Efficient Asynchronous Verifiable Secret Sharing and Multiparty Computation *

Arpita Patra[1] Ashish Choudhury[2] † C. Pandu Rangan[3]

[1]Department of Computer Science
ETH Zurich, Switzerland
arpita.patra@inf.ethz.ch, arpitapatra10@gmail.com

[2]Department of Computer Science
University of Bristol, United Kingdom
Ashish.Choudhary@bristol.ac.uk, partho31@gmail.com

[3]Department of Computer Science and Engineering
Indian Institute of Technology Madras, India
rangan@cse.iitm.ac.in, prangan55@gmail.com

Abstract

Secure Multi-Party Computation (MPC) providing information theoretic security allows a set of n parties to securely compute an agreed function $\mathcal{F}$ over a finite field $\mathbb{F}$, even if t parties are under the control of a computationally unbounded active adversary. Asynchronous MPC (AMPC) is an important variant of MPC, which works over an asynchronous network. It is well known that perfect AMPC is possible if and only if $n \geq 4t+1$, while statistical AMPC is possible if and only if $n \geq 3t+1$. In this paper, we study the communication complexity of AMPC protocols (both statistical and perfect) designed with exactly $n = 4t+1$ parties. Our major contributions in this paper are as follows:

1. Asynchronous Verifiable Secret Sharing (AVSS) is one of the main building blocks for AMPC. In this paper, we design two AVSS protocols with $4t+1$ parties: the first one is statistically secure and has non-optimal resilience, while the second one is perfectly secure and has optimal resilience. Both these schemes achieve a common interesting property, which was not achieved by the previous schemes. Specifically, our AVSS schemes allow to share a secret through a polynomial of degree at most d, where $t \leq d \leq 2t$. In contrast, the existing AVSS schemes can share a secret only through a polynomial of degree at most t. The new property of our AVSS simplifies the degree reduction step for the evaluation of multiplication gates in an AMPC protocol.
2. Using our statistical AVSS, we design a statistical AMPC protocol with $n = 4t+1$ which communicates $\mathcal{O}(n^2)$ field elements per multiplication gate. Though this protocol has non-optimal resilience, it significantly improves the communication complexity of the existing statistical AMPC protocols.
3. We then present a perfect AMPC protocol with $n = 4t+1$ (using our perfect AVSS scheme), which also communicates $\mathcal{O}(n^2)$ field elements per multiplication gate. This protocol improves on our statistical AMPC protocol as it has optimal resilience. To the best of our knowledge, this is the most communication efficient perfect AMPC protocol in the information theoretic setting.

*A preliminary version of this paper appeared in [39].

†The work in this paper was partially supported by EPSRC via grant EP/I03126X/1, and by the European Commission through the ICT Programme under Contract ICT2007216676 ECRYPT II.

1 Introduction

Threshold Multi-Party Computation (MPC) [45, 29, 17, 9, 42] allows a set of n mutually distrusting parties to securely compute an agreed function $\mathcal{F}$ over a finite field $\mathbb{F}$, even if t out of the n parties are under the control of a computationally unbounded active adversary $\mathcal{A}_t$. MPC is one of the most important and fundamental problems in secure distributed computing. Over the past three decades, the problem has been studied extensively in different settings [45, 29, 9, 17, 41, 2, 23, 6, 8, 10, 5]. In any general MPC protocol, the function $\mathcal{F}$ is expressed as an arithmetic circuit over $\mathbb{F}$, consisting of input, linear (e.g. addition), multiplication, random and output gates over $\mathbb{F}$. The evaluation of multiplication gates require the maximum communication among the parties and so the communication complexity of any general MPC protocol is usually expressed in terms of the communication complexity per multiplication gate.

The MPC problem has been studied extensively over the synchronous networks where it is assumed that there exists a global clock and the delay of any message in the network is bounded. However, though theoretically impressive, such networks do not model adequately real life networks like the Internet. Thus a new line of research was initiated and dedicated for MPC in the asynchronous networks [8, 44, 5, 10, 38], where the messages are allowed to be delayed arbitrarily.

Unlike synchronous MPC protocols, designing asynchronous MPC (AMPC) protocols has received less attention due to their inherent difficulty. Roughly speaking, the main difficulty in designing asynchronous protocols is that we cannot distinguish between a slow but honest party, whose messages are delayed in the network and a corrupted party, who did not send messages at all. Due to this, at any stage of an asynchronous protocol, no party can afford to wait to receive the communication from all the n parties and so the communication from t (potentially slow but honest) parties may have to be ignored. In this paper, our focus is on the protocols achieving information theoretic security (that is security against a computationally unbounded adversary). Such AMPC protocols can be categorized mainly into two types:

1. *Perfectly Secure AMPC or Perfect AMPC*: The protocols of this type do not involve any error in the computation. In [8], it is shown that perfectly secure AMPC is possible if and only if $n \geq 4t+1$. Thus any perfectly secure AMPC protocol designed with exactly $n = 4t+1$ parties is said to be an optimally resilient perfectly secure AMPC protocol. Optimally resilient, perfectly secure AMPC protocols are reported in [8, 44, 5]. Among these, the AMPC protocol of [5] provides the best communication complexity, which is $\mathcal{O}(n^3 \log(|\mathbb{F}|))$ bits per multiplication gate, where the computation is done over a finite field $\mathbb{F}$, such that $|\mathbb{F}| > n$.

2. *Statistically Secure AMPC or Statistical AMPC*: The protocols of this type can incur a negligible error (specified by a given error parameter ϵ) in the computation. From [10], it is known that statistically secure AMPC is possible if and only if $n \geq 3t + 1$. Thus any statistical AMPC protocol designed with exactly $n = 3t + 1$ parties is said to be an optimally resilient statistically secure AMPC protocol. Optimally resilient, statistically secure AMPC protocols are reported in [10, 38]. Among these, the AMPC protocol of [38] provides the best communication complexity of $\mathcal{O}(n^5 \log |\mathbb{F}|)$ bits per multiplication gate, where the protocol works over a field $\mathbb{F} = GF(q)$, where $q > \max(n, 2^{\kappa})$, such that $\kappa = \log \frac{1}{\epsilon}$.

From the above discussion, we find that optimally resilient statistical AMPC protocols require higher communication in comparison to their perfect counterpart. This is quite intriguing because it is easier to design protocols that involve negligible error, in comparison to the error free protocols. There are two reasons behind this anomaly: First, the corruption threshold is different for statistical and perfect protocols. Namely, perfect protocols can only tolerate $t < n/4$ corruptions, while in comparison, statistical protocols have to tolerate more corruptions, namely $t < n/3$. It is well-known that asynchronous verifiable secret sharing (AVSS) is a major building block used in the design of information theoretically secure AMPC protocols.

The second reason for the anomaly stems from the difficulty in designing statistical AVSS with $n = 3t + 1$ parties, whose communication complexity matches the communication complexity of perfect AVSS protocols with $n = 4t + 1$. An excellent informal description of this difficulty is outlined in [16].

An interesting approach used to obtain a communication efficient statistical AMPC is to trade the resilience for efficiency. That is, to design communication efficient statistical AMPC protocols tolerating a smaller number of corruptions. This approach is not new as it has been used earlier in the synchronous settings to achieve efficiency (see for example [22, 23]). In the asynchronous settings, this approach was reported in [40], where the authors presented a statistical AMPC protocol with $n = 4t+1$. Most recently, in [31], the authors presented a statistical AMPC protocol with $n = 4t+1$ (we will show later that this protocol is flawed). The communication complexity (per multiplication gate) of the best known AMPC protocols is summarized in Table 1.

Table 1: Communication complexity (CC) in bits per multiplication gate of the known AMPC protocols. For the perfect protocols $|\mathbb{F}| > n$, while for the statistical protocols $\mathbb{F} = GF(q)$, where $q > \max(n, 2^\kappa)$, such that $\kappa = \log \frac{1}{\epsilon}$

Reference	Type	Resilience	CC in bits
[8, 15]	Perfect	$t < n/4$ (optimal)	$\mathcal{O}(n^6 \log \lvert\mathbb{F}\rvert)$
[44]	Perfect	$t < n/4$ (optimal)	$\Omega(n^5 \log \lvert\mathbb{F}\rvert)$
[5]	Perfect	$t < n/4$ (optimal)	$\mathcal{O}(n^3 \log \lvert\mathbb{F}\rvert)$
[10]	Statistical	$t < n/3$ (optimal)	$\Omega(n^{11} (\log \lvert\mathbb{F}\rvert)^4)$
[38]	Statistical	$t < n/3$ (optimal)	$\mathcal{O}(n^5 \log \lvert\mathbb{F}\rvert)$
[40]	Statistical	$t < n/4$ (non-optimal)	$\mathcal{O}(n^4 \log \lvert\mathbb{F}\rvert)$
[31]	Statistical	$t < n/4$ (non-optimal)	$\mathcal{O}(n^2 \log \lvert\mathbb{F}\rvert)$
This article	Statistical	$t < n/4$ (non-optimal)	$\mathcal{O}(n^2 \log \lvert\mathbb{F}\rvert)$
This article	Perfect	$t < n/4$ (optimal)	$\mathcal{O}(n^2 \log \lvert\mathbb{F}\rvert)$

Recently in [7], communication efficient MPC protocols over networks that exhibit partial asynchrony were presented. Such networks are synchronous up to a "certain point" and then behave in a completely asynchronous way afterwards. In another work, Damgård et al. [21] have reported an efficient MPC protocol over a network that assumes the concept of a "synchronization point"; i.e. the network is asynchronous before and after the synchronization point. We will not consider the protocols of [7] and [21] for further discussion as they are not designed in a completely asynchronous setting which we consider in this article.

1.1 Our Contributions for AMPC

In this paper our focus is on AMPC with $4t + 1$ parties. Our main contributions are as follows:

1. From Table 1, we find that the most communication efficient statistical AMPC protocol is due to [31]. However, we show that this protocol is flawed.

2. We design a new statistically secure AMPC protocol with $n = 4t+1$, which communicates $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits per multiplication gate. Our protocol achieves its goal without using the player elimination framework of [30], which is used in [31]. We note that our statistical AMPC protocol has non-optimal resilience.

3. We present a perfectly secure AMPC protocol with $n = 4t+1$ which communicates $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits per multiplication gate. Our perfect AMPC protocol has optimal resilience. From Table 1, the best known perfect AMPC with optimal resilience [5] communicates $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits per multiplication.

Hence our AMPC protocol provides the best communication complexity among all the known AMPC protocols.

To achieve the above AMPC protocols, we present two AVSS schemes with $n = 4t + 1$: the first one is statistically secure (has non optimal resilience and is used in our statistical AMPC), while the second one is perfectly secure (has optimal resilience and is used in our perfect AMPC). Both these AVSS schemes achieve some interesting properties, which are not achieved by the previous schemes. Despite of the fact that both the statistical and perfect AVSS protocol achieve the same communication complexity, we decided to present them due to the fact that they employ completely different techniques. In the next section we informally discuss about AVSS and the properties achieved by our AVSS schemes.

1.2 Verifiable Secret Sharing (VSS)

Verifiable Secret Sharing (VSS) is one of the fundamental building blocks for many secure distributed computing tasks, such as MPC, Byzantine Agreement (BA) [25, 16, 33, 1, 37], etc. Any VSS scheme consists of two phases: the sharing phase and the reconstruction phase and is implemented by a pair of protocols (Sh,Rec). Here Sh is the protocol for the sharing phase, while Rec is the protocol for the reconstruction phase. Protocol Sh allows a special party called the dealer (denoted as D), to share a secret s among a set of n mutually distrusting parties in a way that later allows for a unique reconstruction of s by every party using the protocol Rec. Moreover, if D is honest, then the secrecy of s is preserved till the end of Sh.

Over the last three decades, active research has been carried out in this area and many interesting and significant results have been obtained dealing with high efficiency, security against general adversaries, security against mixed type of corruptions, long-term security, provable security, etc (see [18, 24, 9, 17, 41, 20, 10, 16, 42, 26, 28, 32, 35, 4, 6] and their references). However, almost all these solutions are for the synchronous model, where it is assumed that every message in the network is delayed by a given constant. This assumption is very strong because a single delayed message can completely break the overall security of the protocol. Therefore, VSS protocols for the synchronous model are not well suited for use in the real world networks. Hence a new line of research on VSS over the asynchronous networks was initiated. VSS protocols that are designed to work over the asynchronous networks are called Asynchronous VSS or AVSS.

We are interested in AVSS schemes for the threshold access structure. Informally, such an AVSS scheme shares the secret in a way that any set of t or less parties does not get any information about the secret (in the information theoretic sense) from their shares, while any set of $t+1$ or more (correct) shares are enough to reconstruct the secret. Moreover, we want the scheme to be linear, meaning that the shares are computed as a linear function of the secret and the associated randomness. Such an AVSS scheme is what is typically used in the information theoretically secure AMPC protocols, as it allows the parties to locally perform any linear computation on shared values.

Information theoretically secure AVSS (for the threshold access structures) can be categorized into two classes:

1. *Perfectly Secure AVSS or Perfect AVSS:* A scheme of this type satisfies the properties of an AVSS without any error. Perfectly secure AVSS is possible if and only if $n \geq 4t + 1$ [8, 15]. Hence, we call any perfectly secure AVSS protocol designed with exactly $n = 4t+1$ parties as an optimally resilient, perfectly secure AVSS protocol. Such AVSS protocols are proposed in [8, 15, 5].

2. *Statistically Secure AVSS or Statistical AVSS:* A scheme of this type satisfies the properties of an AVSS except with a negligible error (specified by a given error parameter ϵ). Statistical AVSS is possible if and only if $n \geq 3t + 1$ [16, 10]. To the best of our knowledge, the AVSS protocols of [16, 10, 37, 38] are the only known optimally resilient statistical AVSS protocols (i.e. with $n = 3t + 1$).

The AVSS schemes based on the polynomial interpolation are the most popular ones and they have been used for the AMPC protocols in the literature (all the papers referred above follow the polynomial based implementation). Such schemes are linear and allow to share a secret using polynomials. In the rest of the paper, we consider AVSS schemes with polynomial based implementation. Before we discuss about our AVSS schemes, the new properties that they achieve and how the newly attained properties bring efficiency in evaluating the multiplication gates in the AMPC protocol, it is important to see how AVSS schemes are used in the AMPC protocols. Therefore we dedicate the rest of this section to describe how AVSS serves in AMPC protocols.

The AVSS schemes (based on polynomials) are one of the important building blocks for designing information theoretically secure AMPC protocols. The sharing phase of such an AVSS scheme enforces the dealer to t-share a value (even if the dealer is corrupted). Informally, an element v is said to be d-shared among n parties $P_1, \ldots, P_n$, if there exists a polynomial $f(x)$ of degree at most d such that $f(0) = v$ and every (honest) party P_i has a share $Sh_i = f(i)$ of v. We denote such a sharing by $[v]_d$. The AVSS schemes are used in the AMPC protocols at two places: first to make the parties commit and share their inputs and second to generate sharings of random values (satisfying some conditions) which are used to evaluate the multiplication gates of the circuit. The general approach followed in the AMPC protocols is that every party P_i first t-share his input x_i, where x_i is P_i's input for the computation. Then the parties agree on a set of $(n-t)$ parties, denoted as C, such that $[x_i]_t$ has been generated for every $P_i \in C$ (in any AMPC protocol, the inputs of all the n parties cannot be considered for the computation due to the asynchronous nature of the network, as it may result in infinite waiting). The input x_i of each honest party $P_i \in C$ remains information theoretically secure because for every such x_i, the adversary obtains at most t shares.

