Efficient Statistical Asynchronous Verifiable Secret Sharing and Multiparty Computation with Optimal Resilience

Arpita Patra [*] **Ashish Choudhary** [†] **C. Pandu Rangan** [‡]
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai India 600036
Email:{ `arpitapatra_10, partho_31` }@yahoo.co.in, prangan55@yahoo.com

Abstract

In this paper, we present a new statistical *asynchronous verifiable secret sharing* (AVSS) protocol with *optimal resilience*; i.e. with $n = 3t + 1$, where n is the total number of participating parties and t is the maximum number of parties that can be under the control of a *computationally unbounded active adversary* $\mathcal{A}_t$. Our protocol privately communicates $\mathcal{O}((\ell n^3 + n^4\kappa)\kappa)$ bits and $\mathsf{A\text{-}casts}$ $\mathcal{O}(n^3\log(n))$ bits to simultaneously share $\ell \geq 1$ elements from a finite field $\mathbb{F}$, where κ is the error parameter. Here $\mathsf{A\text{-}cast}$ is an asynchronous broadcast primitive, which allows a party to send some information *identically* to all other parties.

There are only two known statistical AVSS protocols with $n = 3t+1$ (i.e., with optimal resilience) reported in [11] and [31]. The AVSS protocol of [11] requires a private communication of $\mathcal{O}(n^9\kappa^4)$ bits and $\mathsf{A\text{-}cast}$ of $\mathcal{O}(n^9\kappa^2\log(n))$ bits to share a *single* element from $\mathbb{F}$. Thus our AVSS protocol shows a significant improvement in communication complexity over the AVSS of [11]. The AVSS protocol of [31] requires a private communication of $\mathcal{O}((\ell n^3 + n^4)\kappa)$ bits and $\mathsf{A\text{-}cast}$ of $\mathcal{O}((\ell n^3 + n^4)\kappa)$ 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 [31] due to the following reasons:

1. The $\mathsf{A\text{-}cast}$ communication of our AVSS is *independent* of the number of secrets i.e. ℓ;
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 is an essential building block of *asynchronous multiparty computation* (AMPC). Using our ACSS scheme, we design a statistical AMPC with *optimal resilience*; i.e., with $n = 3t + 1$, that privately communicates $\mathcal{O}(n^5\kappa)$ bits *per multiplication gate*. This significantly improves the only known statistical AMPC of [8] with $n = 3t + 1$, which privately communicates $\Omega(n^{11}\kappa^4)$ bits and $\mathsf{A\text{-}cast}$ $\Omega(n^{11}\kappa^2\log(n))$ bits per multiplication gate. Both our ACSS and AVSS employs several new techniques, which are of independent interest.

Keywords: Asynchronous Networks, AVSS, Optimal Resilience, AMPC.

1 Introduction

A Verifiable Secret Sharing (VSS) [13] 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. Any VSS scheme consists of a pair of protocols (Sh, Rec). Protocol Sh allows a special party in $\mathcal{P}$, called *dealer* (denoted as D), to share a secret $s \in \mathbb{F}$ (an element from a finite field $\mathbb{F}$) among all the parties in a way that allow for a unique reconstruction of s by every body using protocol Rec. Moreover, if D is *honest*, then the secrecy of s from $\mathcal{A}_t$ should be preserved till the end of Sh.

[*]Financial Support from Microsoft Research India Acknowledged

[†]Financial Support from Infosys Technology India Acknowledged

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

VSS is one of the fundamental building blocks for many secure distributed computing tasks, such as multiparty computation (MPC) [7, 35, 2, 15, 25, 3, 4, 5], Byzantine Agreement (BA) [20, 11, 29, 1, 31], etc. Over the past three decades, the problem has been studied in different settings and computational models (see [24, 7, 12, 19, 35, 15, 16, 23, 21, 28]). The VSS problem has been studied extensively over *synchronous* networks, which assumes that there is a global clock and the delay of any message in the network is bounded. However, VSS in asynchronous network has got comparatively less attention, due to its inherent hardness. As asynchronous networks model real life networks like Internet more precisely, it is important to investigate fundamental problem like VSS in asynchronous network.

1.1 Definitions

Asynchronous Networks: In an asynchronous network, the communication channels 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 not access the messages communicated between honest parties. Here 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 it is impossible to consider the values sent by all uncorrupted parties and hence the values of up to t (potentially honest) parties may get ignored, as waiting for them could turn out to be endless. Due to this the protocols in asynchronous network are generally involved in nature and require new set of primitives. For an excellent introduction to asynchronous protocols, see [10].

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

Definition 1 (Statistical Asynchronous Weak Secret Sharing (AWSS) [31]) *Let (Sh, Rec) be a pair of protocols in which a dealer $D \in \mathcal{P}$ shares a secret $s \in \mathbb{F}$ using Sh. We say that (Sh, Rec) is a t-resilient statistically secure AWSS scheme if all the following hold:*

- **Termination***: With probability at least $1 - 2^{-\Omega(\kappa)}$, all the following holds:*
  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 the honest parties have terminated Sh and if all the honest parties invoke protocol Rec, then each honest party will eventually terminate Rec.*
- **Correctness***: With probability at least $1 - 2^{-\Omega(\kappa)}$, all the following holds:*
  1. *If D is honest then each honest party upon completing Rec outputs s.*
  2. *If D is corrupted and some honest party has terminated Sh, then there exists a fixed $\overline{s} \in \mathbb{F} \cup \{NULL\}$, such that each honest party upon completing Rec, will output either $\overline{s}$ or $NULL$.*
- **Secrecy***: If D is honest and no honest party has begun Rec, then $\mathcal{A}_t$ has no information about s.*

Definition 2 (Statistical Asynchronous Verifiable Secret Sharing (AVSS) [6, 10]) *It is same as AWSS except that **Correctness 2** property is strengthened as follows:*

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

Definition 3 (t-sharing [3, 5]) *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 of a degree-t polynomial. For example, see [7, 28]. On the other hand, there do exists VSS scheme which do not generate t-sharing of secret. They only ensure that a unique secret is shared (committed) which will be uniquely reconstructed during reconstruction phase. Such schemes are presented in [21, 30]. 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 requires the following completeness property to hold at the end of* Sh *with probability at least* $1 - 2^{-\Omega(\kappa)}$*:*

- **Completeness***: at the end of* Sh*, there exists a random degree-t polynomial* $f(x)$ *over* $\mathbb{F}$*, with* $f(0) = \overline{s}$ *such that each (honest) party* $P_i \in \mathcal{P}$ *holds his share* $s_i = f(i)$ *of secret* $\overline{s}$*. Moreover, if* D *is honest, then* $\overline{s} = s$*.*

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 protocols with only private reconstruction. However, the public reconstruction for these protocols can be obtained by doing slight modification.*

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

Definition 5 (A-cast [11, 10]) *It is an asynchronous broadcast primitive, which allows a special party in* $\mathcal{P}$ *(called sender) to identically distribute a message among all parties in* $\mathcal{P}$*. It was introduced and elegantly implemented by Bracha [9] with* $n = 3t+1$*. Let* Π *be an asynchronous protocol initiated by a special party (called the sender), having input* m *(the message to be broadcast). We say that* Π *is a* t-resilient A-cast *protocol if the following hold, for every possible* $\mathcal{A}_t$*:*

- **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* $m^* = m$*.*

The A-cast *protocol of [9] requires a private communication of* $\mathcal{O}(n^2 b)$ *bits to* A-cast *a* b *bit message.*

1.2 Existing Results for Statistical AVSS with Optimal Resilience

Statistical AVSS tolerating $\mathcal{A}_t$ is possible iff $n \geq 3t+1$ [11]. Any statistical AVSS with $n = 3t+1$ is said to have *optimal resilience*. The only known statistical AVSS with optimal resilience are due to [11] and [31], which are used in designing *Asynchronous Byzantine Agreement* (ABA) schemes. In the following, we summarize these two AVSS schemes.

1. The authors of [11] 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 [11] 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 $\mathcal{O}(n^9\kappa^4)$ bits and A-cast $\mathcal{O}(n^9\kappa^2\log(n))$ bits to share a *single* element from $\mathbb{F}$. However, the AVSS scheme of [11] fails to generate t-sharing of the secret. That is, the AVSS scheme of [11] is not an ACSS scheme and hence is not suitable for AMPC. More detailed discussion on this point will appear in Section 9.

2. Pictorially, the authors in [31] used the following simpler route to design their AVSS scheme: *ICP → AWSS → AVSS*. Moreover, the authors in [31] significantly improved each of the underlying building blocks, namely ICP and AWSS, by employing new design approaches. The AVSS protocol of [31] requires a private communication of $\mathcal{O}((\ell n^3 + n^4)\kappa)$ bits and A-cast of $\mathcal{O}((\ell n^3 + n^4)\kappa)$ bits to share $\ell \geq 1$ elements. However, the AVSS scheme of [31] has the following shortcomings:
   (a) The AVSS scheme of [31] is not an ACSS scheme and hence is not suitable for AMPC. More detailed discussion on this point will appear in Section 9.
   (b) In AVSS of [31], a *corrupted* D may choose secrets from $\mathbb{F} \cup \{NULL\}$ instead of only $\mathbb{F}$.

1.3 Our Contribution

We present a new statistical AVSS with optimal resilience by following the simple route of [31]. In the following table, we compare the communication complexity of our AVSS with the AVSS of [11, 31]. The table also shows the communication complexity (CC) after simulating A-cast using the protocol of [9].

Ref.	CC in bits	CC in bits using A-cast of [9]	# Secrets
[11]	Private– $\mathcal{O}(n^9\kappa^4)$ A-cast– $\mathcal{O}(n^9\kappa^2\log(n))$	private– $\mathcal{O}(n^9\kappa^4 + n^{11}\kappa^2\log n)$	1
[31]	Private– $\mathcal{O}((\ell n^3 + n^4)\kappa)$ A-cast– $\mathcal{O}((\ell n^3 + n^4)\kappa)$	private– $\mathcal{O}((\ell n^5 + n^6)\kappa)$	ℓ
This Article	Private– $\mathcal{O}((\ell n^3 + n^4\kappa)\kappa)$ A-cast– $\mathcal{O}(n^3\log(n))$	private– $\mathcal{O}((\ell n^3 + n^4\kappa)\kappa + n^5\log n)$	ℓ

As shown in the table, our AVSS attains significantly better communication complexity than the AVSS of [11] and [31] for any value of ℓ. As mentioned in the previous section, the AVSS of [31] has a *weaker* property than the AVSS of this article and [11]: *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 $\mathbb{F}$ [8]. 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) [8]. Though there exists CSS in synchronous settings, our ACSS scheme is first of its kind in asynchronous settings with $n = 3t + 1$. In fact, using our ACSS, we can design an efficient statistical AMPC with *optimal resilience*; i.e., with $n = 3t + 1$, which privately communicates $\mathcal{O}(n^5\kappa)$ bits *per multiplication gate*. This will be a significant improvement over the *only known* statistical AMPC of [8] with $n = 3t+1$, which privately communicates $\Omega(n^{11}\kappa^4)$ bits and A-cast $\Omega(n^{11}\kappa^2\log(n))$ bits per multiplication gate.

In order to design AVSS, we first propose a new ICP which significantly improves the communication complexity of the ICP of [31]. Using our ICP, we design an AWSS which is inspired by AWSS of [31]. Finally our new AWSS is used in designing our new AVSS protocol. The design approach of our AVSS and ACSS are novel and first of their kind.

1.4 Organization of the Paper

For ease of presentation, we divide the paper into two parts. The first part, consisting of Section 2 upto Section 8 deals with the design of our AVSS and ACSS scheme. The second part consisting of Section 9 and subsequent sections deals with the design of our AMPC protocol.

