# Efficient Statistical Asynchronous Verifiable Secret Sharing and Multiparty Computation with Optimal Resilience[☆]

**Arpita Patra[*,1,2], Ashish Choudhury[1,3], C. Pandu Rangan[1,4]**

**Abstract**

Verifiable Secret Sharing (VSS) is a fundamental primitive used as a building block in many distributed cryptographic tasks, such as Secure Multiparty Computation (MPC) and Byzantine Agreement (BA). An important variant of VSS is Asynchronous VSS (AVSS) which is designed to work over asynchronous networks. AVSS is a two phase (Sharing, Reconstruction) protocol carried out among $n$ parties in the presence of a *computationally unbounded active* adversary, who can corrupt up to $t$ parties. We assume that every two parties in the network are directly connected by a pairwise secure channel.

In this paper, we present a new statistical AVSS protocol with *optimal resilience*; i.e. with $n = 3t + 1$. Our protocol privately communicates [5] $\mathcal{O}((\ell n^3 + n^4 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits and A-casts[6] $\mathcal{O}(n^3 \log(n))$ bits to simultaneously share $\ell \geq 1$ elements from a finite field $\mathbb{F}$, where $\epsilon$ is the error parameter of our protocol.

There are only two known statistical AVSS protocols with $n = 3t+1$ reported in [22] and [61]. The AVSS protocol of [22] requires a private communication of $\mathcal{O}(n^9(\log \frac{1}{\epsilon})^4)$ bits and A-cast of $\mathcal{O}(n^9(\log \frac{1}{\epsilon})^2 \log(n))$ bits to share a *single* element from $\mathbb{F}$. Thus our AVSS protocol shows a significant improvement in

[☆]This is an extended and elaborate version of [62]

[*]Corresponding author

*Email addresses:* arpitapatra_10@yahoo.co.in, arpitapatra10@gmail.com (Arpita Patra), partho_31@yahoo.co.in, partho31@gmail.com (Ashish Choudhary), prangan55@gmail.com, prangan55@yahoo.com (C. Pandu Rangan)

[1]Department of Computer Science and Engineering, IIT Madras, Chennai India 600036.

[2]Financial support from Microsoft Research India acknowledged. The author would also like to thank the organizing committee of ICITS 2009 for financial support to attend the conference, where a preliminary version of the paper was presented.

[3]Financial Support from Infosys Technology India Acknowledged. The author would also like to thank IARCS for providing financial support to attend ICITS 2009, where a preliminary version of the paper was presented.

[4]Work Supported by Project No. CSE/05/06/076/DITX/CPAN on Protocols for Secure Computation and Communication, Sponsored by Department of Information Technology, Govt. of India.

[5]Communication over secure channels

[6]A-cast is the asynchronous equivalent of broadcast in synchronous world. A-cast allows a party to send a value to all other parties identically.

communication complexity over the AVSS of [22]. The AVSS protocol of [61] requires a private communication and A-cast of $\mathcal{O}((\ell n^3 + n^4) \log \frac{1}{\epsilon})$ bits to share $\ell \geq 1$ elements. However, the shared element(s) may be $NULL \notin \mathbb{F}$. Thus our AVSS is better than the AVSS of [61] due to the following reasons:

1. The A-cast communication of our AVSS is *independent* of the number of secrets i.e. $\ell$;
2. Our AVSS makes sure that the shared value(s) always belong to $\mathbb{F}$.

Using our AVSS, we design a new primitive called Asynchronous Complete Secret Sharing (ACSS) which acts as an important building block of *asynchronous multiparty computation* (AMPC). Using our ACSS scheme, we design a statistical AMPC protocol with *optimal resilience*; i.e., with $n = 3t+1$, that privately communicates $\mathcal{O}(n^5 \log \frac{1}{\epsilon})$ bits *per multiplication gate*. This significantly improves the communication complexity of only known optimally resilient statistical AMPC of [15] that privately communicates $\Omega(n^{11}(\log \frac{1}{\epsilon})^4)$ bits and A-cast $\Omega(n^{11}(\log \frac{1}{\epsilon})^2 \log(n))$ bits per multiplication gate. Both our ACSS and AVSS employ several new techniques, which are of independent interest.

*Key words:* Asynchronous Networks, AVSS, Optimal Resilience, AMPC, Information Theoretic Security.

---

## 1. Introduction

VSS is one of the fundamental building blocks for many secure distributed computing tasks, such as multiparty computation (MPC) [2, 12, 3, 4, 5, 13, 9, 10, 11, 15, 7, 20, 23, 27, 28, 31, 42, 41, 45, 47, 49, 50, 65, 67, 69, 60], Byzantine Agreement (BA) [37, 22, 55, 1, 61], etc. Any VSS scheme consists of a pair of protocols (Sh, Rec). Protocol Sh [5] allows a special party called *dealer* (denoted as $D$), to share a secret $s \in \mathbb{F}$ (an element from a finite field $\mathbb{F}$) among a set of $n$ parties in a way that allow for a unique reconstruction of $s$ by every party using protocol Rec [6]. Moreover, if $D$ is *honest*, then the secrecy of $s$ is preserved till the end of Sh.

Over the last three decades, active research has been carried out in this area by several researchers, and many interesting and significant results have been obtained dealing with high efficiency, security against general adversaries, security against mixed types of corruptions, long-term security, provable security, etc (see [24, 33, 53, 7, 42, 13, 23, 34, 35, 66, 27, 15, 22, 67, 38, 39, 41, 54, 59, 9, 11, 45, 60, 29, 26, 19, 43, 16, 64, 37, 36, 6, 21, 14, 32, 40, 58, 70, 18, 44] and their references). However, almost all of these solutions are for the synchronous model, where it is assumed that every message in the network is delayed at most by a given constant. This assumption is very strong because a single delayed message would completely break down the overall security of the protocol.

[5] Sh is the protocol for sharing phase of AVSS scheme
[6] Rec is the protocol for reconstruction phase of AVSS scheme

Therefore, VSS protocols for the synchronous model are not suited for real world networks like the Internet.

Later, VSS protocols for the asynchronous network were developed [15, 22]. Here, messages are allowed to be delayed arbitrarily. However, in comparison to the VSS in synchronous settings, research in VSS in the asynchronous settings has attracted much less attention. Known asynchronous VSS protocols are of theoretical interest only and involve high communication complexity. Surprisingly, the techniques used for designing efficient VSS protocols in synchronous networks cannot be adapted directly to the asynchronous setting. This motivated us to design asynchronous VSS protocols taking a fresh look at the problem.

### *1.1. Model*

In this paper, we follow the network model of [19]. Specifically, we assume that an AVSS protocol is carried out among a set of $n$ parties, say $\mathcal{P} = \{P_1, \ldots, P_n\}$, where every two parties are directly connected by a secure channel and $t$ out of the $n$ parties can be under the influence of a *computationally unbounded Byzantine (active) adversary*, denoted as $\mathcal{A}_t$. The Byzantine adversary $\mathcal{A}_t$ completely dictates the parties under its control and can force them to deviate from a protocol, in any arbitrary manner. We assume $\mathcal{A}_t$ to be *rushing* [59, 39, 27], who first listens all the messages sent to the corrupted parties by the honest parties, before allowing the corrupted parties to send their messages. The parties not under the influence of $\mathcal{A}_t$ are called *honest or uncorrupted*. We assume that there is a specific party in $\mathcal{P}$, called the *dealer* $D$, who wants to share the secret in AVSS protocol.

The underlying network is asynchronous, where the communication channels between the parties have arbitrary, yet finite delay (i.e the messages are guaranteed to reach eventually). To model this, $\mathcal{A}_t$ is given the power to schedule the delivery of *all* messages in the network. However, $\mathcal{A}_t$ can only schedule the messages communicated between honest parties, without having any access to the contents of the message. In asynchronous network, the inherent difficulty in designing a protocol comes from the fact that when a party does not receive an expected message then he cannot decide whether the sender is corrupted (and did not send the message at all) or the message is just delayed. So a party can not wait to consider the values sent by all parties, as waiting for all of them could turn out to be endless. Hence the values of up to $t$ (potentially honest) parties may have to be ignored. Due to this the protocols in asynchronous network are generally involved in nature and require new set of primitives. For an comprehensive introduction to asynchronous protocols, see [19].

### *1.2. Definitions*

We now give the definition of primitives which are used in this article. For all these primitives, we assume a finite field $\mathbb{F} = GF(2^\kappa)$, where $\epsilon = 2^{-\Omega(\kappa)}$ and $\epsilon$ is the error parameter. Also without loss of generality, we assume $n = \text{poly}(\kappa) = \text{poly}(\log \frac{1}{\epsilon})$. Thus each field element can be represented by $\mathcal{O}(\kappa) = \mathcal{O}(\log \frac{1}{\epsilon})$ bits.

**Definition 1 (Statistical Asynchronous Weak Secret Sharing (AWSS) [61]).** *Let (Sh, Rec) be a pair of protocols in which a dealer $D \in \mathcal{P}$ shares a secret $s$ using Sh. We say that (Sh[7], Rec[8]) is a $t$-resilient statistical AWSS scheme if all the following hold:*

- **Termination***: With probability at least $1 - \epsilon$, the following requirements hold:*
  1. *If $D$ is honest then each honest party will eventually terminate protocol Sh.*
  2. *If some honest party has terminated protocol Sh, then irrespective of the behavior of $D$, each honest party will eventually terminate Sh.*
  3. *If all honest parties have terminated Sh and invoked Rec, then each honest party will eventually terminate Rec.*

- **Correctness***: With probability at least $1 - \epsilon$, the following requirements hold:*
  1. **Correctness 1 (AWSS)***: If $D$ is honest then each honest party upon terminating Rec, outputs the shared secret $s$.*
  2. **Correctness 2 (AWSS)***: If $D$ is faulty and some honest party has terminated Sh, then there exists a unique $s' \in \mathbb{F} \cup \{NULL\}$, such that each honest party upon terminating Rec will output either $s'$ or $NULL$. This property is also called as weak-commitment.*

- **Secrecy***: If $D$ is honest and no honest party has begun executing protocol Rec, then $\mathcal{A}_t$ has no information about $s$.*

**Definition 2 (Statistical AVSS [12, 19]).** *It is same as statistical AWSS except that **Correctness 2 (AWSS)** property is strengthened as follows:*

- **Correctness 2 (AVSS)***: If $D$ is corrupted and some honest party has terminated Sh, then there exists a fixed $s' \in \mathbb{F}$, such that each honest party upon completing Rec, will output only $s'$.*

**Definition 3 ($t$-sharing [9, 11]).** *A value $s \in \mathbb{F}$ is said to be $t$-shared among the parties in $\mathcal{P}$ if there exists a random degree-$t$ polynomial $f(x)$ over $\mathbb{F}$, with $f(0) = s$ such that each (honest) party $P_i \in \mathcal{P}$ holds his share $s_i = f(i)$ of secret $s$. The vector of shares of $s$ corresponding to the honest parties is called $t$-sharing of $s$ and is denoted by $[s]_t$.*

Typically, VSS is used as a tool for generating $t$-sharing of secret. That is, at the end of sharing phase, *each* honest party holds his share of the secret such that shares of *all* honest parties constitute distinct points on a degree-$t$

[7] Sh is the protocol for sharing phase of AWSS scheme
[8] Rec is the protocol for reconstruction phase of AWSS scheme

polynomial. Such VSS protocols are reported in [13, 54]. On the other hand, there are VSS schemes that do not generate $t$-sharing of secret. They only ensure that a unique secret is shared / committed (during sharing phase) which will be uniquely reconstructed during reconstruction phase. Such schemes are presented in [38, 59, 22]. So we call a VSS scheme as *Complete Secret Sharing* (CSS) scheme if it generates $t$-sharing of secret. More formally, we have the following definition:

**Definition 4 (Statistical Asynchronous Complete Secret Sharing (ACSS)).** *The* **termination, correctness** *and* **secrecy** *property of ACSS are same as in AVSS. In addition, ACSS achieves the following completeness property at the end of* Sh *with probability at least* $(1 - \epsilon)$*:*

- **Completeness***: If some honest party terminates* Sh*, then there exists a random degree-t polynomial $f(x)$ over $\mathbb{F}$, with $f(0) = s'$ such that each (honest) party $P_i \in \mathcal{P}$ will eventually hold his share $s_i = f(i)$ of secret $s'$. Moreover, if $D$ is honest, then $s' = s$.*

The above definitions of AWSS, AVSS and ACSS can be extended for secret $S$ containing multiple elements (say $\ell$ with $\ell > 1$) from $\mathbb{F}$.

**Remark 1 (AWSS, AVSS and ACSS with Private Reconstruction).** *The definitions of AWSS, AVSS and ACSS as given above consider "public reconstruction", where all parties publicly reconstruct the secret in* Rec*. A common variant of these definitions consider "private reconstruction", where only some specific party, say $P_\alpha \in \mathcal{P}$, is allowed to reconstruct the secret in* Rec*. As per our requirement in this paper, we present our AWSS and AVSS schemes with only private reconstruction. However, the protocols for public reconstruction for these schemes can be obtained by doing slight modification in the corresponding protocols for "private reconstruction".*

In our protocols, we use A-cast primitive, which is formally defined as follows:

**Definition 5 (A-cast [22]).** *A-cast is an asynchronous broadcast primitive. It was introduced and elegantly implemented by Bracha [17] with $n = 3t+1$ parties. Let $\Pi$ be an asynchronous protocol initiated by a special party (called the sender), having input $m$ (the message to be broadcast). We say that $\Pi$ is a $t$-resilient A-cast protocol if the following hold, for every possible behavior of $\mathcal{A}_t$:*

- **Termination***:*
  1. *If the sender is honest and all the honest parties participate in the protocol, then each honest party will eventually terminate the protocol.*
  2. *Irrespective of the behavior of the sender, if any honest party terminates the protocol then each honest party will eventually terminate the protocol.*

- **Correctness**: *If the honest parties terminate the protocol then they do so with a common output $m^*$. Furthermore, if the sender is honest then $m^* = m$.*

For the sake of completeness, we recall Bracha's A-cast protocol from [19] and present it in Fig. 1.

Figure 1: Bracha's A-cast Protocol with $n = 3t + 1$

Bracha-A-cast$(S, \mathcal{P}, M)$

Code for the sender $S$ (with input $M$): only $S$ executes this code

1. Send message $(MSG, M)$ privately to all the parties.

Code for party $P_i$: every party in $\mathcal{P}$ executes this code

1. Upon receiving a message $(MSG, M)$, send $(ECHO, M)$ privately to all parties.
2. Upon receiving $n - t$ messages $(ECHO, M')$ that agree on the value of $M'$, send $(READY, M')$ privately to all the parties.
3. Upon receiving $t + 1$ messages $(READY, M')$ that agree on the value of $M'$, send $(READY, M')$ privately to all the parties.
4. Upon receiving $n - t$ messages $(READY, M')$ that agree on the value of $M'$, send $(OK, M')$ privately to all the parties, accept $M'$ as the output message and terminate the protocol.

**Theorem 1 ([19]).** *Protocol A-cast privately communicates $\mathcal{O}(\ell n^2)$ bits for an $\ell$ bit message.*

**Notation 1 (Notation for Using A-cast).** *In the rest of the paper, we use the following convention: we say that $P_j$ receives $m$ from the A-cast of $P_i$, if $P_j$ completes the execution of $P_i$'s A-cast (the A-cast protocol where $P_i$ is the sender), with $m$ as the output.*

**Definition 6 (Online Error Correction (OEC)).** *Let $s$ be a secret which is $t$-shared among the parties in $\mathcal{P}$ by a degree-$t$ polynomial $f(x)$. So $f(0) = s$. Let $P_\alpha \in \mathcal{P}$ be a specific party, who wants to reconstruct $s$. Towards this every party $P_i$ sends his share $s_i$ of $s$ to $P_\alpha$. The shares may reach $P_\alpha$ in any arbitrary order. Moreover, up to $t$ of the shares may be incorrect or missing. In such a situation, by applying OEC on the received $s_i$'s, party $P_\alpha$ can get the interpolation polynomial $f(x)$ and reconstruct the secret $s = f(0)$ in an online fashion. The OEC method uses the properties of Reed-Solomon error correcting codes [56] and enables $P_\alpha$ to recognize when the received shares define a unique degree-$t$ interpolation polynomial.*

Since OEC is a very well known asynchronous primitive, we avoid giving complete details here. The interested reader can refer [19] for complete details.

*1.3. Existing Results for Statistical AVSS with Optimal Resilience*

From [22], statistical AVSS tolerating $\mathcal{A}_t$ is possible iff $n \geq 3t+1$. Therefore, any statistical AVSS with $n = 3t + 1$ parties is said to have *optimal resilience*. The known statistical AVSS protocols with optimal resilience are due to [22] and [61]. Both these AVSS schemes were designed to be used for constructing *Asynchronous Byzantine Agreement* (ABA) protocols. In the following, we summarize these two AVSS schemes.

1. The authors of [22] have presented a series of protocols for designing their AVSS scheme. They first designed a tool called *Information Checking Protocol* (ICP) which is used as a black box for another primitive *Asynchronous Recoverable Sharing* (A-RS). Subsequently, using A-RS, the authors have designed an AWSS scheme, which is further used to design a variation of AWSS called *Two & Sum AWSS*. Finally using their *Two & Sum AWSS*, an AVSS scheme was presented. Pictorially, the route taken by AVSS scheme of [22] is as follows: *ICP* $\rightarrow$ *A-RS* $\rightarrow$ *AWSS* $\rightarrow$ *Two & Sum AWSS* $\rightarrow$ *AVSS*. Since the AVSS scheme is designed on top of so many sub-protocols, it becomes highly communication intensive as well as very much involved. The scheme requires a private communication of $\mathcal{O}(n^9(\log \frac{1}{\epsilon})^4)$ bits and A-cast $\mathcal{O}(n^9(\log \frac{1}{\epsilon})^2 \log(n))$ bits [9] to share a *single* element from $\mathbb{F}$. However, the AVSS scheme of [22] does not generate $t$-sharing of the secret. That is, the AVSS scheme of [22] is not an ACSS scheme and hence can not be used directly in AMPC.
2. The authors of [61] used the following simpler route to design their AVSS scheme: *ICP* $\rightarrow$ *AWSS* $\rightarrow$ *AVSS*. Moreover, due to the new design approach used in their ICP, AWSS and AVSS protocol, the AVSS of [61] provides much better communication complexity than the AVSS of [22]. So the AVSS protocol of [61] requires a private communication of $\mathcal{O}((\ell n^3 + n^4) \log \frac{1}{\epsilon})$ bits and A-cast of $\mathcal{O}((\ell n^3 + n^4) \log \frac{1}{\epsilon})$ bits to share $\ell \geq 1$ elements. While the AVSS scheme of [61] is suitable for ABA problem, it is not suitable for AMPC because:
   (a) The AVSS scheme of [61] is not an ACSS scheme.
   (b) In AVSS of [61], a *corrupted* $D$ may choose secrets from $\mathbb{F} \cup \{NULL\}$ rather than from $\mathbb{F}$ only.

*1.4. Our Contribution*

We present a new statistical AVSS scheme with optimal resilience by following the simple route of [61]. In the following table, we compare the communication complexity of our AVSS with the AVSS of [22, 61]. The table also shows the private communication complexity (CC) of the AVSS protocols after simulating A-cast using the protocol of [17].

[9] The communication complexity analysis of the AVSS scheme of [22] was not done earlier and has been recently carried out in [61].

| Ref. | CC in bits | CC in bits using A-cast of [17] | # Secrets |
|---|---|---|---|
| [22] | Private– $\mathcal{O}(n^9(\log \frac{1}{\epsilon})^4)$<br>A-cast– $\mathcal{O}(n^9(\log \frac{1}{\epsilon})^2 \log(n))$ | $\mathcal{O}(n^9(\log \frac{1}{\epsilon})^4 + n^{11}(\log \frac{1}{\epsilon})^2 \log n)$ | 1 |
| [61] | Private– $\mathcal{O}((\ell n^3 + n^4) \log \frac{1}{\epsilon})$<br>A-cast– $\mathcal{O}((\ell n^3 + n^4) \log \frac{1}{\epsilon})$ | $\mathcal{O}((\ell n^5 + n^6) \log \frac{1}{\epsilon})$ | $\ell$ |
| This Article | Private– $\mathcal{O}((\ell n^3 + n^4 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$<br>A-cast– $\mathcal{O}(n^3 \log(n))$ | $\mathcal{O}((\ell n^3 + n^4 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon} + n^5 \log n)$ | $\ell$ |

As shown in the table, our AVSS attains significantly better communication complexity than the AVSS of [22] and [61] for any value of $\ell$. As mentioned in the previous section, the AVSS of [61] has a *weaker* property than the AVSS of this article and [22]: *A corrupted D may choose secrets from* $\mathbb{F} \cup \{NULL\}$. Such an AVSS is sufficient for designing ABA protocols. However, to be applicable for AMPC, we require that AVSS should allow to share secret(s) *only* from $\mathbb{F}$ [15]. Our AVSS achieves this crucial property at a lesser communication cost. Using our AVSS, we design a new ACSS scheme, which is an essential component of *asynchronous multiparty computation* (AMPC) [15]. Though there are CSS schemes in synchronous settings, our ACSS scheme is first of its kind in asynchronous settings with $n = 3t + 1$. In fact, using our ACSS, we construct an efficient statistical AMPC with *optimal resilience*; i.e., with $n = 3t + 1$, which privately communicates $\mathcal{O}(n^5 \log \frac{1}{\epsilon})$ bits *per multiplication gate*. This is a significant improvement over the *only known* statistical AMPC of [15] with $n = 3t + 1$ that privately communicates $\Omega(n^{11}(\log \frac{1}{\epsilon})^4)$ bits and A-cast $\Omega(n^{11}(\log \frac{1}{\epsilon})^2 \log(n))$ bits per multiplication gate.

In order to design AVSS, we first propose a new Information Checking Protocol (ICP) which significantly improves the communication complexity of the ICP of [61]. Using our ICP, we design an AWSS which is inspired by the AWSS of [61]. Finally our AWSS is used in constructing our new AVSS protocol. The design approach of our AVSS and ACSS are the main essence of this article. In sum, our route for constructing the AMPC protocol is as follows: $ICP \rightarrow AWSS \rightarrow AVSS \rightarrow ACSS \rightarrow AMPC$.

### *1.5. Organization of the Paper*

For ease of presentation, we divide the paper into three parts. The first part, consisting of Section 2 to Section 6 deals with the the construction of AVSS protocol and corresponding building blocks, namely ICP and AWSS. For the sake of simplicity, we first present our AWSS and AVSS protocols sharing single secret and then extend them to share multiple secrets. Then second part, consisting of Section 7 and Section 8 presents our ACSS protocol in detail (Again for simplicity, Section 7 presents ACSS sharing single secret and Section 8 extends the ACSS for multiple secrets). Finally, the third part consisting of Section 9 and subsequent sections deals with the design of our AMPC protocol using ACSS scheme as a building block.

## 2. Information Checking Protocol (ICP) and IC Signature

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 [67] who have designed an ICP in *synchronous* settings. The ICP of Rabin et. al. was also used as a tool by Canetti et. al. [22] for designing their AVSS scheme.

As described in [67, 22, 27], 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 $s$ 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 $R$ [67, 27, 22]. 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: a dealer $D \in \mathcal{P}$, an intermediary $INT \in \mathcal{P}$ and the entire set $\mathcal{P}$ acting as verifiers. This will be later helpful in using ICP as a tool in our AWSS protocol. Moreover, in contrast to the existing ICP protocols that deal with single secret, our ICP can deal with *multiple* secrets *concurrently* and thus achieves better communication complexity than multiple execution of ICP dealing with single secret. Note that, as opposed to the case of a single verifier, when multiple verifiers *simultaneously* participate in ICP, we need to distinguish between synchronity and asynchronity of the network. Our ICP is executed in asynchronous settings and thus we refer it as AICP. As in [67, 22], our AICP is also structured into sequence of following three phases:

1. **Generation Phase**: This phase is initiated by $D$. Here $D$ hands over the secret $S$ containing $\ell$ elements from $\mathbb{F}$ to *intermediary* $INT$. In addition, $D$ sends some *authentication information* to $INT$ and some *verification information* to individual verifiers in $\mathcal{P}$.
2. **Verification Phase**: This phase is initiated by $INT$ to acquire an IC Signature on $S$ that will be later accepted by every honest verifiers in $\mathcal{P}$. Depending on the nature of $D$, $INT$ may or may not receive IC Signature from $D$. When $INT$ receives IC Signature, he decides to continue AICP and later participate in **Revelation Phase**. On the other hand, when $INT$ does not receive IC Signature, he aborts AICP and does not participate in **Revelation Phase** later. The IC signature (when $INT$ receives it), denoted by $ICSig(D, INT, \mathcal{P}, S)$ is nothing but the $S$ along with the *authentication information* which is/are held by $INT$ at the end of **Verification Phase**.
3. **Revelation Phase**: This phase is carried out by $INT$ (only when he receives $ICSig(D, INT, \mathcal{P}, S)$ from $D$ by the end of **Verification Phase**) and the verifiers in $\mathcal{P}$. **Revelation Phase** can be presented in two flavors:
   (a) *Public Revelation* of $ICSig(D, INT, \mathcal{P}, S)$ to all the verifiers in $\mathcal{P}$: Here all the verifiers can publicly verify whether $INT$ indeed received IC signature on $S$ from $D$. If they are convinced then every verifier $P_i$ sets $\mathsf{Reveal}_i = S$. Otherwise every $P_i$ sets $\mathsf{Reveal}_i = NULL$.

(b) $P_\alpha$-*private-revelation* of $ICSig(D, INT, \mathcal{P}, S)$: Here $INT$ privately reveals $ICSig(D, INT, \mathcal{P}, S)$ to *only* $P_\alpha$. After doing some checking, if $P_\alpha$ believes that $INT$ indeed received IC signature on $S$ from $D$ then $P_\alpha$ sets $\mathsf{Reveal}_\alpha = S$. Otherwise $P_\alpha$ sets $\mathsf{Reveal}_\alpha = NULL$.

Any AICP should satisfy the following properties, assuming public revelation of signature (these properties are almost same as the properties of ICP defined in [22]). In the properties, $\epsilon$ denotes the error parameter of AICP. In order to bound the error probability by $\epsilon$, any AICP protocol operates over field $\mathbb{F} = GF(2^\kappa)$, where $\epsilon = 2^{-\Omega(\kappa)}$. So $\kappa = \lceil \log \frac{1}{\epsilon} \rceil$.

1. **AICP-Correctness1:** If $D$ and $INT$ are *honest*, then $ICSig(D, INT, \mathcal{P}, S)$ will be accepted in **Revelation Phase** by every *honest* verifier.
2. **AICP-Correctness2:** If an *honest* $INT$ holds an $ICSig(D, INT, \mathcal{P}, S)$ at the end of **Verification Phase**, then $ICSig(D, INT, \mathcal{P}, S)$ will be accepted in **Revelation Phase** by every honest verifier, except with probability $\epsilon$.
3. **AICP-Correctness3:** If $D$ is *honest*, then during **Revelation Phase**, with probability at least $(1 - \epsilon)$, every $ICSig(D, INT, \mathcal{P}, S')$ with $S' \neq S$ produced by a *corrupted* $INT$ will not be accepted by an *honest* verifier.
4. **AICP-Secrecy:** If $D$ and $INT$ are *honest* and $INT$ has not started **Revelation Phase**, then $\mathcal{A}_t$ will have no information about $S$.

For AICP with $P_\alpha$-*private-revelation* in **Revelation Phase**, the above properties can be modified by replacing "every/any honest verifier" with "honest $P_\alpha$".

In the following, we first present an informal idea of our novel AICP called MVMS-AICP and then describe protocol MVMS-AICP in Fig. 2.

**The Intuition behind Protocol MVMS-AICP:** $D$ selects a random polynomial $f(x)$ of degree $\ell + t\kappa$, whose first $\ell$ coefficients are the elements of $S$ and delivers $f(x)$ to $INT$. In addition, to each individual verifier, $D$ privately gives the value of $f(x)$ at $\kappa$ random *evaluation points*. This distribution of information by $D$ helps to achieve **AICP-Correctness3** property. Specifically, if $D$ is *honest*, then a *corrupted* $INT$ cannot produce an *incorrect* $f'(x) \neq f(x)$ during **Revelation Phase** without being detected by an *honest* verifier. This is because a corrupted $INT$ will have no information about the evaluation points of an honest verifier and hence with very high probability, $f'(x)$ will not match with the evaluation points held by an honest verifier.

