# Information Theoretically Secure Multi Party Set Intersection Re-Visited

**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**

We re-visit the problem of secure multiparty set intersection in information theoretic settings. In [16], Li et.al have proposed a protocol for multiparty set intersection problem with $n$ parties, that provides information theoretic security, when $t < \frac{n}{3}$ parties are corrupted by an active adversary having *unbounded computing power*. In [16], the authors claimed that their protocol takes six rounds of communication and communicates $\mathcal{O}(n^4m^2)$ field elements, where each party has a set containing $m$ field elements. However, we show that the round and communication complexity of the protocol in [16] is much more than what is claimed in [16]. We then propose a *novel* information theoretically secure protocol for multiparty set intersection with $n > 3t$, which significantly improves the "actual" round and communication complexity (as shown in this paper) of the protocol given in [16]. To design our protocol, we use several tools which are of independent interest.

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

## 1 Introduction

**Secure Multiparty Set Intersection (MPSI):** Consider a complete synchronous network $\mathcal{N}$, consisting of $n$ parties $\mathcal{P} = \{P_1, \ldots, P_n\}$, who are pairwise connected by a reliable and private channel. The parties do not trust each other and the distrust in the network is modelled by a centralized adversary $\mathcal{A}_t$, who has *unbounded computing* power and can actively corrupt at most $t$ parties in Byzantine fashion, where $t < \frac{n}{3}$. A Byzantine (or actively) corrupted party is under complete control of $\mathcal{A}_t$, who may force the party to behave arbitrarily. Any protocol over $\mathcal{N}$ is assumed to operate in a sequence of rounds. In each round, a party performs some local computation, sends new messages to the other parties through the private channels and publicly *broadcasts* some information, receives the messages that were sent by the other parties in current round on the private channels and the messages that were publicly broadcast by the other parties in current round. Here broadcast is a primitive, which allows a party to send some information *identically* to all other parties. If a physical broadcast channel is available in the system, then broadcast will take one round. Otherwise, we can simulate broadcast using a protocol among the parties in $\mathcal{P}$. Each party $P_i$ has a private data-set $S_i$, containing $m$ elements from a finite field $\mathbb{F}$. The goal of MPSI is to design a protocol that can compute the intersection of these $n$ sets, satisfying the following properties:

(1) Correctness: At the end of the protocol, each honest party correctly gets the intersection of the $n$ sets, irrespective of the behavior of the corrupted parties and (2) Secrecy: The protocol should not leak any *extra* information to the corrupted parties, other than what is implied by the input of the corrupted parties (i.e., the data-sets possessed by corrupted parties) and the final output (i.e., the intersection of all the $n$ data-sets).

[*]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.

MPSI problem is an interesting secure distributed computing problem and has huge practical applications such as online recommendation services, medical databases, data mining etc. [11].

**Existing Literature on MPSI:** The MPSI problem was first studied in *cryptographic* model in [11, 15], under the assumption that the adversary has *bounded computing power.* By representing the data-sets as polynomials, the set intersection problem is converted into the task of computing the common roots of $n$ polynomials in [11, 15]. This is done as follows: Let $S = \{s_1, s_2, \ldots, s_m\}$ be a set of size $m$, where $\forall i, s_i \in \mathbb{F}$. Now set $S$ can be represented by a polynomial $f(x)$ of degree $m$, where $f(x) = \prod_{i=1}^{m}(x - s_i) = a_0 + a_1x + \ldots + a_mx^m$. It is obvious that if an element $s$ is a root of $f(x)$, then $s$ is a root of $r(x)f(x)$ too, where $r(x)$ is a *random* polynomial of degree-$m$ over $\mathbb{F}$. Now for MPSI, party $P_i$ represents his set $S_i$, by a degree-$m$ polynomial $f^{(P_i)}(x)$ and supplies $f^{(P_i)}(x)$ (i.e. its $m+1$ coefficients), as his input, in a secure manner. Then all the parties jointly and securely compute

$$F(x) = (r^{(1)}(x)f^{(P_1)}(x) + r^{(2)}(x)f^{(P_2)}(x) + \ldots + r^{(n)}(x)f^{(P_n)}(x)) \qquad (1)$$

where $r^{(1)}(x), \ldots r^{(n)}(x)$ are $n$ secret random polynomials of degree-$m$ over $\mathbb{F}$, jointly generated by the $n$ parties. Note that $F(x)$ preserves all the common roots of $f^{(P_1)}(x), \ldots, f^{(P_n)}(x)$. Every element $s \in (S_1 \cap S_2 \cap \ldots \cap S_n)$ is a root of $F(x)$, i.e. $F(s) = 0$. Hence after computing $F(x)$ in a secure manner, it can be reconstructed towards every party, who locally checks if $F(s) = 0$ for every $s$ in his private set. All $s$'s at which the evaluation of $F(x)$ is zero forms the intersection set $(S_1 \cap S_2 \cap \ldots \cap S_n)$. In [15], it has been proved formally that $F(x)$ does not reveal any *extra* information to the adversary, other than what is deduced from $(S_1 \cap S_2 \cap \ldots \cap S_n)$ and input set $S_i$ of the corrupted parties.

**Remark 1** *Even though every $s \in (S_1 \cap S_2 \cap \ldots \cap S_n)$ is a root of $F(x)$, there may exist some $s' \in \mathbb{F}$, such that $F(s') = 0$, even though $s' \notin (S_1 \cap S_2 \cap \ldots \cap S_n)$. This is possible if $s'$ happens to be the common root of all $r^{(i)}(x)$'s. However, as stated in [15], the probability of this event is negligible.*

In [15], the MPSI problem is solved by securely computing $F(x)$, assuming $\mathcal{A}_t$ to be *computationally bounded.* In [16], the authors have presented the first information theoretically secure protocol for MPSI, assuming $\mathcal{A}_t$ to be *computationally unbounded* and $n \geq 3t + 1$. Specifically, the authors have shown how to securely compute $F(x)$ in the presence of a computationally unbounded $\mathcal{A}_t$. To the best of our knowledge, this is the only known information theoretically secure MPSI protocol. *Notice that, although not explicitly stated in [16], the MPSI protocol of [16] involves a negligible error probability in correctness. This is due to the argument given in Remark 1.*

**Our Motivation and Contribution:** The authors in [16] claimed that their MPSI protocol takes *six* rounds and communicates $\mathcal{O}(n^4m^2)$ elements from $\mathbb{F}$. [1] However, we show that the round and communication complexity of the MPSI protocol of [16] is much more than what is claimed in [16]. We then propose a new information theoretically secure protocol for MPSI with $n > 3t$, which significantly improves the "actual" round and communication complexity (as shown in this paper) of the MPSI protocol given in [16].

## 2 Round and Communication Complexity of MPSI Protocol of [16]

In order to securely compute $F(x)$ given in (1) against a computationally unbounded $\mathcal{A}_t$, the MPSI protocol of [16] is divided into three phases. We briefly recall the steps performed in each phase.

**1. Input Phase**: Here each party represent his private data-set as a polynomial and $t$-shares[2] the coefficients of the polynomial among the $n$ parties. In order to do the sharing, the parties use a two dimensional verifiable secret sharing (VSS). A two dimensional VSS [12, 10, 14], ensures that each party (including a corrupted party) "consistently" and correctly $t$-shares the coefficients of his polynomial with everybody. Now, the authors in [16] claimed that this takes two rounds, where in the first round, each party does the sharing and in the second round verification is done by all the parties to

[1]In [16], $k$ is used to denote the size of each set.

[2]We say that an element $c \in \mathbb{F}$ is $t$-shared among the $n$ parties, if there exists a random polynomial $p(x)$ over $\mathbb{F}$ of degree $t$ such $p(0) = c$ and each (*honest*) party $P_i$ has the share $p(i)$.

ensure whether everybody has received correct and consistent shares (see sec. 4.2 in [16]). Moreover, the authors have not provided the communication complexity of this phase. Now it is well known in the literature that the minimum number of rounds taken by any VSS protocol with $n \geq 3t+1$ is at least *three* [12, 10, 14]. Moreover, the current best three round VSS protocol with $n = 3t+1$ requires a private communication of $\mathcal{O}(n^3)$ and broadcast of $\mathcal{O}(n^3)$ field elements [10, 14]. Now in the **Input Phase** of [16], each party executes $(m+1)$ VSS's to share the coefficients of his secret polynomial. In addition, each party also executes $n(m+1)$ VSS's to share the coefficients of $n$ random polynomials, each of degree $m$. These polynomials are used to generate secret random polynomials $r^{(1)}(x), \ldots, r^{(n)}(x)$. So the total number of VSS done in **Input Phase** is $\mathcal{O}(n^2 m)$. Hence, the **Input Phase** will take at least three rounds, with a private communication of $\mathcal{O}(n^5 m)$ and broadcast of $\mathcal{O}(n^5 m)$ field elements. If the broadcast channel is not available, then simulation of broadcast of a single field element requires a private communication of $\mathcal{O}(n^2)$ field elements and $\Omega(t)$ rounds [17]. Thus, in the absence of broadcast channel, the **Input Phase** will require $\Omega(t)$ rounds and a communication complexity of $\mathcal{O}(n^7 m)$ field elements.

**2. Computation Phase**: Given that the coefficients of $f^{(P_1)}(x), \ldots, f^{(P_n)}(x), r^{(1)}(x), \ldots, r^{(n)}(x)$ are correctly $t$-shared, in the computation phase, the parties jointly try to compute $F(x) = r^{(1)}(x)f^{(P_1)}(x) + r^{(2)}(x)f^{(P_2)}(x) + \ldots + r^{(n)}(x)f^{(P_n)}(x)$, such that the coefficients of $F(x)$ are $t$-shared. For this, the parties execute a sequence of steps. But we recall only the first two steps, which are crucial in the communication and round complexity analysis of the **Computation Phase**.

During **step 1**, the parties locally multiply the shares of the coefficients of $r^{(i)}(x)$ and $f^{(P_i)}(x)$, for $1 \leq i \leq n$. This results in $2t$-sharing[3] of the coefficients of $f^{(P_i)}(x)r^{(i)}(x)$ for $1 \leq i \leq n$. During **step 2**, each party invokes a *re-sharing protocol* and converts the $2t$-sharing of the coefficients of $f^{(P_i)}(x)r^{(i)}(x)$ into $t$-sharing, for $1 \leq i \leq n$. The re-sharing protocol enables a party to generate $t$-sharing of an element, given the $t'$-sharing of the same element, where $t' > t$. In [16], the authors have called a re-sharing protocol, without giving the actual details and claimed that the re-sharing and other additional verifications will take *only three rounds*, with a private communication of $\mathcal{O}(n^4 m^2)$ field elements (see sec. 4.2 of [16]). The authors in [16] have given the reference of [13] for the details of re-sharing protocol. However, the protocol given in [13] is a protocol for general secure *Multiparty Computation* (MPC), which uses "circuit based approach" to securely evaluate a function. Specifically, the MPC protocol of [13] assumes that the (general) function to be computed is represented as an arithmetic circuit over $\mathbb{F}$, consisting of addition, multiplication, random, input and output gates. The re-sharing protocol of [13] was used to evaluate a multiplication gate. But the protocol was *non-robust* in the sense that it fails to achieve its goal when at least one of the parties misbehaves, in which case the protocol outputs a pair of parties such that at least one of them is corrupted. The authors in [16] have not mentioned clearly what will be the outcome of their protocol if the re-sharing protocol (whose details they have not given) fails during the **computation phase**.

To summarize, we can say that the details of the protocol given in [16] are incomplete. In addition, the round complexity and communication complexity of the protocol are not consistent with the given protocol. Moreover, it is not mentioned whether they have assumed a broadcast channel in the system. An estimation of the round complexity and communication complexity of the the MPSI protocol of [16] in the presence and in the absence of a physical broadcast channel is as follows:

1. If a physical broadcast channel is available in the system, then the **Input Phase** will require a private communication of $\Omega(n^5 m)$ field elements and broadcast of $\Omega(n^5 m)$ field elements. Moreover, the **Computation Phase** will take $\Omega(t)$ rounds.