Once the set C is agreed upon, the computation of the function $\mathcal{F}$ is performed gate-by-gate in a shared fashion, following the classical approach of [9]. More specifically, the parties interact according to the protocol to generate t-sharing of the output of each gate from t-sharing of the input(s) of the gate. Once t-sharing of the final output is generated, the parties reveal their output shares and an error correction mechanism is applied to identify the corrupted shares and the final output is robustly reconstructed. The robust reconstruction is guaranteed due to the fact that AMPC demands at least $3t+1$ parties and the reconstruction of a t-shared value with at least $3t+1$ parties is robust. Intuitively, the secrecy of the entire computation is preserved, as each intermediate value in the computation remains t-shared.

In more detail, the shared computation of the circuit is done in the following fashion: the linear gates, for example, the addition gates, can be evaluated locally by the parties, without any interaction, due to the linearity property of t-sharing. More specifically, given $[c]_t$ and $[d]_t$, the t-sharing of $e = (c + d)$ can be locally generated as $[e]_t = [c]_t + [d]_t$. However, the multiplication gates cannot be evaluated locally, as $[c] \cdot [d]_t = [e]_{2t}$, instead of $[e]_t$. If $[e]_{2t}$ is not converted to $[e]_t$ then further multiplication of e with another t-shared value will raise the degree of the sharing, which makes it impossible to robustly reconstruct the value. So the major bottleneck in the shared evaluation of the circuit is to evaluate the multiplication gates. To generate $[e]_t$ from $[c]_t$ and $[d]_t$ (where $e = c \cdot d$), the parties have to interact with each other. The amount of interaction varies from protocol to protocol and actually depends upon the method used to reduce the degree of the sharing of e from $2t$ to t. And this is why, the communication complexity of any MPC/AMPC protocol is usually expressed in terms of the communication done to evaluate a single multiplication gate.

The most common method to generate $[e]_t$ from $[c]_t$ and $[d]_t$ is the well known Beaver's circuit randomization method [3], where the multiplication gates are evaluated using t-sharing of pre-computed random multiplication triples (which can be generated in a pre-processing stage, prior to the beginning of the computation). This approach is used in al most all the MPC schemes (both synchronous as well as asynchronous) proposed in the recent years ([4, 23, 6]). An alternative of the above mentioned approach proposed in [5] and used by us in this paper is to evaluate the multiplication gates using pre-computed $(t, 2t)$-sharing of random values. A $(t, 2t)$-sharing [5] of a value $r \in \mathbb{F}$ consists of a t-sharing and a $2t$-sharing of r using independent polynomials of degree at most t and $2t$ respectively. So both $[r]_t$, as well as $[r]_{2t}$ will be available to the parties. Given $(t, 2t)$-sharing of a pre-computed random value r, the parties can generate $[e]_t$ from $[c]_t$ and $[d]_t$ as follows: the parties first locally generate $[e]_{2t} = [c]_t \cdot [d]_t$ and then $[\delta]_{2t} = [e]_{2t} - [r]_{2t}$. The later computation follows from the linearity property of $2t$-sharing. The above step is followed by the robust reconstruction of δ, which is possible with $n = 4t + 1$. Notice that reconstructing δ does not compromise the secrecy of e, c and d because r is random. Once δ is publicly known, the parties can locally generate $[e]_t = [\delta]_t + [r]_t$ (the parties consider a default t-sharing of δ using the constant polynomial of degree 0).

So the problem of efficiently evaluating the multiplication gates boils down to the problem of either efficiently generating $(t, 2t)$-sharing of random values or t-sharing of random triples. The evaluation cost of a multiplication gate in both the approaches is nearly the same. For the triple based approach, it requires the reconstruction of two t-shared values (see [3]), while for the $(t, 2t)$-sharing based approach, it requires the reconstruction of a single $2t$-shared value. We note that the triple based approach is robust when $n \geq 3t + 1$ (as it requires robustly reconstructing t-shared values). However, $n \geq 4t + 1$ is required to make the $(t, 2t)$-sharing based approach to be robust (as it requires robustly reconstructing $2t$-shared values). Since we deal with $n = 4t + 1$, we attack the problem of efficiently evaluating the multiplication gates by asking how efficiently we can generate a $(t, 2t)$-sharing. In what follows, we show how the existing AVSS schemes have been used to generate $(t, 2t)$-sharing of a random value and how the AVSS schemes introduced in this article allow us to achieve the same goal with more efficiency.

In [5], an approach to generate $(t, 2t)$-sharing of a random value from t-sharing of $(3t + 1)$ random values has been described. The t-sharing of a value can be generated by using any existing AVSS scheme. Thus the existing approach of generating a $(t, 2t)$-sharing requires t-sharing of $(3t+1)$ values which in turn requires to invoke AVSS $(3t + 1)$ times. We bring down the complexity of generating a $(t, 2t)$-sharing by a factor of n by noting that a $(t, 2t)$-sharing can be generated from a t-sharing and a $(2t - 1)$-sharing (more on this in the sequel) and by introducing AVSS schemes that can produce a d-sharing for any given d, where $t \leq d \leq 2t$. We emphasize that prior to our work, there was no AVSS scheme to produce a d-sharing, for any given d, where $d > t$. Our AVSS schemes not only achieve this new property, but it does so with the same communication complexity as the best known existing AVSS scheme of [5], which generates t-sharing.

Our Contributions for AVSS: We present two AVSS schemes with $4t + 1$ parties; one is statistically secure (non-optimally resilient) and the other one is perfectly secure (optimally resilient). Both these schemes have an interesting property: the sharing phase of these schemes allows the dealer (possibly corrupted) to d-share a value v, for a given d, where $t \leq d \leq 2t$. More specifically, given a value $v \in \mathbb{F}$ [1] to be shared and a given degree d for sharing v, where $t \leq d \leq 2t$, at the end of the sharing phase, there will exist a polynomial over $\mathbb{F}$, say $f(x)$, of degree at most d, such that $f(0) = v$ and every honest P_i will possess a share $Sh_i = f(i)$ of v. Moreover, we also enhance our basic AVSS schemes that generate d-sharing of a single value and make them generate d-sharing of several values (specifically ℓ values, where $\ell \geq 1$) simultaneously, such that each individual value is d-shared. The advantage of the enhanced schemes that give simultaneous sharing of several values over several instances of the basic schemes for individual generation of the sharings is that the former allows us to combine the broadcast (public) communication for all the values and therefore the broadcast communication remains independent of the number of values shared. This is important since implementing broadcast by a protocol in the asynchronous settings [14] is expensive and we must keep the broadcast communication independent of the the number of values to be shared. In the paper, we first present the basic AVSS schemes for simplicity and later enhance them for multiple values. Subsequently, we use the enhanced schemes to efficiently generate $(t, 2t)$-sharing of several values simultaneously in an amortized sense (the details will be available later). Table 2 gives a comparison of our AVSS schemes with the existing AVSS schemes (with $4t + 1$ parties) in the literature.

Now we highlight how our AVSS schemes can be looked at from two different perspective, the first one

[1] For the perfect scheme $|\mathbb{F}| > n$, while for the statistical scheme $\mathbb{F} = GF(q)$, where $q > \max{(n, 2^{\kappa})}$, such that $\kappa = \log \frac{1}{\epsilon}$.

Table 2: Comparison of our AVSS protocols with the existing AVSS protocols designed with $4t+1$ parties.

Reference	Type	Number of Secrets Shared	Degree of the Sharing	Communication Complexity In Bits
[8]	Perfect	1	Only t-sharing	$\mathcal{O}(n^3 \log(\lvert\mathbb{F}\rvert))$
[5]	Perfect	ℓ, where $\ell \geq 1$	Only t-sharing	$\mathcal{O}(\ell n^2 \log(\lvert\mathbb{F}\rvert))$
This article	Statistical	ℓ, where $\ell \geq 1$	d-sharing, for any given d, where $t \leq d \leq 2t$	$\mathcal{O}(\ell n^2 \log(\lvert\mathbb{F}\rvert))$
This article	Perfect	ℓ, where $\ell \geq 1$	d-sharing, for any given d, where $t \leq d \leq 2t$	$\mathcal{O}(\ell n^2 \log(\lvert\mathbb{F}\rvert))$

being the way we presented them so far: **(a).** They generate d-sharing of ℓ values where $\ell \geq 1$ at the cost of $\mathcal{O}(\ell n^2 \log(|\mathbb{F}|))$ bits; **(b).** They share $\ell(d-t+1)$ values in the sense of "packed secret sharing" [27] at the cost of $\mathcal{O}(\ell n^2 \log(|\mathbb{F}|))$ bits for $\ell \geq 1$. The two different perspectives have two different implication. The first perspective allows us to design a method for generating $(t, 2t)$-sharing of random values with cheaper cost than the existing method of [5]. The second perspective implies that the amortized cost of sharing a single value tolerating malicious adversaries is $\mathcal{O}(n)$ field elements which matches the complexity of sharing a single value tolerating a passive adversary (e.g. consider the Shamir secret sharing [43]). For designing our AMPC protocols, we use the first perspective of our AVSS schemes. We elaborate more in the following:

1. *Efficient generation of* $(t, 2t)$*-sharing of random values*: We start with the method of [5] to generate $(t, 2t)$-sharing of a single random value from t-sharing of $3t+1$ random values. Let $r^0, r^1, \ldots, r^{3t}$ be the $3t+1$ random values which are t-shared. Then consider the polynomials $P(x) = r^0 + r^1 \cdot x + \ldots + r^t \cdot x^t$ and $Q(x) = r^0 + r^{t+1} \cdot x + \ldots + r^{3t} \cdot x^{2t}$ of degree at most t and $2t$ respectively. It is easy to see that $[r^0]_t$ using $P(x)$ and $[r^0]_{2t}$ using $Q(x)$ gives a $(t, 2t)$-sharing of r^0 because $P(0) = Q(0) = r^0$. Both $[r^0]_{2t}$ and $[r^0]_t$ can be computed given $[r^0]_t, [r^1]_t, \ldots, [r^{3t}]_t$. To obtain t-sharing of $3t+1$ random values, that is, $[r^0]_t, [r^1]_t, \ldots, [r^{3t}]_t$, each party in [5] is asked to act as a dealer and t-share $(3t+1)$ random values. This step is followed by some additional "randomness extraction" steps. Using the AVSS scheme of [5], this costs $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits for one party (by substituting $\ell = (3t+1)$ and $t = \Theta(n)$ in the second row of Table 2) and $\mathcal{O}(n^4 \log |\mathbb{F}|)$ bits[2] for n parties.

   On the other hand, we generate $(t, 2t)$-sharing of a random value from t-sharing of a random value and $(2t-1)$-sharing of a random value. Specifically, assume that we are given $[r]_t$ for a random value r and $[s]_{2t-1}$ for another random value s. Moreover, let $f(x)$ and $g(x)$ be the polynomials of degree at most t and $2t-1$ respectively that define $[r]_t$ and $[s]_{2t-1}$ respectively. It is easy to note that $[r]_{2t}$ can be obtained using the polynomial $h(x) = f(x) + x \cdot g(x)$ of degree at most $2t$. Every party can locally compute its share of $[r]_{2t}$ by computing $h(i) = f(i) + i \cdot g(i)$. This gives us a $(t, 2t)$-sharing of r. To obtain $[r]_t$ and $[s]_{2t-1}$ for a random r and s, we ask every party to act as a dealer and invoke two instances of our AVSS scheme to generate a t-sharing and a $(2t-1)$-sharing of random values. This step is followed by some additional randomness extraction steps. This costs $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits for one party (by substituting $\ell = 1, d = t$ and $\ell = 1, d = (2t-1)$ in the last two rows of Table 2) and $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits [3] for n parties. Thus, we note a reduction of $\Theta(n)$ over [5]. This saving of $\Theta(n)$ further allows our AMPC protocols to gain $\Theta(n)$ in the communication complexity over the AMPC protocol of [5]. We stress that the gain of $\Theta(n)$ is not merely due to our different way of generating a

[2] In [5], this cost is reduced by a factor of n by using additional tricks.

[3] This cost is further reduced by a factor of n by using additional tricks as in [5]. The details will be presented later.

$(t, 2t)$-sharing. The approach used by us is not applicable to [5] because neither the AVSS of [5], nor any prior AVSS can be used to generate $(2t-1)$-sharing of a value.

2. *Packed secret sharing in the asynchronous settings*: Our AVSS schemes allow the dealer to share a value using a polynomial of degree d, where d can be at most $2t$. If the dealer is honest then at most t points on the polynomial will be known to the adversary. Intuitively, this implies that from the view point of the adversary, $(d-t)+1$ coefficients of the polynomial are "free" and hence secure in the information theoretic sense. This further implies that using a single polynomial of degree at most d, an honest dealer can share $(d-t)+1$ secrets. This is the reminiscent of packed secret sharing, introduced in [27] for the synchronous settings. Our constructions provide packed secret sharing schemes in the asynchronous settings for the first time in the literature. We show that using our packed secret sharing, the amortized cost of sharing a single element from $\mathbb{F}$ is $\mathcal{O}(n)$ field elements, even in the presence of an active adversary. This matches the cost of sharing a single element from $\mathbb{F}$ in the presence of a passive adversary (for example, the Shamir secret sharing scheme [43]).

   Our schemes are useful in applications where a party needs to share multiple values. For example, *common coin* [15] is known as an important primitive for unconditionally secure asynchronous Byzantine Agreement (ABA) protocols. In a common coin protocol, every party needs to share/commit n values. In the existing common coin protocols, a party does so by invoking n instances of a secret sharing protocol (specifically, an AVSS protocol). Using our packed secret sharing, a party can share n values using $\ell = n/(d-t+1)$ polynomials, each of degree at most d (through a single polynomial, the party can share $(d-t+1)$ values). Substituting the maximum value of $d = 2t$ and using the fact that $t = \Theta(n)$, we find that $\ell = n/(d-t+1) = \mathcal{O}(1)$. This implies that each party can now share n values by invoking a single instance of AVSS by setting $\ell = n/(d-t+1)$. This overall reduces the communication complexity of the ABA protocol[4].

We conclude this section with a brief comparison of our proposed AVSS schemes and from now onwards, we focus on the first perspective of our AVSS schemes.

Comparison of the two AVSS Protocols: In this paper, we present two AVSS schemes that share the following common properties:

1. Designed with $n = 4t+1$;

2. Generates d-sharing of a value for any given d, where $t \leq d \leq 2t$.

3. Have the same communication complexity of $\mathcal{O}(\ell n^2 \log(|\mathbb{F}|))$ bits for sharing ℓ values.

However, our first AVSS scheme is statistical (thus has non-optimal resilience) while the second one is perfect (thus has optimal resilience). Technique wise, both the schemes are completely independent. We further believe that some of the techniques may lead to an improved AVSS scheme that will allow achieving linear communication complexity per multiplication gate in an AMPC protocol. Once we have a statistical/perfect AVSS scheme that generates d-sharing for any $t \leq d \leq 2t$, we can obtain a statistical/perfect AMPC by using the approach outlined earlier. It is the underlying AVSS (either statistical or perfect) which makes the resulting AMPC either statistical or perfect. We next informally discuss the approach used in our AVSS schemes.

[4]Since giving the exact details of the common coin and ABA is out of scope of the current article, we avoid any further discussion of it.

