Communication Efficient Perfectly Secure VSS and MPC in Asynchronous Networks with Optimal Resilience

Arpita Patra*, Ashish Choudhury**, and C. Pandu Rangan***

Dept of Computer Science and Engineering
IIT Madras, Chennai, India 600036
{arpitapatra10,prangan55}@gmail.com, partho_31@yahoo.co.in

Abstract. Verifiable Secret Sharing (VSS) is a fundamental primitive used in many distributed cryptographic tasks, such as Multiparty Computation (MPC) and Byzantine Agreement (BA). It is a two phase (sharing, reconstruction) protocol. The VSS and MPC protocols are carried out among n parties, where t out of n parties can be under the influence of a *Byzantine (active)* adversary, having *unbounded computing power*. It is well known that protocols for perfectly secure VSS and perfectly secure MPC exist in an asynchronous network iff $n \geq 4t+1$. Hence, we call any perfectly secure VSS (MPC) protocol designed over an asynchronous network with $n = 4t + 1$ as *optimally resilient* VSS (MPC) protocol.

A secret is d-shared among the parties if there exists a random degree-d polynomial whose constant term is the secret and each honest party possesses a distinct point on the degree-d polynomial. Typically VSS is used as a primary tool to generate t-sharing of secret(s). In this paper, we present an *optimally resilient, perfectly secure* Asynchronous VSS (AVSS) protocol that can generate d-sharing of a secret for any d, where $t \leq d \leq 2t$. This is the first *optimally resilient, perfectly secure* AVSS of its kind in the literature. Specifically, our AVSS can generate d-sharing of $\ell \geq 1$ secrets from $\mathbb{F}$ concurrently, with a communication cost of $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits, where $\mathbb{F}$ is a finite field. Communication complexity wise, the best known optimally resilient, perfectly secure AVSS is reported in [2]. The protocol of [2] can generate t-sharing of ℓ secrets concurrently, with the same communication complexity as our AVSS. However, the AVSS of [2] and [4] (the only known optimally resilient perfectly secure AVSS, other than [2]) does not generate d-sharing, for any $d > t$.

Interpreting in a different way, we may also say that our AVSS shares $\ell(d + 1 - t)$ secrets simultaneously with a communication cost of $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits. Putting $d = 2t$ (the maximum value of d), we notice that the amortized cost of sharing a single secret using our AVSS is only $\mathcal{O}(n \log |\mathbb{F}|)$ bits. This is a clear improvement over the AVSS of [2] whose amortized cost of sharing a single secret is $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.

* 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, Government of India.

D.J. Bernstein and T. Lange (Eds.): AFRICACRYPT 2010, LNCS 6055, pp. 184–202, 2010.

As an interesting application of our AVSS, we propose a new *optimally resilient, perfectly secure Asynchronous Multiparty Computation* (AMPC) protocol that communicates $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits *per multiplication gate*. The best known *optimally resilient perfectly secure* AMPC is due to [2], which communicates $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits per multiplication gate. Thus our AMPC improves the communication complexity of the best known AMPC of [2] by a factor of $\Omega(n)$.

Keywords: Verifiable Secret Sharing, Multiparty Computation.

1 Introduction

VSS or MPC 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 parties can be under the influence of a *computationally unbounded Byzantine (active) adversary* $\mathcal{A}_t$. The adversary $\mathcal{A}_t$ completely dictates the parties under its control and can force them to deviate from a protocol, in any arbitrary manner.

VSS: Any VSS scheme consists of two phases: (i) a *sharing phase* in which a special party in $\mathcal{P}$, called *dealer* (denoted as D), on having a secret $s \in \mathbb{F}$ (an element from a finite field $\mathbb{F}$), shares it among all the parties; (ii) a *reconstruction phase*, in which the parties reconstruct the secret from their shares. Informally, the goal of any VSS scheme is to allow D to share his secret s during the sharing phase, among the parties in $\mathcal{P}$ in such a way that the shares would later allow for a unique reconstruction of s in the reconstruction phase. Moreover, if D is *honest*, then the secrecy of s from $\mathcal{A}_t$ should be preserved until the reconstruction phase. VSS is one of the fundamental building blocks for many secure distributed computing tasks, such as MPC, Byzantine Agreement (BA), etc. VSS has been studied extensively over the past three decades in different settings and computational models (see [5,23,15,14,16,17] and their references).

MPC: MPC [25,10,5,23] 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 $x_i \in \mathbb{F}$, where $\mathbb{F}$ is a finite field and $|\mathbb{F}| \geq n$. 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 is usually given in terms of the communication complexity per multiplication gate* [3,2,1,20].

The VSS and MPC problem has been studied extensively over synchronous network, which assumes that there is a global clock and the delay of any message in the network channel is bounded. However, VSS and MPC in asynchronous network has got comparatively less attention, due to its inherent hardness. As asynchronous networks model the real life networks like Internet more appropriately than synchronous networks, the fundamental problems like VSS and MPC are worthy of deep investigation over asynchronous networks.

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 delivery of *all* messages in the network. However, $\mathcal{A}_t$ can *only schedule* the messages communicated between honest parties, without having any access to them. 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 a party can not wait to consider the values sent by all parties, as waiting for them could turn out to be endless. Hence the values of up to t (potentially honest) parties may have to be ignored. Due to this the protocols in asynchronous network are generally involved in nature and require new set of primitives. For comprehensive introduction to asynchronous protocols, see [8].

Asynchronous Verifiable Secret Sharing (AVSS) [4,8]**:** Let (Sh, Rec) be a pair of protocols in which a dealer $D \in \mathcal{P}$ shares a secret s from a finite field $\mathbb{F}$ using Sh. We say that (Sh, Rec) is a t-resilient *perfectly secure* AVSS scheme with n parties if the following hold for every possible $\mathcal{A}_t$:

• **Termination:** (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 all the honest parties invoke protocol Rec, then each honest party will eventually terminate Rec.

• **Correctness:** (1) If D is *honest* then each honest party upon terminating protocol Rec, outputs the shared secret s. (2) If D is *faulty* 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 $\overline{s}$.

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

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

Asynchronous Multi Party Computation (AMPC) [8]**:** A *perfectly secure* AMPC should satisfy the **Correctness** and **Secrecy** property of MPC. In addition, it should also satisfy **Termination** property, according to which, every honest party should eventually terminate the protocol.

d-sharing and $(t, 2t)$-sharing [1,3]: A value $s \in \mathbb{F}$ is d-shared among a set of parties $\overline{\mathcal{P}} \subseteq \mathcal{P}$ with $|\overline{\mathcal{P}}| \geq d + 1$ if there exists a degree-d polynomial $f(x)$ with $f(0) = s$ such that each honest $P_i \in \overline{\mathcal{P}}$ holds $s_i = f(i)$. The vector of shares of s belonging to the honest parties is called d-sharing of s and is denoted by $[s]_d$. A value s is said to be $(t, 2t)$-shared among the parties in $\mathcal{P}$, denoted as $[s]_{(t,2t)}$, if s is simultaneously t-shared, as well as $2t$-shared among the parties in $\mathcal{P}$.

A-cast[9,8]**:** It is an asynchronous broadcast primitive, which allows a special party in $\mathcal{P}$ (called the sender) to distribute a message identically among the parties in $\mathcal{P}$. If sender is honest, then every honest party eventually terminates A-cast with the sender's message. For a corrupted sender, if some honest party terminates with some message, then every other honest party will eventually terminate with **same** message. A-cast is elegantly implemented in [7] with $n = 3t+1$, which incurs a private communication of $\mathcal{O}(n^2 b)$ bits, for a b-bit message.

Agreement on Common Subset (ACS)[2,6]**:** It is an asynchronous primitive presented in [4,6]. It is used to determine and output a common set, containing at least $n-t$ parties, who correctly shared their values. Each honest party will eventually get a share, corresponding to each value, shared by the parties in the common set. ACS requires private communication of $\mathcal{O}(\text{poly}(n) \log |\mathbb{F}|)$ bits.