2. If a physical broadcast channel is not available in the system, then the **Input Phase** will require a private communication of $\Omega(n^7 m)$ field elements. Moreover, the **Computation Phase** will take $\Omega(t^2)$ rounds.

[3]We say that an element $c \in \mathbb{F}$ is $2t$-shared among the $n$ parties if there exists a polynomial $p(x)$ over $\mathbb{F}$ of degree $2t$, such that $p(0) = c$ and each (*honest*) party $P_i$ has the share $p(i)$.

## 3 Our Results

We propose a new, information theoretically secure MPSI protocol with $n = 3t + 1$, tolerating a *computationally unbounded* $\mathcal{A}_t$. Our protocol is based on the approach of solving the MPSI by securely computing the function given in (1). Moreover, our protocol involves a negligible error probability in correctness [4]. However, as mentioned in Remark 1, any protocol for MPSI, based on computing the function in (1) will involve a negligible error probability in correctness. In the following table, we compare the round complexity (RC) and communication complexity (CC) of our MPSI protocol with the RC and CC of the MPSI protocol of [16] (as stated in the previous section). In the table, the CC is in terms of field elements. Moreover, CC/RC with (out) BC stands for communication complexity/round complexity in the presence (absence) of physical broadcast channel [5].

| Reference | CC with BC | | RC with BC | CC without BC | RC without BC |
|---|---|---|---|---|---|
| | Private | Broadcast | | Private | |
| [16] | $\Omega(n^5m)$ | $\Omega(n^5m)$ | $\Omega(t)$ | $\Omega(n^7m)$ | $\Omega(t^2)$ |
| This Paper | $\mathcal{O}((m^2n^3 + n^4\log(\lvert\mathbb{F}\rvert))$ | $\mathcal{O}(m^2n^3 + n^4\log(\lvert\mathbb{F}\rvert))$ | 58 | $\mathcal{O}(m^2n^5 + n^6\log(\lvert\mathbb{F}\rvert))$ | $\mathcal{O}(t)$ |

From the table, we find that our MPSI protocol significantly improves the round complexity and communication complexity of the MPSI protocol of [16], both in presence of a physical broadcast channel and in the absence of a physical broadcast channel.

### 3.1 Comparison of Our MPSI Protocol with Existing General MPC Protocols

The MPSI problem is a particular variant of general secure MPC problem [21]. Informally, in MPC problem, each party $P_i$ has a private input $x_i \in \mathbb{F}$. There is a publicly known function $f : \mathbb{F}^n \rightarrow \mathbb{F}^n$. At the end of computation of $f$, party $P_i$ gets $y_i \in \mathbb{F}$, such that $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$. The goal of any general MPC protocol is to securely compute $f$, where at the end of the protocol, all parties (honest) receive correct outputs, irrespective of the behavior of the adversary $\mathcal{A}_t$. Moreover, the messages seen by the adversary $\mathcal{A}_t$ during the protocol, should contain no *additional* information about the inputs and outputs of the honest parties, other than what can be computed from the inputs and outputs of the corrupted parties. The function $f$ to be computed is represented as an arithmetic circuit over the finite field $\mathbb{F}$, consisting of five type of gates, namely addition, multiplication, random, input and output. The number of gates of these types are denoted by $c_A, c_M, c_R, c_I$ and $c_O$ respectively. Any general MPC protocol tries to securely evaluate the circuit gate-by-gate, keeping all the inputs and intermediate results of the circuit as $t$-shared [3, 19].

The MPSI problem can be solved using any general MPC protocol. However, since a general MPC protocol does not exploit the nuances and the special properties of the problem, it is not efficient in general. Moreover, we do not know how to customize the generic MPC protocols to solve MPSI problem in an optimal fashion. However, we outline below a general approach and use the same to estimate the complexity of MPSI protocols, that could have been derived from general MPC protocols.

Suppose, we try to solve the MPSI by computing the function given in (1), using a general MPC protocol. The arithmetic circuit, representing the function in (1), will roughly require $c_I = n(m + 1)$ input gates (every party $P_i$ inputs $(m + 1)$ coefficients of his polynomial $f^{(P_i)}(x)$), $c_R = n(m + 1)$ random gates ($n$ polynomials $r^{(1)}(x), \ldots, r^{(n)}(x)$ have in total $n(m + 1)$ random coefficients), $c_M = n(m+1)^2$ multiplication gates (computing $r^{(1)}(x)f^{(P_i)}(x)$ requires $(m+1)^2$ co-efficient multiplications) and $c_O = 2m + 1$ output gates (the $2m + 1$ coefficients of $F(x)$ should be output). In the following table, we give the round complexity (RC) and communication complexity (CC) of the best known, information theoretically secure, general MPC protocols with $n = 3t + 1$, to securely compute the function given in (1), with the above number of gates. In the table, CC is in terms of field elements.

[4]There is no compromise in the secrecy.

[5]If a physical broadcast channel is not available, then we use the protocol of [4, 6], which takes $\mathcal{O}(t)$ rounds and private communication of $\mathcal{O}(n^2\ell)$ bits to simulate the broadcast of $\ell$ bit message.

| Reference | CC with BC | | RC with BC | CC without BC | RC without BC |
|---|---|---|---|---|---|
| | Private | Broadcast | | Private | |
| [3] | $\mathcal{O}(n^5m^2)$ | $\mathcal{O}(n^5m^2)$ | $\mathcal{O}(1)$ | $\mathcal{O}(n^7m^2)$ | $\mathcal{O}(n)$ |
| [13] | $\mathcal{O}(n^4m^2)$ | $\mathcal{O}(n^2)$ | $\mathcal{O}(n)$ | $\mathcal{O}(n^6m^2)$ | $\mathcal{O}(n^2)$ |
| [9] | $\mathcal{O}(n^2m^2)$ | $\mathcal{O}(n^2)$ | $\mathcal{O}(n)$ | $\mathcal{O}(n^4m^2)$ | $\mathcal{O}(n^2)$ |
| [2] | $\mathcal{O}(n^2m^2)$ | $\mathcal{O}(n^3)$ | $\mathcal{O}(n)$ | $\mathcal{O}(n^4m^2+n^5)$ | $\mathcal{O}(n^2)$ |
| This Paper | $\mathcal{O}(m^2n^3+n^4\log(\|\mathbb{F}\|))$ | $\mathcal{O}(m^2n^3+n^4\log(\|\mathbb{F}\|))$ | 58 | $\mathcal{O}(m^2n^5+n^6\log(\|\mathbb{F}\|))$ | $\mathcal{O}(n)$ |

From the table, we find that our protocol incurs much lesser communication complexity than the protocol of [3], while keeping round complexity same. But the protocols of [13, 9, 2] provides slightly better communication complexity than ours at the cost of increased round complexity. Round complexity and communication complexity are two important parameters of any distributed protocol. If we ever hope to practically implement MPSI protocols, then we should look for a solution, which tries to *simultaneously optimize* both these parameters. In this context, our MPSI protocol fits the bill more appropriately, then the protocols mentioned in the table.

*Though our main motive in this paper is to present a clean solution for MPSI problem, as a bi-product we have shown that our proposed protocol simultaneously optimizes both communication and round complexity, whereas existing general MPC protocols (when applied to solve MPSI) optimize only one of these two parameters.*

### 3.2 Overview of Our Protocol

As mentioned earlier, our MPSI protocol tries to securely compute the function given in (1). Our protocol is divided into three phases, namely (a) Input and Preparation phase; (b) Computation Phase and (c) Output Phase. In the Input and Preparation phase, the parties $t$-share the coefficients of their input polynomials. Moreover, the parties jointly generate the $t$-sharing of the secret random $r^{(i)}(x)$ polynomials. To achieve the first task, we design a new protocol called 1DShare, which further uses a new information checking protocol (ICP) called Multi-Secret-Multi-Receiver-ICP. The second task is achieved by a sub-protocol called Random. In the Computation Phase, the parties generate the $t$-sharing of the coefficients of $r^{(i)}(x)f^{(i)}(x)$. For this, we use sub-protocol Mult, which is a combination of few existing ideas from the literature and few new ideas presented in this paper. Finally, in the Output Phase, the coefficients of $F(x)$ are reconstructed by each party, by using sub-protocol ReconsPublic. In the next section, we give the technical details of each of the above mentioned sub-protocols.

## 4 Tools Used

Here we present a number of sub-protocols each solving a specific task. Finally, we combine them to design our MPSI protocol. All the sub-protocols that are presented here are designed to *concurrently* deal with $\ell \geq 1$ values. In the literature, the sub-protocols that achieve the same functionality as ours, were designed to deal with *single* value at a time. Our sub-protocols, concurrently dealing with $\ell$ values, are better in terms of communication complexity, than $\ell$ concurrent executions of the existing sub-protocols working with single value. Thus, our sub-protocols harness the advantage offered by dealing with multiple values concurrently (this fact will be more clear in the following sections).

*For convenience, we analyze the round complexity and communication complexity of the sub-protocols assuming the existence of physical broadcast channel in the system.* While presenting the sub-protocols, we assume that all computations and communications are done over a finite field $\mathbb{F}$, where $\mathbb{F} = GF(2^\kappa)$ and $\kappa$ is the error parameter. Thus, each field element can be represented by $\log(\mathbb{F}) = \mathcal{O}(\kappa)$ bits. Moreover, without loss of generality, we assume that $n = \text{poly}(\kappa)$.

### 4.1 Information Checking Protocol and IC Signatures

The Information Checking Protocol (ICP) is a tool for authenticating messages in the presence of computationally unbounded corrupted parties. The notion of ICP was first introduced by Rabin et.al [19]. As described in [19, 8], an ICP is executed among three parties: a dealer $D$, an intermediary $INT$ and a verifier $R$. The dealer $D$ hands over a secret value $s \in \mathbb{F}$ to $INT$. At a later stage, $INT$ is required to hand over $s$ to $R$ and convince $R$ that $s$ is indeed the value which $INT$ received from $D$.

The basic definition of ICP involves only a *single* verifier $R$ and deals with *only one* secret $s$ [19, 8]. We extend this notion to *multiple* verifiers, where all the $n$ parties in $\mathcal{P}$ act as verifiers. Thus our ICP is executed among three entities: the dealer $D \in \mathcal{P}$, an intermediary $INT \in \mathcal{P}$ and entire set $\mathcal{P}$ acting as verifiers. This will be later helpful in using ICP as a tool in our MPSI protocol. Moreover, we extend our ICP to deal with *multiple* secrets, denoted by $S$, which contains $\ell \geq 1$ secret values. Thus, our ICP is executed with respect to *multiple* verifiers and deals with *multiple* secrets concurrently. We call our ICP as Multi-Secret-Multi-Receiver-ICP. Now similar to the ICP defined in [19, 8], our Multi-Secret-Multi-Receiver-ICP is a sequence of following three protocols:

1. Distr($D, INT, \mathcal{P}, S$): is initiated by $D$, who hands over secret $S = [S^{(1)} \ldots S^{(\ell)}]$, containing $\ell \geq 1$ elements from $\mathbb{F}$ to $INT$. In addition, $D$ hands over some **authentication information** to $INT$ and **verification information** to the individual parties (verifiers) in $\mathcal{P}$.

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

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

The **authentication information**, along with $S$, which is held by $INT$ at the end of AuthVal is called $D$'s *IC signature* on $S$, denoted as $ICSig(D, INT, S)$. Multi-Secret-Multi-Receiver-ICP satisfies the following properties (which are almost same as the properties, satisfied by the ICP of [19, 8]):

1. If $D$ and $INT$ are uncorrupted, then $S$ will be accepted in RevealVal by each honest verifier.

2. If $INT$ is uncorrupted, then at the end of AuthVal, $INT$ knows an $S$, which will be accepted in RevealVal by each honest verifier, except with probability $2^{-\Omega(\kappa)}$.