The above distribution of information by $D$ also maintains **AICP-Secrecy** property. This is because the degree of $f(x)$ is $\ell + t\kappa$ and $\mathcal{A}_t$ will know the value of $f(x)$ at most at $t\kappa$ evaluation points.

However, a *corrupted* $D$ might do the following: he may distribute $f(x)$ to $INT$ and value of some other polynomial (different from $f(x)$) to each honest verifier. To avoid this situation, $INT$ and the verifiers interact in *zero knowledge* fashion, using *cut-and-choose* technique to check the consistency of $f(x)$ and the

values of $f(x)$ held by individual verifier. The specific details of the *cut-and-choose*, along with other formal steps of protocol MVMS-AICP are given in Fig. 2.

Since in our AWSS, we require only $P_\alpha$-private-revelation of $ICSig(D, INT, \mathcal{P}, S)$, we present protocol MVMS-AICP with **Revelation Phase** describing $P_\alpha$-private-revelation of $ICSig(D, INT, \mathcal{P}, S)$.

Figure 2: **AICP with** $n = 3t + 1$**. Here** $\kappa = \lceil \log \frac{1}{\epsilon} \rceil$

**Protocol MVMS-AICP**$(D, INT, \mathcal{P}, S, \epsilon)$

**Generation Phase: Gen(**$D, INT, \mathcal{P}, S, \epsilon$**)**

1. $D$ selects a random $\ell + t\kappa$ degree polynomial $f(x)$ whose lower order $\ell$ coefficients are the secrets in $S = (s^1, \ldots, s^\ell)$. $D$ also picks $n\kappa$ random, non-zero, distinct *evaluation points* from $\mathbb{F}$, denoted by $\alpha_1^i, \ldots, \alpha_\kappa^i$, for $i = 1, \ldots, n$.
2. $D$ privately sends $f(x)$ to $INT$ and the verification tags $z_1^i = (\alpha_1^i, a_1^i), \ldots, z_\kappa^i = (\alpha_\kappa^i, a_\kappa^i)$ to party $P_i$. Here $a_j^i = f(\alpha_j^i)$, for $j = 1, \ldots, \kappa$.

**Verification Phase: Ver(**$D, INT, \mathcal{P}, S, \epsilon$**)**

1. Every verifier $P_i$ randomly partitions the index set $\{1, \ldots, \kappa\}$ into two sets $I^i$ and $\overline{I^i}$ of equal size and sends $I^i$ and $z_j^i$ for all $j \in I^i$ to $INT$.
2. Local Computation (only for $INT$):
   (a) For every verifier $P_i$ from which $INT$ has received $I^i$ and corresponding verification tags, $INT$ checks whether for *every* $j \in I^i$, $f(\alpha_j^i) \stackrel{?}{=} a_j^i$.
   (b) If for at least $2t + 1$ verifiers, the above condition is satisfied, then $INT$ sets $ICSig(D, INT, \mathcal{P}, S) = f(x)$ and concludes that he has received $ICSig(D, INT, \mathcal{P}, S)$ from $D$.
   (c) If for at least $t + 1$ verifiers, the above condition is not satisfied, then $INT$ sets $ICSig(D, INT, \mathcal{P}, S) = NULL$ and concludes that he has not received $ICSig(D, INT, \mathcal{P}, S)$ from $D$.

**Revelation Phase: Reveal-Private(**$D, INT, \mathcal{P}, S, P_\alpha, \epsilon$**)**: *$P_\alpha$-private-revelation* of $ICSig(D, INT, \mathcal{P}, S)$

1. To party $P_\alpha$, $INT$ sends $ICSig(D, INT, \mathcal{P}, S) = f(x)$.
2. To party $P_\alpha$, every verifier $P_i$ sends the index set $\overline{I^i}$ and all $z_j^i$ such that $j \in \overline{I^i}$.
3. Local Computation (only for $P_\alpha$):
   (a) Upon receiving $f(x)$ from $INT$ and the values from verifier $P_i$, check whether for *some* $j \in \overline{I^i}$, $f(\alpha_j^i) \stackrel{?}{=} a_j^i$.
   (b) If for at least $t + 1$ verifiers the condition is satisfied, then accept $ICSig(D, INT, \mathcal{P}, S)$ and set $\mathsf{Reveal}_\alpha = S$, where $S$ is lower order $\ell$ coefficients of $f(x)$.
   (c) If for at least $2t + 1$ verifiers the above condition is not satisfied, then reject $ICSig(D, INT, \mathcal{P}, S)$ and set $\mathsf{Reveal}_\alpha = NULL$.

We now prove the properties of protocol MVMS-AICP.

**Lemma 1 (AICP-Correctness1).** *If $D$, $INT$ and $P_\alpha$ are honest, then $S$ will be accepted by $P_\alpha$.*

PROOF: If $D$ is honest then he will honestly deliver $f(x)$ to $INT$ and its value at $\kappa$ points to individual verifier. So eventually, the condition stated in step 2(a) of **Verification Phase** will be satisfied for at least $2t+1$ verifiers and hence $INT$, who is honest in this case will set $ICSig(D, INT, \mathcal{P}, S) = f(x)$. Now it is easy to see that the condition stated in step 3(a) of **Revelation Phase** will be eventually satisfied, corresponding to the honest verifiers in $\mathcal{P}$ (there are at least $2t+1$ honest verifiers). Hence $P_\alpha$, who is honest in this case, will eventually accept $ICSig(D, INT, \mathcal{P}, S)$ at the end of **Revelation phase**. □

**Lemma 2 (AICP-Correctness2).** *If an honest $INT$ holds an $ICSig(D, INT, \mathcal{P}, S)$ at the end of* **Verification Phase***, then $ICSig(D, INT, \mathcal{P}, S)$ will be accepted in* **Revelation Phase** *by honest $P_\alpha$, except with probability $\epsilon$.*

PROOF: We have to consider the case when $D$ *is corrupted* as otherwise the proof will follow from Lemma 1. Since $INT$ is honest and it holds an $ICSig(D, INT, \mathcal{P}, S)$ at the end of **Verification phase**, $INT$ has ensured that for at least $2t+1$ verifiers the condition specified in step 2(a) of **Verification phase** has been satisfied. Let $\mathcal{H}$ be the set of *honest* verifiers among these $2t+1$ verifiers. Note that $|\mathcal{H}| \geq t+1$. To prove the lemma, we prove that corresponding to each verifier in $\mathcal{H}$, the condition stated in step 3(a) of **Revelation Phase** will be satisfied with very high probability. Note that corresponding to a verifier $P_i$ in $\mathcal{H}$, the condition stated in step 3(a) of **Revelation Phase** will fail if for *all* $j \in \overline{I^i}$, $f(\alpha_j^i) \neq a_j^i$. This implies that (corrupted) $D$ must have distributed $f(x)$ (to $INT$) and $z_j^i$ (to $P_i$) *inconsistently* for *all* $j \in \overline{I^i}$ and it so happens that $P_i$ has partitioned $\{1, \ldots, \kappa\}$ into $I^i$ and $\overline{I^i}$ during **Verification Phase**, such that $\overline{I_i}$ contains only inconsistent tuples ($z_j^i$'s). Thus corresponding to a verifier $P_i \in \mathcal{H}$, the probability that the condition stated in step 3(a) of **Revelation Phase** fails is same as the probability of $P_i$ selecting all consistent (inconsistent) tuples in $I^i$ ($\overline{I^i}$), which is $\frac{1}{\binom{\kappa}{\kappa/2}} \approx 2^{-\Omega(\kappa)}$. Now as there are at least $t+1$ parties in $\mathcal{H}$, except with probability $(t+1)2^{-\Omega(\kappa)} \approx \epsilon$, $P_\alpha$ will eventually find step 3(a) of **Revelation Phase** to be true for all parties in $\mathcal{H}$ and will accept $ICSig(D, INT, \mathcal{P}, S)$. □

**Lemma 3 (AICP-Correctness3).** *If $D$ is honest, then during* **Revelation Phase***, with probability at least $(1-\epsilon)$, every $ICSig(D, INT, \mathcal{P}, S')$ with $S' \neq S$ produced by a corrupted $INT$ will be rejected by honest verifier $P_\alpha$.*

PROOF: It is easy to see that $S' \neq S$ produced by a *corrupted* $INT$ will be accepted by an *honest* $P_\alpha$, if the condition stated in step 3(a) of **Revelation Phase** gets satisfied corresponding to *at least one honest* verifier (for $t$ corrupted verifiers, the condition may always satisfy). However, the condition will be satisfied corresponding to honest verifier $P_i$ if corrupted $INT$ can *correctly guess* a verification tag $z_i^j$ for at least one $j \in \overline{I^i}$, which he can do with probability $\frac{1}{|\mathbb{F}|} = 2^{-\Omega(\kappa)} = \epsilon$. □

**Lemma 4 (AICP-Secrecy).** *If $D$ and $INT$ are honest and $INT$ has not started* **Revelation Phase***, then $S$ is information theoretically secure from $\mathcal{A}_t$.*

PROOF: If $D$ and $INT$ are honest, then at the end of **Verification Phase**, $\mathcal{A}_t$ will get $t\kappa$ distinct values on $f(x)$. However, $f(x)$ is of degree $\ell + t\kappa$ and hence the lower order $\ell$ coefficients of $f(x)$ which are the elements of $S$ will remain information theoretically secure. □

**Lemma 5 (AICP-Communication-Complexity).** *Protocol* Gen *privately communicates* $\mathcal{O}((\ell + n\log\frac{1}{\epsilon})\log\frac{1}{\epsilon})$ *bits. Protocol* Ver *privately communicates* $\mathcal{O}((n\log\frac{1}{\epsilon})\log\frac{1}{\epsilon})$ *bits. Protocol* Reveal-Private *privately communicates* $\mathcal{O}((\ell + n\log\frac{1}{\epsilon})\log\frac{1}{\epsilon})$ *bits.*

PROOF: In protocol Gen, $D$ privately gives $\ell + t\kappa$ field elements to $INT$ and $\kappa$ field elements to each verifier. Since each field element can be represented by $\mathcal{O}(\kappa)$ bits and $\kappa = \lceil\log\frac{1}{\epsilon}\rceil$, protocol Gen incurs a private communication of $\mathcal{O}((\ell + n\log\frac{1}{\epsilon})\log\frac{1}{\epsilon})$ bits. In protocol Ver, every verifier privately sends $\frac{\kappa}{2}$ field elements to $INT$, thus incurring a total private communication of $\mathcal{O}((n\log\frac{1}{\epsilon})\log\frac{1}{\epsilon})$ bits. In protocol Reveal-Private, $INT$ sends to $P_\alpha$ the polynomial $f(x)$, consisting of $\ell + t\kappa$ field elements, while each verifier sends $\overline{I^i}$ and corresponding verification tags. So Reveal-Private involves private communication of $\mathcal{O}((\ell + n\log\frac{1}{\epsilon})\log\frac{1}{\epsilon})$ bits. □

**Theorem 2.** *Protocol* Multi-Verifier-AICP *is an efficient AICP.*

PROOF: The theorem follows from Lemma 1, Lemma 2, Lemma 3 and Lemma 4.

**Notation 2 (Notation for Using Multi-Verifier-AICP).** *Recall that $D$ and $INT$ can be any party from $\mathcal{P}$. In the sequel we use the following convention. We say that:*

1. *"$P_i$ sends $ICSig(P_i, P_j, \mathcal{P}, S)$ to $P_j$ with error parameter $\epsilon$" to mean that $P_i$ acting as dealer $D$ and considering $P_j$ as $INT$, executes* Gen$(P_i, P_j, \mathcal{P}, S, \epsilon)$*;*
2. *"$P_i$ receives $ICSig(P_j, P_i, \mathcal{P}, S)$ from $P_j$ with error parameter $\epsilon$" to mean that $P_i$ as $INT$ has received $ICSig(P_j, P_i, \mathcal{P}, S)$ after executing* Ver$(P_j, P_i, \mathcal{P}, S, \epsilon)$*;*
3. *"$P_i$ reveals $ICSig(P_j, P_i, \mathcal{P}, S)$ to $P_\alpha$ with error parameter $\epsilon$" to mean $P_i$ as $INT$ executes* Reveal-Private$(P_j, P_i, \mathcal{P}, S, P_\alpha, \epsilon)$ *along with the participation of the verifiers in $\mathcal{P}$;*
4. *"$P_\alpha$ completes revelation of $ICSig(P_j, P_i, \mathcal{P}, S)$ with* Reveal$_\alpha = S$*" to mean that $P_\alpha$ has successfully completed* Reveal-Private$(P_j, P_i, \mathcal{P}, S, P_\alpha, \epsilon)$ *with* Reveal$_\alpha = S$*.*

## 3. Statistical AWSS Scheme for Sharing a Single Secret

We now present an AWSS scheme, called AWSS with $n = 3t + 1$, consisting of a pair of protocols (AWSS-Share, AWSS-Rec-Private). While AWSS-Share allows $D$ to share *a single secret* $s$ among $\mathcal{P}$, AWSS-Rec-Private enables private reconstruction of $s$ or $NULL$ by a specific party, say $P_\alpha \in \mathcal{P}$. We call the private

reconstruction as $P_\alpha$-*weak-private-reconstruction*. In AWSS-Share, a *corrupted* $D$ may commit to $s = NULL$ instead of an element from $\mathbb{F}$ (the meaning of it will be clear in the sequel).

Our AWSS-Share protocol is similar to AWSS-Share protocol given in [61]. However, instead of using the AICP of [61], we use MVMS-AICP presented in this paper in AWSS-Share. This leads to better communication complexity.

**High Level Idea of AWSS-Share:** We follow the general strategy used in [13, 27, 39, 38, 54] for synchronous settings for sharing the secret $s$ with a symmetric bivariate polynomial $F(x, y)$ of degree-$t$ in $x$ and $y$, where each party $P_i$ gets the univariate polynomial $f_i(x) = F(x, i)$. So in AWSS-Share, $D$ chooses a symmetric bivariate polynomial $F(x, y)$ of degree-$t$ in $x$ and $y$ such that $F(0, 0) = s$. $D$ then hands over $ICSig(D, INT, \mathcal{P}, f_i(j))$ for every $j = 1, \ldots, n$ to $P_i$. This step implicitly implies that $P_i$ will receive $f_i(x)$ from $D$. After receiving these IC signatures from $D$, the parties then exchange IC signature on their common values (a pair $(P_i, P_j)$ has one common value, namely $F(i, j)$; $P_i$ has $f_i(j)$ and $P_j$ has $f_j(i)$ where $F(i, j) = f_i(j) = f_j(i)$). Then $D$, in conjunction with all other parties, perform a sequence of communication and computation. As a result of this, at the end of AWSS-Share, every party agrees on a set of $2t + 1$ parties, called $WCORE$, such that every party $P_j \in WCORE$ is *IC-committed* to $f_j(0)$ using $f_j(x)$ to a set of $2t+1$ parties, called as $OKP_j$. $P_j$ is *IC-committed* to $f_j(0)$ using $f_j(x)$ among the parties in $OKP_j$ only when every $P_k \in OKP_j$ received (a) $ICSig(D, P_k, \mathcal{P}, f_k(j))$ and (b) $ICSig(P_j, P_k, \mathcal{P}, f_j(k))$ and ensures $f_k(j) = f_j(k)$ (this should ideally hold due to the selection and distribution of symmetric bivariate polynomial). In some sense, we may view this as every $P_j \in WCORE$ is attempting to commit his received (from $D$) polynomial $f_j(x)$ among the parties in $OKP_j$ (by giving his *IC Signature* on one point of $f_j(x)$ to each party) and the parties in $OKP_j$ allowing him to do so after verifying that they have got $D$'s IC signature on the same value of $f_j(x)$. We will show that later in the reconstruction phase, every honest $P_j$'s (in $WCORE$) *IC-commitment* will be reconstructed correctly irrespective of whether $D$ is honest or corrupted. Moreover, a corrupted $P_j$'s *IC-commitment* will be reconstructed correctly when $D$ is honest. But on the other hand, a corrupted $P_j$'s *IC-commitment* can be reconstructed to any value when $D$ is corrupted. These properties are at the heart of our AWSS protocol.

Achieving the agreement (among the parties) on $WCORE$ and corresponding $OKP_j$s is a bit tricky in asynchronous network. Even though these sets are constructed on the basis of information that are A-casted by parties, parties may end up with different versions of $WCORE$ and $OKP_j$'s while attempting to generate them locally, due to the asynchronous nature of the network. We solve this problem by asking $D$ to construct $WCORE$ and $OKP_j$s based on A-casted information and then ask $D$ to A-cast the same. After receiving $WCORE$ and $OKP_j$s from the A-cast of $D$, individual parties ensure the validity of these sets by receiving the same A-cast using which $D$ would have formed these sets. A similar approach was used in the protocols of [1]. Protocol AWSS-Share is formally presented in Fig. 3.

Figure 3: **Sharing Phase of Protocol AWSS for Sharing a Single Secret $s$ with $n = 3t + 1$**

## Protocol AWSS-Share($D, \mathcal{P}, s, \epsilon$)

DISTRIBUTION: CODE FOR $D$ – Only $D$ executes this code.

1. Select a random, symmetric bivariate polynomial $F(x, y)$ of degree-$t$ in $x$ and $y$, such that $F(0, 0) = s$. For $i = 1, \ldots, n$, let $f_i(x) = F(x, i)$.
2. For $i = 1, \ldots, n$, send $ICSig(D, P_i, \mathcal{P}, f_i(j))$ to $P_i$ with error parameter $\epsilon' = \frac{\epsilon}{n^2}$ for each $j = 1, \ldots, n$.

VERIFICATION: CODE FOR $P_i$ – Every party including $D$ executes this code.

1. Wait to receive $ICSig(D, P_i, \mathcal{P}, f_i(j))$ with error parameter $\epsilon'$ for each $j = 1, \ldots, n$ from $D$.
2. Check if $(f_i(1), \ldots, f_i(n))$ defines degree-$t$ polynomial. If yes then send $ICSig(P_i, P_j, \mathcal{P}, f_i(j))$ to $P_j$ with error parameter $\epsilon'$ for all $j = 1, \ldots, n$.
3. If $ICSig(P_j, P_i, \mathcal{P}, f_j(i))$ is received from $P_j$ with error parameter $\epsilon'$ and if $f_i(j) = f_j(i)$, then A-cast $\texttt{OK}(P_i, P_j)$.

WCORE CONSTRUCTION : CODE FOR $D$ – Only $D$ executes this code.

1. For each $P_j$, build a set $OKP_j = \{P_k | D$ receives $\texttt{OK}(P_k, P_j)$ from the A-cast of $P_k\}$. When $|OKP_j| = 2t + 1$, then $P_j$'s *IC-commitment* on $f_j(0)$ is over (or we may say that $P_j$ is *IC-committed* to $f_j(0)$) and add $P_j$ in $WCORE$ (which is initially empty).
2. Wait until $|WCORE| = 2t + 1$. Then A-cast $WCORE$ and $OKP_j$ for all $P_j \in WCORE$.

WCORE VERIFICATION & AGREEMENT ON WCORE : CODE FOR $P_i$

1. Wait to obtain $WCORE$ and $OKP_j$ for all $P_j \in WCORE$ from $D$'s A-cast, such that $|WCORE| = 2t + 1$ and $|OKP_j| = 2t + 1$ for each $P_j \in WCORE$.
2. Wait to receive $\texttt{OK}(P_k, P_j)$ for all $P_k \in OKP_j$ and $P_j \in WCORE$. After receiving all these OKs, accept the $WCORE$ and $OKP_j$'s received from $D$ and terminate AWSS-Share.

Before moving into the discussion and description of AWSS-Rec-Private, we now define what we call as $D$'s AWSS-commitment.

**Remark 2 ($D$'s AWSS-commitment).** *We say that $D$ is AWSS-committed to a secret $s \in \mathbb{F}$ in AWSS-Share if there is a unique degree-$t$ univariate polynomial $f(x)$ such that $f(0) = s$ and every honest $P_i$ in $WCORE$ receives $f(i)$ from $D$ and IC-commits to $f(i)$ among the parties in $OKP_i$. Otherwise, we say that $D$ has committed $NULL$. An honest $D$ always commits $s$ from $\mathbb{F}$ as in this case $f(x)$ is $f_0(x)(= F(x, 0))$, where $F(x, y)$ is the symmetric bivariate polynomial of degree-$t$ in $x$ and $y$, chosen by honest $D$. Moreover, every honest party $P_i$ in $WCORE$ will receive $f_0(i)$ which is same as $f_i(0)$ (this can be obtained from $f_i(x)$). But AWSS-Share can not ensure that corrupted $D$ also commits $s \in \mathbb{F}$. This means that a corrupted $D$ may distribute information to the parties such*

*that, polynomial $f_0(x)$ defined by the $f_0(i)(= f_i(0))$ values possessed by honest $P_i$'s in $WCORE$ may not be a degree-t polynomial. In this case we say $D$ has AWSS-committed $NULL$.*

Our discussion in the sequel will show that for a corrupted $D$, irrespective of the behavior of the corrupted parties, either $D$'s *AWSS-committed* secret $s$ (which belongs to $\mathbb{F} \cup \{NULL\}$) or NULL will be reconstructed by honest $P_\alpha$.

**High Level Idea of AWSS-Rec-Private:** In AWSS-Rec-Private, the parties in $WCORE$ and corresponding $OKP_j$'s are used in order to reconstruct $D$'s AWSS-committed secret. Specifically, for every $P_j \in WCORE$, $P_j$'s *IC-commitment* ($f_j(0)$) is reconstructed by asking every party $P_k \in OKP_j$ to reveal $ICSig(D, P_k,$
$\mathcal{P}, f_k(j))$ and $ICSig(P_j, P_k, \mathcal{P}, f_j(k))$ such that $f_k(i) = f_j(k)$. Since there are at least $t+1$ honest parties in $OKP_j$, eventually at least $t+1$ $f_j(k)$'s will be revealed with which $f_j(x)$ and thus $f_j(0)$ will be reconstructed. Then $f_j(0)$'s are used to construct the univariate polynomial $f_0(x)$ that is committed by $D$ during AWSS-Share.

Asking $P_k \in OKP_j$ to reveal $D$'s IC signature ensures that if $D$ is *honest*, then even for a *corrupted* $P_j \in WCORE$, the reconstructed polynomial $f_j(x)$ will be same as the one handed over by $D$ to $P_j$ in sharing phase (that is, a corrupted $P_j$'s IC-commitment $f_j(0)$ will be reconstructed correctly). This helps our AWSS protocol to satisfy **Correctness 1** property of AWSS. Now asking $P_k$ in $OKP_j$ to reveal $P_j$'s signature ensures that even if $D$ is *corrupted*, for an *honest* $P_j \in WCORE$, the reconstructed polynomial $f_j(x)$ will be same as the one received by $P_j$ from $D$ in AWSS-Share (that is, an honest $P_j$'s IC-commitment $f_j(0)$ will be reconstructed correctly even though $D$ is corrupted). This helps to ensure **Correctness 2** property. Summing up, when at least one of $D$ and $P_j$ is honest, $P_j$'s *IC-commitment* (i.e $f_j(0)$) will be revealed properly. But when both $D$ and $P_j$ are corrupted, $P_j$'s *IC-commitment* can be revealed as any $\overline{f_j(0)}$ which may or may not be equal to $f_j(0)$. It is the later property that makes our protocol to qualify as a AWSS protocol rather than a AVSS protocol. Protocol AWSS-Rec-Private is formally given in Fig. 4.

The proof of the properties of our AWSS scheme follows using similar arguments as given for the AWSS scheme of [61]. However, for the sake of completeness we recall them here.

**Lemma 6 (AWSS-Termination).** *Protocols (AWSS-Share, AWSS-Rec-Private) satisfy termination property of Definition 1.*

PROOF: **Termination 1:** When $D$ is honest then eventually all honest parties will receive desired IC signatures from $D$ and will also eventually exchange IC signatures on their common values and will A-cast OK for each other. Hence every honest $P_j$ will eventually complete his *IC-commitment* on $f_j(0)$ with at least $2t+1$ honest parties in $OKP_j$. So $D$ will eventually include $2t+1$ parties in $WCORE$ (of which at least $t+1$ are honest) and A-cast the same. Now by

Figure 4: **Reconstruction Phase of Protocol AWSS Scheme for Sharing a Single Secret $s$ with $n = 3t + 1$**

**Protocol AWSS-Rec-Private($D, \mathcal{P}, s, P_\alpha, \epsilon$): $P_\alpha$-weak-private-reconstruction of $s$**

SIGNATURE REVELATION: CODE FOR $P_i$ — Every party executes this code

1. If $P_i$ belongs to $OKP_j$ for some $P_j \in WCORE$, then reveal $ICSig(D, P_i, \mathcal{P}, f_i(j))$ and $ICSig(P_j, P_i, \mathcal{P}, f_j(i))$ to $P_\alpha$, each with error parameter $\epsilon'$.

LOCAL COMPUTATION: CODE FOR $P_\alpha$ — Only $P_\alpha$ executes this code

1. For every $P_j \in WCORE$, reconstruct $P_j$'s *IC-commitment*, say $\overline{f_j(0)}$ as follows:
   (a) Construct a set $ValidP_j = \emptyset$.
   (b) Add $P_k \in OKP_j$ to $ValidP_j$ if the following conditions hold:
      i. Revelation of $ICSig(D, P_k, \mathcal{P}, f_k(j))$ and $ICSig(P_j, P_k, \mathcal{P}, f_j(k))$ are completed with $\mathsf{Reveal}_\alpha = \overline{f_k(j)}$ and $\mathsf{Reveal}_\alpha = \overline{f_j(k)}$; and
      ii. $\overline{f_k(j)} = \overline{f_j(k)}$.
   (c) Wait until $|ValidP_j| = t + 1$. Construct a polynomial $\overline{f_j(x)}$ passing through the points $(k, \overline{f_j(k)})$ where $P_k \in ValidP_j$. Associate $\overline{f_j(0)}$ with $P_j \in WCORE$.
2. Wait for $\overline{f_j(0)}$ to be reconstructed for every $P_j$ in $WCORE$.
3. Check whether the points $(j, \overline{f_j(0)})$ for $P_j \in WCORE$ lie on a unique degree-$t$ polynomial $\overline{f_0(x)}$. If yes, then set $\overline{s} = \overline{f_0(0)}$ and terminate AWSS-Rec-Private. Else set $\overline{s} = NULL$ and terminate AWSS-Rec-Private.

the property of A-cast, each honest party will eventually receive $WCORE$ from the A-cast of $D$. Finally, since honest $D$ had included $P_j$ in $WCORE$ after receiving the OK signals from the parties in $OKP_j$'s, each honest party will also receive the same and will eventually terminate AWSS-Share.

**Termination 2:** If an honest $P_i$ has terminated AWSS-Share, then he must have received $WCORE$ and $OKP_j$'s from the A-cast of $D$ and verified their validity by receiving the desired A-casts. By properties of A-cast, each honest party will also receive the same and will eventually terminate AWSS-Share.

