{\mathbf{D}}$ =**ProveCeqAB**$(\mathbf{D}, \mathcal{P}', t', \ell, \langle a^{(1)}, \ldots, a^{(\ell)}\rangle_{t'}^{\mathbf{D}}, \langle b^{(1)}, \ldots, b^{(\ell)}\rangle_{t'}^{\mathbf{D}})$

1. **D** randomly generates a random non-zero ℓ length tuple $(\beta^{(1)}, \ldots, \beta^{(\ell)}) \in \mathbb{F}^{\ell}$. **D** then invokes $2D^{(+,\ell)}$**Share**$(\mathbf{D}, \mathcal{P}', t', \ell, c^{(1)}, \ldots, c^{(\ell)})$, $2D^{(+,\ell)}$**Share**$(\mathbf{D}, \mathcal{P}', t', \ell, \beta^{(1)}, \ldots, \beta^{(\ell)})$ and $2D^{(+,\ell)}$**Share**$(\mathbf{D}, \mathcal{P}', t', \ell, b^{(1)}\beta^{(1)}, \ldots, b^{(\ell)}\beta^{(\ell)})$ to verifiably t'-$2D^{(+,\ell)}$-share $(c^{(1)}, \ldots, c^{(\ell)})$, $(\beta^{(1)}, \ldots, \beta^{(\ell)})$ and $(b^{(1)}\beta^{(1)}, \ldots, b^{(\ell)}\beta^{(\ell)})$ respectively. If any of the **Share** protocol fails, then **D** fails and the protocol terminates. For $l = 1, \ldots, \ell$, let $a^{(l)}$, $b^{(l)}$, $c^{(l)}$, $\beta^{(l)}$ and $\beta^{(l)}b^{(l)}$ are implicitly shared using polynomials $f^{a^{(l)}}(x)$, $f^{b^{(l)}}(x)$, $f^{c^{(l)}}(x)$, $f^{\beta^{(l)}}(x)$ and $f^{\beta^{(l)}b^{(l)}}(x)$ respectively.

2. Now all the players in $\mathcal{P}'$ jointly generate a random number r. This is done as follows: first the players in $\mathcal{P}$ execute the protocol **Random**$(\mathcal{P}', t', 1)$ to generate $\langle\langle r\rangle\rangle_{t'}$. Then the players compute $r = 2D^{+}$**Recons**$(\mathcal{P}', t', \langle\langle r\rangle\rangle_{t'})$.

3. Now **D** broadcasts the polynomials $F^{(l)}(x) = rf^{a^{(l)}}(x) + f^{\beta^{(l)}}(x)$ for $l = 1 \ldots, \ell$.

4. Player $P_i \in \mathcal{P}'$ checks whether $F^{(l)}(i) \stackrel{?}{=} rf^{a^{(l)}}(i) + f^{\beta^{(l)}}(i)$ for $l = 1, \ldots, \ell$. If the test fails for at least one l, then P_i and all players invoke **EfficientRevealVal**$(\mathbf{D}, P_i, \mathcal{P}', \ell, f^{a^{(1)}}(i), \ldots, f^{a^{(\ell)}}(i))$ and **EfficientRevealVal**$(\mathbf{D}, P_i, \mathcal{P}', \ell, f^{\beta^{(1)}}(i), \ldots, f^{\beta^{(\ell)}}(i))$ to reveal $ICSig_{(f^{a^{(1)}}(i), \ldots, f^{a^{(\ell)}}(i))}(\mathbf{D}, P_i)$ and $ICSig_{(f^{\beta^{(1)}}(i), \ldots, f^{\beta^{(\ell)}}(i))}(\mathbf{D}, P_i)$. If the signature is invalid, ignore P_i's complaints. Otherwise all the values $(f^{a^{(1)}}(i), \ldots, f^{a^{(\ell)}}(i))$ and $(f^{\beta^{(1)}}(i), \ldots, f^{\beta^{(\ell)}}(i))$ become public. Using these values all players publicly checks whether $F^{(l)}(i) \stackrel{?}{=} rf^{a^{(l)}}(i) + f^{\beta^{(l)}}(i)$ for $l = 1, \ldots, \ell$. If the test fails for at least one l, then **D** fails and the protocol terminates here.

5. Now **D** broadcasts the polynomials $G^{(l)}(x) = F^{(l)}(0)f^{b^{(l)}}(x) - f^{\beta^{(l)}b^{(l)}}(x) - rf^{c^{(l)}}(x)$ for $l = 1 \ldots, \ell$.

6. Player $P_i \in \mathcal{P}'$ checks whether $G^{(l)}(i) \stackrel{?}{=} F^{(l)}(0)f^{b^{(l)}}(i) - f^{\beta^{(l)}b^{(l)}}(i) - rf^{c^{(l)}}(i)$ for $l = 1, \ldots, \ell$. If the test fails for at least one l, then P_i and all players in $\mathcal{P}'$ invoke **EfficientRevealVal**$(\mathbf{D}, P_i, \mathcal{P}', \ell, f^{b^{(1)}}(i), \ldots, f^{b^{(\ell)}}(i))$, **EfficientRevealVal**$(\mathbf{D}, P_i, \mathcal{P}', \ell, f^{\beta^{(1)}b^{(1)}}(i), \ldots, f^{\beta^{(\ell)}b^{(\ell)}}(i))$ and **EfficientRevealVal**$(\mathbf{D}, P_i, \mathcal{P}', \ell, f^{c^{(1)}}(i), \ldots, f^{c^{(\ell)}}(i))$ to reveal $ICSig_{(f^{b^{(1)}}(i), \ldots, f^{b^{(\ell)}}(i))}(\mathbf{D}, P_i)$, $ICSig_{(f^{\beta^{(1)}b^{(1)}}(i), \ldots, f^{\beta^{(\ell)}b^{(\ell)}}(i))}(\mathbf{D}, P_i)$ and $ICSig_{(f^{c^{(1)}}(i), \ldots, f^{c^{(\ell)}}(i))}(\mathbf{D}, P_i)$.