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

4. If $D$ and $INT$ are uncorrupted, then at the end of AuthVal, $\mathcal{A}_t$ has no information about $S$.

We now present our novel protocol Multi-Secret-Multi-Receiver-ICP, with $n = 3t + 1$. The high level idea of the protocol is as follows: $D$ selects a random polynomial $F(x)$ of degree $\ell + n\kappa$ over $\mathbb{F}$, whose lower order $\ell$ coefficients are elements of $S$. In addition, $D$ also selects a random polynomial $R(x)$ of degree $\ell + n\kappa$ over $\mathbb{F}$, which is independent of $F(x)$. $D$ hands over $F(x)$ and $R(x)$ to $INT$. $D$ then associates $\kappa$ *random evaluation points* with each verifier $P_i$ and gives the value of $F(x), R(x)$ at those evaluation points to $P_i$. This prevents with very high probability, a corrupted $INT$, to produce an incorrect $F(x)$ during RevealVal, without being un-noticed by an honest verifier $P_i$. This ensures third property of ICP. In order to ensure second property, an honest $INT$ has to ensure that his $F(x)$ is consistent with the evaluation points of the honest verifiers. For this, $INT$ and the verifiers interact in a zero knowledge fashion and check the consistency of $F(x)$ and secret evaluation points. To maintain the secrecy of $S$ during the zero knowledge interaction, $INT$ uses the $R(x)$ polynomial. Due to space constraints, we prove the properties of protocol Multi-Secret-Multi-Receiver-ICP in **APPENDIX A** and only state the following lemma.

**Lemma 1** *Protocol Multi-Secret-Multi-Receiver-ICP takes five rounds and correctly generates IC signature on $\ell$ field elements, by privately communicating $\mathcal{O}((\ell+n\kappa)\kappa)$ bits and broadcasting $\mathcal{O}((\ell+n\kappa)\kappa)$ bits. The protocol works correctly, except with error probability of* $2^{-\Omega(\kappa)}$.

<u>**Important Notation**</u>: *In the rest of the paper, whenever we say that D hands over $ICSig(D, INT, S)$ to $INT$, we mean that Distr and AuthVal are executed in the background. Similarly, $INT$ reveals $ICSig(D, INT, S)$ can be interpreted as $INT$, along with other parties, invoking RevealVal.*

Multi-Secret-Multi-Receiver-ICP$(D, INT, \mathcal{P}, \ell, S = (s^{(1)}, \ldots, s^{(\ell)}))$

Distr$(D, INT, \mathcal{P}, \ell, S)$ **Round 1**: $D$ selects a random polynomial $F(x)$ of degree $\ell + n\kappa$ over $\mathbb{F}$, whose lower order $\ell$ coefficients are elements of $S$. In addition, $D$ selects another random polynomial $R(x)$ of degree $\ell + n\kappa$ over $\mathbb{F}$, which is independent of $F(x)$. $D$ also selects $n\kappa$ random, non-zero, distinct evaluation points from $\mathbb{F}$, denoted by $\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,\kappa}$, for $1 \leq i \leq n$. $D$ privately gives $F(x)$ and $R(x)$ to $INT$. To verifier $P_i \in \mathcal{P}$, $D$ privately gives $(\alpha_{i,l}, a_{i,l}, b_{i,l})$, for $l = 1, \ldots, \kappa$, where $a_{i,l} = F(\alpha_{i,l})$ and $b_{i,l} = R(\alpha_{i,l})$.

AuthVal$(D, INT, \mathcal{P}, \ell, S)$ **Round 2**: $INT$ chooses a random $d \in \mathbb{F} \setminus \{0\}$ and broadcasts $(d, B(x) = F(x) + dR(x))$. Parallely, each verifier $P_i \in \mathcal{P}$ broadcasts a random subset of indices $l_1, ..., l_{\frac{\kappa}{2}}$, the evaluation points $\alpha_{i,l_1}, ..., \alpha_{i,l_{\frac{\kappa}{2}}}$ and the values $a_{i,l_1}, ..., a_{i,l_{\frac{\kappa}{2}}}$ and $b_{i,l_1}, ..., b_{i,l_{\frac{\kappa}{2}}}$. *Notice that each verifier randomly selects the subset of indices* $l_1, ..., l_{\frac{\kappa}{2}}$*, independent of other verifiers.*

**Round 3:** $D$ checks if for at least $2t + 1$ verifiers $P_i$, it holds that $a_{i,l} + db_{i,l} = B(\alpha_{i,l})$, for all $l$ in the set of random indices broadcasted by $P_i$ in **Round 2**. If the above condition is not satisfied for at least $2t + 1$ verifiers, then $D$ broadcasts the polynomial $F(x)$.

**Local Computation (by each party):** IF $F(x)$ is broadcasted in **Round 3**, then $INT$ replaces the $F(x)$ received from $D$ during **Round 1**, with the $F(x)$ which is broadcasted in **Round 3**. Accordingly, each verifier $P_i$ adjust his $a_{i,l}$ (as received in **Round 1**), for $l = 1, \ldots, \kappa$, such that $F(\alpha_{i,l}) = a_{i,l}$ holds. ELSE say that verifier $P_i$ *accepts* $INT$ if $a_{i,l} + db_{i,l} = B(\alpha_{i,l})$, for all $l$ in the set of random indices, broadcasted by $P_i$ in **Round 2**.

The polynomial $F(x)$ is called $D$'s **IC signature** on $S = (s^{(1)}, \ldots, s^{(\ell)})$ given to $INT$, which is denoted by $ICSig(D, INT, S)$.

RevealVal$(D, INT, \mathcal{P}, \ell, S)$: (a) **Round 4**: $INT$ broadcasts $F(x)$; (b) **Round 5**: Each verifier $P_i \in \mathcal{P}$ broadcasts all the evaluation points $\alpha_{i,l}$ which were not broadcasted during **Round 2** and $a_{i,l}$ corresponding those indices.

**Local Computation (by each party)**: Say that verifier $P_i$ *re-accepts* $INT$ if for one of the newly revealed (by $P_i$) points, it holds that $a_{i,l} = F(\alpha_{i,l})$. If there are at least $t + 1$ verifiers who *re-accepts* $INT$, then accept the lower order $\ell$ coefficients of $F(x)$ as $S = (s^{(1)}, \ldots, s^{(\ell)})$. In this case, we say that $D$'s signature on $S$ is correct. Else reject $F(x)$ broadcasted by $INT$ and we say that $INT$ has failed to produce $D$'s signature.

**Remark 2 (Comparison with Existing ICP)** *The current best known ICP is due to [8], which privately communicates and broadcasts $\mathcal{O}(n\kappa)$ bits, to generate IC signature on a single secret.* [6] *Had we executed $\ell$ times the ICP of [8], dealing with single secret, the communication complexity would turn out to be $\mathcal{O}(\ell n\kappa)$ bits (both private and broadcast). However, the communication complexity of* Multi-Secret-Multi-Receiver-ICP *considering all the $\ell$ secrets concurrently is $\mathcal{O}((\ell + n\kappa)\kappa)$ bits (both private and broadcast). This clearly shows that if $\ell$ is significantly large, which is the case in our MPSI protocol, then executing a single instance of* Multi-Secret-Multi-Receiver-ICP*, dealing with multiple secrets concurrently, is advantageous over executing multiple instances of ICP of [8], dealing with single secret. The same principle holds for other sub-protocols, which are described in the sequel.*

## 4.2 Generating $\ell$ Length Random Vector

We now present a protocol called RandomVector$(\mathcal{P}, \ell)$, which allows the parties in $\mathcal{P}$ to jointly generate a vector, containing $\ell$ random elements from $\mathbb{F}$. Following the idea of [9], protocol RandomVector uses Vandermonde Matrix and its capability to extract randomness. Protocol RandomVector also uses the four round perfect VSS (verifiable secret sharing) protocol of [12] (see Fig 2 of [12]) as black box. The perfect VSS (see the definition of VSS in Section 2.1 of [12]) with $n \geq 3t + 1$ parties consists of two phases, namely Sharing Phase and Reconstruction Phase. The Sharing Phase takes four rounds and allows a dealer $D$ (which can be any party from the set of $n$ parties) to verifiably share a secret $s \in \mathbb{F}$ by privately communicating $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits and broadcasting $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits where $|\mathbb{F}| \geq n$. The Reconstruction Phase takes single round and allows all the (honest) parties to reconstruct the secret $s$ (shared by $D$ in Sharing Phase) by broadcasting $\mathcal{O}(n \log |\mathbb{F}|)$ bits in total. Notice that, in our context, $|\mathbb{F}| = 2^\kappa \geq n$. The VSS protocol has an important property that once $D$ (possibly corrupted) shares a secret $s$ during Sharing Phase, then $D$ is *committed* to $s$. Later, in the Reconstruction Phase, irrespective of the behavior of the corrupted parties, the same $s$ will be reconstructed. Thus a corrupted $D$ will not be able to change his commitment from $s$ to any other value, with the help of corrupted parties, during Reconstruction Phase.

[6]Though the ICP of [8] is designed with $n = 2t + 1$, the protocol when executed with $n = 3t + 1$, will result in the same communication complexity.

$(r^{(1)}, \ldots, r^{(\ell)}) =$ **RandomVector**$(\mathcal{P}, \ell)$

1. Every party $P_i \in \mathcal{P}$ selects $L = \lceil \frac{\ell}{2t+1} \rceil$ random elements $r^{(1,P_i)}, \ldots, r^{(L,P_i)}$ from $\mathbb{F}$.
2. Every party $P_i \in \mathcal{P}$ as a dealer invokes Sharing Phase of four round VSS protocol of [12] with $n \geq 3t+1$ for sharing each of the values $r^{(1,P_i)}, \ldots, r^{(L,P_i)}$.
3. For reconstructing the values $r^{(1,P_i)}, \ldots, r^{(L,P_i)}$ (shared by $P_i$ in Sharing Phase), the Reconstruction Phase of four round VSS of [12] with $n \geq 3t+1$ is invoked for $L$ times separately. Now corresponding to every $P_i \in \mathcal{P}$, the values $r^{(1,P_i)}, \ldots, r^{(L,P_i)}$ are public.
4. Now parties compute $(r^{(1,1)}, \ldots, r^{(1,2t+1)}) = (r^{(1,P_1)}, \ldots, r^{(1,P_n)})V$, $(r^{(2,1)}, \ldots, r^{(2,2t+1)}) = (r^{(2,P_1)}, \ldots, r^{(2,P_n)})V, \ldots,$ $(r^{(L,1)}, \ldots, r^{(L,2t+1)}) = (r^{(L,P_1)}, \ldots, r^{(L,P_n)})V$. Here $V$ is a $n \times (2t+1)$ publicly known Vandermonde matrix over $\mathbb{F}$.

The values $r^{(1,1)}, \ldots, r^{(1,2t+1)}, \ldots, r^{(L,1)}, \ldots, r^{(L,2t+1)}$ constitute the elements of $\ell$ length random vector.

As mentioned above, protocol RandomVector uses the properties of Vandermonde matrix to generate the random vector. We now provide a brief discussion on Vandermonde matrix.