2 Information Checking Protocol and IC Signature

Information Checking Protocol (ICP) is a tool for authenticating messages in the presence of computationally unbounded $\mathcal{A}_t$. The notion of ICP was first introduced by Rabin et.al [35, 34]. The ICP of Rabin et. al. was also used as a tool by Canetti et. al. [11] for designing ABA with optimal resilience (i.e $n = 3t + 1$). Here we present an ICP, called A-ICP(D, INT, P, S) in asynchronous settings.

As in [31], A-ICP is executed among three entities: the dealer $D \in \mathcal{P}$, an intermediary $INT \in \mathcal{P}$ and entire set $\mathcal{P}$ acting as verifiers. The dealer D hands a secret s to INT. At a later stage, INT is has to hand over s to the verifiers in $\mathcal{P}$ and convince them that s is indeed the value which INT received from D. We may also run A-ICP to *concurrently* work on *multiple* secrets, denoted by S containing $\ell \geq 1$ secrets. So, instead of repeating multiple instances of ICP dealing with single secret, we can run a single instance of our A-ICP dealing with multiple secrets *concurrently*, leading to significant reduction in communication complexity. We use A-ICP in our AWSS scheme, where it is required to execute instances of A-ICP dealing with multiple secrets concurrently.

For ℓ secrets, the A-ICP of [31] incurs a private communication of $\mathcal{O}((\ell + n)\kappa)$ bits and A-cast of $\mathcal{O}((\ell + n)\kappa)$ bits. On the other hand, our A-ICP incurs *only* private communication of $\mathcal{O}((\ell + n\kappa)\kappa)$ bits (and *no* A-cast). As in [11, 31], our A-ICP is also structured into sequence of following three phases:

1. **Generation Phase**: This is initiated D. Here D hands over the secret S, containing ℓ elements from $\mathbb{F}$ along with some *authentication information* to *intermediary* INT and some *verification information* to individual *verifiers* in $\mathcal{P}$.

2. **Verification Phase**: is carried out by INT and the set of verifiers $\mathcal{P}$. Here INT decides whether to continue or abort the protocol depending upon the prediction whether in **Revelation Phase**, the secret S held by INT will be (eventually) accepted/will be considered as valid by the honest verifier(s) in $\mathcal{P}$. INT achieves this by setting a boolean variable $\mathsf{Ver} = 0/1$, where $\mathsf{Ver} = 0$ (resp. 1) implies abortion (resp. continuation) of the protocol. If $\mathsf{Ver} = 1$, then the *authentication information*, along with S, held by INT at the end of **Verification Phase** is called D's IC *signature* on S. We denote it by $ICSig(D, INT, \mathcal{P}, S)$.

3. **Revelation Phase**: is carried out by INT 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 $\mathcal{P}$ where all the verifiers can publicly verify whether INT indeed received IC signature on S from D.

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

Protocol A-ICP satisfies the following properties (assuming *Public Revelation* in **Revelation Phase**):

1. If D and INT are honest, then S will be accepted in **Revelation phase** by each honest verifier.

2. If INT is honest and Ver =1, then S held by INT will be accepted in **Revelation phase** by each honest verifier, except with probability $2^{-\Omega(\kappa)}$.

3. If D is honest, then during **Revelation phase**, with probability at least $1 - 2^{-\Omega(\kappa)}$, every $S' \neq S$ produced by a corrupted INT will be not be accepted by any honest verifier.

4. If D and INT are honest and INT has not started **Revelation phase**, then S will be information theoretically secure.

For protocol A-ICP with P_α*-private-revelation* in **Revelation Phase**, the above properties are modified by replacing "every/any honest verifier" with "honest P_α".

Notice that unlike other asynchronous primitives (e.g. AWSS, AVSS), we do not concentrate on defining termination property for A-ICP. The reason is that A-ICP will never be executed as a stand alone application. Rather, A-ICP will act as a tool to design AWSS, which has its own termination properties. This is in line with [11, 31], where ICP is defined without termination property and is used as a tool in AWSS/AVSS protocol. In the sequel, we present protocol A-ICP. As in our of AWSS we require only P_α-private-revelation of $ICSig(D, INT, \mathcal{P}, S)$, we present only that (though we have an implementation for public revelation of $ICSig(D, INT, \mathcal{P}, S)$).

Protocol A-ICP$(D, INT, \mathcal{P}, S)$

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

1. The dealer D, on having secret $S = (s^1, \ldots, s^\ell)$, selects a random $\ell + t\kappa$ degree polynomial $f(x)$ whose lower order ℓ coefficients are in S. D also picks $n\kappa$ random non-zero elements from $\mathbb{F}$, denoted by $\alpha_1^i, \ldots, \alpha_\kappa^i$, for $i = 1, \ldots, n$.
2. For $i = 1, \ldots, n$, D 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, where $a_j^i = f(\alpha_j^i)$, for $j = 1, \ldots, \kappa$.

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

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. For every verifier P_i from which INT has received values, INT checks whether for *every* $j \in I^i$, $f(\alpha_j^i) \overset{?}{=} a_j^i$.
3. (a) If for at least $2t + 1$ verifiers, the above condition is satisfied, then INT sets Ver $= 1$. If Ver $= 1$, then the information held by INT is denoted by $ICSig(D, INT, \mathcal{P}, S)$.
   (b) If for at least $t + 1$ verifiers, the above condition is not satisfied, then INT sets Ver $= 0$.

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

1. To party P_α, INT sends $f(x)$.
2. To party P_α, every verifier P_i sends the index set $\overline{I^i}$ and all z_j^i such that $j \in \overline{I^i}$.
3. Upon receiving the values from verifier P_i, party P_α checks whether for *some* $j \in \overline{I^i}$, $f(\alpha_j^i) \overset{?}{=} a_j^i$.
   (a) If for at least $t+1$ verifiers the condition is satisfied, then P_α sets Reveal$_\alpha = S$, where S is lower order ℓ coefficients of $f(x)$. In this case, we say that INT is 'successful' in producing $ICSig(D, INT, \mathcal{P}, S)$ to P_α.
   (b) If for at least $2t+1$ verifiers the above condition is not satisfied, then P_α sets Reveal$_\alpha = NULL$. In this case, we say that INT 'fails' in producing $ICSig(D, INT, \mathcal{P}, S)$ to P_α.

We now prove the properties of protocol A-ICP.

Lemma 1 *If D, INT and P_α are honest, then S will be accepted by P_α.*

Proof: If D is honest then he will honestly deliver $f(x)$ to INT and its value at κ points to individual verifiers. So eventually, the condition stated in step 3(a) of **Verification Phase** will be satisfied and hence INT, who is honest in this case will set Ver $= 1$. Let $\mathcal{V}$ be the set of these $2t + 1$ verifiers. The set $\mathcal{V}$ will contain at least $t + 1$ honest verifiers. Let $\mathcal{H}$ be the set of honest verifiers in $\mathcal{V}$. Now it is easy to see that the condition stated in step 3(a) will be eventually satisfied, corresponding to the verifiers in $\mathcal{H}$. Hence P_α, who is honest in this case will eventually output Reveal$_\alpha = S$ at the end of **Revelation phase**. □

Lemma 2 *If INT is honest and Ver =1, then S held by INT will be accepted in Reveal-Private by honest P_α, except with probability $2^{-\Omega(\kappa)}$.*

Proof: We have to consider the case when D *is corrupted.* Since INT is honest and Ver $= 1$ at the end of **Verification phase**, INT has ensured that for at least $2t + 1$ verifiers the condition specified in step 2 of **Verification phase** has been satisfied. Let H be the set of *honest* verifiers among these $2t+1$ verifiers. Note that $|H| \geq t+1$. To prove the lemma, we prove that corresponding to each verifier in H, the condition stated in step 3 of Reveal-Private will be satisfied with very high probability. Note that corresponding to a verifier P_i in H, the condition stated in step 3 of Reveal-Private 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 $\overline{I^i}$ such that $\overline{I_i}$ contains only inconsistent tuples (z_j^i's). Thus corresponding to a verifier $P_i \in H$, the probability that the condition stated in step 3 of Reveal-Private fails is same as the probability of P_i selecting all consistent (inconsistent) tuples in I^i ($\overline{I^i}$), which is $\frac{1}{\binom{\kappa}{\kappa/2}} \approx 2^{-\Omega(\kappa)}$. □

Lemma 3 *If D is honest, then in Reveal-Private, with probability $1 - 2^{-\Omega(\kappa)}$, every $S' \neq S$ produced by a corrupted INT will be rejected by honest P_α.*

Proof: It is easy to see that $S' \neq S$ produced by a *corrupted* INT will be accepted by an *honest* P_α, if the condition stated in step 3 of Reveal-Private 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 an honest verifier P_i if 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 $2^{-\Omega(\kappa)}$. □

Lemma 4 *If D and INT are honest and INT has not started* Reveal-Private*, then S is information theoretically secure from $\mathcal{A}_t$.*

Proof: The adversary will know $t\kappa$ points on $f(x)$. Since $f(x)$ is a polynomial of degree $\ell + t\kappa$, S will remain information theoretically secure. □

Lemma 5 *Protocol* Gen*,* Ver *and* Reveal-Private *privately communicate $\mathcal{O}((\ell + n\kappa)\kappa)$ bits each.*

Notation 1 (Notation for Using A-ICP) *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" to mean that P_i acting as dealer D executes $\mathsf{Gen}(P_i, P_j, \mathcal{P}, S)$;*
2. *"P_i receives $ICSig(P_j, P_i, \mathcal{P}, S)$ from P_j" to mean that P_i as INT has successfully completed $\mathsf{Ver}(P_j, P_i, \mathcal{P}, S)$ with $\mathsf{Ver} = 1$ with the help of the verifiers in $\mathcal{P}$;*
3. *"P_i reveals $ICSig(P_j, P_i, \mathcal{P}, S)$ to P_α" to mean P_i as INT executes $\mathsf{Reveal\text{-}Private}(P_j, P_i, \mathcal{P}, S, P_\alpha)$ along with the participation of the verifiers in $\mathcal{P}$;*
4. *"P_α completes revelation $ICSig(P_j, P_i, \mathcal{P}, S)$ with $\mathsf{Reveal}_\alpha = S$ " to mean that P_α has successfully completed $\mathsf{Reveal\text{-}Private}(P_j, P_i, \mathcal{P}, S, P_\alpha)$ with $\mathsf{Reveal}_\alpha = S$.*

3 AWSS Scheme for Sharing a Single Secret

We now present an AWSS scheme called AWSS-SS with $n = 3t + 1$. AWSS-SS consists of two protocols AWSS-SS-Share and AWSS-SS-Rec-Private. While AWSS-SS-Share allows D to share *a single secret s* among $\mathcal{P}$, AWSS-SS-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_α-weak-private-reconstruction*. In AWSS-SS-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-SS-Share is inspired by the sharing phase of AWSS-Single-Secret given in [31]. However, instead of using the A-ICP of [31], we use our A-ICP in AWSS-SS-Share, which leads to better communication complexity. The high level idea of AWSS-SS-Share is as follows: we follow the general idea of [7, 15, 23, 21] in synchronous settings for sharing the secret s with a degree-t symmetric bivariate polynomial $F(x, y)$, where each party P_i gets the univariate polynomial $f_i(x) = F(x, i)$. D first hands over n points on $f_i(x)$ to P_i, with his IC signature on these values. Then D, in conjunction with all other parties, perform a sequences of communications and computations. As a result of this, at the end of the sharing phase, every honest party agrees on a set of $2t + 1$ parties, called $WCORE$, such that every party $P_j \in WCORE$ is *confirmed* by a set of $2t+1$ parties, called as $OKSetP_j$. A party $P_k \in OKSetP_j$ provides the *confirmation* to P_j, only when it possesses proper IC signature of D on $f_k(j)$ (j^{th} point on polynomial $f_k(x)$, which P_k is entitled to receive from D) as well as IC signature of P_j on the point $f_j(k)$ (k^{th} point on polynomial $f_j(x)$, which P_j is entitled to receive from D), such that $f_j(k) = f_k(j)$ holds (which should ideally hold due to the selection and distribution of symmetric bivariate polynomial). In some sense, we may view these checkings as every $P_j \in WCORE$ is attempting to commit to his received (from D) polynomial $f_j(x)$ among the parties in $OKSetP_j$ (by giving his *IC Signature* on one point of $f_j(x)$ to each party) and the parties in $OKSetP_j$ 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 refer this commitment as P_j's *IC-Commitment* on $f_j(x)$ and hence on $f_j(0)$.