1.3 Approach Used in Our AVSS Schemes

For simplicity, we explain the underlying ideas of our AVSS schemes assuming that they share a single secret. We use the idea of sharing a secret by a bi-variate polynomial, as used in several existing schemes (see for example [20, 26, 28, 32, 36]). In the existing schemes, the dealer D selects a random bi-variate polynomial $F(x, y)$ of degree at most t in x and y, subject to the condition that $F(0, 0) = s$. We observe that given $n = 4t + 1$, the dealer can use a bi-variate polynomial of degree at most d in x and t in y for all d with $t \leq d \leq 2t$, to share a secret s. This of course does not come for free and certainly calls for new ideas on top of the existing schemes. By being able to hide the secret in a bi-variate polynomial of degree-(d, t) (we use this notation to denote bi-variate polynomials with degree at most d in x and t in y), we achieve a d-sharing of the secret. Our discussion below clarifies that it is not necessary to use polynomials of degree-(d, d) in order to generate d-sharing. In fact, we take advantage of the fact that the degree of one of the variables remains t.

So our scheme starts with the dealer D selecting a random bi-variate polynomial $F(x, y)$ of degree-(d, t) with $F(0, 0) = s$ and handing the univariate polynomials $f_i(x) = F(x, i)$ of degree at most d and $g_i(y) = F(i, y)$ of degree at most t to every party P_i. Let us first assume that D is honest. In this case, we can view the above distribution of information as if s is shared using a matrix M consisting of $n \times n$ values, as shown in Fig. 1, where every party P_i receives the i^{th} row and the i^{th} column of M via polynomial $f_i(x)$ and $g_i(y)$ respectively. This distribution allows the secret s to be d-shared through the univariate polynomial $f_0(x) = F(x, 0)$ of degree at most d where every (honest) party P_i has its share $Sh_i = f_0(i) = F(i, 0) = g_i(0)$ of the secret s. Moreover, each share Sh_i is t-shared among the n parties through the polynomial $g_i(y)$, where every party P_j has the *share-share* $Sh_{ij} = g_i(j)$ of the share Sh_i. Thus the bi-variate polynomial $F(x, y)$ facilitates *two-level sharing* of s (see Fig. 1): at the top level, s will be d-shared through the polynomial $f_0(x)$ and then in the second level, every share Sh_i is t-shared through the polynomial $g_i(y)$. Reconstruction of the secret s can be ensured by asking every party to reveal his share of s and then by applying the error correction on the revealed shares. Since $n = 4t + 1$ and $d \leq 2t$, the error correction will be robust, ensuring the correct reconstruction of $f_0(x)$ and hence s.

$$
\begin{array}{ccccccccccc}
F(1,1) & \cdots & F(i,1) & \cdots & F(j,1) & \cdots & F(n,1) & \Longrightarrow & f_1(x) \\
F(1,2) & \cdots & F(i,2) & \cdots & F(j,2) & \cdots & F(n,2) & \Longrightarrow & f_2(x) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
F(1,i) & \cdots & F(i,i) & \cdots & F(j,i) & \cdots & F(n,i) & \Longrightarrow & f_i(x) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
F(1,j) & \cdots & F(i,j) & \cdots & F(j,j) & \cdots & F(n,j) & \Longrightarrow & f_j(x) \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
F(1,n) & \cdots & F(i,n) & \cdots & F(j,n) & \cdots & F(n,n) & \Longrightarrow & f_n(x) \\
\Downarrow & \vdots & \Downarrow & \vdots & \Downarrow & \vdots & \Downarrow & & \\
g_1(y) & \cdots & g_i(y) & \cdots & g_j(y) & \cdots & g_n(y) & & \\
\Downarrow & \vdots & \Downarrow & \vdots & \Downarrow & \vdots & \Downarrow & & \\
Sh_1 = g_1(0) & \cdots & Sh_i = g_i(0) & \cdots & Sh_j = g_j(0) & \cdots & Sh_n = g_n(0) & \Longrightarrow & f_0(x)
\end{array}
$$

Figure 1: Matrix representation of the values distributed by (an honest) D in our AVSS schemes.

Although the second level sharing of the shares of the secret does not seem to serve any meaningful purpose for an honest dealer D, they are required for two different reasons to deal with a corrupted D. First, they ensure that D indeed d-shares (i.e. the underlying polynomial has degree at most d) the secret. Second, they are required to "complete" d-sharing of s, since a corrupted D may not hand the share Sh_i of s to every honest P_i. We use the second level t-sharing of Sh_i to reconstruct $g_i(y)$ for the party P_i and this enables P_i to receive $Sh_i = g_i(0)$. Now it is important to note that the second level sharings are t-sharings. So we can guarantee their robust reconstruction if we "ensure" that the t-sharing (of Sh_i's) have been done among a subset of $3t+1$ parties. Ensuring the above can be done with some additional ideas on top of the existing schemes. Had we used a bi-variate polynomial of degree-(d, d), we could not claim the same if $d > t$. This is because in this case, the second level sharings will have degree more than t and the impossibility of robust reconstruction of such sharings with $3t+1$ parties comes from the theory of error correction.

We now explain how the above idea is implemented in our schemes. After the dealer distributes the univariate polynomials, the honest parties try to identify and agree on a set of $3t+1$ parties, called CORE, such that the $f_i(x)$ polynomials of the honest parties in CORE lie on a unique bi-variate polynomial, say $\overline{F}(x, y)$, of degree-(d, t). Ideally, if D is honest then such a CORE always exists, as there are at least $3t+1$ honest parties and in this case, $\overline{F}(x, y) = F(x, y)$. However, if such a CORE is identified even in the case of a corrupted D, then it implies that D has distributed consistent polynomials to at least $(2t+1)$ honest parties, namely the honest parties in CORE. These consistent polynomials will uniquely define the bi-variate polynomial $\overline{F}(x, y)$ of degree-(d, t), which will be considered as D's committed bi-variate polynomial and the value $\overline{s} = \overline{F}(0, 0)$ will be considered as D's committed secret. To ensure that $\overline{s}$ is d-shared, it is enough that every (honest) P_i possesses $\overline{Sh_i} = \overline{f_0}(i)$, where $\overline{f_0}(x) = \overline{F}(x, 0)$ (a polynomial of degree at most d) and $\overline{s} = \overline{f_0}(0)$. Here we use the idea of "completing" the top level d-sharing of $\overline{s}$ with the help of the second level t-sharing of each of its shares $\overline{Sh_i}$. We note that each share $\overline{Sh_i}$ of $\overline{s}$ is shared among the parties in CORE through the polynomial $\overline{g_i}(y)$, where $\overline{g_i}(y) = \overline{F}(i, y)$ and has degree at most t. Since $|\mathsf{CORE}| \geq 3t+1$, the parties in CORE can send their share-share of $\overline{Sh_i}$ to P_i and enable P_i to reconstruct $\overline{g_i}(y)$ robustly by applying the error correction.

An interesting aspect of the described approach is that even though D distributes information on a bivariate polynomial of degree-(d, t) where d may be greater than t, we create a situation where the parties are required to reconstruct polynomials of degree at most t in order to obtain their shares of the secret. Now the main crux of our AVSS schemes is to identify and agree on a CORE. Once CORE is agreed upon, d-sharing can be completed by reconstructing the second level t-sharings of the shares of the committed secret, which is committed to the parties in CORE. We provide two methods to identify such a CORE: the first method applies random checks on the univariate polynomials distributed by D and has a negligible chance of incorrectly identifying a CORE. This results a statistical AVSS scheme. The second method identifies a CORE without any error and results in a perfect AVSS scheme. The details of these techniques will be available in the respective sections.

1.4 Organization of the Paper

The rest of the paper is organized as follows: in the next section, we describe the asynchronous network model and the definition of AVSS and AMPC. We also briefly discuss the existing tools which are used as building blocks in our AVSS and AMPC protocols. We present our AVSS schemes (both statistical and perfect) for sharing a single secret in section 3. This is followed by the discussion on the modifications required to extend these schemes to share multiple values simultaneously in section 4. The protocols for generating $(t, 2t)$-sharing using our AVSS schemes are presented in section 5. In section 6, we present our AMPC protocols, followed by a brief discussion on the application of our AVSS schemes in packed secret sharing in section 7. In section 8, we discuss the proposed statistical AMPC protocol of [31] and show that it is flawed. The paper ends with a few directions for future research.

2 Definitions and Preliminaries

2.1 Model

We follow the asynchronous network model of [15], where we have a set of $n = 4t + 1$ parties, say $\mathcal{P} = \{P_1, \ldots, P_n\}$, connected by point-to-point secure and authentic channels. A computationally unbounded active adversary $\mathcal{A}_t$ can actively corrupt at most t out of the n parties and make them behave in any arbitrary manner. The parties not under the influence of $\mathcal{A}_t$ are called honest or uncorrupted.

The underlying network is asynchronous, where the communication channels among the parties have arbitrary, yet finite delay (i.e. the messages are guaranteed to reach eventually). To model the worst case scenario, $\mathcal{A}_t$ is given the power to schedule the delivery of every message in the network. However, $\mathcal{A}_t$ can only schedule the messages communicated between the honest parties, without having any access to the content of these messages. In any asynchronous network, the inherent difficulty in designing a protocol comes from the fact that when a party does not receive an expected message then he cannot decide whether the sender is corrupted (and did not send the message at all) or the message is just delayed. So at any stage, a party can not wait for the communication from every party, as waiting for all of them could turn out to be endless. Hence the values from t (potentially honest) parties may have to be ignored. Due to this the protocols in the asynchronous networks are generally involved in nature and require a new set of primitives. For a comprehensive introduction to the asynchronous protocols, see [15].

For simplicity, we assume the adversary to be static, who decides the set of t parties to be corrupted at the beginning of the protocol (obviously, the honest parties will not know the identity of the corrupted parties). However, our protocols can be proved secure even in the presence of an adaptive adversary, who can decide which parties to corrupt after analyzing the information obtained so far during the execution of the protocol, provided that not more than t parties are under the control of the adversary.

2.2 Definitions

The computation in our protocols are performed over a finite field $\mathbb{F}$, with the following conditions: for the perfect AVSS and AMPC protocol, we require that $|\mathbb{F}| > n$. On the other hand, for the statistical AVSS and AMPC, we require $\mathbb{F} = GF(q)$, where $q > \max(n, 2^{\kappa})$, such that $\kappa = \log \frac{1}{\epsilon}$, for a given error parameter ϵ. Moreover, without loss of generality, we assume that $n = \text{poly}(\kappa)$. Every element from $\mathbb{F}$ can be represented by $\log |\mathbb{F}|$ bits.

We next recall the definition of AVSS from [8, 15].

Definition 1 (Asynchronous Verifiable Secret Sharing (AVSS) [8, 15])**.** *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 AVSS scheme with n parties if the following hold for every possible $\mathcal{A}_t$:*

1. **Termination***:*
   - *(a) If D is honest then each honest party will eventually terminate the protocol Sh.*
   - *(b) If some honest party has terminated the protocol Sh, then irrespective of the behavior of D, each honest party will eventually terminate Sh.*
   - *(c) If all honest parties have terminated Sh and all honest parties invoke the protocol Rec, then each honest party will eventually terminate Rec.*
2. **Correctness***:*
   - *(a) If D is honest then each honest party upon completing the protocol Rec, outputs the shared secret s.*

(b) *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 $\overline{s}$, irrespective of the behavior of the corrupted parties. This property is also known as the strong commitment property and we often say that D is committed to $\overline{s}$.*

3. **Secrecy***: If D is honest then the adversary's view during the protocol* **Sh** *reveals no information about s in the information theoretic sense. In other words, the adversary's view is identically distributed for all different values of s.*

The above definition can be extended in a straight forward way for a secret $S = (s_1, \ldots, s_\ell)$, containing ℓ elements from $\mathbb{F}$, where $\ell > 1$. We now dispose the definition of statistical and perfect AVSS.

Definition 2 (Statistical and Perfect AVSS)**.** *If an AVSS scheme satisfies the termination and the correctness condition with probability at least $(1 - \epsilon)$, for a given error parameter ϵ, then such a scheme is called a statistical AVSS. On the other hand, if the termination as well as the correctness condition is satisfied with probability 1 then such a scheme is called a perfect AVSS.*

Note that there is no compromise in the secrecy property for statistical AVSS. We now formally define d-sharing and $(t, 2t)$-sharing.

Definition 3 (d-sharing and $(t, 2t)$-sharing [5])**.** *We say that a value $s \in \mathbb{F}$ is d-shared among the n parties if there exists a polynomial $f(x)$ over $\mathbb{F}$ of degree at most d such that $f(0) = s$ and every (honest) party P_i holds a share Sh_i of s, where $Sh_i = f(i)$. We denote by $[s]_d$, the vector $(Sh_1, \ldots, Sh_n)$ of shares of s.*

A value $s \in \mathbb{F}$ is said to be $(t, 2t)$-shared among the n parties, denoted as $[s]_{t,2t}$, if s is both t-shared and $2t$-shared among the n parties through independent polynomials.

Notice that d-sharing is linear in the sense that by applying any linear function to d-sharings, we obtain a d-sharing as the output. This allows the parties to locally compute any linear function of the shares of d-shared values. Specifically, let $x^{(1)}, \ldots, x^{(m)}$ be m values which are d-shared among the parties, where $x_i^{(1)}, \ldots, x_i^{(m)}$ denotes the i^{th} share of $x^{(1)}, \ldots, x^{(m)}$ respectively. Let $H : \mathbb{F}^m \rightarrow \mathbb{F}^{m'}$ be a linear function, such that $H(x^{(1)}, \ldots, x^{(m)}) = (y^{(1)}, \ldots, y^{(m')})$. Then the parties can locally apply the function H on their shares of $x^{(1)}, \ldots, x^{(m)}$ and compute their shares of $(y^{(1)}, \ldots, y^{(m')})$. That is, every (honest) party P_i can locally compute $(y_i^{(1)}, \ldots, y_i^{(m')}) = H(x_i^{(1)}, \ldots, x_i^{(m)})$, where $y_i^{(1)}, \ldots, y_i^{(m')}$ denotes the i^{th} share of $y^{(1)}, \ldots, y^{(m')}$ respectively. We then say that the parties (locally) compute/generate $([y^{(1)}]_d, \ldots, [y^{(m')}]_d) = H([x^{(1)}]_d, \ldots, [x^{(m)}]_d)$.

Throughout the paper, we say that a bi-variate polynomial $F(x, y)$ over $\mathbb{F}$ has degree-(d, t) if the degree of x in $F(x, y)$ is at most d and the degree of y in $F(x, y)$ is at most t.

We now proceed to present the definition of AMPC. The definition of secure AMPC in the "real-world/ideal-world" paradigm was presented in [8]. Later [10] follows the same definition; though they present the definition in the style of "property based" definition. In the information theoretic world, the definitions of [10] and [8] are in essence "equivalent". All the papers on AMPC since then follow the definition presented in [10] and we follow the same. Since the main aim of this article is to provide an efficient AMPC protocol, to avoid making the paper complicated, we keep the formalities to a bare minimum and instead prove the security of our protocols using the definition of [10] presented below. However, our protocols can be proved secure according to the real world/ideal-world definition of [8], without affecting their efficiency.