**Vandermonde Matrix and Randomness Extraction [20, 9]:** Let $\beta_1, \ldots, \beta_c$ be distinct and publicly known elements from $\mathbb{F}$. We denote an $(r \times c)$ Vandermonde matrix by $V^{(r,c)}$, where for $1 \leq i \leq c$, the $i^{th}$ column of $V^{(r,c)}$ is $(\beta_i^0, \ldots, \beta_i^{r-1})^T$. The idea behind extracting randomness using $V^{(r,c)}$ is as follows: without loss of generality, assume that $r > c$. Moreover, let $(x_1, \ldots, x_r)$ be such that (a) *any* $c$ elements of it are chosen uniformly at random from $\mathbb{F}$ and are unknown to adversary $\mathcal{A}_t$, (b) the remaining $r - c$ elements are chosen with an arbitrary distribution from $\mathbb{F}$, independent of the $c$ elements, and are also known to $\mathcal{A}_t$ . Now if we compute $(y_1, \ldots, y_c) = (x_1, \ldots, x_r)V$, then $(y_1, \ldots, y_c)$ is a random vector of length $c$ unknown to $\mathcal{A}_t$, extracted from $(x_1, \ldots, x_r)$ [20, 9]. This principle is used in protocol RandomVector.

**Lemma 2** *Protocol RandomVector generates $\ell$ length random vector in five rounds and privately communicates $\mathcal{O}(\ell n^2 \kappa)$ bits and broadcasts $\mathcal{O}(\ell n \kappa)$ bits.*

PROOF: Communication and round complexity is easy to understand. The correctness follows from the correctness of the four round perfect VSS of [12] and the above discussion. □

## 4.3 Unconditional Verifiable Secret Sharing and Reconstruction

**Definition 1 ($d$-1D-sharing [1])** *: A value $s$ is correctly $d$-1D-shared among the parties in $\mathcal{P}$ if every honest party $P_i \in \mathcal{P}$ is holding a share $s_i$ of $s$, such that there exists a degree-$d$ polynomial $f(x)$ over $\mathbb{F}$ with $f(0) = s$ and $f(i) = s_i$ for every $P_i \in \mathcal{P}$. The vector $(s_1, s_2, \ldots, s_n)$ of shares is called a $d$-sharing of $s$ and is denoted by $[s]_d$. We may skip the subscript $d$ when it is clear from the context.*

If a secret $s$ is $d$-1D-shared by $D \in \mathcal{P}$, then we denote it as $[s]_d^D$. In the sequel, we describe a new protocol called 1DShare, which allows a dealer $D \in \mathcal{P}$ to $t$-1D-share $\ell$ secret values $s^{(1)}, \ldots, s^{(\ell)}$, where $\ell \geq 1$, with very high probability. If $D$ behaves correctly during the protocol, then each honest $P_i \in \mathcal{P}$ will hold $i^{th}$ shares $s_i^{(1)}, \ldots, s_i^{(\ell)}$, of the secrets $s^{(1)}, \ldots, s^{(\ell)}$ (respectively), at the end of the protocol.

*Notice that the desired sharing for each $s^{(i)}$ (separately) can be produced using a perfect (i.e., without any error) VSS protocol with $n \geq 3t+1$ [12, 10, 14]. However, this will involve more communication complexity than 1DShare which performs the same task with less communication complexity (but with a negligible error probability). 1DShare achieves this by incorporating the ideas of [9] and using Multi-Secret-Multi-Receiver-ICP as building block.*

The goal of 1DShare is as follows: (a) If $D$ is honest, then the protocol generates $[s^{(1)}]_t, \ldots, [s^{(\ell)}]_t$ with very high probability, such that the secrets $s^{(1)}, \ldots, s^{(\ell)}$ remain information theoretically secure from $\mathcal{A}_t$. (b) If $D$ is corrupted and has not generated $t$-1D-sharing of secrets, then with high probability, $D$ will be detected as corrupted during a public verification process. Moreover, every honest party accepts a pre-defined $t$-1D-sharing of $\ell$ 1's, namely $[1]_t, [1]_t, \ldots, [1]_t$ ($\ell$ times), on behalf of $D$.

Informally, the protocol works as follows: $D$ chooses $\ell + 1$ random polynomials $f^{(0)}(x), \ldots, f^{(\ell)}(x)$ over $\mathbb{F}$, each of degree $t$, such that $f^{(0)}(0) = r$ and $f^{(l)}(0) = s^{(l)}$ for $l = 1, \ldots, \ell$. Here $r$ is a random non-zero element from $\mathbb{F}$. $D$ then hands over his IC signature on $i^{th}$ points of $f^{(l)}(x)$ polynomials *concurrently* to party $P_i$. After this, the parties jointly produce a non-zero random value $z$. Now $D$ is asked to broadcast a linear combination of the $\ell + 1$ polynomials, where the scalars of the linear combination are function of $z$. At the same time, each party $P_i$ is asked to broadcast his corresponding linear combination of points. Ideally, the linear combination of points, broadcasted by the individual parties, should lie on the linear combination of the polynomial broadcasted by $D$. If this happens, then with very high probability, $D$ has correctly $t$-1D-shared each $s^{(l)}$. Otherwise, there is a party, say $P_i$, for which the above condition is not satisfied. In this case, $P_i$ is asked to reveal $D$'s signature on the $i^{th}$ points of $f^{(l)}(x)$ polynomials that he has received from $D$. In case $P_i$ is able to correctly produce the signature, $D$ is detected to be corrupted and the protocol terminates, with each party assuming predefined $t$-1D-sharing of $\ell$ 1's, namely $[1]_t, [1]_t, \ldots, [1]_t$, on behalf of $D$.

**Lemma 3** *In protocol 1DShare, if $D$ is honest, then $t$-1D-sharing of $s^{(1)}, \ldots, s^{(\ell)}$ are generated, except with error probability of $2^{-\Omega(\kappa)}$. Moreover, $\mathcal{A}_t$ will have no information about the secrets. On the other hand, if $D$ is corrupted and any of the values $r, s^{(1)}, \ldots, s^{(\ell)}$ is not $t$-1D-shared, then $D$ will be caught, except with error probability of $2^{-\Omega(\kappa)}$. The protocol takes eleven rounds, privately communicates $\mathcal{O}((\ell n + n^2\kappa)\kappa)$ bits and broadcasts $\mathcal{O}((\ell n + n^2\kappa)\kappa)$ bits.*

Proof (sketch): The communication complexity and number of rounds can be checked easily by inspection. We now prove the correctness. If $D$ is honest, then $f(i) = y_i$ for all honest $P_i$'s. However, a corrupted party $P_i$ may broadcast incorrect $y'_i \neq y_i$ , such that $y'_i \neq f(i)$ and can forge honest $D$'s IC signature on the corresponding incorrect $S'_i \neq S_i$, where $S'_i = (r'_i, {s'}^{(1)}_i, \ldots, {s'}^{(\ell)}_i)$ and $r'_i \neq r_i$ or/and ${s'}^{(l)}_i \neq s^{(l)}_i$, for $l = 1 \ldots \ell$. In this case, everybody will reject the sharing done by $D$. However, from the properties of Multi-Secret-Multi-Receiver-ICP protocol, the above event can happen with error probability $2^{-\Omega(\kappa)}$. The secrecy of the secrets $s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}$ for an honest $D$, follows from the fact that $\mathcal{A}_t$ will have $t$ shares for each $s^{(i)}, 1 \leq i \leq n$. In addition, the value $f(0)$ is blinded with a random value $r$, chosen by $D$. Thus, $\mathcal{A}_t$ will have no information about the secrets.

Next, we consider the case, when $D$ is corrupted and the sharing of at least one of the values $r, s^{(1)}, \ldots, s^{(\ell)}$ is not a correct $t$-1D-sharing, i.e. the shares of the honest parties lie on a polynomial of degree higher than $t$. In this case, it can be shown that with very high probability, the polynomial $f^{def}(x)$, defined by the $y_i$'s, corresponding to the honest parties, will be of degree more than $t$ (see **APPENDIX B**). Now if $f^{def}(x)$ is of degree more than $t$, then for at least one honest party $P_i$, it will hold that $f(i) \neq y_i$. Moreover, from the properties of Multi-Secret-Multi-Receiver-ICP, the honest $P_i$ will be able to correctly produce $ICSig(D, P_i, S_i)$, except with probability $2^{-\Omega(\kappa)}$, where $S_i = (r_i, s^{(1)}_i, \ldots, s^{(\ell)}_i)$. Furthermore, everybody will verify that $f(i) \neq y_i (= r_i + \sum_{l=1}^{\ell} s^{(l)}_i z^l)$ and hence will reject the sharings done by $D$. For complete proof, see **APPENDIX B**. □

$([s^{(1)}]^D_t, \ldots, [s^{(\ell)}]^D_t) =$ **1DShare**$(D, \mathcal{P}, \ell, s^{(1)}, s^{(2)}, \ldots, s^{(\ell)})$

1. For $l = 1, \ldots, \ell$, $D$ picks a random polynomial $f^{(l)}(x)$ over $\mathbb{F}$ of degree-$t$, with $f^{(l)}(0) = s^{(l)}$. $D$ also chooses a random polynomial $f^{(0)}(x)$ of degree-$t$ with $f^{(0)}(0) = r$ where $r$ is a random, non-zero element from $\mathbb{F}$. For $i = 1, \ldots, n$, let $S_i = (r_i, s^{(1)}_i, s^{(2)}_i, \ldots, s^{(\ell)}_i)$, where $r_i = f^{(0)}(i)$ and $s^{(l)}_i = f^{(l)}(i)$. $D$ hands over $ICSig(D, P_i, S_i)$ to party $P_i$.
2. All the parties in $\mathcal{P}$ invoke RandomVector$(\mathcal{P}, 1)$ to generate a non-zero random value $z \in \mathbb{F}$.
3. $D$ broadcasts the polynomial $f(x) = f^{(0)}(x) + \sum_{l=1}^{\ell} f^{(l)}(x) z^l = \sum_{l=0}^{\ell} f^{(l)}(x) z^l$. Parallely, every party $P_i$ computes and broadcasts $y_i = r_i + \sum_{l=1}^{\ell} s^{(l)}_i z^l$.
4. If the polynomial $f(x)$ broadcasted by $D$ is of degree more than $t$, then each party agrees that $D$ is corrupted and outputs $t$-1D-sharing of $\ell$ 1's i.e $[1]_t, [1]_t, \ldots, [1]_t$. The protocol terminates here.
5. Every party checks whether $f(i) \overset{?}{=} y_i$ for all $i = 1, \ldots, n$. If yes then everybody accepts the $t$-1D-sharings $[s^{(1)}]_t, [s^{(2)}]_t, \ldots, [s^{(\ell)}]_t$ and the protocol terminates. Otherwise, let $P_i \in \mathcal{P}$, such that $f(i) \neq y_i$. In this case, $P_i$ reveals $ICSig(D, P_i, S_i)$. If $P_i$ succeeds to prove $D$'s signature on $S_i = (r_i, s^{(1)}_i, \ldots, s^{(\ell)}_i)$ and $f(i) \neq r_i + \sum_{l=1}^{\ell} s^{(l)}_i z^l$, then each party agrees that $D$ is corrupted and outputs $t$-1D-sharing of $\ell$ 1's i.e $[1]_t, [1]_t, \ldots, [1]_t$ ($\ell$ times) and the protocol terminates here. We say that $P_i$ has raised a valid **complaint** against $D$. But if the signature is invalid then ignore $P_i$'s complaint against $D$ and everybody accepts $[s^{(1)}]_t, \ldots, [s^{(\ell)}]_t$.

**Reconstruction of $t$-1D-Sharing**: We now present a protocol called ReconsPublic, that reconstruct a secret $s$, given $[s]_t$. In the protocol, every party broadcasts his share of $s$. Now out of these $n$ shares, at most $t$ could be corrupted. But since $n \geq 3t+1$, by applying Reed-Solomon error correction algorithm (e.g. Berlekamp Welch Algorithm [18]), $s$ can be recovered.

$s = \mathsf{ReconsPublic}(\mathcal{P}, [s]_t)$

Each party $P_i$ broadcasts his share $s_i$ of $s$. The parties apply error correction to reconstruct $s$ from the $n$ shares.