Achieving the agreement (among the honest parties) on $WCORE$ and corresponding $OKSet$s is a bit tricky in asynchronous network. Even though the *confirmations* are A-casted by parties, parties may end up with different versions of $WCORE$ and $OKSet$'s while attempting to generate them locally, due to the asynchronous nature of the network. We solve this problem by asking D to first construct $WCORE$ and $OKSet$s after receiving *confirmations* and ask D to A-cast the same. After receiving $WCORE$ and

$OKSet$s from the A-cast of D, individual parties ensure the validity of (verifies) these sets by receiving the same *confirmations* from the parties in the received $OKSet$s. A similar approach was used in [1].

Protocol AWSS-SS($D, \mathcal{P}, s$)

AWSS-SS-Share($D, \mathcal{P}, s$)

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

1. Select a random, symmetric bivariate polynomial $F(x,y)$ over $\mathbb{F}$ 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 for each $j = 1, \ldots, n$.

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

1. Wait to receive $ICSig(D, P_i, \mathcal{P}, f_i(j))$ 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 for all $j = 1, \ldots, n$.
3. If $ICSig(P_j, P_i, \mathcal{P}, f_j(i))$ is received from P_j 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 $OKP_j = \{P_i | D$ receives $\texttt{OK}(P_i, P_j)$ from the A-cast of $P_i\}$. When $|OKP_j| = 2t+1$, then P_j's *IC-Commitment* on $f_j(0)$ 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 $\texttt{OK}$s, accept the $WCORE$ and OKP_j's received from D and terminate **AWSS-SS-Share**.

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

SIGNATURE REVELATION: CODE FOR P_i

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_α.

LOCAL COMPUTATION: CODE FOR P_α

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-SS-Rec-Private. Else set $\overline{s} = NULL$ and terminate AWSS-SS-Rec-Private.

In AWSS-SS-Rec-Private, the parties in $WCORE$ and corresponding $OKSet$'s enables P_α to privately reconstruct the secret. Precisely, P_j's *IC-Commitment* on $f_j(x)$ and hence on $f_j(0)$ is revealed to P_α by reconstructing it with the help of the parties in $OKSetP_j$, for every $P_j \in WCORE$. Then $f_j(0)$'s are used to interpolate the degree-t univariate polynomial (if possible) $f_0(x) = F(x,0)$ and hence the secret $s = f_0(0) = F(0,0)$ that is committed by D during sharing phase. Since $f_j(x)$ is a degree-t polynomial, any $t+1$ points on it are enough to interpolate $f_j(x)$. The points on $f_j(x)$ are obtained by requesting each party P_k in $OKSetP_j$ to reveal IC signature of D on $f_k(j)$ and IC signature of P_j on $f_j(k)$ such that $f_j(k) = f_k(j)$ holds. Asking $P_k \in OKSetP_j$ to reveal D's signature ensures that when 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. This helps our AWSS protocol to satisfy **Correctness 1** property of AWSS. Now asking P_k in $OKSetP_j$ 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 the sharing phase. This ensure **correctness 2** property. Summing up, when at least one of D and P_j is honest, P_j's *IC-Commitment* on $f_j(x)$ and hence $f_j(0)$ is revealed properly. But when both D and P_j are corrupted, P_j's *IC-Commitment* on $f_j(0)$ can be revealed as any $\overline{f_j(0)} \neq f_j(0)$. It is the later property that makes our protocol to qualify as an AWSS protocol instead of AVSS.

Remark 2 (D's Commitment in AWSS-SS-Share) *We say that D is committed to a secret $s \in \mathbb{F}$ in AWSS-SS-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. Otherwise, we say that D is committed to $NULL$. An honest D is always committed to $s \in \mathbb{F}$, as in this case $f(x) = f_0(x) = F(x, 0)$ and $f(i) = f_0(i) = f_i(0) = F(0, i)$ where $F(x, y)$ is the symmetric degree-t bivariate polynomial chosen by honest D. But AWSS-SS-Share can not ensure that corrupted D also commits $s \in \mathbb{F}$.*

Notation 2 (Notation for Using AWSS-SS-Share) *In subsequent sections, we will invoke AWSS-SS-Share as AWSS-SS-Share($D, \mathcal{P}, f(x)$) to mean that D commits to $f(x)$ in AWSS-SS-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-SS-Rec-Private will be invoked as AWSS-SS-Rec-Private($D, \mathcal{P}, f(x), P_\alpha$) for P_α-weak-private-reconstruction of $f(x)$.* □

The proof of the properties of AWSS-SS follows using similar arguments as in AWSS-Single-Secret [31]. However, for the sake of completeness we prove them here.

Lemma 6 *AWSS-SS satisfies termination property of Definition 1.*

PROOF:

- **Termination 1:** When D is honest, then every honest P_j will eventually complete its *IC-Commitment* on $f_j(0)$ with at least $2t + 1$ honest parties in OKP_j. Hence, D will eventually include $2t + 1$ parties in $WCORE$ (of which at least $t + 1$ are honest) and A-cast the same. Now by the property of A-cast, each honest party will eventually listen $WCORE$ from the A-cast of D. Finally, since honest D had included P_j in $WCORE$ after listening the OK signals from the parties in OKP_j's, each honest party will also listen them and will terminate AWSS-SS-Share.
- **Termination 2:** If an honest P_i has terminated AWSS-SS-Share, then he must have listened $WCORE$ and OKP_j's from the A-cast of D and verified their validity. By properties of A-cast, each honest party will also listen the same and will eventually terminate AWSS-SS-Share.
- **Termination 3:** By Lemma 2, if P_i (acting as INT) is honest and Ver $= 1$ at the end of **Verification Phase**, then IC signature produced by P_i during Reveal-Private will be accepted by an honest P_α, except with probability $2^{-\Omega(\kappa)}$. Since for every $P_j \in WCORE$, $|OKP_j| = 2t + 1$, there are at least $t + 1$ honest parties in OKP_j who will be present in $ValidP_j$ with very high probability. Hence for every $P_j \in WCORE$, P_j's *IC-Commitment* will be reconstructed. Thus with very high probability, an *honest* P_α will terminate AWSS-SS-Rec-Private after executing remaining steps of [LOCAL COMPUTATION]. □

Lemma 7 *AWSS-SS satisfies secrecy property of Definition 1.*

PROOF: Follows from the secrecy of A-ICP and properties of symmetric bivariate polynomial of degree-t in x and y. □

Lemma 8 *AWSS-SS satisfies correctness property of Definition 1.*

PROOF:

- **Correctness 1:** Here we have to consider the case when D is *honest*. We first prove that if D is honest, then except with probability $2^{-\Omega(\kappa)}$, for each $P_j \in WCORE$, the value $\overline{f_j(0)}$ which is reconstructed by P_α, is same as $f_j(0)$ that was selected by D. From the property of A-ICP, for an *honest* $P_j \in WCORE$, a corrupted $P_k \in OKP_j$ can produce P_j's valid signature on incorrect $\overline{f_j(k)} \neq f_j(k)$ with negligible probability (see Lemma 3). Hence with very high probability $\overline{f_j(k)}$ is same as $f_j(k)$ for all $P_k \in ValidP_j$. Thus the polynomial $\overline{f_j(x)}$ reconstructed by P_α corresponding to an honest P_j in $WCORE$ is same as $f_j(x)$ that was selected by honest D. On the other hand, for a *corrupted* $P_j \in WCORE$, a corrupted $P_k \in OKP_j$ can produce P_j's valid signature on any $\overline{f_j(k)} \neq f_j(k)$ but P_k will fail to produce honest D's signature on $\overline{f_k(j)} = \overline{f_j(k)}$, with very high probability. Hence P_k will not be included in $ValidP_j$. Thus again the reconstructed polynomial $\overline{f_j(x)}$ corresponding to a corrupted P_j in $WCORE$ is same as $f_j(x)$. So P_α will correctly reconstruct $f_0(x) = F(x, 0)$ and hence the secret $s = f_0(0)$ with very high probability.

- **Correctness 2:** Here we have to consider the case, when D is *corrupted*. Since in AWSS-SS-Share, every honest party agrees on $WCORE$ and OKP_j for $P_j \in WCORE$, a unique secret $s' \in \mathbb{F} \cup \{NULL\}$ is defined by (at least $t+1$) honest parties in $WCORE$ at the end of sharing phase. The committed secret s' is the constant term of the polynomial passing through points $(j, f_j(0))$'s, corresponding to *honest* P_j's in $WCORE$. If the points $(j, f_j(0))$ corresponding to honest P_j's in $WCORE$ define a unique degree-t polynomial, say $f_0(x)$, then we say that D's committed secret is $s' = f_0(0)$. Otherwise, we say that D's committed secret is $s' = NULL$. Whatever may be case, we show that with very high probability, an honest P_α will either reconstruct s' or $NULL$.

  - We consider the first case when $s' = f_0(0)$. This implies that the points $(j, f_j(0))$ corresponding to honest P_j's in $WCORE$ define a unique degree-t polynomial $f_0(x)$. We now claim that with very high probability, the value $\overline{f_j(0)}$ (hence the polynomial $\overline{f_j(x)}$) corresponding to an honest $P_j \in WCORE$, as reconstructed by P_α, is same as $f_j(x)$ that P_j received in sharing phase. This claim follows from the argument given in CORRECTNESS 1. We next claim that the value $\overline{f_j(0)}$ (hence the polynomial $\overline{f_j(x)}$) corresponding to a *corrupted* $P_j \in WCORE$, as reconstructed by P_α can be any value. This is because for a *corrupted* P_j in $WCORE$, 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 $\overline{f_k(j)}$. Also adversary can delay the messages such that the values (along with the signatures) of all corrupted $P_k \in OKP_j$ are revealed to P_α before the values of honest parties in OKP_j. Thus the reconstructed polynomial $\overline{f_j(x)}$ can be any t-degree polynomial according to the choice of $\mathcal{A}_t$. Now there are two possibilities: if the points $(j, \overline{f_j(0)})$ corresponding to *honest* P_j's in $WCORE$, along with the points $(j, \overline{f_j(0)})$ corresponding to *corrupted* P_j's in $WCORE$ lie on $f_0(x)$, then s' will be reconstructed. Otherwise $NULL$ will be reconstructed. Notice that since for all *honest* P_j's in $WCORE$, $\overline{f_j(0)} = f_j(0)$, no other secret (other than s') can be reconstructed with very high probability.
  - We next consider the second case when D's 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 each honest $P_j \in WCORE$ will be reconstructed correctly with very high probability. □

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

PROOF: In AWSS-SS-Share, there are $\mathcal{O}(n^2)$ instances of A-ICP, each dealing with $\ell = 1$ value. 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 $OKSet$s corresponding to each party in $WCORE$, where each $OKSet$ 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-SS-Share incurs a private communication of $\mathcal{O}(n^3\kappa^2)$ bits and A-cast of $\mathcal{O}(n^2 \log(n))$ bits. In AWSS-SS-Rec-Private, there are $\mathcal{O}(n^2)$ instances of private revelation of A-ICP, each dealing with $\ell = 1$ value. This requires a private communication of $\mathcal{O}(n^3\kappa^2)$ bits.