**Termination 3:** Since each of the IC signatures are given with an error parameter $\epsilon' = \frac{\epsilon}{n^2}$, if $P_i$ (acting as $INT$) is honest and has received an IC signature, then IC signature produced by $P_i$ during Reveal-Private will be accepted by honest $P_\alpha$ without any error probability when $D$ is honest (by **AICP-Correctness1** i.e Lemma 1) and except with probability $\epsilon'$ when $D$ is corrupted (by **AICP-Correctness2** i.e Lemma 2). Since for every $P_j \in WCORE$, $|OKP_j| = 2t + 1$, there are at least $t + 1$ honest parties in $OKP_j$ and each of them may be present in $ValidP_j$ except with probability $\epsilon'$. Thus except with probability at most $n^2\epsilon' = \epsilon$, $P_j$'s *IC-commitment* will be reconstructed for

all $P_j \in WCORE$. So except with probability $\epsilon$, *honest* $P_\alpha$ will terminate AWSS-Rec-Private after executing remaining steps of [LOCAL COMPUTATION] (as specified in protocol AWSS-Rec-Private). □

**Lemma 7 (AWSS-Secrecy).** *Protocol AWSS-Share satisfies secrecy property of Definition 1.*

PROOF: We have to consider the case when $D$ is honest. The proof follows from the secrecy of protocol MVMS-AICP and properties of symmetric bivariate polynomial of degree-$t$ in $x$ and $y$ [25]. Specifically, without loss of generality, assume that $P_1, \ldots, P_t$ are the parties under the control of $\mathcal{A}_t$. So during the execution of AWSS-Share, $\mathcal{A}_t$ will know $f_1(x), \ldots, f_t(x)$ and $t$ points on $f_{t+1}(x), \ldots, f_n(x)$. However, $\mathcal{A}_t$ still lacks one more point to uniquely interpolate $F(x, y)$. Hence, $s = F(0, 0)$ will be information theoretically secure. □

**Lemma 8 (AWSS-Correctness).** *Protocols (AWSS-Share, AWSS-Rec-Private) satisfy correctness property of Definition 1.*

PROOF: **Correctness 1:** Here we have to consider the case when $D$ is *honest*. We show that $D$'s *AWSS-commitment* will be reconstructed correctly by honest $P_\alpha$, except with probability $\epsilon$. We prove the lemma by showing that when $D$ is *honest*, $P_j$'s *IC-commitment* $f_j(0)$ will be correctly reconstructed for $P_j \in WCORE$, except with probability $\frac{\epsilon}{n}$, irrespective of whether $P_j$ is honest or corrupted. Consequently, as $|WCORE| = 2t + 1$, all honest parties will reconstruct $f_0(x) = F(x, 0)$ and hence the secret $s = f_0(0)$ with probability at least $(1 - (2t + 1)\frac{\epsilon}{n}) \approx (1 - \epsilon)$. So we consider the following two cases:

1. Consider an honest $P_j$ in $WCORE$. From **AICP-Correctness3** (Lemma 3), a corrupted $P_k \in OKP_j$ will be able to successfully produce $ICSig(P_j, P_k, \mathcal{P}, \overline{f_j(k)})$ such that $\overline{f_j(k)} \neq f_j(k)$, with probability at most $\epsilon'$. As there can be at most $t$ corrupted parties in $ValidP_j$, except with probability $t\epsilon' = \frac{\epsilon}{n}$, the value $\overline{f_j(k)}$ is same as $f_j(k)$ for all $P_k \in ValidP_j$. Hence honest $P_j$'s *IC-commitment* $f_j(0)$ will be correctly reconstructed, except with probability $\frac{\epsilon}{n}$.
2. Consider a corrupted $P_j$ in $WCORE$. Now a corrupted $P_k \in OKP_j$ will be able to produce $ICSig(D, P_k, \mathcal{P}, \overline{f_k(j)})$ such that $\overline{f_k(j)} \neq f_k(j)$, with probability at most $\epsilon'$ according to **AICP-Correctness3**. Thus except with probability $t\epsilon' = \frac{\epsilon}{n}$, corresponding to a corrupted $P_j \in WCORE$, the parties in $ValidP_j$ have produced correct points on $f_j(x)$.

**Correctness 2:** Here we consider the case, when $D$ is *corrupted*. Now there are two cases: (a) $D$'s *AWSS-committed* secret $s$ belongs to $\mathbb{F}$; (b) $D$'s *AWSS-committed* secret $s$ is $NULL$. Whatever may be case, we show that except with probability $\epsilon$, honest $P_\alpha$ will either reconstruct $s$ or $NULL$.

1. We first consider the case when $s \in \mathbb{F}$. This implies that the $f_j(0)$ values received by the honest parties in $WCORE$ lie on a degree-$t$ polynomial

$f_0(x)$. Moreover every honest $P_j$ in $WCORE$ is *IC-committed* to $f_j(0)$. We now show that in AWSS-Rec-Private, *IC-commitment* of all honest parties in $WCORE$ will be reconstructed correctly by $P_\alpha$ with probability at least $(1-\epsilon)$. So let $P_j$ be an honest party in $WCORE$. Now from **AICP-Correctness3**, a corrupted $P_k \in OKP_j$ can not produce $ICSig(P_j, P_k, \mathcal{P}, \overline{f_j(k)})$ such that $\overline{f_j(k)} \neq f_j(k)$ except with probability $\epsilon'$. Hence for honest $P_j$ in $WCORE$, $f_j(x)$ and thus $f_j(0)$ will be reconstructed correctly, except with probability $t\epsilon'$. As there are at least $t+1$ honest parties in $WCORE$, the probability that the above event happens for all honest parties in $WCORE$ is at most $t(t+1)\epsilon' \approx \epsilon$. So *IC-commitment* of all honest parties in $WCORE$ will be reconstructed correctly by $P_\alpha$ with probability at least $(1-\epsilon)$.
However, for a *corrupted* $P_j$ in $WCORE$, $P_j$'s *IC-commitment* can be revealed to any value $\overline{f_j}(0)$. This is because a corrupted $P_k \in OKP_j$ can produce a valid signature of $P_j$ on any $\overline{f_j(k)}$ as well as a valid signature of $D$ (who is corrupted as well) on $\overline{f_k(j)} = \overline{f_j(k)}$. Also the adversary can delay the messages such that the values of corrupted $P_k \in OKP_j$ are revealed to $P_\alpha$ before the values of honest parties in $OKP_j$. Now if reconstructed $\overline{f_j(0)} = f_j(0)$ for all corrupted $P_j \in WCORE$, then $s$ will be reconstructed by $P_\alpha$. Otherwise, $NULL$ will be reconstructed. However, since for all the honest parties of $WCORE$, *IC-commitment* will be reconstructed correctly with probability at least $(1-\epsilon)$ (who in turn define $f_0(x)$), no other secret (other than $s$) can be reconstructed by $P_\alpha$.

2. We next consider the second case when $D$'s AWSS-committed secret is $NULL$. This implies that the points $(j, f_j(0))$ corresponding to honest $P_j$'s in $WCORE$ do not define a unique degree-$t$ polynomial. It is easy to see that in this case, irrespective of the behavior of the corrupted parties $NULL$ will be reconstructed. This is because the points $f_j(0)$ corresponding to all honest $P_j \in WCORE$ will be reconstructed correctly except with probability $\epsilon$ (following the argument given in previous case).

□

**Lemma 9 (AWSS-Communication-Complexity).** *Protocol AWSS-Share incurs a private communication of $\mathcal{O}(n^3(\log \frac{1}{\epsilon})^2)$ bits and A-cast of $\mathcal{O}(n^2 \log n)$ bits. Protocol AWSS-Rec-Private privately communicates $\mathcal{O}(n^3(\log \frac{1}{\epsilon})^2)$ bits.*

PROOF: In AWSS-Share, there are $\mathcal{O}(n^2)$ instances of Gen and Ver (of MVMS-AICP), each dealing with one value (substituting $\ell = 1$) and executed with an error parameter of $\epsilon' = \frac{\epsilon}{n^2}$. From Theorem 5, this requires a private communication of $\mathcal{O}(n^3(\log \frac{n^2}{\epsilon})^2) = \mathcal{O}(n^3(\log \frac{1}{\epsilon})^2)$ bits, as $n = \text{poly}(\log \frac{1}{\epsilon})$. Moreover, there are A-cast of $\mathcal{O}(n^2)$ OK signals. In addition, there is A-cast of $WCORE$ containing the identity of $2t+1$ parties and $OK$ sets corresponding to each party in $WCORE$, where each $OK$ set contains the identity of $2t+1$ parties. Now the identity of a party can be represented by $\mathcal{O}(\log n)$ bits. So in total,

AWSS-Share incurs a private communication of $\mathcal{O}(n^3(\log \frac{1}{\epsilon})^2)$ bits and A-cast of $\mathcal{O}(n^2 \log n)$ bits.

In AWSS-Rec-Private, there are $\mathcal{O}(n^2)$ instances of Reveal-Private of our MVMS-AICP, each dealing with $\ell = 1$ value. This requires a private communication of $\mathcal{O}(n^3(\log \frac{1}{\epsilon})^2)$ bits. □

**Theorem 3.** *Protocol AWSS consisting of (AWSS-Share, AWSS-Rec-Private) constitutes a valid statistical AWSS scheme with $n = 3t + 1$ parties with private reconstruction.*

PROOF: The proof follows from Lemma 6, Lemma 7 and Lemma 8. □

**Notation 3 (Notation for Using AWSS-Share, AWSS-Rec-Private).** *In our AVSS scheme (that shares a single secret), we will invoke AWSS-Share as AWSS-Share$(D, \mathcal{P}, f(x), \epsilon)$ to mean that $D$ commits to $f(x)$ in AWSS-Share. Essentially here $D$ is asked to choose a symmetric bivariate polynomial $F(x, y)$ of degree-$t$ in $x$ and $y$, where $F(x, 0) = f(x)$ holds. $D$ then tries to give $F(x, i)$ and hence $F(0, i) = f(i)$ to party $P_i$. Similarly, AWSS-Rec-Private will be invoked as AWSS-Rec-Private$(D, \mathcal{P}, f(x), P_\alpha, \epsilon)$ for $P_\alpha$-weak-private-reconstruction of $f(x)$.*

## 4. Statistical AWSS Scheme for Sharing Multiple Secrets

In this section, we extend protocol AWSS-Share and AWSS-Rec-Private to AWSS-MS-Share and AWSS-MS-Rec-Private respectively [10]. Now our new AWSS scheme called AWSS-MS consists of (AWSS-MS-Share, AWSS-MS-Rec-Private). Protocol AWSS-MS-Share allows $D \in \mathcal{P}$ to concurrently share a secret $S = (s^1 \dots s^\ell)$, containing $\ell$ elements. On the other hand, protocol AWSS-MS-Rec-Private allows a specific party $P_\alpha \in \mathcal{P}$ to reconstruct either $S$ or $NULL$.

Notice that we could have executed protocol AWSS-Share $\ell$ times parallely, each sharing individual elements of $S$. However, from Lemma 9 this would incur a private communication of $\mathcal{O}(\ell n^3(\log \frac{1}{\epsilon})^2)$ bits and A-cast of $\mathcal{O}(\ell n^2 \log n)$ bits. On the other hand, AWSS-MS-Share shares all elements of $S$ concurrently, requiring a private communication of $\mathcal{O}((\ell n^2 + n^3 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits and A-cast of A-cast of $\mathcal{O}(n^2 \log n)$ bits. Thus for sufficiently large $\ell$, the communication complexity of AWSS-MS-Share is less than what would have been required by $\ell$ parallel executions of AWSS-Share. Similarly, protocol AWSS-MS-Rec-Private reconstructs all the $\ell$ secrets simultaneously, incurring a private communication of $\mathcal{O}((\ell n^2 + n^3 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits.

**The Intuition:** The high level idea of protocol AWSS-MS-Share is similar to AWSS-Share. For each $s^l, l = 1, \dots, \ell$, the dealer $D$ selects a random symmetric bivariate polynomial $F^l(x, y)$ of degree-$t$ in $x$ and $y$, where $F^l(0, 0) = s^l$ and gives his IC signature on $f_i^l(1), \dots, f_i^l(n)$ to party $P_i$, for $i = 1, \dots, n$. For

[10] Here MS stands for multiple secrets

this, $D$ can execute $n$ instances of Gen, one for each $f_i^l(j)$, for $j = 1, \ldots, n$ (this approach was used in AWSS-Share). However, this would require a total of $\ell n$ instances of Gen (each dealing with a *single* secret) to be executed by $D$ for every party $P_i$. Clearly, this would require a private communication of $\mathcal{O}(\ell n^2(\log \frac{1}{\epsilon})^2)$ bits. Instead of this, a better solution would be to ask $D$ to execute $n$ instances of Gen, where in the $j^{th}$ instance, $D$ gives his IC signature *collectively* on $(f_i^1(j), f_i^2(j), \ldots, f_i^l(j))$ to party $P_i$. This requires private communication of $\mathcal{O}((\ell n + n^2 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits.

Next each party $P_i$ tries to *IC-commit* $(f_i^1(0), \ldots, f_i^\ell(0))$ simultaneously. For this, every pair of parties $P_i$ and $P_j$ privately exchange $(f_i^1(j), \ldots, f_i^\ell(j))$ and $(f_j^1(i), \ldots, f_j^\ell(i))$, along with their respective IC signature on these values. Again notice that $P_i$ and $P_j$ pass on their IC signature *collectively* on $(f_i^1(j), \ldots, f_i^\ell(j))$ and $(f_j^1(i), \ldots, f_j^\ell(i))$ respectively. Next the parties pair-wise check whether $f_i^l(j) = f_j^l(i)$ *for all* $l = 1, \ldots, \ell$ and if so they A-cast OK signal. After this, the remaining steps (like $WCORE$ construction, agreement on $WCORE$, etc) are same as in AWSS-Share. So essentially, the differences between AWSS-Share and AWSS-MS-Share are: (1) the way the parties give their IC signature and (2) the conditions required for A-casting OK signal. Protocol AWSS-MS-Share is formally given in Fig. 5.

**Remark 3 ($D$'s AWSS-commitment).** *We say that $D$ is AWSS-committed to $S = (s^1, \ldots, s^\ell) \in \mathbb{F}^\ell$ if for every $l = 1, \ldots, \ell$ there is a unique degree-$t$ polynomial $f^l(x)$ such that $f^l(0) = s^l$ and every honest $P_i$ in $WCORE$ receives $f^l(i)$ from $D$ and IC-commits $f^l(i)$ among the parties in $OKP_i$. Otherwise, we say that $D$ is AWSS-committed to $NULL$. An honest $D$ always AWSS-commits $S \in \mathbb{F}^\ell$ as in this case $f^l(x) = f_0^l(x) = F^l(x, 0)$, where $F^l(x, y)$ is the symmetric bivariate polynomial of degree-$t$ in $x$ and $y$ chosen by $D$. But AWSS-MS-Share can not ensure that corrupted $D$ also AWSS-commits $S \in \mathbb{F}^\ell$. This means that a corrupted $D$ may distribute information to the parties such that, polynomial $f_0^l(x)$ defined by the $f_0^l(i)(= f_i^l(0))$ values possessed by honest $P_i$'s in $WCORE$ may not be a degree-$t$ polynomial for some $l$. In this case we say $D$ has AWSS-committed $NULL$.*

Protocol AWSS-MS-Rec-Private is a straightforward extension of protocol AWSS-Rec-Private and is given in Fig. 6.

Since technique wise, protocols (AWSS-MS-Share, AWSS-MS-Rec-Private) are very similar to protocols (AWSS-Share, AWSS-Rec-Private), we do not provide the proofs of the properties of protocols (AWSS-MS-Share, AWSS-MS-Rec-Private) for the sake of avoiding repetition. Rather, we give the following theorem on the communication complexity.

**Theorem 4 (AWSS-MS-Communication -Complexity).** *Protocol AWSS-MS-Share incurs a private communication of $\mathcal{O}((\ell n^2 + n^3 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits and A-cast of $\mathcal{O}(n^2 \log n)$ bits. Protocol AWSS-MS-Rec-Private involves private communication of $\mathcal{O}((\ell n^2 + n^3 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits.*

Figure 5: **Sharing Phase of Protocol AWSS-MS for Sharing $S$ Containing $\ell \geq 1$ Secrets**

**AWSS-MS-Share($D, \mathcal{P}, S = (s^1 \dots s^\ell), \epsilon$)**

DISTRIBUTION: CODE FOR $D$ – Only $D$ executes this code.

1. For $l = 1, \dots, \ell$, select a random, symmetric bivariate polynomial $F^l(x, y)$ of degree-$t$ in $x$ and $y$ such that $F^l(0, 0) = s^l$. Let $f_i^l(x) = F^l(x, i)$, for $l = 1, \dots, \ell$.
2. For $i = 1, \dots, n$, send $ICSig(D, P_i, \mathcal{P}, (f_i^1(j), \dots, f_i^\ell(j))$ with error parameter $\epsilon' = \frac{\epsilon}{n^2}$ for each $j = 1, \dots, n$ to $P_i$.

VERIFICATION: CODE FOR $P_i$ – Every party including $D$ executes this code.

1. Wait to receive $ICSig(D, P_i, \mathcal{P}, (f_i^1(j), \dots, f_i^\ell(j)))$ with error parameter $\epsilon'$ for $j = 1, \dots, n$ from $D$.
2. Check if $(f_i^l(1), \dots, f_i^l(n))$ defines degree-$t$ polynomial for every $l = 1, \dots, \ell$. If yes then send $ICSig(P_i, P_j, \mathcal{P}, (f_i^1(j), \dots, f_i^\ell(j)))$ with error parameter $\epsilon'$ to $P_j$ for all $j = 1, \dots, n$.
3. If $ICSig(P_j, P_i, \mathcal{P}, (f_j^1(i), \dots, f_j^\ell(i)))$ is received from $P_j$ with error parameter $\epsilon'$ and if $f_j^l(i) = f_i^l(j)$ for all $l = 1, \dots, \ell$, then A-cast $\mathtt{OK}(P_i, P_j)$.

WCORE CONSTRUCTION : CODE FOR $D$ – Only $D$ executes this code.

1. For each $P_j$, build a set $OKP_j = \{P_i | D$ receives $\mathtt{OK}(P_i, P_j)$ from the A-cast of $P_i\}$. When $|OKP_j| = 2t + 1$, then $P_j$'s *IC-commitment* on $(f_j^1(0), \dots, f_j^\ell(0))$ is over (or we may say that $P_j$ is *IC-committed* to $(f_j^1(0), \dots, f_j^\ell(0))$) and add $P_j$ in $WCORE$ (which is initially empty).
2. Wait until $|WCORE| = 2t + 1$. Then A-cast $WCORE$ and $OKP_j$ for all $P_j \in WCORE$.

WCORE VERIFICATION & AGREEMENT ON WCORE : CODE FOR $P_i$

1. Wait to obtain $WCORE$ and $OKP_j$ for all $P_j \in WCORE$ from $D$'s A-cast, such that $|WCORE| = 2t + 1$ and $|OKP_j| = 2t + 1$ for each $P_j \in WCORE$.
2. Wait to receive $\mathtt{OK}(P_k, P_j)$ for all $P_k \in OKP_j$ and $P_j \in WCORE$. After receiving all these OKs, accept the $WCORE$ and $OKP_j$'s received from $D$ and terminate AWSS-MS-Share.

PROOF: The proof follows from the fact that in AWSS-MS-Share and AWSS-MS-Rec-Private, there are $\mathcal{O}(n^2)$ instances of MVMS-AICP, each dealing with $\ell$ values and having an error parameter of $\epsilon' = \frac{\epsilon}{n^2}$. □

**Notation 4 (Notation for Using AWSS-MS-Share, AWSS-MS-Rec-Private).** *We will invoke AWSS-MS-Share as AWSS-MS-Share($D, \mathcal{P}, (f^1(x), \dots, f^\ell(x)), \epsilon$) where $D$ is asked to choose symmetric bivariate polynomials $F^1(x, y), \dots, F^\ell(x, y)$ each of degree-$t$ in $x$ and $y$ such that $F^l(x, 0) = f^l(x)$ holds for $l = 1, \dots, \ell$. $D$ then tries to give $F^l(x, i)$ and hence $F^l(0, i) = f^l(i)$ to party $P_i$, for $l = 1, \dots, \ell$. Similarly, AWSS-MS-Rec-Private will be invoked as AWSS-MS-Rec-Private($D, \mathcal{P}, (f^1(x), \dots, f^\ell(x)), P_\alpha, \epsilon$). to enable the $P_\alpha$-weak-private-reconstruction of $(f^1(x), \dots, f^\ell(x))$.*

Figure 6: **Reconstruction Phase of Protocol AWSS-MS for Sharing $S$ Containing $\ell$ Secrets**

**AWSS-MS-Rec-Private($D, \mathcal{P}, S = (s^1, \ldots, s^\ell), \epsilon$)**
**$P_\alpha$-weak-private-reconstruction of $S$**

SIGNATURE REVELATION: CODE FOR $P_i$ — Every party executes this code

1. If $P_i$ belongs to $OKP_j$ for some $P_j \in WCORE$, then reveal $ICSig(D, P_i, \mathcal{P}, (f_i^1(j), \ldots, f_i^\ell(j)))$ and $ICSig(P_j, P_i, \mathcal{P}, (f_j^1(i), \ldots, f_j^\ell(i)))$, each with error parameter $\epsilon'$ .

LOCAL COMPUTATION: CODE FOR $P_\alpha$ — Only $P_\alpha$ executes this code

1. For every $P_j \in WCORE$, reconstruct $P_j$'s *IC-commitment*, say $(\overline{f_j^1}(0), \ldots, \overline{f_j^\ell}(0))$ as follows:
   (a) Construct a set $ValidP_j = \emptyset$.
   (b) Add $P_k \in OKP_j$ to $ValidP_j$ if the following conditions hold:
      i. Revelation of $ICSig(D, P_k, \mathcal{P}, (f_k^1(j), \ldots, f_k^\ell(j)))$ and $ICSig(P_j, P_k, \mathcal{P}, (f_j^1(k), \ldots, f_j^\ell(k)))$ are completed with $\mathsf{Reveal}_\alpha = (\overline{f_k^1}(j), \ldots, \overline{f_k^\ell}(j))$ and $\mathsf{Reveal}_\alpha = (\overline{f_j^1}(k), \ldots, \overline{f_j^\ell}(k))$ respectively; and
      ii. $\overline{f_k^l}(j) = \overline{f_j^l}(k)$, for $l = 1, \ldots, \ell$.
   (c) Wait until $|ValidP_j| = t+1$. For $l = 1, \ldots, \ell$, construct a degree-$t$ polynomial $\overline{f_j^l}(x)$ passing through the points $(k, \overline{f_j^l}(k))$ where $P_k \in ValidP_j$. For $l = 1, \ldots, \ell$, associate $\overline{f_j^l}(0)$ with $P_j \in WCORE$.
2. Wait for $\overline{f_j^1}(0), \ldots, \overline{f_j^\ell}(0)$ to be reconstructed for every $P_j$ in $WCORE$.
3. For $l = 1, \ldots, \ell$, do the following:
   (a) Check whether the points $(j, \overline{f_j^l}(0))$ for $P_j \in WCORE$ lie on a unique degree-$t$ polynomial $\overline{f_0^l}(x)$. If yes, then set $\overline{s^l} = \overline{f_0^l}(0)$, else set $\overline{s^l} = NULL$.
4. If $\overline{s^l} = NULL$ for any $l \in \{1, \ldots, \ell\}$, then output $\overline{S} = NULL$ and terminate AWSS-MS-Rec-Private. Else output $\overline{S} = (\overline{s^1}, \ldots, \overline{s^\ell})$ and terminate AWSS-MS-Rec-Private.

## 5. Statistical AVSS Protocol for Sharing a Single Secret

We now present an AVSS scheme called AVSS consisting of a pair of protocols (AVSS-Share, AVSS-Rec-Private). AVSS-Share allows $D$ to share a *single secret* from $\mathbb{F}$. *Notice that unlike AWSS-Share (presented in Section 3), AVSS-Share ensures that even a corrupted $D$ commits to a secret from $\mathbb{F}$.* Protocol AVSS-Rec-Private allows a specific party, say $P_\alpha$, to *privately* reconstruct $D$'s committed secret. We call the private reconstruction as *$P_\alpha$-private-reconstruction*. While *$P_\alpha$-private-reconstruction* can always ensure that $P_\alpha$ reconstructs $D$'s committed secret with high probability, *$P_\alpha$-weak-private-reconstruction* (introduced in Section 3) could only ensure that $P_\alpha$ reconstructs either $D$'s committed secret or $NULL$. Structurally, we divide AVSS-Share into a sequence of following three phases. *Each of the phases will be eventually completed by every honest party when $D$ is honest. Moreover, if some honest party completes all the three phases*

*then eventually every other honest party will also complete all the three phases.*

1. **Commitment by $D$:** Here $D$ on having a secret $s$, commits the secret by AWSS-committing $n$ shares of $s$ using $n$ different instances of AWSS-Share protocol.
2. **Verification of $D$'s commitment:** Here the parties verify whether indeed $D$ has committed a secret from $\mathbb{F}$.
3. **Re-commitment by Individual Parties:** If the parties are convinced in previous phase, then every party $P_i$ re-commits his share of $D$'s committed secret using an instance of AWSS-Share protocol.

While first two phases of AVSS-Share are enough to ensure that $D$ has committed a secret from $\mathbb{F}$, the sole purpose of third phase is to enable robust reconstruction of $D$'s committed secret in AVSS-Rec-Private. That is if protocol AVSS-Share stops after the second phase, then we may only ensure that either $D$'s committed secret or $NULL$ will be reconstructed in AVSS-Rec-Private. This would violate the claim that AVSS is an AVSS scheme.

*5.1. Commitment by D Phase*

In this phase, $D$ on having a secret $s$, selects a random bivariate polynomial $F(x, y)$ of degree-$(t, t)$ (i.e degree-$t$ in both $x$ and $y$) such that $F(0, 0) = s$. Now to party $P_i$, $D$ passes $f_i(x) = F(x, i)$ and $g_i(y) = F(i, y)$. We refer $f_i(x)$ polynomials as *row polynomials* and $g_i(y)$ polynomials as *column polynomials*. Now $D$ commits $f_1(x), \ldots, f_n(x)$ using $n$ distinct invocations of AWSS-Share protocol (see Notation 3 in Section 3 for the interpretation of committing polynomial using AWSS-Share). During the course of executing these $n$ instances of AWSS-Share, a party $P_i$ receives $i^{th}$ point on the polynomials $f_1(x), \ldots, f_n(x)$, namely $f_1(i), \ldots, f_n(i)$ which should be $n$ distinct points on $g_i(y)$. So $P_i$ checks whether $g_i(j) = f_j(i)$ for all $j = 1, \ldots, n$ and informs this by A-casting a signal. While executing the $n$ instances of AWSS-Share, $D$ employ a trick to guarantee that all the $n$ instances of AWSS-Share terminate with a common $WCORE$. Once $WCORE$ is agreed among all the honest parties in $\mathcal{P}$, **Commitment by $D$ Phase** ends. The code for this phase is presented in Fig. 7.

We now prove the properties of **Commitment by $D$ Phase**.

**Lemma 10.** *In Code Commitment:*

1. *If $D$ is honest then eventually he will generate a common $WCORE$ of size $2t + 1$ for all the $n$ instances of AWSS-Share initiated by him. Moreover, each honest party will eventually agree on the common $WCORE$.*
2. *If $D$ is corrupted and some honest party has accepted the $WCORE$ and $OKP_j$s received from the A-cast of $D$, then every other honest party will also eventually accept the same.*

PROOF: In Code Commitment, $D$ keeps on adding new parties in each $WCORE^i$ and $OKP_j^i$ even after their cardinality reach $2t + 1$. So if $D$ is honest, then eventually he will include all the $2t + 1$ honest parties in each $WCORE^i$ and

Figure 7: Code for Commitment by $D$ Phase

## Code **Commitment**$(D, \mathcal{P}, s, \epsilon)$

i. Distribution by $D$: – Only $D$ executes this code

1. Select a random degree-$(t,t)$ bivariate polynomial $F(x,y)$ such that $F(0,0) = s$.
2. For $i = 1, \ldots, n$, send *row polynomial* $f_i(x) = F(x,i)$ and *column polynomial* $g_i(y) = F(i,y)$ to $P_i$.
3. For $i = 1, \ldots, n$, initiate AWSS-Share$(D, \mathcal{P}, f_i(x), \epsilon')$ for sharing $f_i(x)$, where $\epsilon' = \frac{\epsilon}{n}$.

ii. Code for $P_i$ – Every party in $\mathcal{P}$, including $D$, executes this code

1. Wait to receive degree-$t$ row polynomial $f_i(x)$ and degree-$t$ column polynomial $g_i(y)$ from $D$.
2. Participate in AWSS-Share$(D, \mathcal{P}, f_j(x), \epsilon')$ by executing steps in [Verification: Code for $P_i$] (of AWSS-Share) for all $j = 1, \ldots, n$.
3. After the completion of step 1 of [Verification: Code for $P_i$] for all the $n$ invocations of AWSS-Share, check whether $g_i(j) = f_j(i)$ holds for all $j = 1, \ldots, n$. Here $f_j(i)$ is obtained by $P_i$ from $D$ during the execution of first step of [Verification: Code for $P_i$] of AWSS-Share$(D, \mathcal{P}, f_j(x), \epsilon')$. If yes then A-cast `Matched-Column` and execute the rest of the steps of AWSS-Share$(D, \mathcal{P}, f_j(x), \epsilon')$, for all $j = 1, \ldots, n$.

iii. WCORE Construction: Code for $D$ – Only $D$ executes this code.