**Lemma 4** *ReconsPublic takes one round and broadcasts $\mathcal{O}(n\kappa)$ bits.*

**Important Notation:** We now define few notations which will be used heavily in the subsequent sections (these notations are also commonly used in the literature). By saying that the parties in $\mathcal{P}$ compute (locally) $([y^{(1)}]_d, \ldots, [y^{(\ell')}]_d) = \varphi([x^{(1)}]_d, \ldots, [x^{(\ell)}]_d)$ (for any function $\varphi : \mathbb{F}^\ell \to \mathbb{F}^{\ell'}$), we mean that each $P_i$ computes $(y_i^{(1)}, \ldots, y_i^{(\ell')}) = \varphi(x_i^{(1)}, \ldots, x_i^{(\ell)})$. Note that applying an affine (linear) function $\varphi$ to a number of $d$-1D-sharings, we get $d$-1D-sharings of the outputs. So by adding two $d$-1D-sharings of secrets, we get $d$-1D-sharing of the sum of the secrets, i.e. $[a]_d + [b]_d = [a+b]_d$. However, by multiplying two $d$-1D-sharings of secrets, we get $2d$-1D-sharing of the product of the secrets, i.e. $[a]_d[b]_d = [ab]_{2d}$. $\diamondsuit$

### 4.4 Upgrading $t$-1D-sharing to $t$-2D-sharing

**Definition 2** *We say that a value $s$ is correctly $d$-2D-shared among the parties in $\mathcal{P}$, if there exists degree-$d$ 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 (honest) party $P_i \in \mathcal{P}$ holds a share $s_i = f(i)$ of $s$, the polynomial $f^i(x)$ for sharing $s_i$ and share-share $s_{ji} = f^j(i)$ for the share $s_j$ of every other (honest) party $P_j$. We denote $d$-2D-sharing of $s$ as $[[s]]_d$.*

If a secret $s$ is $d$-2D-shared by a party $D \in \mathcal{P}$, then we denote it as $[[s]]_d^D$. Notice that if $s$ is $d$-2D-shared, then its $i^{th}$ share $s_i$ is $d$-1D-shared. We now present a new protocol, called Upgrade1Dto2D for upgrading $t$-1D-sharing to $t$-2D-sharing. Specifically, given $t$-1D-sharing of $\ell$ secrets, namely $[s^{(1)}]_t, \ldots, [s^{(\ell)}]_t$, Upgrade1Dto2D, outputs $t$-2D-sharing $[[s^{(1)}]]_t, [[s^{(2)}]]_t, \ldots, [[s^{(\ell)}]]_t$, except with probability of $2^{-\Omega(\kappa)}$. Moreover, $\mathcal{A}_t$ learns nothing about the secrets during Upgrade1Dto2D. Furthermore, if a party tries to cheat during the protocol, then with very high probability, he will be caught.

$([[s^{(1)}]]_t, [[s^{(2)}]]_t, \ldots, [[s^{(\ell)}]]_t) = \mathsf{Upgrade1Dto2D}(\mathcal{P}, \ell, [s^{(1)}]_t, [s^{(2)}]_t, \ldots, [s^{(\ell)}]_t)$

1. Each $P_i \in \mathcal{P}$ invokes $\mathsf{1DShare}(P_i, \mathcal{P}, 1, s^{(0,P_i)})$ to generate $[s^{(0,P_i)}]_t$, where $s^{(0,P_i)} \in \mathbb{F} \setminus \{0\}$ is a random value.
2. The parties in $\mathcal{P}$ computes $[s^{(0)}]_t = \sum_{j=1}^n [s^{(0,P_j)}]_t$.
3. Now every $P_i$ invokes $\mathsf{1DShare}(P_i, \mathcal{P}, \ell+1, s_i^{(0)}, s_i^{(1)}, \ldots, s_i^{(\ell)})$ to generate $[s_i^{(0)}]_t, [s_i^{(1)}]_t, \ldots, [s_i^{(\ell)}]_t$, where $s_i^{(0)}, s_i^{(1)}, \ldots, s_i^{(\ell)}$ are the $i^{th}$ shares of secrets $s^{(0)}, s^{(1)}, \ldots, s^{(\ell)}$ respectively.
4. Now to detect the parties $P_k$ (at most $t$), who have generated $[\overline{s_k^{(0)}}]_t, [\overline{s_k^{(1)}}]_t, \ldots, [\overline{s_k^{(\ell)}}]_t$ such that $\overline{s_k^{(l)}} \neq s_k^{(l)}$ for some $l \in \{0, 1, \ldots, \ell\}$, the parties in $\mathcal{P}$ jointly generate an $\ell$ length random vector $(r^{(1)}, \ldots, r^{(\ell)})$ by invoking Protocol $\mathsf{RandomVector}(\mathcal{P}, \ell)$. Now all the parties publicly reconstruct $s_i = s_i^{(0)} + \sum_{l=1}^{\ell} r^{(l)} s_i^{(l)}$ and $s = s^{(0)} + \sum_{l=1}^{\ell} r^{(l)} s^{(l)}$ by executing following steps:
   (a) The parties in $\mathcal{P}$ compute $[s_i]_t = [s_i^{(0)}]_t + \sum_{l=1}^{\ell} r^{(l)} [s_i^{(l)}]_t$ and invoke $\mathsf{ReconsPublic}(\mathcal{P}, [s_i]_t)$ to publicly reconstruct $s_i$, for $i = 1, \ldots, n$.
   (b) Every party apply Reed-Solomon error correction algorithm (e.g. Berlekamp Welch Algorithm [18]) to $s_1, s_2, \ldots, s_n$, to recover $s$. Reed-Solomon error correction algorithm also points out the corrupted shares. Hence if $s_i$ is pointed as a corrupted share, then $[s_i^{(0)}]_t, [s_i^{(1)}]_t, \ldots, [s_i^{(\ell)}]_t$ are ignored by every party.
5. Output $[[s^{(1)}]]_t, [[s^{(2)}]]_t, \ldots, [[s^{(\ell)}]]_t$.

**Lemma 5** *Protocol Upgrade1Dto2D upgrades $t$-1D-sharing of $\ell$ secrets to $t$-2D-sharing, except with negligible error probability. The protocol consumes twenty eight rounds, privately communicates $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits and broadcasts $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits. Moreover, $\mathcal{A}_t$ learns nothing about the secrets.*

PROOF: The communication and round complexity of the protocol is easy to follow. We now prove the correctness. Provided $\ell$ $t$-1D-sharing $[s^{(1)}]_t, [s^{(2)}]_t, \ldots, [s^{(\ell)}]_t$, every honest party $P_i$ correctly $t$-1D-shares his shares $s_i^{(0)}, s_i^{(1)}, \ldots, s_i^{(\ell)}$. Now for every honest party $P_h$, the value $s_h = s_h^{(0)} + \sum_{l=1}^{\ell} r^{(l)} s_h^{(l)}$ will be reconstructed correctly, where $s_h$ is the $h^{th}$ share of $s = s^{(0)} + \sum_{l=1}^{\ell} r^{(l)} s^{(l)}$. But a corrupted party $P_c$ may share $\overline{s_c^{(0)}}, \overline{s_c^{(1)}}, \ldots, \overline{s_c^{(\ell)}}$ with $\overline{s_c^{(l)}} \neq s_c^{(l)}$ for some $l \in \{0, 1, \ldots, \ell\}$. In this case with very high probability $\overline{s_c} = \overline{s_c^{(0)}} + \sum_{l=1}^{\ell} r^{(l)} \overline{s_c^{(l)}}$ will not be equal to $s_c$ (which is the actual $c^{th}$ share of

$s$) as the $\ell$ length vector $(r^{(1)}, \ldots, r^{(\ell)})$ is chosen uniformly at random. Hence Reed-Solomon Error correction algorithm will point $\bar{s}_c$ as a corrupted share, in which case $P_c$ will be caught. It is easy to see that at any stage of the protocol, $\mathcal{A}_t$ learns not more than $t$ shares for each $s^{(l)}, 1 \leq l \leq \ell$. Hence all the secrets will be secure. $\square$

**Remark 3 (Comparison with Existing Protocols)** *In [1], the authors have given a protocol to upgrade d-1D-Sharing to d-2D-Sharing, where $n = 2t + 1$. However, the protocol is non-robust. That is, if all the $n$ parties behave honestly, then the protocol will perform the upgradation. Otherwise, the protocol will fail to do the upgradation, but will output a pair of parties, of which at least one is corrupted. On the other hand, our upgradation protocol is designed with $n = 3t + 1$ and hence will always perform the upgradation successfully, irrespective of the behavior of the corrupted parties.*

## 4.5 Proving $c = ab$

Given $t$-1D-sharing of $\ell$ pairs, $([a^{(1)}]_t^D, [b^{(1)}]_t), \ldots, ([a^{(\ell)}]_t^D, [b^{(\ell)}]_t^D)$, let $c^{(l)} = a^{(l)}b^{(l)}$ for $l = 1, \ldots, \ell$. $D \in \mathcal{P}$ now wants to generate $[c^{(1)}]_t^D, \ldots, [c^{(\ell)}]_t^D$ such that the (honest) parties in $\mathcal{P}$ know that the shared $c^l$ values satisfy $c^{(l)} = a^{(l)}b^{(l)}$ for $l = 1, \ldots, \ell$. If $D$ is honest, then during this process all $a^{(l)}$, $b^{(l)}$ and $c^{(l)}$ values should remain secure.

We propose a protocol ProveCeqAB to achieve the above task. The idea of the protocol is inspired from [8], where a protocol for the same purpose is proposed, with *a single* pair of values, namely $(a, b)$. Our protocol concurrently deals with $\ell$ pairs, which leads to a gain in communication complexity. Our protocol uses 1DShare as a building block.

We try to explain the idea of the protocol with a single pair $(a, b)$. According to the problem definition, $D$ has generated $[a]_t^D$ and $[b]_t^D$. Now he wants to generate $[c]_t^D$, 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 $[c]_t^D, [\beta]_t^D$ and $[\beta b]_t^D$. Now all the parties in $\mathcal{P}$ jointly generate a random value $r$ and compute $[Y]_t = r[a]_t^D + [\beta]_t^D$. $D$ then broadcasts the value $\Lambda = ra + \beta$, while the parties publicly reconstruct $Y$. Everybody then verifies whether $Y$ is same as $\Lambda$. If $D$ has correctly generated $[c]_t^D, [\beta]_t^D$ and $[\beta b]_t^D$, then $Y = \Lambda$ will hold. Otherwise all the parties will conclude that $D$ is corrupted and he fails to prove '$c = ab$'. However, if $Y = \Lambda$, then the parties proceed further and compute $[X]_t = Y[b]_t^D - [\beta b]_t^D - r[c]_t^D$. The parties then publicly reconstruct $X$. Now again if $D$ has correctly generated $[c]_t^D, [\beta]_t^D$ and $[\beta b]_t^D$, then $X = Yb - \beta b - rc = 0$ will hold. So after reconstructing $X$, every body checks whether $X = 0$. If $X = 0$ then everybody accepts $[c]_t^D$ as valid $t$-1D-sharing of $ab$. Otherwise all the parties will conclude that $D$ is corrupted and he fails to prove '$c = ab$'.

The error probability of the protocol is negligible because of the random $r$, generated by all the parties jointly. Specifically, though a corrupted $D$ may share $\overline{\beta b} \neq \beta b$ or $\overline{c} \neq c$, $X$ can be zero when $\overline{\beta b} + r\overline{c} = \beta b + rc$. However this equality is satisfied for only one value of $r$. Since $r$ is randomly generated, independent of the sharings done by $D$, the probability that the equality will hold is $\frac{1}{|\mathbb{F}|}$ which is negligibly small. The secrecy follows from the fact that the broadcasted value $\Lambda$ is randomly distributed. Now we can extend the above idea for the $\ell$ pairs $(a^{(l)}, b^{(l)})$ concurrently. As the protocol is based on existing idea of [8], we present it in **APPENDIX C**.