7. If the signature is invalid, ignore P_i's complaint. Otherwise all the values $(f^{b^{(1)}}(i), \ldots, f^{b^{(\ell)}}(i))$, $(f^{\beta^{(1)}b^{(1)}}(i), \ldots, f^{\beta^{(\ell)}b^{(\ell)}}(i))$ and $(f^{c^{(1)}}(i), \ldots, f^{c^{(\ell)}}(i))$ become public. Using these values all players publicly checks whether $G^{(l)}(i) \stackrel{?}{=} F^{(l)}(0)f^{b^{(l)}}(i) - f^{\beta^{(l)}b^{(l)}}(i) - rf^{c^{(l)}}(i)$ for $l = 1, \ldots, \ell$. If the test fails for at least one l, then **D** fails and protocol terminates here.

8. Every player checks whether $G^{(l)} \stackrel{?}{=} 0$ for $l = 1, \ldots, \ell$. If the test fails for at least one l, then **D** fails and protocol terminates here. Otherwise **D** has proved that $c^{(l)} = a^{(l)}b^{(l)}$ for $l = 1, \ldots, \ell$.

Lemma 9 *In protocol* **ProveCeqAB***, if* **D** *does not fail, then with overwhelming probability, every* $(a^{(l)}, b^{(l)})$*,* $c^{(l)}$ *satisfies* $c^{(l)} = a^{(l)}b^{(l)}$ *for* $l = 1, \ldots, \ell$*.* **ProveCeqAB** *takes twenty five rounds and communicates* $\mathcal{O}((\ell n^2 + n^4)\kappa)$ *bits and broadcasts* $\mathcal{O}((\ell n^2 + n^4)\kappa)$ *bits. Moreover, if* **D** *is honest then* $\mathcal{A}_t$ *learns no information about* $a^{(l)}, b^{(l)}$ *and* $c^{(l)}$*, for* $1 \leq l \leq \ell$*.*

2.5 Multiplication

Let two sets of ℓ values $a^{(1)}, \ldots, a^{(\ell)}$ and $b^{(1)}, \ldots, b^{(\ell)}$ are correctly t'-$2D^{(+,\ell)}$-shared among the players in $\mathcal{P}'$, i.e. $\langle\langle a^{(1)}, \ldots, a^{(\ell)}\rangle\rangle_{t'}$ and $\langle\langle b^{(1)}, \ldots, b^{(\ell)}\rangle\rangle_{t'}$. We now present a protocol called **Mult**, which allows the players to compute t'-$2D^{(+,\ell)}$-sharing $\langle\langle c^{(1)}, \ldots, c^{(\ell)}\rangle\rangle_{t'}$ such that $c^{(l)} = a^{(l)}b^{(l)}$ for $l = 1, \ldots, \ell$. Our protocol is based on the technique used in [11] with the following difference: we make use of our protocol **ProveCeqAB**, which provides us with high efficiency. We explain the idea with only one pair, say (a, b). Given that a and b are correctly t'-$2D^{(+)}$-shared among the players in $\mathcal{P}'$, it implies that implicitly P_i holds a_i and b_i where a_i and b_i are the i^{th} share of a and b respectively. Now multiplying a_i and b_i, P_i obtains i^{th} share $e_i = a_ib_i$ of c where $(e_1, \ldots, e_{n'})$ is $2t'$-1D-sharing of c. This is not what we desire. We want P_i to hold c_i such that c_i is the i^{th} share of t'-1D-sharing of c. For this, each player P_i t'-$2D^{(+)}$-shares the value $e_i = a_ib_i$ with the proof that e_i is indeed the multiplication of a_i and b_i. Now, all the players jointly hold $\langle\langle e_1\rangle\rangle_{t'}, \ldots, \langle\langle e_{n'}\rangle\rangle_{t'}$. Since $e_1, \ldots, e_{n'}$ are n' points on a $2t'$ degree polynomial, say $C(x)$ whose constant term is c, by Lagrange interpolation formula [10], c can be computed as $c = \sum_{i=1}^{n} r_ie_i$ where $r_i = \prod_{j=1, j\neq i}^{n'} \frac{x-j}{i-j}$. The vector $(r_1, \ldots, r_{n'})$ is called recombination vector [10] which is public and known to every player. So for shorthand notation, we write $c = Lagrange(e_1, \ldots, e_{n'}) = \sum_{i=1}^{n'} r_ie_i$. Now all players compute $\langle\langle c\rangle\rangle_{t'} = Lagrange(\langle\langle e_1\rangle\rangle_{t'}, \ldots, \langle\langle e_{n'}\rangle\rangle_{t'})$, to obtain the desired output. But notice that c is the constant term of a $2t'$ degree polynomial $C(x)$. So we need to have

$2t'+1$ of the n' t'-$2D^{(+)}$-shared values $\langle\langle e_1\rangle\rangle_{t'},\ldots,\langle\langle e_{n'}\rangle\rangle_{t'}$). The reason is that any $2t'$ degree polynomial needs at least $2t'+1$ points for its interpolation. So if $t'=t$ and $n'=2t+1$, then c can not be computed even if at least one player fails to t'-$2D^{(+)}$-share his e value. In general, to fail Protocol **Mult** at least $n'-(2t'+1)+1 = n'-2t'$ players must fail to t'-$2D^{(+)}$-share their corresponding e values. All such players will be removed from $\mathcal{P}'$ and will be added to $\mathcal{C}$.