1. Construct $WCORE$ and corresponding $OKP_j$'s for each AWSS-Share$(D, \mathcal{P}, f_i(x), \epsilon')$ following the steps in [WCORE Construction] (of AWSS-Share). Denote them by $WCORE^i$ and $OKP_j^i$'s.
2. Keep updating $WCORE^i$'s and corresponding $OKP_j^i$'s. That is, even after $WCORE^i$ reaches the size of $2t+1$, $D$ keeps on adding new parties in $WCORE^i$ and corresponding $OK$ sets after receiving new A-cast of the form `OK`(*, *) in AWSS-Share$(D, \mathcal{P}, f_i(x), \epsilon')$.
3. Wait to obtain $WCORE = \cap_{i=1}^{n} WCORE^i$ of size at least $2t+1$ and for every $P_j \in WCORE$, $OKP_j = \cap_{i=1}^{n} OKP_j^i$ of size at least $2t+1$, such that `Matched-Column` is received from A-cast of every $P_j \in WCORE$ .
4. A-cast $WCORE$ and $OKP_j$ for every $P_j \in WCORE$.

iv. WCORE verification & Agreement: Code for $P_i$ – Every party including $D$ will execute this code.

1. Wait to receive $WCORE$ and $OKP_j$ for every $P_j \in WCORE$ from A-cast of $D$, such that $|WCORE| \geq 2t+1$ and each $|OKP_j| \geq 2t+1$.
2. Wait to receive $OK(P_k, P_j)$ from the A-cast of $P_k$ for every $P_k \in OKP_j$ and every $P_j \in WCORE$ for all the $n$ executions of AWSS-Share.
3. Wait to receive `Matched-Column` from A-cast of every $P_j \in WCORE$.
4. After receiving all desired `OK`s and `Matched-Column` signals, accept $WCORE$ and $OKP_j$ for every $P_j \in WCORE$ received from A-cast of $D$ and proceed to the next phase (**Verification of $D$'s Commitment Phase**).

$OKP_j^i$ for every honest $P_j$. Moreover, each honest $P_i$ will eventually A-cast `Matched-Column` signal, as $f_j(i) = g_i(j)$ will hold for all $j = 1, \ldots, n$ when $D$ is *honest*. Therefore, if $D$ is honest then eventually he will find a common set of at least $2t+1$ parties in the $WCORE^i$ of all the $n$ instances of AWSS-Share, who have A-cast `Matched-Column` signal. This common set of at least

$2t+1$ parties will form the common $WCORE$ which $D$ will A-cast. Similarly, corresponding to each $P_j \in WCORE$, the honest $D$ will eventually find a common set of at least $2t+1$ parties in $OKP_j^1, \ldots, OKP_j^n$. This common set of at least $2t+1$ parties will constitute $OKP_j$ which $D$ will A-cast. Now from the property of A-cast, each honest party will eventually receive $WCORE$ of size at least $2t+1$ and $OKP_j$ of size at least $2t+1$ for each $P_j \in WCORE$ from the A-cast of $D$. Now it is easy to see that each honest party will accept this common $WCORE$ and $OKP_j$ for each $P_j \in WCORE$, after executing the steps in [WCORE VERIFICATION AND AGREEMENT]. This proves the first part.

If $D$ is corrupted and some honest party, say $P_i$ has accepted the $WCORE$ and $OKP_j$'s received from the A-cast of $D$ then it implies the following: $P_i$ has received $OK(P_k, P_j)$ from the A-cast of $P_k$ for every $P_k \in OKP_j$ and every $P_j \in WCORE$ for all the $n$ executions of AWSS-Share. Moreover, $P_i$ has also received `Matched-Column` from A-cast of every $P_j \in WCORE$. Now from the property of A-cast, every other honest party $P_j$ will also eventually receive the same `OK`s and `Matched-Column` and hence will accept the $WCORE$ and $OKP_j$ for each $P_j \in WCORE$.

*5.2. Verification of D's Commitment Phase*

After agreeing on $WCORE$ and corresponding $OKP_j$'s, in this phase, the parties verify whether indeed $D$ has committed a secret from $\mathbb{F}$. For this, the parties try to check whether there is a set $\mathcal{R}$ of size at least $2t+1$ and another set $\mathcal{C}$ of size at least $2t+1$ (possibly different from $\mathcal{R}$), such that for every $P_i \in \mathcal{R}$ and every $P_j \in \mathcal{C}$, $f_i(j) = g_j(i)$ holds. If they can ensure that such sets exist then it implies that the row and column polynomials of the *honest* parties in $\mathcal{R}$ and $\mathcal{C}$ define a unique bivariate polynomial of degree-$(t,t)$ and the constant term of the polynomial is $D$'s committed secret. Checking for the existence of such sets is quiet easy in synchronous settings, where the parties can simply pairwise exchange common values on their row and column polynomial, as done in several synchronous VSS protocols [13, 39, 38, 54, 59]. However, doing the same is not so straightforward in asynchronous settings, especially when we have only $3t+1$ parties.

To check the existence of the sets described above, the parties proceed as follows: recall that in the **Commitment by $D$ phase**, $D$ is committed to $f_1(x), \ldots, f_n(x)$ using $n$ distinct instances of AWSS-Share. Now the parties execute AWSS-Rec-Private$(D, \mathcal{P}, f_j(x), P_j, \epsilon')$ for enabling $P_j$-weak-private-reconstruction of $f_j(x)$. If $P_j$ reconstructs $\overline{f_j}(x)$ from the execution of AWSS-Rec-private and $\overline{f_j}(x)$ is same as $f_j(x)$ received from $D$ in the previous phase, then $P_j$ informs this to everyone by A-casting `Matched-Row` signal. This is a public notification by $P_j$ that the polynomial committed by $D$ in AWSS-Share$(D, \mathcal{P}, f_j(x), P_j, \epsilon')$ is same as the one which $P_j$ has privately received from $D$. Now if at least $2t+1$ parties, say $\mathcal{R}$, A-cast `Matched-Row`, then it implies that $D$ is committed to a unique degree-$(t,t)$ bivariate polynomial, say $\overline{F}(x,y)$ (hence a unique secret $\overline{s} = \overline{F}(0,0)$) such that for every *honest* $P_i \in \mathcal{R}$, the row polynomial $f_i(x)$ held by $P_i$ satisfies $\overline{F}(x,i) = f_i(x)$ and for every *honest* $P_j \in WCORE$, the column polynomial $g_j(y)$ held by $P_j$ satisfies $\overline{F}(j,y) = g_j(y)$ (For proof see

Lemma 11). The code for implementing this phase is very easy and is given in Fig. 8.

Figure 8: Code for Verification of $D$'s Commitment Phase

**Code Verification($D, \mathcal{P}, s, \epsilon$)**

$P_j$-Weak-Private-Reconstruction of $f_j(x)$ for $j = 1, \ldots, n$:

i. Code for $P_i$ – Every party in $\mathcal{P}$ executes this code.

1. After agreeing on $WCORE$ and corresponding $OKP_j$'s, participate in AWSS-Rec-Private$(D, \mathcal{P}, f_j(x), P_j, \epsilon')$, for $j = 1, \ldots, n$, to enable $P_j$-weak-private-reconstruction of $f_j(x)$. Notice that the same $WCORE$ and $OKP_j$ for $P_j \in WCORE$ are used in each AWSS-Rec-Private$(D, \mathcal{P}, f_j(x), P_j, \epsilon')$, for $j = 1, \ldots, n$
2. At the completion of AWSS-Rec-Private$(D, \mathcal{P}, f_i(x), P_i, \epsilon')$, obtain either degree-$t$ polynomial $\overline{f_i}(x)$ or $NULL$.
3. If $f_i(x) = \overline{f_i}(x)$, then A-cast `Matched-Row`.

ii. Code for $D$ — Only executes this code.

1. Wait to receive `Matched-Row` from A-cast of at least $2t + 1$ parties, say $\mathcal{R}$.
2. A-cast the set $\mathcal{R}$.

iii. [Verification of $D$'s Commitment] : Code for $P_i$ – Every party in $\mathcal{P}$ executes this code

1. Wait to receive $\mathcal{R}$ of size at least $2t + 1$ from A-cast of $D$.
2. Wait to receive `Matched-Row` from A-cast of every party in $\mathcal{R}$ and then proceed to third phase.

**Lemma 11.** *In Code Verification, if an honest party receives* `Matched-Row` *from the A-cast of the parties in $\mathcal{R}$, then in code Commitment, $D$ is committed to a unique degree-$(t, t)$ bivariate polynomial $\overline{F}(x, y)$ (and hence to a secret $\overline{s} = \overline{F}(0, 0)$) such that the row polynomial $f_i(x)$ held by every honest $P_i \in \mathcal{R}$ satisfies $\overline{F}(x, i) = f_i(x)$ and the column polynomial $g_j(y)$ held by every honest $P_j \in WCORE$ satisfies $\overline{F}(j, y) = g_j(y)$. Moreover if $D$ is honest then $\overline{F}(x, y) = F(x, y)$ and hence $\overline{s} = s$.*

Proof: Let $l$ and $m$ be the number of honest parties in $\mathcal{R}$ and $WCORE$ respectively. As $|WCORE| \geq 2t + 1$ and $|\mathcal{R}| \geq 2t + 1$, both $l \geq t + 1$ and $m \geq t + 1$. For convenience, we assume $P_1, \ldots, P_l$ and respectively $P_1, \ldots, P_m$ are the set of honest parties in $\mathcal{R}$ and $WCORE$. Now for every $(P_i, P_j)$ with $P_i \in \{P_1, \ldots, P_l\}$ and $P_j \in \{P_1, \ldots, P_m\}$, $f_i(j) = g_j(i)$ holds. This is due to the fact that $P_i$ has checked that $D$ is indeed committed to $f_i(x)$ (by checking $f_i(x) = \overline{f_i}(x)$, where $\overline{f_i}(x)$ is obtained from $P_i$-weak-private-reconstruction and $f_i(x)$ is obtained from $D$ in Commitment). The above implies that honest $P_j \in WCORE$ has received $f_i(j)$ from $D$ and checked $g_j(i) = f_i(j)$ during the execution of Commitment. We now claim that if $f_i(j) = g_j(i)$ holds for every $(P_i, P_j)$ with $P_i \in \{P_1, \ldots, P_l\}$ and $P_j \in \{P_1, \ldots, P_m\}$ then there exists a unique bivariate polynomial $\overline{F}(x, y)$ of degree-$(t, t)$, such that for $i = 1, \ldots, l$, we have $\overline{F}(x, i) =$

$f_i(x)$ and for $j = 1, \ldots, m$, we have $\overline{F}(j, y) = g_j(y)$. The proof now completely follows from the proof of Lemma 4.26 of [19].

Specifically, let $V^{(k)}$ denote $k \times k$ Vandermonde matrix, where $i^{th}$ column is $[i^0, \ldots, i^{k-1}]^T$, for $i = 1, \ldots, k$. Now consider the degree-$t$ row polynomials $f_1(x), \ldots, f_{t+1}(x)$ and let $E$ be the $(t+1) \times (t+1)$ matrix, where $E_{ij}$ is the coefficient of $x^j$ in $f_i(x)$, for $i = 1, \ldots, t+1$ and $j = 0, \ldots, t$. Thus for $i = 1, \ldots, t+1$ and $j = 1, \ldots, t+1$, the $(i,j)^{th}$ entry in $E \cdot V^{(t+1)}$ is $f_i(j)$.

Let $H = ((V^{(t+1)})^T)^{-1} \cdot E$ be a $(t+1) \times (t+1)$ matrix. Let for $i = 0, \ldots, t$, the $(i+1)^{th}$ column of $H$ be $[r_{i0}, r_{i1}, \ldots, r_{it}]^T$. Now we define a degree-$(t,t)$ bivariate polynomial $\overline{F}(x, y) = \sum_{i=0}^{i=t} \sum_{j=0}^{j=t} r_{ij} x^i y^j$. Then from properties of bivariate polynomial, for $i = 1, \ldots, t+1$ and $j = 1, \ldots, t+1$, we have

$$\overline{F}(j, i) = (V^{(t+1)})^T \cdot H \cdot V^{(t+1)} = E \cdot V^{(t+1)} = f_i(j) = g_j(i)$$

This implies that for $i = 1, \ldots, t+1$, the polynomials $\overline{F}(x, i)$ and $f_i(x)$ have same value at $t+1$ values of $x$. But since degree of $\overline{F}(x, i)$ and $f_i(x)$ is $t$, this implies that $\overline{F}(x, i) = f_i(x)$. Similarly, for $j = 1, \ldots, t+1$, we have $\overline{F}(j, y) = g_j(y)$.

Next, we will show that for any $t+1 < i \leq l$, the polynomial $f_i(x)$ also lies on $\overline{F}(x, y)$. In other words, $\overline{F}(x, i) = f_i(x)$, for $t+1 < i \leq l$. This is easy to show because according to theorem statement, $f_i(j) = g_j(i)$, for $j = 1, \ldots, t+1$ and $g_1(i), \ldots, g_{t+1}(i)$ lie on $\overline{F}(x, i)$ and uniquely defines $\overline{F}(x, i)$. Since both $f_i(x)$ and $\overline{F}(x, i)$ are of degree $t$, this implies that $\overline{F}(x, i) = f_i(x)$, for $t+1 < i \leq l$. Similarly, we can show that $\overline{F}(j, y) = g_j(y)$, for $t+1 < j \leq m$. The second part of the lemma is trivially true. □

**Lemma 12.** *In Code Verification, if $D$ is honest then all honest parties will eventually proceed to third phase, except with probability $\epsilon$. Moreover, if $D$ is corrupted and some honest party proceeds to the third phase, then all other honest party will also eventually proceed to the third phase.*

PROOF: If $D$ is honest then every honest party $P_i$ will eventually A-cast `Matched-Row` signal, except with probability $\epsilon'$. The reason is that every honest $P_i$ will privately reconstruct back $f_i(x)$ in AWSS-Rec-Private$(D, \mathcal{P}, f_i(x), P_i, \epsilon')$, except with probability $\epsilon'$. As there are at least $2t+1$ honest parties, except with probability at most $n\epsilon' = \epsilon$, all honest parties will eventually A-cast `Matched-Row` signal. Therefore, eventually there will be a set of $2t+1$ parties, say $\mathcal{R}$, who will A-cast `Matched-Row` signal. $D$ will A-cast $\mathcal{R}$ and eventually every honest party will receive $\mathcal{R}$ from A-cast of $D$ and will eventually receive $2t+1$ `Matched-Row` signal from the parties of $\mathcal{R}$ and will proceed to the third phase, except with probability $\epsilon$.

If $D$ is corrupted and some honest party $P_i$ proceeds to the third phase, then it implies that $P_i$ has received $\mathcal{R}$ from A-cast of $D$ and then `Matched-Row` signal from every party in $\mathcal{R}$. So eventually all other honest parties will also receive $\mathcal{R}$ and corresponding `Matched-Row` signals and will proceed to the third phase. □

From Lemma 11, if the honest parties agree on $\mathcal{R}$, then they are sure that $D$

is committed to a unique bivariate polynomial and thus a unique secret. Now the question is: *If the honest parties stop protocol AVSS-Share after agreeing on $\mathcal{R}$, then is there any possible way of robustly reconstructing $D$'s secret in reconstruction phase*? Here we stop a moment and try to find the possibilities for the above question. Our effort in this direction would also motivate the need of the third phase of AVSS-Share which is actually required to enable robust reconstruction of $D$'s committed secret in the reconstruction phase i.e in AVSS-Rec-Private.

One possible way to reconstruct $D$'s committed secret $s$ is to execute AWSS-Rec-Private$(D, \mathcal{P}, f_j(x), *, \epsilon')$ corresponding to every $P_j \in \mathcal{R}$, which may disclose $f_j(x)$ polynomials and using those polynomial the bivariate polynomial and thus the secret $s$ may be reconstructed. But this does not work, because when $D$ is *corrupted*, *all* instances of AWSS-Rec-Private may output $NULL$. So it seems that most likely there is no way to robustly reconstruct $D$'s committed value $s$ in protocol AVSS-Rec-Private, if AVSS-Share stops after second phase. Hence, we require the third phase which is described in the sequel. Prior to the description of the third phase in the next section, we present the following remark.

**Remark 4.** *In the code* **Commitment**, *$D$ executed $n$ instances of AWSS-Share for individually committing each $f_i(x)$. Later this allowed $f_i(x)$ to be privately reconstructed only by $P_i$ during the code for* **Verification**. *If $D$ executes a single instance of AWSS-MS-Share for concurrently committing to $f_1(x), \ldots, f_n(x)$, instead of $n$ instances of AWSS-Share, then later in the code* **Verification**, *we could not enable $P_1$ to privately reconstruct $f_1(x)$, $P_2$ to privately reconstruct $f_2(x)$ and so on. This is because AWSS-MS-Rec-Private is designed in such a way that it will allow $P_\alpha$ to privately reconstruct back all the $n$ polynomials. This would clearly breach the secrecy property of AVSS as every party will now come to know all the $n$ row polynomials.*

### *5.3. Re-commitment by Individual Parties*

The outline for this phase is as follows: If $P_i$ A-casts `Matched-Row` in code Verification, then $P_i$ acts as a dealer to re-commit his row polynomial $f_i(x)$ by initiating an instance of AWSS-Share. *It is also enforced that if $P_i$ attempts to re-commit $f_i'(x) \neq f_i(x)$, then his re-commitment will not be terminated.* Moreover, when $D$ is honest then an honest $P_i$ will always be able to successfully re-commit $f_i(x)$. Now AVSS-Share terminates only when all the honest parties in $\mathcal{P}$ agree upon a set of at least $2t + 1$ parties, say $VCORE$, who have successfully re-committed their polynomials. Now clearly, if AVSS-Share terminates, then the robust reconstruction of $D's$ committed secret $s$ is guaranteed with very high probability later in reconstruction phase. This is because, the AWSS-Rec-Private instance of an *honest* $P_i \in VCORE$ will always reconstruct back $f_i(x)$. On the other hand, AWSS-Rec-Private instance of a *corrupted* $P_i \in VCORE$ will output either $f_i(x)$ or $NULL$ with probability at least $(1 - \epsilon')$. This guarantees the reconstruction of at least $t+1$ $f_i(x)$ polynomials which are enough to reconstruct $D$'s committed bivariate polynomial and hence $s$. The protocol for this phase is given in Fig. 9.

Figure 9: Code for "Re-commitment by Individual Parties" Phase

## Code **Re-commitment**$(D, \mathcal{P}, s, \epsilon)$

i. Code for $P_i$ — Every party executes this code.

1. If you have A-casted `Matched-Row` in Verification then as a dealer, initiate AWSS-Share$(P_i, \mathcal{P}, f_i(x), \epsilon')$ to re-commit $f_i(x)$ with $\epsilon' = \frac{\epsilon}{n}$. We denote this instance by AWSS-Share$_i$.
2. If $P_j$ has A-casted `Matched-Row` in Verification, then participate in AWSS-Share$_j$ by executing steps in [Verification: Code for $P_i$] (of AWSS-Share) in the following way:
   After the completion of step 1 of [Verification: Code for $P_i$], check whether $g_i(j) = f_j(i)$ holds, where $f_j(i)$ is obtained from the execution of AWSS-Share$_j$ and $g_i(y)$ was obtained from $D$ during **commitment by $D$ phase**. If yes then participate in the remaining steps of [Verification: Code for $P_i$] corresponding to AWSS-Share$_j$.
3. $WCORE^{P_i}$ Construction for AWSS-Share$_i$: If $P_i$ initiated AWSS-Share$_i$ to re-commit $f_i(x)$, then $P_i$ as a dealer, constructs $WCORE$ and corresponding $OKP_j$s for AWSS-Share$_i$ in a slightly different way than what is described in AWSS-Share (*these steps also ensure that a corrupted $P_i$ will not be able to re-commit $\overline{f_i}(x) \neq f_i(x)$*).
   (a) Construct a set $ProbCORE^{P_i}$ ( $= \emptyset$ initially). Include $P_j$ in $ProbCORE^{P_i}$ and A-cast $(P_j, ProbCORE^{P_i})$ if at least $2t+1$ A-casts of the form `OK`$(., P_j)$ are heard in the instance AWSS-Share$_i$.
   (b) Construct $WCORE^{P_i}$. Add $P_j$ in $WCORE^{P_i}$ if both the following holds:
      (A) $P_j \in ProbCORE^{P_i}$ and
      (B) If $(P_j, ProbCORE^{P_k})$ is received from the A-cast of at least $2t+1$ $P_k$'s who have A-casted `Matched-Row`.
   (c) A-cast $WCORE^{P_i}$ and $OKP_j$ for every $P_j \in WCORE^{P_i}$ when $|WCORE^{P_i}| = 2t+1$.

ii. VCORE Construction: Code for $D$ — Only $D$ executs this code

1. If $WCORE^{P_i}$ and $OKP_j$ for every $P_j \in WCORE^{P_i}$ are received from the A-cast of $P_i$, then add $P_i$ to $VCORE$ after performing the following:
   (a) Wait to receive $(P_j, ProbCORE^{P_i})$ for every $P_j \in WCORE^{P_i}$ from the A-cast of $P_i$.
   (b) Wait to receive $(P_j, ProbCORE^{P_k})$ for every $P_j \in WCORE^{P_i}$ from A-cast of at least $2t+1$ $P_k$'s who have A-casted `Matched-Row`.
   (c) Wait to receive $OK(P_j, P_k)$ for every $P_k \in OKP_j$ in execution AWSS-Share$_i$.
2. A-cast $VCORE$ when $|VCORE| = 2t+1$.

iii. VCORE Verification & Agreement on VCORE: Code for $P_i$ — Every party executes this code

1. Wait to receive $VCORE$ from the A-cast of $D$.
2. For every $P_i \in VCORE$, wait to receive $WCORE^{P_i}$ and $OKP_j$ for every $P_j \in WCORE^{P_i}$ from the A-cast of $P_i$.
3. Once received, check the validity of received $WCORE^{P_i}$'s and $OKP_j$'s for every $P_j \in WCORE^{P_i}$ by following the same steps as in ii-1(a), ii-1(b) and ii-1(c).
4. After checking the validity, accept (i) $VCORE$; (ii) $WCORE^{P_i}$ and corresponding $OKP_j$'s for every $P_i \in VCORE$ which are received in previous two steps and terminate AVSS-Share.

**Lemma 13.** *If $D$ is honest then $D$ will eventually generate $VCORE$ of size $2t+1$, except with probability $\epsilon$. Moreover each honest party will agree on this $VCORE$. If $D$ is corrupted and some honest party has accepted $VCORE$ received from $D$, then every other honest party will also eventually do the same.*

PROOF: From the proof of Lemma 12, if $D$ is honest, then all honest parties (at least $2t+1$) will eventually A-cast `Matched-Row` in code Verification, except with probability $\epsilon$. So except with probability $\epsilon$, all these honest $P_i$'s will eventually complete AWSS-Share$_i$ as a dealer and thus will re-commit $f_i(x)$ successfully. Therefore, $D$ will eventually find a set of $2t+1$ $P_i$'s for which the conditions stated in step 1 of [VCORE CONSTRUCTION] will be eventually satisfied. Hence $D$ will add all these $2t+1$ $P_i$'s in $VCORE$ and A-cast the same. Now it is easy to see that every honest party will agree on this $VCORE$ after performing the steps in [VCORE VERIFICATION AND AGREEMENT ON VCORE].

If $D$ is corrupted and some honest party $P_i$ has accepted $VCORE$ received from $D$, then it implies that $P_i$ has checked the validity of received $VCORE$ by performing the steps in [VCORE VERIFICATION AND AGREEMENT ON VCORE]. Now it is easy to see that all other honest parties will also do the same and will accept $VCORE$ eventually. □

**Lemma 14.** *If $VCORE$ is generated, then there exists a unique degree-$(t,t)$ bivariate polynomial $\overline{F}(x,y)$ such that every $P_i \in VCORE$ is re-committed to $f_i(x) = \overline{F}(x,i)$. Moreover, if $D$ is honest then $\overline{F}(x,y) = F(x,y)$.*

PROOF: By Lemma 11, there is a unique degree-$(t,t)$ bivariate polynomial $\overline{F}(x,y)$ such that the row polynomial of every *honest* $P_i$ who has A-casted `Matched-Row`, satisfies $f_i(x) = \overline{F}(x,i)$. Since an honest party $P_i$ who has re-committed his row polynomial $f_i(x)$ in Re-commitment, has also A-casted `Matched-Row` in Verification, $f_i(x) = \overline{F}(x,i)$ satisfies for every *honest* $P_i$ in $VCORE$. Now we show that even a *corrupted* $P_i \in VCORE$ has re-committed $f_i(x)$ satisfying $f_i(x) = \overline{F}(x,i)$.

We prove this by showing that every *honest* $P_j \in WCORE^{P_i}$ has received $f_i(j)$ from $P_i$ during AWSS-Share$_i$ (and hence honest $P_j$ is *IC-committed* to $f_i(j)$). An honest $P_j$ belongs to $WCORE^{P_i}$ implies that $P_j$ belongs to $ProbCORE$ of at least $2t+1$ parties (who have A-casted `Matched-Row`) out of which at least $t+1$ are honest. Let $\mathcal{H}$ be the set of these $(t+1)$ honest parties. So $P_j$'s column polynomial $g_j(y)$ satisfies $g_j(k) = f_k(j)$ for every $P_k \in \mathcal{H}$ (due to step i-(2) in Re-commitment). This implies that $g_j(y) = \overline{F}(j,y)$. Now honest $P_j \in WCORE^{P_i}$ implies that $P_j$ belongs to $ProbCORE$ of $P_i$ as well which means $P_j$ has ensured $g_j(i) = f_i(j)$ (due to step i-(2)) in Re-commitment.

Now the second part of the lemma is trivially true. □

*5.4. Protocol AVSS*

Now the protocols for our AVSS scheme is presented in Fig. 10.

In the following, we prove the properties of our AVSS scheme.

**Lemma 15 (AVSS-Termination).** *Protocol AVSS satisfies termination property of Definition 2.*

Figure 10: AVSS for Sharing Secret $s$ with $n = 3t + 1$

## Protocol AVSS($D, \mathcal{P}, s, \epsilon$)

**AVSS-Share**($D, \mathcal{P}, s, \epsilon$)

1. Replicate Code Commitment($D, \mathcal{P}, s, \epsilon$), Code Verification($D, \mathcal{P}, s, \epsilon$) and Code Re-commitment($D, \mathcal{P}, s, \epsilon$).

**AVSS-Rec-Private**($D, \mathcal{P}, s, P_\alpha, \epsilon$)**:** $P_\alpha$-private-reconstruction of $s$:

$P_\alpha$-WEAK-PRIVATE-RECONSTRUCTION OF $f_j(x)$ FOR EVERY $P_j \in VCORE$: (CODE FOR $P_i$)

1. Participate in AWSS-Rec-Private($P_j, \mathcal{P}, f_j(x), P_\alpha, \epsilon'$) for every $P_j \in VCORE$. We denote AWSS-Rec-Private($P_j, \mathcal{P}, f_j(x), P_\alpha, \epsilon'$) by AWSS-Rec-Private$_j$

LOCAL COMPUTATION**:** CODE FOR $P_\alpha$

1. For every $P_j \in VCORE$, obtain either $\overline{f_j}(x)$ or $NULL$ from $P_\alpha$-weak-private-reconstruction. Add $P_j \in VCORE$ to $REC$ if $\overline{f_j}(x)$ is obtained.
2. Wait until $|REC| = t + 1$. Construct bivariate polynomial $\overline{F}(x, y)$ such that $\overline{F}(x, j) = \overline{f_j}(x)$ for every $P_j \in REC$. Compute $\overline{s} = \overline{F}(0, 0)$ and terminate AVSS-Rec-Private.

