Unconditionally Secure Multiparty 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

In this paper, we re-visit the problem of unconditionally secure multiparty set intersection in information theoretic model. In ACNS 2007 [23], Li et.al have proposed a protocol for n-party set intersection problem, which provides unconditional security when $t < \frac{n}{3}$ players are corrupted by an active adversary having *unbounded computing power*. Moreover, they have claimed that their protocol takes six rounds of communication and incurs a communication complexity of $\mathcal{O}(n^4m^2)$, where each player has a set of size m. However, we show that the round complexity and communication complexity of the protocol in [23] is much more than what is claimed in [23]. We then propose a *novel* unconditionally secure protocol for multiparty set intersection problem with $n > 3t$ players, which significantly improves the "actual" round and communication complexity (as shown in this paper) of the protocol given in [23]. 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 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. An MPC protocol securely evaluates the circuit gate-by-gate [5, 26, 2, 26, 18, 20, 4].

The MPC problem was first defined and solved by Yao [28] in his seminal work in two-party scenario. The first generic solutions presented in [17, 9, 15] were based on cryptographic

[*] 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.

intractability assumptions. Later, the research on MPC in information theoretic model was initiated by Ben-Or et. al. [5] and Chaum et. al. [8] in two different independent work and carried forward by the works of [26, 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 $t < n/2$ [26]. 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 [18, 20, 4]. UMPC protocols with non-optimal resilience i.e $t < n/3$ are presented in [19, 12]. Finally, UMPC protocols with optimal resilience i.e $t < n/2$ are presented in [11, 3].

Secure Multiparty Set Intersection (MPSI): Secure multiparty set intersection (MPSI) problem is a specific but interesting instance of general secure multiparty computation (MPC) [28]. The MPSI problem can be stated as follows: A set of n players, each with a private input set want to compute the intersection of these sets, without leaking any *extra* information to the corrupted players, other than what is implied by the input of the corrupted players and the final output (which is the intersection of all the sets). So for the MPSI problem, the inputs are the secret sets of the players and the output is the intersection of these sets. The goal of a secure MPSI protocol is to achieve the above mentioned task correctly even in the presence of a centralized adversary who can control at most t players out of the n players. Secure MPSI problem has huge practical applications such as online recommendation services, online dating services, medical databases, data mining etc. [14].

The set intersection problem was first studied in cryptographic model (where the adversary has bounded computing power) in [14, 22] where the protocols presented are based on homomorphic public key encryption. By representing the sets as polynomials, the set intersection problem is converted into the task of computing the common roots of the polynomials. The same approach was carried over in information theoretic model (where the adversary has unbounded computing power) by the authors of [23], where the authors have claimed to present the first unconditionally secure MPSI protocol with $n \geq 3t + 1$.

Our Motivation: Two important parameters of generic 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. 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 secure multiparty computation protocol for the set intersection problem which achieves efficiency in both the parameters simultaneously.

Note that we can solve MPSI problem by using the protocols for generic MPC. However, since a generic solution does not exploit the nuances and the special properties of the problem, it is not efficient in general. Therefore, it is very interesting and useful to solve some specific practically-on-demand problems through direct approach. Moreover, it is not known how to modify the existing general MPC protocols to solve the MPSI problem. Also as no article has considered applying general MPC protocols to the MPSI problem before, we could not estimate the total round complexity and communication complexity when general MPC protocols are applied to set intersection problem. However, we provide a partial comparison in section 2.2.

Our Model: We denote the set of n players (parties) involved in the secure computation by $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ where player P_i possess set S_i. We assume that each set S_i for $1 \leq i \leq n$ is of size m. The protocols presented in this paper can easily be extended to work for the case when players have sets with different size. We assume that all the n players are connected with each other by pairwise secure channels. 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, receives values from neighbors and may again perform some more computation, in that order.

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 $t < \frac{n}{3}$. 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* [16], 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.* Our protocol provides unconditional security i.e. information theoretic security with a negligible error probability of $2^{-\mathcal{O}(\kappa)}$ in correctness for some security parameter κ. To bound the error probability, all our computations 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.

Broadcast: 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 system, 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 system, then broadcasting an ℓ bit(s) message can be simulated perfectly (without any error probability) by executing a protocol which takes $\mathcal{O}(t)$ rounds and communicates $\mathcal{O}(n^2\ell)$ bits, where $n \geq 3t + 1$ [6, 7].

2 Existing Solution for Unconditionally Secure MPSI Problem and Our Contribution

The MPSI problem was first studied in cryptographic model in [14, 22] where the protocols presented are based on homomorphic public key encryption. By representing the sets as polynomials, the set intersection problem is converted into the task of computing the common roots of the polynomials. The same approach was carried over by the authors in [23], where the authors have presented the first unconditionally secure MPC protocol for set intersection problem with $n \geq 3t+1$. We first recall the strategy to convert the problem of multiparty set intersection into the task of computing the common roots of the polynomials.

2.1 Polynomial Representation of Set and Reducing the MPSI problem to Finding Common Roots of Polynomials

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 $f(x) = \prod_{i=1}^{m}(x - s_i) = a_0 + a_1 x + \ldots + a_m x^m$. Thus the elements of set S are the m roots of the m-degree polynomial $f(x)$. Now 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 multiparty set intersection, player P_i represents his set S_i by a m-degree polynomial $f^{(P_i)}(x)$ and supplies $f^{(P_i)}(x)$ (its $m+1$ coefficients) as his input. Then all the players jointly and securely compute the function

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

where $r^{(1)}(x), \ldots r^{(n)}(x)$ are n random polynomials of degree m over $\mathbb{F}$, which are jointly generated by the n players. Note that $F(x)$ preserves all the common roots of $f^{(P_1)}(x), \ldots, f^{(P_n)}(x)$. Every element $s \in (S_1 \cap S_2 \cap \ldots S_n)$ 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 player who locally checks if $F(s) = 0$ for every s in his private set. All the elements at which the evaluation of $F(x)$ is zero forms the intersection set $(S_1 \cap S_2 \cap \ldots S_n)$. In [22], it has been proved formally that $F(x)$ does not reveal any *extra* information, other than what is deduced from intersection set $(S_1 \cap S_2 \cap \ldots S_n)$.

2.2 Existing Unconditionally Secure MPSI Protocol: Its Shortcomings and Analysis

The authors of [23] have used the same approach of [22] and converted the multiparty set intersection problem to finding common roots of polynomials. The authors claimed that their protocol takes 6 rounds and communicates $\mathcal{O}(n^4 m^2)$ elements from $\mathbb{F}$ (see sec 4.2 of [23]). [1] In order to securely compute function $F(x)$ given in Equation 1, the protocol of [23] is divided into three phases, namely **Input, Computation** and **Output Phase**. We briefly mention the steps performed in each phase, along with the corresponding round complexity and communication complexity as claimed in [23]. We then show that the round complexity and communication complexity of the protocol of [23] is much more than what is claimed in [23] and finally try to provide a correct complexity analysis for each of the phases.

1. Input Phase: Here each of the n players represent their secret set as a polynomial and t-shares[2] the coefficients of the polynomials among the n players. In order to do the sharing, the players use a two dimensional verifiable secret sharing. A two dimensional verifiable secret sharing (VSS) [16, 13, 21], which is an extension of Shamir secret sharing [27], ensures that each player (including a corrupted player) "consistently" and correctly t-shares the coefficients of his polynomial with everybody. Now, the authors in [23] claimed that this takes two rounds, where in the first round, each player does the sharing and in the second round verification is done by

[1] In [23], the authors have used k to denote the size of each set.