$\langle\langle c^{(1)},\ldots,c^{(\ell)}\rangle\rangle_{t'} = $ **Mult**$(\mathcal{P}',\ell,\langle\langle a^{(1)},\ldots,a^{(\ell)}\rangle\rangle_{t'},\langle\langle b^{(1)},\ldots,b^{(\ell)}\rangle\rangle_{t'})$

1. Given $\langle\langle a^{(1)},\ldots,a^{(\ell)}\rangle\rangle_{t'}$, it implies that i^{th} shares of $a^{(1)},\ldots,a^{(\ell)}$ are t'-$1D^{(+)}$-shared by P_i i.e. $\langle a_i^{(1)},\ldots,a_i^{(\ell)}\rangle_{t'}^{P_i}$ for $P_i\in\mathcal{P}'$. Similarly given $\langle\langle b^{(1)},\ldots,b^{(\ell)}\rangle\rangle_{t'}$, we have $\langle b_i^{(1)},\ldots,b_i^{(\ell)}\rangle_{t'}^{P_i}$ for $P_i\in\mathcal{P}'$.
2. Player P_i in $\mathcal{P}'$ invokes **ProveCeqAB**$(P_i,\mathcal{P}',t',\ell,\langle a_i^{(1)},\ldots,a_i^{(\ell)}\rangle_{t'}^{P_i},\langle b_i^{(1)},\ldots,b_i^{(\ell)}\rangle_{t'}^{P_i})$ to generate $\langle\langle c_i^{(1)},\ldots,c_i^{(\ell)}\rangle\rangle_{t'}^{P_i}$.
3. If at least $n'-2t'$ players fails in executing **ProveCeqAB**, then remove them from $\mathcal{P}'$, adjust t' and terminate the protocol without generating the expected result. Otherwise for simplicity assume that the first $2t'+1$ players are successful in executing **ProveCeqAB**.
4. All the players compute: $\langle\langle c^{(1)},\ldots,c^{(\ell)}\rangle\rangle_{t'} = \sum_{i=1}^{2t'+1} r_i\langle\langle c_i^{(1)},\ldots,c_i^{(\ell)}\rangle\rangle_{t'}^{P_i}$, where $(r_1,\ldots,r_{2t'+1})$ represents the recombination vector.

Lemma 10 *With overwhelming probability, protocol* **Mult** *produces* $\langle\langle c^{(1)},\ldots,c^{(\ell)}\rangle\rangle_{t'}$ *from* $\langle\langle a^{(1)},\ldots,a^{(\ell)}\rangle\rangle_{t'}$ *and* $\langle\langle b^{(1)},\ldots,b^{(\ell)}\rangle\rangle_{t'}$ *such that* $c^{(l)}=a^{(l)}b^{(l)}$ *if less then* $n'-2t'$ *players are added to* $\mathcal{C}$. **Mult** *takes twenty five rounds and communicates* $\mathcal{O}((\ell n^3+n^5)\kappa)$ *bits and broadcasts* $\mathcal{O}((\ell n^3+n^5)\kappa)$ *bits. Moreover,* $\mathcal{A}_t$ *learns nothing about* $c^{(l)},a^{(l)}$ *and* $b^{(l)}$, *for* $1\le l\le\ell$.

2.6 Proving a=b

Consider the following scenario: Let **D**$\in\mathcal{P}'$ has t'-$1D^{(+,\ell)}$-shared ℓ values $a^{(1)},\ldots,a^{(\ell)}$ among the players in $\mathcal{P}'$. Now some more computation has been carried out after the sharing done by **D** and during the computation some players have been detected as faulty and removed from $\mathcal{P}'$. Let us denote the snapshot of $\mathcal{P}'$ before and after the computation by $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. Also assume $|\mathcal{P}_1|=n_1$ and the number of corrupted players in $\mathcal{P}_1$ is t_1 with $n_1\ge t+1+t_1$. Similarly $|\mathcal{P}_2|=n_2$ and the number of corrupted players in $\mathcal{P}_2$ is t_2 with $n_2\ge t+1+t_2$, $t_1>t_2$. Now **D** wants to correctly t_2-$2D^{(+,\ell)}$-share $b^{(1)},\ldots,b^{(\ell)}$ among the players of $\mathcal{P}_2$ such that $b^{(l)}=a^{(l)}$, without leaking any *additional* information about $a^{(l)}$. We propose a protocol **ProveAeqB** to achieve this task. We try to explain the idea of the protocol with a single value a. **D** has already t_1-$1D^{(+)}$-shared a among the players in $\mathcal{P}_1$ by implicitly using polynomial say $f_a(x)$. Now he wants to t_2-$2D^{(+)}$-share a among the players in $\mathcal{P}_2$. For this, he first generates t_2-$2D^{(+)}$-sharing of b with $b=a$. Let $f_b(x)$ be the polynomial implicitly used for sharing b. **D** then selects a random element $c\in\mathbb{F}$ and (t_1-1)-$2D^{(+)}$-shares c among the players in $\mathcal{P}_2$. Let $f_c(x)$ be the polynomial implicitly used for sharing c. Now to prove that $f_a(x)$ and $f_b(x)$ share the same value a, **D** broadcasts the polynomial $F(x)=f_a(x)+xf_c(x)-f_b(x)$. Every player from $\mathcal{P}_2$ locally checks whether the appropriate linear combination of his shares lies on the broadcasted polynomial $F(x)$. If not then the player broadcasts **D**'s signature on the shares of a, b and c. If the signature is valid and indeed the player's value does not lie on $F(x)$ then all players will conclude that **D** has failed to prove $b=a$. Otherwise every player checks whether $F(0)\overset{?}{=}0$. If so then everybody accepts the t_2-$2D^{(+)}$-sharing of b as valid t_2-$2D^{(+)}$-sharing of a. Our protocol **ProveAeqB** achieves this task for ℓ values simultaneously. The secrecy follows from the fact that $F(x)$ is randomly distributed with $F(0)=0$.

Lemma 11 *In protocol* **ProveAeqB**, *if* **D** *does not fail, then with overwhelming probability, every* $(a^{(l)},b^{(l)})$ *satisfies* $a^{(l)}=b^{(l)}$. **ProveAeqB** *takes thirteen rounds and communicates and broadcasts* $\mathcal{O}((\ell n^2+n^3)\kappa)$ *bits. Moreover, if* **D** *is honest then* $\mathcal{A}_t$ *learns no information about* $a^{(l)}$, *for* $1\le l\le\ell$.