PROOF: **Termination 1** and **Termination 2** property follows from Lemma 10, Lemma 12 and Lemma 13. **Termination 3** property is proved as follows: By **Termination 3** and **Correctness 1** of our AWSS scheme (see Lemma 6 and Lemma 8), AWSS-Share$_i$ initiated by an *honest* $P_i$ in $VCORE$ during Re-commitment, will reconstruct $f_i(x)$ in its reconstruction phase, except with probability at most $\epsilon'$ (In AVSS, the AWSS schemes were executed with error probability $\epsilon' \approx \frac{\epsilon}{n}$). But AWSS-Share$_i$ initiated by a *corrupted* $P_i$ in $VCORE$, may lead to the reconstruction of $NULL$ in its reconstruction phase. Since $|VCORE| = 2t+1$, for at least $t+1$ honest parties in $VCORE$, reconstruction of $f_i(x)$'s will be successful, except with probability $(t+1)\epsilon' \approx \epsilon$. This is enough to reconstruct the secret $s$. Hence if all honest parties terminate AVSS-Share and every (honest) party starts AVSS-Rec-Private, then an honest $P_\alpha$ will eventually terminate AVSS-Rec-Private with probability at least $(1 - \epsilon)$. □

**Lemma 16 (AVSS-Secrecy).** *Protocol AVSS satisfies secrecy property of Definition 2.*

PROOF: We have to consider the case when $D$ is honest. Without loss of generality, assume that $P_1, \ldots, P_t$ are the parties under the control of $\mathcal{A}_t$. So throughout AVSS-Share, $\mathcal{A}_t$ will know $f_1(x), \ldots, f_t(x), g_1(y), \ldots, g_t(y)$ and $t$ points on $f_{t+1}(x), \ldots, f_n(x)$. Moreover, honest parties only exchange common values on their row and column polynomials and by the secrecy property of AWSS-Share, these values will be unknown to $\mathcal{A}_t$. Hence by the property of bivariate polynomial of degree-$(t, t)$ [25], $\mathcal{A}_t$ will lack one more point to uniquely interpolate $F(x, y)$. Hence $s = F(0, 0)$ will be information theoretically secure. □

**Lemma 17 (AVSS-Correctness).** *Protocol AVSS satisfies correctness property of Definition 2.*

PROOF: By Lemma 14, there is a unique degree-$(t,t)$ bivariate polynomial $\overline{F}(x,y)$ such that every $P_i \in VCORE$ has re-committed $f_i(x) = \overline{F}(x,i)$. Moreover, if $D$ is honest then $\overline{F}(x,y) = F(x,y)$. Now by Lemma 8, in AWSS-Rec-Private$_i$, except with probability $\epsilon'$ the following will happen:

1. For every honest $P_i \in VCORE$, $f_i(x)$ will be reconstructed;
2. For every corrupted $P_i \in VCORE$, $f_i(x)$ or $NULL$ will be reconstructed.

As $|VCORE| = 2t+1$, for at least $t+1$ parties $P_i$'s in $VCORE$, $f_i(x)$ will be reconstructed with probability at least $(1-(t+1)\epsilon') \approx \epsilon$. Using those polynomials $\overline{F}(x,y)$ and $\overline{s} = \overline{F}(0,0)$ will be reconstructed with probability $1-\epsilon$. Moreover, $s = \overline{s} = F(0,0)$ if $D$ is honest. □

**Lemma 18 (AVSS-Communication-Complexity).** *AVSS-Share incurs a private communication of $\mathcal{O}((n^4 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits and A-cast of $\mathcal{O}(n^3 \log n)$ bits. Protocol AVSS-Rec-Private incurs private communication of $\mathcal{O}((n^4 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits.*

PROOF: The communication complexity of AVSS-Share follows from the fact that in the protocol, $\mathcal{O}(n)$ instances of AWSS-Share have been executed, each with an error parameter of $\frac{\epsilon}{n}$. Similarly, the communication complexity of AVSS-Rec-Private follows from the fact that in the protocol, $\mathcal{O}(n)$ instances of AWSS-Rec-Private have been executed, each with an error parameter of $\frac{\epsilon}{n}$. □

**Theorem 5.** *Protocol AVSS consisting of (AVSS-Share, AVSS-Rec-Private) constitutes a valid statistical AVSS scheme with private reconstruction.*

PROOF: The proof follows from Lemma 15, Lemma 16 and Lemma 17. □

**Remark 5 ($D$'s commitment in AVSS-Share).** *We say that $D$ has committed secret $s \in \mathbb{F}$ in AVSS-Share if there is a degree-$t$ univariate polynomial, $f(x)$, such that $f(0) = s$ and every honest $P_i$ in $VCORE$ receives $f(i)$ from $D$ and commits to $f(i)$ using AWSS-Share. In protocol AVSS-Share, $f(x) = f_0(x) = \overline{F}(x,0)$, where $\overline{F}(x,y)$ is $D$'s committed bivariate polynomial. When $D$ is honest, $\overline{F}(x,y) = F(x,y)$.*

**Notation 5 (Notation for Using AVSS-Share and AVSS-Rec-Private).** *In our ACSS scheme (that shares a single secret), we will invoke AVSS-Share as AVSS-Share$(D, \mathcal{P}, f(x), \epsilon)$ to mean that $D$ commits $f(x)$ in AVSS-Share. Essentially here $D$ is asked to choose bivariate polynomial $F(x,y)$ of degree-$(t,t)$ such that $F(x,0) = f(x)$ holds. Similarly, AVSS-Rec-Private will be invoked as AVSS-Rec-Private$(D, \mathcal{P}, f(x), P_\alpha, \epsilon)$ to enable $P_\alpha$-private-reconstruction of $f(x)$.*

Figure 11: Code for Commitment by $D$ Phase for $\ell \geq 1$ secrets

**Code Commitment-MS($D, \mathcal{P}, S, \epsilon$)**

i. DISTRIBUTION BY $D$: CODE FOR $D$ — Only $D$ executes this code

1. Select $\ell$ random degree-$(t,t)$ bivariate polynomials $F^1(x,y), \ldots, F^\ell(x,y)$ such that $F^l(0,0) = s^l$ for $l = 1, \ldots, \ell$.
2. Send $f_i^l(x) = F^l(x,i)$ and $g_i^l(y) = F^l(i,y)$ for $l = 1, \ldots, \ell$ to $P_i$, for $i = 1, \ldots, n$.
3. For $i = 1, \ldots, n$, initiate AWSS-MS-Share$(D, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x)), \epsilon')$ for sharing $(f_i^1(x), \ldots, f_i^\ell(x))$, where $\epsilon' = \frac{\epsilon}{n}$.

ii. CODE FOR $P_i$ – Every party in $\mathcal{P}$, including $D$, executes this code

1. Wait to receive degree-$t$ polynomials $f_i^l(x)$ and $g_i^l(y)$ for $l = 1, \ldots, \ell$ from $D$.
2. Participate in AWSS-MS-Share$(D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x)), \epsilon')$ by executing steps in [VERIFICATION: CODE FOR $P_i$] (of AWSS-MS-Share) for all $j = 1, \ldots, n$.
3. After the completion of step 1 of [VERIFICATION: CODE FOR $P_i$] for all the $n$ invocations of AWSS-MS-Share, check whether $g_i^l(j) = f_j^l(i)$ holds for all $j = 1, \ldots, n$ and $l = 1, \ldots, \ell$, where $f_j^l(i)$ is obtained from the execution of AWSS-MS-Share$(D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x)), \epsilon')$. If yes then A-cast `Matched-Column`.

Rest of the steps are same as in Commitment.

## 6. Statistical AVSS for Sharing Multiple Secrets

We now present an AVSS scheme AVSS-MS, consisting of protocols (AVSS-MS-Share, AVSS-MS-Rec-Private). Protocol AVSS-MS-Share allows $D$ to share a secret $S = (s^1, \ldots, s^\ell)$, consisting of $\ell > 1$ elements from $\mathbb{F}$. While using $\ell$ invocations of AVSS-Share, one for each $s^l \in S$, $D$ can share $S$ with a private communication of $\mathcal{O}((\ell n^4 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits and A-cast of $\mathcal{O}(\ell n^3 \log n)$ bits, protocol AVSS-MS-Share achieves the same task with a private communication of $\mathcal{O}((\ell n^3 + n^4 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits and A-cast of $\mathcal{O}(n^3 \log n)$ (independent of $\ell$) bits. This shows that executing a *single instance* of AVSS-MS dealing with *multiple secrets* concurrently is advantageous over executing *multiple instances* of AVSS dealing with *single secret*.

The structure of AVSS-MS-Share is divided into same three phases as in AVSS-SS-Share. The corresponding protocols are Commitment-MS, Verification-MS and Re-commitment-MS. They are simple extension of the corresponding protocols in AVSS-Share and are presented in Fig. 11, Fig. 12 and Fig. 13.

Now protocol AVSS-MS-Share$(D, \mathcal{P}, S, \epsilon)$ consists of the code presented in Commitment-MS$(D, \mathcal{P}, S, \epsilon)$, Verification-MS$(D, \mathcal{P}, S, \epsilon)$ and Re-commitment-MS $(D, \mathcal{P}, S, \epsilon)$ in this order. Protocol AVSS-MS-Rec-Private$(D, \mathcal{P}, S, P_\alpha, \epsilon)$ is very straight forward extension of AVSS-Rec-Private. The protocol AVSS-MS is presented in Fig. 14. The proofs for the properties of the protocols dealing with multiple secrets will be similar to the proofs of the protocols dealing with single secret.

Figure 12: Code for Verification of $D$'s Commitment Phase for $\ell \geq 1$ secrets

## Code **Verification-MS**$(D, \mathcal{P}, S, \epsilon)$

$P_j$-WEAK-PRIVATE-RECONSTRUCTION OF $(f_j^1(x), \ldots, f_j^\ell(x))$ FOR $j = 1, \ldots, n$:

i. CODE FOR $P_i$ — Every party in $\mathcal{P}$ executes this code.

1. After agreeing on $WCORE$ and corresponding $OKP_j$'s, participate in AWSS-MS-Rec-Private$(D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x)), P_j, \epsilon')$, for $j = 1, \ldots, n$, to enable $P_j$-weak-private-reconstruction of $(f_j^1(x), \ldots, f_j^\ell(x))$. Notice that same $WCORE$ is used in each AWSS-SS-Rec-Private$(D, \mathcal{P}, f_j(x), P_j, \epsilon')$, for $j = 1, \ldots, n$
2. At the completion of AWSS-MS-Rec-Private$(D, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x)), P_i, \epsilon')$, obtain either degree-$t$ polynomials $\overline{f_i^1}(x), \ldots, \overline{f_i^\ell}(x)$ or $NULL$.
3. If $f_i^l(x) = \overline{f_i^l}(x)$ for all $l = 1, \ldots, \ell$, then A-cast `Matched-Row`.

Rest of the steps are same as in Verification.

Figure 13: Code for "Re-commitment by Individual Parties" Phase for $\ell \geq 1$ secrets

## Code **Re-commitment-MS**$(D, \mathcal{P}, S, \epsilon)$

i. CODE FOR $P_i$ — Every party executes this code

1. If you have A-casted `Matched-Row` in Verification-MS then initiate AWSS-MS-Share$(P_i, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x), \epsilon')$ to re-commit $(f_i^1(x), \ldots, f_i^\ell(x))$, where $\epsilon' = \frac{\epsilon}{n}$.
2. For each $j$, such that $P_j$ has A-casted `Matched-Row` in Verification-MS, participate in AWSS-MS-Share$(P_j, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x), \epsilon')$ by executing steps in [VERIFICATION: CODE FOR $P_i$] (of AWSS-MS-Share) in the following way:
   After the completion of step 1 of [VERIFICATION: CODE FOR $P_i$], check whether $g_i^l(j) = f_j^l(i)$ for $l = 1, \ldots, \ell$ holds, where $(f_j^1(i), \ldots, f_j^\ell(i))$ are obtained from the execution of AWSS-SS-Share$(P_j, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x), \epsilon')$ and $(g_i^1(y), \ldots, g_i^\ell(y))$ was obtained from $D$ in code Commitment-MS. If yes then participate in the next steps in [VERIFICATION: CODE FOR $P_i$] corresponding to AWSS-MS-Share$(P_j, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x), \epsilon')$.

**Rest of the steps are same as in Re-commitment except that at every place AWSS-Share$(P_i, \mathcal{P}, f_i(x), \epsilon')$ is replaced by AWSS-MS-Share$(P_i, \mathcal{P}, (f_i^l(x), \ldots, f_i^\ell(x)), \epsilon')$.**

Figure 14: AVSS for Sharing $\ell \geq 1$ Secrets with $n = 3t + 1$

Protocol **AVSS-MS**$(D, \mathcal{P}, S, \epsilon)$

**AVSS-MS-Share**$(D, \mathcal{P}, S, \epsilon)$

1. Replicate Code Commitment-MS$(D, \mathcal{P}, S, \epsilon)$, Code Verification-MS$(D, \mathcal{P}, S, \epsilon)$ and Code Re-commitment-MS$(D, \mathcal{P}, S, \epsilon)$.

**AVSS-MS-Rec-Private**$(D, \mathcal{P}, S, P_\alpha, \epsilon)$**:** $P_\alpha$-private-reconstruction of $S$:

$P_\alpha$-WEAK-PRIVATE-RECONSTRUCTION OF $(f_j^1(x), \ldots, f_j^\ell(x))$ FOR EVERY $P_j \in VCORE$: (CODE FOR $P_i$)

1. Participate in AWSS-MS-Rec-Private$(P_j, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x)), P_\alpha, \epsilon')$ for every $P_j \in VCORE$, where $\epsilon' = \frac{\epsilon}{n}$.

LOCAL COMPUTATION**:** CODE FOR $P_\alpha$

1. For every $P_j \in VCORE$, obtain either $(\overline{f_j^1}(x), \ldots, \overline{f_j^\ell}(x))$ or $NULL$ from $P_\alpha$-weak-private-reconstruction. Add party $P_j \in VCORE$ to $REC$ if non-NULL output is obtained.
2. Wait until $|REC| = t + 1$. Construct bivariate polynomial $\overline{F^1}(x, y), \ldots, \overline{F^\ell}(x, y)$ such that $\overline{F^l}(x, j) = \overline{f_j^l}(x)$ for every $P_j \in REC$ and every $l = 1, \ldots, \ell$. Compute $\overline{s^l} = \overline{F^l}(0, 0)$ for every $l = 1, \ldots, \ell$ and terminate AVSS-MS-Rec-Private.

**Theorem 6.** *Protocol AVSS-MS-Share incurs a private communication of* $\mathcal{O}((\ell n^3 + n^4 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ *bits and A-cast* $\mathcal{O}(n^3 \log n)$ *bits. Protocol AVSS-Rec-Private incurs a private communication of* $\mathcal{O}((\ell n^3 + n^4 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ *bits.*

PROOF: The communication complexity of AVSS-MS-Share follows from the fact that $\mathcal{O}(n)$ instances of AWSS-MS-Share each with $\ell$ secrets have been executed in AVSS-MS-Share. Similarly, the communication complexity of AVSS-MS-Rec-Private follows from the fact that in the protocol, $\mathcal{O}(n)$ instances of AWSS-Rec-Private each with $\ell$ secrets have been executed. □

**Theorem 7.** *Protocol AVSS-MS consisting of (AVSS-MS-Share, AVSS-MS-Rec-Private) constitutes a valid statistical AVSS scheme with private reconstruction for* $\ell \geq 1$ *secrets.*

**Remark 6 ($D$'s commitment in AVSS-MS-Share).** *We say that $D$ has committed secret $S \in \mathbb{F}^\ell$ in AVSS-MS-Share if there are $\ell$ degree-$t$ univariate polynomials, $f^1(x), \ldots, f^\ell(x)$, such that $f^l(0) = s^l$ for $l = 1, \ldots, \ell$ and every honest $P_i$ in $VCORE$ receives $(f^1(i), \ldots, f^\ell(i))$ from $D$ and commits to $(f^1(i), \ldots, f^\ell(i))$ using AWSS-MS-Share. In protocol AVSS-MS-Share, $f^l(x) = f_0^l(x) = \overline{F^l}(x, 0)$ for every $l = 1, \ldots, \ell$, where $\overline{F^1}(x, y), \ldots, \overline{F^\ell}(x, y)$ are $D$'s committed bivariate polynomial. When $D$ is honest, $\overline{F^l}(x, y) = F^l(x, y)$ for $l = 1, \ldots, \ell$.*

**Notation 6 (Notation for Using AVSS-MS-Share and AVSS-MS-Rec-Private).** *In our ACSS scheme (that shares $\ell$ secrets), we will invoke AVSS-MS-Share as*

*AVSS-MS-Share* $(D, \mathcal{P}, (f^1(x), \ldots, f^\ell(x)), \epsilon)$ *to mean that* $D$ *commits* $f^1(x), \ldots, f^\ell(x)$ *in* AVSS-MS-Share. *Essentially here* $D$ *is asked to choose bivariate polynomials* $F^1(x,y), \ldots, F^\ell(x,y)$*, each of degree-*$(t,t)$ *such that* $F^l(x,0) = f^l(x)$ *holds for* $l = 1, \ldots, \ell$. *Similarly,* AVSS-MS-Rec-Private *will be invoked as* AVSS-MS-Rec-Private$(D, \mathcal{P}, (f^1(x), \ldots, f^\ell(x)), P_\alpha, \epsilon)$ *to enable* $P_\alpha$*-private-reconstruction of* $(f^1(x), \ldots, f^\ell(x))$.

## 7. Statistical ACSS for Sharing a Single Secret

Though protocol AVSS is an AVSS scheme, it is not an ACSS scheme as it does not achieve *completeness* property. This is because in AVSS-Share, only the honest parties in $VCORE$ receive their respective shares of the committed secret. But it may happen that potentially $t$ honest parties are *not* present in $VCORE$. So we now present an ACSS scheme called ACSS, which consists of sub-protocols (ACSS-Share, ACSS-Rec-Private, ACSS-Rec). Protocol ACSS-Share allows $D$ to generate $t$-sharing of a secret $s \in \mathbb{F}$. Given $t$-sharing of secret $s$, protocol ACSS-Rec-Private allows a specific party in $\mathcal{P}$, say $P_\alpha$, to privately reconstruct $s$. On the other hand, ACSS-Rec allows every party in $\mathcal{P}$ to reconstruct $D$'s committed secret $s$. Protocol ACSS-Share, ACSS-Rec-Private and ACSS-Rec will be used in our statistical AMPC protocol.

**The Intuition:** The high level idea of ACSS-Share is similar to AVSS-Share. But now in ACSS-Share, we use AVSS-Share as a black-box, in place of AWSS-Share. It is this change which helps ACSS to achieve *completeness property*. We now show that AVSS-Share which uses AWSS-Share as a black box may not output $t$-sharing of $D$'s committed secret. Subsequently, we also point out how ACSS-Share overcome this problem by using AVSS-Share as a black-box.

So let us consider protocol AVSS-Share when $D$ is *corrupted* and also assume that $D$ is committed to a unique secret and thus a unique bi-variate polynomial $\overline{F}(x,y)$ of degree-$(t,t)$. But in spite of this, we could only ensure that every honest $P_i$ who A-cast `Matched-Row` signal, holds the corresponding row polynomial $f_i(x) = \overline{F}(x,i)$ and hence his share $f_i(0)$ of the secret $\overline{s} = \overline{F}(0,0)$. However, it is possible that there are potential $t$ honest $P_i$'s who have not A-casted `Matched-Row` signal due to the reconstruction of $NULL$ from $P_i$-weak-private-reconstruction during **Verification of $D$'s Commitment Phase**. Also a corrupted $D$ may not even pass on $\overline{F}(x,i)$ or may pass some wrong polynomial other than $\overline{F}(x,i)$ to these $P_i$'s. So in this case $t$ potential honest parties may not hold shares of secret $\overline{s}$.

On the other hand, AVSS-Share is used as a black-box in ACSS-Share. This overcomes the above problem because now $D$ would commit each $f_i(x)$ using AVSS-Share, instead of AWSS-Share. So once it is ensured that $D$ is committed to a unique bi-variate polynomial $\overline{F}(x,y)$ of degree-$(t,t)$, by the property of AVSS-Rec-Private, each honest $P_i \in \mathcal{P}$ would successfully reconstruct $f_i(x) = \overline{F}(x,i)$ and hence his share $f_i(0)$ of the secret $\overline{s} = \overline{F}(0,0)$. Protocol ACSS-Share is provided in Fig. 15. Protocol ACSS-Rec-Private and ACSS-Rec uses OEC (Online Error Correction method) and are presented in Fig. 16.

Figure 15: Protocol ACSS-Share for Sharing Secret $s$ with $n = 3t + 1$

## Protocol ACSS-Share($D, \mathcal{P}, s, \epsilon$)

i. Distribution by $D$: Code for $D$ – Only $D$ executes this code

1. Select a random degree-$(t, t)$ bivariate polynomial $F(x, y)$ such that $F(0, 0) = s$.
2. For $i = 1, \ldots, n$, send $g_i(y) = F(i, y)$ to party $P_i$. We call $g_i(y)$ as $i^{th}$ column polynomial.
3. For $i = 1, \ldots, n$, initiate AVSS-Share$(D, \mathcal{P}, f_i(x), \epsilon')$ for sharing $f_i(x)$, where $f_i(x) = F(x, i)$ and $\epsilon' = \frac{\epsilon}{n}$. We call $f_i(x)$ as $i^{th}$ row polynomial. We refer AVSS-Share$(D, \mathcal{P}, f_i(x), \epsilon')$ by AVSS-Share$_i$

ii. Code for $P_i$ – Every party in $\mathcal{P}$, including $D$, executes this code

1. Wait to receive degree-$t$ column polynomial $g_i(y)$ from $D$.
2. Participate in AVSS-Share$_j$ for all $j = 1, \ldots, n$.
3. If $f_j(i)$ is received from $D$ during AVSS-Share$_j$ then check whether $g_i(j) = f_j(i)$. When the test passes for all $j = 1, \ldots, n$, then A-cast `Matched-Column`.

iii. CCORE Construction: Code for $D$ – Only $D$ executes this code.

1. For $i = 1, \ldots, n$, construct $VCORE$ for AVSS-Share$(D, \mathcal{P}, f_i(x), \epsilon')$. Denote it by $VCORE^i$.
2. For $i = 1, \ldots, n$, keep updating $VCORE^i$ even after $|VCORE^i| = 2t + 1$. Wait to obtain $CCORE = \cap_{i=1}^{n} VCORE^i$ of size at least $2t + 1$ such that `Matched-Column` is received from A-cast of every $P_j \in CCORE$ .
3. A-cast $CCORE$.

iv. CCORE verification & Agreement: Code for $P_i$ — Every party including $D$ will execute this code.

1. Wait to receive $CCORE$ from the A-cast of $D$.
2. Check whether $CCORE$ is a valid $VCORE$ for AVSS-Share$_j$ for every $j = 1, \ldots, n$ (by following the steps 2-4 as specified under [VCORE Verification & Agreement on VCORE: Code for $P_i$] in code Re-commitment of AVSS-Share). If yes then wait to receive `Matched-Column` from A-cast of every $P_j \in CCORE$ and then accept $CCORE$.

v. $P_j$-private-reconstruction of $f_j(x)$ for $j = 1, \ldots, n$: Code for $P_i$ – Every party in $\mathcal{P}$ executes this code.

1. If CCORE is a valid $VCORE$ for AVSS-Share$_j$ for every $j = 1, \ldots, n$, then participate in AVSS-Rec-Private$(D, \mathcal{P}, f_j(x), P_j, \epsilon')$, for $j = 1, \ldots, n$, to enable $P_j$-private-reconstruction of $f_j(x)$. We refer AVSS-Rec-Private$(D, \mathcal{P}, f_j(x), P_j, \epsilon')$ as AVSS-Rec-Private$_j$. Notice that CCORE is used as VCORE in each AVSS-Rec-Private$_j$, for $j = 1, \ldots, n$.
2. At the completion of AVSS-Rec-Private$_i$, obtain degree-$t$ polynomial $f_i(x)$.
3. Assign $s_i = f_i(0)$. Output $s_i$ as $i^{th}$ share of $s$ and terminate ACSS-Share.

Figure 16: Protocol ACSS-Rec-Private and ACSS-Rec for Reconstructing Secret $s$ privately and publicly (respectively) with $n = 3t + 1$

**ACSS-Rec-Private**$(D, \mathcal{P}, s, P_\alpha, \epsilon)$: $P_\alpha$-private-reconstruction of $s$:

i. CODE FOR $P_i$ – Every party in $\mathcal{P}$ executes this code.

1. Privately send $s_i$, the $i^{th}$ share of $s$ to $P_\alpha$.

ii. CODE FOR $P_\alpha$ – Only $P_\alpha \in \mathcal{P}$ executes this code.

1. Apply OEC on received shares of $s$ to reconstruct $s$ and terminate ACSS-Rec-Private.

**ACSS-Rec**$(D, \mathcal{P}, s, \epsilon)$: Public reconstruction of $s$:

i. CODE FOR $P_i$ – Every party in $\mathcal{P}$ executes this code.

1. Privately send $s_i$, the $i^{th}$ share of $s$ to every party $P_j \in \mathcal{P}$.
2. Apply OEC on received shares of $s$ to reconstruct $s$ and terminate ACSS-Rec.

We now prove the properties of ACSS.

**Lemma 19.** *In protocol ACSS-Share:*

1. *If $D$ is honest then eventually he will generate a $CCORE$ of size $2t + 1$ except with probability $\epsilon$. Moreover, each honest party will eventually agree on $CCORE$.*
2. *If $D$ is corrupted and some honest party has accepted the $CCORE$ received from the A-cast of $D$, then every other honest party will eventually accept $CCORE$.*

PROOF: In ACSS-Share if $D$ is honest then from the proof of Lemma 13, an *honest* party may be added in each $VCORE^i$ except with probability $\epsilon' = \frac{\epsilon}{n}$ (recall that each instance of AVSS-Share has an associated error probability of $\epsilon' = \frac{\epsilon}{n}$). So even though there are no common corrupted parties among $VCORE^i$'s, eventually all the honest parties will be common among $n$ $VCORE^i$'s with probability at least $1 - (2t+1)\epsilon' \approx 1 - \epsilon$. Moreover, each honest $P_i$ will eventually A-cast `Matched-Column` signal, as $f_j(i) = g_i(j)$ will hold for all $j = 1, \ldots, n$ when $D$ is *honest*. It may be possible that some corrupted parties are also added in each $VCORE^i$. Moreover those corrupted parties may even A-cast `Matched-Column` signal. So except with probability $\epsilon$, at some point of time $CCORE = \cap_{i=1}^{n} VCORE^i$ will contain at least $2t+1$ parties who have A-casted `Matched-Column` signal. So honest $D$ will find $CCORE$ and A-cast the same. Now it is easy to see that each honest party will accept $CCORE$ after receiving it from A-cast of $D$ and verifying its' validity after following steps in iv(2) of protocol ACSS-Share.