Definition 4 (Secure Asynchronous Multi-Party Computation (AMPC)[10])**.** *Let $\mathcal{F} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ be a publicly known function and let party P_i has input $x_i \in \mathbb{F}$. Any asynchronous multiparty computation consists of three stages. In the first stage, each party P_i commits to its input. Even if P_i is faulty, if he completed this step, then he is committed to some value (not necessarily x_i). Let x'_i be the value committed by P_i. If P_i is* *honest then $x'_i = x_i$. Then the parties agree on a subset C of size at least $n - t$ of committed inputs. The subset C will be the same for all honest parties. In the last stage the (honest) parties will compute the value $\mathcal{F}(y_1, \ldots, y_n)$, where $y_i = x'_i$ if $P_i \in C$, otherwise $y_i = 0$.*

An asynchronous protocol Π among the n parties securely computes the function $\mathcal{F}$ if it satisfies the following conditions for every possible behavior of $\mathcal{A}_t$:

1. **Termination**: *Every honest party will eventually terminate Π.*

2. **Correctness**: *Every honest party will correctly output $\mathcal{F}(y_1, \ldots, y_n)$ after completing Π, irrespective of the behavior of the corrupted parties.*

3. **Secrecy**: *The adversary obtains no additional information (in the information theoretic sense), other than what is inferred from the input and the output of the corrupted parties.*

Based on whether the above properties are achieved with negligible error or without any error, we obtain statistical and perfect AMPC respectively.

Definition 5 (Statistical and Perfect AMPC). *If an AMPC protocol satisfies the termination and the correctness condition with probability at least $(1 - \epsilon)$, for a given error parameter ϵ, then such a protocol is called a statistical AMPC. On the other hand, if the termination as well as the correctness condition is satisfied with probability 1 then such a protocol is called a perfect AMPC.*

Note that there is no compromise in the secrecy property for statistical AMPC.

2.3 Primitives Used

A-cast: In our protocols, we use the asynchronous broadcast primitive, called A-cast, which was introduced and elegantly implemented by Bracha [14] with $3t + 1$ parties. Formally, A-cast is defined as follows:

Definition 6 (A-cast [16]). *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 behavior of $\mathcal{A}_t$:*

- **Termination**:

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

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

For the sake of completeness, we recall the Bracha's A-cast protocol from [15] and present it in Fig. 2.

Theorem 1 ([15]). *Protocol A-cast requires a private communication of $\mathcal{O}(\ell n^2)$ bits to broadcast an ℓ bit message.*

In the rest of the paper, we use the following notation while invoking the A-cast protocol:

Notation 1. *We say that party P_j receives m from the A-cast of P_i, if P_j (as a receiver) completes the execution of P_i's A-cast (the instance of the A-cast protocol where P_i is the sender), with m as the output.*

Figure 2: Bracha's A-cast protocol with $n = 3t + 1$.

A-cast

CODE FOR THE SENDER S (WITH INPUT m): only S executes this code

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

CODE FOR THE PARTY P_i: every party in $\mathcal{P}$ executes this code

1. Upon receiving a message (MSG, m), send $(ECHO, m)$ privately to all the parties.
2. Upon receiving $(n - t)$ messages $(ECHO, m^*)$ that agree on the value of m^*, send $(READY, m^*)$ privately to all the parties.
3. Upon receiving $(t + 1)$ messages $(READY, m^*)$ that agree on the value of m^*, send $(READY, m^*)$ privately to all the parties.
4. Upon receiving $(n - t)$ messages $(READY, m^*)$ that agree on the value of m^*, send (OK, m^*) privately to all the parties, accept m^* as the output message and terminate the protocol.

Agreement on a Common Subset (ACS): The next primitive we discuss is the ACS primitive [8, 10], which is used in all the existing AMPC protocols (including ours). It allows the (honest) parties to agree on a common set of $(n - t)$ parties satisfying "certain" property, say Q. To make the ACS protocol work, we must guarantee Q to be such that:

1. Every honest party will satisfy Q eventually. However, there are no restrictions for the corrupted parties. A corrupted party may or may not choose to satisfy Q.
2. If some honest party $P_j \in \mathcal{P}$ finds some party (probably corrupted) P_α to satisfy Q, then every other honest party in $\mathcal{P}$ will also eventually find P_α to satisfy Q.

Later in this article, we point out a flaw in the AMPC protocol of [31] that stems from the fact that [31] overlooked the second precondition on Q for employing an ACS instance.

For a better understanding, we consider the following scenario when ACS can be employed: Assume that every party in $\mathcal{P}$ is asked to A-cast a value and the property Q is whether a party has A-cast or not. The termination property of A-cast ensures that if some honest P_j finds some party, say P_α, to satisfy Q (that is P_α A-casted a value), then every other honest party in $\mathcal{P}$ will also eventually find P_α to satisfy Q. Thus using the ACS protocol, the (honest) parties can eventually agree on a common set of $(n - t)$ parties who have broadcast some value.

The idea behind the ACS protocol is to execute n instances of *asynchronous Byzantine Agreement* (ABA) [15], one on the behalf of each party to decide whether it will be in the common set. For the sake of completeness, we present the description of the protocol ACS (taken from [10]) in Fig. 3.

Theorem 2 ([10]). *Using the protocol ACS, the (honest) parties in $\mathcal{P}$ can agree on a common subset of at least $(n - t)$ parties, who will eventually satisfy the property Q. The communication complexity of the protocol is $\mathcal{O}(poly(n))$.*

The communication complexity of the ACS protocol depends on the cost of the underlying ABA protocol. Since ACS is invoked constant number of times in our AMPC protocols, we choose not to be explicit on its communication complexity.

Online Error Correction (OEC) [15, 5]: The next primitive we discuss is the online error correction, which can be viewed as the method of applying the Reed-Solomon (RS) error correction [34] in the asynchronous settings. Given a value which is τ-shared among a set of parties $\overline{\mathcal{P}} \subseteq \mathcal{P}$ with $\tau < (|\overline{\mathcal{P}}| - 2t)$, the

Figure 3: Protocol for the agreement on a common subset with $n = 4t + 1$.

Protocol ACS

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

1. For each $P_j \in \mathcal{P}$ such that $Q(j) = 1$ (i.e. P_j satisfies the property Q), participate in ABA_j with input 1. Here for $j = 1, \ldots, n$, ABA_j denotes the instance of ABA executed for $P_j \in \mathcal{P}$ to decide whether P_j will be in the common set.
2. Upon terminating $(n - t)$ instances of ABA with output 1, enter input 0 to all other instances of ABA, for which you have not entered a value yet.
3. Upon terminating all the n ABA protocols, let your $SubSet_i$ be the set of all indices j for which ABA_j had output 1.
4. Output the set of parties corresponding to the indices in $SubSet_i$ and terminate ACS.

goal is to make some party, say P_α, reconstruct the value robustly (actually OEC allows P_α to reconstruct the entire polynomial through which the value is τ-shared). In a synchronous network, this can be achieved by asking every party in $\overline{\mathcal{P}}$ to send its share to P_α, who can apply the RS error correction to reconstruct the value. Given the condition $\tau < (|\overline{\mathcal{P}}| - 2t)$, the reconstruction will be robust. In an asynchronous network, achieving the same goal requires a bit of trick as explained in the OEC of [15].

The intuition behind the OEC is that P_α keeps waiting till he receives $\tau + t + 1$ values, all of which lie on a unique polynomial of degree τ. This step requires applying the RS error correction repeatedly. We denote an RS error correcting procedure as $RS - DEC(\tau, r, W)$ that takes as input a vector W of shares (probably incorrect) of a τ-shared value (that we would like to reconstruct) and tries to output a polynomial of degree τ, by correcting at most r errors in W. Coding theory [34] says that $RS - DEC$ can correct r errors in W and correctly interpolate the original polynomial provided that $|W| \geq \tau + 2r + 1$. There are several efficient implementations of $RS - DEC$ (for example, the Berlekamp-Welch algorithm [34]). Once P_α receives $\tau + t + 1$ values that lie on a unique polynomial of degree τ (returned by $RS - REC$), then that unique polynomial is the actual polynomial, say $P(x)$, of degree τ that defines τ-sharing of $P(0)$. This is because at least $\tau + 1$ values out of the $\tau + t + 1$ values are from the honest parties, which uniquely define the original polynomial $P(x)$. Note that the corrupted parties in $\overline{\mathcal{P}}$ may send wrong values to P_α. But there are at least $|\overline{\mathcal{P}}| - t \geq (\tau + t + 1)$ honest parties in the set $\overline{\mathcal{P}}$ whose values will be eventually received by P_α and so P_α will eventually terminate the process. The above procedure is nothing but applying the RS error correction algorithm in an "online" fashion.

The steps for the OEC are now presented in Fig. 4. The current description is inspired from [15] (skipping several other formal details).

Theorem 3 ([15, 5])**.** *OEC achieves the following properties:*

1. *Correctness: Eventually party P_α will be able to correctly reconstruct the τ-sharing when $\tau < (|\overline{\mathcal{P}}| - 2t)$.*
2. *Privacy: If P_α is honest and the value $s = P(0)$ was information theoretically secure then s remains to be information theoretically secure at the end of OEC.*

PROOF: Let $\mathcal{A}_t$ corrupts $\hat{r} \leq t$ parties in $\overline{\mathcal{P}}$. During the $\hat{r}^{th}$ iteration, P_α receives $\tau + t + 1 + \hat{r}$ distinct values on $P(x)$, of which $\hat{r}$ are corrupted. Since $|I_{\hat{r}}| = \tau + t + 1 + \hat{r} \geq \tau + 2\hat{r} + 1$, $RS - DEC$ will correct $\hat{r}$ errors and will return $P(x)$ in this iteration. Thus $P(x)$ will be output in the $\hat{r}^{th}$ iteration and all the previous iterations up to the iteration $\hat{r}$ will be unsuccessful, as either no polynomial of degree τ is output or the output polynomial will not satisfy $\tau + t + 1$ values in I_r. This proves the correctness property.

Figure 4: Steps for the OEC.

Protocol OEC

Setting: A set of parties $\overline{\mathcal{P}} \subseteq \mathcal{P}$ hold τ-sharing of some value defined by a polynomial $P(x)$ of degree τ, where $\tau < (|\overline{\mathcal{P}}| - 2t)$. Namely, party $P_i \in \overline{\mathcal{P}}$ holds $v_i = P(i)$. A party, say $P_\alpha \in \mathcal{P}$, expects to reconstruct the τ-sharing (i.e. the polynomial $P(x)$).

CODE FOR THE PARTY P_α: For $r = 0, \ldots, t$, party P_α does the following in iteration r:

1. Let $\mathcal{W}$ denote the set of parties in $\overline{\mathcal{P}}$ from whom P_α has received the values and I_r denote the values received from the parties in $\mathcal{W}$, when $\mathcal{W}$ contains exactly $\tau + t + 1 + r$ parties.
2. Wait until $|\mathcal{W}| \geq \tau + t + 1 + r$. Apply $RS - DEC(\tau, r, I_r)$ to get a polynomial $P'(x)$ of degree τ. If no polynomial is output, then skip the next step and proceed to the next iteration.
3. If for at least $\tau + t + 1$ values $v_i \in I_r$ it holds that $P'(i) = v_i$, then P_α outputs $P'(x)$ and terminates. Otherwise, P_α proceeds to the next iteration.

It is easy to see that if P_α is honest and $s = P(0)$ was information theoretically secure, then even at the end of OEC, s will remain information theoretically secure. This is because the (honest) parties in $\overline{\mathcal{P}}$ only send the values to P_α. So no additional information about s or the values of $P(x)$ is revealed to $\mathcal{A}_t$. $\square$

Randomness Extraction: Here, we discuss about a well known method for randomness extraction in the information theoretic settings. We are given a set of values from $\mathbb{F}$, say $a_1, \ldots, a_N$, such that at least K out of these N values are selected uniformly and randomly from $\mathbb{F}$ and are information theoretically secure. The exact identity of those K values are not known. The goal is to compute K values $b_1, \ldots, b_K$ from $a_1, \ldots, a_N$, which are uniformly distributed over $\mathbb{F}^K$ and are information theoretically secure. This is achieved through the following well-known method introduced in [13, 12]: let $f(x)$ be a polynomial of degree at most $N-1$, such that $f(i) = a_{i+1}$, for $i = 0, \ldots, (N-1)$. Then set $b_1 = f(N), \ldots, b_K = f(N + K - 1)$ (of course we require $|\mathbb{F}| \geq N + K$ for this). We call this algorithm Ext and invoke it as $(b_1, \ldots, b_K) = \mathsf{Ext}(a_1, \ldots, a_N)$. Notice that Ext is a linear function of its inputs as it is based on polynomial interpolation.

Finding (n, t)-star: The last primitive we discuss here is finding an (n, t)-star in an undirected graph. We exploit some interesting properties of (n, t)-star in order to build our perfect AVSS protocol. An (n, t)-star is defined as follows:

Definition 7 (((n, t)-star[15, 8])**.** *Let G be an undirected graph with the n parties in $\mathcal{P}$ as its vertex set. We say that a pair (C, D) of sets with $\mathsf{C} \subseteq \mathsf{D} \subseteq \mathcal{P}$ is an (n, t)-star in G, if the following hold:*

1. $|\mathsf{C}| \geq n - 2t$;
2. $|\mathsf{D}| \geq n - t$;
3. *For every $P_j \in \mathsf{C}$ and every $P_k \in \mathsf{D}$ the edge (P_j, P_k) exists in G.*

In [8], the authors have presented an elegant and efficient algorithm for finding an (n, t)-star, provided 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, the algorithm Find-STAR takes the complementary graph $\overline{G}$ of G as input and tries to find an (n, t)-$\overline{\mathsf{star}}$ in $\overline{G}$ where an (n, t)-$\overline{\mathsf{star}}$ is a pair (C, D) of sets with $\mathsf{C} \subseteq \mathsf{D} \subseteq \mathcal{P}$, satisfying the following conditions:

1. $|\mathsf{C}| \geq n - 2t$;
2. $|\mathsf{D}| \geq n - t$;
3. There are no edges between the nodes in C and the nodes in D in $\overline{G}$.

Clearly, a pair (C, D) representing an (n,t)-$\overline{\mathsf{star}}$ in $\overline{G}$, is an (n,t)-star in G. Recasting the task of Find-STAR in terms of the complementary graph $\overline{G}$, we say that Find-STAR outputs either an (n,t)-$\overline{\mathsf{star}}$, or the message `star-Not-Found`. Whenever, the input graph $\overline{G}$ contains an independent set of size $n - t$, algorithm Find-STAR always outputs an (n,t)-$\overline{\mathsf{star}}$. For simple notation, we denote $\overline{G}$ by H. The algorithm Find-STAR is presented in Fig. 5.

Figure 5: Algorithm for finding an (n,t)-star.

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 an (n,t)-$\overline{\mathsf{star}}$ or the message `star-Not-Found`):
   (a) Let $T = \{P_i \in \overline{N} | \exists P_j, P_k \text{ s.t } (P_j, P_k) \in M \text{ and } (P_i, P_j), (P_i, P_k) \in E\}$. T is called the set of triangle-heads.
   (b) Let $\mathsf{C} = \overline{N} \setminus T$.
   (c) Let B be the set of matched nodes that have neighbours in C. So $B = \{P_j \in N | \exists P_i \in \mathsf{C} \text{ s. t. } (P_i, P_j) \in E\}$.
   (d) Let $\mathsf{D} = \mathcal{P} \setminus B$. If $|\mathsf{C}| \geq n - 2t$ and $|\mathsf{D}| \geq n - t$, output (C, D). Otherwise, output `star-Not-Found`.

Lemma 1 ([15]). *If Find-STAR outputs (C, D) on input graph H, then (C, D) is an (n,t)-$\overline{\mathsf{star}}$ in H.*

This completes our discussion on the tools and the preliminaries that are required for the rest of the article.