2.7 Resharing

As described in previous section, consider the time-stamps before and after some computation where before and after the computation, $\mathcal{P}'$ is denoted by $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively. Let the players in $\mathcal{P}_1$ holds a t_1-$2D^{(+,\ell)}$-sharing of ℓ values $s^{(1)},\ldots,s^{(\ell)}$ i.e. $\langle\langle s^{(1)},\ldots,s^{(\ell)}\rangle\rangle_{t_1}$. Now the players want to jointly generate t_2-$2D^{(+,\ell)}$-sharings of same values i.e $\langle\langle s^{(1)},\ldots,s^{(\ell)}\rangle\rangle_{t_2}$ among the players

$\langle\langle a^{(1)},\ldots,a^{(\ell)}\rangle\rangle^{\mathbf{D}}_{t_2}$ = **ProveAeqB**$(\mathbf{D},\mathcal{P}_2,t_1,t_2,\ell,\langle a^{(1)},\ldots,a^{(\ell)}\rangle^{\mathbf{D}}_{t_1})$

1. **D** invokes $2D^{(+,\ell)}$**Share**$(\mathbf{D},\mathcal{P}_2,t_2,\ell,a^{(1)},\ldots,a^{(\ell)})$ to verifiably t_2-$2D^{(+,\ell)}$-share $(a^{(1)},\ldots,a^{(\ell)})$. **D** selects a random ℓ length tuple $(c^{(1)},\ldots,c^{(\ell)})\in\mathbb{F}^{\ell}$ and invokes $2D^{(+,\ell)}$**Share**$(\mathbf{D},\mathcal{P}_2,t_1-1,\ell,c^{(1)},\ldots,c^{(\ell)})$. If protocol $2D^{(+,\ell)}$**Share** fails then **D** fails here and the protocol terminates here. Otherwise for convenience we say that **D** has t_2-$2D^{(+,\ell)}$-shared $(b^{(1)},\ldots,b^{(\ell)})$. For an honest **D**, $a^{(l)}=b^{(l)}$ for $l=1,\ldots,\ell$.
2. For $l=1,\ldots,\ell$, let $a^{(l)}$, $b^{(l)}$ and $c^{(l)}$ are implicitly shared using polynomials $f^{a^{(l)}}(x)$ (degree t_1), $f^{b^{(l)}}(x)$ (degree t_2) and $f^{c^{(l)}}(x)$ (degree t_1-1) respectively. Now **D** broadcasts the polynomials $F^{(l)}(x)=f^{a^{(l)}}(x)+xf^{c^{(l)}}(x)-f^{b^{(l)}}(x)$ for $l=1\ldots,\ell$.
3. Player $P_i\in\mathcal{P}_2$ checks whether $F^{(l)}(i)\overset{?}{=}f^{a^{(l)}}(i)+if^{c^{(l)}}(i)-f^{b^{(l)}}(i)$ for $l=1,\ldots,\ell$. If the test fails for at least one l, **EfficientRevealVal**$(\mathbf{D},P_i,\mathcal{P}_1,\ell,f^{a^{(1)}}(i),\ldots,f^{a^{(\ell)}}(i))$, **EfficientRevealVal**$(\mathbf{D},P_i,\mathcal{P}_2,\ell,f^{b^{(1)}}(i),\ldots,f^{b^{(\ell)}}(i))$ and **EfficientRevealVal**$(\mathbf{D},P_i,\mathcal{P}_2,\ell,f^{c^{(1)}}(i),\ldots,f^{c^{(\ell)}}(i))$ are invoked to reveal signature of **D** on $(f^{a^{(1)}}(i),\ldots,f^{a^{(\ell)}}(i))$, $(f^{b^{(1)}}(i),\ldots,f^{b^{(\ell)}}(i))$ and $(f^{c^{(1)}}(i),\ldots,f^{c^{(\ell)}}(i))$ respectively.
4. If the signatures are invalid, then ignore P_i's complaints. Otherwise all the values $(f^{a^{(1)}}(i),\ldots,f^{a^{(\ell)}}(i))$, $(f^{b^{(1)}}(i),\ldots,f^{b^{(\ell)}}(i))$ and $(f^{c^{(1)}}(i),\ldots,f^{c^{(\ell)}}(i))$ become public. Using these values all players publicly checks whether $F^{(l)}(i)\overset{?}{=}f^{a^{(l)}}+if^{c^{(l)}}(i)-f^{b^{(l)}}(i)$ for $l=1,\ldots,\ell$. If the test fails for at least one l, then **D** fails and protocol terminates here.
5. Every player checks whether $F^{(l)}(0)\overset{?}{=}0$ for $l=1,\ldots,\ell$. If the test fails for at least one l, then **D** fails and protocol terminates here. Otherwise **D** has proved that $a^{(l)}=b^{(l)}$.

in $\mathcal{P}_2$ where $t_2<t_1$. Since $n_2\geq t+1+t_2$, generating t_2-$2D^{(+,\ell)}$-sharing among the players in $\mathcal{P}_2$ does not cause any loss of secrecy. We now present a protocol **Reshare** to perform this task.

$\langle\langle a^{(1)},\ldots,a^{(\ell)}\rangle\rangle_{t_2}$ = **Reshare**$(\mathcal{P}',t_1,t_2,\ell,\langle\langle a^{(1)},\ldots,a^{(\ell)}\rangle\rangle_{t_1})$

1. Given $\langle\langle a^{(1)},\ldots,a^{(\ell)}\rangle\rangle_{t_1}$, it implies that the i^{th} shares of $a^{(1)},\ldots,a^{(\ell)}$ are already t_1-$1D^{(+,\ell)}$-shared by P_i, i.e. $\langle a_i^{(1)},\ldots,a_i^{(\ell)}\rangle^{P_i}_{t_1}$ for $P_i\in\mathcal{P}_1$. Player P_i in $\mathcal{P}_2$ invokes **ProveAeqB**$(P_i,\mathcal{P}_2,t_1,t_2,\ell,\langle a_i^{(1)},\ldots,a_i^{(\ell)}\rangle^{P_i}_{t_1})$ to generate $\langle\langle a_i^{(1)},\ldots,a_i^{(\ell)}\rangle\rangle^{P_i}_{t_2}$. For simplicity assume that first t_1+1 players are successful in **ProveAeqB**. Since $n_2=t+1+t_2$ and $t>t_1$, at least $t+1\geq t_1+1$ honest players will always be successful.
2. All the players compute: $\langle\langle a^{(1)},\ldots,a^{(\ell)}\rangle\rangle_{t_2}=\sum_{i=1}^{t_1+1}r_i\langle\langle a_i^{(1)},\ldots,a_i^{(\ell)}\rangle\rangle^{P_i}_{t_2}$, where $(r_1,\ldots,r_{t_1+1})$ represents the recombination vector.