1.2 Our Contributions and Comparison with Existing Results

Our Contribution: From [4,8], perfectly secure AVSS and AMPC is possible iff $n \geq 4t + 1$. Hence, we call any *perfectly secure* AVSS (AMPC) protocol with $n = 4t + 1$ as *optimally resilient, perfectly secure* AVSS (AMPC) protocol. Typically, AVSS is used as a tool for generating t-sharing of secrets. In this paper, we present a novel *optimally resilient, perfectly secure* AVSS protocol that can generate d-sharing of ℓ secrets concurrently for any d, where $t \leq d \leq 2t$, with a private communication of $\mathcal{O}(\ell n^2 \log(|\mathbb{F}|))$ bits and A-cast of $\mathcal{O}(n^2 \log(|\mathbb{F}|))$ bits. Interpreting in a different way, we may also say that the amortized cost of sharing a single secret using our AVSS is only $\mathcal{O}(n \log |\mathbb{F}|)$ bits (this will be discussed in detail after the presentation of AVSS protocol in section 4).

To design our AVSS, we exploit several interesting properties of (n, t)-star (a graph theoretic concept presented in section 4.4.2 of [8]) in conjunction with some properties of bivariate polynomial with *different* degree in variable x and y. The (n, t)-star was used to design a perfectly secure optimally resilient AVSS protocol in [8] (the details of (n, t)-star is presented in Section 2 of current article). While the properties of (n, t)-star that our AVSS explores were not required in the AVSS of [8] (which generates only t-sharing of secrets), our AVSS uses them for the first time for generating d-sharing of secrets, where $t \leq d \leq 2t$.

As an interesting application of our AVSS, we design a new *optimally resilient, perfectly secure* AMPC protocol that communicates $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits *per multiplication gate*. This solves an open problem posed in [20]. Using our AVSS, we

first design an efficient protocol that generates $(t, 2t)$-sharing of a secret with a communication cost of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. Then following the approach of [11,2], given $(t, 2t)$-sharing of a secret, a multiplication gate is evaluated by communicating $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. This is how our AMPC attains quadratic communication complexity per multiplication gate.

Comparison of Our AVSS with Existing AVSS Protocols: In Table 1, we compare our AVSS with existing optimally resilient, perfectly secure AVSS protocols. We emphasize that the *AVSS protocols of [4,2] does not generate d-sharing for any* $d > t$. When negligible error probability is allowed in **Termination** and/or **Correctness**, we arrive at the notion of *statistical* AVSS. Statistical AVSS is possible iff $n \geq 3t + 1$ [9,6]. To the best of our knowledge, the AVSS protocols of [9,6,19,18] are the only known *optimally resilient statistical* AVSS (i.e., with $n = 3t + 1$). As these protocols have comparatively high communication complexity and are designed with $n = 3t + 1$ (with which perfectly secure AVSS is impossible), we do not compare our AVSS with the statistical AVSS protocols of [9,6,19,18]. Recently in [20], a statistical AVSS with $n = 4t + 1$ (i.e., with non-optimal resilience) is reported. Clearly, our AVSS achieves stronger properties than the AVSS of [20] (*the AVSS of [20] involves a negligible error probability, while our AVSS is perfect*), though both of them generate d-sharing for any $t \leq d \leq 2t$, with same communication complexity. For achieving perfectness, we use techniques which are completely different from [20].

Table 1. Our AVSS vs with Existing Optimally Resilient Perfectly Secure AVSS

Reference	# Secrets Shared	Type of Sharing Generated	Communication Complexity In Bits (CCIB)
[4]	1	Only t-sharing	Private–$\mathcal{O}(n^3 \log(\|\mathbb{F}\|))$; A-cast–$\mathcal{O}(n^2 \log(\|\mathbb{F}\|))$
[2]	$\ell \geq 1$	Only t-sharing	Private– $\mathcal{O}(\ell n^2 \log(\|\mathbb{F}\|))$; A-cast–$\mathcal{O}(n^2 \log(\|\mathbb{F}\|))$
This article	$\ell \geq 1$	d-sharing, for any $t \leq d \leq 2t$	Private– $\mathcal{O}(\ell n^2 \log(\|\mathbb{F}\|))$; A-cast–$\mathcal{O}(n^2 \log(\|\mathbb{F}\|))$

Comparison of Our AMPC with Existing AMPC Protocols: In Table 2, we compare our AMPC protocol with the existing optimally resilient, perfectly secure AMPC protocols in terms of communication complexity. We observe that our AMPC gains by a factor of $\Omega(n)$ as compared to the best known perfectly secure AMPC of [2]. If negligible error probability is allowed in **Termination** and/or **Correctness**, we arrive at the notion of *statistical* AMPC. Statistical AMPC is possible iff $n \geq 3t + 1$ [6]. Optimally resilient statistical AMPC protocols (i.e., with $n = 3t + 1$) are reported in [6,18]. As these

Table 2. Comparison of Our AMPC with Existing Optimally Resilient Perfectly Secure AMPC Protocols

Reference	CCIB / Multiplication Gate
[4,8]	$\mathcal{O}(n^6 \log(\|\mathbb{F}\|))$
[2]	$\mathcal{O}(n^3 \log(\|\mathbb{F}\|))$
This Article	$\mathcal{O}(n^2 \log(\|\mathbb{F}\|))$

protocols have very high communication complexity and are designed with $n = 3t+1$ (with which perfectly secure AMPC is impossible), we do not compare them with our AMPC protocol. Statistical AMPC with $n = 4t+1$ (i.e., with non-optimal resilience) are reported in [24,22,20]. Among them, the AMPC reported in [20] is the best, which communicates $\mathcal{O}(n^2 \log(|\mathbb{F}|))$ bits per multiplication gate. Though the protocol of [20] attains the same communication complexity as our AMPC protocol (per multiplication gate), *our protocol is perfect in all respects, while the protocol of [20] has error probability in both Termination and Correctness.*

2 Finding (n, t)-star Structure in a Graph

We now describe an existing solution for a graph theoretic problem, called finding (n, t)-star in an undirected graph $G = (V, E)$. Our AVSS protocol exploits several interesting properties of (n, t)-star.

Definition 1 ((n, t)-star[8,4]:). *Let G be an undirected graph with the n parties in $\mathcal{P}$ as its vertex set. We say that a pair $(\mathcal{C}, \mathcal{D})$ of sets with $\mathcal{C} \subseteq \mathcal{D} \subseteq \mathcal{P}$ is an (n, t)-star in G, if the following hold: (i) $|\mathcal{C}| \geq n - 2t$; (ii) $|\mathcal{D}| \geq n - t$; (iii) for every $P_j \in \mathcal{C}$ and every $P_k \in \mathcal{D}$ the edge (P_j, P_k) exists in G.*

Ben-Or et. al [4] have presented an elegant and efficient algorithm for finding an (n, t)-star in a graph of n nodes, *provided that the graph contains a clique of size $n - t$.* The algorithm, called Find-STAR outputs either an (n, t)-star or the message `star-Not-Found`. Whenever the input graph contains a clique of size $n - t$, Find-STAR always outputs an (n, t)-star in the graph.