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

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

4 AWSS Scheme for Sharing Multiple Secrets

We now extend AWSS-SS to AWSS-MS, which consists of protocols AWSS-MS-Share and 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 ℓ 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-SS-Share ℓ times parallely, each sharing individual elements of S. However, from Lemma 9 this would incur a private communication of $\mathcal{O}(\ell n^3 \kappa^2)$ bits and A-cast of $\mathcal{O}(\ell n^2 \log(n))$ bits. However, instead of sharing each individual element, AWSS-MS-Share shares all elements of S concurrently, requiring a private communication of $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits and A-cast of $\mathcal{O}(n^2 \log n)$ bits. Thus the private communication of AWSS-MS-Share is less than what would have been required by ℓ parallel executions of AWSS-SS-Share. Moreover, the A-cast communication of AWSS-MS-Share is independent of the number of secrets; i.e., ℓ. The properties of AWSS-MS will follow from the properties of AWSS-SS in a straight forward manner.

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

<u>**AWSS-MS-Share**($D, \mathcal{P}, S$)</u>

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

1. Select ℓ random, symmetric bivariate polynomials $F^1(x,y), \dots, F^\ell(x,y)$ of degree-t in x and y over $\mathbb{F}$, such that for $l = 1, \dots, \ell$, $F^l(0,0) = s^l$.
2. For $i = 1, \dots, n$, send $ICSig(D, P_i, \mathcal{P}, (f_i^1(j), \dots, f_i^\ell(j))$ to P_i, for each $j = 1, \dots, n$. Here $f_i^l(x) = F^l(x,i)$, for $l = 1, \dots, \ell$.

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

1. Wait to receive $ICSig(D, P_i, \mathcal{P}, (f_i^1(j), \dots, f_i^\ell(j))$ 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 $ICSig(P_i, P_j, \mathcal{P}, (f_i^1(j), \dots, f_i^\ell(j)))$ 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 and if $f_j^l(i) = f_i^l(j)$ for all $l = 1, \dots, \ell$, then A-cast OK(P_i, P_j).

[WCORE CONSTRUCTION] and [WCORE VERIFICATION & AGREEMENT ON WCORE] are same as in AWSS-SS-Share.

<u>**AWSS-MS-Rec-Private**($D, \mathcal{P}, S, P_\alpha$)**:**</u> P_α-weak-private-reconstruction of S: This is very straight forward extension of AWSS-SS-Rec-Private. Here either $\overline{S} = (\overline{s^1}, \dots, \overline{s^\ell})$ or $NULL$ (if corresponding to one secret $NULL$ is reconstructed, then overall $NULL$ is reconstructed) is reconstructed.

Theorem 2 *(AWSS-MS-Share, AWSS-MS-Rec-Private) constitutes a valid statistical AWSS scheme with private reconstruction, which shares ℓ secrets. AWSS-MS-Share privately communicates $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits and A-casts $\mathcal{O}(n^2 \log n)$ bits. AWSS-MS-Rec-Private privately communicates $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits.*

PROOF: In AWSS-MS-Share, n^2 instances of A-ICP are executed. In addition, there are n^2 A-cast of OK(*, *) signals. So AWSS-MS-Share involves a private communication of $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits and A-casts $\mathcal{O}(n^2 \log n)$ bits. □

Remark 3 *As in the case of AWSS-SS, a corrupted D may commit ℓ $NULL$ values in AWSS-MS instead of committing ℓ elements from $\mathbb{F}$.*

Notation 3 (Notation for Using AWSS-MS) *As in AWSS-SS, we will invoke AWSS-MS-Share as AWSS-MS-Share($D, \mathcal{P}, (f^1(x), \dots, f^\ell(x))$) where D is asked to choose symmetric bivariate polynomials $F^1(x,y), \dots, F^\ell(x,y)$ such that $F^l(x,0) = f^l(x)$ holds for $l = 1, \dots, \ell$. Similarly, AWSS-SS-Rec-Private will be invoked as AWSS-SS-Rec-Private($D, \mathcal{P}, (f^1(x), \dots, f^\ell(x)), P_\alpha$) to enable P_α-weak-private-reconstruction of $(f^1(x), \dots, f^\ell(x))$. This may lead to the private reconstruction of either $(f^1(x), \dots, f^\ell(x))$ or $NULL$.*

5 AVSS Protocol for Sharing a Single Secret

We now present an AVSS scheme called AVSS-SS, consisting of sub-protocols AVSS-SS-Share and AVSS-SS-Rec-Private. AVSS-SS-Share allows D to share a *single secret* from $\mathbb{F}$. *Notice that unlike AWSS-SS-Share (presented in Section 3), AVSS-SS-Share ensures that a corrupted D always commits to a secret from*

$\mathbb{F}$. Protocol AVSS-SS-Rec-Private allows a specific party, say P_α, to *privately* reconstruct D's committed secret. We call the private reconstruction as *P_α-private-reconstruction*. While *P_α-private-reconstruction* can always ensure that P_α reconstructs D's committed secret with high probability, *P_α-weak-private-reconstruction* (introduced in Section 3) could only ensure that P_α reconstructs either D's committed secret or $NULL$. Structurally, we divide AVSS-SS-Share into a sequence of following three phases. *Each of the phases will be eventually completed by every honest party when D is honest.*

1. **Commitment by D:** Here D on having a secret s, commits the secret by transferring information to individual parties and by executing several instances of AWSS-SS-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 they together re-commit D's committed secret using instances of AWSS-SS-Share protocol.

While first two phases of AVSS-SS-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-SS-Rec-Private. That is if protocol AVSS-SS-Share stops after the second phase, then we may only ensure that either D's committed secret or $NULL$ will be reconstructed in AVSS-SS-Rec-Private. This would violate the claim that AVSS-SS 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-SS-Share protocol (see Notation 2 in Section 3 for the interpretation of committing polynomial using AWSS-SS-Share). During the course of executing these n instances of AWSS-SS-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-SS-Share, D employ a trick to guarantee that all the n instances of AWSS-SS-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 Figure 1 on next page. We now prove the properties of **Commitment by D Phase**.

Lemma 10 *In the code for* **Commitment by D Phase***:*

1. *If D is honest then eventually he will generate a common $WCORE$ of size $2t + 1$ for all the n AWSS-SS-Share. 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_js received from the A-cast of D, then every other honest party will also eventually accept the same.*

PROOF: In the code for **Commitment by D Phase**, D keeps on adding new parties in each $WCORE^i$ even after $WCORE^i$ becomes of size $2t + 1$. In addition, D also keeps on adding new parties in each OKP^i_j even after OKP^i_j becomes of size $2t + 1$. So if D is honest, then eventually all the $2t + 1$ honest parties will be added in each $WCORE^i$ and OKP^i_j. Moreover, each honest P_i will eventually A-cast `Matched-Column` signal, as $f_j(i) = g_i(j)$ will hold for all $j = 1, \ldots, n$ for an *honest* D. However, some corrupted parties may also be present in $WCORE^i$'s and OKP^i_j's. Moreover, these corrupted parties may also A-cast `Matched-Column` signal. But what ever may be the case, if D is honest then eventually he will find a common set $2t + 1$ parties in the $WCORE$ of all the n instances of AWSS-SS-Share, who have A-cast `Matched-Column` signal. This common set of $2t + 1$ 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 $2t + 1$ parties in $OKP^1_j, \ldots, OKP^n_j$. This common set of $2t + 1$ parties will constitute OKP_j which D will A-cast. Now from the property of A-cast, each honest party will eventually receive a common $WCORE$ of size $2t + 1$ and OKP_j of size $2t + 1$ 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-SS-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 same $WCORE$ and OKP_j for each $P_j \in WCORE$.

```
Code Commitment($D, \mathcal{P}, s$)

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-SS-Share($D, \mathcal{P}, f_i(x)$) for sharing $f_i(x)$.
ii. Code for $P_i$ – Every party in $\mathcal{P}$, including $D$, executes this code
    1. Wait to receive $f_i(x)$ and $g_i(y)$ from $D$.
    2. Participate in AWSS-SS-Share($D, \mathcal{P}, f_j(x)$) by executing steps in [Verification: Code for $P_i$] (of AWSS-SS-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-SS-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-SS-Share($D, \mathcal{P}, f_j(x)$). If yes then A-cast Matched-Column and execute the rest of the steps of AWSS-SS-Share($D, \mathcal{P}, f_j(x)$).
iii. WCORE Construction: Code for $D$ – Only $D$ executes this code.
    1. Construct $WCORE$ and corresponding $OKP_j$'s for each AWSS-SS-Share($D, \mathcal{P}, f_i(x)$) following the steps in [WCORE Construction] (of AWSS-SS-Share). Denote them by $WCORE^i$ and $OKP_j^i$'s.
    2. Keep updating $WCORE^i$'s and corresponding $OKP_j^i$'s.
    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 $P_i$
    1. Wait to receive $WCORE$ and $OKP_j$ for every $P_j \in WCORE$ from A-cast of $D$, such that $|WCORE| = 2t+1$ and each $|OKP_j| = 2t+1$.
    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-SS-Share.
    3. Wait to receive Matched-Column from A-cast of every $P_j \in WCORE$.
    4. After receiving all desired OKs 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 Phase).
```

Figure 1: Code for Commitment by D Phase

5.2 Verification of D's Commitment Phase

After agreeing on $WCORE$ and corresponding OKP_j's, in this phase, the parties verifies whether indeed D has committed a secret from $\mathbb{F}$. For this, we try to check whether there exists a set of *honest parties* of size at least $t+1$, such that for every two parties P_i, P_j in this set, $f_i(j) = g_j(i)$ holds. If we can ensure the availability of such a set then it implies that the row and column polynomials of the parties in this set define a unique bivariate polynomial of degree-(t,t) and the constant term of the polynomial is D's committed secret. Checking for the availability of such a set is quiet easy in synchronous settings, where the parties can simply pair-wise exchange their common values on their row and column polynomial, as done in several synchronous VSS protocols [7, 23, 21, 28, 30]. However, doing the same is not so straightforward in asynchronous settings with $n = 3t+1$.

To check the availability of the set of parties described above, we proceed as follows: recall that in the **Commitment by D phase**, D is committed to $f_1(x), \ldots, f_n(x)$. So we execute AWSS-SS-Rec-Private

$(D, \mathcal{P}, f_j(x), P_j)$ for enabling P_j-weak-private-reconstruction of $f_j(x)$. If P_j has reconstructed $\overline{f_j}(x)$ from the execution of AWSS-SS-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 indication by P_j that $f_j(x)$ which is committed by D to the parties in $WCORE$ 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 $g_i(y)$ 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 Figure 2.