Lemma 12 *With overwhelming probability, protocol* **Reshare** *produces t_2-$2D^{(+,\ell)}$-sharing $\langle\langle a^{(1)},\ldots,a^{(\ell)}\rangle\rangle_{t_2}$ from $\langle\langle a^{(1)},\ldots,a^{(\ell)}\rangle\rangle_{t_1}$ with $t_2<t_1$.* **Reshare** *takes thirteen rounds and communicates $\mathcal{O}((\ell n^3+n^4)\kappa)$ bits and broadcasts $\mathcal{O}((\ell n^3+n^4)\kappa)$ bits. Moreover, $\mathcal{A}_t$ learns nothing about $a^{(l)}$, for $1\leq l\leq\ell$.*

2.8 Preparation Phase

We call a triple (a,b,c) as a multiplication triple if $c=ab$. The goal of the preparation phase is to generate correct d-$2D^{+}$-sharings of (c_M+c_R) secret multiplication triples where d denotes the number of corrupted players still present in $\mathcal{P}'$ at the end of preparation phase. So in total there will be c_M+c_R multiplication triples $(a^{(i)},b^{(i)},c^{(i)})$ such that $c^{(i)}=a^{(i)}b^{(i)}$ for $i=1,\ldots,(c_M+c_R)$. The generation of c_M+c_R multiplication triples is divided into $\lceil\log(t)\rceil$ segments, wherein each segment is responsible for generating d-$2D^{(+,\ell)}$-sharings of $\ell=\lceil\frac{c_M+c_R}{\log t}\rceil$ triples. Here we use a *novel* technique called *rapid player elimination* (RPE) which works in the following way: The computation of a segment is non-robust i.e. a segment may fail due to the (mis)behavior of certain number of corrupted players who reveal their identity during the execution of the segment. At the beginning of preparation phase we set the counter for keeping track the number of (segment) failures to zero i.e $f=0$. We create a win-win situation, where if a segment fails, then it must be due to the revelation/detection of at least 2^f new corrupted players. After removing the corrupted players from $\mathcal{P}'$, a fresh execution of the same segment will be started with f incremented by 1. This ensures that total $\lceil\log(t)\rceil-1$ failures can occur (i.e $f\leq\lceil\log(t)\rceil-1$) since $\lceil\log(t)\rceil-1$ failures are enough to reveal all the t corrupted players. Once all the t corrupted players are revealed, rest of the computation can run without occurrence of any failure. Specifically, in every segment one of the following occurs: (a) t_1-$2D^{(+,\ell)}$-sharings of $\ell = \lceil \frac{c_M + c_R}{\log t} \rceil$ secret multiplication triples will be generated, where t_1 denotes the number of corrupted players present in $\mathcal{P}'$ ($t_1 = t - |\mathcal{C}|$) at the beginning of the segment's execution; (b) the segment fails with at least 2^f corrupted players being eliminated from $\mathcal{P}'$ (and added to $\mathcal{C}$), where f denotes the number of failures occurred so far. But there are two problems here. We want all the triples to be d-$2D^+$-shared. But since the number of corrupted players in $\mathcal{P}'$ may change dynamically after every failure, the sharings produced in different segment may be of different degree. Also the sharing produced by segments are t_1-$2D^{(+,\ell)}$-sharings where t_1 may vary segment to segment. So, to achieve our goal, we first use protocol **Reshare** to obtain uniform d-$2D^{(+,\ell)}$-sharings for all the segments. Then, we use Protocol **Convert**$2D^{(+,\ell)}$**to**$2D^+$ to produce d-$2D^+$-sharings from d-$2D^{(+,\ell)}$-sharings. By using this approach, we can efficiently generate the triples with less communication overhead. Note that we could directly generate d-$2D^+$-sharings of the triples instead of generating them from d-$2D^{(+,\ell)}$-sharing. But this would require more communication overhead. So by first generating d-$2D^{(+,\ell)}$-sharings and then converting them into d-$2D^+$-sharings, we require less communication overhead.

```
Preparation Phase
1. Let $\mathcal{P}' = \mathcal{P}$, $n' = n$, $\ell = \lceil \frac{c_M + c_R}{\log t} \rceil$ and $t' = t$. Let $f = 0$. /* The counter for counting the number of failures */
2. For every segment $k = 1, \ldots, \lceil \log(t) \rceil$, do the following:
   (a) Set $\mathcal{P}_1 = \mathcal{P}'$, $n_1 = n'$ and $t_1 = t'$.
   (b) Invoke Random($\mathcal{P}_1, t_1, \ell$) twice in parallel to generate $\langle\langle a_k^{(1)}, \ldots, a_k^{(\ell)} \rangle\rangle_{t_1}$ and $\langle\langle b_k^{(1)}, \ldots, b_k^{(\ell)} \rangle\rangle_{t_1}$.
   (c) Invoke Mult($\mathcal{P}_1, \ell, \langle\langle a_k^{(1)}, \ldots, a_k^{(\ell)} \rangle\rangle_{t_1}, \langle\langle b_k^{(1)}, \ldots, b_k^{(\ell)} \rangle\rangle_{t_1}$) to generate $\langle\langle c_k^{(1)}, \ldots, c_k^{(\ell)} \rangle\rangle_{t_1}$ such that $c_k^{(l)} = a_k^{(l)} b_k^{(l)}$.
   (d) IF Mult fails to produce $\langle\langle c_k^{(1)}, \ldots, c_k^{(\ell)} \rangle\rangle_{t_1}$, then it must have removed $2^f$ new corrupted players from $\mathcal{P}'$ (i.e $|\mathcal{P}_1| - |\mathcal{P}'| = n_1 - n' \geq 2^f$). Thus repeat this segment with $f = f + 1$. ELSE increment $k$.
3. Now let $\mathcal{P}_2 = \mathcal{P}'$, $n_2 = n'$ and $d = t'$. For every segment $k = 1, \ldots, \lceil \log(t) \rceil$, check whether $c_k^{(1)}, \ldots, c_k^{(\ell)}$ are $d$-$2D^{(+,\ell)}$-shared. If $c_k^{(1)}, \ldots, c_k^{(\ell)}$ are $\alpha$-$2D^{(+,\ell)}$-shared with $\alpha > d$ then invoke Reshare($\mathcal{P}_2, \alpha, d, \ell, \langle\langle a_k^{(1)}, \ldots, a_k^{(\ell)} \rangle\rangle_\alpha$), Reshare($\mathcal{P}_2, \alpha, d, \ell, \langle\langle b_k^{(1)}, \ldots, b_k^{(\ell)} \rangle\rangle_\alpha$) and Reshare($\mathcal{P}_2, \alpha, d, \ell, \langle\langle c_k^{(1)}, \ldots, c_k^{(\ell)} \rangle\rangle_\alpha$) to generate $\langle\langle a_k^{(1)}, \ldots, a_k^{(\ell)} \rangle\rangle_d$, $\langle\langle b_k^{(1)}, \ldots, b_k^{(\ell)} \rangle\rangle_d$ and $\langle\langle c_k^{(1)}, \ldots, c_k^{(\ell)} \rangle\rangle_d$.
4. For every segment $k = 1, \ldots, \lceil \log(t) \rceil$, invoke Convert$2D^{(+,\ell)}$to$2D^+$($\mathcal{P}_2, d, \ell, \langle\langle a_k^{(1)}, \ldots, a_k^{(\ell)} \rangle\rangle_d$), Convert$2D^{(+,\ell)}$to$2D^+$($\mathcal{P}_2, d, \ell, \langle\langle b_k^{(1)}, \ldots, b_k^{(\ell)} \rangle\rangle_d$) and Convert$2D^{(+,\ell)}$to$2D^+$($\mathcal{P}_2, d, \ell, \langle\langle c_k^{(1)}, \ldots, c_k^{(\ell)} \rangle\rangle_d$) to obtain $\langle\langle a_k^{(1)} \rangle\rangle_d, \ldots, \langle\langle a_k^{(\ell)} \rangle\rangle_d$, $\langle\langle b_k^{(1)} \rangle\rangle_d, \ldots, \langle\langle b_k^{(\ell)} \rangle\rangle_d$ and $\langle\langle c_k^{(1)} \rangle\rangle_d, \ldots, \langle\langle c_k^{(\ell)} \rangle\rangle_d$.
```