**Lemma 6** *In protocol ProveCeqAB, if $D$ does not fail, then $(a^{(l)}, b^{(l)})$, $c^{(l)}$ satisfies $c^{(l)} = a^{(l)}b^{(l)}$ for $l = 1, \ldots, \ell$, except with error negligible probability. ProveCeqAB takes eighteen rounds, privately communicates $\mathcal{O}((\ell n + n^2\kappa)\kappa)$ bits and broadcasts $\mathcal{O}((\ell n + n^2\kappa)\kappa)$ bits. Moreover, if $D$ is honest then $\mathcal{A}_t$ learns no information about $a^{(l)}, b^{(l)}$ and $c^{(l)}$, for $1 \leq l \leq \ell$.*

## 4.6 Multiplication

Given $t$-1D-sharing of $\ell$ pairs of secrets, say $([a^{(1)}]_t, [b^{(1)}]_t), \ldots, ([a^{(\ell)}]_t, [b^{(\ell)}]_t)$, we now present a protocol called Mult which allows the parties to compute $t$-1D-sharing $[c^{(1)}]_t, \ldots, [c^{(\ell)}]_t$ such that $c^{(l)} = a^{(l)}b^{(l)}$ for $l = 1, \ldots, \ell$. Our protocol is motivated from the protocol of [8], which deals with *a single* pair of $t$-1D-sharing. However, our protocol concurrently deals with $\ell$ pairs of $t$-1D-sharing. This leads to a gain in communication complexity.

We explain the idea considering a single pair, say $(a, b)$. Given $([a]_t, [b]_t)$, the parties want to compute $[c]_t$. For that the parties first compute $[c]_{2t} = [a]_t[b]_t$. So every party $P_i$ now has $a_ib_i$, where $a_i$ and $b_i$ are the $i^{th}$ shares of $a$ and $b$. Now every party $P_i$ generates $[a_ib_i]_t^{P_i}$ by executing ProveCeqAB (though the corrupted parties may fail to generate $[a_ib_i]_t^{P_i}$). Since $a_1b_1, \ldots, a_nb_n$ are $n$ points on a $2t$ degree polynomial, say $C(x)$, whose constant term is $c$, by Lagrange interpolation formula [7], $c$ can be computed as $c = \sum_{i=1}^{n} r_i(a_ib_i)$ where $r_i = \prod_{j=1, j\neq i}^{n} \frac{-j}{i-j}$. The vector $(r_1, \ldots, r_n)$ is called recombination vector [7] which is public and known to every party. So for shorthand notation, we write $c = Lagrange(a_1b_1, \ldots, a_nb_n) = \sum_{i=1}^{n} r_i(a_ib_i)$. Now all parties compute $[c]_t = Lagrange([a_1b_1]_t, \ldots, [a_nb_n]_t) = \sum_{i=1}^{n} r_i[a_ib_i]_t$, to obtain the desired output. Notice that since $C(x)$ is of degree $2t$, we need $2t + 1$ parties to successfully generate $a_ib_i$ value (a $2t$ degree polynomial requires $2t + 1$ points on it to be interpolated correctly). So, even if $t$ corrupted parties fail to generate $[a_ib_i]_t$, our protocol will work. Our protocol Mult follows the above technique for $\ell$ pairs simultaneously. The protocol is given in **APPENDIX C**.

**Lemma 7** *Except with negligible error probability, Mult produces $[c^{(1)}]_t, \ldots, c^{(\ell)}]_t$ from $\ell$ pairs $([a^{(1)}]_t, [b^{(1)}]_t), \ldots, ([a^{(\ell)}]_t, [b^{(\ell)}]_t)$. The protocol takes 46 rounds, privately communicates $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits and broadcasts $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits. Moreover, $\mathcal{A}_t$ learns nothing about $c^{(l)}, a^{(l)}$ and $b^{(l)}$, for $1 \leq l \leq \ell$.*

## 4.7 Generating Random $t$-1D-Sharing

We now present a protocol called Random$(\mathcal{P}, \ell)$, which allows the parties in $\mathcal{P}$ to jointly generate $\ell$ random $t$-1D-sharings, $[r^{(1)}]_t, \ldots, [r^{(\ell)}]_t$, where each $r^{(i)}$ is a random element in $\mathbb{F}$.

Random$(\mathcal{P}, \ell)$

Every party $P_i \in \mathcal{P}$ invokes 1DShare$(P_i, \mathcal{P}, \ell, r^{(1,P_i)}, \ldots, r^{(\ell,P_i)})$ to verifiably $t$-1D-share $\ell$ random values $r^{(1,P_i)}, \ldots, r^{(\ell,P_i)}$ from $\mathbb{F}$. Now all the parties in $\mathcal{P}$ jointly computes $[r^{(l)}]_t = \sum_{i=1}^{n} [r^{(l,P_i)}]_t$ for $l = 1, \ldots, \ell$

**Lemma 8** *With overwhelming probability, Random generates $\ell$ random t-1D-sharing $[r^{(1)}]_t, \ldots, [r^{(\ell)}]_t$ in 11 rounds, by privately communicating $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits and broadcasting $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits.*

## 5 Unconditionally Secure MPSI Protocol with $n = 3t + 1$

We now present our unconditionally secure MPSI protocol with $n = 3t + 1$. We first present the protocol for Input and Preparation Phase, where $t$-1D-sharing of the coefficients of $r^{(i)}(x)$ and $f^{(i)}(x)$ polynomials are generated.

Input and Preparation Phase

1. Every $P_i \in \mathcal{P}$ represents his set $S_i = \{e_i^{(1)}, e_i^{(2)}, \ldots, e_i^{(m)}\}$ by a polynomial $f^{(P_i)}(x)$ of degree $m$ such that $f^{(P_i)}(x) = (x - e_i^{(1)}) \ldots (x - e_i^{(m)}) = a^{(0,P_i)} + a^{(1,P_i)}x + \ldots + a^{(m,P_i)}x^m$. $P_i$ then invokes 1DShare$(P_i, \mathcal{P}, m, a^{(0,P_i)}, \ldots, a^{(m-1,P_i)})$ to generate $[a^{(0,P_i)}]_t, \ldots, [a^{(m-1,P_i)}]_t$. Since $a^{(m,P_i)} = 1$ always, every party in $\mathcal{P}$ assumes a predefined $t$-1D-sharing for 1, namely $[1]_t$ on behalf of $a^{(m,P_i)}$ (see Remark 4).
2. The parties in $\mathcal{P}$ invoke $n$ times Random$(\mathcal{P}, m+1)$ parallely, where $i^{th}$ invocation of Random$(\mathcal{P}, m+1)$ generates $m + 1$ $t$-1D-sharings $[b^{(0,i)}]_t, \ldots, [b^{(m,i)}]_t$. Now the parties assume that $r^{(i)}(x) = b^{(0,i)} + b^{(1,i)}x + \ldots + b^{(m,i)}x^m$ for $i = 1, \ldots, n$. This step can be executed parallely with step 1.

**Remark 4** *In any MPSI protocol that computes the intersection of the sets of the parties using the function given in (1), $\mathcal{A}_t$ may disrupt the security of the protocol by forcing a corrupted party to input a zero polynomial representing his set. This is because $\mathcal{A}_t$ will come to know the intersection of the sets of the remaining parties at the end of computation of the protocol [16, 15]. So to stop a corrupted party to input a zero polynomial, the authors of [16, 15] specified the following trick. They have noticed that the coefficient of $m^{th}$ degree term in every $P_j$'s polynomial $f^{(P_j)}(x) = \prod_{k=1}^{m}(x - e_j^{(k)})$ is 1 always. Hence, every party assumes a predefined $[1]_t$ on behalf of the $m^{th}$ coefficient of every parties $f^{(P_j)}(x)$ polynomial (instead of allowing individual parties to t-1D-share the $m^{th}$ coefficient). This stops the corrupted parties to commit a zero polynomial.*

**Theorem 1** *Input and Preparation phase takes 11 rounds (step 1 and step 2 can be executed parallelly), privately communicates $\mathcal{O}((mn^3 + n^4\kappa)\kappa)$ bits and broadcasts $\mathcal{O}((mn^3 + n^4\kappa)\kappa)$ bits.*

After input and preparation phase, the parties jointly compute the coefficients of $F(x) = \sum_{i=1}^{n} r^{(i)}(x)f^{(P_i)}(x)$ in a shared manner in Computation Phase. In Output Phase, the coefficients $F(x)$ are publicly reconstructed. Then each party locally evaluates $F(x)$ at each element of his private set. All the elements at which $F(x) = 0$ belongs to the intersection of the $n$ sets, with very high probability.

**Theorem 2** *Computation and Output phase takes 47 rounds, privately communicates $\mathcal{O}((m^2n^3 + n^4\kappa)\kappa$ bits and broadcasts $\mathcal{O}((m^2n^3 + n^4\kappa)\kappa$ bits.*

**Theorem 3** *MPSI protocol with $3t+1$ takes 58 rounds, privately communicates $\mathcal{O}((m^2n^3+n^4\kappa)\kappa$ bits and broadcasts $\mathcal{O}((m^2n^3 + n^4\kappa)\kappa$ bits, when physical broadcast channel is available in the system. In the absence of a physical broadcast channel, the protocol takes $\mathcal{O}(t)$ rounds and privately communicates $\mathcal{O}((m^2n^5 + n^6\kappa)\kappa$ bits. The protocol correctly solves secure MPSI problem with very high probability. Moreover, $\mathcal{A}_t$ will not get any extra information, other than what can be inferred by the data sets of the corrupted parties and the intersection of the data sets of all the parties.*

PROOF: The round complexity and communication complexity follows easily from the protocol. The security follows from the security of 1DShare and Mult [7]. The correctness follows from the protocol steps and the argument given in Remark 1. □

Computation Phase

1. Let $F^{(i)}(x) = r^{(i)}(x)f^{(P_i)}(x) = c^{(0,i)} + c^{(1,i)}x + \ldots + c^{(2m,i)}x^{2m}$ for $i = 1, \ldots, n$. For $i = 1, \ldots, n$, to generate $[c^{(0,i)}]_t, \ldots, [c^{(2m,i)}]_t$, the parties in $\mathcal{P}$ do the following:
   (a) The parties invoke Mult($\mathcal{P}, (m+1)^2, ([a^{(0,i)}]_t, [b^{(0,i)}]_t), ([a^{(0,i)}]_t, [b^{(1,i)}]_t), \ldots, ([a^{(m,i)}]_t, [b^{(m-1,i)}]_t), ([a^{(m,i)}]_t, [b^{(m,i)}]_t)$) with $(m+1)^2$ pairs (every coefficient of $r^{(i)}(x)$ should be multiplied with every coefficient of $f^{(P_i)}(x)$) to produce $(m+1)^2$ $t$-1D-sharings $[a^{(0,i)}b^{(0,i)}]_t, [a^{(0,i)}b^{(1,i)}]_t, \ldots, [a^{(m,i)}b^{(m-1,i)}]_t, [a^{(m,i)}b^{(m,i)}]_t$.
   (b) The parties compute $[c^{(0,i)}]_t = [a^{(0)}b^{(0,i)}]_t$, $[c^{(1,i)}]_t = [a^{(0,i)}b^{(1,i)}]_t + [a^{(1,i)}b^{(0,i)}]_t, \ldots, [c^{(2m,i)}]_t = [a^{(m,i)}b^{(m,i)}]_t$.
2. Let $F(x) = \sum_{i=1}^{n} F^{(i)}(x) = d^{(0)} + d^{(1)}x + \ldots + d^{(2m)}x^{2m}$. To generate $[d^{(0)}]_t, \ldots, [d^{(2m)}]_t$, the parties compute $[d^{(j)}]_t = \sum_{i=1}^{n}[c^{(j,i)}]_t$ for $j = 0, \ldots, 2m$.