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

```
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-SS-Rec-Private$(D, \mathcal{P}, f_j(x), P_j)$,
       for $j = 1, \ldots, n$, to enable $P_j$-weak-private-reconstruction of $f_j(x)$. Notice that the common $WCORE$ acts as
       $WCORE$ in each AWSS-SS-Rec-Private$(D, \mathcal{P}, f_j(x), P_j)$, for $j = 1, \ldots, n$
    2. At the completion of AWSS-SS-Rec-Private$(D, \mathcal{P}, f_i(x), P_i)$, 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.
    4. If Matched-Row is received from A-cast of at least $2t+1$ parties then proceed to third phase.
```

Figure 2: Code for Verification of D's Commitment Phase

Lemma 11 *In code Verification, if `Matched-Row` is received from the A-cast of at least $2t+1$ parties, say $\mathcal{R}$, then in code Commitment, D is committed to a unique degree-(t,t) bivariate polynomial $\overline{F}(x,y)$ 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 $\overline{F}(x,y) = F(x,y)$.*

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) over $\mathbb{F}$, such that for $i = 1, \ldots, l$, we have $\overline{F}(x,i) = f_i(x)$ and for $j = 1, \ldots, m$, we have $\overline{F}(j,y) = p_j(y)$. The proof completely follow from the proof of Lemma 4.26 of [10].

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 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 $H = \left((V^{(t+1)})^T\right)^{-1} \cdot E$ 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 $t+1 < j \leq n$. The second part of the lemma is trivially true. □

Lemma 12 *In Verification, if D is honest then all the honest parties will eventually proceed to third phase. 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 will eventually A-cast `Matched-Row` signal. However, some corrupted parties may also A-cast `Matched-Row` signal. But what ever may be the case, eventually there will be a set of $2t+1$ parties, say $\mathcal{R}$, who will A-cast `Matched-Row` signal. So every honest party will eventually receive $2t+1$ `Matched-Row` signal and will proceed to the third phase.

If D is corrupted and some honest party P_i proceeds to the third phase, then it implies that P_i has received `Matched-Row` signal from $2t+1$ parties. So eventually all other honest parties will also receive these `Matched-Row` signals and will proceed to the third phase. □

From Lemma 11, if an honest party, say P_i, receives A-cast of `Matched-Row` signal during Verification from at least $2t+1$ parties, say $\mathcal{R}$, then he is sure that D is committed to a unique bivariate polynomial and thus a unique secret. Now the question is: *If P_i stops protocol AVSS-SS-Share here after finding such a set $\mathcal{R}$, 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-SS-Share which is actually required to enable robust reconstruction of D's committed secret in the reconstruction phase i.e in AVSS-SS-Rec-Private.

One possible way to reconstruct D's committed secret s is to execute AWSS-SS-Rec-Private($D, \mathcal{P}, f_j(x), *$) 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 for a *corrupted D*, *all* instances of AWSS-SS-Rec-Private may output $NULL$. So it seems that most likely there is no way to robustly reconstruct D's committed secret s in protocol AVSS-SS-Rec-Private, if AVSS-SS-Share stops after current phase. Hence, we require the third phase which is described in the sequel. Before we formally describe the next phase, we would like to give the following remark.

Remark 4 *In the code* **Commitment***, D executed n instances of AWSS-SS-Share for individually committing to each $f_i(x)$. This later 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-SS-Share, then later in the code for* **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_α 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 Verification, then P_i acts as a dealer to re commit his row polynomial $f_i(x)$ by initiating an instance of AWSS-SS-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-SS-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-SS-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-SS-Rec-private instance of an *honest* $P_i \in VCORE$ will always reconstruct back $f_i(x)$. On the other hand, AWSS-SS-Rec-private instance of a *corrupted* $P_i \in VCORE$ will output either $f_i(x)$ or $NULL$. 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 the s. The protocol for this phase is given in next page.

Lemma 13 *If D is honest then D will eventually generate $VCORE$ of size $2t+1$ and 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: If D is honest, then every honest P_i will eventually successfully complete AWSS-SS-Share$(P_i, \mathcal{P}, f_i(x))$ as a dealer and thus will successfully re-commit $f_i(x)$. In addition, some corrupted P_i's may also successfully re-commit $f_i(x)$ as a dealer. But what ever may be the case, 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 eventually do the same and hence will accept $VCORE$. □

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

i. CODE FOR P_i:

1. If you have A-casted `Matched-Row` in Verification then as a dealer, initiate AWSS-SS-Share$(P_i, \mathcal{P}, f_i(x))$ to re commit $f_i(x)$.
2. If P_j has A-casted `Matched-Row` in Verification, then participate in AWSS-SS-Share$(P_j, \mathcal{P}, f_j(x))$ by executing steps in [VERIFICATION: CODE FOR P_i] (of AWSS-SS-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-SS-Share$(P_j, \mathcal{P}, f_j(x))$ and $g_i(y)$ was obtained from D during **commitment by D phase**. If yes then participate in the remaining steps in [VERIFICATION: CODE FOR P_i] corresponding to AWSS-SS-Share$(P_j, \mathcal{P}, f_j(x))$.
3. $WCORE^{P_i}$ CONSTRUCTION FOR AWSS-SS-Share$(P_i, \mathcal{P}, f_i(x))$: If P_i as a dealer initiated AWSS-SS-Share$(P_i, \mathcal{P}, f_i(x))$ to re commit $f_i(x)$, then P_i as a dealer, constructs $WCORE$ and corresponding OKP_js for AWSS-SS-Share$(P_i, \mathcal{P}, f_i(x))$ in a slightly different way than what is described in AWSS-SS-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-SS-Share$(P_i, \mathcal{P}, f_i(x))$.
   (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) for at least $2t+1$ P_k's who are re-committing their corresponding $f_k(x)$'s, $(P_j, ProbCORE^{P_k})$ is received from their A-cast.
   (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

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 are re-committing their corresponding $f_k(x)$'s.
   (c) Wait to receive $OK(P_j, P_k)$ for every $P_k \in OKP_j$ in execution AWSS-SS-Share$(P_i, \mathcal{P}, f_i(x))$.
2. A-cast $VCORE$ when $|VCORE| = 2t+1$.

iii. VCORE VERIFICATION & AGREEMENT ON VCORE: CODE FOR P_i

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-SS-Share.

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-SS-Share$(P_i, \mathcal{P}, f_i(x))$ (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 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 $P_k \in \mathcal{H}$ (see 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 $g_j(i) = f_i(j)$ (see step i-(2)) in Re-Commitment.

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

5.4 Protocol AVSS-SS

Now the protocol for our AVSS scheme is as follows:

Protocol **AVSS-SS**$(D, \mathcal{P}, s)$

AVSS-SS-Share$(D, \mathcal{P}, S)$

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

AVSS-SS-Rec-Private$(D, \mathcal{P}, s, P_\alpha)$**:** Private reconstruction of s by party P_α:

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

1. Participate in AWSS-SS-Rec-Private$(P_j, \mathcal{P}, f_j(x), P_\alpha)$ for every $P_j \in VCORE$.

LOCAL COMPUTATION: CODE FOR P_α

1. For every $P_j \in VCORE$, obtain either $\overline{f_j(x)}$ or $NULL$ from P_α-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.

We now prove the properties of our AVSS scheme.

Lemma 15 *AVSS-SS satisfies termination property of Definition 2.*

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 AWSS-SS (see Lemma 6 and Lemma 8), AWSS-SS-Share$(P_i, \mathcal{P}, f_i(x))$ initiated by an *honest* P_i in $VCORE$ during Re-Commitment, will reconstruct $f_i(x)$ in its reconstruction phase with high probability. But AWSS-SS-Share$(P_i, \mathcal{P}, f_i(x))$ 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 from $VCORE$, reconstruction of $f_i(x)$'s will be successful. This is enough to reconstruct the secret s. Hence if all honest parties terminate AVSS-SS-Share and every (honest) party starts AVSS-SS-Rec-Private, then an honest P_α will eventually terminate AVSS-SS-Rec-Private. □

Lemma 16 *AVSS-SS satisfies secrecy property of Definition 2.*

PROOF: We have to consider the case when D is honest. Throughout AVSS-SS-Share, essentially the parties only exchange common values on their row and column polynomials. Hence by the property of bivariate polynomial of degree-(t,t), the constant term of it is always secure. The rest follows from the secrecy of AWSS-SS-Share. □

Lemma 17 *AVSS-SS 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-SS-Rec-Private$(P_i, \mathcal{P}, f_i(x), P_\alpha)$, with very high probability 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 $t+1$ honest parties, $f_i(x)$ will be reconstructed. Using those polynomials $\overline{F}(x,y)$ and $\overline{s} = \overline{F}(0,0)$ will be reconstructed. Moreover, $s = \overline{s} = F(0,0)$ if D is honest. □

Lemma 18 *AVSS-SS incurs private communication of $\mathcal{O}((n^4\kappa)\kappa)$ bits and A-cast of $\mathcal{O}(n^3\log(n))$ bits.*

Proof: The communication complexity follows from the fact that in the protocol, $\mathcal{O}(n)$ instances of AWSS scheme are executed, dealing with a *single* secret. □

Theorem 3 *Protocols (AVSS-SS-Share, AVSS-SS-Rec-Private) constitutes a valid statistical AVSS scheme.*

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

6 AVSS for Sharing Multiple Secrets

We now present a statistical AVSS scheme AVSS-MS, consisting of sub-protocols AVSS-MS-Share and 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 ℓ executions of AVSS-SS-Share, one for each $s^l \in S$, D can share S with a private communication of $\mathcal{O}((\ell n^4\kappa)\kappa)$ 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 $\mathcal{O}((\ell n^3 + n^4\kappa)\kappa)$ and A-cast of $\mathcal{O}(n^3\log(n))$ (independent of ℓ) bits. This shows that executing a *single instance* of AVSS-MS dealing with *multiple secrets* concurrently is advantageous over executing *multiple instances* of AVSS-SS 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-SS-Share and are presented below.

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

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 $P_i$.
    3. For $i = 1, \ldots, n$, initiate AWSS-MS-Share($D, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x))$) for sharing $(f_i^1(x), \ldots, f_i^\ell(x))$.
ii. CODE FOR $P_i$ – Every party in $\mathcal{P}$, including $D$, executes this code
    1. Wait to receive $f_i^l(x)$ and $g_i^l(y)$ for $l = 1, \ldots, \ell$ from $D$.
    2. Participate in AWSS-MS-Share($D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x))$) 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($D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x))$). If yes then A-cast Matched-Column.
Rest of the steps are same as in Commitment.
```

```
Code Verification-MS($D, \mathcal{P}, S$)