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

all the players to ensure whether everybody has received correct and consistent shares (see sec. 4.2 in [23]). Moreover, they 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 $n \geq 3t+1$ players is at least three [16, 13, 21]. 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[13, 21]. Now in the **Input Phase** of [23], each player executes $(m+1)$ VSS's to share the coefficients of his secret polynomial. In addition, each player also executes $n(m+1)$ VSS's to share the coefficients of n random polynomials of degree m each. So the total number of VSS done in **Input Phase** is $\mathcal{O}(n^2m)$. Hence, the **Input Phase** will take at least three rounds with a communication complexity of $\mathcal{O}(n^5m)$ and broadcast of $\mathcal{O}(n^5m)$ field elements. Moreover, if the broadcast channel is not available, then the **Input Phase** will take $\Omega(t)$ rounds with a communication overhead of $\mathcal{O}(n^7m)$ field elements.

2. Computation Phase: Given that the coefficients of $f^{(P_1)}(x), f^{(P_2)}(x), \ldots, f^{(P_n)}(x), r^{(1)}(x), r^{(2)}(x), \ldots, r^{(n)}(x)$ are correctly t-shared, in the computation phase, the players 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. In order to do so, the players 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 players locally multiply the shares of the coefficients of $r^{(i)}(x)$ and $f^{(P_i)}(x)$, for $1 \leq i \leq n$, which results in $2t$-sharing[3] of the coefficients of $f^{(P_1)}(x)r^{(1)}(x), \ldots, f^{(P_n)}(x)r^{(n)}(x)$. During **step 2**, each player calls a *re-sharing protocol* and converts the $2t$-shares of the coefficients of $f^{(P_1)}(x)r^{(1)}(x), \ldots, f^{(P_n)}(x)r^{(n)}(x)$ into t-sharing of these coefficients. The re-sharing protocol allows a party to produce t-sharing of an element given its t'-sharing, where $t' > t$. In [23], the authors have called a re-sharing protocol, without giving the actual details and claimed that re-sharing and other additionally required verifications will take *only three rounds* with a communication overhead of $\mathcal{O}(n^4m^2)$ field elements (see sec. 4.2 of [23]). The authors in [23] have given the reference of [18] for the details of re-sharing protocol. However, the protocol given in [18] is a protocol for general MPC, which uses "circuit based approach" to securely evaluate a function. Specifically, the MPC protocol of [18] assumes that the function (general) to be securely computed is represented as an arithmetic circuit over $\mathbb{F}$. The protocol divides the circuit as a sequence of *segments*, where each segment consists of addition and multiplication gate(s). The protocol evaluates one segment at a time, where either the segment is evaluated correctly or it may fail. In the case of failure, the protocol outputs a pair (P_i, P_j), where at least one of P_i or P_j is corrupted. In the case of failure, the segment is re-executed by neglecting P_i, P_j. Now the re-sharing protocol given in [18] is executed after each multiplication in a particular segment. In the worst case, a particular segment may fail t times (due to t corrupted players), in which case the re-sharing will be done at least t times (every segment contains at least one multiplication gate). In fact, the MPC protocol of [18] takes $\Omega(t)$ rounds in the presence of broadcast channel in the system, where as in the absence of broadcast channel it will take $\Omega(t^2)$ rounds. The authors in [23] 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 [23] are incomplete. In addition, it seems that the round complexity and communication complexity of the protocol as claimed in [23] are not consistent with the given protocol. Moreover, it is not mentioned whether they have assumed a broadcast channel in the system. The protocol in [23] makes use of two dimensional VSS and sub-protocols for multiplication of two shared secrets as black-boxes. If no

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

broadcast channel is available in the system, then even by using the current best known protocols for VSS and multiplication of shared secrets, any protocol needs $\Omega(t)$ rounds to securely evaluate the function given in Eqn (1). On the other hand, even if a broadcast channel is available in the system, then also by using the current best protocols for VSS and multiplication of shared secrets, any protocol requires more than six rounds and communication overhead of more than $\mathcal{O}(n^4m^2)$ field elements to evaluate the function given in Eqn (1).

Roughly, computation of the function given in Eqn. 1 requires $C_I = n(m+1)$ input gates (every player 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 outputted). The communication complexity of any multiparty computation protocol computing an agreed function f is the function of C_I, C_R, C_M and C_O where the arithmetic circuit realizing f requires C_I, C_R, C_M and C_O input, random, multiplication and output gates respectively. Now assuming that the circuit realizing set intersection problem requires the above mentioned number of gates, we give a rough estimation of the round complexity and communication complexity of general MPC protocols proposed in [5, 1, 18, 12, 4] to solve set intersection. Assuming the absence of a broadcast channel in the system[4], the communication complexity and round complexity of the current best known MPC protocols to implement the function given in Eqn. (1) is given in the following Table.

Reference of the protocol used	Communication complexity in bits	Round Complexity
[5]	$\mathcal{O}(n^7m^2\kappa)$	$\mathcal{O}(n)$
[1]	$\mathcal{O}(n^7m^2\kappa)$	$\mathcal{O}(n)$
[18]	$\mathcal{O}((n^5m + n^4m^2)\kappa)$	$\mathcal{O}(n^2)$
[12]	$\mathcal{O}((n^2m^2 + n^4)\kappa)$	$\mathcal{O}(n^2)$
[4]	$\mathcal{O}((n^2m^2 + n^3)\kappa)$	$\mathcal{O}(n^2)$

2.3 Our Contribution

We now propose a new protocol for unconditional MPSI problem with $n = 3t + 1$ and with complete complexity analysis. Our protocol communicates $\mathcal{O}((m^2n^3 + n^4)\kappa)$ bits and broadcasts $\mathcal{O}((m^2n^3 + n^4)\kappa)$ bits and requires constant number of rounds provided that broadcast channel is available. So in the absence of broadcast channel, our protocol requires communication of $\mathcal{O}((m^2n^5 + n^6)\kappa)$ bits and takes $\mathcal{O}(t)$ rounds. Comparing our results with the first two rows of the Table (provided above), we find that our protocol incurs much lesser communication complexity than the protocols of [5, 1], while keeping round complexity same. But the protocols of [18, 12, 4] provides better communication complexity than ours at the cost of increased round complexity. Our proposed protocol achieves its task by using some new techniques (which are presented first time in this paper) and some existing techniques.

3 Sub-Protocols Used in Our Protocol

In this section, 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. For convenience, we analyze the round complexity and communication complexity of the sub-protocols assuming the existence of physical broadcast channel in the system. Finally, the round and communication complexity of our multi-party set intersection protocol is evaluated

[4] We assume that a broadcast of ℓ bit message can be simulated in $\mathcal{O}(t)$ rounds with a communication overhead of $\mathcal{O}(n^2\ell)$ [6, 7].

in (a) the presence of physical broadcast and (b) the absence of physical broadcast. Recall that in the absence of broadcast channel, broadcast is simulated using a protocol.

3.1 Information Checking

Information Checking (IC) and IC Signatures [11, 26]: IC is an information theoretically secure method for authenticating data and is used to generate IC signatures. The IC signatures can be used as semi "digital 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 $S = [S^{(1)} \dots S^{(\ell)}] \in \mathbb{F}^{\ell}$, where $\ell \geq 1$ to intermediary INT. In addition, $\mathbf{D}$ hands some **authentication information** to INT and **verification information** to the 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 knows an S, which will be accepted in **RevealVal**, except with probability $2^{-\mathcal{O}(\kappa)}$.