Now we explain why in the f^{th} failure 2^f corrupted player must have revealed their identity. Initially $n' = n = 2t+1$ and $t' = t$ and $f = 0$. So in protocol **Mult** all the $2t'+1 = 2t+1$ players must be able to execute **ProveCeqAB** successfully (recall Protocol **Mult**). If a single ($2^f = 2^0 = 1$) player fails to do so, then **Mult** will fail and so the segment. So the parameters will be updated to $n' = 2t$, $t' = t-1$ and $f = 1$. Now the same segment will be executed with the changed parameters. This time **Mult** requires at least $2t' + 1 = 2(t-1) + 1 = 2t - 1$ players to successfully execute **ProveCeqAB**. Since there are $n' = 2t$ players, at least $2t - (2t-1) + 1 = 2 = 2^1 = 2^f$ players must fail to execute **ProveCeqAB** to fail **Mult** and hence the segment. Now the parameters will be updated to $n' = 2t - 2$, $t' = t - 3$ and $f = 2$. The same segment will be executed with the changed parameters. This time **Mult** requires at least $2t' + 1 = 2(t-3) + 1 = 2t - 5$ players to successfully execute **ProveCeqAB**. Since there are $n' = 2t-2$ players, at least $2t-2-(2t-5)+1 = 4 = 2^2 = 2^f$ players must fail to execute **ProveCeqAB** to fail **Mult** and hence the segment. Like this a segment may fail $\lceil \log(t) \rceil - 1$ times. But after the last failure, all the t faults will be revealed.

Lemma 13 *With overwhelming probability, protocol* **Preparation Phase** *produces correct d-$2D^+$-sharings of $(c_M + c_R)$ secret multiplication triples.* **Preparation Phase** *takes $\mathcal{O}(\log(t))$ rounds and communicates $\mathcal{O}((c_M + c_R)n^3 + n^4)\kappa$ bits and broadcasts $\mathcal{O}((c_M + c_R)n^3 + n^4)\kappa$ bits. Moreover, $\mathcal{A}_t$ learns nothing about $(a^{(i)}, b^{(i)}, c^{(i)})$ for $1 \leq i \leq \ell$.*

2.9 Input Phase

Once **Preparation Phase** is complete, the execution of **Input Phase** begins. The goal of the input phase is to generate d-$2D^+$-sharings of c_I inputs. Assume that player $P_i \in \mathcal{P}$ has c_i inputs. Thus

$c_I = \sum_{i=1}^{n} c_i$. We stress that though some players from $\mathcal{P}$ might have failed and removed during preparation phase, we still allow them to feed their input. Recall that at the end of preparation phase, $\mathcal{P}_2 = \mathcal{P}'$. So once all the players in $\mathcal{P}$ feed their input, the rest of the computation will be performed among the players in $\mathcal{P}_2$. For this, each input sharing is then reshared among the players in $\mathcal{P}_2$. Also if some player $\mathcal{P}_i$ fails to correctly share their input, then everybody accepts a default $d - 2D^{(+,c_i)}$ sharing on behalf of that player.