Actually, algorithm Find-STAR takes the complementary graph $\overline{G}$ of G as input and tries to find (n, t)-$\overline{\text{star}}$ in $\overline{G}$ where (n, t)-$\overline{\text{star}}$ is a pair $(\mathcal{C}, \mathcal{D})$ of sets with $\mathcal{C} \subseteq \mathcal{D} \subseteq \mathcal{P}$, satisfying the following conditions: (i) $|\mathcal{C}| \geq n - 2t$; (ii) $|\mathcal{D}| \geq n - t$; (iii) there are no edges between the nodes in $\mathcal{C}$ and nodes in $\mathcal{C} \cup \mathcal{D}$ in $\overline{G}$. Clearly, a pair $(\mathcal{C}, \mathcal{D})$ representing an (n, t)-$\overline{\text{star}}$ in $\overline{G}$, is an (n, t)-star in G. Recasting the task of Find-STAR in terms of complementary graph $\overline{G}$, we say that Find-STAR outputs either a (n, t)-$\overline{\text{star}}$, or a message `star-Not-Found`. Whenever, *the input graph $\overline{G}$ contains an independent set of size $n - t$*, Find-STAR always outputs an (n, t)-$\overline{\text{star}}$. For simple notation, we denote $\overline{G}$ by H. The algorithm Find-STAR is presented in the following table. For properties of Find-STAR, please see [8].

Algorithm Find-STAR(H)

1. Find a maximum matching M in H. Let N be the set of matched nodes (namely, the endpoints of the edges in M), and let $\overline{N} = \mathcal{P} \setminus N$.
2. Compute output as follows (which could be either (n, t)-$\overline{\text{star}}$ or a message `star-Not-Found`):
   (a) Let $T = \{P_i \in \overline{N} | \exists P_j, P_k$ s.t $(P_j, P_k) \in M$ and $(P_i, P_j), (P_i, P_k) \in E\}$. T is called the set of triangle-heads.
   (b) Let $\mathcal{C} = \overline{N} \setminus T$.
   (c) Let B be the set of matched nodes that have neighbors in $\mathcal{C}$. So $B = \{P_j \in N | \exists P_i \in \mathcal{C}$ s. t. $(P_i, P_j) \in E\}$.
   (d) Let $\mathcal{D} = \mathcal{P} \setminus B$. If $|\mathcal{C}| \geq n - 2t$ and $|\mathcal{D}| \geq n - t$, output $(\mathcal{C}, \mathcal{D})$. Otherwise, output `star-Not-Found`.

3 AVSS for Generating d-Sharing of a Single Secret

We now present a novel AVSS protocol consisting of two sub-protocols, namely AVSS-Share-SS and AVSS-Rec-SS. The AVSS-Share-SS allows a dealer $D \in \mathcal{P}$ (dealer can be any party from $\mathcal{P}$) to d-share a single secret from $\mathbb{F}$, among the parties in $\mathcal{P}$, where $t \leq d \leq 2t$. Protocol AVSS-Rec-SS allows the parties in $\mathcal{P}$ to reconstruct the secret, given its d-sharing. The structure of AVSS-Share-SS is divided into a sequence of following three phases.

1. **Distribution Phase:** As the name suggests, in this phase, D on having a secret s, distributes information to the parties in $\mathcal{P}$.
2. **Verification & Agreement on CORE Phase:** Here parties jointly perform some computation and communication in order to verify consistency of the information distributed by D in **Distribution Phase**. In case of successful verification, all honest parties agree on a set of at least $3t+1$ parties called $CORE$, satisfying certain property (mentioned in the sequel).
3. **Generation of d-sharing Phase:** If $CORE$ is agreed upon in previous phase, then here every party performs local computation on the data received (during **Verification & Agreement on CORE Phase**) from the parties in $CORE$ to finally generate the d-sharing of secret s.

An honest party will terminate AVSS-Share-SS, if it successfully completes the last phase, namely **Generation of d-sharing Phase**. If D is honest then each honest party will eventually terminate the last phase. Moreover, if D is corrupted and some honest party terminates the last phase, then each honest party will also eventually terminate the last phase (and hence AVSS-Share-SS).

Remark 1. The sharing phase of statistical AVSS protocol of [20] is also structured into above three phases. However, our implementation of **Verification & Agreement on CORE Phase** is completely different from [20]. More importantly, the last two phases in [20] involves a negligible error probability, whereas our implementation of all the three phases are perfect (error free) in all respects.

3.1 Distribution Phase

Here D on having a secret s, selects a random bivariate polynomial $F(x,y)$ of degree-(d,t) (i.e., the degree of the polynomial in x is d and the degree of the polynomial in y is t), such that $F(0,0) = s$ and sends $f_i(x) = F(x,i)$ and $p_i(y) = F(i,y)$ to party P_i. As in [20], we will call the degree-d $f_i(x)$ polynomials as *row polynomials* and degree-t $p_i(y)$ polynomials as *column polynomials*.

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

CODE FOR D:

1. Select a random bivariate polynomial $F(x,y)$ of degree-(d,t) over $\mathbb{F}$, such that $F(0,0) = s$. Send $f_i(x) = F(x,i)$ and $p_i(y) = F(i,y)$ to party P_i, for $i = 1, \ldots, n$.

3.2 Verification and Agreement on CORE Phase

The goal of this phase is to check the existence of a set of parties called $CORE$. If a $CORE$ exists then every honest party will agree on $CORE$, where $CORE$ is defined as follows

Definition 2 (Property of $CORE$). *$CORE$ is a set of at least $3t+1$ parties such that the row polynomials (received in* **Distribution Phase***) of the honest parties in $CORE$ define a unique bivariate polynomial say, $\overline{F}(x,y)$ of degree-(d,t). Moreover, if D is honest, then $\overline{F}(x,y) = F(x,y)$, where $F(x,y)$ was chosen by D in* **Distribution Phase**.

The property of $CORE$ ensures that for every $j \in \{1,\ldots,n\}$, the j^{th} point on row polynomials of honest parties in $CORE$ define degree-t column polynomial $\overline{p_j}(y) = \overline{F}(j,y)$. So once $CORE$ is constructed and agreed upon by each honest party then $\overline{p_j}(0)$ can be privately reconstructed by P_j with the help of the parties in $CORE$ by using *online error correction* (OEC) [8] [1]. This will generate d-sharing of $\overline{s} = \overline{F}(0,0)$, where $\overline{s}$ will be d-shared using degree-d polynomial $\overline{f_0}(x) = \overline{F}(x,0)$ and each (honest) P_j will have his share $\overline{f_0}(j) = \overline{p_j}(0)$ of $\overline{s}$. Moreover, if D is honest, then $\overline{s} = s$ as $\overline{F}(x,y) = F(x,y)$. Note that even though the degree of row polynomials is more than t (if $d > t$), we create a situation where parties need not have to reconstruct them. To obtain the shares corresponding to d-sharing of s, the parties need to reconstruct degree-t column polynomials only. We now give an outline of this phase.

Outline of Current Phase: Here the parties upon receiving row and column polynomials (from D), interact with each other to check the consistency of their common values (on their polynomials). After successfully verifying the consistency, parties A-cast OK signals. Using these signals, a graph with the parties as vertex set is formed and applying Find-STAR on the graph, a sequence of distinct (n,t)-stars are obtained. The reason for constructing a sequence of (n,t)-stars will be clear in the sequel. Each (n,t)-star in such a graph defines a unique bivariate polynomial of degree-(d,t).

For every generated (n,t)-star, D tries to find whether $CORE$ can be generated from it. The generation process of $CORE$ attempts to use several interesting features of (n,t)-star (mainly its $\mathcal{C}$ component). We show that if D is honest and $\mathcal{C}$ component of some (n,t)-star $(\mathcal{C},\mathcal{D})$ contains at least $2t+1$ honest parties, then $CORE$ will be eventually generated from $(\mathcal{C},\mathcal{D})$. Moreover, we also show that if D is honest, then eventually some (n,t)-star $(\mathcal{C},\mathcal{D})$ will be generated, where $\mathcal{C}$ will contain at least $2t+1$ honest parties (though the dealer D may not know which (n,t)-star it is). *These two important properties of (n,t)-star in our context are the heart our AVSS protocol.* Furthermore, we show that if $CORE$ is generated from some (n,t)-star $(\mathcal{C},\mathcal{D})$ (irrespective of whether D is honest or corrupted), then both $CORE$, as well as $(\mathcal{C},\mathcal{D})$ define the same bivariate polynomial of degree-(d,t).