Output Phase

1. The parties invoke ReconsPublic($\mathcal{P}, [d^{(j)}]_t$) to publicly reconstruct $d^{(j)}$ for $j = 0, \ldots, 2m$. Thus now parties have reconstructed $F(x)$.
2. Each $P_i$ with his private set $S_i = \{e_i^{(1)}, \ldots, e_i^{(m)}\}$ locally checks whether $F(e_i^{(l)}) \overset{?}{=} 0$ for $l = 1, \ldots, m$. If $F(e_i^{(l)}) = 0$, the $P_i$ adds $e_i^{(l)}$ in a set $IS_i$ (initially $IS_i = \emptyset$). $P_i$ outputs $IS_i$ as the intersection set $S_1 \cap S_2 \ldots, \cap S_n$.

## 6 Conclusion and Open Problems

In this paper, we have shown that the round complexity and communication complexity of the information theoretically secure MPSI protocol of [16] is much more than what is claimed in [16]. We then presented a new information theoretically secure MPSI protocol with $n = 3t + 1$, which significantly improves the actual round complexity and communication complexity of the MPSI protocol of [16]. Towards this, we have designed new sub-protocols, namely Multi-Secret-Multi-Receiver-ICP, 1DShare and Upgrade1Dto2D, which along with existing techniques from the literature, led to our efficient MPSI protocol. It would be interesting to improve the resilience of the MPSI protocol of [16] and this paper, by designing an information theoretically secure MPSI protocol with optimal resilience (i.e., with $n = 2t + 1$). Improving the round complexity and communication complexity of information theoretically secure MPSI protocol is another interesting problem.

[7]The security of the MPSI protocol can be proved in UC framework [5].

## References

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

[2] 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.

[3] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proc. of 20th ACM STOC*, pages 1–10, 1988.

[4] 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.

[5] Ran Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000.

[6] 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.

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

[8] 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.

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

[10] M. Fitzi, J. Garay, S. Gollakota, C. Pandu Rangan, and K. Srinathan. Round-optimal and efficient verifiable secret sharing. In *Proc. of TCC 2006*, volume 3876 of *LNCS*, pages 329–342. Springer Verlag, 2006.

[11] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Proc. of EURORYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer Verlag, 2004.

[12] 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.

[13] 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.

[14] J. Katz, C. Yuen Koo, and R. Kumaresan. Improving the round complexity of vss in point-to-point networks. In *Proc. of ICALP (2)*, pages 499–510, 2008.

[15] L. Kissner and D. Song. Privacy preserving set operations. In *Proc. of CRYPTO 2005*, volume 3621 of *LNCS*, pages 241–257. Springer Verlag, 2005.

[16] R. Li and C. Wu. An unconditionally secure protocol for multi-party set intersection. In *Proc. of ACNS 2007*, volume 4521 of *LNCS*, pages 222–236. Springer Verlag, 2007.

[17] N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann, 1996.

[18] F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error Correcting Codes.* North-Holland Publishing Company, 1978.

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

[20] K. Srinathan, A. Narayanan, and C. Pandu Rangan. Optimal perfectly secure message transmission. In *Proc. of Advances in Cryptology: CRYPTO 2004*, LNCS 3152, pages 545–561. Springer-Verlag, 2004.

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

## APPENDIX A: Properties of Protocol Multi-Secret-Multi-Receiver-ICP

**Lemma 9 (Property 1)** *If $D$ and $INT$ are honest, then each honest verifier will accept $S$ at the end of RevealVal, without any error.*

PROOF: If $D$ and $INT$ are honest, then all the honest verifiers (at least $2t+1$) will *accept* $INT$ during AuthVal. The same honest verifiers will *re-accept* $INT$ during RevealVal. Hence $S$ will be accepted by each honest verifier. □

**Lemma 10 (Property 2)** *If $INT$ is uncorrupted, then at the end of AuthVal, $INT$ knows an $S$, which will be accepted in RevealVal by each honest verifier, except with probability $2^{-\Omega(\kappa)}$.*

PROOF: If $D$ is honest, then the proof follows from Lemma 9. So we consider the case when $D$ is corrupted. Here also, there are two possible sub-cases. If $D$ broadcasts $F(x)$ during **Round 3**, then the lemma holds trivially, without any error. So we consider the case, when $D$ (corrupted) has not broadcasted $F(x)$ during **Round 3**. This implies that at least $2t+1$ verifiers have *accepted* $INT$ during AuthVal. Now, out of these $2t+1$ verifiers, at least $t+1$ are honest. If we can show that these honest verifiers will *re-accept* $INT$ during RevealVal with high probability, then the proof is over. So we now proceed to prove the same.

In order that an honest $P_i$ *accept* $INT$ during AuthVal but does not *re-accept* it during RevealVal, it must be the case that the data (evaluation points and values) that $P_i$ exposed during AuthVal satisfies the polynomial $B(x)$ that $INT$ broadcasted during AuthVal, but on the other hand, out of the remaining evaluation points that are used by $P_i$ in RevealVal, none satisfy the polynomial $F(x)$ produced by $INT$. That is, for the selected $\frac{\kappa}{2}$ indices $l_1, ..., l_{\frac{\kappa}{2}}$, it holds that $a_{i,l} + db_{i,l} = B(\alpha_{i,l})$, for all $l$ in the set of indices $\{l_1, ..., l_{\frac{\kappa}{2}}\}$ and $F(\alpha_{i,l}) \neq a_{i,l}$ for all $l$ in the *remaining* set of indices. Notice that $INT$ chooses $d$ independently of the values given by $D$. Also, $P_i$ chooses the $\frac{\kappa}{2}$ indices randomly out of $\kappa$ indices. So the probability that the above event happens is $\frac{1}{\binom{\kappa}{\kappa/2}} \approx 2^{-\Omega(\kappa)}$, which is negligible. This shows that with high probability all honest verifiers (at least $t+1$), who have *accepted* $INT$ during AuthVal, will *re-accept* $INT$ during RevealVal, thus proving our lemma. □

**Lemma 11 (Property 3)** *If $D$ is uncorrupted, then during RevealVal, with probability at least $1 - 2^{-\Omega(\kappa)}$, every $S' \neq S$ produced by a corrupted $INT$ will be rejected by each honest verifier.*

PROOF: If a corrupted $INT$ produces $S' \neq S$ during RevealVal, then it implies that $D$ has broadcasted $F'(x) \neq F(x)$ during **Round 4**. Moreover, while broadcasting $F'(x)$, $INT$ will have no information about the $\frac{\kappa}{2}$ random secret evaluation points (which were not broadcasted during AuthVal), corresponding to each honest verifer. We now claim that with very high probability, none of the honest verifier will *re-accept* $INT$ (i.e., $F'(x)$), at the end of **Round 5**. The polynomial $F'(x)$ can agree with $F(x)$ in at most $\ell + n\kappa$ evaluation points. Without knowing the $\frac{\kappa}{2}$ secret evaluation points of an honest verifier, say $P_i$, the probability that $INT$ will be re-accepted by $P_i$ is at most $\frac{\ell+nk}{|\mathbb{F}|}$. Thus, the total probability that any honest verifier will accept $INT$ (who broadcasts $F'(x) \neq F(x)$) is $\frac{(\ell+nk)(2t+1)}{|\mathbb{F}|} \approx 2^{-\Omega(k)}$. Hence with very high probability, none of the honest verifiers will *re-accept* a corrupted $INT$. Moreover, there can be at most $t$ corrupted verifiers, who will *re-accept* a corrupted $INT$. These two facts together implies that each honest verifier will reject $S' \neq S$ with very high probability. □

**Lemma 12 (Property 4)** *If $D$ and $INT$ are honest, then $\mathcal{A}_t$ will have no information about $S$ at the end of AuthVal.*

Proof: For simplicity, assume that first $t$ verifiers are corrupted. So in the **Round 1**, the adversary will know $\kappa t$ points on $F(x)$ and $R(x)$. In **Round 2**, the adversary will come to know about additional $\frac{k}{2}(2t+1)$ points on $F(x)$ and $R(x)$. Moreover, since $D$ and $INT$ are both honest, $2t+1$ honest verifiers will accept $INT$ and hence $D$ will not broadcast $F(x)$ during **Round 3**. So at the end of AuthVal, adversary will know $\kappa t + \frac{\kappa}{2}(2t+1)$ points on each of $F(x)$ and $R(x)$. However, since $n = 3t+1$ and degree of $F(x)$ and $R(x)$ is $\ell + n\kappa$, the adversary will have no information about the lower order $\ell$ coefficients of $F(x)$, which further implies information theoretic security for $S$ at the end of AuthVal. $\square$

# APPENDIX B

**Lemma** 3: *In protocol 1DShare, if $D$ is honest, then $t$-1D-sharing of $s^{(1)}, \ldots, s^{(\ell)}$ are generated, except with error probability of $2^{-\Omega(\kappa)}$. Moreover, $\mathcal{A}_t$ will have no information about the secrets. On the other hand, if $D$ is corrupted and any of the values $r, s^{(1)}, \ldots, s^{(\ell)}$ is not $t$-1D-shared, then $D$ will be caught, except with error probability of $2^{-\Omega(\kappa)}$. The protocol takes eleven rounds, privately communicates $\mathcal{O}((\ell n + n^2\kappa)\kappa)$ bits and broadcasts $\mathcal{O}((\ell n + n^2\kappa)\kappa)$ bits.*

Proof: The communication complexity and number of rounds can be checked easily by inspection. We now prove the correctness. If $D$ is honest, then all the honest parties will correctly verify the $t$-1D-sharing of $\ell$ secrets. Specifically, $f(i) = y_i$ will hold, corresponding to every honest $P_i$. However, a corrupted party $P_i$ may broadcast incorrect $y_i' \neq y_i$ , such that $y_i' \neq f(i)$. Moreover, $P_i$ can forge honest $D$'s IC signature on the corresponding incorrect $r_i' \neq r_i$ or/and $s_i^{'(j)} \neq s_i^{(j)}$, for $j = 1 \ldots \ell$. In this case, everybody will reject the sharing done by $D$. However, from the properties of Multi-Secret-Multi-Receiver-ICP protocol, the above event can happen with probability $2^{-\Omega(\kappa)}$. The secrecy of the secrets $s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}$ for an honest $D$, follows from the fact that $\mathcal{A}_t$ will have only $t$ shares for each $s^{(i)}, 1 \leq i \leq n$ and random $r$. In addition, the value $f(0)$ is blinded with a random value $r$, chosen by $D$. Thus, $\mathcal{A}_t$ will have no information about the secrets.

Next, we consider the case, when $D$ is corrupted and the sharing of at least one of the values $r, s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}$ is not a correct $t$-1D-sharing, i.e., the shares of the honest parties lie on a polynomial of degree higher than $t$. Let $H$ be the set of honest parties in $\mathcal{P}$. Moreover, let $h^0(x), \ldots, h^\ell(x)$ denote the minimum degree polynomial, defined by the points on $f^{(0)}(x), \ldots, f^{(\ell)}(x)$ respectively, held by the parties in $H$. Then according to the condition, degree of at least one of the polynomials $h^0(x), \ldots, h^\ell(x)$ is more than $t$. Moreover, degree of $h^{(0)}(x), \ldots, h^{(\ell)}(x)$ can be at most $|H| - 1$. This is because $|H|$ distinct points can define a polynomial of degree at most $|H| - 1$. Now the value $y_i$ broadcasted by an honest $P_i$ can be defined as $y_i = \sum_{j=0}^{\ell} z^j h^j(i)$.

We next claim that if degree of at least one of the polynomials $h^0(x), \ldots, h^\ell(x)$ is more than $t$, then the minimum degree polynomial, say $h^{min}(x)$, defined by $y_i$'s, corresponding to $P_i \in H$ will be of degree more than $t$, with very high probability. This will clearly imply that $f(x) \neq h^{min}(x)$ and hence $y_i \neq f(i)$, for at least one $P_i \in H$.