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 present an IC protocol, called **EfficientIC**, which allows $\mathbf{D}$ 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 $n = 3t+1$. Let $S = (s^{(1)}, \dots, s^{(\ell)}) \in \mathbb{F}^{\ell}$. The idea of this protocol is taken from [25]. The protocol **EfficientIC** is given in Table 1. In the protocol **EfficientDistr** takes one round, **EfficientAuthVal** takes two rounds while **EfficientRevealVal** takes two rounds.

Lemma 1 *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 [25] and hence is omitted. □

When we say that $\mathbf{D}$ hands over $ICSig_{(s^{(1)}, \dots, s^{(\ell)})}(\mathbf{D}, INT)$ to INT, we mean **EfficientDistr** and **EfficientAuthVal** are executed in the background. Similarly, INT reveals $ICSig_{(s^{(1)}, \dots, s^{(\ell)})}(\mathbf{D}, INT)$ can be interpreted as INT along with other players invoking **EfficientRevealVal**.

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: **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, **D** selects another random $\ell + t - 1$ degree polynomial $R(x)$, over $\mathbb{F}$, which is independent of $F(x)$. **D** selects n distinct random elements $\alpha_1, \alpha_2, \ldots, \alpha_n$ from $\mathbb{F}$ such that each $\alpha_i \in \mathbb{F} - \{0, 1, \ldots, n-1\}$. **D** privately gives $F(x)$ and $R(x)$ to INT. To receiver $P_i \in \mathcal{P}$, **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 $1 \leq j \leq n$, **D** checks $dv_j + r_j \overset{?}{=} B(\alpha_j)$. If **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 **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 **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 **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 **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.

Table 1: An IC Protocol to Sign ℓ Secrets where $n \geq 3t + 1$

3.2 Generating ℓ Length Random Vector

We now present a protocol called **RandomVector($\mathcal{P}, \ell$)** (given in Table 2) which allows the players in $\mathcal{P}$ to jointly generate ℓ length random vector $(r^{(1)}, \ldots, r^{(\ell)}) \in \mathbb{F}^{\ell}$, where each $r^{(i)}$ is a random element in $\mathbb{F}$. The protocol uses the ideas from [12]. 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 [16] (see Fig 2 of [16]) as black box. The perfect VSS (see the definition of VSS in Section 2.1 of [16] with $n \geq 3t+1$ players consists of two phases, namely **Sharing Phase** and **Reconstruction Phase**. **Sharing Phase** takes four rounds and allows a dealer **D**(which can be any player from the set of n players) to verifiably share a secret $s \in \mathbb{F}$ by communicating $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits and broadcasting $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits where $|\mathbb{F}| \geq n$. **Reconstruction Phase** takes single round and allows all the (honest) players to reconstruct the secret s (shared by dealer **D** in **Sharing Phase**) by broadcasting $\mathcal{O}(n \log |\mathbb{F}|)$ bits, where $|\mathbb{F}| \geq n$. Notice that in this paper $|\mathbb{F}| = GF(2^{\kappa}) \geq n$. The VSS protocol has an important property that once **D** (possibly corrupted) shares a secret s during **Sharing Phase**, then **D** is committed s. Later, in the **Reconstruction Phase**, irrespective of the behavior of the corrupted players, the same s will be reconstructed. Thus a bad **D** will not be able to change his commitment from s to any other value during **Reconstruction Phase**.

As mentioned above, protocol **RandomVector** uses the properties of Vandermonde matrix to generate the random vector. We now briefly mention how this can be done. The current description is from [12]

Vandermonde Matrix and Randomness Extraction [12]: We express a matrix $M \in \mathbb{F}^{(r,c)}$ with r rows and c columns as $M = \{m_{i,j}\}_{i=1,\ldots,r}^{j=1,\ldots,c}$. We use M^T to denote the transpose of M. For distinct elements $\beta_1, \ldots, \beta_r$ from $\mathbb{F}$, we use $V^{(r,c)}$ to denote the Vandermonde Matrix $V^{(r,c)} = \{\beta_i^j\}_{i=1\ldots,r}^{j=0,\ldots,c-1}$. Now we can use $V^{(r,c)}$ for randomness extraction. Let $U = V^{(r,c)^T}$ be the transpose of $V^{(r,c)}$ with $r > c$. Now assume that $(x_1, \ldots, x_r)$ is generated by picking up c elements from $\mathbb{F}$ uniformly at random and then picking the remaining $r - c$ elements from $\mathbb{F}$ with an arbitrary distribution independent of the first c elements. Now we compute $(y_1, \ldots, y_c) = U(x_1, \ldots, x_r)$. $(y_1, \ldots, y_c)$ is an uniformly random vector of length c extracted from $(x_1, \ldots, x_r)$ [12]. This principle is used in protocol **RandomVector**.

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

1. Every player $P_i \in \mathcal{P}$ selects $L = \lceil \frac{\ell}{2t+1} \rceil$ random values $r^{(1,P_i)}, \ldots, r^{(L,P_i)}$.
2. Every player $P_i \in \mathcal{P}$ as a dealer invokes **Sharing Phase** of four round VSS protocol of [16] 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**), **Reconstruction Phase** of four round *perfect* VSS of [16] with $n \geq 3t + 1$ is invoked for L times separately. Now for every player $P_i \in \mathcal{P}$, the values $r^{(1,P_i)}, \ldots, r^{(L,P_i)}$ are public.
4. Now players compute $(r^{(1,1)}, \ldots, r^{(1,2t+1)}) = U(r^{(1,P_1)}, \ldots, r^{(1,P_n)})$, $(r^{(2,1)}, \ldots, r^{(2,2t+1)}) = U(r^{(2,P_1)}, \ldots, r^{(2,P_n)}), \ldots, (r^{(L,1)}, \ldots, r^{(L,2t+1)}) = U(r^{(L,P_1)}, \ldots, r^{(L,P_n)})$, where $U = V^{(n,2t+1)^T}$. Now let $r^{((i-1)(2t+1)+j)} = r^{(i,j)}$ for $1 \leq i \leq L$ and $1 \leq j \leq (2t + 1)$. $r^{(1)}, \ldots, r^{(\ell)}$ is the desired ℓ length random vector, generated jointly by all the players in $\mathcal{P}$.

Table 2: A Five Round Protocol for Generating ℓ length Random vector.