3 AVSS for Sharing a Single Secret

In this section, we present AVSS schemes that allow a dealer $D \in \mathcal{P}$ (the dealer can be any party from $\mathcal{P}$) to d-share a secret $s \in \mathbb{F}$ among the n parties, for a given d, where $t \leq d \leq 2t$. In the next section, we will show how to extend these schemes to share multiple secrets concurrently. We call our statistical AVSS scheme as SAVSS, while our perfect AVSS scheme is called PAVSS. In the rest of the paper, we distinguish the names of the statistical and perfect protocols/sub-protocols by their first character ('S' for statistical and 'P' for perfect). Some of the protocols (for example the protocol for the reconstruction phase) will be common for both the statistical as well as the perfect scheme. The names of such common protocols are not prefixed by 'S' or 'P'.

Structurally, the sharing protocol (SAVSS-Share and PAVSS-Share) of both the AVSS schemes is divided into a sequence of three phases as presented below. The sub-protocols implementing these phases are such that every honest party eventually terminates them when D is honest. On the other hand, if D is corrupted and some honest party terminates these phases, then every other honest party will also eventually terminate them.

1. *Distribution by D*: The protocols for this phase are called S-Distr (resp. P-Distr). Here D, on having a secret s and a publicly known degree of sharing d, distributes information to the parties in $\mathcal{P}$ to d-share

s. Specifically, as discussed in Sec.1.3, D selects a random bi-variate polynomial $F(x, y)$ of degree-(d, t), with s as the constant term. In protocol S-Distr, D hands the i^{th} polynomial $f_i(x) = F(x, i)$ to P_i. In addition to these polynomials, D will also distribute some "additional" information, which will be used in the later phases (of the statistical scheme) for some probabilistic checks. In protocol P-Distr, D hands the polynomial $f_i(x)$ and $g_i(y) = F(i, y)$ to P_i and no "additional" information is distributed to the parties. From now onwards, we call the $f_i(x)$ and $g_i(y)$ polynomial as the i^{th} row and column polynomial respectively (in connection with Fig. 1).

2. *Verification & Agreement on* CORE: The protocols for this phase are called S-Ver-Agree and P-Ver-Agree respectively. Though the goal is same, these two protocols are completely independent and are implemented with different techniques. In this phase, on receiving the information from D, the parties check whether D has distributed consistent information to the "enough" number of parties. For this, the statistical protocol S-Ver-Agree applies random checks on the row polynomials distributed by D and the protocol involves a negligible chance of incorrectly identifying such a "consistent set" of parties. On the other hand, in the perfect protocol P-Ver-Agree, each pair of parties exchange "common information" on their row and column polynomials and then we exploit some interesting properties of (n, t)-star to check the consistency of the information distributed by D. Protocol P-Ver-Agree identifies such a consistent set of parties without any error.

On a high level, the goal of the (honest) parties in this phase is to verify and agree on a set of at least $3t+1$ parties, called CORE, such that the row polynomials $\overline{f_i}(x)$ of the honest parties in CORE define a unique bivariate polynomial, say $\overline{F}(x, y)$, of degree-(d, t). That is, $\overline{f_i}(x) = \overline{F}(x, i)$ should hold for every honest $P_i \in$ CORE. Moreover, we also require that if D is honest, then the secrecy of s is still preserved during this verification process. If D is honest, then such a CORE always exists, as in this case $\overline{F}(x, y) = F(x, y)$ and for every honest P_i, $\overline{f_i}(x) = f_i(x)$. Moreover, there are at least $3t + 1$ honest parties in $\mathcal{P}$.

A common but crucial fact from the linear algebra used in S-Ver-Agree, as well as in P-Ver-Agree (to identify CORE) is as follows: given a set of at least $(t + 1)$ univariate polynomials of degree at most d and another set of at least $(d + 1)$ univariate polynomials of degree at most t, which are "pair-wise consistent", then all these polynomials lie on a unique bi-variate polynomial of degree-(d, t). More formally:

Lemma 2. *Let $\overline{f_1}(x), \ldots, \overline{f_l}(x)$ be l polynomials of degree at most d over $\mathbb{F}$ and let $\overline{g_1}(y), \ldots, \overline{g_m}(y)$ be m polynomials of degree at most t over $\mathbb{F}$, where $l \geq (t + 1)$ and $m \geq (d + 1)$, such that for every $1 \leq i \leq l$ and for every $1 \leq j \leq m$, we have $\overline{f_i}(j) = \overline{g_j}(i)$. Then there exists a unique bi-variate polynomial $\overline{F}(x, y)$ over $\mathbb{F}$ of degree-(d, t), such that $\overline{F}(x, i) = \overline{f_i}(x)$ and $\overline{F}(j, y) = \overline{g_j}(y)$, for $1 \leq i \leq l$ and $1 \leq j \leq m$.*

PROOF: The proof is very similar to the proof of Lemma 4.26 of [15]. For the sake of completeness, the proof is given in APPENDIX A. □

3. *Generation of d-sharing*: The goal of this phase is to enable every honest party P_i to receive his share Sh_i of the secret. If the parties agree on a CORE of size at least $3t + 1$ in the previous phase, then it implies that there exists some bi-variate polynomial, say $\overline{F}(x, y)$ of degree-(d, t), such that $\overline{F}(x, i) = \overline{f_i}(x)$ for every honest P_i in CORE, where $\overline{f_i}(x)$ is the row polynomial held by P_i. We consider $\overline{s} = \overline{F}(0, 0)$ as D's committed secret. If D is honest then $\overline{F}(x, y) = F(x, y)$ and $\overline{s} = s$. Now we note that the univariate polynomial $\overline{f_0}(x) = \overline{F}(x, 0)$ is of degree at most d and $\overline{s} = \overline{f_0}(0)$. So d-sharing of $\overline{s}$ with $Sh_i = \overline{f_0}(i)$ being the i^{th} share of $\overline{s}$ can be completed if every (honest) party P_i holds $\overline{f_0}(i)$. This can be easily achieved since each Sh_i is t-shared among the parties in CORE through the polynomial $\overline{g_i}(y)$, where $\overline{g_i}(y) = \overline{F}(i, y)$ and $Sh_i = \overline{g_i}(0)$ since $\overline{g_i}(0) = \overline{f_0}(i)$. In order words, the second level t-sharing of the shares of $\overline{s}$ is already done among the parties in CORE. Since $|\text{CORE}| \geq 3t + 1$, OEC allows P_i to reconstruct $\overline{g_i}(y)$. Party P_i now gets his share $Sh_i = \overline{g_i}(0)$. The protocol for this phase is common for both the AVSS schemes. We call this protocol as Gen and present it in Fig. 6.

Figure 6: Protocol for the generation of d-sharing phase. The protocol is common for the sharing phase of both the statistical and the perfect AVSS scheme.

Protocol Gen

Setting: The parties in $\mathcal{P}$ have agreed on CORE, where $|\mathsf{CORE}| \geq 3t+1$. Party $P_i \in \mathsf{CORE}$ holds the row polynomial $\overline{f_i}(x)$ received from the dealer D during the distribution by D phase. Moreover, the row polynomials of the honest parties in CORE define a unique bi-variate polynomial, say $\overline{F}(x,y)$ of degree-(d,t). The goal is to make every party $P_i \in \mathcal{P}$ reconstruct the polynomial $\overline{g_i}(y) = \overline{F}(i,y)$ and its share of the secret $\overline{F}(0,0)$.

CODE FOR P_i: Every party executes this code

1. If $P_i \in \mathsf{CORE}$, then for $j = 1,\ldots,n$, privately send $\overline{f_i}(j)$ to the party P_j. Recall that $\overline{g_j}(i) = \overline{f_i}(j)$. So P_i actually sends a value on $\overline{g}_j(y)$ to P_j.
2. Apply the protocol OEC on $\overline{f_j}(i)$'s received from P_j's belonging to CORE to privately reconstruct the polynomial $\overline{g_i}(y)$ of degree at most t, output $Sh_i = \overline{g_i}(0)$ and terminate.

We state the following lemma for the protocol Gen.

Lemma 3. *Assume that every honest party has agreed upon* CORE*. Then protocol Gen satisfies the following properties:*

1. *Protocol Gen generates d-sharing of $\overline{s} = \overline{F}(0,0)$. If D is honest, then $\overline{s} = s$ where s is D's secret.*
2. *Protocol Gen requires a private communication of $\mathcal{O}(n^2 \log|\mathbb{F}|)$ bits.*

PROOF: The property of CORE implies that the row polynomial $\overline{f_i}(x)$ of every (honest) $P_i \in \mathsf{CORE}$ lies on a bi-variate polynomial $\overline{F}(x,y)$ of degree-(d,t). Moreover, if D is honest then $\overline{F}(x,y)$ is the same bi-variate polynomial $F(x,y)$ selected by D in the first phase. Let $\overline{f_0}(x) \stackrel{def}{=} \overline{F}(x,0), \overline{s} \stackrel{def}{=} \overline{F}(0,0)$ and $\overline{g_i}(y) \stackrel{def}{=} \overline{F}(i,y)$. The polynomial $\overline{g_i}(y)$ is of degree at most t and $|\mathsf{CORE}| \geq 3t+1$. Substituting $\overline{\mathcal{P}} = \mathsf{CORE}$ and $\tau = t$ in the protocol OEC (see Fig. 4), we find that each honest P_i will eventually compute $Sh_i = \overline{g_i}(0)$ from the $\overline{f_j}(i)$'s (which are same as $\overline{g_i}(j)$'s) received from the parties in CORE. Moreover, $\overline{g_i}(0) = \overline{f_0}(i)$. So $\overline{s}$ will be d-shared through the polynomial $\overline{f_0}(x)$. If D is honest then $\overline{f_0}(x) = f_0(x) = F(x,0)$.

In the protocol, every party in CORE does a private communication of n elements from $\mathbb{F}$. So this requires a total private communication of $\mathcal{O}(n^2 \log|\mathbb{F}|)$ bits. □

The protocols for the sharing phase and the reconstruction phase of our AVSS schemes are presented in Fig. 7. By substituting appropriate protocol for a phase (presented in the sequel), we get either the statistical AVSS scheme SAVSS or the perfect AVSS scheme PAVSS.

In the sequel, we describe the protocols S-Distr and S-Ver-Agree, followed by the description of their perfect counter parts P-Distr and P-Ver-Agree. Before that, we state the property of the protocol Rec, which is the common protocol for the reconstruction phase of both the AVSS schemes.

Lemma 4. *Let s be a value which is d-shared among the n parties, where $t \leq d \leq 2t$. Then by executing the protocol Rec, every honest party will eventually reconstruct s and terminate. The protocol requires a private communication of $\mathcal{O}(n^2 \log|\mathbb{F}|)$ bits.*

PROOF: Follows when we substitute $|\overline{\mathcal{P}}| = |\mathcal{P}| = 4t+1$ and $\tau = d \leq 2t$ in the protocol OEC. In protocol Rec, every party sends its share to every other party resulting in $\mathcal{O}(n^2 \log|\mathbb{F}|)$ bits of communication. □

3.1 Sub-Protocols for the Statistical AVSS Scheme

We now present the protocols S-Distr and S-Ver-Agree.

Figure 7: The AVSS scheme for sharing a single secret. Here s is the secret, D is the dealer and d is the degree of the sharing.

Protocol for the Sharing Phase

1. *Distribution by D Phase*: D either executes the protocol S-Distr (for SAVSS) or the protocol P-Distr (for PAVSS).
2. *Verification & Agreement on* CORE *Phase*: The parties check the existence of CORE and agree on it if it exists by executing the protocol S-Ver-Agree (for SAVSS) or P-Ver-Agree (for PAVSS).
3. *Generation of d-sharing Phase*: If CORE is generated and agreed upon in the previous phase, then the parties execute the protocol Gen.

Protocol Rec for the Reconstruction Phase

CODE FOR P_i: every party executes this code

1. Privately send Sh_i, the i^{th} share of the secret to every $P_j \in \mathcal{P}$.
2. Apply the OEC on the received Sh_j's, reconstruct the secret and terminate.

3.1.1 Protocol S-Distr

Here D on having a secret s, selects a random bivariate polynomial $F(x, y)$ of degree-(d, t) with the constant term s and sends to P_i the i^{th} row polynomial. In addition, D also distributes some "additional" information which will be used to preserve the secrecy of s during the probabilistic checks performed in protocol S-Ver-Agree. Precisely, D distributes the shares of $(t+1)n$ random univariate polynomials of degree at most t. Since these polynomials will be used for "masking" later, we call them as the masking polynomials. The reason for taking $(t+1)n$ masking polynomials will be clear when we present the protocol S-Ver-Agree. Now the protocol S-Distr is presented in Fig. 8.

Figure 8: Protocol S-Distr. Here D is the dealer, s is the secret and d is the degree of the sharing.

Protocol S-Distr

CODE FOR D: Only D executes this code

1. Select a random bivariate polynomial $F(x, y)$ of degree-(d, t) over $\mathbb{F}$, such that $F(0, 0) = s$. For $i = 0, \ldots, n$, let $f_i(x) \stackrel{def}{=} F(x, i)$ and $g_i(y) \stackrel{def}{=} F(i, y)$.
2. Select $(t+1)n$ random masking polynomials of degree at most t over $\mathbb{F}$, denoted by $m_{(P_i,1)}(y), \ldots, m_{(P_i,t+1)}(y)$, for $i = 1, \ldots, n$.
3. For $i = 1, \ldots, n$, send the following to the party P_i:
   (a) The row polynomial $f_i(x)$;
   (b) For $j = 1, \ldots, n$, the i^{th} shares $m_{(P_j,1)}(i), \ldots, m_{(P_j,t+1)}(i)$ of the masking polynomials.

We make the following claim about S-Distr, that trivially follows from the protocol description.

Claim 1. *In protocol S-Distr, D privately communicates $\mathcal{O}((nd + n^3) \log |\mathbb{F}|)$ bits. Since $d \leq 2t$, the communication complexity is $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits.*

3.1.2 Protocol S-Ver-Agree

Recall that the goal of the protocol S-Ver-Agree is to enable the (honest) parties in $\mathcal{P}$ to check whether there exists a set CORE of at least $3t + 1$ parties, such that the row polynomials of the honest parties in CORE lie on a unique bi-variate polynomial of degree-(d, t) and if such a set exists then the parties agree on it. Let $\overline{f_i}(x)$ be the row polynomial of degree at most d, received by P_i from D. If D is honest then $\overline{f_i}(x) = f_i(x)$. The properties of bi-variate polynomial of degree-(d, t) say that if indeed such a CORE exists then the points $\{\overline{f_i}(j) : P_i \in \mathsf{CORE}\}$ will define some polynomial, say $\overline{g_j}(y)$, of degree at most t, for every $j = 1, \ldots, n$. So the goal of protocol S-Ver-Agree is to enable the parties to check whether D has distributed the row polynomials in such a way that the j^{th} point on the row polynomials of at least $(3t + 1)$ parties define polynomials of degree at most t. Such a set of $3t + 1$ parties can be considered as CORE.