$P_j$-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($D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x)), P_j$), for $j = 1, \ldots, n$, to enable $P_j$-weak-private-
       reconstruction of $(f_j^1(x), \ldots, f_j^\ell(x))$. Notice that the common $WCORE$ acts as $WCORE$ in each
       AWSS-SS-Rec-Private($D, \mathcal{P}, f_j(x), P_j$), for $j = 1, \ldots, n$
    2. At the completion of AWSS-MS-Rec-Private($D, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x)), P_i$), 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.
    4. If Matched-Row is received from A-cast of at least $2t+1$ parties then proceed to Re-Commitment-MS phase.
```

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

i. CODE FOR P_i:

1. If you have A-casted Matched-Row in Verification-MS then initiate AWSS-MS-Share($P_i, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x))$ to recommit $(f_i^1(x), \ldots, f_i^\ell(x))$.
2. For each j, such that P_j has A-casted Matched-Row in Verification-MS, participate in AWSS-MS-Share($P_j, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x))$ 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($P_j, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x))$ and $(g_i^1(y), \ldots, g_i^\ell(y))$ was obtained from D in protocol Commitment-MS. If yes then participate in the next steps in [VERIFICATION: CODE FOR P_i] corresponding to AWSS-MS-Share($P_j, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x))$.

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

Now protocol AVSS-MS-Share($D, \mathcal{P}, S$) consists of the code presented in Code Commitment-MS($D, \mathcal{P}, S$), Code Verification-MS($D, \mathcal{P}, S$) and Code Re-commitment-MS($D, \mathcal{P}, S$) in this order. Protocol AVSS-MS-Rec-Private($D, \mathcal{P}, S, P_\alpha$) is very straight forward extension of AVSS-SS-Rec-Private. Here $\overline{S} = (\overline{s^1}, \ldots, \overline{s^\ell})$ is reconstructed. 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.

Theorem 4 *Protocols (AVSS-MS-Share, AVSS-MS-Rec-Private) constitutes a valid statistical AVSS scheme with private reconstruction, which privately communicates $\mathcal{O}((\ell n^3 + n^4\kappa)\kappa)$ and A-cast $\mathcal{O}(n^3 \log(n))$ bits.*

Remark 5 (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 ℓ 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 AVSS-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, $\overline{F^l}(x, y) = F^l(x, y)$.*

Notation 4 (Notation for Using AVSS-MS-Share) *In the subsequent sections, we will invoke AVSS-MS-Share as AVSS-MS-Share ($D, \mathcal{P}, (f^1(x), \ldots, f^\ell(x))$) 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 $F^1(x, y), \ldots, F^\ell(x, y)$ such that $F^l(x, 0) = f^l(x)$ holds for $l = 1, \ldots, \ell$. Similarly, AVSS-SS-Rec-Private will be invoked as AVSS-SS-Rec-Private($D, \mathcal{P}, (f^1(x), \ldots, f^\ell(x)), P_\alpha$) to enable P_α-private-reconstruction of $(f^1(x), \ldots, f^\ell(x))$.*

7 ACSS for Sharing a Single Secret

Though AVSS-SS is an AVSS scheme, it is not an ACSS scheme because it fails to achieve *completeness* property. This is because in AVSS-SS-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 a statistical ACSS scheme called ACSS-SS, which consists of sub-protocols ACSS-SS-Share, ACSS-SS-Rec-Private and ACSS-SS-Rec-Public. Protocol ACSS-SS-Share allows D to generate t-sharing of a secret $s \in \mathbb{F}$. Given t-sharing of secret s, protocol ACSS-SS-Rec-Private allows a specific party in $\mathcal{P}$, say P_α, to privately reconstruct s. On the other hand, ACSS-SS-Rec-Public allows every party in $\mathcal{P}$ to reconstruct D's committed secret s. Protocol ACSS-SS-Rec-Public will be used in our AMPC protocol.

The high level idea of ACSS-SS-Share is similar as that of AVSS-SS-Share with the following difference: in AVSS-SS-Share, we used AWSS-SS-Share as a black-box. So if D is corrupted and even if it is ensured that D is committed to a unique bi-variate polynomial $\overline{F}(x, y)$ during **Verification Phase**, 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 $\overline{s} = \overline{F}(0, 0)$. It may happen that there are potential t honest P_i's who have not A-cast Matched-Row signal and who do not hold their corresponding $\overline{F}(x, i)$'s, as P_i-weak-private-reconstruction of $f_i(x)$'s corresponding to these parties would have reconstructed $NULL$ during **Verification Phase**.

On the other hand, we use AVSS-SS-Share as a black-box in ACSS-SS-Share. This avoids the above problem because now D would AVSS-Share each $f_i(x)$, instead of AWSS-Share. So once it is ensured that D is committed to a unique bi-variate polynomial $\overline{F}(x, y)$, by the property of AVSS-SS-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-SS-Rec-Private and ACSS-SS-Rec-Public uses the properties of *Online Error Correction* (OEC) [10]. Informally, given a t-sharing of s which is t-shared using degree-t polynomial $f(x)$, OEC allows to reconstruct $f(x)$ and hence $s = f(0)$ in an on-line fashion in asynchronous settings by using the properties of Reed-Solomon error correcting codes.

Protocol ACSS-SS$(D, \mathcal{P}, s)$

ACSS-SS-Share$(D, \mathcal{P}, s)$

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. 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-SS-Share$(D, \mathcal{P}, f_i(x))$ for sharing $f_i(x)$, where $f_i(x) = F(x, i)$. We call $f_i(x)$ as i^{th} row polynomial.

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

1. Wait to receive $g_i(y)$ from D.
2. Participate in AVSS-SS-Share$(D, \mathcal{P}, f_j(x))$ for all $j = 1, \ldots, n$.
3. If $f_j(i)$ is received from D during AVSS-SS-Share$(D, \mathcal{P}, f_j(x))$ 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-SS-Share$(D, \mathcal{P}, f_i(x))$. Denote it by $VCORE^i$.
2. Keep updating $VCORE^i$. 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-SS-Share$(D, \mathcal{P}, f_j(x))$ 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-SS-Share).

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-SS-Share$(D, \mathcal{P}, f_j(x))$ for every $j = 1, \ldots, n$, then participate in AVSS-SS-Rec-Private$(D, \mathcal{P}, f_j(x), P_j)$, for $j = 1, \ldots, n$, to enable P_j-private-reconstruction of $f_j(x)$. Notice that CCORE is used as VCORE in each AVSS-SS-Rec-Private$(D, \mathcal{P}, f_j(x), P_j)$, for $j = 1, \ldots, n$.
2. At the completion of AVSS-SS-Rec-Private$(D, \mathcal{P}, f_i(x), P_i)$, obtain degree-$t$ polynomial $f_i(x)$.
3. Output $f_i(0)$ as i^{th} share of s and terminate ACSS-SS-Share.

ACSS-SS-Rec-Private$(D, \mathcal{P}, s, P_\alpha)$: P_α-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_α.

ii. Code for P_α – Only $P_\alpha \in \mathcal{P}$ executes this code.

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

ACSS-SS-Rec-Public$(D, \mathcal{P}, s)$: 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-SS-Rec.

We now prove the properties of ACSS-SS.

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

1. *If D is honest then eventually he will generate a common $CCORE$ of size $2t + 1$ for all the n instances of AVSS-SS-Share. Moreover, each honest party will eventually agree on the common $CCORE$.*

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 also eventually accept the same.*

Proof: The proof follows using similar argument as in Lemma 10. $\square$

Lemma 20 *In ACSS-SS-Share, if the honest parties agree on a common* 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-SS-Share$(D,\mathcal{P},f_i(x))$ 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 using similar argument as in Lemma 11. $\square$

Lemma 21 *In ACSS-SS-Share, if the honest parties agree on a common* CCORE, *then eventually every honest party will get his share of D's committed secret with very high probability.*

Proof: From the previous lemma, if the honest parties agree on a common 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-SS-Share$(D,\mathcal{P},f_i(x))$ satisfies $\overline{F}(x,i)=f_i(x)$. So from the properties of AVSS-SS-Rec-Private, P_i-private-reconstruction of $f_i(x)$ will result in every honest P_i reconstructing $f_i(x)$ and hence his share $f_i(0)$ of the secret with very high probability. $\square$

Lemma 22 *In ACSS-SS-Share, if D is honest then $\mathcal{A}_t$ will have no information about secret s.*

Proof: In ACSS-SS-Share, the parties essentially privately exchange their common values on their row and column polynomials. The proof now follows from the properties of bi-variate polynomial of degree-(t,t) and secrecy property of AVSS-SS-Share. $\square$

Lemma 23 *In ACSS-SS-Share:*

1. *If D is honest, then at the end of ACSS-SS-Share, every honest party will hold his share corresponding to t-sharing of s with high probability. Moreover, every honest party will output s at the end of ACSS-SS-Rec-Public.*

2. *If D is corrupted and some honest party has terminated ACSS-SS-Share, then there exists a unique $\overline{s} \in \mathbb{F}$, such that each honest party will eventually hold his share corresponding to t-sharing of $\overline{s}$ with high probability. Moreover, every honest party will output $\overline{s}$ at the end of ACSS-SS-Rec-Public.*

Proof: The lemma follows from Lemma 19, Lemma 20, Lemma 21 and properties of OEC. If D is honest then s will be t-shared using degree-t polynomial $f_0(x)=F(x,0)$, where $F(x,y)$ is the bivariate polynomial selected by D. On the other hand, if D is corrupted, then $\overline{s}$ will be t-shared using degree-t polynomial $\overline{f}_0(x)=\overline{F}(x,0)$, where $\overline{F}(x,y)$ is the bivariate polynomial committed by D. $\square$

Lemma 24 *Protocol ACSS-SS-Share privately communicates $\mathcal{O}(n^5\kappa^2)$ bits and A-casts $\mathcal{O}(n^4\log n)$ bits. Protocol ACSS-SS-Rec-Private and ACSS-SS-Rec-Public incurs a private communication of $\mathcal{O}(n\kappa)$ and $\mathcal{O}(n^2\kappa)$ bits respectively.*

Proof: The communication complexity of ACSS-SS-Share follows from the fact that in ACSS-SS-Share, there are n executions of AVSS-SS-Share. In ACSS-SS-Rec-Private, each party sends his share to P_α, incurring a communication cost of $\mathcal{O}(n\kappa)$ bits. In ACSS-SS-Rec, each party sends his share to every other party, incurring a communication cost of $\mathcal{O}(n^2\kappa)$ bits. $\square$

8 ACSS for Sharing Multiple Secrets

We now present a statistical ACSS scheme ACSS-MS, consisting of sub-protocols ACSS-MS-Share, ACSS-MS-Rec-Private and ACSS-MS-Rec-Public. 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 using ℓ executions of ACSS-SS-Share, one for each $s^l \in S$, D can ACSS-share S with a private communication of $\mathcal{O}((\ell n^5\kappa)\kappa)$ 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

$\mathcal{O}((\ell n^4 + n^5\kappa)\kappa)$ and A-cast of $\mathcal{O}(n^4 \log(n))$ (independent of ℓ) bits. This shows that executing a *single instance* of ACSS-MS dealing with *multiple secrets* concurrently is advantageous over executing *multiple instances* of ACSS-SS dealing with *single secret.* The proof of the properties of ACSS-MS follows in a straight forward manner from the proof of the properties of ACSS-SS.

Theorem 5 *Protocols (ACSS-MS-Share, ACSS-MS-Rec-Public) constitutes a valid statistical ACSS scheme with public reconstruction. Protocol ACSS-MS-Share privately communicates $\mathcal{O}((\ell n^4 + n^5\kappa)\kappa)$ bits and A-casts $\mathcal{O}(n^4 \log n)$ bits. Protocol ACSS-MS-Rec-Private and ACSS-MS-Rec-Public incurs a private communication of $\mathcal{O}(\ell n\kappa)$ and $\mathcal{O}(\ell n^2\kappa)$ bits respectively.*

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

<u>ACSS-MS-Share($D, \mathcal{P}, S$)</u>

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

1. Select ℓ 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($D, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x))$) for sharing $(f_i^1(x), \ldots, f_i^\ell(x))$, where $f_i^l(x) = F^l(x,i)$. We call polynomials $f_i^1(x), \ldots, f_i^\ell(x)$ as i^{th} row polynomials.

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

1. Wait to receive $g_i^l(y)$ for $l = 1, \ldots, \ell$ from D.
2. Participate in AVSS-MS-Share($D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x))$) 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($D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x))$) 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($D, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x))$). Denote it by $VCORE^i$.