[1] OEC allows to reconstruct a degree-t polynomial by applying error correction in an online fashion in asynchronous settings. For details see [8].

The generation of many (n,t)-stars in our case is essential as the $\mathcal{C}$ component of the first (n,t)-star may not contain at least $2t+1$ honest parties and hence may never lead to $CORE$. *This implies that if our protocol stops after generating the first (n,t)-star then the protocol may not terminate even for an honest D.* However, we stress that existing AVSS of [4] need not generate a sequence of

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

i. CODE FOR P_i: Every party $P_i \in \mathcal{P}$ (including D) executes this code.

1. Wait to receive polynomials $\overline{f_i}(x)$ of degree-d and $\overline{p_i}(y)$ of degree-t from D. Upon receiving, send $\overline{f_{ij}} = \overline{f_i}(j)$ and $\overline{p_{ij}} = \overline{p_i}(j)$ to party P_j, for $j = 1, \ldots, n$.
2. Upon receiving $\overline{f_{ji}}$ and $\overline{p_{ji}}$ from P_j, check if $\overline{f_i}(j) \stackrel{?}{=} \overline{p_{ji}}$ and $\overline{p_i}(j) \stackrel{?}{=} \overline{f_{ji}}$. If both the equalities hold, A-cast $\mathtt{OK}(P_i, P_j)$.
3. Construct an undirected graph G_i with $\mathcal{P}$ as vertex set. Add an edge (P_j, P_k) in G_i upon receiving (a) $\mathtt{OK}(P_k, P_j)$ from the A-cast of P_k and (b) $\mathtt{OK}(P_j, P_k)$ from the A-cast of P_j.

ii. CODE FOR D: Only D executes this code.

1. For every new receipt of some $\mathtt{OK}(*,*)$ update G_D. If a new edge is added to G_D, then execute Find-STAR($\overline{G_D}$). Let there are $\alpha \geq 0$ distinct (n,t)-stars that are found in the past from different executions of Find-STAR($\overline{G_D}$).
   (a) Now if an (n,t)-star is found from the current execution of Find-STAR($\overline{G_D}$) that is distinct from all the α (n,t)-stars obtained before, do the following:
      i. Call the new (n,t)-star as $(\mathcal{C}^{\alpha+1}, \mathcal{D}^{\alpha+1})$.
      ii. Create a list $\mathcal{F}^{\alpha+1}$ as follows: Add P_j to $\mathcal{F}^{\alpha+1}$ if P_j has at least $2t+1$ neighbors in $\mathcal{C}^{\alpha+1}$ in G_D.
      iii. Create a list $\mathcal{E}^{\alpha+1}$ as follows: Add P_j to $\mathcal{E}^{\alpha+1}$ if P_j has at least $d+t+1$ neighbors in $\mathcal{F}^{\alpha+1}$ in G_D.
      iv. For every γ, with $\gamma = 1, \ldots, \alpha$ update $\mathcal{F}^{\gamma}$ and $\mathcal{E}^{\gamma}$:
         A. Add P_j to $\mathcal{F}^{\gamma}$, if $P_j \notin \mathcal{F}^{\gamma}$ and P_j has at least $2t+1$ neighbors in $\mathcal{C}^{\gamma}$ in G_D.
         B. Add P_j to $\mathcal{E}^{\gamma}$, if $P_j \notin \mathcal{E}^{\gamma}$ and P_j has at least $d+t+1$ neighbors in $\mathcal{F}^{\gamma}$ in G_D.
   (b) If no (n,t)-star is found or an (n,t)-star that has been already found in the past is obtained, then execute step (a).iv(A-B) to update existing $\mathcal{F}^{\gamma}$'s and $\mathcal{E}^{\gamma}$'s.
   (c) Now let β be the first index among already generated $\{(\mathcal{E}^1, \mathcal{F}^1), \ldots, (\mathcal{E}^{\delta}, \mathcal{F}^{\delta})\}$ such that both $\mathcal{E}^{\beta}$ and $\mathcal{F}^{\beta}$ contains at least $3t+1$ parties (Note that if step (a) is executed, then $\delta = \alpha + 1$; else $\delta = \alpha$). Assign $CORE = \mathcal{E}^{\beta}$ and A-cast $((\mathcal{C}^{\beta}, \mathcal{D}^{\beta}), (\mathcal{E}^{\beta}, \mathcal{F}^{\beta}))$.

iii. CODE FOR P_i: Every party $P_i \in \mathcal{P}$ (including D) executes this code.

1. Wait to receive $((\mathcal{C}^{\beta}, \mathcal{D}^{\beta}), (\mathcal{E}^{\beta}, \mathcal{F}^{\beta}))$ from the A-cast of D.
2. Wait until $(\mathcal{C}^{\beta}, \mathcal{D}^{\beta})$ becomes a valid (n,t)-star in G_i.
3. Wait until every party $P_j \in \mathcal{F}^{\beta}$ has at least $2t+1$ neighbors in $\mathcal{C}^{\beta}$ in G_i.
4. Wait until every party $P_j \in \mathcal{E}^{\beta}$ has at least $d+t+1$ neighbors in $\mathcal{F}^{\beta}$ in G_i.
5. Accept $CORE = \mathcal{E}^{\beta}$.

(n,t)-stars because it has to generate only t-sharing. Hence the AVSS of [4] stops after generating the first (n,t)-star and then using the $\mathcal{D}$ component of the generated (n,t)-star, it could generate t-sharing of s. Finally, once $CORE$ is obtained, it is then verified and agreed among the set of all honest parties in $\mathcal{P}$. The steps of this phase are given in protocol Verification-SS.

Lemma 1. *For any (n,t)-star $(\mathcal{C},\mathcal{D})$ in graph G_k of honest P_k, the row polynomials held by honest parties in $\mathcal{C}$ define a unique polynomial of degree-(d,t), say $\overline{F}(x,y)$, such that column polynomial $\overline{p_j}(y)$ held by every honest $P_j \in \mathcal{D}$ satisfies $\overline{p_j}(y) = \overline{F}(j,y)$. Moreover, if D is honest, then $\overline{F}(x,y) = F(x,y)$.*

Proof. For any (n,t)-star $(\mathcal{C},\mathcal{D})$, $|\mathcal{C}| \geq n-2t$ and $|\mathcal{D}| \geq n-t$. So $\mathcal{C}$ and $\mathcal{D}$ contain at least $n-3t \geq t+1$ and $n-2t \geq 2t+1$ honest parties, respectively. Let l and m be the number of honest parties in $\mathcal{C}$ and $\mathcal{D}$ respectively where $l \geq t+1$ and $m \geq 2t+1$. Without loss of generality, we assume that $P_1,\ldots,P_l$, respectively $P_1,\ldots,P_m$ are the set of honest parties in $\mathcal{C}$ and $\mathcal{D}$. Now by the construction of (n,t)-star, for every pair of honest parties (P_i,P_j) with $P_i \in \mathcal{C}$ and $P_j \in \mathcal{D}$, the row polynomial $\overline{f_i}(x)$ of honest P_i and the column polynomial $\overline{p_j}(y)$ of honest P_j satisfy $\overline{f_i}(j) = \overline{p_j}(i)$. We now prove that the above statement implies that there exists a unique bivariate polynomial $\overline{F}(x,y)$ of degree-(d,t), such that for $i=1,\ldots,l$, we have $\overline{F}(x,i) = \overline{f_i}(x)$ and for $j=1,\ldots,m$, we have $\overline{F}(j,y) = \overline{p_j}(y)$. The proof is similar to the proof of Lemma 4.26 of [8]. Due to space constraints, we give the complete proof in full version of the paper [21].