Lemma 2 *Protocol* **RandomVector** *generates ℓ length random vector $(r^{(1)}, \ldots, r^{(\ell)})$.* **RandomVector** *takes five rounds and communicates $\mathcal{O}(\ell n^2 \kappa)$ bits and broadcasts $\mathcal{O}(\ell n \kappa)$ bits.*

PROOF: The correctness of Protocol **RandomVector** follows from the correctness of the four round perfect VSS proposed in [16] and the randomness extraction capability of Vandermonde Matrix [12]. In protocol **RandomVector**, in total nL instances of **Sharing Phase** and nL instances of **Reconstruction Phase** will be invoked. Since each instance of **Sharing Phase** communicates $\mathcal{O}(n^2\kappa)$ bits, broadcasts $\mathcal{O}(n^2\kappa)$ bits and each instance of**Reconstruction Phase** broadcasts $\mathcal{O}(n\kappa)$ bits, protocol **RandomVector** communicates $\mathcal{O}(nLn^2\kappa) = \mathcal{O}(\ell n^2\kappa)$ bits and broadcasts $\mathcal{O}(\ell n\kappa)$ bits. □

3.3 Unconditional Verifiable Secret Sharing and Reconstruction

Definition 1 ***d*-1D-Sharing:** *We say that a value s is correctly d-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 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. A set of shares (possibly incomplete) is called d-consistent if these shares lie on a d degree polynomial.*

If a secret s is d-1D-shared by a player $\mathbf{D} \in \mathcal{P}$, then we denote it as $[s]_d^{\mathbf{D}}$. We say that a dealer $\mathbf{D} \in \mathcal{P}$ correctly t-1D-shares a secret s, if $\mathbf{D}$ picks a random t degree polynomial $f(x)$

over $\mathbb{F}$ with $f(0) = s$ and sends the share $f(i)$ to player P_i. This is simple Shamir-sharing [27]. However, a corrupted $\mathbf{D}$ may distribute shares which may not be t-consistent; i.e., the shares lie on a polynomial of degree greater than t. So, in the sequel, we describe a protocol **1DShare** which allows a dealer $\mathbf{D} \in \mathcal{P}$ to *verifiably* t-1D-share ℓ secret values $s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}$, with $\ell \geq 1$ resulting in each player $P_i \in \mathcal{P}$ holding the shares $s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{(\ell)}$, where $s_i^{(l)}$ denotes the i^{th} share of secret $s^{(l)}$. Verifiably 1D-sharing ensures correct t-1D-sharing even for a corrupted $\mathbf{D}$. The protocol uses few ideas from [12]. In the protocol, we use our **EfficientIC** protocol, which provides us with high efficiency. Notice that $\mathbf{D}$ can also produce the desired sharing by using a perfect VSS protocol with $n \geq 3t + 1$ [16, 13, 21] to share each $s^{(i)}$ separately. However, this will involve lot of communication overhead. Rather, by using the ideas of [12] and incorporating our **EfficientIC** protocol, $\mathbf{D}$ can do the same task with less communication overhead, but with a negligible error probability. Before describing the protocol, we define few notations which are commonly used in the literature.

By saying that the players in $\mathcal{P}$ compute (locally) $([y^{(1)}]_d, \ldots, [y^{(\ell')}]_d) = \varphi([x^{(1)}]_d, \ldots, [x^{(\ell)}]_d)$ (for any function $\varphi : \mathbb{F}^{\ell} \rightarrow \mathbb{F}^{\ell'}$), 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 φ to correct d-1D-sharings, we get correct d-1D-sharings of the output. So by adding two correct d-1D-sharings , we get another correct d-1D-sharing of the sum, i.e. $[a]_d + [b]_d = [a+b]_d$. However, by multiplying two correct d-1D-sharings, we get a correct $2d$-1D-sharing of the product, i.e. $[a]_d[b]_d = [ab]_{2d}$.

The protocol **1DShare** is given in Table 3. The goal of the protocol is as follows: (a) If $\mathbf{D}$ is honest then he correctly generates t-1D-sharing of the secrets $(s^{(1)}, s^{(2)}, \ldots, s^{(\ell)})$, such that all the honest players publicly verify that $\mathbf{D}$ has correctly generated the sharing. Also when $\mathbf{D}$ is honest, then the secrets will be information theoretically secure from the adversary $\mathcal{A}_t$. (b) If $\mathbf{D}$ is corrupted and has not generated correct t-1D-Sharing, then with very high probability, everybody will detect it and protocol will terminate with everybody accepting a pre-defined ℓ t-1D-sharings of 1, namely $[1]_t, [1]_t, \ldots, [1]_t$ on behalf of $\mathbf{D}$.

Informally the protocol works as follows: $\mathbf{D}$ chooses $\ell{+}1$ random polynomials $f^{(0)}(x), \ldots, f^{(\ell)}(x)$ such that $f^{(0)}(0) = r$ and $f^{(l)}(0) = s^{(l)}$ for $l = 1, \ldots, \ell$ where r is a random non-zero value. $\mathbf{D}$ then hands over $ICSig_{(f^{(0)}(i), \ldots, f^{(\ell)}(i))}(\mathbf{D}, P_i) = ICSig_{(r_i, s_i^{(1)}, \ldots, s_i^{(\ell)})}(\mathbf{D}, P_i)$ to P_i where $r_i, s_i^{(1)}, \ldots, s_i^{(\ell)}$ denotes i^{th} shares of r and $s^{(1)}, \ldots, s^{(\ell)}$ respectively. All the players then combinedly generate a non-zero random value z using Protocol **RandomVector**. Now $\mathbf{D}$ 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 player P_i computes and broadcasts $y_i = r_i + \sum_{l=1}^{\ell} s_i^{(l)} z^l$. Every player checks whether $f(i) \overset{?}{=} y_i$ for all $i = 1, \ldots, n$. If yes then $\mathbf{D}$ has succeeded to produce $(s^{(1)}]_t^{\mathbf{D}}, \ldots, [s^{(\ell)}]_t^{\mathbf{D}}$. Otherwise, there exists at least one player, say P_i, such that $f(i) \neq y_i$. In this case, player P_i is asked to reveal $ICSig_{(r_i, s_i^{(1)}, \ldots, s_i^{(\ell)})}(\mathbf{D}, P_i)$; i.e., $\mathbf{D}$'s IC signature on the shares of $s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}$ and r. If the signature is correct then P_i has raised a **valid complaint** against $\mathbf{D}$ (with very probability). So $\mathbf{D}$ is detected to be corrupted and the protocol terminates with the players assuming a predefined ℓ t-1D-sharings of 1: $[1]_t, [1]_t, \ldots, [1]_t$.

Lemma 3 *If* $\mathbf{D}$ *is honest then protocol* **1DShare** *correctly and securely generates t-1D-sharing of* ℓ *secrets except with error probability of* $2^{-\mathcal{O}(\kappa)}$. *If* $\mathbf{D}$ *is corrupted and if some of the* ℓ *sharings dealt by* $\mathbf{D}$ *is not t-1D-sharing, then* $\mathbf{D}$ *fails to generate the required sharings and* ℓ *t-1D-sharings of 1 will be outputted, except with error probability of* $2^{-\mathcal{O}(\kappa)}$. *The protocol takes eleven rounds, communicates* $\mathcal{O}((\ell n + n^2)\kappa)$ *bits and broadcasts* $\mathcal{O}((\ell n + n^2)\kappa)$ *bits.*

Proof: The communication complexity and number of rounds can be checked easily by inspection. We now prove it's correctness. If $\mathbf{D}$ is honest then all the honest players will correctly

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

1. For every $l = 1, \ldots, \ell$, **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 $r \in_R \mathbb{F}$ is an uniformly random non-zero element. For every $i = 1, ..., n$, **D** and player P_i (acting as INT), respectively executes **EfficientDistr** and **EfficientAuthVal** of **EfficientIC**$(\mathbf{D}, P_i, \mathcal{P}, \ell + 1, r_i, s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{(\ell)})$, so that player P_i (as an INT) gets **D**'s IC signature on $r_i, s_i^{(1)}, s_i^{(2)}, \ldots, s_i^{(\ell)}$, where $r_i = f^{(0)}(i)$ and $f^{(l)}(i) = s_i^{(l)}$ for $l = 1, \ldots, \ell$.
2. All the players 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 player P_i computes and broadcasts $y_i = r_i + \sum_{l=1}^{\ell} s_i^{(l)} z^l$.
4. If the polynomial $f(x)$ broadcasted by **D** is of degree more than t, then output t-1D-sharing of ℓ 1's i.e $[1]_t, [1]_t, \ldots, [1]_t$, and the protocol terminates here.
5. Every player 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, for some P_i, with $f(i) \neq y_i$, player P_i (acting as INT), along with the receivers in $\mathcal{P}$ execute **EfficientRevealVal**$(\mathbf{D}, P_i, \mathcal{P}, \ell + 1, r_i, s_i^{(1)}, \ldots, s_i^{(\ell)})$ to reveal $ICSig_{(r_i, s_i^{(1)}, \ldots, s_i^{(\ell)})}(\mathbf{D}, P_i)$. If P_i succeeds to prove **D**'s signature on $r_i, s_i^{(1)}, \ldots, s_i^{(\ell)}$ and $f(i) \neq r_i + \sum_{l=1}^{\ell} s_i^{(l)} z^l$, then output t-1D-sharing of ℓ 1's i.e $[1]_t, [1]_t, \ldots, [1]_t$ 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 the t-1D-sharings $[s^{(1)}]_t, [s^{(2)}]_t, \ldots, [s^{(\ell)}]_t$.