Input Phase

1. Every player $P_i \in \mathcal{P}$ with c_i inputs $s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{(c_i)}$, invokes $2D^{(+,c_i)}$**Share**$(P_i, \mathcal{P}, t, c_i, s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{(c_i)})$ to generate $\langle\langle s_i^{(1)}, \ldots, s_i^{(c_i)} \rangle\rangle_t$.
2. For every P_i, invoke **Reshare**$(\mathcal{P}_2, t, d, c_i, \langle\langle s_i^{(1)}, \ldots, s_i^{(c_i)} \rangle\rangle_t)$ to generate $\langle\langle s_i^{(1)}, \ldots, s_i^{(c_i)} \rangle\rangle_d$ provided $d < t$.
3. For every player P_i, invoke **Convert**$2D^{(+,c_i)}$**to**$2D^{+}(\mathcal{P}_2, d, c_i, \langle\langle s_i^{(1)}, \ldots, s_i^{(c_i)} \rangle\rangle_d)$ to obtain $\langle\langle s_i^{(1)} \rangle\rangle_d, \ldots, \langle\langle s_i^{(c_i)} \rangle\rangle_d$.

Lemma 14 *With overwhelming probability, protocol* **Input Phase** *produces correct* d*-*$2D^+$*-sharings of* c_I *inputs.* **Input Phase** *takes twenty eight rounds and communicates* $\mathcal{O}(c_I n^3 + n^5)\kappa$ *bits and broadcasts* $\mathcal{O}(c_I n^3 + n^5)\kappa$ *bits.*

2.10 Computation Phase

Once **Preparation Phase** and **Input Phase** are over, the computation of the circuit (of the agreed upon function f) proceeds gate-by-gate. First, to every random and every multiplication gate, a prepared d-$2D^+$-shared random triple is assigned. And a d-$2D^+$-shared input is assigned to the corresponding input gates. A gate (except output gate) g is said to be *computed* if a d-$2D^+$-sharings $\langle\langle x_g \rangle\rangle_d$ is computed for the gate. Note that all the random and input gates will be *computed* as soon as we assign d-$2D^+$-shared random triples (generated in **Preparation Phase**) and d-$2D^+$-shared inputs (generated in **Input Phase**) to them respectively. A gate is said to be in *ready state*, when all its fanin gates have been *computed*. In the **Computation Phase**, the circuit evaluation proceeds in rounds wherein each round all the ready gates will be computed parallely. Evaluation of input, random and addition gates do not require any communication. Evaluation of multiplication and output gate requires 2 and 1 call to Protocol $2D^+$**Recons** respectively. So the individual gates in the circuit are evaluated as shown in the table given in the sequel.

The correctness of the steps described for multiplication gate follows from [1] which introduced the technique called *Circuit Randomization.* So the *Circuit Randomization* [1] allows to evaluate a multiplication gate at the cost of two public reconstructions, given a preprocessed random multiplication triple. The trick is as follows: Let $z = xy$. Now z can be expressed as $z = ((x - a) + a)((y - b) + b) = (\alpha + a)(\beta + b)$, where (a, b, c) is a multiplication triple. So given $(\langle\langle a \rangle\rangle_d, \langle\langle b \rangle\rangle_d, \langle\langle c \rangle\rangle_d)$, $\langle\langle z \rangle\rangle_d$ can be computed as $\langle\langle z \rangle\rangle_d = \alpha\beta + \alpha\langle\langle b \rangle\rangle_d + \beta\langle\langle a \rangle\rangle_d + \langle\langle c \rangle\rangle_d$ after reconstructing α and β publicly. The security follows from the fact that α and β are random and independent of x and y for a random multiplication triple (a, b, c).

Computation Phase

Input Gate: $\langle\langle s\rangle\rangle_d = \text{IGate}(\langle\langle s\rangle\rangle_d)$

1. Assign a d-$2D^+$-sharing of an input, say $\langle\langle s\rangle\rangle_d$.

Random Gate: $\langle\langle a\rangle\rangle_d = \text{RGate}(\langle\langle a\rangle\rangle_d, \langle\langle b\rangle\rangle_d, \langle\langle c\rangle\rangle_d)$

1. Assign a d-$2D^+$-sharing of a multiplication triple, say $(\langle\langle a\rangle\rangle_d, \langle\langle b\rangle\rangle_d, \langle\langle c\rangle\rangle_d)$, where only the first component is used.

Addition Gate: $\langle\langle z\rangle\rangle_d = \text{AGate}(\langle\langle x\rangle\rangle_d, \langle\langle y\rangle\rangle_d)$

1. If $\langle\langle x\rangle\rangle_d$ and $\langle\langle y\rangle\rangle_d$ are the inputs to the addition gate, all players in $\mathcal{P}_2$ compute $\langle\langle z\rangle\rangle_d = \langle\langle x\rangle\rangle_d + \langle\langle y\rangle\rangle_d$ with $\langle\langle z\rangle\rangle_d$ as the output of the gate.

Multiplication Gate: $\langle\langle z\rangle\rangle_d = \text{MGate}(\langle\langle x\rangle\rangle_d, \langle\langle y\rangle\rangle_d, (\langle\langle a\rangle\rangle_d, \langle\langle b\rangle\rangle_d, \langle\langle c\rangle\rangle_d))$

1. Let $\langle\langle x\rangle\rangle_d$ and $\langle\langle y\rangle\rangle_d$ are the inputs to the multiplication gate and $(\langle\langle a\rangle\rangle_d, \langle\langle b\rangle\rangle_d, \langle\langle c\rangle\rangle_d)$ is the random multiplication triple assigned to it. Then all players in $\mathcal{P}_2$ compute the output $\langle\langle z\rangle\rangle_d)$ in the following way.
2. All players in $\mathcal{P}_2$ compute $\langle\langle \alpha\rangle\rangle_d = \langle\langle x\rangle\rangle_d - \langle\langle a\rangle\rangle_d$ and $\langle\langle \beta\rangle\rangle_d = \langle\langle y\rangle\rangle_d - \langle\langle b\rangle\rangle_d$.
3. All players in $\mathcal{P}_2$ invoke $2D^+$**Recons(**$\mathcal{P}_2, d, \langle\langle \alpha\rangle\rangle_d$**)** and $2D^+$**Recons(**$\mathcal{P}_2, d, \langle\langle \beta\rangle\rangle_d$**)** to reconstruct α and β.
4. All players in $\mathcal{P}_2$ compute $\langle\langle z\rangle\rangle_d = \alpha\beta + \alpha\langle\langle b\rangle\rangle_d + \beta\langle\langle a\rangle\rangle_d + \langle\langle c\rangle\rangle_d$.