Lemma 2. *For an honest D, an (n,t)-star $(\mathcal{C}^\beta,\mathcal{D}^\beta)$ with $\mathcal{C}^\beta$ containing at least $2t+1$ honest parties will be generated eventually.*

Proof. For an honest D, eventually the edges between each pair of honest parties will vanish from the complementary graph $\overline{G_D}$. So the edges in $\overline{G_D}$ will be either (a) between an honest and a corrupted party OR (b) between a corrupted and another corrupted party. Let β be the first index, such that (n,t)-star $(\mathcal{C}^\beta,\mathcal{D}^\beta)$ is generated in $\overline{G_D}$, when $\overline{G_D}$ contains edges of above two types only. Now, by construction of $\mathcal{C}^\beta$ (see Algorithm Find-STAR), it excludes the parties in N (set of parties that are endpoints of the edges of maximum matching M) and T (set of parties that are triangle-head). An honest P_i belonging to N implies that $(P_i,P_j) \in M$ for some P_j and hence P_j is corrupted (as the current $\overline{G_D}$ does not have edge between two honest parties). Similarly, an honest party P_i belonging to T implies that there is some $(P_j,P_k) \in M$ such that (P_i,P_j) and (P_j,P_k) are edges in $\overline{G_D}$. This clearly implies that both P_j and P_k are surely corrupted. So for every honest P_i outside $\mathcal{C}^\beta$, at least one (if P_i belongs to N, then one; if P_i belongs to T, then two) corrupted party also remains outside $\mathcal{C}^\beta$. As there are at most t corrupted parties, C^β may exclude at most t honest parties. But still $\mathcal{C}^\beta$ is bound to contain at least $2t+1$ honest parties.

Notice that the above event happens after finite number of steps. This is because D applies Find-STAR on $\overline{G_D}$ every time after an edge is added to G_D and there can be $\mathcal{O}(n^2)$ edges in G_D. So there can be $\mathcal{O}(n^2)$ distinct (n,t)-stars generated by D. Now $(\mathcal{C}^\beta,\mathcal{D}^\beta)$ with $\mathcal{C}^\beta$ containing at least $2t+1$ honest parties will be one among these $\mathcal{O}(n^2)$ (n,t)-stars. □

Lemma 3. *In protocol Verification-SS, if D is honest, then eventually $CORE$ will be generated.*

Proof. By Lemma 2, the honest D will eventually generate an $(\mathcal{C}^\beta, \mathcal{D}^\beta)$ in G_D, with $\mathcal{C}^\beta$ containing at least $2t+1$ honest parties. Furthermore, if D is honest then eventually there will be edges between every pair of honest parties in the graph G_i of every honest P_i (including G_D). Thus, as all honest parties in $\mathcal{P}$ will have edges with the honest parties in $\mathcal{C}^\beta$, they will be eventually added to $\mathcal{F}^\beta$. Similarly, as all honest parties in $\mathcal{P}$ will have edges with the honest parties in $\mathcal{F}^\beta$, they will be eventually added to $\mathcal{E}^\beta$. Hence $|\mathcal{E}^\beta| \geq n-t$ and $|\mathcal{F}^\beta| \geq n-t$ will be satisfied and $CORE$ will be obtained by D. □

Lemma 4. *If an honest P_i has accepted $CORE$, then the row polynomials of the honest parties in $CORE$ define a unique bivariate polynomial of degree-(d,t).*

Proof. If an honest P_i has accepted $CORE$, then he has received $((\mathcal{C}^\beta, \mathcal{D}^\beta), (\mathcal{E}^\beta, \mathcal{F}^\beta))$ from the A-cast of D and checked their validity with respect to his own graph G_i. By Lemma 1, the row polynomials of the honest parties in $\mathcal{C}^\beta$ define a unique bivariate polynomial of degree-(d,t), say $\overline{F}(x,y)$. So the row polynomial held by an honest $P_i \in \mathcal{C}$ satisfies $\overline{f_i}(x) = \overline{F}(x,i)$. Now by the construction of $\mathcal{F}^\beta$, every *honest* $P_j \in \mathcal{F}^\beta$ has at least $2t+1$ neighbors in $\mathcal{C}^\beta$ which implies that $\overline{f_{kj}}$ values received from at least $2t+1$ parties in $\mathcal{C}^\beta$ lie on column polynomial $\overline{p_j}(y)$. This clearly implies $\overline{p_j}(y) = \overline{F}(j,y)$, as $t+1$ out of these $2t+1$ values are sent by honest parties in $\mathcal{C}$, who define $\overline{F}(j,y)$.

Similarly, by construction of $\mathcal{E}^\beta$, every *honest* $P_j \in \mathcal{E}^\beta$ has at least $d+t+1$ neighbors in $\mathcal{F}^\beta$ which implies that $\overline{p_{kj}}$ values received from at least $d+t+1$ parties in $\mathcal{F}^\beta$ lie on $\overline{f_j}(x)$. This implies that $\overline{f_j}(x) = \overline{F}(x,j)$, as at least $d+1$ out of these $d+t+1$ values are sent by honest parties in $\mathcal{F}^\beta$, who define $\overline{F}(x,j)$. Hence row polynomials of the honest parties in $CORE$ define $\overline{F}(x,y)$. □

3.3 Generation of d-Sharing Phase

Assuming that the honest parties in $\mathcal{P}$ have agreed upon a $CORE$, protocol d-Share-Generation-SS generates d-sharing in the following way: From the properties of $CORE$, the row polynomials of honest parties in $CORE$ define a unique bivariate polynomial say $\overline{F}(x,y)$ of degree-(d,t), such that each honest party P_i in $CORE$ possesses $\overline{f_i}(x) = \overline{F}(x,i)$. So the j^{th} point on $\overline{f_i}(x)$ polynomials corresponding to all honest P_i's in $CORE$, define degree-t polynomial $\overline{p_j}(y) = \overline{F}(j,y)$. Furthermore, $|CORE| \geq 3t+1$. So the parties in $CORE$ can enable each $P_j \in \mathcal{P}$ to privately reconstruct $\overline{p_j}(y)$ using OEC [8]. Once this is done, every P_j can output $\overline{p_j}(0)$ as the share of D's committed secret. Since $\overline{f_0}(j) = \overline{p_j}(0)$, it follows that $\overline{f_0}(0)(= \overline{F}(0,0))$ will be d-shared using the degree-d polynomial $\overline{f_0}(x) = \overline{F}(x,0)$. Clearly if D is honest, D's secret s will be d-shared using polynomial $f_0(x) = F(x,0)$, as $\overline{F}(x,y) = F(x,y)$ for honest D. The protocol for this phase is as follows:

Protocol d-Share-Generation-SS($D, \mathcal{P}, s, d$)

CODE FOR P_i:

1. Apply On-line Error Correcting (OEC) technique [8] on $\overline{f_{ji}}$'s received from every P_j in $CORE$ (during Protocol Verification-SS) and reconstruct degree-t polynomial $\overline{p_i}(y)$ and output $\overline{s_i} = \overline{p_i}(0) = \overline{f_0}(i)$ as the i^{th} share of $\overline{s}$ and terminate. $\overline{s}$ is now d-shared using polynomial $\overline{f_0}(x)$.

Lemma 5. *Assume that every honest party has agreed on $CORE$ where the row polynomials of the honest parties in $CORE$ define a unique bivariate polynomial of degree-(d,t), say $\overline{F}(x,y)$. Then protocol d-Share-Generation-SS will generate d-sharing of $\overline{s} = \overline{F}(0,0)$.*

Proof. See [21] for complete details. □

3.4 Protocol AVSS-Share-SS and AVSS-Rec-SS

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

(a) D executes Distribution-SS$(D, \mathcal{P}, s, d)$;
(b) Each party P_i participates in Verification-SS$(D, \mathcal{P}, s, d)$;
(c) After agreeing on $CORE$, each party P_i participates in d-Share-Generation-SS$(D, \mathcal{P}, s, d)$ and terminates AVSS-Share-SS after locally outputting the share corresponding to D's committed secret.