If $D$ is corrupted and some honest party, say $P_i$ has accepted $CCORE$ received from the A-cast of $D$, then $P_i$ must have checked the condition specified in iv(2) of protocol ACSS-Share. The same will hold for all other honest parties who will eventually accept $CCORE$. □

**Lemma 20.** *In ACSS-Share, if the honest parties agree on* CCORE*, then it implies that $D$ is committed to a unique degree-$(t,t)$ bivariate polynomial $\overline{F}(x,y)$ such that each row polynomial $f_i(x)$ committed by $D$ in AVSS-Share$_i$ satisfies $\overline{F}(x,i) = f_i(x)$ and the column polynomial $g_j(y)$ held by every honest $P_j \in CCORE$ satisfies $\overline{F}(j,y) = g_j(y)$. Moreover if $D$ is honest then $\overline{F}(x,y) = F(x,y)$.*

PROOF: The proof follows from the same argument as given in Lemma 11. □

**Lemma 21.** *In ACSS-Share, if the honest parties agree on* CCORE*, then eventually all honest parties will get their share of $D$'s committed secret $\overline{s}$, except with probability at most $\epsilon$. That is, protocol ACSS-Share will generate $t$-sharing of $\overline{s}$ except with probability $\epsilon$. Moreover if $D$ is honest then $\overline{s} = s$.*

PROOF: From the previous lemma, if the honest parties agree on CCORE then it implies that $D$ is committed to a unique degree-$(t,t)$ bivariate polynomial $\overline{F}(x,y)$ such that each row polynomial $f_i(x)$ committed by $D$ in AVSS-Share$(D, \mathcal{P}, f_i(x), \epsilon')$ satisfies $\overline{F}(x,i) = f_i(x)$. Now from the properties of AVSS-Rec-Private, $P_i$-private-reconstruction of $f_i(x)$ will enable honest $P_i$ to obtain $f_i(x)$ and hence his share $f_i(0)$, except with probability $\epsilon'$. As there are $2t+1$ honest parties in $\mathcal{P}$, all honest parties will obtain their share of the secret $\overline{s} = \overline{F}(0,0)$, except with probability $(2t+1)\epsilon' \approx \epsilon$. Hence the secret $\overline{s} = \overline{F}(0,0)$ will be $t$-shared by the degree-$t$ polynomial $\overline{F}(x,0)$, except with probability at most $\epsilon$. □

**Lemma 22 (ACSS-Termination).** *Protocol ACSS satisfies termination property of Definition 4.*

PROOF: **Termination 1** and **Termination 2** follows from Lemma 19 and Lemma 21. **Termination 3** follows from Lemma 21 and properties of OEC. □

**Lemma 23 (ACSS-Secrecy).** *Protocol ACSS satisfies secrecy property of Definition 4.*

PROOF: Here we have to consider the case when $D$ is honest. Without loss of generality, assume that $P_1, \ldots, P_t$ are the parties under the control of $\mathcal{A}_t$. So during ACSS-Share, $\mathcal{A}_t$ will know $f_1(x), \ldots, f_t(x), g_1(y), \ldots, g_t(y)$ and $t$ points on $f_{t+1}(x), \ldots, f_n(x)$. From the secrecy property of AVSS-Share (Lemma 16), $\mathcal{A}_t$ will have no information about $f_{t+1}(0), \ldots, f_n(0)$ during the execution of corresponding instances of AVSS-Share. So from the properties of bi-variate polynomial of degree-$(t,t)$ [25], the adversary $\mathcal{A}_t$ will lack one more point to uniquely interpolate $F(x,y)$ during ACSS-Share. Hence hence the secret $s = F(0,0)$ will remain information theoretically secure from $\mathcal{A}_t$. □

**Lemma 24 (ACSS-Correctness).** *Protocol ACSS satisfies correctness property of Definition 4.*

PROOF: Follows from Lemma 19, Lemma 20, Lemma 21 and from the properties of OEC. □

**Lemma 25 (ACSS-Completeness).** *Protocol ACSS satisfies completeness property of Definition 4.*

PROOF: Follows from Lemma 21. □

**Lemma 26.** *Protocol ACSS-Share privately communicates $\mathcal{O}(n^5(\log \frac{1}{\epsilon})^2)$ bits and A-casts $\mathcal{O}(n^4 \log n)$ bits. Protocol ACSS-Rec-Private and ACSS-Rec incurs a private communication of $\mathcal{O}(n \log \frac{1}{\epsilon})$ and $\mathcal{O}(n^2 \log \frac{1}{\epsilon})$ bits respectively.*

PROOF: The communication complexity of ACSS-Share follows from the fact that in ACSS-Share, there are $n$ executions of AVSS-Share, each with an error parameter of $\frac{\epsilon}{n}$. In ACSS-Rec-Private, each party sends his share to $P_\alpha$, incurring a total communication cost of $\mathcal{O}(n \log \frac{1}{\epsilon})$ bits. In ACSS-Rec, each party sends his share to every other party, incurring a total communication cost of $\mathcal{O}(n^2 \log \frac{1}{\epsilon})$ bits. □

**Theorem 8.** *Protocol ACSS consisting of (AVSS-Share, ACSS-Rec-Private, AVSS-Rec) constitutes a valid statistical ACSS scheme for sharing a single secret.*

PROOF: Follows from Lemma 22, Lemma 24, Lemma23 and Lemma 25. □

## 8. Statistical ACSS for Sharing Multiple Secrets

We now present an ACSS scheme ACSS-MS, consisting of sub-protocols (ACSS-MS-Share, ACSS-MS-Rec-Private, ACSS-MS-Rec). Protocol ACSS-MS-Share allows $D$ to generate $t$-sharing of secret $S = (s^1, \ldots, s^\ell)$, consisting of $\ell > 1$ elements from $\mathbb{F}$. While $D$ can ACSS-share $S$ using $\ell$ executions of ACSS-Share, one for each $s^l \in S$, with a private communication of $\mathcal{O}((\ell n^5 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ and A-cast of $\mathcal{O}(\ell n^4 \log(n))$ bits, protocol ACSS-MS-Share achieves the same task with a private communication of $\mathcal{O}((\ell n^4 + n^5 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits and A-cast of $\mathcal{O}(n^4 \log(n))$ (independent of $\ell$) bits. This shows that executing a *single instance* of ACSS-MS dealing with *multiple secrets* concurrently is advantageous over executing *multiple instances* of ACSS dealing with *single secret*. Protocol ACSS-MS-Share is provided in Fig. 17. Protocol ACSS-MS-Rec-Private and ACSS-MS-Rec are presented in Fig. 18. The proof of the properties of ACSS-MS can be directly extended from the proof of the properties of ACSS.

**Theorem 9.** *Protocol ACSS-MS-Share privately communicates $\mathcal{O}((\ell n^4 + n^5 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits and A-casts $\mathcal{O}(n^4 \log n)$ bits. Protocol ACSS-MS-Rec-Private and ACSS-MS-Rec incurs a private communication of $\mathcal{O}(\ell n \log \frac{1}{\epsilon})$ bits and $\mathcal{O}(\ell n^2 \log \frac{1}{\epsilon})$ bits respectively.*

PROOF: The communication complexity of ACSS-MS-Share follows from the fact that $\mathcal{O}(n)$ instances of AVSS-MS-Share each with $\ell$ secrets have been executed in ACSS-MS-Share. In ACSS-MS-Rec-Private, each party sends his shares of $\ell$ secrets to $P_\alpha$, incurring a total communication cost of $\mathcal{O}(\ell n \log \frac{1}{\epsilon})$ bits. In ACSS-Rec, each party sends his shares of $\ell$ secrets to every other party, incurring a total communication cost of $\mathcal{O}(\ell n^2 \log \frac{1}{\epsilon})$ bits. □

**Theorem 10.** *Protocol ACSS-MS consisting of sub-protocols (ACSS-MS-Share, ACSS-MS-Rec-Private, ACSS-MS-Rec) is a valid statistical ACSS scheme for sharing $\ell \geq 1$ secrets.*

**Notation 7 (Notation for Using ACSS-MS).** *In the subsequent sections, we will invoke ACSS-MS-Share as ACSS-MS-Share $(D, \mathcal{P}, (f^1(x), \ldots, f^\ell(x)), \epsilon)$ to mean that $D$ commits to $f^1(x), \ldots, f^\ell(x)$ in ACSS-MS-Share. Essentially here $D$ is asked to choose bivariate polynomials $F^1(x,y), \ldots, F^\ell(x,y)$, each of degree-$(t,t)$, such that $F^l(x,0) = f^l(x)$ holds for $l = 1, \ldots, \ell$. As a result of this execution, each honest party $P_i$ will get the shares $f^1(i), \ldots, f^\ell(i)$. Similarly, ACSS-MS-Rec-Private will be invoked as ACSS-MS-Rec-Private$(D, \mathcal{P}, (f^1(x), \ldots, f^\ell(x)), P_\alpha, \epsilon)$ to enable $P_\alpha \in \mathcal{P}$ to privately reconstruct $(f^1(x), \ldots, f^\ell(x))$. Similarly, ACSS-MS-Rec will be invoked as ACSS-MS-Rec$(D, \mathcal{P}, (f^1(x), \ldots, f^\ell(x)), \epsilon)$ to enable each party in $\mathcal{P}$ to reconstruct $(f^1(x), \ldots, f^\ell(x))$.*

## 9. Statistical Asynchronous Multiparty Computation with Optimal Resilience

In this section, we construct an efficient statistical *asynchronous multiparty computation* (AMPC) protocol with optimal resilience; i.e., with $n = 3t+1$ using our proposed ACSS scheme as an important building block. In the following, we first present the definition of MPC and AMPC, literature survey on related AMPC protocols, the primitives used for designing our AMPC protocol and then brief description of the approaches used in designing existing AMPC and our AMPC protocol.

### *9.1. Multiparty Computation*

A Multiparty Computation (MPC) [71, 23, 13, 67] protocol allows the parties in $\mathcal{P}$ to securely compute an agreed function $f$, even in the presence of $\mathcal{A}_t$. More specifically, assume that the agreed function $f$ can be expressed as $f : \mathbb{F}^n \rightarrow \mathbb{F}^n$ and party $P_i$ has input $x_i \in \mathbb{F}$. At the end of the computation of $f$, each honest $P_i$ gets $y_i \in \mathbb{F}$, where $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$, irrespective of the behavior of $\mathcal{A}_t$ (**correctness**). Moreover, $\mathcal{A}_t$ should not get any information about the input and output of the honest parties, other than what can be inferred from the input and output of the corrupted parties (**secrecy**). In any general MPC protocol, the function $f$ is specified by an arithmetic circuit over $\mathbb{F}$, consisting of input, linear (e.g. addition), multiplication, random and output gates. We denote the number of gates of these types in the circuit by $c_I, c_A, c_M, c_R$ and $c_O$ respectively. *Among all the different type of gates, evaluation of a multiplication gate requires the maximum communication complexity. So the communication complexity of any general MPC protocol is usually given in terms of the communication complexity per multiplication gate* [11, 10, 9, 31, 52].

The MPC problem has been studied extensively over synchronous networks [2, 3, 4, 5, 13, 9, 11, 7, 20, 23, 27, 28, 31, 42, 41, 45, 46, 48, 8, 47, 49, 50, 57, 67, 60, 72]. However, MPC in asynchronous network has got comparatively less

Figure 17: Protocol ACSS-MS-Share for Sharing Secret $S$ Containing $\ell$ Eelements with $n = 3t + 1$

## Protocol ACSS-MS-Share$(D, \mathcal{P}, S, \epsilon)$

i. Distribution by $D$: Code for $D$ – Only $D$ executes this code

1. Select $\ell$ random degree-$(t, t)$ bivariate polynomials $F^1(x, y), \ldots, F^\ell(x, y)$ such that $F^l(0, 0) = s^l$ for $l = 1, \ldots, \ell$.
2. For $i = 1, \ldots, n$, send $g_i^l(y) = F^l(i, y)$ for $l = 1, \ldots, \ell$ to $P_i$. We call polynomials $g_i^1(y), \ldots, g_i^\ell(y)$ as $i^{th}$ column polynomials.
3. For $i = 1, \ldots, n$, initiate AVSS-MS-Share$(D, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x)), \epsilon')$ for sharing $(f_i^1(x), \ldots, f_i^\ell(x))$, where $f_i^l(x) = F^l(x, i)$ and $\epsilon' = \frac{\epsilon}{n}$. We call polynomials $f_i^1(x), \ldots, f_i^\ell(x)$ as $i^{th}$ row polynomials. We refer AVSS-MS-Share$(D, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x)), \epsilon')$ as AVSS-MS-Share$_i$

ii. Code for $P_i$ – Every party in $\mathcal{P}$, including $D$, executes this code

1. Wait to receive degree-$t$ polynomials $g_i^l(y)$ for $l = 1, \ldots, \ell$ from $D$.
2. Participate in AVSS-MS-Share$_j$ for all $j = 1, \ldots, n$.
3. If $(f_j^1(i), \ldots, f_j^\ell(i))$ is received from $D$ during AVSS-MS-Share$_j$ then check whether $g_i^l(j) = f_j^l(i)$ holds for all $l = 1, \ldots, \ell$. When the test passes for all $j = 1, \ldots, n$, then A-cast `Matched-Column`.

iii. CCORE Construction: Code for $D$ – Only $D$ executes this code.

1. For $i = 1, \ldots, n$, construct $VCORE$ for AVSS-MS-Share$_i$. Denote it by $VCORE^i$.
2. For $i = 1, \ldots, n$, keep updating $VCORE^i$, even after $|VCORE^i| = 2t + 1$. Wait to obtain $CCORE = \cap_{i=1}^n VCORE^i$ of size at least $2t + 1$ such that `Matched-Column` is received from A-cast of every $P_j \in CCORE$.
3. A-cast $CCORE$.

iv. CCORE verification & Agreement: Code for $P_i$ – Every party including $D$ will execute this code.

1. Wait to receive $CCORE$ from the A-cast of $D$.
2. Check whether $CCORE$ is a valid $VCORE$ for AVSS-MS-Share$_j$ for every $j = 1, \ldots, n$ (by following the steps 2-4 as specified under [VCORE Verification & Agreement on VCORE: Code for $P_i$] in protocol Re-commitment-MS).

v. $P_j$-private-reconstruction of $(f_j^1(x), \ldots, f^\ell(x))$ for $j = 1, \ldots, n$: Code for $P_i$

1. If CCORE is a valid $VCORE$ for AVSS-MS-Share$_j$ for every $j = 1, \ldots, n$, then participate in AVSS-MS-Rec-Private$(D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x)), P_j, \epsilon')$, for $j = 1, \ldots, n$, to enable $P_j$-private-reconstruction of $(f_j^1(x), \ldots, f_j^\ell(x))$. We refer AVSS-MS-Rec-Private$(D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x)), P_j, \epsilon')$ by AVSS-MS-Rec-Private$_j$. Notice that CCORE is used as VCORE in each AVSS-MS-Rec-Private$_j$, for $j = 1, \ldots, n$.
2. At the completion of AVSS-MS-Rec-Private$_i$, obtain degree-$t$ polynomials $(f_i^1(x), \ldots, f_i^\ell(x))$.
3. Assign $s_i^l = f_i^l(0)$. Output $(s_i^1, \ldots, s_i^\ell)$ as $i^{th}$ share of $(s^1, \ldots, s^\ell)$ and terminate ACSS-MS-Share.

Figure 18: Protocol ACSS-MS-Rec-Private and ACSS-MS-Rec for Reconstructing Secret $S$ privately and publicly (respectively) with $n = 3t + 1$

**ACSS-MS-Rec-Private($D, \mathcal{P}, S, P_\alpha, \epsilon$)**: $P_\alpha$-private-reconstruction of $S$:

i. CODE FOR $P_i$ – Every party in $\mathcal{P}$ executes this code.

1. Privately send $s_i^1, \ldots, s_i^\ell$, the $i^{th}$ shares of $s^1, \ldots, s^\ell$ (respectively) to party $P_\alpha \in \mathcal{P}$.

ii. CODE FOR $P_\alpha$ – Only $P_\alpha \in \mathcal{P}$ executes this code.

1. For $l = 1, \ldots, \ell$, apply OEC on received shares of $s^l$ to reconstruct $s^l$ and terminate ACSS-MS-Rec-Private.

**ACSS-MS-Rec($D, \mathcal{P}, S, \epsilon$)**: Public reconstruction of $S$:

i. CODE FOR $P_i$ – Every party in $\mathcal{P}$ executes this code.

1. Privately send $s_i^1, \ldots, s_i^\ell$, the $i^{th}$ shares of $s^1, \ldots, s^\ell$ (respectively) to every party $P_j \in \mathcal{P}$.
2. For $l = 1, \ldots, \ell$, apply OEC on received shares of $s^l$ to reconstruct $s^l$ and terminate ACSS-MS-Rec.

attention, due to its inherent hardness. As asynchronous networks model real life networks like Internet more appropriately than synchronous networks, fundamental problems like MPC is worthy of deep investigation over asynchronous networks.

### *9.2. Asynchronous Multiparty Computation (AMPC)*

Any asynchronous MPC (AMPC) protocol should satisfy **termination** condition, in addition to **correctness** and **secrecy** condition (specified earlier). According to the termination condition, every honest party should eventually terminate the protocol. There are mainly two types of AMPC protocols:

1. A *perfectly secure* AMPC protocol satisfies all the properties of AMPC *without any error*;
2. A *statistically secure* (statistical in short) AMPC protocol involves a *negligible error* probability of $\epsilon$ in **correctness** and/or **termination**. However, note that there is *no* compromise in **secrecy** property.

From [12], *perfectly secure* AMPC is possible iff $n \geq 4t + 1$. On the other hand, *statistically secure* AMPC is possible iff $n \geq 3t + 1$ [15]. In this paper, we concentrate on *statistically secure* AMPC with *optimal resilience*; i.e., with $n = 3t + 1$. The communication complexity *per multiplication gate* of existing statistically secure AMPC protocols are as follows:

| Reference | Resilience | Communication Complexity in bits |
|---|---|---|
| [15] | $t < n/3$ (optimal) | private– $\Omega(c_M n^{11}(\log \frac{1}{\epsilon})^4)$; A-cast– $\Omega(c_M n^{11}(\log \frac{1}{\epsilon})^2 \log(n))$ |
| [65] | $t < n/4$ (non-optimal) | private– $\mathcal{O}(c_M n^4(\log \frac{1}{\epsilon}))$ |
| [63] | $t < n/4$ (non-optimal) | private–$\mathcal{O}(c_M n^2(\log \frac{1}{\epsilon}))$ |

From the table, we find that the only known statistically secure AMPC with *optimal resilience* (i.e., with $n = 3t + 1$), involves very high communication complexity (the communication complexity analysis of the AMPC of [15] was not done earlier and for the sake of completeness, we carry out the same in subsection 9.5). Recently [30] presented an efficient MPC protocol over networks that have a synchronization point (the network is asynchronous before and after the synchronization point) and hence we do not compare it with our AMPC protocol, which is designed over completely asynchronous settings. Also we do not compare our protocol with the known cryptographically secure AMPC (where the adversary has bounded computing power) protocols presented in [51] and [52].

### *9.3. Our New Statistical AMPC Protocol with $n = 3t + 1$*

We design a statistically secure AMPC protocol with $n = 3t + 1$ which privately communicates $\mathcal{O}(n^5(\log \frac{1}{\epsilon}))$ bits per multiplication gate. Thus our AMPC protocol significantly improves the communication complexity of only known optimally resilient statistically secure AMPC protocol of [15]. For designing our AMPC protocol, we use our proposed ACSS scheme presented in section 8.

### *9.4. Primitives Used in Our AMPC Protocol*

In addition to the ACSS scheme proposed by us, our AMPC protocol also uses a well known primitive called *Agreement on Common Subset* (ACS) [10, 15].

**Agreement on Common Subset (ACS)** [10, 15]: It is an asynchronous primitive presented in [12, 15]. It outputs a common set, containing at least $n - t$ parties, who correctly shared their values. Moreover, each honest party will eventually get a share, corresponding to each value, shared by the parties in the common set. ACS requires private communication of $\mathcal{O}(\text{poly}(n, \log \frac{1}{\epsilon}))$ bits.

In our AMPC protocol, we use another simple protocol RNG very frequently, that allows the parties in $\mathcal{P}$ to jointly generate a random, non-zero element $r \in \mathbb{F}$. The protocol uses our ACSS scheme and the ACS protocol as black-boxes.

**Random Number Generation (RNG):** The protocol for random number generation works as follows: each $P_i \in \mathcal{P}$ shares a random non-zero $r_i \in \mathbb{F}$ using ACSS-Share. The parties then run ACS to agree on a common set, say $\mathcal{C}$ of at least $2t + 1$ parties who did proper sharing of their random values. Once $\mathcal{C}$ is agreed upon, ACSS-Rec is executed for every $P_i \in \mathcal{C}$ in order to reconstruct back $P_i$'s committed secret. Now every party in $\mathcal{P}$ locally add the committed secret of every $P_i \in \mathcal{C}$. It is easy to see that the sum value is random. We call this protocol as RNG. The protocol communicates $\mathcal{O}(\text{poly}(n, \log \frac{1}{\epsilon}))$ bits.

*9.5. The Approach Used in the AMPC of [15] and Current Article*

**AMPC of [15]:** The AMPC protocol of [15] consists of input phase and computation phase. In input phase every party $P_i$ shares (or commits to) his input $x_i$. All the parties then decide on a common set of $n-t$ parties (using ACS) who have done proper sharing of their input. Once this is done, in the computation phase the arithmetic circuit representing $f$ is computed gate by gate, such that the intermediate gate outputs are always kept as secret and are properly shared among the parties, following the approach of [13]. Now for sharing/committing inputs, a natural choice is to use AVSS protocol which can be treated as a form of *commitment*, where the commitment is held in a distributed fashion among the parties. Before [15], the only known statistical AVSS scheme with $n = 3t + 1$ was due to [22]. But it is shown in [15] that the use of the AVSS protocol of [22] for committing inputs (secrets), does not allow to compute the circuit robustly in a straight-forward way. This is because *for robust computation of the circuit, it is to be ensured that at the end of AVSS sharing phase, every honest party should have access to share of the secret.* Unfortunately the AVSS of [22] does not guarantee the above property, which we may refer as *ultimate* property. This very reason motivated Ben-Or et. al [15] to introduce a new asynchronous primitive called *Ultimate Secret Sharing* (USS) which not only ensures that every honest party has access to his share of the secret, but also offers all the properties of AVSS. Thus [15] presents an USS scheme with $n = 3t + 1$ using the AVSS protocol of [22] as a building block. Essentially, in the USS protocol of [15], every share of the secret is committed using AVSS of [22] which ensures that each honest party $P_i$ can have an access to the $i^{th}$ share of secret by means of private reconstruction of AVSS. A secret $s$ that is shared using USS is called *ultimately shared*. Now in the input phase of AMPC in [15], parties *ultimately share* their inputs. Then in the computation phase, for every gate (except output gate), *ultimate sharing* of the output is computed from the *ultimate sharing* of the inputs, following the approach of [13, 67].

Now we carry out the communication complexity analysis for the AMPC of [15]. Recently, in [61], the authors have analyzed that the sharing phase of the statistical AVSS scheme of [22] involves a private communication of $\Omega(n^9(\log \frac{1}{\epsilon})^4)$ bits and A-cast of $\Omega(n^9(\log \frac{1}{\epsilon})^2 \log(n))$ bits, for sharing a single secret. As the sharing phase of the USS scheme of [15] requires $n$ invocations to the sharing phase of AVSS of [22], it incurs a private communication of $\Omega(n^{10}(\log \frac{1}{\epsilon})^4)$ bits and A-cast of $\Omega(n^{10}(\log \frac{1}{\epsilon})^2 \log(n))$ bits. Finally in the AMPC protocol, each multiplication requires $n$ invocations to the sharing phase of USS. So evaluation of each multiplication gate incurs a private communication of $\Omega(n^{11}(\log \frac{1}{\epsilon})^4)$ and A-cast of $\Omega(n^{11}(\log \frac{1}{\epsilon})^2 \log(n))$ bits.

**AMPC of Current Article:** Our statistical AMPC protocol follows the preprocessing model of [3] and proceeds in a sequence of three phases: preparation phase, input phase and computation phase. Every honest party will eventually complete each phase with very high probability. We call a triple $(a, b, c)$ as a random multiplication triple if $a$, $b$ are random and $c = ab$. In the preparation phase, $t$-sharing of $c_M + c_R$ random multiplication triples are generated. Each

multiplication and random gate of the circuit is associated with a multiplication triple. In the input phase the parties $t$-share (commit to) their inputs and then agree on a common subset of $n - t$ parties (using ACS) who correctly shared their inputs. In the computation phase, the actual circuit will be computed gate by gate, based on the inputs of the parties in common set. Due to the linearity of the used secret-sharing, the linear gates can be computed locally. Each multiplication gate will be evaluated using the circuit randomization technique of [3] with the help of the associated multiplication triple (generated in preparation phase).

For committing/sharing secrets, we use our ACSS scheme. There is a slight *definitional difference* between the USS of [15] and our ACSS, though both of them offer all the properties of AVSS. While USS of [15] ensures that *every* honest party has *access* to share of secret (*but may not hold the share directly*), our ACSS ensures that *every honest party holds his share* of secret. This property of ACSS is called *completeness* property as mentioned in the definition of ACSS. The advantages of ACSS over USS are as follows:

1. It makes the computation of the gates very simple;
2. Reconstruction phase of ACSS is very simple, efficient and can be achieved using OEC of [19].

Apart from these advantages, our ACSS is strikingly better than USS of [15] in terms of communication complexity. While sharing phase of our ACSS privately communicates $\mathcal{O}((\ell n^4 + n^5 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits and A-casts $\mathcal{O}(n^4 \log n)$ bits to share $\ell$ *secrets concurrently*, the sharing phase of USS in [15] privately communicates $\Omega(n^{10}(\log \frac{1}{\epsilon})^4)$ bits and A-casts $\Omega(n^{10}(\log \frac{1}{\epsilon})^2 \log(n))$ bits to share *only one secret*.

We now proceed to the specific details of each of the three phases of our AMPC protocol. Prior to that, we design another protocol for generating $t$-2$d$-sharing (which is defined in the following section) that uses our ACSS scheme as a building block.

## 10. Generating $t$-2$d$-Sharing

For generating multiplication triples, we need to generate $t$-2$d$-sharing of secret(s) where $t$-2$d$-sharing is defined as follows:

**Definition 7 ($t$-2$d$-sharing [9]).** *A value $s$ is $t$-2$d$-shared among the parties in $\mathcal{P}$ if there exist degree-$t$ polynomials $f(x), f^1(x), \ldots, f^n(x)$ with $f(0) = s$ and for $i = 1, \ldots, n$, $f^i(0) = f(i)$ and* **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 a share-share $s_{ji} = f^j(i)$ of the share $s_j$ of every party $P_j \in \mathcal{P}$. We denote $t$-2$d$-sharing of $s$ by $[[s]]_t$.*

The $t$-2$d$-sharing of $s$ implies that $s$ as well as its shares are individually $t$-shared. Now we present a protocol t-2d-Share which allows a special party $D \in \mathcal{P}$ to simultaneously generate $t$-2$d$-sharing of $\ell \geq 1$ elements from $\mathbb{F}$, namely $s^1, \ldots, s^\ell$. In protocol t-2d-Share, the following happen with probability at least $(1-\epsilon)$: (a)

If $D$ is honest, then every honest party will eventually terminate t-2d-Share; (b) Moreover, if $D$ is corrupted and some honest party has terminated t-2d-Share, then eventually every other honest party will also terminate t-2d-Share; (c) Furthermore, if some honest party has terminated t-2d-Share, then it implies that $D$ has done correct $t$-$2d$-sharing of $s^1, \ldots, s^\ell$.

**The intuition**: The high level idea of the protocol is as follows: $D$ selects a random value $s^0 \in \mathbb{F}$ and hides each $s^i$ (where $i = 0, \ldots, \ell$) in the constant term of a random degree-$t$ polynomial $q^i(x)$. $D$ then $t$-shares the secret $S^0 = (s^0, \ldots, s^\ell)$, as well as their $i^{th}$ shares $S^i = (q^0(i), \ldots, q^\ell(i))$. The parties then jointly employ a verification technique to ensure that $D$ indeed $t$-shared $S^i$ for $i = 1, \ldots, n$ which are the shares of $S^0$. A similar verification technique was used in [9] in synchronous settings. The secret $s^0$ is used to ensure the secrecy of $s^1, \ldots, s^\ell$ during the verification process. After verification, the polynomials used for $t$-sharing $S_i$ are privately reconstructed by $P_i$, thus completing the $t$-$2d$-sharing of $s^1, \ldots, s^\ell$. The above idea is implemented in protocol t-2d-Share which is given in Fig. 19.

We now prove the properties of protocol t-2d-Share.