So we proceed to prove that $h^{min}(x)$ will be of degree more than $t$ with very high probability, when one of $h^0(x), \ldots, h^\ell(x)$ has degree more than $t$. For this, we show the following:

1. We first show that $h^{def}(x) = \Sigma_{j=0}^{\ell} z^j h^j(x)$ will of degree more than $t$ with very high probability, if one of $h^0(x), \ldots, h^\ell(x)$ has degree more than $t$.
2. We then show that $h^{min}(x) = h^{def}(x)$, implying that $h^{min}(x)$ will be of degree more than $t$ with very high probability

To prove the first point, assume that at least one of $h^0(x), \ldots, h^\ell(x)$, has degree more than $t$. Let $m$ be such that $h^m(x)$ has maximal degree among $h^0(x), \ldots, h^\ell(x)$, and let $t_m$ be the degree of $h^m(x)$. Then according to the condition, $t_m > t$. Also recall that $t_m < |H|$. This is because given $|H|$ values (recall that $h^0(x), \ldots, h^\ell(x)$ are defined by the points on polynomials $f^{(0)}(x), \ldots, f^{(\ell)}(x)$, held by the honest parties in $H$), the maximum degree polynomial that can be defined using them is $|H|-1$. Now each polynomial $h^i(x)$ can be written as $h^i(x) = c^i_{t_m} x^{t_m} + \widehat{h^i(x)}$ where $\widehat{h^i(x)}$ has degree lower than $t_m$. Thus the polynomial $h^{def}(x)$ can be written as:

$$\begin{aligned} h^{def}(x) &= [c^0_{t_m} x^{t_m} + \widehat{h^0(x)}] + z[c^1_{t_m} x^{t_m} + \widehat{h^1(x)}] + \ldots + z^\ell[c^\ell_{t_m} x^{t_m} + \widehat{h^\ell(x)}] \\ &= x^{t_m}(c^0_{t_m} + \ldots + z^\ell c^\ell_{t_m}) + \Sigma_{j=0}^{\ell} z^j \widehat{h^j(x)} \\ &= x^{t_m} c_{t_m} + \Sigma_{j=0}^{n} z^j \widehat{h^j(x)} \end{aligned}$$

By assumption $c^m_{t_m} \neq 0$. It implies that the vector $(c^0_{t_m}, \ldots, c^\ell_{t_m})$ is not a complete 0 vector. Hence $c_{t_m} = c^0_{t_m} + \ldots + z^\ell c^\ell_{t_m}$ will be zero with probability $\frac{\ell}{|\mathbb{F}|-1} \approx 2^{-\Omega(\kappa)}$ (which is negligible). This is because the vector $(c^0_{t_m}, \ldots, c^\ell_{t_m})$ may be considered as the set of coefficients of a degree-$\ell$ polynomial, say $\mu(x)$, and hence the value $c_{t_m}$ is the value of $\mu(x)$ evaluated at $x = z$. Now $c_{t_m}$ will be zero if $z$ happens to be one of the $\ell$ roots of $\mu(x)$ (since degree of $\mu(x)$ is at most $\ell$). Now since $z$ is generated randomly from $\mathbb{F} \setminus \{0\}$, independent of the polynomials $h^0(y), \ldots, h^\ell(x)$, the probability that it is a root of $\mu(x)$ is $\frac{\ell}{|\mathbb{F}|-1} \approx 2^{-\Omega(\kappa)}$. So with very high probability $c_{t_m}$, which is the $t_m^{th}$ coefficient of $h^{def}(x)$ is non-zero. This implies that $h^{def}(x)$ will be of degree at least $t_m > t$. Notice that each $y_i$ broadcasted by an honest $P_i$, will lie on $h^{def}(x)$.

Now we will show that $h^{min}(x) = h^{def}(x)$ and thus $h^{min}(x)$ has degree at least $t_m$, which is greater than $t$. So consider the difference polynomial $dp(x) = h^{def}(x) - h^{min}(x)$. Clearly, $dp(x) = 0$, for all $x = i$, where $P_i \in H$. Thus $dp(x)$ will have at least $|H|$ roots. On the other hand, maximum degree of $dp(x)$ could be $t_m$, which is at most $|H| - 1$. These two facts together imply that $dp(x)$ is the zero polynomial, implying that $h^{def}(x) = h^{min}(x)$ and thus $h^{min}(x)$ has degree $t_m > t$.

Since $h^{min}(x)$ has degree more than $t$, it implies that $h^{min}(x) \neq f(x)$ (which is of degree-$t$ and broadcasted by $D$). This further imply that $f(i) \neq y_i$, for at least one $P_i \in H$. So $P_i$ will raise a valid **complaint** against $D$ by revealing $ICSig(D, P_i, S_i)$, where $S_i = (r_i, s_i^{(1)}, \ldots, s_i^{(\ell)})$. Since $P_i$ is honest, the signature will be revealed successfully, except with an error probability of $2^{-\Omega(\kappa)}$ (this follows from the properties of Multi-Secret-Multi-Receiver-ICP). Moreover, everybody will publicly verify that $f(i) \neq r_i + \sum_{l=1}^{\ell} s_i^{(l)} z^l$ and hence will catch the corrupted $D$ with very high probability. □

## APPENDIX C

$([c^{(1)}]^D_t, \ldots, [c^{(\ell)}]^D_t) =$ ProveCeqAB($D, \mathcal{P}, \ell, [a^{(1)}]^D_t, [b^{(1)}]^D_t, \ldots, [a^{(\ell)}]^D_t, [b^{(\ell)}]^D_t$)

1. $D$ chooses a random non-zero $\ell$ length tuple $(\beta^{(1)}, \ldots, \beta^{(\ell)}) \in \mathbb{F}^\ell$. $D$ then invokes 1DShare($D, \mathcal{P}, \ell, c^{(1)}, \ldots, c^{(\ell)}$), 1DShare($D, \mathcal{P}, \ell, \beta^{(1)}, \ldots, \beta^{(\ell)}$) and 1DShare($D, \mathcal{P}, \ell, b^{(1)}\beta^{(1)}, \ldots, b^{(\ell)}\beta^{(\ell)}$) to verifiably $t$-1D-share $(c^{(1)}, \ldots, c^{(\ell)})$, $(\beta^{(1)}, \ldots, \beta^{(\ell)})$ and $(b^{(1)}\beta^{(1)}, \ldots, b^{(\ell)}\beta^{(\ell)})$ respectively. If in any of these 1DShare protocol, $D$ is found to be corrupted, then every party conclude that $D$ fails in this protocol and hence this protocol terminates.
2. Now all the parties in $\mathcal{P}$ invoke RandomVector($\mathcal{P}, 1$) to generate a random value $r \in \mathbb{F}$.
3. For every $l \in \{1, \ldots, \ell\}$, all parties locally compute $[Y^{(l)}]_t = (r[a^{(l)}]_t + [\beta^{(l)}]_t)$ and invoke ReconsPublic($\mathcal{P}, [Y^{(l)}]_t$) to reconstruct $Y^{(l)}$. Parallely, $D$ broadcasts the values $Z^{(1)} = (ra^{(1)} + \beta^{(1)}), \ldots, Z^{(\ell)} = (ra^{(\ell)} + \beta^{(\ell)})$. All the parties check whether $Z^{(l)} \stackrel{?}{=} Y^{(l)}$. If not then every party concludes that $D$ fails in this protocol and hence the protocol terminates.
4. For every $l \in \{1, \ldots, \ell\}$, the parties locally compute $[X^{(l)}]_t = \left(Y^{(l)}[b^{(l)}]_t - [b^{(l)}\beta^{(l)}]_t - r[c^{(l)}]_t\right)$ and invoke ReconsPublic($\mathcal{P}, [X^{(l)}]_t$) to reconstruct $X^{(l)}$. The parties then check $X^{(l)} \stackrel{?}{=} 0$. If not then every party concludes that $D$ fails in this protocol and hence the protocol terminates. Otherwise $D$ has proved that $c^{(l)} = a^{(l)}b^{(l)}$.

Table 1: A Eighteen Round Protocol to Generate $t$-1D-Sharing of $c^{(l)}$ Where $c^{(l)} = a^{(l)}b^{(l)}$

$([c^{(1)}]_t, \ldots, [c^{(\ell)}]_t) = \mathsf{Mult}(\mathcal{P}, \ell, ([a^{(1)}]_t, [b^{(1)}]_t), \ldots, ([a^{(\ell)}]_t, [b^{(\ell)}]_t))$

1. All the parties invoke $\mathsf{Upgrade1Dto2D}(\mathcal{P}, \ell, [a^{(1)}]_t, \ldots, [a^{(\ell)}]_t)$ and $\mathsf{Upgrade1Dto2D}(\mathcal{P}, \ell, [b^{(1)}]_t, \ldots, [b^{(\ell)}]_t)$ to upgrade $t$-1D-sharings of $2\ell$ values to $t$-2D-sharings, i.e., to generate $[[a^{(1)}]]_t, \ldots, [[a^{(\ell)}]]_t$ and $[[b^{(1)}]]_t, \ldots, [[b^{(\ell)}]]_t$ respectively.
2. Let $(a_1^{(l)}, \ldots, a_n^{(l)})$ and $(b_1^{(l)}, \ldots, b_n^{(l)})$ denote the 1D sharings of $a^{(l)}$ and $b^{(l)}$ respectively. Since $a^{(l)}$ and $b^{(l)}$ is $t$-2D-shared, their $i^{th}$ shares $a_i^{(l)}$ and $b_i^{(l)}$ are $t$-1D-shared (see the definition of $t$-2D-sharing). The parties in $\mathcal{P}$ locally computes $[c^{(l)}]_{2t} = [a^{(l)}]_t[b^{(l)}]_t$ for $l = 1, \ldots, \ell$ where $[c^{(l)}]_{2t} = (a_1^{(l)}b_1^{(l)}, \ldots, a_n^{(l)}b_n^{(l)})$.
3. Each party $P_i$ has in his possession $i^{th}$ share of $[c^{(l)}]_{2t}$ i.e. $a_i^{(l)}b_i^{(l)}$ for $l = 1, \ldots, \ell$ where both $a_i^{(l)}$ and $b_i^{(l)}$ are already $t$-1D-shared by $P_i$ during Protocol $\mathsf{Upgrade1Dto2D}$ executed in step 1 of this protocol. Now each party $P_i$ invokes $\mathsf{ProveCeqAB}(P_i, \mathcal{P}, \ell, [a_i^{(1)}]_t^{P_i}, [b_i^{(1)}]_t^{P_i}, \ldots, [a_i^{(\ell)}]_t^{P_i}, [b_i^{(\ell)}]_t^{P_i})$ to produce $[c_i^{(1)}]_t^{P_i}, \ldots, [c_i^{(\ell)}]_t^{P_i}$ such that $c_i^{(l)} = a_i^{(l)}b_i^{(l)}$ for $l = 1, \ldots, \ell$. At most $t$ (corrupted) parties may fail to execute $\mathsf{ProveCeqAB}$. For simplicity assume first $2t+1$ parties are successful in executing $\mathsf{ProveCeqAB}$.
4. Now for each $l \in \{1, \ldots, \ell\}$, first $(2t+1)$ parties have produced $[c_1^{(l)}]_t^{P_1}, \ldots, [c_{(2t+1)}^{(l)}]_t^{P_{(2t+1)}}$. So for $l = 1, \ldots, \ell$, parties in $\mathcal{P}$ compute $[c^{(l)}]_t$ as follows: $[c^{(l)}]_t = Lagrange([c_1^{(l)}]_t^{P_1}, \ldots, [c_{(2t+1)}^{(l)}]_t^{P_{(2t+1)}})$.

Table 2: A protocol for computing $[c^{(l)}]_t = [a^{(l)}]_t[b^{(l)}]_t$