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

Party $P_i \in \mathcal{P}$ sends s_i, the i^{th} share of s to every $P_j \in \mathcal{P}$. Party P_i applies OEC on received s_j's, reconstructs s and terminates AVSS-Rec-SS.

Theorem 1. *Protocols (AVSS-Share-SS, AVSS-Rec-SS) constitute a valid perfectly secure AVSS scheme for d-sharing a single secret from $\mathbb{F}$.*

Proof. **Termination:** Part (1) of **Termination** says that if D is honest then every honest party will terminate AVSS-Share-SS eventually. By Lemma 3, D will eventually generate $CORE$ and A-cast the corresponding information i.e. $((\mathcal{C}^\beta, \mathcal{D}^\beta), (\mathcal{E}^\beta, \mathcal{F}^\beta))$. By the property of A-cast (and as graph G_i is constructed on the basis of A-casted information) every honest party will receive, verify the validity of D's A-casted information with respect to his own graph G_i and agree on the $CORE$. Now the proof for this part follows from Lemma 5.

Part (2) of **Termination** says that if an honest party terminated AVSS-Share-SS, then every other honest party will terminate AVSS-Share-SS eventually. An honest P_i has terminated the protocol implies that he has agreed on $CORE$. This means that P_i has received and verified the validity of D's A-casted information with respect to his own graph G_i. The same will happen eventually for all other

honest parties. Hence they will agree on $CORE$. Now the proof follows from Lemma 5. Part (3) of **Termination** follows from the correctness of OEC.

Correctness: If the honest parties terminate AVSS-Share-SS, then it implies that $\overline{s}(= \overline{F}(0,0))$ is properly d-shared among the parties in $\mathcal{P}$ (by Lemma 5), where $\overline{F}(x,y)$ is the unique bivariate polynomial of degree-(d,t) defined by the honest parties in $CORE$. Moreover if D is honest then $\overline{F}(x,y) = F(x,y)$ (follows from Lemma 1 and Lemma 4) and hence $\overline{s} = s$. Now the **Correctness** follows from the correctness of OEC.

Secrecy: Let $\mathcal{A}_t$ controls $P_1, \ldots, P_t$. So $\mathcal{A}_t$ will know $f_1(x), \ldots, f_t(x)$ and $p_1(y), \ldots, p_t(y)$. Throughout the protocol, the parties exchange common values (on row and column polynomials), which do not add any extra information to the view of $\mathcal{A}_t$. Now by the property of bivariate polynomial of degree-(d,t), $d-t+1$ coefficients of $f_0(x) = F(x,0)$ will remain secure, where $F(x,y)$ is the polynomial used by D to hide his secret s. So $s = f_0(0) = F(0,0)$ will remain secure. □

Theorem 2. *AVSS-Share-SS privately communicates $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits and A-casts $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. Protocol AVSS-Rec-SS privately communicates $\mathcal{O}(n^2 \log |\mathbb{F}|)$.*

Proof. See the full version of the paper [21]. □

4 AVSS for Generating d-Sharing of Multiple Secrets

We now present an AVSS protocol consisting of two sub-protocols, namely AVSS-Share-MS and AVSS-Rec-MS: AVSS-Share-MS allows a dealer $D \in \mathcal{P}$ to d-share $\ell \geq 1$ secret(s) from $\mathbb{F}$, denoted as $S = (s^1, \ldots, s^\ell)$, among the parties in $\mathcal{P}$, with $t \leq d \leq 2t$; AVSS-Rec-MS allows the parties to reconstruct the secrets, given their d-sharing. Notice that we can generate d-sharing of S by concurrently executing protocol AVSS-Share-SS (given in the previous section) ℓ times, once for each $s^i \in S$. But this will require a private communication of $\mathcal{O}(\ell n^2 \log(|\mathbb{F}|))$ and A-cast of $\mathcal{O}(\ell n^2 \log(|\mathbb{F}|))$ bits. However, our protocol AVSS-Share-MS requires a private communication of $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ and A-cast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. *Thus, the A-cast communication of our AVSS-Share-MS protocol is independent of ℓ.* The idea behind protocol AVSS-Share-MS is same as AVSS-Share-SS. Protocol AVSS-Share-MS is divided into a sequence of same three phases, as in AVSS-Share-SS. We now present the corresponding protocols.

Protocol Distribution-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d$)

CODE FOR D:

1. For $l = 1, \ldots, \ell$, select a random bivariate polynomials $F^l(x,y)$ of degree-(d,t), such that $F^l(0,0) = s^l$ and send the row polynomial $f_i^l(x) = F^l(x,i)$ and column polynomial $p_i^l(y) = F^l(i,y)$ to P_i.

Protocol Verification-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d$)

i. CODE FOR P_i: Every party $P_i \in \mathcal{P}$ (including D) executes this code.

1. Wait to receive $\overline{f_i^l}(x)$ and $\overline{p_i^l}(y)$ for all $l = 1, \ldots, \ell$, from D.
2. Upon receiving, check whether (i) $\overline{f_i^l}(x)$ is a degree-d polynomial for all $l = 1, \ldots, \ell$; and (ii) $\overline{p_i^l}(y)$ is a degree-t polynomial for all $l = 1, \ldots, \ell$. If yes, then send $\overline{f_{ij}^l} = \overline{f_i^l}(j)$ and $\overline{p_{ij}^l} = \overline{p_i^l}(j)$ for all $l = 1, \ldots, \ell$, to P_j.
3. Upon receiving $\overline{f_{ji}^1}, \ldots, \overline{f_{ji}^\ell}$ and $\overline{p_{ji}^1}, \ldots, \overline{p_{ji}^\ell}$ from P_j, check if $\overline{f_i^l}(j) \stackrel{?}{=} \overline{p_{ji}^l}$ and $\overline{f_{ji}^l} \stackrel{?}{=} \overline{p_i^l}(j)$ for all $l = 1, \ldots, \ell$. If the equality holds, then confirm the consistency by A-casting $\mathtt{OK}(P_i, P_j)$.
4. Construct an undirected graph G_i with $\mathcal{P}$ as vertex set. Add an edge (P_j, P_k) in G_i upon receiving (a) $\mathtt{OK}(P_k, P_j)$ from the A-cast of P_k and (b) $\mathtt{OK}(P_j, P_k)$ from the A-cast of P_j.

ii. CODE FOR D: (Only D executes this code): Same as in Protocol Verification-SS.

iii. CODE FOR P_i: (Every party $P_i \in \mathcal{P}$ (including D) executes this code): Same as in Protocol Verification-SS.

Protocol d-Share-Generation-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d$)

CODE FOR P_i:

1. For $l = 1, \ldots, \ell$, apply On-line Error Correcting (OEC) technique on $\overline{f_{ji}^l}$'s received from every P_j in $CORE$ (during Protocol Verification-SS) and reconstruct degree-t polynomial $\overline{p_i^l}(y)$ and output $\overline{s_i^l} = \overline{p_i^l}(0) = \overline{f_0^l}(i)$ as the i^{th} share of $\overline{s^l}$ and terminate. $\overline{s^l}$ is now d-shared using polynomial $\overline{f_0^l}(x)$.

Protocol AVSS-Share-MS and AVSS-Rec-MS are now given in the following table.

Protocol AVSS-Share-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d$)

(a) D executes Distribution-MS$(D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d)$;
(b) Each party P_i participates in Verification-MS$(D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d)$;
(c) After agreeing on $CORE$, each party P_i participates in d-Share-Generation-MS$(D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d)$ and terminates AVSS-Share-MS after locally outputting the shares corresponding to D's committed secrets.

Protocol AVSS-Rec-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell), d$)

1. For $l = 1, \ldots, \ell$, each party $P_i \in \mathcal{P}$ privately sends the i^{th} share of s^l, namely s_i^l, to every party $P_j \in \mathcal{P}$.
2. For $l = 1, \ldots, \ell$, party $P_i \in \mathcal{P}$ applies OEC on the received s_j^l's to privately reconstruct s^l and terminate AVSS-Rec-MS.