To check the above, we use the following known fact about probabilistic checks on polynomials: if a random linear combination of a set of polynomials has degree at most t, then with very high probability, each individual polynomial in the set has also degree at most t. Formally,

Lemma 5. *Let $h_0(y), \ldots, h_l(y)$ be polynomials where $l \geq 1$ and let r be a random, non-zero element from $\mathbb{F}$. Assuming $\ell = poly(\kappa)$, if the polynomial $h_{com}(y) \stackrel{def}{=} h_0(y) + rh_1(y) + \ldots + r^l h_l(y)$ is of degree at most t, then except with probability $2^{-\Omega(\kappa)} \approx \epsilon$, each polynomial $h_0(y), \ldots, h_l(y)$ has also degree at most t.*

PROOF: For the sake of completeness, the proof is given in APPENDIX A. □

In more detail, consider a set of at least $3t+1$ parties, say ReceivedSet, who claimed to receive their row polynomials $\overline{f_i}(x)$ and their shares of the masking polynomials (recall that $(t+1)n$ masking polynomials are also shared in the distribution phase). Let the points $\{\overline{f_i}(j) : P_i \in \mathsf{ReceivedSet}\}$ define some polynomial $\overline{g_j}(y)$. In order to check if each of the polynomials $\overline{g_j}(y)$ for $j = 1, \ldots, n$ is of degree at most t, we check if the polynomial $E(y) = \overline{m}(y) + r\overline{g_1}(y) + \ldots + r^n\overline{g_n}(y)$ is of degree at most t. Here $\overline{m}(y)$ is a masking polynomial. And r is a random combiner, which is made public, after D's delivery of the row polynomials to the (honest) parties in ReceivedSet (we will show in the sequel how such an r will be available). We ask D to publish $E(y)$ and every $P_i \in \mathsf{ReceivedSet}$ to publish the random linear combination $\overline{m}(i) + r\overline{f_i}(1) + \ldots + r^n\overline{f_i}(n)$. If D's published polynomial has degree at most t and there are at least $3t+1$ parties in ReceivedSet such that their published points lie on D's published polynomial of degree at most t, then except with probability $2^{-\Omega(\kappa)} \approx \epsilon$, the polynomials $\overline{g_j}(y)$ for every $j = 1, \ldots, n$ are of degree at most t. This holds since D had no idea about the random r when he distributed the row polynomials and the shares of the masking polynomials to the (honest) parties in ReceivedSet. The set of parties in ReceivedSet whose points match with D's polynomial constitute a candidate for CORE when the set admits a size of at least $3t + 1$.

The secrecy of the row polynomials of the honest parties in ReceivedSet will be preserved during the above probabilistic check due to the use of the masking polynomial $\overline{m}(y)$. Having said the core idea, we now disclose some crucial issues that we face when we try to implement the above idea in the asynchronous settings. The issues are when and how to generate the random combiner r and who decides a ReceivedSet and how many such candidate ReceivedSet need to be examined to finally get a CORE. We must generate the random r in such a way that it remains secret from D during his distribution of the row polynomials and shares of the masking polynomials to the parties in ReceivedSet. Otherwise, a corrupted D can go undetected even after distributing inconsistent polynomials of higher degree to the parties in ReceivedSet. We solve this issue by asking some party $V \in \mathcal{P}$ to act as the *verifier* and select the random challenge r. If V is honest and the parties in ReceivedSet receive their row polynomials and points on masking polynomials before V makes r public, then the above described probabilistic check works. However, it is difficult to identify an honest verifier V and so we ask every party in $\mathcal{P}$ to individually play the role of a verifier. So we first construct a protocol navigated by a single verifier V. The protocol outputs a number of candidates for CORE, which are indeed "true" candidates for CORE, if V is honest. Later when running this single verifier protocol for each of the verifiers in $\mathcal{P}$, we show how to choose the CORE from many candidates, making sure that it is "approved" by at least one honest verifier. Our first goal is thus to construct a protocol for a single verifier V.

Protocol for the Navigation by a Single Verifier: Since V generates the random challenge r, it must ensure that indeed the (honest) parties in ReceivedSet already received their values from D; otherwise the probabilistic check is of no use if a corrupted D is able to know r before distributing the values to the honest parties in ReceivedSet. For this, we let every party inform V when they receive their values from D and let V to construct and decide the ReceivedSet, based on from whom he receives responses, before it generates the challenge for the set. Now an interesting question is the following: Is it enough for V to generate a single ReceivedSet containing the $3t+1$ parties who respond to V and stopping immediately? Does this allow to stumble on a candidate CORE? The answer is no. Specifically, ReceivedSet may contain t corrupted parties, who can reveal incorrect linear combination of the points on their row polynomials. Even when D is honest, we can only guarantee that the honest parties in ReceivedSet (say exactly $2t+1$) respond correctly and thus agree with D's published polynomial. But recall that in order to be considered as a candidate for CORE, the set of parties who agree with D's published polynomial should admit a size of at least $3t+1$. This implies that V cannot find a candidate for CORE by examining a single ReceivedSet.

As a remedy for the above problem, we ask V to start with a ReceivedSet of size $3t+1$ and keep "expanding" the ReceivedSet as in when it receives confirmations from more parties about their receipt of row polynomial and shares of the masking polynomials. Whenever ReceivedSet is expanded, V generates a new random challenge r and makes the corresponding version of ReceivedSet and the newly generated random r public. When a ReceivedSet and random challenge r is made public by V, the dealer D as well as the parties in that ReceivedSet respond to the challenge. Specifically, D broadcasts the linearly combined polynomial and the parties in ReceivedSet publish the linearly combined points. This can be perceived as a game between D and the parties in ReceivedSet, navigated by the verifier V, who decides ReceivedSet, creates the challenge and then asks D and ReceivedSet to play the game. In the game, D wishes to convince everyone that the information that he handed to the parties in ReceivedSet are consistent (without violating the privacy of s). Clearly, if a ReceivedSet has at least $3t+1$ parties such that the linearly combined points of those parties match with the polynomial published by D, then such a set of $3t+1$ parties is a contender for CORE. We denote by AgreeSet the set of parties in ReceivedSet, whose linear combination of points matches the linear combination of polynomials published by D.

For an honest D and V, we are guaranteed to eventually see at least one candidate for CORE, namely when all the honest parties will be in ReceivedSet, whose response will match the polynomial published by D. We further note that with very high probability, a corrupted D can not cheat when V is honest, since V creates r only after getting the confirmation of the receipt of row polynomials and the shares of the masking polynomials from the parties in ReceivedSet and more importantly, a random r is chosen for every new (expanded) version of ReceivedSet. Our final observation is that there can be at most $t+1$ different versions of ReceivedSet, since the initial version may have $3t+1$ parties and the final version may have all the $4t+1$ parties. So V may need to generate a random challenge at most $t+1$ times and the checking game will be performed at most $t+1$ times. Each time the game is played between D and a distinct version of ReceivedSet, using the associated random challenge r, published along with the version of ReceivedSet. This clearly implies that in order to maintain the privacy of the row polynomials of the honest parties during the probabilistic checks, every time a different masking polynomial is to be used. Thus we require $t+1$ masking polynomials on the behalf of a single V (and total $(t+1)n$ masking polynomials for n verifiers). We now present the protocol Single-Verifier in Fig. 9 that captures the above discussion for a verifier V.

We next prove some important properties of the protocol Single-Verifier: the first property is that if D and V are honest, then eventually some $\mathsf{AgreeSet}_{(V,\beta)}$ will be generated (Lemma 6). This property

Figure 9: Verification with respect to a verifier $V \in \mathcal{P}$

Protocol Single-Verifier

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

1. Wait to receive the row polynomial $\overline{f_i}(x)$ and the shares $\overline{m}_{(P_j,1)}(i), \ldots, \overline{m}_{(P_j,t+1)}(i)$ of the masking polynomials, for $j = 1, \ldots, n$ from D.
2. Check if $\overline{f_i}(x)$ is a polynomial of degree at most d. If yes, then privately send an $(\texttt{ECHO}, i)$ signal to V.

ii. CODE FOR V (TO GENERATE THE CHALLENGE): Only V executes this code

1. Wait to receive $(\texttt{ECHO}, i)$ signal from $3t+1$ parties. Put the identity of these $3t+1$ parties in the set $\mathsf{ReceivedSet}_{(V,1)}$. Select a random, non-zero value $r_{(V,1)}$ from $\mathbb{F}$ and A-cast $(r_{(V,1)}, \mathsf{ReceivedSet}_{(V,1)})$.
   /* $\mathsf{ReceivedSet}_{(V,1)}$ denotes the first set of $3t+1$ parties, who in V's view have received their row polynomials and shares of the masking polynomials from D. */
2. After the previous step, for every new receipt of $(\texttt{ECHO}, i)$ signal from a party $P_i \notin \mathsf{ReceivedSet}_{(V,\beta-1)}$, where $1 < \beta \leq t+1$, construct $\mathsf{ReceivedSet}_{(V,\beta)} = \mathsf{ReceivedSet}_{(V,\beta-1)} \cup \{P_i\}$, select a random, non-zero $r_{(V,\beta)} \in \mathbb{F}$ and A-cast $(r_{(V,\beta)}, \mathsf{ReceivedSet}_{(V,\beta)})$.
   /* This step is for the expansion of ReceivedSet and the challenge generation for the expanded set. */

iii. CODE FOR D (TO RESPOND TO THE CHALLENGE OF V): Only D executes this code

1. If $(r_{(V,\beta)}, \mathsf{ReceivedSet}_{(V,\beta)})$ is received from the A-cast of V, then A-cast the linear combination $E_{(V,\beta)}(y)$ of the masking polynomial $m_{(V,\beta)}(y)$ and the n column polynomials $g_1(y), \ldots, g_n(y)$, where

$$E_{(V,\beta)}(y) \stackrel{def}{=} m_{(V,\beta)}(y) + r_{(V,\beta)} g_1(y) + \ldots + r^n_{(V,\beta)} g_n(y).$$

   /* D broadcasts the linear combination $E_{(V,\beta)}(y)$ of the polynomials in response to the challenge $r_{(V,\beta)}$ to publicly demonstrate that the row polynomials of the parties in $\mathsf{ReceivedSet}_{(V,\beta)}$ satisfy the properties of CORE. */

iv. CODE FOR P_i (TO RESPOND TO THE CHALLENGE OF V): Every party in $\mathcal{P}$ executes this code

1. If $(r_{(V,\beta)}, \mathsf{ReceivedSet}_{(V,\beta)})$ is received from the A-cast of V, then check if $P_i \in \mathsf{ReceivedSet}_{(V,\beta)}$. If yes, then A-cast the linear combination $e_{(V,\beta,i)}$ of the share $\overline{m}_{(V,\beta)}(i)$ of the masking polynomial $m_{(V,\beta)}(y)$ and the points $\overline{f_i}(1), \ldots, \overline{f_i}(n)$ on the row polynomial $\overline{f_i}(x)$, where

$$e_{(V,\beta,i)} \stackrel{def}{=} \overline{m}_{(V,\beta)}(i) + r_{(V,\beta)} \overline{f_i}(1) + \ldots + r^n_{(V,\beta)} \overline{f_i}(n).$$

   /* This step denotes that every party P_i in $\mathsf{ReceivedSet}_{(V,\beta)}$ broadcasts the linear combination $e_{(V,\beta,i)}$ of the points on his row polynomial and the appropriate masking polynomial, in response to the challenge $r_{(V,\beta)}$. Ideally, the point $e_{(V,\beta,i)}$ should lie on the polynomial $E_{(V,\beta)}(y)$. */
2. Consider a party P_j to agree with D with respect to the pair $(r_{(V,\beta)}, \mathsf{ReceivedSet}_{(V,\beta)})$, where $\beta \in \{1, \ldots, t+1\}$, if all the following holds:
   (a) $E_{(V,\beta)}(y)$ A-casted by D is a polynomial of degree at most t;
   (b) $P_j \in \mathsf{ReceivedSet}_{(V,\beta)}$ and
   (c) $e_{(V,\beta,j)} = E_{(V,\beta)}(j)$, where $e_{(V,\beta,j)}$, $E_{(V,\beta)}(y)$ and $(r_{(V,\beta)}, \mathsf{ReceivedSet}_{(V,\beta)})$ are received from the A-casts of P_j, D and V respectively.

   /*This step checks if the row polynomial of a party $P_j \in \mathsf{ReceivedSet}_{(V,\beta)}$ satisfies the challenge $r_{(V,\beta)}$. */
3. With respect to the pair $(r_{(V,\beta)}, \mathsf{ReceivedSet}_{(V,\beta)})$, when there are $3t+1$ P_j's who agree with D, add such P_j's in the set $\mathsf{AgreeSet}_{(V,\beta)}$.
   /*Here $\mathsf{AgreeSet}_{(V,\beta)}$ denotes a set of $3t+1$ parties in $\mathsf{ReceivedSet}_{(V,\beta)}$, whose row polynomials satisfy the random challenge $r_{(V,\beta)}$. */

is essential to guarantee the termination of the protocol S-Ver-Agree (where Single-Verifier is used as a black-box) when D is honest. We then show that if V is honest and some $\mathsf{AgreeSet}_{(V,\beta)}$ is generated, then the

j^{th} point on the row polynomials of the honest parties in $\mathsf{AgreeSet}_{(V,\beta)}$ indeed define polynomials of degree at most t (Lemma 7). This will further imply that the row polynomials of the honest parties in $\mathsf{AgreeSet}_{(V,\beta)}$ lie on a unique bivariate polynomial of degree-(d,t) (Lemma 8), implying that $\mathsf{AgreeSet}_{(V,\beta)}$ is a candidate for CORE. Finally, we show that if D is honest, then the secret s remains information theoretically secure at the end of Single-Verifier, even if V is corrupted. This will ensure information theoretic security for s in protocol S-Ver-Agree.

Lemma 6. *In protocol Single-Verifier, if V and D are honest, then eventually an $\mathsf{AgreeSet}_{(V,\beta)}$ with $|\mathsf{AgreeSet}_{(V,\beta)}| \geq 3t+1$ will be generated, where $\beta \in \{1,\ldots,t+1\}$.*

PROOF: If D is honest, then eventually the set of (at least) $3t+1$ honest parties will correctly receive their row polynomials and these polynomials will satisfy any random challenge r generated by an honest V. That is, the linear combination of the points revealed by these parties will lie on the corresponding linear combination of the polynomials revealed by D. Thus, for some $\beta \in \{1,\ldots,t+1\}$, $\mathsf{ReceivedSet}_{(V,\beta)}$ will contain $3t+1$ honest parties who will also appear in $\mathsf{AgreeSet}_{(V,\beta)}$. □

Lemma 7. *In protocol Single-Verifier, if V is honest and some $\mathsf{AgreeSet}_{(V,\beta)}$ (containing at least $3t+1$ parties) has been generated, then the following holds with probability at least $(1-\epsilon)$:*

1. *For all $j = 1,\ldots,n$, the j^{th} point on the row polynomials of the honest parties in $\mathsf{AgreeSet}_{(V,\beta)}$ define some polynomial, say $\overline{g_j}(y)$, of degree at most t.*

2. *The shares of the masking polynomial $m_{(V,\beta)}(y)$ held by the honest parties in $\mathsf{AgreeSet}_{(V,\beta)}$ define some polynomial of degree at most t.*

PROOF: If D is honest, then the lemma will be true, without any error. Hence we consider the case when D is corrupted. So let us assume that an $\mathsf{AgreeSet}_{(V,\beta)}$, where $|\mathsf{AgreeSet}_{(V,\beta)}| \geq 3t+1$ is generated from $\mathsf{ReceivedSet}_{(V,\beta)}$ and let $H_{(V,\beta)}$ denote the set of honest parties in $\mathsf{AgreeSet}_{(V,\beta)}$. Since V is honest, a corrupted D while distributing the row polynomials and the shares of the masking polynomials to the (honest) parties in $\mathsf{ReceivedSet}_{(V,\beta)}$, is oblivious of the random challenge $r_{(V,\beta)}$. The challenge $r_{(V,\beta)}$ is generated when V receives the $(\texttt{ECHO}, \star)$ signal from every (honest) party in $\mathsf{ReceivedSet}_{(V,\beta)}$. Let the shares $\{\overline{m}_{(V,\beta)}(i) : P_i \in H_{(V,\beta)}\}$ define the polynomial $\overline{m}_{(V,\beta)}(y)$ and let for $j = 1,\ldots,n$, the points $\{\overline{f_i}(j) : P_i \in H_{(V,\beta)}\}$ define the polynomial $\overline{g_j}(y)$. Then the value $e_{(V,\beta,i)}$, broadcasted by $P_i \in H_{(V,\beta)}$ in response to the challenge $r_{(V,\beta)}$ is:

$$e_{(V,\beta,i)} = \overline{m}_{(V,\beta)}(i) + r_{(V,\beta)}\overline{g_1}(i) + \ldots + r^n_{(V,\beta)}\overline{g_n}(i).$$

We will now show that except with probability ϵ, the polynomials $\overline{m}_{(V,\beta)}(y), \overline{g_1}(y), \ldots, \overline{g_n}(y)$ are of degree at most t. On the contrary, if at least one of these $(n+1)$ polynomials has degree more than t, then we can show that the minimum degree polynomial, say $E_{min}(y)$, defined by the points $\{e_{(V,\beta,i)} : P_i \in H_{(V,\beta)}\}$ will have degree more than t with probability at least $(1-\epsilon)$. This will clearly imply $E_{(V,\beta)}(y) \neq E_{min}(y)$ and hence $e_{(V,\beta,i)} \neq E_{(V,\beta)}(i)$ will hold for at least one $P_i \in H_{(V,\beta)}$. This will be a contradiction, as $e_{(V,\beta,i)} = E_{(V,\beta)}(i)$ holds for every $P_i \in \mathsf{AgreeSet}_{(V,\beta)}$ and $H_{(V,\beta)}$ is a subset of $\mathsf{AgreeSet}_{(V,\beta)}$.