**Lemma 27.** *Protocol t-2d-Share satisfies the following properties:*

1. **Termination:** *If $D$ is honest, then except with probability $\epsilon$, all honest parties will eventually terminate t-2d-Share. Moreover if some honest party has terminated t-2d-Share, then every honest party will eventually terminate t-2d-Share, except with probability $\epsilon$.*
2. **Correctness:** *If some honest party has terminated the protocol then except with probability $\epsilon$, it is ensured that $D$ has done correct $t$-$2d$-sharing of $s^1, \ldots, s^\ell$.*
3. **Secrecy:** *If $D$ is honest then $s^1, \ldots, s^\ell$ will remain secure.*

PROOF: **Termination:** When $D$ is honest, every ACSS-MS-Share$_i$ initiated by honest $D$ will terminate with its desired output ($t$-sharings) except with probability $\epsilon'$. Therefore all the $n+1$ instances of ACSS-MS-Share$_i$ will terminate with their desired output (i.e., $t$-sharing of $S^0, \ldots, S^n$), except with probability $\epsilon'(n+1) \approx \epsilon$. Since $t$-sharing of $S^0, \ldots, S^n$ are generated properly, the verification steps specified in t-2d-Share will pass. Subsequently, $n$ instances of ACSS-MS-Rec-Private are executed in order to complete $t$-$2d$-sharing of $S$. Due to the property of ACSS-MS-Rec-Private, each honest $P_i$ will correctly reconstruct the required information corresponding to $t$-$2d$-sharing of $S$, namely $q_i^0(x), \ldots, q_i^\ell(x)$ and will terminate t-2d-Share except with probability $\epsilon'$. As there are at least $2t+1$ honest parties, except with probability $(2t+1)\epsilon' \approx \epsilon$, all honest parties will eventually terminate t-2d-Share with correct $t$-$2d$-sharing of $s^1, \ldots, s^\ell$. This completes the proof of the first part of **Termination** property. We now proceed to prove second part of the **Termination** property.

Let $P_i$ be an honest party who has terminated protocol t-2d-Share. This implies that $P_i$ has reconstructed $q_i^0(x), \ldots, q_i^\ell(x)$ from ACSS-MS-Rec-Private$_i$. This further means there is at least one honest party who checked that the

Figure 19: Protocol t-2d-Share for Generating $t$-$2d$-Sharing of $S = (s^1, \ldots, s^\ell)$, $n = 3t + 1$

## Protocol t-2d-Share($D, \mathcal{P}, S, \epsilon$)

SHARING BY $D$: CODE FOR $D$ — Only $D$ executes this code

1. Select a random $s^0$ and $\ell+1$ degree-$t$ random polynomials $q^0(x), \ldots, q^\ell(x)$ such that for $l = 0, \ldots, \ell$, $q^l(0) = s^l$. Let $s_i^l = q^l(i)$ and $S^i = (q^0(i), \ldots, q^\ell(i))$ for $i = 0, \ldots, n$. So $S^0 = (s^0, \ldots, s^\ell)$ and $S^i = (s_i^0, \ldots, s_i^\ell)$.
2. For $l = 0, \ldots, \ell$ and $i = 1, \ldots, n$, select random degree-$t$ polynomials $q_i^l(x)$, such that $q_i^l(0) = q^l(i) = s_i^l$. Let $S^{ij} = (q_i^0(j), q_i^1(j), \ldots, q_i^\ell(j)) = (s_{ij}^0, s_{ij}^1, \ldots, s_{ij}^\ell)$, for $j = 1, \ldots, n$.
3. Invoke ACSS-MS-Share($D, \mathcal{P}, (q^0(x), q^1(x), \ldots, q^\ell(x)), \epsilon'$), where $\epsilon' = \frac{\epsilon}{n+1}$, for generating $t$-sharing of $S^0$. Denote this instance of ACSS-MS-Share by ACSS-MS-Share$_0$. During ACSS-MS-Share$_0$, party $P_j$ receives the shares $S^j$ for $j = 1, \ldots, n$.
4. For $i = 1, \ldots, n$, invoke ACSS-MS-Share($D, \mathcal{P}, (q_i^0(x), q_i^1(x), \ldots, q_i^\ell(x)), \epsilon'$), where $\epsilon' = \frac{\epsilon}{n+1}$, for generating $t$-sharing of $S^i$. Denote this instance of ACSS-MS-Share by ACSS-MS-Share$_i$. During ACSS-MS-Share$_i$, party $P_j$ receives the share-shares $S^{ij}$, for $j = 1, \ldots, n$.

VERIFICATION: CODE FOR $P_i$ — Every party in $\mathcal{P}$ executes this code

1. Upon completion of ACSS-MS-Share$_j$ for all $j \in \{0, \ldots, n\}$, participate in protocol RNG to generate random $r$.
2. Wait to terminate RNG with $r$ as output. Compute $s_i^* = \sum_{l=0}^{\ell} r^l s_i^l$ which is the $i^{th}$ share of $s^* = \sum_{l=0}^{\ell} r^l s^l$. In addition, for $j = 1, \ldots, n$, locally compute $s_{ji}^* = \sum_{l=0}^{\ell} r^l s_{ji}^l$ which is the $i^{th}$ share-share of $s_j^*$.
3. Participate in ACSS-MS-Rec($D, \mathcal{P}, (s^*, s_1^*, \ldots, s_n^*), \epsilon$) to publicly reconstruct $s^*, s_1^*, \ldots, s_n^*$. This results in every party reconstructing $q^*(x)$ and $q_1^*(x), \ldots, q_n^*(x)$ with $q^*(0) = s^*$ and $q_i^*(0) = s_i^*$.
4. Check whether for $i = 1, \ldots, n$, $q^*(i) \stackrel{?}{=} q_i^*(0)$. If yes proceed to the next step assuming that $D$ has done proper $t$-sharing of $S^j$ for $j = 0, \ldots, n$.

$P_j$-PRIVATE RECONSTRUCTION OF POLYNOMIALS USED FOR SHARING $S^j$: (CODE FOR $P_i$): — Every party in $\mathcal{P}$ executes this code

1. For $j = 1, \ldots, n$, participate in ACSS-MS-Rec-Private($D, \mathcal{P}, S^j, P_j, \epsilon'$) for enabling $P_j$ to privately reconstruct the polynomials $q^{(0,j)}(x), \ldots, q^{(\ell,j)}(x)$ which were used by $D$ to share $S^j$. We refer ACSS-MS-Rec-Private($D, \mathcal{P}, S^j, P_j, \epsilon'$) by ACSS-MS-Rec-Private$_j$.
2. Wait to privately reconstruct degree-$t$ polynomials $q_i^0(x), \ldots, q_i^\ell(x)$ from ACSS-MS-Rec-Private$_i$ and terminate t-2d-Share.

public verification has passed and participated in every ACSS-MS-Rec-Private$_j$ for $j = 1, \ldots, n$. As the verification process is public, every other honest party will eventually see that the verification passes and then they will participate in ACSS-MS-Rec-Private$_j$ for $j = 1, \ldots, n$. Now by the termination property of ACSS-MS-Rec-Private, all honest $P_j$s will terminate ACSS-MS-Rec-Private$_j$, output $q_j^0(x), \ldots, q_j^\ell(x)$ and finally terminate protocol t-2d-Share, except with probability $n\epsilon' \approx \epsilon$.

**Correctness:** Here we consider two cases: (a) when $D$ is honest; (b) when $D$ is corrupted.

1. When $D$ is honest, then correctness follows from the proof of the first part of termination property.
2. Now we consider $D$ to be corrupted. For $i = 0, \ldots, n$, ACSS-MS-Share$_i$ ensures that $D$ has correctly $t$-shared some $\overline{S^i}$, except with probability $\epsilon'$ (by Completeness property of ACSS-MS). But it may happen that $\overline{S^i}$ which is $t$-shared by $D$ in ACSS-MS-Share$_i$, is not the correct $i^{th}$ shares of $S^0$. Assume that $D$ has $t$-shared $\overline{S^j} \neq S^j$ in ACSS-MS-Share$_j$, for some $j \in \{1, \ldots, n\}$. This implies that in ACSS-MS-Share$_j$, $D$ has used polynomials $\overline{q_j^0}(x), \ldots, \overline{q_j^\ell}(x)$ to share $\overline{S^j}$, such that for at least one $l \in \{0, \ldots, \ell\}$, $\overline{q_j^l}(0) \neq q^l(j) = s_j^l$. That is, $\overline{q_j^l}(0) = \overline{s_j^l} \neq s_j^l$. Now consider $q_j^*(0) = s_j^0 + rs_j^1 + \ldots + r^l\overline{s_j^l} + \ldots + r^\ell s_j^\ell$. We claim that with very high probability $q^*(j) \neq q_j^*(0)$. The probability that $q^*(j) = q_j^*(0)$ is same as the probability that two different degree-$\ell$ polynomials with coefficients $(s_j^0, \ldots, \overline{s_j^l}, \ldots, s_j^\ell)$ and $(s_j^0, \ldots, s_j^l, \ldots, s_j^\ell)$ respectively, intersect at a random value $r$. Since any two degree-$\ell$ polynomial can intersect each other at most at $\ell$ values, $r$ has to be one of these $\ell$ values. But $r$ is chosen randomly after the completion of all ACSS-MS-Share$_j$ for $j = 0, \ldots, n$ (so during executions of ACSS-MS-Share$_i$'s, $D$ is unaware of $r$). So the above event can happen with probability at most $\frac{\ell}{|\mathbb{F}|} \approx \epsilon$. Thus with probability at least $1 - \epsilon$, $q^*(j) \neq q_j^*(0)$ and hence no honest party will terminate the protocol if $D$ has $t$-shared $\overline{S^j} \neq S^j$ for some $j$. So if some honest party has terminated t-2d-Share, then corrupted $D$ must have attempted to $t$-share correct $S^i$ in each ACSS-MS-Share$_i$, for $i = 1, \ldots, n$. Now rest of the proof will follow from the proof of part 1 of **Correctness**.

**Secrecy:** When $D$ is honest, ACSS-MS-Share$_0$ does not leak any information on $s^1, \ldots, s^\ell$ (from the secrecy property of ACSS-MS-Share). Later $s^* = q^*(0)$ does not leak any information about $s^1, \ldots, s^\ell$ during verification process, as $s^0$ is randomly chosen by $D$ and therefore $s^*$ will look completely random for the adversary $\mathcal{A}_t$. □

**Lemma 28.** *Protocol t-2d-Share privately communicates* $\mathcal{O}((\ell n^5 + n^6 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ *bits and A-cast* $\mathcal{O}(n^5 \log(n))$ *bits.*

PROOF: Follows from the fact that in protocol t-2d-Share, there are $n+1$ instances of ACSS-MS-Share, one instance of RNG, one instance of ACSS-MS-Rec and $n$ instances of ACSS-MS-Rec-Private. □

## 11. Preparation Phase

Here we generate $t$-sharing of $c_M + c_R$ *secret* random *multiplication triples* $(a^k, b^k, c^k)$, such that for $k = 1, \ldots, c_M + c_R$, $c^k = a^k b^k$. For this we first generate $t$-$2d$-sharing of *secret* random doubles $([[a^k]]_t, [[b^k]]_t)$ for $k = 1, \ldots, c_M + c_R$. Given these random doubles, we generate $t$-sharing of $c^k$, for $k = 1, \ldots, c_M + c_R$, by adapting a technique from [27] which was given for synchronous settings.

### *11.1. Generating Secret Random t-2d-Sharing*

In section 10, we have presented a protocol called t-2d-Share which allows a $D \in \mathcal{P}$ to generate $t$-$2d$-sharing of $\ell$ secrets. We now present a protocol called Random-t-2d-Share which allows all the parties in $\mathcal{P}$ to jointly generate random $t$-$2d$-sharing of $\ell$ secrets, unknown to $\mathcal{A}_t$. The protocol terminates and generates its desired output except with probability $\epsilon$. Random-t-2d-Share asks individual party to act as dealer and $t$-$2d$-Share $\frac{\ell}{n-2t}$ random secrets. Then we run ACS protocol to agree on a common set of $n-t$ parties who have correctly $t$-$2d$-shared $\frac{\ell}{n-2t}$ random secrets. Now out of these $n-t$ parties, at least $n-2t$ are honest. Hence the secrets that are $t$-$2d$-shared by these $n-2t$ honest parties are truly random and unknown to $\mathcal{A}_t$. So if we consider the $\frac{\ell}{n-2t}$ $t$-$2d$-sharing done by each of the honest parties in common set, then we will get $\frac{\ell}{n-2t} * (n-2t) = \ell$ random $t$-$2d$-sharing in total. For this, we use *Vandermonde Matrix* [31] and its ability to extract randomness which was also exploited in [68, 31, 10].

**Vandermonde Matrix and Randomness Extraction [31]:** Let $\beta_1, \ldots, \beta_c$ be distinct 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 Vandermonde matrix is as follows: without loss of generality, assume that $r > c$. Moreover, let $(x_1, \ldots, x_r)$ be 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 if we compute $(y_1, \ldots, y_c) = (x_1, \ldots, x_r)V$, then $(y_1, \ldots, y_c)$ is an uniformly random vector of length $c$, extracted from $(x_1, \ldots, x_r)$. For proof of this, see [31, 10].

Protocol Random-t-2d-Share is now presented in Fig. 20.

**Lemma 29.** *Protocol Random-t-2d-Share satisfies the following properties:*

1. **Termination***: All honest parties eventually terminate the protocol with probability at least* $(1-\epsilon)$.
2. **Correctness***: The protocol outputs correct $t$-$2d$-sharing of $\ell$ values with probability at least* $(1-\epsilon)$.

Figure 20: Protocol for Collectively Generating $t$-2$d$-Sharing of $\ell$ secrets, $n = 3t + 1$

## Protocol Random-t-2d-Share$(\mathcal{P}, \ell, \epsilon)$

CODE FOR $P_i$: — Every party executes this code

1. Select $L = \frac{\ell}{n-2t}$ random secret elements $(s^{(i,1)}, \ldots, s^{(i,L)})$. As a dealer, invoke t-2d-Share$(P_i, \mathcal{P}, S^i, \epsilon')$, with $\epsilon' = \frac{\epsilon}{n}$, to generate $t$-2$d$-sharing of $S^i = (s^{(i,1)}, \ldots, s^{(i,L)})$.
2. For $j = 1, \ldots, n$, participate in t-2d-Share$(P_j, \mathcal{P}, S^j, \epsilon')$.

AGREEMENT ON A COMMON SET: CODE FOR $P_i$ — Every party executes this code

1. Create an accumulative set $C^i = \emptyset$. Upon terminating t-2d-Share$(P_j, \mathcal{P}, S^j, \epsilon')$, include $P_j$ in $C^i$.
2. Take part in ACS with the accumulative set $C^i$ as input.

GENERATION OF RANDOM $t$-2$d$-SHARING: CODE FOR $P_i$: — Every party executes this code

1. Wait until ACS completes with output $C$ containing $n-t$ parties. For every $P_j \in C$, obtain the $i^{th}$ shares $s_i^{(j,1)}, \ldots, s_i^{(j,L)}$ of $S^j$ and $i^{th}$ share-shares $s_{ki}^{(j,1)}, \ldots, s_{ki}^{(j,L)}$ of shares $s_k^{(j,1)}, \ldots, s_k^{(j,L)}$, corresponding to each $P_k$, for $k = 1, \ldots, n$. Without loss of generality, let $C = \{P_1, \ldots, P_{n-t}\}$.
2. Let $V$ denote an $(n-t) \times (n-2t)$ publicly known *Vandermonde Matrix*.
   (a) For every $k \in \{1, \ldots, L\}$, let $(r^{(1,k)}, \ldots, r^{(n-2t,k)}) = (s^{(1,k)}, \ldots, s^{(n-t,k)})V$.
   (b) Locally compute $i^{th}$ share of $r^{(1,k)}, \ldots, r^{(n-2t,k)}$ as $(r_i^{(1,k)}, \ldots, r_i^{(n-2t,k)}) = (s_i^{(1,k)}, \ldots, s_i^{(n-t,k)})V$.
   (c) For each $1 \leq j \leq n$, locally compute the $i^{th}$ share-share of share $(r_j^{(1,k)}, \ldots, r_j^{(n-2t,k)})$ as $(r_{ji}^{(1,k)}, \ldots, r_{ji}^{(n-2t,k)}) = (s_{ji}^{(1,k)}, \ldots, s_{ji}^{(n-t,k)})V$ and terminate **Random-t-2d-Share**.

The values $r^{(1,1)}, \ldots, r^{(n-2t,1)}, \ldots, r^{(1,L)}, \ldots, r^{(n-2t,L)}$ denote the $\ell$ random secrets which are $t$-2$d$-shared.

3. **Secrecy**: *The $\ell$ values whose $t$-2$d$-sharing is generated by the protocol will be completely random and unknown to $\mathcal{A}_t$.*

PROOF: **Termination:** By the **Termination** property of t-2d-Share (see Lemma 27), every instance of t-2d-Share initiated by an honest $P_i$ as a dealer will be eventually terminated by all honest parties, except with probability $\epsilon'$. Moreover, if an honest party terminates an instance of t-2d-Share (initiated by some party), then eventually every other honest party will terminate that instance of t-2d-Share , except with probability $\epsilon'$. Now since there are at least $2t + 1$ honest parties, except with probability $(2t + 1)\epsilon' \approx \epsilon$, all instances of t-2d-Share initiated by honest parties will be terminated by every honest party. So eventually protocol ACS will output a common set $C$ of size $n - t$, except with probability $\epsilon$, such that all instances of t-2d-Share initiated by the parties in $C$ will be eventually terminated by all honest parties in $\mathcal{P}$. This proves the **Termination** property.

**Correctness:** From the **Correctness** property of t-2d-Share (see Lemma 27), each instance of t-2d-Share initiated by a party in $C$ will correctly generate $t$-$2d$-sharing of corresponding secrets, except with probability $\epsilon'$. So with probability at most $(n-t)\epsilon' \approx \epsilon$, all instances of t-2d-Share initiated by the parties in $C$ will correctly generate $t$-$2d$-sharing of corresponding secrets. Hence except with probability $\epsilon$, protocol Random-t-2d-Share will correctly generate the $t$-$2d$-sharing of $\ell$ values. This proves the **Correctness** property.

**Secrecy:** From the **Secrecy** property of t-2d-Share (see Lemma 27), the values which are $t$-$2d$-shared by an honest party using t-2d-Share are completely random and are unknown to $\mathcal{A}_t$. Now there are at least $(n-t)-t = n-2t$ honest parties in $C$ and hence the $L$ values which are $t$-$2d$-shared by each them will be completely random and unknown to $\mathcal{A}_t$. Now from the randomness extraction property of Vandermonde Matrix, the values $r^{(1,1)}, \ldots, r^{(n-2t,1)}, \ldots, r^{(1,L)}, \ldots, r^{(n-2t,L)}$ will be completely random and unknown to $\mathcal{A}_t$. This proves the **Secrecy** property. □

**Lemma 30.** *Protocol Random-t-2d-Share privately communicates* $\mathcal{O}((\ell n^5 + n^7 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ *bits, A-cast* $\mathcal{O}(n^6 \log n)$ *bits and requires one invocation of ACS.*

PROOF: The communication complexity follows from the fact that in protocol Random-t-2d-Share, $n$ instances of t-2d-Share, each dealing with $\frac{\ell}{n-2t} = \frac{\ell}{\Theta(n)}$ values and having an error probability of $\frac{\epsilon}{n}$ are executed. □

*11.2. Proving $c = ab$*

Consider the following problem: let $D \in \mathcal{P}$ has $t$-shared $\ell$ pair of values $(a^1, b^1), \ldots, (a^\ell, b^\ell)$. Now $D$ wants to $t$-share $c^1, \ldots, c^\ell$ where $c^l = a^l b^l$, for $l = 1, \ldots, \ell$. Moreover, during this process, $D$ does not want to leak any *additional* information about $a^l$, $b^l$ and $c^l$. We propose a protocol ProveCeqAB to achieve this task in asynchronous settings, following a technique proposed in [27] for synchronous settings. The idea of the protocol for a single pair $(a, b)$ is as follows. $D$ selects a random non-zero $\beta \in \mathbb{F}$ and generates $t$-sharing of $c, \beta$ and $\beta b$. Then all the parties in $\mathcal{P}$ jointly generate a random value $r$. Each party locally computes the sharing of $p = ra + \beta$ and then $p$ is publicly reconstructed. Then each party locally computes the sharing of $q = pb - b\beta - rc = (ra+\beta)b - b\beta - rc$ and then $q$ is publicly reconstructed. If $q = 0$, then each party believes that with very high probability, $D$ has indeed $t$-shared $c = ab$. Moreover, if $D$ is honest then $a, b$ and $c$ will remain information theoretically secure. If $D$ is honest, then every honest party will eventually complete ProveCeqAB, and if some honest party has completed ProveCeqAB, then all the honest parties will eventually complete ProveCeqAB.

The error probability of the protocol is negligible because of the random $r$ which is jointly generated by all the parties after $c$, $\beta$ and $b\beta$ is $t$-shared by $D$. Specifically, a corrupted D might have shared $\overline{\beta b} \neq \beta b$ and $\overline{c} \neq c$ but still $q$ can be zero and this will happen iff $\overline{\beta b} + r\overline{c} = \beta b + rc$. However this equation is satisfied by only one value of $r$. Since $r$ is randomly generated, independent of

$D$, the probability that the equality will hold is $\frac{1}{|\mathbb{F}|}$ which is negligibly small. The secrecy follows from the fact that $p$ and $q$ are independent of $a$, $b$ and $c$. Now we can extend the above idea parallely for each of the $\ell$ pairs $(a^l, b^l)$.

Figure 21: Protocol for Generating $t$-sharing of $[c^1]_t = [a^1]_t.[b^1]_t, \ldots, [c^\ell]_t = [a^\ell]_t.[b^\ell]_t$, $n = 3t+1$

**Protocol ProveCeqAB($D, \mathcal{P}, [a^1]_t, \ldots, [a^\ell]_t, [b^1]_t, \ldots, [b^\ell]_t, \epsilon$)**

SHARING BY $D$:

1. CODE FOR D:
   (a) Select $\ell$ non-zero random elements $\beta^1, \ldots, \beta^\ell$. For $l = 1, \ldots, \ell$, let $c^l = a^l b^l$ and $d^l = b^l \beta^l$. Let $\mathcal{B} = (\beta^1, \ldots, \beta^\ell)$, $\mathcal{C} = (c^1, \ldots, c^\ell)$ and $\Lambda = (d^1, \ldots, d^\ell)$.
   (b) Invoke ACSS-MS-Share($D, \mathcal{P}, \mathcal{B}, \frac{\epsilon}{3}$), ACSS-MS-Share($D, \mathcal{P}, \mathcal{C}, \frac{\epsilon}{3}$) and ACSS-MS-Share($D, \mathcal{P}, \Lambda, \frac{\epsilon}{3}$).
2. CODE FOR $P_i$: Participate in the ACSS-MS-Share protocols initiated by $D$ to obtain the $i^{th}$ share $(\beta_i^1, \ldots, \beta_i^\ell)$, $(c_i^1, \ldots, c_i^\ell)$ and $(d_i^1, \ldots, d_i^\ell)$ of $\mathcal{B}, \mathcal{C}$ and $\Lambda$ respectively.

VERIFYING WHETHER $c^l = a^l.b^l$: CODE FOR $P_i$ — Every party executes this code

1. Once the three instances of ACSS-MS-Share initiated by $D$ are terminated, participate in protocol RNG to jointly generate a random non-zero value $r$.
2. For $l = 1, \ldots, \ell$, locally compute $p_i^l = ra_i^l + \beta_i^l$, the $i^{th}$ share $p^l = ra^l + \beta^l$. Participate in ACSS-MS-Rec($D, \mathcal{P}, (p^1, \ldots, p^\ell), \epsilon$) to publicly reconstruct $p^l$ for $l = 1, \ldots, \ell$.
3. Upon reconstruction of $p^l$'s, locally compute $q_i^l = p^l b_i^l - d_i^l - rc_i^l$ for $l = 1, \ldots, \ell$, to get the $i^{th}$ share of $q^l = p^l b^l - d^l - rc^l$. Participate in ACSS-MS-Rec($D, \mathcal{P}, (q^1, \ldots, q^\ell), \epsilon$) to publicly reconstruct $q^l$ for $l = 1, \ldots, \ell$.
4. Upon reconstruction of $q^l$'s, locally check whether for $l = 1, \ldots, \ell$, $q^l \stackrel{?}{=} 0$. If yes then terminate ProveCeqAB.

**Lemma 31.** *Protocol ProveCeqAB satisfies the following properties:*

1. **Termination:** *If $D$ is honest then all honest parties will terminate the protocol, except with probability $\epsilon$. If some honest party terminates the protocol, then eventually every other honest party will terminate the protocol, except with probability $\epsilon$.*
2. **Correctness:** *If some honest party terminates the protocol, then except with probability $\epsilon$, $D$ has $t$-shared $c^l = a^l b^l$, for $l = 1, \ldots, \ell$.*
3. **Secrecy:** *If $D$ is honest then $a^l, b^l, c^l$ will be information theoretically secure for all $l = 1, \ldots, \ell$.*

PROOF: **Termination**: When $D$ is honest, all three instances of ACSS-MS-Share will terminate and correctly generate $t$-sharing of $\mathcal{B}$, $\mathcal{C}$ and $\Lambda$ with probability at least $(1 - 3\frac{\epsilon}{3}) = (1 - \epsilon)$. Consequently in verification steps, both the instances of ACSS-MS-Rec will terminate and reconstruct proper values with probability at least $1 - \epsilon$. Now it follows that the condition specified in step

4 of the protocol will pass with probability at least $1-\epsilon$. Hence when $D$ is honest then the protocol will terminate with probability at least $1-\epsilon$. If some honest party has terminated the protocol, then $q^l = 0$ has been satisfied for all $l$. Every other honest party will also check the same and terminate the protocol.

**Correctness**: If some honest party terminates the protocol, then it implies that $q^l = 0$ for all $l = 1, \ldots, \ell$. Now notice that $q^l = p^l b^l - d^l - rc^l = (ra^l + \beta^l)b^l - b^l\beta^l - rc^l = ra^l b^l - rc^l = r(a^l b^l - c^l)$. Now if corrupted $D$ shares $c^l \neq a^l b^l$, then with probability $(1 - \frac{1}{|\mathbb{F}|}) = (1-\epsilon)$, the value $q^l = r(a^l b^l - c^l)$ will not be equal to zero. This is because random $r$ has been generated only after $D$ has committed all $a^l, b^l$ and $c^l$s. Thus if some honest party terminates the protocol, then with probability $(1-\epsilon)$, $D$ has $t$-shared $a^l, b^l$ and $c^l$ satisfying $c^l = a^l b^l$ for all $l = 1, \ldots, \ell$.

**Secrecy:** We now prove the secrecy of $a^l, b^l, c^l$ for all $l = 1, \ldots, \ell$ when $D$ is honest. From the secrecy property of ACSS-MS-Share, $a^l, b^l, c^l$ will remain secure after their $t$-sharing. Now we will show that both $p^l$ and $q^l$ will not leak any information about $a^l, b^l, c^l$. Clearly $p^l = (ra^l + \beta^l)$ will look completely random to adversary $\mathcal{A}_t$ as $\beta^l$ is randomly chosen. Furthermore $q^l = 0$ and hence $q^l$ does not leak any information on $a^l, b^l, c^l$. Hence the lemma. □

**Lemma 32.** *Protocol ProveCeqAB privately communicates* $\mathcal{O}((\ell n^4 + n^5 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ *bits and A-casts* $\mathcal{O}(n^4 \log n)$ *bits.*

PROOF: The proof follows from the fact that $\Theta(1)$ instances of ACSS-MS-Share and ACSS-MS-Rec are executed in protocol ProveCeqAB. □

### *11.3. Generating Multiplication Triples: The Main Protocol For Preparation Phase*

We now outline protocol PreparationPhase which generates $t$-sharing of $c_M + c_R$ random multiplication triples. We explain the idea for a single triplet $(a, b, c)$. First, Random-t-2d-Share is invoked to generate $t$-$2d$-sharing of $(a, b)$ which results in $P_i$ holding $i^{th}$ share of $a$ and $b$, namely $a_i$ and $b_i$ respectively. Now if each $P_i$ locally computes $e_i = a_i b_i$, then this results in $2t$-sharing of $c$. But we want each (honest) $P_i$ to hold $c_i$, where $(c_1, \ldots, c_n)$ is the $t$-sharing of $c$. For this we adapt a technique given in [41] for synchronous settings: Each $P_i$ invokes ProveCeqAB to $t$-share $e_i$. Now an instance of ACS will be executed to agree on a common set of $n - t = 2t + 1$ parties whose instances of ProveCeqAB has been terminated. For simplicity let this set contains $P_1, \ldots, P_{2t+1}$. Since $e_1, \ldots, e_{2t+1}$ are $2t+1$ distinct points on a degree-$2t$ polynomial, say $C(x)$ where $C(0) = c$ (and $C(i) = e_i$ for $i = 1, \ldots, 2t+1$), by Lagrange interpolation formula [25], $c$ can be computed as $c = \sum_{i=1}^{2t+1} r_i e_i$ where $r_i = \prod_{j=1, j\neq i}^{2t+1} \frac{x-j}{i-j}$. The vector $(r_1, \ldots, r_{2t+1})$ is called recombination vector [25] and is known publicly. Now to get $t$-sharing of $c$, $P_j$ locally computes $c_j = \sum_{i=1}^{2t+1} r_i e_{ij}$ where $e_{ij}$ is $j^{th}$ share of $e_i$. By the properties of ProveCeqAB, each $P_i$ in common set has indeed $t$-shared $e_i = a_i b_i$ with very high probability. So by performing the

above computation, correct $t$-sharing of $c = ab$ will be generated with very high probability. Moreover, $a, b$ and $c$ will remain secure. Protocol PreparationPhase is now given in Fig. 22.