Theorem 3. *(AVSS-Share-MS, AVSS-Rec-MS) constitute a perfectly secure AVSS scheme for sharing $\ell \geq 1$ secret(s) from $\mathbb{F}$. AVSS-Share-MS privately communicates $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits and A-casts $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. AVSS-Rec-MS privately communicates $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits.*

4.1 A Different Interpretation of AVSS-Share-MS

In AVSS-Share-MS, every secret s^l for $l = 1, \ldots, \ell$ is d-shared using degree-d polynomial $f_0^l(x) = F^l(x, 0)$. Now by the **Secrecy** proof of AVSS-Share-SS, given in Theorem 1, we can claim that $(d+1)-t$ coefficients of $f_0^l(x)$ are information theoretically secure for every $l = 1, \ldots, \ell$. This implies that AVSS-Share-MS shares $\ell(d+1-t)$ secrets with a private communication of $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits and A-cast $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. As the A-cast communication is independent of ℓ, we may ignore it and conclude that the amortized cost of sharing a single secret using AVSS-Share-MS is only $\mathcal{O}(n \log |\mathbb{F}|)$. This is because by setting $d = 2t$, we see that AVSS-Share-MS can share $\ell(t+1) = \Theta(\ell n)$ secrets by privately communicating $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits. Now putting it in other way, D can share $\ell(t+1)$ secrets using AVSS-Share-MS by choosing a random polynomial $f_0^l(x)$ (of degree $d = 2t$) with lower order $t+1$ coefficients as secrets and then choosing a random degree-(d, t) bivariate polynomial $F^l(x, y)$ with $F^l(x, 0) = f_0^l(x)$ for $l = 1, \ldots, \ell$ and finally executing AVSS-Share-MS with $F^1(x, y), \ldots, F^\ell(x, y)$.

Through we do not elaborate, we now mention another application of AVSS-Share-MS which uses the above interpretation. We can design an Asynchronous BA (ABA) protocol with an amortized communication cost of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits for reaching agreement on a single bit. To the best of our knowledge, there is only one ABA with $4t+1$ due to [13] which requires very high communication complexity (though polynomial in n).

Remark 2. The best known AVSS of [2] requires an amortized cost $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits for sharing a single secret. Hence AVSS-Share-MS shows a clear improvement over the AVSS of [2].

Remark 3. The idea of hiding multiple secrets in a single polynomial was explored earlier in [12] in the context of passive adversary in synchronous network. Doing the same in asynchronous network, in the presence of active adversary is bit tricky and calls for new techniques. Though we can hide $(d+1-t)$ secrets in each degree-d polynomial $f_0^l(x)$ using protocol AVSS-Share, we hide only one secret, namely s^l in $f_0^l(x)$. This is because in our AMPC protocol, we require that each degree-d polynomial hides only one secret. However, hiding multiple secrets in a degree-d polynomial will be useful in the context of ABA.

5 Protocol for Generating $(t, 2t)$-Sharing

We now present a protocol called (t,2t)-Share-MS that allows a dealer $D \in \mathcal{P}$ to concurrently generate $(t, 2t)$-sharing of $\ell \geq 1$ secrets. We explain the idea of the protocol for a single secret s. D invokes AVSS-Share-MS to t-share s. Let $f(x)$ be the degree-t polynomial using which s is t-shared. D also invokes AVSS-Share-MS to $(2t-1)$-share a random value r. Let $g(x)$ be the degree-$(2t-1)$ polynomial using which r is $(2t-1)$-shared. It is easy to see that $h(x) = f(x) + xg(x)$ will be a degree-$2t$ polynomial, such that $h(0) = s$. So if party P_i locally computes $h(i) = f(i) + i \cdot g(i)$, then this will generate the $2t$-sharing of s. Protocol (t,2t)-Share-MS follows this principle for all the ℓ secrets concurrently.

Theorem 4. *Protocol (t,2t)-Share-MS satisfies the following properties:*

1. TERMINATION: *(a) If D is honest, then each honest P_i will terminate (t,2t)-Share-MS. (b) If D is corrupted and some honest P_i terminates (t,2t)-Share-MS, then all honest parties will also eventually terminate the protocol.*
2. CORRECTNESS: *If honest parties terminate (t,2t)-Share-MS, then there are ℓ values, that are $(t, 2t)$-shared among the parties in $\mathcal{P}$.*
3. SECRECY: *$\mathcal{A}_t$ will have no information about the secrets of an honest D.*
4. COMMUNICATION COMPLEXITY: *The protocol privately communicates $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits and A-cast $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.*

Proof. Follows from the properties of AVSS-Share-MS and protocol steps. □

Protocol (t,2t)-Share-MS($D, \mathcal{P}, S = (s^1, \ldots, s^\ell)$)

CODE FOR D:

1. Invoke AVSS-Share-MS$(D, \mathcal{P}, S = (s^1, \ldots, s^\ell), t)$ and AVSS-Share-MS$(D, \mathcal{P}, R = (r^1, \ldots, r^\ell), 2t-1)$, where the elements of R are random.

CODE FOR P_i:

1. Participate in AVSS-Share-MS$(D, \mathcal{P}, S = (s^1, \ldots, s^\ell), t)$ and AVSS-Share-MS$(D, \mathcal{P}, R = (r^1, \ldots, r^\ell), 2t-1)$. Wait to terminate AVSS-Share-MS$(D, \mathcal{P}, S, t)$ with i^{th} shares of $S = (s^1, \ldots, s^\ell)$, say $(\varphi_i^1, \ldots, \varphi_i^\ell)$. Wait to terminate AVSS-Share-MS$(D, \mathcal{P}, R, 2t-1)$ with i^{th} shares of $R = (r^1, \ldots, r^\ell)$, say $(\chi_i^1, \ldots, \chi_i^\ell)$.
2. For $l = 1, \ldots, \ell$, locally compute $\psi_i^l = \varphi_i^l + i \cdot \chi_i^l$, output φ_i^l and ψ_i^l as i^{th} share of s corresponding to t and $2t$-sharing respectively and terminate (t,2t)-Share-MS.

Remark 4. In [2] the authors presented a perfectly secure protocol, that privately communicates $\mathcal{O}(\ell n^3 \log |\mathbb{F}|)$ bits and A-casts $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits to generate $(t, 2t)$-sharing of ℓ secrets (For complete details, see [21]). Thus (t,2t)-Share-MS gains a factor of $\Omega(n)$ in communication complexity for generating $(t, 2t)$-sharing. In fact, it is this gain of $\Omega(n)$, which helps our AMPC protocol to gain $\Omega(n)$ in communication complexity, compared to the AMPC of [2]. In [20], a protocol with same communication complexity as ours is given for generating $(t, 2t)$-sharing. However, the protocol has negligible error probability in CORRECTNESS and TERMINATION.

6 AMPC Protocol Overview

Once we have an efficient protocol for generating $(t, 2t)$-sharing, our AMPC protocol proceeds in the same way as that of [2,20]. Specifically, our AMPC protocol is a sequence of three phases: preparation, input and computation. In the preparation phase, corresponding to each multiplication and random gate, a $(t, 2t)$-sharing of random secret will be generated. In the input phase the parties t-share their inputs and agree on a common set of at least $n - t$ parties who correctly t-shared their inputs. In the computation phase, based on the inputs of the parties in this common set, the actual circuit will be computed gate by gate, such that the output of the intermediate gates are always kept as secret and are t-shared among the parties. We now elaborate on each of the three phases.