2. Keep updating $VCORE^i$. 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($D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x))$) 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($D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x))$) for every $j = 1, \ldots, n$, then participate in AVSS-MS-Rec-Private($D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x)), P_j$), for $j = 1, \ldots, n$, to enable P_j-private-reconstruction of $(f_j^1(x), \ldots, f_j^\ell(x))$. Notice that CCORE is used as VCORE in each AVSS-MS-Rec-Private($D, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x)), P_j$), for $j = 1, \ldots, n$.
2. At the completion of AVSS-MS-Rec-Private($D, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x)), P_i$), obtain degree-$t$ polynomials $(f_i^1(x), \ldots, f_i^\ell(x))$.
3. Output $(f_i^1(0), \ldots, f_i^\ell(0))$ as i^{th} share of $(s^1, \ldots, s^\ell)$ and terminate ACSS-MS-Share.

<u>ACSS-MS-Rec-Private($D, \mathcal{P}, S, P_\alpha$)</u>: P_α-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} share of $s^1, \ldots, s^\ell$ to party $P_\alpha \in \mathcal{P}$.

ii. CODE FOR P_α – 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.

<u>ACSS-MS-Rec-Public($D, \mathcal{P}, S$)</u>: 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} share of $s^1, \ldots, s^\ell$ 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-Public.

Notation 5 (Notation for Using ACSS-MS) *In the subsequent sections, we will invoke ACSS-MS-Share as ACSS-MS-Share $(D, \mathcal{P}, (f^1(x), \ldots, f^\ell(x)))$ 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 $F^1(x,y), \ldots, F^\ell(x,y)$ 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($D,\mathcal{P},(f^1(x),\ldots,f^\ell(x)),P_\alpha$) to enable $P_\alpha \in \mathcal{P}$ to privately reconstruct $(f^1(x),\ldots,f^\ell(x))$. Similarly, ACSS-MS-Rec-Public will be invoked as ACSS-MS-Rec-Public($D,\mathcal{P},(f^1(x),\ldots,f^\ell(x))$) to enable each party in $\mathcal{P}$ to reconstruct $(f^1(x),\ldots,f^\ell(x))$.

9 Statistical Asynchronous Multiparty Computation with Optimal Resilience

We now show how to use our proposed ACSS scheme to design an efficient statistical *asynchronous multiparty computation* (AMPC) protocol with optimal resilience; i.e., with $n = 3t+1$.

9.1 Multiparty Computation

A Multiparty Computation (MPC) [36, 12, 7, 35] 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 most communication complexity. So the communication complexity of any general MPC protocol is usually given in terms of the communication complexity per multiplication gate* [5, 4, 3, 17, 27].

The MPC problem has been studied extensively over synchronous networks. However, MPC in asynchronous network has got comparatively less 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 $2^{-\Omega(\kappa)}$ in **correctness** and/or **termination**, for an error parameter κ. However, note that there is *no* compromise in **secrecy** property.

From [6], *perfectly secure* AMPC is possible iff $n \geq 4t+1$. On the other hand, *statistically secure* AMPC is possible iff $n \geq 3t+1$ [8]. 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
[8]	$t < n/3$ (optimal)	private– $\Omega(c_M n^{11} \kappa^4)$; A-cast– $\Omega(c_M n^{11} \kappa^2 \log(n))$
[33]	$t < n/4$ (non-optimal)	private– $\mathcal{O}(c_M n^4 \kappa)$
[32]	$t < n/4$ (non-optimal)	private–$\mathcal{O}(c_M n^2 \kappa)$

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 [8] was not done earlier and for the sake of completeness, we carry out the same in **APPENDIX A**). Recently [18] presented an efficient MPC protocol over networks that has 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 [26] and [27].

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 $\mathcal{O}(n^5\kappa)$ bits per multiplication gate. Thus our AMPC protocol significantly improves the communication complexity of only known optimally resilient statistically secure AMPC protocol of [8]. For designing our AMPC protocol, we use our proposed ACSS scheme.

9.4 The Approach Used in the AMPC of [8] and Current Article

1. **AMPC of [8]:** The AMPC protocol of [8] consists of input phase and computation phase. In input phase every party 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/ distributed among the parties, using the approach of [7]. 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 [8], the only known AVSS scheme with $n = 3t + 1$ was due to [11]. But it is shown in [8] that the use of the AVSS protocol of [11] 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 [11] does not guarantee the above property, which we may refer as *ultimate* property. This very reason motivated Ben-Or et. al [8] 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 [8] presents an USS scheme with $n = 3t + 1$ using the AVSS protocol of [11] as a building block. Essentially, in the USS protocol of [8], every share of the secret is committed using AVSS of [11] 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 [8], 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 [7, 35].

2. **AMPC of Current Article:** Our AMPC protocol is presented in preprocessing model of [2] 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, 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 share (commit to) their inputs and 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 [2] 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 [8] and our ACSS, though both of them offer all the properties of AVSS. While USS of [8] 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:

(a) It makes the computation of the gates very simple;

(b) Reconstruction phase of ACSS is very simple, efficient and can be achieved using *on-line error correction* of [10].

Apart from these advantages, our ACSS is strikingly better than USS of [8] in terms of communication complexity. While sharing phase of our ACSS privately communicates $\mathcal{O}((\ell n^4 + n^5\kappa)\kappa)$ bits and A-casts $\mathcal{O}(n^4 \log n)$ bits to share ℓ *secrets concurrently*, the sharing phase of USS in [8] privately communicates $\Omega(n^{10}\kappa^4)$ bits and A-casts $\Omega(n^{10}\kappa^2 \log(n))$ bits to share *only one secret.*

9.5 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) [4, 8]. It is an asynchronous primitive presented in [6, 8]. It outputs a common set, containing at least $n - t$ parties, who correctly shared their values (using ACSS). 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 $\mathcal{O}(\text{poly}(n, \kappa))$ bits.

In addition to the ACS protocol, we also use another protocol, which allows the parties in $\mathcal{P}$ to jointly generate a random, non-zero element $r \in \mathbb{F}$. The protocol works as follows: each $P_i \in \mathcal{P}$ shares a random non-zero $r_i \in \mathbb{F}$ using ACSS-SS-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 r_i's. Once $\mathcal{C}$ is agreed upon, ACSS-SS-Rec-Public 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 $P_i \in \mathcal{C}$. Now it is easy to see that the sum value is random. We call this protocol as RNG, which privately communicates $\mathcal{O}(n^6\kappa^2)$ bits and A-cast $\mathcal{O}(n^5 \log(n))$ bits.

10 Generating t-2D-Sharing

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

Definition 6 (t-2D-sharing [3]) *: A value s is t-2D-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-2D-sharing of s as $[[s]]_t$.*

The t-2D-sharing of s implies that s as well as it's shares are individually t-shared. Now we present a protocol t-2D-Share which allows D to simultaneously generate t-2D-sharing of $\ell \geq 1$ elements from $\mathbb{F}$, namely $s^1, \ldots, s^\ell$. If D is honest, then every honest party will eventually terminate t-2D-Share, and if some honest party has terminated t-2D-Share, then all the honest parties will eventually terminate t-2D-Share. 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 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 defines S^0. A similar verification technique was used in [3] 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 $s^1, \ldots, s^\ell$.

Lemma 25 *In protocol t-2D-Share, if D is honest, then each honest party will eventually terminate with correct t-2D-sharing. Moreover, $s^1, \ldots, s^\ell$ will remain secure. If D is corrupted, then with very high probability, the honest parties will terminate only if D has done correct t-2D-sharing.*

PROOF: The first part is easy to proof. For the second part, we consider the case when D is corrupted. For $i = 0, \ldots, n$, ACSS-MS-Share^i ensures that D has correctly t-shared some S^i. But it may happen that some S^i which is t-shared by D does not contain 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. This implies that in ACSS-MS-Share^j, D has used polynomials $\overline{q^{(0,j)}(x)}, \ldots, \overline{q^{(\ell,j)}(x)}$ to share $\overline{S^j}$, such that for at least one $l \in \{0, \ldots, \ell\}$, $\overline{q^{(l,j)}(0)} \neq q^l(j) = s^l_j$. That is, $\overline{q^{(l,j)}(0)} = \overline{s^l_j} \neq s^l_j$. Now consider $q^*_j(0) = s^0_j + rs^1_j + \ldots + r^l\overline{s^l_j} + \ldots + r^\ell s^\ell_j$. 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 ℓ degree polynomials with coefficients $(s_j^0, \ldots, \overline{s_j^l}, \ldots, s_j^\ell)$ and $(s_j^0, \ldots, s_j^l, \ldots, s_j^\ell)$ intersect at a random value r. Since any two ℓ degree polynomial can intersect each other at most at ℓ values, r has to be one of the ℓ 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 $\frac{\ell}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)}$. Thus with probability at least $1 - 2^{-\Omega(\kappa)}$, $q^*(j) \neq q_j^*(0)$ and thus no honest party will terminate the protocol. □

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

SHARING BY D: CODE FOR D

1. Select $s^0 \in_R \mathbb{F}$ 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 $q^{(l,i)}(x)$, such that $q^{(l,i)}(0) = q^l(i) = s_i^l$. Let $S^{ij} = (q^{(0,i)}(j), q^{(1,i)}(j), \ldots, q^{(\ell,i)}(j)) = (s_{ij}^0, s_{ij}^1, \ldots, s_{ij}^\ell)$.
3. Invoke ACSS-MS-Share$(D, \mathcal{P}, (q^0(x), q^1(x), \ldots, q^\ell(x)))$ for generating t-sharing of S^0 where P_j receives the shares S^j. Denote this instance of ACSS-MS-Share by ACSS-MS-Share^0.
4. For $i = 1, \ldots, n$, invoke ACSS-MS-Share$(D, \mathcal{P}, (q^{(0,i)}(x), q^{(1,i)}(x), \ldots, q^{(\ell,i)}(x)))$ for generating t-sharing of S^i where P_j receives the share-shares S^{ij}. Denote this instance of ACSS-MS-Share by ACSS-MS-Share^i.

VERIFICATION: CODE FOR P_i

1. Upon completion of ACSS-MS-Share^j for all $j \in \{0, \ldots, n\}$, participate in protocol RNG to generate random $r \in \mathbb{F}$.
2. Once r is generated, locally 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-Public$(D, \mathcal{P}, (s^*, s_1^*, \ldots, s_n^*))$ to 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 $i = 1, \ldots, n$, $q^*(i) \overset{?}{=} q_i^*(0)$. 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):

1. For $j = 1, \ldots, n$, participate in ACSS-MS-Rec-Private$(D, \mathcal{P}, S^j, P_j)$ 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.
2. Wait to privately reconstruct $q^{(0,i)}(x), \ldots, q^{(\ell,i)}(x)$ from ACSS-MS-Rec-Private$(D, \mathcal{P}, S^i, P_i)$ and terminate.

Theorem 6 *t-2D-Share privately communicates $\mathcal{O}((\ell n^5 + n^6\kappa)\kappa)$ bits and A-cast $\mathcal{O}(n^5 \log(n))$ bits.*

PROOF: Follows from the fact that there are $n + 1$ instances of ACSS-MS-Share, one instance of RNG, one instance of ACSS-MS-Rec-Public 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 [15] which was given for synchronous settings.

11.1 Generating Secret and 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 ℓ 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 ℓ secrets, unknown to $\mathcal{A}_t$. 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 core 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 only the honest parties in core set, then we will get $\frac{\ell}{n-2t} * (n - 2t) = \ell$ random t-2D-sharing. For this, we use *Vandermonde Matrix* [17] and its ability to extract randomness which has been exploited by [17, 4].

Vandermonde Matrix and Randomness Extraction [17]: 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 [17, 4].

Protocol **Random-t-2D-Share**$(\mathcal{P}, \ell)$

Code for P_i:

1. Select $L = \frac{\ell}{n-2t}$ random secret elements $(s^{(i,1)}, \ldots, s^{(i,L)})$. As a dealer, invoke t-2D-Share$(P_i, \mathcal{P}, S^i)$ to generate t-2D-sharing of $S^i = (s^{(i,1)}, \ldots, s^{(i,L)})$.