Figure 22: Protocol for Generating $t$-sharing of $c_M + c_R$ secret random multiple triples

**Protocol PreparationPhase($\mathcal{P}, \epsilon$)**

CODE FOR $P_i$ — Every party executes this code

1. Participate in two instances of Random-t-2$d$-Share($\mathcal{P}, c_M + c_R, \frac{\epsilon}{2}$) to generate $t$-2$d$-sharing of $a^1, \ldots, a^{c_M+c_R}$ and $b^1, \ldots, b^{c_M+c_R}$. Obtain the $i^{th}$ shares $a_i^1, \ldots, a_i^{c_M+c_R}, b_i^1, \ldots, b_i^{c_M+c_R}$ and share-shares $a_{ji}^1, \ldots, a_{ji}^{c_M+c_R}, b_{ji}^1, \ldots, b_{ji}^{c_M+c_R}$, for $j = 1, \ldots, n$.
2. Let $c^k = a^k b^k$, for $k = 1, \ldots, c_M + c_R$. Upon termination of both the instances of Random-t-2d-Share, invoke ProveCeqAB($P_i, \mathcal{P}, [a_i^1]_t, \ldots, [a_i^{c_M+c_R}]_t, [b_i^1]_t, \ldots, [b_i^{c_M+c_R}]_t, \frac{\epsilon}{n}$) as a dealer, to generate $t$-sharing of $c_i^1, \ldots, c_i^{c_M+c_R}$, where $c_i^k$ is the $i^{th}$ share of $c^k$. We refer the instance of ProveCeqAB initiated by $P_i$ as ProveCeqAB$_i$.
3. For $j = 1, \ldots, n$, participate in ProveCeqAB$_j$.

AGREEMENT ON A COMMON-SET: CODE FOR $P_i$ — Every party executes this code

1. Create an accumulative set $C^i = \emptyset$. Upon completing ProveCeqAB$_j$ initiated by $P_j$ as a dealer, add $P_j$ in $C^i$.
2. Take part in ACS with the accumulative set $C^i$ as input.

GENERATION OF $t$-SHARING OF $c^1, \ldots, c^{c_M+c_R}$: CODE FOR $P_i$ — Every party executes this code

1. Wait until ACS completes with output $C$ containing $2t + 1$ parties. For simplicity, assume that $C = \{P_1, \ldots, P_{2t+1}\}$.
2. For $k = 1, \ldots, c_M + c_R$, locally compute $c_i^k = \sum_{j=1}^{2t+1} r_j c_{ji}^k$ the $i^{th}$ share of $c^k = r_1 c_1^k + \ldots + r_{2t+1} c_{2t+1}^k$, where $(r_1, \ldots, r_{2t+1})$ is the publicly known recombination vector.

We now prove the properties of protocol PreparationPhase.

**Lemma 33.** *Protocol PreparationPhase satisfies the following properties:*

1. **Termination***: All honest parties will eventually terminate PreparationPhase, except with probability $\epsilon$.*
2. **Correctness***: The protocol correctly outputs $t$-sharing of $c_M + c_R$ multiplication triples, except with probability $\epsilon$.*
3. **Secrecy***: The adversary $\mathcal{A}_t$ will have no information about $(a^k, b^k, c^k)$, for $k = 1, \ldots, c_M + c_R$.*

PROOF: **Termination:** Following the termination property of Random-t-2d-Share, both the instances of Random-t-2d-Share will terminate except with probability $2\frac{\epsilon}{2} = \epsilon$. Now ProveCeqAB$_j$ invoked by an honest $P_j$ will be eventually terminated by all honest parties, except with probability $\frac{\epsilon}{n}$. Moreover, if some

honest party terminates protocol ProveCeqAB$_j$ for any $P_j$, then eventually every other honest party will terminate the protocol, except with probability $\frac{\epsilon}{n}$. Now since there are at least $2t+1$ honest parties, except with probability $(2t+1)\frac{\epsilon}{n} \approx \epsilon$, at least $2t+1$ instances of ProveCeqAB will be eventually terminated by all honest parties. So eventually, all honest parties will agree on a common set $C$ containing $n-t$ parties, except with probability $\epsilon$, such that the instances of ProveCeqAB initiated by every party in $C$ is terminated by all honest parties in $\mathcal{P}$. Once this is done, every party will terminate protocol PreparationPhase after doing location computation. So all honest parties will terminate protocol PreparationPhase with probability at least $(1-\epsilon)$.

**Correctness**: Follows from the correctness property of protocol Random-t-2d-Share and ProveCeqAB.

**Secrecy**: Follows from the secrecy property of protocol Random-t-2d-Share and ProveCeqAB. □

**Lemma 34.** *Protocol PreparationPhase privately communicates* $\mathcal{O}(((c_M+c_R)n^5+n^7\log\frac{1}{\epsilon})\log\frac{1}{\epsilon})$ *bits, A-cast* $\mathcal{O}(n^6\log n)$ *bits and requires three invocations of ACS.*

## 12. Input Phase

In protocol InputPhase, each $P_i$ acts as a dealer to $t$-share his input $X_i$ containing $c_i$ values. So $c_I = \sum_{i=1}^{n} c_i$, where $c_I$ is the number of input gates in the circuit. The parties then agree on a set of at least $n-t$ parties (whose inputs will be taken into consideration for computation), by executing an ACS. Protocol InputPhase is now presented in Fig. 23.

Figure 23: Protocol for Input Phase (Sharing Inputs), $n = 3t+1$

### Protocol InputPhase($\mathcal{P}, \epsilon$)

Secret Sharing: Code for $P_i$ — Every party executes this code

1. On input $X_i$, invoke ACSS-MS-Share($P_i, \mathcal{P}, X_i, \frac{\epsilon}{n}$) as a dealer to generate $t$-sharing of $X_i$.
2. For every $j = 1, \ldots, n$, participate in ACSS-MS-Share($P_j, \mathcal{P}, X_j, \frac{\epsilon}{n}$).

Agreement on a Common-Set: Code for $P_i$ — Every party executes this code

1. Create an accumulative set $C^i = \emptyset$. Upon completing ACSS-MS-Share($P_j, \mathcal{P}, X_j, \frac{\epsilon}{n}$) invoked by $P_j$ as a dealer, add $P_j$ in $C^i$.
2. Participate in ACS with the accumulative set $C^i$ as input.
3. Output common set $C$ containing $n-t$ parties and local shares of all inputs corresponding to parties in $C$.

**Lemma 35.** *Protocol InputPhase satisfies the following properties:*

1. **Termination***: All honest parties will eventually terminate the protocol, except with probability $\epsilon$.*
2. **Correctness***: The protocol correctly outputs $t$-sharing of inputs of the parties in agreed common set $C$, except with probability $\epsilon$.*
3. **Secrecy***: The adversary $\mathcal{A}_t$ will have no information about the inputs of the honest parties in set $C$.*

PROOF: The proof follows from the properties of ACSS-MS-Share. □

**Lemma 36.** *Protocol InputPhase privately communicates $\mathcal{O}((c_I n^4 + n^6 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits, A-casts $\mathcal{O}(n^5 \log n)$ bits and requires one invocation of ACS.*

## 13. Computation Phase

Once the input phase is over, in the computation phase, the circuit is evaluated gate by gate, where all inputs and intermediate values are $t$-shared among the parties. As soon as a party holds his shares of the input values of a gate, he joins the computation of the gate.

Due to the linearity of $t$-sharing, linear gates can be computed locally by applying the linear function to the shares, i.e. for any linear function $c = f(a, b)$, the sharing $[c]_t$ is computed by letting every party $P_i$ to compute $c_i = f(a_i, b_i)$, where $a_i, b_i$ and $c_i$ are the $i^{th}$ shares of $a, b$ and $c$ respectively. With every random gate, one random triple (from the preparation phase) is associated, whose first component is directly used as outcome of the random gate. With every multiplication gate, one random triple (from the preparation phase) is associated, which is then used to compute $t$-sharing of the product, following the circuit randomization technique of Beaver [3]. Given a pre-generated random multiplication triple (which is already correctly $t$-shared) *Circuit Randomization* [3] allows to evaluate a multiplication gate at the cost of two public reconstructions. Let $z = xy$, where $x, y$ are the inputs of the multiplication gate. Now $z$ can be expressed as $z = ((x-a)+a)((y-b)+b) = (\alpha+a)(\beta+b)$, where $(a, b, c)$ is a random multiplication triple. So given $([a]_t, [b]_t, [c]_t)$, $[z]_t$ can be computed as $[z]_t = \alpha\beta + \alpha[b]_t + \beta[a]_t + [c]_t$ after reconstructing $\alpha$ and $\beta$ publicly. The security follows from the fact that $\alpha$ and $\beta$ are random, for a random $(a, b, c)$. Protocol ComputationPhase is now presented in Fig. 24.

**Lemma 37.** *Given that protocol PreparationPhase and InputPhase satisfy their properties specified in Lemma 33 and Lemma 35 respectively, Protocol ComputationPhase satisfies the following with probability at least $(1 - \epsilon)$:*

1. **Termination***: All honest parties will eventually terminate the protocol.*
2. **Correctness***: Given $t$-sharing of $c_M + c_R$ secret random triples, the protocol computes the outputs of the circuit correctly and privately.*

Figure 24: Protocol for Computation Phase (Evaluating the Circuit), $n = 3t + 1$

**Protocol ComputationPhase($\mathcal{P}, \epsilon$)**

FOR EVERY GATE IN THE CIRCUIT: CODE FOR $P_i$ — Every party executes this code
Wait until the $i^{th}$ share of each of the inputs of the gate is available. Now depending on the type of the gate, proceed as follows:

1. **Input Gate:** $[s]_t = \mathsf{IGate}([s]_t)$: There is nothing to be done here.
2. **Linear Gate:** $[z]_t = \mathsf{LGate}([x]_t, [y]_t, \ldots)$: Compute $z_i = LGate(x_i, y_i, \ldots)$, the $i^{th}$ share of $z = \mathsf{LGate}(x, y, \ldots)$, where $x_i, y_i, \ldots$ denotes $i^{th}$ share of $x, y, \ldots$.
3. **Multiplication Gate:** $[z]_t = \mathsf{MGate}([x]_t, [y]_t, ([a^k]_t, [b^k]_t, [c^k]_t))$:
   (a) Let $([a^k]_t, [b^k]_t, [c^k]_t)$ be the random triple associated with the multiplication gate.
   (b) Compute $\alpha_i = x_i - a_i$ and $\beta_i = y_i - b_i$, the $i^{th}$ share of $\alpha = (x - a)$ and $\beta = (y - b)$ respectively.
   (c) Participate in ACSS-Rec to reconstruct $\alpha$ and $\beta$.
   (d) Upon reconstructing $\alpha$ and $\beta$, compute $z_i = \alpha\beta + \alpha b_i + \beta a_i + c_i$, the $i^{th}$ share of $z = \alpha\beta + \alpha b + \beta a + c = xy$.
4. **Random Gate:** $[r]_t = \mathsf{RGate}([a^k]_t, [b^k]_t, [c^k]_t)$: Let $([a^k]_t, [b^k]_t, [c^k]_t)$ be the random triple associated with the random gate. Compute $r_i = a_i^k$ as the $i^{th}$ share of $r$.
5. **Output Gate:** $x = \mathsf{OGate}([x]_t)$: Participate in ACSS-Rec to reconstruct $x$.

PROOF: Given that protocol PreparationPhase and InputPhase satisfy their **Termination** property specified in Lemma 33 and Lemma 35 respectively, termination of protocol ComputationPhase follows from the finiteness of the circuit representing function $f$ and the termination property of ACSS-Rec. Protocol PreparationPhase terminates with proper $t$-sharing of $c_M + c_R$ secret random triples, except with probability $\epsilon$. Also protocol InputPhase terminates with proper $t$-sharing of the inputs of the parties in common set $C$, except with probability $\epsilon$. Hence protocol ComputationPhase will correctly compute the circuit and eventually terminate with probability at least $(1 - (\epsilon + \epsilon)) \approx (1 - \epsilon)$. □

**Lemma 38.** *Protocol ComputationPhase privately communicates* $O(n^2(c_M + c_O) \log \frac{1}{\epsilon})$ *bits.*

PROOF: Follows from the fact that in protocol ComputationPhase, $2c_M + c_O$ instances of ACSS-Rec are executed, corresponding to $c_M$ multiplication gates and $c_O$ output gates. □

## 14. The New Statistical AMPC Protocol with Optimal Resilience

Now our new AMPC protocol called AMPC for evaluating function $f$ which is represented by a circuit containing $c_I, c_L, c_M, c_R$ and $c_O$ input, linear, multiplication, random and output gates respectively, is as follows: (1). Invoke

PreparationPhase($\mathcal{P}, \epsilon$); (2). Invoke InputPhase($\mathcal{P}, \epsilon$); (3). Invoke Computation-Phase($\mathcal{P}, \epsilon$).

**Theorem 11.** *Let $n = 3t + 1$. Then protocol AMPC satisfies the following properties:*

1. **Termination***: Except with probability $\epsilon$, all honest parties will eventually terminate the protocol.*
2. **Correctness***: Except with probability $\epsilon$, the protocol correctly computes the outputs of the circuit.*
3. **Secrecy***: The adversary $\mathcal{A}_t$ will get no extra information other than what can be inferred by the input and output of the corrupted parties.*
4. **Communication Complexity***: The protocol privately communicates $\mathcal{O}((c_I n^4 + c_M n^5 + c_R n^5 + c_O n^2 + n^7 \log \frac{1}{\epsilon})\frac{1}{\epsilon})$ bits, A-cast $\mathcal{O}(n^6 \log n)$ bits and requires 4 invocations of ACS.*

Proof: The proof follows from the properties of protocol Preparation Phase, Input Phase and Computation Phase. □

## 15. Conclusion and Open Problems

In this paper, we have presented a new statistical AVSS scheme with optimal resilience (i.e. with $n = 3t + 1$ parties), which significantly improves the communication complexity of only known optimally resilient statistical AVSS of [22] and [61]. Moreover, our AVSS achieves stronger properties than the AVSS of [61] with lesser communication complexity. Furthermore, using our AVSS scheme, we designed a new ACSS scheme which is an essential building block of statistical AMPC protocol with optimal resilience (i.e., with $n = 3t+1$). In fact, our ACSS scheme is the first ACSS scheme in the literature (in asynchronous settings). Our ACSS when employed for designing AMPC results in significant improvement over the only known statistical AMPC protocol of [15] (which does not employ any ACSS). The design approach of our AVSS and ACSS are novel and first of their kind. It is an interesting problem to further reduce the communication complexity of our AVSS and ACSS scheme which may lead to further reduction in the communication complexity of AMPC.

## References

[1] I. Abraham, D. Dolev, and J. Y. Halpern. An almost-surely terminating polynomial protocol for asynchronous Byzantine Agreement with optimal resilience. In R. A. Bazzi and B. Patt-Shamir, editors, *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC 2008, Toronto, Canada, August 18-21, 2008*, pages 405–414. ACM Press, 2008.

[2] J. Bar-Ilan and D. Beaver. Non-cryptographic fault-tolerant computing in constant number of rounds of interaction. In *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing, August 14-16, 1989, Edmonton, Alberta, Canada*, pages 201–209. ACM Press, 1989.

[3] D. Beaver. Efficient multiparty protocols using circuit randomization. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 420–432. Springer Verlag, 1991.

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

[5] D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway. Security with low communication overhead. In A. Menezes and S. A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 62–76. Springer Verlag, 1990.

[6] D. Beaver and S. Haber. Cryptographic protocols provably secure against dynamic adversaries. In R. A. Rueppel, editor, *Advances in Cryptology - EUROCRYPT '92, Workshop on the Theory and Application of of Cryptographic Techniques, Balatonfüred, Hungary, May 24-28, 1992, Proceedings*, volume 658 of *Lecture Notes in Computer Science*, pages 307–323. Springer Verlag, 1992.

[7] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, May 13-17, 1990, Baltimore, Maryland, USA*, pages 503–513. ACM Press, 1990.

[8] Z. Beerliová-Trubíniová, M. Fitzi, M. Hirt, U. M. Maurer, and V. Zikas. MPC vs. SFE: Perfect security in a unified corruption model. In R. Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 231–250. Springer Verlag, 2008.

[9] Z. Beerliová-Trubíniová and M. Hirt. Efficient multi-party computation with dispute control. In S. Halevi and T. Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 305–328. Springer Verlag, 2006.

[10] Z. Beerliová-Trubíniová and M. Hirt. Simple and efficient perfectly-secure asynchronous MPC. In K. Kurosawa, editor, *Advances in Cryptology -*

*ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 376–392. Springer Verlag, 2007.

[11] Z. Beerliová-Trubíniová and M. Hirt. Perfectly-secure MPC with linear communication complexity. In R. Canetti, editor, *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 213–230. Springer Verlag, 2008.

[12] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous secure computation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, 1993*, pages 52–61. ACM Press, 1993.

[13] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 1–10. ACM Press, 1988.

[14] J. C. Benaloh and J. Leichter. Generalized secret sharing and monotone functions. In S. Goldwasser, editor, *Advances in Cryptology - CRYPTO '88, 8th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1988, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 27–35. Springer Verlag, 1988.

[15] M. BenOr, B. Kelmer, and T. Rabin. Asynchronous secure computations with optimal resilience. In *Proceedings of the Thirteenth Annual ACM Symposium on Principles of Distributed Computing, Los Angeles, California, USA, August 14-17*, pages 183–192. ACM Press, 1994.

[16] M. Blum, A. D. Santis, S. Micali, and G. Persiano. Noninteractive zero-knowledge. *SIAM Journal of Computing*, 20(6):1084–1118, 1991.

[17] G. Bracha. An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. In *Proceedings of the Third Annual ACM Symposium on Princiles of Distributed Computing, Vancouver, B. C., Canada, August 27-29, 1984*, pages 154 – 162. ACM Press, 1984.

[18] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In V. Atluri, editor, *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washingtion, DC, USA, November 18-22, 2002*, pages 88–97. ACM Press, 2002.

[19] R. Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, Weizmann Institute, Israel, 1995.

[20] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[21] R. Canetti, U. Feige, O. Goldreich, and M. Naor. Adaptively secure multiparty computation. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 639–648. ACM Press, 1996.

[22] R. Canetti and T. Rabin. Fast asynchronous Byzantine Agreement with optimal resilience. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 42–51. ACM Press, 1993.

[23] D. Chaum, C. Crpeau, and I. Damgård. Multiparty unconditionally secure protocols (extended abstract). In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA. ACM 1988*, pages 11–19. ACM Press, 1988.

[24] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 383–395. ACM Press, 1985.

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

[26] R. Cramer, I. Damgård, and S. Dziembowski. On the complexity of verifiable secret sharing and multiparty computation. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA. ACM*, pages 325–334. ACM Press, 2000.

[27] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, and T. Rabin. Efficient multiparty computations secure against an adaptive adversary. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 311–326. Springer Verlag, 1999.

[28] R. Cramer, I. Damgård, and S. Fehr. On the cost of reconstructing a secret, or VSS with optimal reconstruction phase. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 503–523. Springer Verlag, 2001.

[29] R. Cramer, I. Damgård, and U. M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium,*

*May 14-18, 2000, Proceeding*, volume 1807 of *Lecture Notes in Computer Science*, pages 316–334. Springer Verlag, 2000.

[30] I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen. Asynchronous multiparty computation: Theory and implementation. In S. Jarecki and G. Tsudik, editors, *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*, volume 5443 of *Lecture Notes in Computer Science*, pages 160–179. Springer Verlag, 2009.

[31] I. Damgård and J. B. Nielsen. Scalable and unconditionally secure multiparty computation. In A. Menezes, editor, *Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings*, volume 4622 of *Lecture Notes in Computer Science*, pages 572–590. Springer Verlag, 2007.

[32] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer Verlag, 1989.

[33] D. Dolev, C. Dwork, O. Waarts, and M. Yung. Perfectly secure message transmission. *JACM*, 40(1):17–47, 1993.

[34] C. Dwork. Strong verifiable secret sharing (extended abstract). In J. van Leeuwen and N. Santoro, editors, *Distributed Algorithms, 4th International Workshop, WDAG '90, Bari, Italy, September 24-26, 1990, Proceedings*, volume 486 of *Lecture Notes in Computer Science*, pages 213–227. Springer Verlag, 1990.

[35] C. Dwork. On verification in secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 114–128. Springer Verlag, 1991.

[36] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, 27-29 October 1987*, pages 427–437. IEEE Computer Society, 1987.

[37] P. Feldman and S. Micali. An optimal algorithm for synchronous Byzantine Agreemet. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 639–648. ACM Press, 1988.

[38] M. Fitzi, J. Garay, S. Gollakota, C. Pandu Rangan, and K. Srinathan. Round-optimal and efficient verifiable secret sharing. In S. Halevi and

T. Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 329–342. Springer Verlag, 2006.

[39] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. The round complexity of verifiable secret sharing and secure multicast. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece. ACM*, pages 580–589. ACM Press, 2001.

[40] R. Gennaro and S. Micali. Verifiable secret sharing as secure computation. In L. C. Guillou and J. Quisquater, editors, *Advances in Cryptology - EUROCRYPT '95, International Conference on the Theory and Application of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995, Proceeding*, volume 921 of *Lecture Notes in Computer Science*, pages 168–182. Springer Verlag, 1995.

[41] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fact-track multiparty computations with applications to threshold cryptography. In *PODC '98. Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing, June 28 - July 2, 1998, Puerto Vallarta, Mexico*, pages 101–111. ACM Press, 1998.

[42] O. Golderich, S. Micali, and A. Wigderson. How to play a mental game– a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 218–229. ACM Press, 1987.

[43] O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems. *Journal of ACM*, 38(3):691–729, 1991.

[44] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In D. Coppersmith, editor, *Advances in Cryptology - CRYPTO '95, 15th Annual International Cryptology Conference, Santa Barbara, California, USA, August 27-31, 1995, Proceedings*, volume 963 of *Lecture Notes in Computer Science*, pages 339–352. Springer Verlag, 1995.

[45] M. Hirt, U. Maurer, and B. Przydatek. Efficient secure multiparty computation. In T. Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*, pages 143–161. Springer Verlag, 2000.

[46] M. Hirt and U. M. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, 2000.

[47] M. Hirt and U. M. Maurer. Robustness for free in unconditional multi-party computation. In J. Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 101–118. Springer Verlag, 2001.

[48] M. Hirt, U. M. Maurer, and V. Zikas. MPC vs. SFE : Unconditional and computational security. In J. Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, volume 5350 of *Lecture Notes in Computer Science*, pages 1–18. Springer Verlag, 2008.

[49] M. Hirt and J. B. Nielsen. Upper bounds on the communication complexity of optimally resilient cryptographic multiparty computation. In B. K. Roy, editor, *Advances in Cryptology - ASIACRYPT 2005, 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005, Proceedings*, volume 3788 of *Lecture Notes in Computer Science*, pages 79–99. Springer Verlag, 2005.

[50] M. Hirt and J. B. Nielsen. Robust multiparty computation with linear communication complexity. In C. Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 463–482. Springer Verlag, 2006.

[51] M. Hirt, J. B Nielsen, and B. Przydatek. Cryptographic asynchronous multi-party computation with optimal resilience (extended abstract). In R. Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 322–340. Springer Verlag, 2005.

[52] M. Hirt, J. B Nielsen, and B. Przydatek. Asynchronous multi-party computation with quadratic communication. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 473–485. Springer Verlag, 2008.

[53] Y. Ishai and E. Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California*, pages 294–304. IEEE Computer Society, 2000.

[54] J. Katz, C. Koo, and R. Kumaresan. Improving the round complexity of VSS in point-to-point networks. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and I. Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 499–510. Springer Verlag, 2008.

[55] J. Katz and C. Y. Koo. On expected constant-round protocols for Byzantine Agreement. In C. Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, Lecture Notes in Computer Science, pages 445–462. Springer Verlag, 2006.

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

[57] U. M. Maurer. Secure multi-party computation made simple. *Discrete Applied Mathematics*, 154(2):370–381, 2006.

[58] W. Ogata and K. Kurosawa. Optimum secret sharing scheme secure against cheating. In U. M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 200–211. Springer Verlag, 1996.

[59] A. Patra, A. Choudhary, T. Rabin, and C. Pandu Rangan. The round complexity of verifiable secret sharing revisited. In S. Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 487–504. Springer Verlag, 2009.

[60] A. Patra, A. Choudhary, and C. Pandu Rangan. Round efficient unconditionally secure multiparty computation protocol. In D. R. Chowdhury, V. Rijmen, and A. Das, editors, *Progress in Cryptology - INDOCRYPT 2008, 9th International Conference on Cryptology in India, Kharagpur, India, December 14-17, 2008. Proceedings*, volume 5365 of *Lecture Notes in Computer Science*, pages 185–199. Springer Verlag, 2008.

[61] A. Patra, A. Choudhary, and C. Pandu Rangan. Efficient asynchronous Byzantine Agreement with optimal resilience. Submitted to Distributed Computing Journal. Also available as Cryptology ePrint Archive: Report 2008/424. A preliminary version of this article appeared in Proceedings of the 28th Annual ACM Symposium on Principles of Distributed Computing, PODC 2009, Calgary, Alberta, Canada, August 10-12, pages 92-101, 2009.

[62] A. Patra, A. Choudhary, and C. Pandu Rangan. Efficient statistical asynchronous verifiable secret sharing with optimal resilience. Accepted for publication in ICITS, 2009.

[63] A. Patra, A. Choudhary, and C. Pandu Rangan. Unconditionally secure asynchronous multiparty computation with quadratic communication per multiplication gate. Cryptology ePrint Archive, Report 2009/087, 2009.

[64] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer Verlag, 1991.

[65] B. Prabhu, K. Srinathan, and C. Pandu Rangan. Trading players for efficiency in unconditional multiparty computation. In S. Cimato, C. Galdi, and G. Persiano, editors, *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, volume 2576 of *Lecture Notes in Computer Science*, pages 342–353. Springer Verlag, 2002.

[66] T. Rabin. Robust sharing of secrets when the dealer is honest or cheating. *Journal of ACM*, 41(6):1089–1109, 1994.

[67] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 73–85. ACM Press, 1989.

[68] K. Srinathan, A. Narayanan, and C. Pandu Rangan. Optimal perfectly secure message transmission. In M. K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 545–561. Springer Verlag, 2004.

[69] K. Srinathan and C. Pandu Rangan. Efficient asynchronous secure multiparty distributed computation. In B. K. Roy and E. Okamoto, editors, *Progress in Cryptology - INDOCRYPT 2000, First International Conference in Cryptology in India, Calcutta, India, December 10-13, 2000, Proceedings*, volume 1977 of *Lecture Notes in Computer Science*, pages 117–129. Springer Verlag, 2000.

[70] M. Stadler. Publicly verifiable secret sharing. In U. M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 190–199. Springer Verlag, 1996.

[71] A. C. Yao. Protocols for secure computations. In *Proceedings of 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, 3-5 November 1982*, pages 160–164. IEEE Computer Society, 1982.

[72] V. Zikas, S. Hauser, and U. M. Maurer. Realistic failures in secure multi-party computation. In O. Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 274–293. Springer Verlag, 2009.