Table 3: A Eleven Round Protocol for verifiably t-1D-share ℓ secrets.

verify the t-1D-sharing of ℓ secrets. However, a corrupted player P_i may broadcast incorrect $y_i' \neq y_i$, such that $y_i' \neq f(i)$ and can forge **D**'s IC signature on the corresponding incorrect $r_i' \neq r_i$ or/and $s_i'^{(j)} \neq s_i^{(j)}, 1 \leq j \leq \ell$. But from the properties of **IC** protocol this can happen with error probability of at most of $2^{-\mathcal{O}(\kappa)}$.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.

Now we consider the case, when **D** is corrupted and the sharing of at least one of the secrets $s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}$ is not a correct t-1D-sharing, i.e. the shares of the honest players lie on a polynomial of degree higher than t. The argument follows from [12]. Let $f^{(l)}(x)$ is the polynomial sharing $s^{(l)}$ and $f^{(0)}(x)$ is the polynomial sharing r. According to our assumption at least one of the polynomials $f^{(l)}(x)$ for $l \in \{0, 1, \ldots, \ell\}$ has degree greater than t. Now we show that the polynomial $f(x) = f^{(0)}(x) + \sum_{l=1}^{\ell} f^{(l)}(x) z^l = \sum_{l=0}^{\ell} f^{(l)}(x) z^l$ will be of degree t with negligible probability. Let m be such that $f^{(m)}(x)$ has maximal degree among $f^{(0)}(x), \ldots, f^{(\ell)}(x)$, and let t_m be the degree of $f^{(m)}(x)$. Then according to the condition, $t_m > t$. Now each polynomial $f^{(l)}(x)$ can be written as $f^{(l)}(x) = \alpha_l x^{t_m} + f'^{(l)}(x)$ where $f'^{(l)}(x)$ has degree lower that t_m. By assumption $\alpha_m \neq 0$. Let $g(x) = \sum_{l=0}^{\ell} \alpha_l x^l$. It is easy to see that $g(x)$ is a non-zero polynomial because at least $\alpha_m \neq 0$. Also, degree of $g(x)$ is at most ℓ. Now in the protocol, a corrupted **D** will not be caught if $g(z) = 0$. But since the polynomial $g(x)$ will be non-zero polynomial with degree at most ℓ and since z is chosen uniformly at random from $\mathbb{F}$, $g(z) = 0$ will occur with probability at most $\frac{\ell}{|\mathbb{F}|} = 2^{-\mathcal{O}(k)}$. So we may assume that $g(z) \neq 0$ with very high probability. This implies that $f(x) = \sum_{l=0}^{\ell} f^{(l)}(x) z^l$ has degree $t_m > t$. It follows that if some of the sharings done by **D** are not t-1D-sharings, then **D** will be detected as faulty with very high probability. □

We now present a simple protocol called **ReconsPublic**, for reconstructing a secret which is correctly t-1D-Shared. In the protocol, every player broadcasts his share. Now out of these n shares, at most t could be corrupted. But since $n \geq 3t+1$, by applying standard error correction algorithm (e.g. Berlekamp Welch Algorithm [24]), the secret can be recovered. The protocol is given in Table 4.

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

Each player P_i broadcasts his share of s, namely s_i. All the players apply error correction (e.g. Berlekamp Welch Algorithm) to reconstruct s from the n shares.

Table 4: A Single Round Reconstruction Protocol

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

3.4 Upgrading t-1D-sharing to t-2D-sharing

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

If a secret s is d-2D-shared by a player $\mathbf{D} \in \mathcal{P}$, then we denote it as $[[s]]_d^{\mathbf{D}}$. Notice that if s is d-2D-shared, i.e. $[[s]]_d$, then the i^{th} shares of the secret, namely s_i will be automatically d-2D-shared by player P_i, i.e $[s_i]_d^{P_i}$. We now present a protocol for upgrading t-1D-sharing to t-2D-sharing. Specifically, given that the players in $\mathcal{P}$ jointly holds the correct t-1D-sharing of ℓ secrets $s^{(1)}, s^{(2)}, \ldots, s^{(\ell)}$, i.e. $[s^{(1)}]_t, [s^{(2)}]_t, \ldots, [s^{(\ell)}]_t$, our protocol **Upgrade1Dto2D**, given in Table 5, outputs t-2D-sharings $[[s^{(1)}]]_t, [[s^{(2)}]]_t, \ldots, [[s^{(\ell)}]]_t$. Some of the ideas of the protocol are taken from [3], where a protocol for upgrading t-1D-sharing to t-2D-sharing of ℓ secrets has been provided with dispute control for $n \geq 2t+1$ players. The protocol of [3] either performs the upgradation or may fail. But in the case of failure, it outputs a pair (P_i, P_j), where at least one of P_i or P_j is corrupted. We use some of their [3] ideas to design our protocol for $n \geq 3t+1$ players. Our protocol always outputs the correct t-2D-sharings of ℓ secrets, with overwhelming probability, given correct t-1D-sharings for the same set of secrets. Moreover, $\mathcal{A}_t$ will learn nothing about the original secrets during the upgradation. Furthermore, if a player tries to cheat during the upgradation, then with very high probability, it will be caught.

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

Proof: The communication and round complexity of Protocol **Upgrade1Dto2D** can be verified by inspection of the protocol (there are two instances of **1DShare** protocol which takes eleven rounds each). We now prove the correctness. Provided ℓ correct t-1D-sharing $[s^{(1)}]_t, [s^{(2)}]_t, \ldots, [s^{(\ell)}]_t$, every honest player P_i will correctly t-1D-share his shares $s_i^{(0)}, s_i^{(1)}, \ldots, s_i^{(\ell)}$. Without loss of generality let the first t players are corrupted and remaining $2t+1$ players are honest. Now for every honest player P_h for some $h \in \{(t+1), \ldots, (3t+1)\}$, the value $s_h = s_h^{(0)} + \sum_{l=1}^{\ell} r^{(l)} s_h^{(l)}$ will be reconstructed correctly, with s_h being the h^{th} share of $s = s^{(0)} + \sum_{l=1}^{\ell} r^{(l)} s^{(l)}$. But a corrupted player P_c, where $c \in \{1, 2, \ldots, t\}$ may share $\bar{s}_c^{(0)}, \bar{s}_c^{(1)}, \ldots, \bar{s}_c^{(\ell)}$ with $\bar{s}_c^{(l)} \neq s_c^{(l)}$ for some $l \in \{0, 1, \ldots, \ell\}$. If P_c shares $\bar{s}_c^{(0)}, \bar{s}_c^{(1)}, \ldots, \bar{s}_c^{(\ell)}$ with