2. For $j = 1, \ldots, n$, participate in t-2D-Share$(P_j, \mathcal{P}, S^j)$.

Agreement on a Core-Set: Code for P_i

1. Create an accumulative set $C^i = \emptyset$. Upon terminating t-2D-Share$(P_j, \mathcal{P}, S^j)$, include P_j in C^i.
2. Take part in ACS with the accumulative set C^i as input.

Generation of Random t-2D-sharing: Code for P_i:

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-share $s_{ki}^{(j,1)}, \ldots, s_{ki}^{(j,L)}$ 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 denotes a $(n-t) \times (n-2t)$ publicly known *Vandermonde Matrix* [17] over $\mathbb{F}$.
   (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.

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

Lemma 26 *Random-t-2D-Share (eventually) terminates with very high probability for every honest party. It outputs t-2D-sharings of ℓ random secret values, unknown to $\mathcal{A}_t$. The protocol privately communicates $\mathcal{O}((\ell n^5 + n^7\kappa)\kappa)$ bits, A-cast $\mathcal{O}(n^6 \log(n))$ bits and requires one invocation of ACS.*

Proof: The termination property is easy to verify. The communication complexity follows from the fact that n instances of t-2D-Share, dealing with $\frac{\ell}{n-2t} = \frac{\ell}{\Theta(n)}$ values are executed. □

11.2 Proving $c = ab$

Consider the following problem: let $D \in \mathcal{P}$ has t-shared ℓ pair of values $(a^1, b^1), \ldots, (a^\ell, b^\ell)$. Now D wants to t-share $c^1, \ldots, c^\ell$ where $c^l = a^l b^l$ without leaking 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 [15] 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, β and βb. 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 theoretic secure. For the proof of correctness and secrecy, see [15]. 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, β and $b\beta$ is t-shared by D. Specifically, a corrupted D might have shared $\overline{\beta b} \neq \beta b$ or $\overline{c} \neq c$ 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 ℓ pairs $(a^{(l)}, b^{(l)})$.

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

SHARING BY D:

1. CODE FOR D: (a) Select ℓ non-zero random elements $\beta^1, \ldots, \beta^\ell$ from $\mathbb{F}$. For $1 \leq l \leq \ell$, 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$(D, \mathcal{P}, \mathcal{B})$, ACSS-MS-Share$(D, \mathcal{P}, \mathcal{C})$ and ACSS-MS-Share$(D, \mathcal{P}, \Lambda)$.
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 Λ respectively.

VERIFYING WHETHER $c^l = a^l . b^l$: CODE FOR P_i

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 $r \in \mathbb{F}$.
2. For $l = 1, \ldots, \ell$, locally compute $p_i^l = r a_i^l + \beta_i^l$, the i^{th} share $p^l = r a^l + \beta^l$. Participate in ACSS-MS-Rec-Public$(D, \mathcal{P}, (p^1, \ldots, p^\ell), \mathcal{P})$ to publicly reconstruct p^l for $l = 1, \ldots, \ell$.
3. Upon reconstruction of p^l's, locally compute $q_i^l = p^l b_i^l - d_i^l - r c_i^l$ for $1 \leq l \leq \ell$, to get the i^{th} share of $q^l = p^l b^l - d^l - r c^l$. Participate in ACSS-MS-Rec-Public$(D, \mathcal{P}, (q^1, \ldots, q^\ell))$ to publicly reconstruct q^l for $l = 1, \ldots, \ell$.
4. Upon reconstruction of q^l's, locally check whether for $l = 1, \ldots, \ell$, $q^l \overset{?}{=} 0$. If yes then terminate.

Lemma 27 *ProveCeqAB privately communicates $\mathcal{O}((\ell n^4 + n^5 \kappa)\kappa)$ bits and A-casts $\mathcal{O}(n^4 \log(n))$ bits. If an honest party terminates, then with high probability, D has t-1D-shared $c^l = a^l b^l$, for $1 \leq l \leq \ell$.*

11.3 Generating Multiplication Triples: The Preparation Phase Main Protocol

We now outline protocol PreparationPhase which generates t-sharing of $c_M + c_R$ 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 [22] 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 $2t$ degree polynomial, say $C(x)$ where $C(0) = c$, by Lagrange interpolation formula [14], c can be computed as $c = \sum_{i=1}^{n-t} r_i e_i$ 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 [14] 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 core set has indeed 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 information theoretically secure.

Protocol PreparationPhase$(\mathcal{P})$

CODE FOR P_i:

1. Participate in two instances of Random-t-2D-Share$(\mathcal{P}, c_M + c_R)$ to generate t-2D-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 $a_{ji}^1, \ldots, a_{ji}^{c_M + c_R}, b_{ji}^1, \ldots, b_{ji}^{c_M + c_R}$.
2. For $1 \leq k \leq c_M + c_R$, let $c^k = a^k b^k$. Upon termination of both the instances of Random-t-2D-Share, invoke ProveCeqAB$(P_i, \mathcal{P}, [a_i^1]_t, \ldots, [a_i^{c_M + c_R}]_t, [b_i^1]_t, \ldots, [b_i^{c_M + c_R}]_t)$ 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.
3. For $j = 1, \ldots, n$, participate in ProveCeqAB$(P_j, \mathcal{P}, [a_j^1]_t, \ldots, [a_j^{c_M + c_R}]_t, [b_j^1]_t, \ldots, [b_j^{c_M + c_R}]_t)$.

AGREEMENT ON A COMMON-SET: CODE FOR P_i

1. Create an accumulative set $C^i = \emptyset$. Upon completing ProveCeqAB$(P_j, \mathcal{P}, [a_j^1]_t, \ldots, [a_j^{c_M + c_R}]_t, [b_j^1]_t, \ldots, [b_j^{c_M + c_R}]_t)$ with dealer P_j, 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

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.

Lemma 28 *Each honest party eventually terminates* PreparationPhase *with very high probability. The protocol privately communicates $\mathcal{O}(((c_M + c_R)n^5 + n^7\kappa)\kappa)$ 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 $c_I = \sum_{i=1}^{n} c_i$. 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($\mathcal{P}$)**

Secret Sharing: Code for P_i

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

Agreement on a Common-Set: Code for P_i

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

Lemma 29 *Each honest party will eventually terminate* InputPhase *and outputs t-sharing of inputs of the parties in agreed common set C with very high probability. The protocol privately communicates $\mathcal{O}((c_I n^4 + n^6\kappa)\kappa)$ 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 the secret-sharing scheme, 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 [2]. Given a preprocessed random multiplication triple, which is already correctly t-shared, *Circuit Randomization* [2] 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 α and β publicly. The security follows from the fact that α and β are random, for a random (a, b, c).

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

For every gate in the circuit: Code for P_i

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-SS-Rec-Public to reconstruct α and β.
   (d) Upon reconstructing α and β, 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 $r_i = a^k_i$ as the i^{th} share of r.
5. **Output Gate:** $x = \mathsf{OGate}([x]_t)$: Participate in ACSS-SS-Rec-Public to reconstruct x.

Lemma 30 *Every honest party eventually terminates ComputationPhase with very high probability. Given t-sharing of $c_M + c_R$ secret random triples, the protocol computes the outputs of the circuit correctly and privately, by privately communicating $O(n^2(c_M + c_O)\kappa)$ bits.*

14 The New AMPC Protocol with Optimal Resilience

Now our new AMPC protocol 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, is: (1). Invoke PreparationPhase (2). Invoke InputPhase (3). Invoke ComputationPhase.

Theorem 7 *For every coalition of up to $t < n/3$ bad parties, the protocol AMPC securely computes the circuit representing function f and eventually terminates with very high probability for all the honest parties. AMPC privately communicates $\mathcal{O}((c_I n^4 + c_M n^5 + c_R n^5 + c_O n^2 + n^7\kappa)\kappa)$ bits, A-Cast $\mathcal{O}(n^6 \log(n))$ bits and requires 4 invocations to ACS.*

15 Conclusion and Open Problems

In this paper, we have presented a new statistical AVSS scheme with optimal resilience (i.e., $n = 3t + 1$), which significantly improves the communication complexity of only known statistical AVSS of [11] and [31]. Moreover, our AVSS achieves stronger properties than the AVSS of [31] with less 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 [8] (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 hence ACSS scheme. This will 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 *PODC*, pages 311–322, 2008.

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

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

[4] Z. Beerliová-Trubíniová and M. Hirt. Simple and efficient perfectly-secure asynchronous MPC. In *ASIACRYPT*, pages 376–392, 2007.

[5] Z. Beerliová-Trubíniová and M. Hirt. Perfectly-secure MPC with linear communication complexity. In *TCC*, pages 213–230, 2008.

[6] M. Ben-Or, R. Canetti, and O. Goldreich. Asynchronous secure computation. In *STOC*, pages 52–61, 1993.

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

[8] M. BenOr, B. Kelmer, and T. Rabin. Asynchronous secure computations with optimal resilience. In *PODC*, pages 183–192, 1994.

[9] G. Bracha. An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. In 3^{rd} *ACM PODC*, pages 154 – 162, 1984.

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

[11] R. Canetti and T. Rabin. Fast asynchronous Byzantine agreement with optimal resilience. In *Proc. of STOC 1993*, pages 42–51. ACM, 1993.

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

[13] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *Proc. of STOC 1985*, pages 383–395, 1985.

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

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

[16] R. Cramer, I. Damgård, and U. M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT*, pages 316–334, 2000.

[17] I. Damgård and J. B. Nielsen. Scalable and unconditionally secure multiparty computation. In *CRYPTO*, pages 572–590, 2007.

[18] I. Damgrd, M. Geisler, M. Krigaard, and J. Buus Nielsen. Asynchronous multiparty computation: Theory and implementation. Cryptology ePrint Archive, Report 2008/415, 2008.

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

[20] P. Feldman and S. Micali. An optimal algorithm for synchronous Byzantine agreemet. In *Proc. of STOC 1988*, pages 639–648. ACM, 1988.

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

[22] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified VSS and fact-track multiparty computations with applications to threshold cryptography. In *PODC*, pages 101–111, 1998.

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

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

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

[26] M. Hirt, J. B Nielsen, and B. Przydatek. Cryptographic asynchronous multi-party computation with optimal resilience (extended abstract). In *EUROCRYPT*, pages 322–340, 2005.

[27] M. Hirt, J. B Nielsen, and B. Przydatek. Asynchronous multi-party computation with quadratic communication. In *ICALP (2)*, pages 473–485, 2008.

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

[29] J. Katz and C. Y. Koo. On expected constant round protocols for Byzantine Agreement. In *CRYPTO*, pages 445–462, 2006.

[30] A. Patra, A. Choudhary, T. Rabin, and C. Pandu Rangan. The round complexity of verifiable secret sharing re-visited. In *CRYPTO*, pages 487–504, 2009.

[31] A. Patra, A. Choudhary, and C. Pandu Rangan. Simple and efficient asynchronous Byzantine Agreement with optimal resilience. In *PODC*, pages 92–101, 2009.

[32] 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.

[33] B. Prabhu, K. Srinathan, and C. Pandu Rangan. Trading players for efficiency in unconditional multiparty computation. In *SCN*, pages 342–353, 2002.

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

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

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

APPENDIX A: Communication Complexity of the AMPC of [8]

Here we carry out the communication complexity analysis for the AMPC of [8]. Recently, in [31], the authors have analyzed that for a single secret, the sharing phase of the AVSS scheme of [11] involves a private communication of $\Omega(n^9\kappa^4)$ bits and A-cast of $\Omega(n^9\kappa^2\log(n))$ bits. As the sharing phase of the USS scheme of [8] requires n invocations to the sharing phase of AVSS of [11], it incurs a private communication of $\Omega(n^{10}\kappa^4)$ bits and A-cast of $\Omega(n^{10}\kappa^2\log(n))$ bits at least. 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 $\Omega(n^{11}\kappa^4)$ and A-cast of $\Omega(n^{11}\kappa^2\log(n))$ bits.