So we proceed to prove that $E_{min}(y)$ will be of degree more than t with probability at least $(1-\epsilon)$, when one of the polynomials $\overline{m}_{(V,\beta)}(y), \overline{g_1}(y), \ldots, \overline{g_n}(y)$ has degree more than t. For this, we show the following:

1. We first claim that if one of the polynomials $\overline{m}_{(V,\beta)}(y), \overline{g_1}(y), \ldots, \overline{g_n}(y)$ has degree more than t, then with probability at least $(1-\epsilon)$, the polynomial $E_{def}(y) \stackrel{def}{=} \overline{m}_{(V,\beta)}(y) + r_{(V,\beta)}\overline{g_1}(y) + \ldots + r^n_{(V,\beta)}\overline{g_n}(y)$ will also have degree more than t, for any random, non-zero challenge $r_{(V,\beta)}$. This follows from the property of polynomials, as stated in Lemma 5.

2. We next claim that $E_{min}(y) = E_{def}(y)$. For this, we first observe that in the protocol, every $e_{(V,\beta,i)}$ broadcasted by every $P_i \in H_{(V,\beta)}$ lies on the polynomial $E_{def}(y)$ (this condition has to be satisfied for P_i to be in the $\mathsf{AgreeSet}_{(V,\beta)}$). Now consider the difference polynomial $dp(y) = E_{def}(y) - E_{min}(y)$. Clearly, $dp(y) = 0$, for all $y = i$, where $P_i \in H_{(V,\beta)}$. Thus $dp(y)$ will have at least $|H_{(V,\beta)}|$ roots. On the other hand, the maximum degree of $dp(y)$ could be $|H^{(V,\beta)}| - 1$. This is because $E_{def}(y)$ is defined by the points on the row polynomials held by the parties in $H_{(V,\beta)}$ and so the maximum degree of $E_{def}(y)$ can be $|H^{(V,\beta)}| - 1$. These two facts together imply that $dp(y)$ is the zero polynomial, implying that $E_{def}(y) = E_{min}(y)$ and so $E_{min}(y)$ will have degree more than t. □

Lemma 8. *In protocol Single-Verifier, if V is honest and some $\mathsf{AgreeSet}_{(V,\beta)}$ (containing at least $3t+1$ parties) has been generated, then with probability at least $(1-\epsilon)$, there exists a unique bi-variate polynomial, say $\overline{F}(x,y)$, of degree-(d,t), such that the row polynomial $\overline{f_i}(x)$ held by every honest $P_i \in \mathsf{AgreeSet}_{(V,\beta)}$ satisfies $\overline{F}(x,i) = \overline{f_i}(x)$. Moreover, if D is honest then $\overline{F}(x,y) = F(x,y)$.*

PROOF: Without loss of generality, let $\mathsf{AgreeSet}_{(V,\beta)}$ contains the first $3t+1$ parties $P_1, \ldots, P_{3t+1}$. The set $\mathsf{AgreeSet}_{(V,\beta)}$ will contain at least $2t+1$ honest parties and again without loss of generality, let these be the first $2t+1$ parties $P_1, \ldots, P_{2t+1}$. Then from Lemma 7, the existence of $\mathsf{AgreeSet}_{(V,\beta)}$ implies that except with probability $(1-\epsilon)$, the points $\{\overline{f_i}(j) : i \in \{1, \ldots, 2t+1\}\}$ define some polynomial, say $\overline{g_j}(y)$ of degree at most t, for $j = 1, \ldots, n$. Thus, we have $2t+1$ polynomials $\overline{f_1}(x), \ldots, \overline{f_{2t+1}}(x)$, each of degree at most d and n polynomials $\overline{g_1}(y), \ldots, \overline{g_n}(y)$, each of degree at most t, such that $\overline{f_i}(j) = \overline{g_j}(i)$ holds for all $i = 1, \ldots, 2t+1$ and all $j = 1, \ldots, n$. So from Lemma 2, there is a unique bi-variate polynomial, say $\overline{F}(x,y)$, of degree-(d,t), such that $\overline{F}(x,i) = \overline{f_i}(x)$ holds for $i = 1, \ldots, 2t+1$. It is easy to see that if D is honest then $\overline{F}(x,y) = F(x,y)$. □

Lemma 9. *If D is honest then s will remain information theoretically secure in protocol Single-Verifier.*

PROOF: To recover the secret s, the adversary $\mathcal{A}_t$ has to learn the polynomial $F(x,y)$ and this requires $(t+1)(d+1)$ distinct points on $F(x,y)$. Without loss of generality, let $\mathcal{A}_t$ control the first t parties $P_1, \ldots, P_t$. So $\mathcal{A}_t$ learns the row polynomials $f_1(x), \ldots, f_t(x)$. Knowing $f_1(x), \ldots, f_t(x)$ also implies that $\mathcal{A}_t$ learns t distinct points on the column polynomials $g_1(y), \ldots, g_n(y)$ (only $d+1$ of them are independent polynomials), each of degree at most t. So the adversary learns $t(d+1)$ distinct points on $F(x,y)$. The adversary still lacks $(t+1)(d+1) - t(d+1) = (d+1)$ points to uniquely reconstruct $F(x,y)$. We next claim that the polynomials that are made public during the probabilistic checks give no extra information about $F(x,y)$. The adversary $\mathcal{A}_t$ learns the polynomial $E_{(V,\beta)}(y)$, for $\beta = 1, \ldots, t+1$. However, each $E_{(V,\beta)}(y) = m_{(V,\beta)}(y) + r_{(V,\beta)} g_1(y) + \ldots + r^n_{(V,\beta)} g_n(y)$, where $m_{(V,\beta)}(y)$ is the masking polynomial and is independent of $g_1(y), \ldots, g_n(y)$. The adversary will know $r_{(V,\beta)}$ and t points on $m_{(V,\beta)}(y)$, which is of degree at most t and so $\mathcal{A}_t$ cannot uniquely reconstruct $m_{(V,\beta)}(y)$. Thus learning $E_{(V,\beta)}(y)$ adds no new information about $F(x,y)$ to the adversary's view. Moreover, each $E_{(V,\beta)}(y)$ uses an independent masking polynomial of degree at most t. Thus overall, $\mathcal{A}_t$ lacks $(d+1)$ points to uniquely reconstruct $F(x,y)$, implying information theoretic security for $s = F(0,0)$. □

Towards the Computation of CORE: So far, we concentrated on the action that is to be carried out with respect to a single verifier V. We proved that if V is honest then protocol Single-Verifier can provide us with a candidate solution for CORE (Lemma 6-8). Since we do not know the identity of the honest parties, we can not place our confidence on any particular party and ask him to play the role of the verifier. Thus we repeat the protocol Single-Verifier on behalf of every party in $\mathcal{P}$, considering it as a verifier. But again since we do not know the exact identity of the honest verifiers, we can not pick any arbitrary $\mathsf{AgreeSet}_{(\star,\star)}$ as CORE. Thus CORE construction requires additional tricks, which are based on some interesting properties of $\mathsf{AgreeSet}_{(\star,\star)}$, which we prove in the sequel. We first show that if there are two different AgreeSets that are generated with respect to an honest verifier V, then the row polynomials of the honest parties in each $\mathsf{AgreeSet}$ define the same bi-variate polynomial of degree-(d, t) (Lemma 10). We further show that corresponding to two different honest verifiers V_α and V_δ, the row polynomials of the honest parties in $\mathsf{AgreeSet}_{(V_\alpha,\star)}$ and $\mathsf{AgreeSet}_{(V_\delta,\star)}$ also define the same bi-variate polynomial of degree-(d, t) (Lemma 11).

Lemma 10. *Let V be an honest verifier and assume that $\mathsf{AgreeSet}_{(V,\gamma)}$ and $\mathsf{AgreeSet}_{(V,\delta)}$ are generated where $\gamma, \delta \in \{1, \ldots, t+1\}$ and $\mathsf{AgreeSet}_{(V,\gamma)} \neq \mathsf{AgreeSet}_{(V,\delta)}$. Then the row polynomials held by the honest parties in $\mathsf{AgreeSet}_{(V,\gamma)}$, as well as in $\mathsf{AgreeSet}_{(V,\delta)}$, define the same bi-variate polynomial of degree-(d, t).*

PROOF: By Lemma 8, if V is honest, then the row polynomials held by the honest parties in $\mathsf{AgreeSet}_{(V,\gamma)}$ as well as in $\mathsf{AgreeSet}_{(V,\delta)}$ define unique bi-variate polynomials of degree-(d, t), say $\overline{F}(x, y)$ and $\widehat{F}(x, y)$ respectively. Now $\overline{F}(x, y) = \widehat{F}(x, y)$, as there are at least $(t + 1)$ common honest parties in $\mathsf{AgreeSet}_{(V,\gamma)}$ and $\mathsf{AgreeSet}_{(V,\delta)}$, whose row polynomials define a unique bi-variate polynomial of degree-(d, t). $\square$

Lemma 11. *For any two honest verifiers V_α and V_δ, the row polynomials of the honest parties in any $AgreeSet_{(V_\alpha,\star)}$ and $\mathsf{AgreeSet}_{(V_\delta,\star)}$ define the same bi-variate polynomial of degree-(d, t).*

PROOF: The proof again follows from the fact that there will be at least $(t + 1)$ common honest parties in $\mathsf{AgreeSet}_{(V_\alpha,\star)}$ and $\mathsf{AgreeSet}_{(V_\delta,\star)}$, whose row polynomials define a single bi-variate polynomial of degree-(d, t). $\square$

Using Lemma 10 and 11, we suggest to check the presence of CORE as follows: We check whether there is a set of $3t + 1$ parties, who are present in AgreeSets, corresponding to at least $(t + 1)$ verifiers. If so, then such a set of $3t + 1$ parties is considered as CORE. The intuition is that at least one of the $(t + 1)$ verifiers, say V_{hon}, will be honest and if the selected $3t + 1$ parties belong to some $\mathsf{AgreeSet}_{(V_{hon},\star)}$, then indeed the row polynomials of the honest parties in the selected set of $3t+1$ parties lie on a unique bi-variate polynomial of degree-(d, t). This intuition is captured formally in the protocol S-Ver-Agree, presented in Fig. 10.

We now prove the properties of the protocol S-Ver-Agree.

Lemma 12. *In protocol S-Ver-Agree, the following holds:*

1. *The parties A-cast $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits.*
2. *If D is honest, then s will remain information theoretically secure.*
3. *If D is honest then eventually every honest party will agree on a CORE, such that the row polynomials of the honest parties in CORE lie on the bi-variate polynomial $F(x, y)$.*
4. *If D is corrupted and some honest party has accepted a CORE, then every other honest party will also eventually accept the same CORE. Moreover, except with probability ϵ, the row polynomials of the honest parties in CORE will lie on a unique bi-variate polynomial of degree-(d, t).*

PROOF: In protocol S-Ver-Agree, n instances of Single-Verifier are executed. In a single instance of Single-Verifier, the parties may have to do the following communication at most $(t + 1)$ times: A-cast of a random challenge from $\mathbb{F}$ by V; A-cast of the linear combination of its column polynomials and a masking polynomial by D; A-cast of the linear combination of the points on its row polynomial and the share of a masking polynomial by every party P_i. This accounts for a total A-cast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits for a single instance of Single-Verifier and so for n instances, it is $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits.

The information theoretic security for s follows from Lemma 9 and the fact that in each instance of Single-Verifier, independent masking polynomials are used.

Figure 10: Protocol for the Verification & Agreement on CORE phase for the statistical scheme.

Protocol S-Ver-Agree

VERIFICATION AND CORE CONSTRUCTION:

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

1. Acting as a verifier, execute an instance of Single-Verifier.
2. Participate in all the instances of Single-Verifier, executed on the behalf of every verifier $P_j \in \mathcal{P}$.
3. Add a verifier P_α to the set $\mathsf{VerifierSet}_i$ if some $\mathsf{AgreeSet}_{(P_\alpha,\beta)}$, where $\beta \in \{1,\ldots,t+1\}$, is generated in the instance of the Single-Verifier, executed on behalf of the verifier P_α.
4. Check whether $|\mathsf{VerifierSet}_i| \geq t+1$ and if so, then perform the following computation:
   (a) For every $P_\alpha \in \mathsf{VerifierSet}_i$, compute $\mathsf{AgreeSet}_{P_\alpha} = \cup_\beta \mathsf{AgreeSet}_{(P_\alpha,\beta)}$.
   /*This denotes taking the union of all $\mathsf{AgreeSet}_{(P_\alpha,\star)}$, generated in the instance of the Single-Verifier, executed on behalf of the verifier P_α. */
   (b) Compute $\mathsf{CORE}_i = \{P_j \mid P_j$ belongs to $\mathsf{AgreeSet}_{P_\alpha}$ of at least t+1 $P'_\alpha s$ in the $\mathsf{VerifierSet}_i\}$.
   (c) Wait for the new updates (such as the generation of new $\mathsf{AgreeSet}_{(P_k,\star)}$'s, expansion of the existing $\mathsf{AgreeSet}_{(P_k,\star)}$'s, etc.) and repeat the same computation (i.e. steps 2-4((a),(b))) to update CORE_i after every new update.

ii. CODE FOR D: This code is executed only by D

1. A-cast $\mathsf{CORE} = \mathsf{CORE}_D$, as soon as $|\mathsf{CORE}_D| = 3t+1$.

AGREEMENT ON CORE: CODE FOR P_i: Every party executes this code

1. Wait to receive a CORE from the A-cast of D, such that $|\mathsf{CORE}| = 3t+1$.
2. Wait until $\mathsf{CORE} \subseteq \mathsf{CORE}_i$ and then accept CORE and terminate.