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

1. Each player $P_i \in \mathcal{P}$ invokes **1DShare**$(P_i, \mathcal{P}, 1, s^{(0,P_i)})$ to verifiably t-1D-share a random non-zero value $s^{(0,P_i)} \in \mathbb{F}$. Let $s_j^{(0,P_i)}$ denotes the j^{th} share of $s^{(0,P_i)}$.
2. Now every player P_i computes his sum share $s_i^{(0)} = \sum_{j=1}^{n} s_i^{(0,P_j)}$ which is the i^{th} share of $s^{(0)} = \sum_{j=1}^{n} s^{(0,P_j)}$. So now players in $\mathcal{P}$ hold correct t-1D-sharings of $\ell + 1$ secrets $s^{(0)}, s^{(1)}, \ldots, s^{(\ell)}$, i.e. $[s^{(0)}]_t, [s^{(1)}]_t, \ldots, [s^{\ell}]_t$.
3. Now every player P_i invokes **1DShare**$(P_i, \mathcal{P}, \ell+1, s_i^{(0)}, s_i^{(1)}, \ldots, s_i^{(\ell)})$ to correctly t-1D-share his shares $s_i^{(0)}, s_i^{(1)}, \ldots, s_i^{(\ell)}$ of the secrets $s^{(0)}, s^{(1)}, \ldots, s^{(\ell)}$ respectively, in that order.
4. Now to detect the players P_k (at most t), who have t-1D-shared wrong shares $\bar{s}_k^{(0)}, \bar{s}_k^{(1)}, \ldots, \bar{s}_k^{(\ell)}$ with $\bar{s}_k^{(l)} \neq s_k^{(l)}$ for some $l \in \{0, 1, \ldots, \ell\}$, the players in $\mathcal{P}$ jointly generate an ℓ length random vector $(r^{(1)}, \ldots, r^{(\ell)})$ by invoking Protocol **RandomVector**$(\mathcal{P}, \ell)$. Now all the players publicly reconstructs $s_i = s_i^{(0)} + \sum_{l=1}^{\ell} r^{(l)} s_i^{(l)}$ and $s = s^{(0)} + \sum_{l=1}^{\ell} r^{(l)} s^{(l)}$. This is done as follows:
   (a) Player P_i computes $s_i = s_i^{(0)} + \sum_{l=1}^{\ell} r^{(l)} s_i^{(l)}$. Then **ReconsPublic**$(\mathcal{P}, [s_i]_t)$ is invoked to publicly reconstruct s_i, for $i = 1, \ldots, n$.
   (b) Once all the players publicly reconstruct $s_1, s_2, \ldots, s_n$, s is reconstructed by applying Berlekamp Welch error correcting algorithm to $s_1, s_2, \ldots, s_n$. Berlekamp Welch Algorithm also provides the error locations. Hence if s_i has been pointed as error location, the sharings done by P_i by executing **1DShare**$(P_i, \mathcal{P}, \ell+1, s_i^{(0)}, s_i^{(1)}, \ldots, s_i^{(\ell)})$ will be ignored.
5. Output $[[s^{(1)}]]_t, [[s^{(2)}]]_t, \ldots, [[s^{(\ell)}]]_t$.

Table 5: A Twenty Eight Round Protocol For Upgrading t-1D-sharing to t-2D-sharing.

$\bar{s}_c^{(l)} \neq s_c^{(l)}$ for some $l \in \{0, 1, \ldots, \ell\}$, then with very high probability P_c will be caught. This is because if P_c executes **1DShare** with $\bar{s}_c^{(0)}, \bar{s}_c^{(1)}, \ldots, \bar{s}_c^{(\ell)}$, then **ReconsPublic**$(\mathcal{P}, [\bar{s}_c]_t)$ will publicly reconstruct the value $\bar{s}_c$. The probability that $\bar{s}_c = s_c$ is negligible since the ℓ length vector $(r^{(1)}, \ldots, r^{(\ell)})$ is chosen uniformly at random. So with very high probability $\bar{s}_c$ will not be the actual c^{th} share of s. Hence Berlekemp Welch Error correction algorithm when applied to the shares $s_1, s_2, \ldots, \bar{s}_c, \ldots s_t, s_{t+1}, \ldots, s_n$ will point $\bar{s}_c$ as error location in which case P_c will be caught and will not be involved in further computation. It is easy to see that at any stage of the protocol, $\mathcal{A}_t$ learns no more than t shares for each $s^{(l)}, 1 \leq l \leq \ell$. Hence all the secrets are information theoretically secure. □

3.5 Proving $c = ab$

Now let **D**$\in \mathcal{P}$ holds ℓ pairs of values $(a^{(1)}, b^{(1)}), \ldots, (a^{(\ell)}, b^{(\ell)})$ such that **D** has already correctly t-1D-shared $a^{(1)}, \ldots, a^{(\ell)}$ and $b^{(1)}, \ldots, b^{(\ell)}$ among the players in $\mathcal{P}$. Now **D** wants to correctly t-1D-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** (given in Table 6) to achieve this task. The idea of the protocol is inspired from [11] with the following modification: we make use of our protocol **1DShare**, which provides us with high efficiency.

We try to explain the idea of the protocol with a single pair (a, b). **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-1D-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-1D-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. All the players then compute $[Y]_t = r[a]_t + [\beta]_t$. **D** then broadcasts the value $Y^{\mathbf{D}} = ra + \beta$, while the players publicly reconstruct Y. Everybody then verifies whether Y is same as $Y^{\mathbf{D}}$. If $\mathbf{D}$ has correctly shared c, β and βb, then $Y = Y^{\mathbf{D}}$. Otherwise all the players will conclude that $\mathbf{D}$ fails to prove $c = ab$. However, if $Y = Y^{\mathbf{D}}$, then the players proceed further and compute $[X]_t = Y[b]_t - [\beta b]_t - r[c]_t$. The players then publicly reconstructs X. Now again if $\mathbf{D}$ has correctly shared c, β and βb, then $X = Yb - \beta b - rc = 0$. So after reconstructing X, every body checks whether $X = 0$. If $X = 0$ then everybody accepts the t-1D-sharing of c as valid t-1D-sharing of ab. Otherwise all the players will conclude that $\mathbf{D}$ fails to prove $c = ab$.

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 $\overline{\beta b} \neq \beta b$ or $\overline{c} \neq c$ but still X can be zero and this will happen iff $f_{\overline{\beta b}}(x) + rf_{\overline{c}}(x) = f_{\beta b}(x) + rf_c(x)$. 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. The secrecy follows from the fact that the broadcasted value $Y^{\mathbf{D}}$ is randomly distributed. Now we can extend the above idea parallely for each of the ℓ pairs $(a^{(l)}, b^{(l)})$.

Lemma 6 *In protocol* **ProveCeqAB**, *if* $\mathbf{D}$ *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 eighteen rounds and communicates* $\mathcal{O}((\ell n + n^2)\kappa)$ *bits and broadcasts* $\mathcal{O}((\ell n + n^2)\kappa)$ *bits. Moreover, if* $\mathbf{D}$ *is honest then* $\mathcal{A}_t$ *learns no information about* $a^{(l)}, b^{(l)}$ *and* $c^{(l)}$, *for* $1 \leq l \leq \ell$.