6.1 Preparation Phase

The goal of protocol PreparationPhase is to generate $(t, 2t)$-sharing of $c_M + c_R$ random *secrets*. For this, each individual party acts as a dealer and $(t, 2t)$-share $\frac{c_M+c_R}{n-2t}$ random values. Then an instance of ACS protocol is executed to agree on a common set C of $n-t$ parties, who have correctly $(t, 2t)$-shared $\frac{c_M+c_R}{n-2t}$ values. Out of these $n-t$ parties, at least $n-2t$ are honest, who have indeed $(t, 2t)$-shared random values, which are unknown to $\mathcal{A}_t$. So if we consider the $(t, 2t)$-sharing done by the honest parties (each of them has done $\frac{c_M+c_R}{n-2t}$ $(t, 2t)$-sharing) in common set C, then we will get $\frac{c_M+c_R}{n-2t} * (n-2t) = c_M + c_R$ random $(t, 2t)$-sharing. For this, we use *Vandermonde Matrix* [11] and its ability to extract randomness which has been exploited in [11,2]. The same approach is also used in the preparation phase of [20]. Due to space constraint, we present the protocol in [21] and state only the following lemma:

Lemma 6. *Each honest party will eventually terminate PreparationPhase. The protocol generates $(t, 2t)$-sharing of $c_M + c_R$ secret random values, unknown to $\mathcal{A}_t$, by privately communicating $O((c_M + c_R)n^2 \log |\mathbb{F}|)$ bits, A-casting $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits and executing one instance of ACS.*

6.2 Input Phase

In protocol InputPhase, each party acts as a dealer and t-share his input(s) by executing an instance of AVSS-Share-MS. The parties then execute ACS to agree on a common set C of $n-t$ parties, whose instances of AVSS-Share-MS have terminated. As the input(s) of the parties in C will be considered for computation (of the circuit), each party considers the t-sharing of all the inputs shared by parties, only in C. As the protocol is very straight forward, we present it in [21] and state the following lemma:

Lemma 7. *Each honest party terminates InputPhase. The protocol generates t-sharing of the inputs of the parties in C, such that $\mathcal{A}_t$ has no information about the inputs of the honest parties in C, by privately communicating $O(c_I n^2 \log |\mathbb{F}|)$ bits, A-casting $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits and executing one ACS.*

6.3 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 used t-sharing, linear gates can be computed locally simply by applying the linear function to the shares. With every random gate, one random $(t, 2t)$-sharing (from the preparation phase) is associated, whose t-sharing is directly used as outcome of the random gate. With every multiplication gate, one random $(t, 2t)$-sharing (from the preparation phase) is associated, which is then used to compute t-sharing of the product, following the technique of [11]: Let $z = xy$, where x, y

are the inputs of the multiplication gate, where x, y are t-shared, i.e. $[x]_t, [y]_t$. Moreover, let $[r]_{(t,2t)}$ be the $(t, 2t)$-sharing associated with the multiplication gate, where r is a secret random value. For computing $[z]_t$, the parties compute $[\Lambda]_{2t} = [x]_t.[y]_t + [r]_{2t}$. Then Λ is privately reconstructed by every $P_i \in \mathcal{P}$. Now every party defines $[\Lambda]_t$ as the default sharing of Λ and computes $[z]_t = [\Lambda]_t - [r]_t$. The secrecy of z follows from [11,2]. The same approach is also used in the computation phase of AMPC protocol of [2,20]. Due to space constraint, we present the protocol in [21] and state the following lemma:

Lemma 8. *Each honest party will eventually terminate* ComputationPhase. *Given $(t, 2t)$-sharing of $c_M + c_R$ secret random values, the protocol securely evaluates the circuit by privately communicating $O((c_M n^2 + c_O n) \log |\mathbb{F}|)$ bits.*

6.4 Final AMPC Protocol

Now our new AMPC protocol called AMPC for evaluating function f is: (1). Invoke PreparationPhase (2). Invoke InputPhase (3). Invoke ComputationPhase.

Theorem 5. *Protocol AMPC is an optimally resilient, perfectly secure AMPC protocol that privately communicates $\mathcal{O}(((c_I + c_M + c_R)n^2 + c_O n) \log |\mathbb{F}|)$ bits, A-casts $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits and requires 2 invocations to ACS. Each honest party will eventually terminate AMPC.*

Acknowledgement. We would sincerely like to thank Tal Rabin for giving several insightful remarks on the earlier version of this paper which significantly improved the presentation of the protocols.

References

1. Beerliová-Trubíniová, Z., Hirt, M.: Efficient multi-party computation with dispute control. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 305–328. Springer, Heidelberg (2006)
2. Beerliová-Trubíniová, Z., Hirt, M.: Simple and efficient perfectly-secure asynchronous MPC. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 376–392. Springer, Heidelberg (2007)
3. Beerliová-Trubíniová, Z., Hirt, M.: Perfectly-secure MPC with linear communication complexity. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 213–230. Springer, Heidelberg (2008)
4. Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous secure computation. In: STOC, pp. 52–61 (1993)
5. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, pp. 1–10 (1988)
6. BenOr, M., Kelmer, B., Rabin, T.: Asynchronous secure computations with optimal resilience. In: PODC, pp. 183–192 (1994)
7. Bracha, G.: An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. In: PODC, pp. 154–162 (1984)
8. Canetti, R.: Studies in Secure Multiparty Computation and Applications. PhD thesis, Weizmann Institute, Israel (1995)

9. Canetti, R., Rabin, T.: Fast asynchronous Byzantine Agreement with optimal resilience. In: STOC, pp. 42–51 (1993)
10. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC, pp. 11–19 (1988)
11. Damgård, I., Nielsen, J.B.: Scalable and unconditionally secure multiparty computation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 572–590. Springer, Heidelberg (2007)
12. Franklin, M.K., Yung, M.: Communication complexity of secure computation. In: STOC, pp. 699–710 (1992)
13. Feldman, P., Micali, S.: An optimal algorithm for synchronous Byzantine Agreemet. In: STOC, pp. 639–648 (1988)
14. Fitzi, M., Garay, J., Gollakota, S., Pandu Rangan, C., Srinathan, K.: Round-optimal and efficient verifiable secret sharing. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 329–342. Springer, Heidelberg (2006)
15. Gennaro, R., Ishai, Y., Kushilevitz, E., Rabin, T.: The round complexity of verifiable secret sharing and secure multicast. In: STOC, pp. 580–589 (2001)
16. Katz, J., Koo, C., Kumaresan, R.: Improving the round complexity of VSS in point-to-point networks. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 499–510. Springer, Heidelberg (2008)
17. Patra, A., Choudhary, A., Rabin, T., Pandu Rangan, C.: The round complexity of verifiable secret sharing revisited. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 487–504. Springer, Heidelberg (2009)
18. Patra, A., Choudhary, A., Pandu Rangan, C.: Efficient asynchronous multiparty computation with optimal resilience. Cryptology ePrint Archive, Report 2008/425 (2008)
19. Patra, A., Choudhary, A., Pandu Rangan, C.: Efficient asynchronous Byzantine Agreement with optimal resilience. In: PODC, pp. 92–101 (2009)
20. Patra, A., Choudhary, A., Pandu Rangan, C.: Unconditionally secure asynchronous multiparty computation with quadratic communication per multiplication gate. Cryptology ePrint Archive, Report 2009/087 (2009)
21. Patra, A., Choudhary, A., Pandu Rangan, C.: Communication Efficient Perfectly Secure VSS and MPC in Asynchronous Networks with Optimal Resilience Cryptology ePrint Archive, Report 2010/007 (2010)
22. Prabhu, B., Srinathan, K., Pandu Rangan, C.: Trading players for efficiency in unconditional multiparty computation. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 342–353. Springer, Heidelberg (2003)
23. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority. In: STOC, pp. 73–85 (1989)
24. Srinathan, K., Pandu Rangan, C.: Efficient asynchronous secure multiparty distributed computation. In: Roy, B., Okamoto, E. (eds.) INDOCRYPT 2000. LNCS, vol. 1977, pp. 117–129. Springer, Heidelberg (2000)
25. Yao, A.C.: Protocols for secure computations. In: FOCS, pp. 160–164 (1982)