Output Gate: $x = \text{OGate}(\langle\langle x\rangle\rangle_d)$

1. If $\langle\langle x\rangle\rangle_d$ is the input to the output gate, all players in $\mathcal{P}_2$ compute $x = 2D^+$**Recons(**$\mathcal{P}_2, d, \langle\langle x\rangle\rangle_d$**)**.

Lemma 15 *With overwhelming probability, protocol* **Computation Phase** *evaluates the circuit gate-by-gate in a shared fashion and outputs the desired outputs.* **Computation Phase** *takes $\mathcal{D}$ rounds and communicates $\mathcal{O}((c_M + c_O)n^3\kappa)$ bits.*

2.11 New UMPC Protocol

Now our new UMPC protocol for evaluating function f is: (1). Invoke **Preparation Phase** (2). Invoke **Input Phase** (3). Invoke **Computation Phase**.

Theorem 1 *With overwhelming probability, our new UMPC protocol can evaluate an agreed upon function securely against an active adaptive rushing adversary $\mathcal{A}_t$ with $t < n/2$ and requires $\mathcal{O}(\log(t) + \mathcal{D})$ rounds and communicates $\mathcal{O}(((c_I + c_R + c_M + c_O)n^3)\kappa)$ bits and broadcasts $\mathcal{O}((c_I + c_M + c_R)n^3 + n^5)\kappa$ bits.*

Using the protocol of [13] to simulate the broadcasts, the communication complexity and round complexity of our UMPC protocol is stated in the following theorem.

Theorem 2 *With overwhelming probability, our new UMPC protocol requires $\mathcal{O}(n\log(t) + \mathcal{D})$ rounds and communicates $\mathcal{O}(((c_I + c_M + c_R + c_O)n^4 + n^7)\kappa)$ bits.*

3 Conclusion

In this paper, we have proposed a new UMPC protocol which is the most round efficient UMPC protocol so far and ranks second according to the communication complexity. To design the protocol we have proposed several new techniques and sub-protocols which are first of their kind. It would be interesting to further reduce the communication complexity and round complexity of our UPMC protocol.

References

[1] D. Beaver. Efficient multiparty protocols using circuit randomization. In *Proc. of CRYPTO 1991*, volume 576 of *LNCS*, pages 420–432. Springer Verlag, 1991.

[2] Donald Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4(4):75–122, 1991.

[3] Z. Beerliová-Trubíniová and M. Hirt. Efficient multi-party computation with dispute control. In *Proc. of TCC*, pages 305–328, 2006.

[4] Z. Beerliová-Trubíniová and M. Hirt. Perfectly-secure MPC with linear communication complexity. In *Proc. of TCC 2008*, volume 4948 of *LNCS*, pages 213–230. Springer Verlag, 2008.

[5] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[6] P. Berman, J. A. Garay, and K. J. Perry. Bit optimal distributed consensus. In *Computer Science Research*, pages 313–322, 1992. Preliminary version appeared in STOC 89.

[7] L. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences (JCSS)*, 18(4):143–154, 1979. Preliminary version appeared in STOC 77.

[8] D. Chaum, C. Crpeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proc. of FOCS 1988*, pages 11–19, 1988.

[9] D. Chaum, I. Damgård, and J. van de Graaf. Multiparty computations ensuring privacy of each party's input and correctness of the result. In *CRYPTO 1987*, LNCS, 1987.

[10] R. Cramer and I. Damgård. *Multiparty Computation, an Introduction.* Contemporary Cryptography. Birkhuser Basel, 2005.

[11] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In *Proc. of EUROCRYPT 1999*, volume 1592 of *LNCS*, pages 311–326. Springer Verlag, 1999.

[12] I. Damgård and J. B. Nielsen. Scalable and unconditionally secure multiparty computation. In *Proc. of CRYPTO*, volume 4622 of *LNCS*, pages 572–590. Springer Verlag, 2007.

[13] Matthias Fitzi and Martin Hirt. Optimally Efficient Multi-Valued Byzantine Agreement. In *Proc. of PODC 2006*, pages 163–168. ACM, 2006.

[14] Z. Galil, S. Haber, and M. Yung. Cryptograpic computation: Secure fault-tolerant protocols and the public-key model. In *CRYPTO 1987*, LNCS, 1987.

[15] Rosario Gennaro, Yuval Ishai, Eyal Kushilevitz, and Tal Rabin. The round complexity of verifiable secret sharing and secure multicast. In *STOC*, pages 580–589, 2001.

[16] O. Golderich, S. Micali, and A. Wigderson. How to play a mental game– a completeness theorem for protocols with honest majority. In *STOC 1987*, pages 218–229, 1987.

[17] M. Hirt, U. Maurer, and B. Przydatek. Efficient secure multiparty computation. In *Proc. of ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 143–161. Springer Verlag, 2000.

[18] M. Hirt and U. M. Maurer. Robustness for free in unconditional multi-party computation. In *Proc. of CRYPTO 2001*, LNCS, 2001.

[19] J. Katz and C. Y. Koo. Round-efficient secure computation in point-to-point networks. In *Proc. of EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 311–328. Springer Verlag, 2007.

[20] Arpita Patra, Ashish Choudhary, AshwinKumar B.V, and C. Pandu Rangan. On Round Complexity of Unconditional VSS. Cryptology ePrint Archive, Report 2008/172, 2008.

[21] B. Pfitzmann and M. Waidner. Information-theoretic pseudosignatures and byzantine agreement for $t \geq n/3$. Technical report, IBM Research, 1996.

[22] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *STOC*, pages 73–85, 1989.

[23] A. C. Yao. Protocols for secure computations. In *Proc. of 23rd IEEE FOCS*, pages 160–164, 1982.