Proof: The round complexity and communication complexity follows from the working of the protocol. If $\mathbf{D}$ is honest then he will always be able to prove that every $(a^{(l)}, b^{(l)})$, $c^{(l)}$ satisfies $c^{(l)} = a^{(l)}, b^{(l)}$ for $l = 1, \ldots, \ell$. We now consider the case when $\mathbf{D}$ is corrupted. Let corrupted $\mathbf{D}$ has shared $\overline{\beta^{(l)}b^{(l)}} \neq \beta^{(l)}b^{(l)}$ or/and $\overline{c^{(l)}} \neq c^{(l)}$ for some $l \in \{1, \ldots, \ell\}$. We show that the probability of $X^{(l)} = 0$ is negligibly small. Notice that $X^{(l)} = 0$ will hold only when $\overline{\beta^{(l)}b^{(l)}} + r\overline{c^{(l)}} = \beta^{(l)}b^{(l)} + rc^{(l)}$. 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.

It is easy to see that if $\mathbf{D}$ is honest then at any stage of the protocol $\mathcal{A}_t$ does not get more than t shares for $a^{(l)}, b^{(l)}$ and $c^{(l)}$, for $1 \leq l \leq \ell$. Also the values $Y^{(l)}$ are randomly distributed over $\mathbb{F}$. This implies that $a^{(l)}, b^{(l)}$ and $c^{(l)}$ are information theoretically secure from $\mathcal{A}_t$. □.

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

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

Table 6: A Eighteen Round Protocol to Generate t-1D-Sharing of $c^{(l)}$ Where $c^{(l)} = a^{(l)}b^{(l)}$

3.6 Multiplication

Let the players in $\mathcal{P}$ hold ℓ pairs of correct t-1D-sharing among themselves, i.e. $([a^{(1)}]_t, [b^{(1)}]_t), \ldots, ([a^{(\ell)}]_t, [b^{(\ell)}]_t)$. We now present a protocol called **Mult** which allows the players to compute ℓ t-1D-sharings $[c^{(1)}]_t, \ldots, [c^{(\ell)}]_t$ such that $c^{(l)} = a^{(l)}b^{(l)}$ for $l = 1, \ldots, \ell$. Our protocol is motivated from the technique used in [11]. We explain the idea with only one pair, say (a, b). Given that the players hold the t-1D-sharings of a and b i.e. 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 $d_i = a_ib_i$ of c where c is $2t$-1D-shared. 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 that each player P_i t-1D-shares the value $d_i = a_ib_i$ with the proof that d_i is indeed the multiplication of a_i and b_i. Now, all the players jointly hold $[d_1]_t \ldots, [d_n]_t$. Since $d_1, \ldots, d_n$ 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 $c = \sum_{i=1}^{n} r_i d_i$ 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 [10] which is public and known to every player. So for shorthand notation, we write $c = Lagrange(d_1, \ldots, d_n) = \sum_{i=1}^{n} r_i d_i$. Now all players compute $[c]_t = Lagrange([d_1]_t, \ldots, [d_n]_t) = \sum_{i=1}^{n} r_i [d_i]_t$, to obtain the desired output. Notice that since $C(x)$ polynomial is of degree $2t$, we need $2t + 1$ players to successfully t-1D-share his d value (a $2t$ degree polynomial requires $2t + 1$ points on it to be interpolated correctly). So, even if t corrupted players fail to t-1D-share their d values, our protocol will work. Protocol **Mult**, achieves this task for ℓ pairs simultaneously.

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

1. All the players invoke **Upgrade1Dto2D**($\mathcal{P}, \ell, [a^{(1)}]_t, \ldots, [a^{(\ell)}]_t$) and **Upgrade1Dto2D**($\mathcal{P}, \ell, [b^{(1)}]_t, \ldots, [b^{(\ell)}]_t$) to upgrade t-1D-sharings of 2ℓ 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 $(b_1^{(l)}, \ldots, b_n^{(l)})$ denotes the 1D sharings of $a^{(l)}$ and $b^{(l)}$ respectively. Since $a^{(l)}$ and $b^{(l)}$ is t-2D-shared, their shares $a_i^{(l)}$ and $b_i^{(l)}$ are t-1D-shared (see the definition of t-2D-sharing) by player P_i. The players 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 player 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 **Upgrade1Dto2D** executed in step 1 of this protocol. Now each player P_i invokes **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) players may fail to execute **ProveCeqAB**. For simplicity assume first $2t + 1$ players are successful in executing **ProveCeqAB**.
4. Now for $l \in \{1, \ldots, \ell\}$, first $(2t+1)$ players have produced $[c_1^{(l)}]_t^{P_1}, \ldots, [c_{(2t+1)}^{(l)}]_t^{P_{(2t+1)}}$. So for $l = 1, \ldots, \ell$, players in $\mathcal{P}$ computes $[c^{(l)}]_t$ as follows: $[c^{(l)}]_t = Lagrange([c_1^{(l)}]_t^{P_1}, \ldots, [c_{(2t+1)}^{(l)}]_t^{P_{(2t+1)}})$.

Table 7: A protocol for computing $[c^{(l)}]_t = [a^{(l)}]_t[b^{(l)}]_t$

Lemma 7 *With overwhelming probability, protocol* **Mult** *produces ℓ t-1D-sharings $[c^{(1)}]_t, \ldots, c^{(\ell)}]_t$ from ℓ pairs of t-1D-sharings $([a^{(1)}]_t, [b^{(1)}]_t), \ldots, ([a^{(\ell)}]_t, [b^{(\ell)}]_t)$.* **Mult** *takes 46 (twenty eight for* **Upgrade1Dto2D** *+ eighteen for* **ProveCeqAB***) rounds and communicates $\mathcal{O}((\ell n^2 + n^3)\kappa)$ bits and broadcasts $\mathcal{O}((\ell n^2 + n^3)\kappa)$ bits. Moreover, $\mathcal{A}_t$ learns nothing about $c^{(l)}, a^{(l)}$ and $b^{(l)}$, for $1 \leq l \leq \ell$.*

3.7 Generating Random t-1D-Sharing

We present a protocol called **Random**($\mathcal{P}, \ell$), given in Table 8, which allows the players in $\mathcal{P}$ to jointly generate ℓ random t-1D-sharings, $[r^{(1)}]_t, \ldots, [r^{(\ell)}]_t$, where each $r^{(i)}$ is a random element in $\mathbb{F}$.

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

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

Table 8: A Eleven Round Protocol for Generating ℓ Random t-1D-sharings.

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

4 Unconditionally Secure MPSI Protocol with $n \geq 3t + 1$

We now present our unconditionally secure MPSI protocol with $n \geq 3t+1$ players. The protocol has three phases: Input and Preparation Phase, Computation Phase and Output Phase. In the input and preparation phase, each party properly shares their inputs and in computation phase, computation is carried out jointly on the shared inputs to get the final output. Recall that players wants to compute securely the function $F(x)$ where $F(x)$ is as follows:

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

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

Input and Preparation Phase

1. Every player $P_i \in \mathcal{P}$ 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 with $f^{(P_i)}(x) = (x - e_i^{(1)}) \ldots (x - e_i^{(m)}) = a^{(0,P_i)} + a^{(1,P_i)}x + \ldots + a^{(m,P_i)}x^m$, such that $f^{(P_i)}(e_i^{(j)}) = 0$, for $1 \leq j \leq m$. Notice that $a^{(m,P_i)} = 1$. P_i then invokes **1DShare**($P_i, \mathcal{P}, m, a^{(0,P_i)}, \ldots, a^{(m-1,P_i)}$) to verifiably t-1D-share first lower order m coefficients of $f^{P_i}(x)$. Since $a^{(m,P_i)} = 1$, every player in $\mathcal{P}$ assumes a predefined t-1D-sharing for 1 on behalf of $a^{(m,P_i)}$ (see Remark 1).
2. Let for $i = 1, \ldots, n$ polynomial $r^{(i)}(x)$ be expressed as $r^{(i)}(x) = b^{(0,i)} + b^{(1,i)}x + \ldots + b^{(m,i)}x^m$. So all the players should jointly generate $n(m+1)$ random t-1D-sharings for $n(m+1)$ random coefficients of the polynomials $r^{(1)}(x), \ldots, r^{(n)}(x)$. So all the players in $\mathcal{P}$ invoke **Random**($\mathcal{P}, m+1$) n times to produce n different set of $m+1$ t-1D-sharings. So the i^{th} invocation of **Random**($\mathcal{P}, m+1$) will generate the t-1D-sharings of the coefficients of polynomial $r^{(i)}(x)$, namely $[b^{(0,i)}]_t, \ldots, [b^{(m,i)}]_t$. This step can be executed in parallel to step 1.

Table 9: Input and Preparation Phase of our Unconditionally secure MPSI Protocol

Remark 1 *If we try to compute set intersection by using the function given in Eqn. (1), then the adversary can disrupt the security by forcing a corrupted player to input a zero polynomial representing his set. If he does so, then he can easily compute the intersection of the sets of the remaining players. Recall that a player P_j's polynomial $f^{(P_j)}(x) = \prod_{k=1}^{m}(x - e_j^{(k)})$. It is clear that the coefficient of m^{th} degree term is 1. This is true for every players polynomial. Hence, as specified in [23, 22], to prevent a corrupted player giving zero polynomial as input, it is enough to consider 1 as the default value for the coefficients of m^{th} degree term of the polynomial. So by default t-1D-sharing of 1 will be taken as every players m^{th} degree coefficient. Notice that we have incorporated this idea in the step 1. of* **Input and Preparation Phase** *shown in Table 9.*

After input and preparation phase, the players jointly compute the coefficients of the polynomial $F(x) = \sum_{i=1}^{n} r^{(i)}(x)f^{(P_i)}(x)$ in a shared manner in the **Computation Phase**. In the **Output Phase**, all the coefficients of polynomial $F(x)$ are publicly reconstructed. Then each player 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. The computation and output phase is shown in Table 10.

Computation Phase

1. All the players in $\mathcal{P}$ compute $F^{(i)}(x) = r^{(i)}(x)f^{(P_i)}(x)$ such that $F^{(i)}(x) = c^{(0,i)}+c^{(1,i)}x+\ldots+c^{(2m,i)}x^{2m}$ is a $2m$ degree polynomial and all its coefficients $c^{(0,i)}, \ldots, c^{(2m,i)}$ are correctly t-1D-shared. For $i = 1, \ldots, n$, the coefficients of all $F^{(i)}(x)$ are computed parallely.
   (a) Since multiplication of polynomials $r^{(i)}(x)$ and $f^{(P_i)}(x)$ requires multiplication of every pair of their coefficients, all players 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 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) All players 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. All the players in $\mathcal{P}$ compute $F(x) = \sum_{i=1}^{n} F^{(i)}(x)$ such that $F(x) = d^{(0)} + d^{(1)}x + \ldots + d^{(2m)}x^{2m}$ is a $2m$ degree polynomial and all its coefficients $d^{(0)}, \ldots, d^{(2m)}$ are correctly t-1D-shared. For this all players in $\mathcal{P}$ compute $[d^{(j)}]_t = \sum_{i=1}^{n}[c^{(j,i)}]_t$ for $j = 0, \ldots, 2m$.

Output Phase

1. The coefficients $d^{(0)}, \ldots, d^{(2m)}$ of $F(x)$ are publicly reconstructed. For that players invoke **ReconsPublic**$(\mathcal{P}, [d^{(j)}]_t)$ to reconstruct $d^{(j)}$ for $j = 0, \ldots, 2m$.
2. Each player P_i locally evaluates $F(x)$ at each element of his set S_i. If the evaluation is zero then the element belongs to the intersection, else the element does not belong to the intersection. All the elements from S_i at which the evaluation is zero form the intersection set $S_1 \cap S_2 \ldots, \cap S_n$.

Table 10: Computation Phase and Output phase of our Multiparty Set Intersection Protocol

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

Theorem 3 *Multiparty set intersection protocol with $3t+1$ players takes 58 rounds and communicates $\mathcal{O}((m^2n^3+n^4)\kappa$ bits and broadcasts $\mathcal{O}((m^2n^3+n^4)\kappa$ 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 communicates $\mathcal{O}((m^2n^5+n^6)\kappa$ bits overall. The protocol correctly and securely solves unconditionally secure MPSI problem with very high probability.*

PROOF: The round complexity and communication complexity can be easily verified from the protocol. The correctness follows from the argument given in Section 2.1. The secrecy follows from the secrecy of the protocols **1DShare, Random** and **Mult**. □

5 Conclusion

The proliferation of Internet has triggered tremendous opportunities for secure distributed computation, where people cooperate with each other by supplying their inputs with the intention to compute a function (task) of their common interest. The group of people participating in this computation may even be competitors, who do no trust each other. In such scenarios privacy becomes a primary concern. Secure multiparty set intersection is one of the most sought for practically important candidate problem of secure distributed computation. Even though, this problem is extensively studied in the settings of computational security, very less attention has been paid to the problem in information theoretic settings. For the first time in the literature, the authors in [23] had attempted to give a protocol for unconditionally secure multiparty set intersection problem with $n \geq 3t+1$ players in information theoretic model. However, in this paper, we have shown that the round complexity and communication complexity of the protocol in [23] is much more than what is claimed in [23]. We then provided the actual complexity analysis of the protocol given in [23]. Finally, we proposed a new unconditionally secure protocol for multiparty set intersection problem with $n \geq 3t+1$, which significantly improves the (actual) communication and round complexity of the protocol given in [23]. To design our protocols, we have used new tools which provides us improved efficiency. In the information theoretic model, other problems like cardinality set intersection, set union, etc. are worthy of consideration.

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] D. Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *J. Cryptology*, 4(2):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] 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.

[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 *Proc. of CRYPTO*, pages 87–119, 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. Robust multiparty computation with linear communication complexity. In *Proc. of CRYPTO 2007*, volume 4622 of *LNCS*, pages 572–590. Springer Verlag, 2007.

[13] 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.

[14] 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.

[15] Z. Galil, S. Haber, and M. Yung. Cryptographic computation: Secure faut-tolerant protocols and the public-key model. In *Proc. of CRYPTO*, pages 135–155, 1987.

[16] 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.

[17] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of 19th ACM STOC*, pages 218–229, 1987.

[18] 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.

[19] M. Hirt and U. M. Maurer. Robustness for free in unconditional multi-party computation. In *Proc. of CRYPTO*, pages 101–118, 2001.

[20] 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.

[21] 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.

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

[23] 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.

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

[25] Arpita Patra, Ashish Choudhary, AshwinKumar B.V, and C. Pandu Rangan. On round complexity of unconditionally secure vss. Cryptology ePrint Archive, Report 2008/172, 2008.

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

[27] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

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