If D is honest, then the set of $3t+1$ honest parties will be present in every $\mathsf{AgreeSet}_{(P_\alpha,\star)}$ eventually, corresponding to every verifier P_α. Moreover, every honest verifier will be included in the set $\mathsf{VerifierSet}_i$ of every honest P_i eventually. If D is honest, then D will construct a CORE_D of size $3t+1$ eventually. It then A-casts that set and by the property of the A-cast, it will be received by every honest party. Moreover, every honest P_i will find that $\mathsf{CORE}_D \subseteq \mathsf{CORE}_i$ and will accept it as CORE. It is easy to see that the row polynomials of the honest parties in CORE will define the original polynomial $F(x,y)$ selected by D.

If D is corrupted and some honest P_i has accepted a CORE, then it implies that P_i has received CORE from the A-Cast of D. Moreover, P_i must have found the condition $\mathsf{CORE} \subseteq \mathsf{CORE}_i$ to hold. From the properties of the A-cast, every other honest party P_j will also eventually receive the same CORE from the A-cast of D. Moreover, from the steps for the construction of CORE_i, we find that eventually, $\mathsf{CORE}_i \subseteq \mathsf{CORE}_j$ will hold and so P_j will also find that $\mathsf{CORE} \subseteq \mathsf{CORE}_j$ and hence will accept CORE. We now show that except with probability ϵ, the row polynomials of the honest parties in CORE lie on a unique bi-variate polynomial of degree-(d,t). By Lemma 10, the row polynomials held by the honest parties in $\mathsf{AgreeSet}_{P_\alpha}$ corresponding to an honest verifier P_α, define a unique bi-variate polynomial, say $\overline{F}(x,y)$, of degree-(d,t), with probability at least $(1-\epsilon)$. Next by Lemma 11, the row polynomials held by the honest parties in the union of all the $\mathsf{AgreeSet}_{P_\alpha}$'s, corresponding to the honest P_α's, will also define the same polynomial $\overline{F}(x,y)$ with probability at least $(1-\epsilon)$. By the construction of CORE, every party in CORE is guaranteed to be present in at least one $\mathsf{AgreeSet}_{P_\alpha}$, where the verifier P_α is honest. This implies that the row polynomials held by the honest parties in CORE define $\overline{F}(x,y)$ with probability at least $(1-\epsilon)$. □

In the next section, we present the statistical AVSS scheme for sharing a single value and prove its properties.

3.1.3 Statistical AVSS Scheme for a Single Secret

The sharing protocol SAVSS-Share for the statistical scheme SAVSS is presented in Fig. 11.

Figure 11: Protocol for the sharing phase of the statistical AVSS scheme.

Protocol SAVSS-Share

1. D executes S-Distr.
2. The parties execute S-Ver-Agree.
3. If CORE is generated and agreed upon, then the parties execute Gen.

Theorem 4. *Protocols (SAVSS-Share, Rec) constitute a statistical AVSS scheme that generates d-sharing of s. In SAVSS-Share, the parties privately communicate $\mathcal{O}(n^3 \log(|\mathbb{F}|))$ bits and A-cast $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits. In Rec, the parties privately communicate $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.*

PROOF: If D is honest, then every honest party will eventually terminate SAVSS-Share with his share of the secret s. This follows from Lemma 12(3) and Lemma 3(1). From Lemma 12(4), if D is corrupted and some honest party has accepted a CORE, then every other honest party will accept the same CORE. Moreover, except with probability ϵ, the row polynomials of the honest parties in CORE will lie on a unique bi-variate polynomial of degree-(d, t). So from Lemma 3(1), executing the protocol Gen generates d-sharing. If the honest parties execute Rec, then s will be reconstructed correctly. This follows from the Lemma 4. This proves the correctness and the termination condition.

For secrecy, we have to consider an honest D. Without loss of generality, let $P_1, \ldots, P_t$ be under the control of $\mathcal{A}_t$. From Lemma 12(2) and Lemma 9, by the end of the protocol S-Ver-Agree, $\mathcal{A}_t$ learns $t(d+1)$ distinct points on the polynomial $F(x, y)$ from t row polynomials of degree at most d. At the end of the protocol Gen, the adversary gets the column polynomials $g_1(y), \ldots, g_t(y)$, which provide him t additional points on $F(x, y)$. So in total, $\mathcal{A}_t$ learns $t(d + 1) + t$ distinct points on $F(x, y)$. This implies that $\mathcal{A}_t$ lacks $(t + 1)(d + 1) - t(d + 1) - t = d + 1 - t$ points on $F(x, y)$ to uniquely reconstruct $F(x, y)$. Since $d \geq t$, we obtain information theoretic security for s.

The communication complexity follows from Claim 1, Lemma 12(1), Lemma 3(2) and Lemma 4. $\square$

This marks the end of our discussion on the statistical AVSS scheme for sharing a single secret.

3.2 Sub-Protocols for the Perfect AVSS Scheme

We now present the protocols P-Distr and P-Ver-Agree which are the sub-protocols for our perfect AVSS scheme PAVSS.

3.2.1 Protocol P-Distr

The protocol is similar to the protocol S-Distr with the following differences: D will not share any masking polynomial. Moreover, he will distribute both row, as well as column polynomials to the parties (recall that in S-Distr, only the row polynomials were distributed by D). Protocol P-Distr is presented in Fig. 12.

The following claim about P-Distr trivially follows from the protocol description.

Claim 2. *In protocol P-Distr, D privately communicates $\mathcal{O}((nd + n^2) \log |\mathbb{F}|)$ bits, which is $\mathcal{O}(n^2 \log |\mathbb{F}|)$ for $d \leq 2t$.*

Figure 12: Protocol for the distribution by D phase of the perfect AVSS scheme. Here D is the dealer, s is the secret to be shared and d is the degree of the sharing.

Protocol P-Distr

CODE FOR D: Only D executes this code

1. Select a random bivariate polynomial $F(x, y)$ of degree-(d, t) over $\mathbb{F}$, such that $F(0, 0) = s$. For $i = 0, \ldots, n$, let $f_i(x) \stackrel{def}{=} F(x, i)$ and $g_i(y) \stackrel{def}{=} F(i, y)$.
2. For $i = 1, \ldots, n$, send the row polynomial $f_i(x)$ and the column polynomial $g_i(y)$ to the party P_i.

3.2.2 Protocol P-Ver-Agree

The goal of the protocol P-Ver-Agree is to enable the (honest) parties identify and agree on CORE. Recall that in protocol S-Ver-Agree, several random checks are applied on the row polynomials and the masking polynomials distributed by D to check the presence of CORE and the process involved a negligible error probability. In protocol P-Ver-Agree, we cannot apply such random checks. Instead we proceed as follows: we ask the parties to interact with each other and check the consistency of their common values (on the polynomials received from D). Specifically, every pair (P_i, P_j) of parties check whether $\overline{f_i}(j) = \overline{g_j}(i)$, which should ideally hold, if D, P_i and P_j are honest. Here $\overline{f_i}(x)$ and $\overline{g_i}(y)$ denote the row and column polynomial received by P_i. The parties broadcast `OK` signals if the consistency check passes. Using these signals, we construct a consistency graph with the edges representing pair-wise consistency and check for the presence of an (n, t)-star (see Section 2.3). The intuition is that if D is honest, then eventually every honest party will receive his row and column polynomial, which will be pair-wise consistent with the polynomials of every other honest party and eventually there will be a clique of size at least $(n - t)$ in the consistency graph. So eventually we should find an (n, t)-star in the consistency graph. Let (C, D) be such a star. Our first observation is that the row polynomials of the honest parties in C and the column polynomials of the honest parties in D will be pair-wise consistent and thus they lie on a unique bi-variate polynomial, say $\overline{F}(x, y)$, of degree-(d, t). This is due to Lemma 2 and the fact that there will be at least $(t + 1)$ and $(2t + 1)$ honest parties in C and D respectively. Moreover if D is honest then $\overline{F}(x, y) = F(x, y)$.

The next obvious question is: does the the presence of (C, D) implies the existence of CORE? Recall that we want CORE to be of size $3t + 1$. Clearly C is not qualified to be CORE. On the other hand, even though D is of size $3t + 1$, it cannot be considered as CORE. This is because we want the *row* polynomials of the honest parties in CORE to lie on a unique bi-variate polynomial; whereas an (n, t)-star ensures that the *column* polynomials of the honest parties in D lie on a unique bi-variate polynomial. If we consider D as CORE, then we cannot "complete" the d-sharing by executing the protocol Gen on D. So we cannot directly confirm the presence of CORE from the presence of an (n, t)-star. However, we observe that if indeed the dealer D is honest then there will be "additional" honest parties, apart from the honest parties in C, whose row polynomial will also lie on $\overline{F}(x, y)$. The reason is that we have at least $(3t+1)$ honest parties. We search for these "additional" honest parties using the following two-fold, non-intuitive strategy:

- We first try to "expand" the set D by identifying additional parties not in D whose column polynomial also lie on $\overline{F}(x, y)$. The expanded set, denoted by F, includes all the parties having edges with at least $(2t + 1)$ parties in C in the consistency graph. The parties in D will be automatically in F. It is easy to note that the column polynomial of an honest $P_j \in \mathcal{P} \setminus \mathsf{D}$ satisfying the above condition will lie on $\overline{F}(x, y)$. This is because the honest P_j ensures that its column polynomial has degree at most t and since P_j will have edge with at least $(2t+1)$ parties in C, this implies that its column polynomial is pair-wise consistent with the row polynomial of at least $(t + 1)$ honest parties in C, which lie on $\overline{F}(x, y)$.

- We then try to "expand" the set C. Specifically, we search for the parties P_j, who have edges with at least $(d+t+1)$ parties in F in the consistency graph. The idea is that the row polynomial of such a P_j has degree at most d and out of the $(d+t+1)$ parties in F (with whom P_j has an edge), at least $(d+1)$ will be honest. Thus, the row polynomial of P_j will be pair-wise consistent with the column polynomials of at least $(d+1)$ honest parties in F, which lie on $\overline{F}(x,y)$. So the row polynomial of P_j will lie on $\overline{F}(x,y)$. We include all such P_j's in a set E. Notice that all the parties in C will be included in E.

If we find E to be of size $3t+1$, then E is taken as CORE. It is easy to see that indeed the row polynomials of all the honest parties in E will lie on $\overline{F}(x,y)$. However there is a subtle issue. In the above approach, the honest parties may have to wait indefinitely for the "expansion" of D and C sets until E admits a size of $3t+1$. Consider the case when $d=2t$ and C and D are exactly of size $(2t+1)$ and $(3t+1)$ respectively, such that they contain t corrupted parties. If the corrupted parties in C choose to be inconsistent with the parties outside D, then the honest parties outside D will have only $(t+1)$ neighbours in C and will not be included in the set F. So $\mathsf{F}=\mathsf{D}$. Similarly, if the corrupted parties in F choose to be inconsistent with the parties outside C, then the honest parties outside C will have only $(2t+1)$ neighbours in F and will be never included in the set E. So it is possible that C may never expand from its initial size of $2t+1$.

To deal with the above situation, we carefully look into the properties of the consistency graph and the algorithm Find-STAR. We observe that if D is honest then eventually all honest parties (at least $3t+1$) will be consistent with each other and there will be a clique in the consistency graph involving all the honest parties. We further note that if the Find-STAR algorithm is executed on "this" graph, containing a clique of size at least $3t+1$ involving the honest parties, then C component of the obtained (n,t)-star will have at least $2t+1$ honest parties. When C contains at least $2t+1$ honest parties, then eventually the set D will expand to set F, which will contain all the $3t+1$ honest parties and eventually the set C will expand to the set E containing at least $3t+1$ parties. This crucial observation is at the heart of protocol P-Ver-Agree. However, it is difficult to identify an instance of the consistency graph that contains a clique involving at least $3t+1$ honest parties. This problem is eliminated by repeating the star finding process and expansion of C and D for every instance of the consistency graph. In a more detail, after every update in the consistency graph (on receiving new `OK` signals), we check for the presence of a new (n,t)-star in the graph (which was not found earlier) along with the corresponding F and E sets and update the existing F and E sets (corresponding to all the previously generated (n,t)-stars). This is continued till we find an instance of E of size $3t+1$. Such an E will be considered as CORE. Surely if D is honest, then we will get E with the desired size. We let D to moderate these repetitions by asking it to repeat the star finding process and expansion of C and D for every instance of the consistency graph. Upon finding the CORE, the dealer D makes all the parties agree on CORE by broadcasting the star and the corresponding E and F sets. The parties then verify if the broadcasted star and the sets are "valid" with respect to their local consistency graph.

This process of repetition after every update in the consistency graph is some what analogous to the situation in the protocol S-Ver-Agree, where we have to keep expanding ReceivedSet till the "appropriate" conditions are satisfied. However, unlike the protocol S-Ver-Agree, where each repetition requires communication, in protocol P-Ver-Agree, each repetition requires only local computation by the parties. The communication is required finally to make an agreement when CORE is found by D.

With the above intuition in mind, we present the protocol P-Ver-Agree in Fig. 13. In the protocol, the pair $(\mathsf{C}_\beta, \mathsf{D}_\beta)$ denotes the β^{th} instance of an (n,t)-star and the pair $(\mathsf{E}_\beta, \mathsf{F}_\beta)$ denotes the corresponding E and F sets respectively. After every update in the consistency graph, $(\mathsf{E}_\beta, \mathsf{F}_\beta)$ may be updated. In the sequel, we will show that there will be finite number of instances of (n,t)-star (and the corresponding E and F sets) that can result from the consistency graph. We will also prove the three key observations on which the protocol P-Ver-Agree is based upon:

1. If D is honest, then eventually some (n,t)-star $(\mathsf{C}_\beta, \mathsf{D}_\beta)$ will be generated, where C_β will contain

at least $2t + 1$ honest parties (Lemma 14). This crucial observation is at the heart of the protocol P-Ver-Agree.

2. If D is honest and the C component of an (n, t)-star $(\mathsf{C}_\beta, \mathsf{D}_\beta)$ contains at least $2t + 1$ honest parties, then CORE will be eventually generated from $(\mathsf{C}_\beta, \mathsf{D}_\beta)$ (Lemma 15).

3. For any (n, t)-star $(\mathsf{C}_\beta, \mathsf{D}_\beta)$, the row polynomials of the honest parties in C_β define a unique bi-variate polynomial of degree-(d, t), irrespective of D (Lemma 13). Moreover, if CORE is generated from this $(\mathsf{C}_\beta, \mathsf{D}_\beta)$, then the row polynomials of the honest parties in CORE define the same bi-variate polynomial (Lemma 17).

Figure 13: Protocol for the Verification & Agreement on CORE phase for the perfect AVSS scheme.

Protocol (P-Ver-Agree)

i. CODE FOR P_i: Every party (including D) executes this code.

1. Wait to receive the row polynomial $\overline{f_i}(x)$ of degree at most d and the column polynomial $\overline{g_i}(y)$ of degree at most t from D. Upon receiving, send $\overline{f_{ij}} = \overline{f_i}(j)$ and $\overline{g_{ij}} = \overline{g_i}(j)$ to the party P_j, for $j = 1, \ldots, n$.
2. Upon receiving $\overline{f_{ji}}$ and $\overline{g_{ji}}$ from P_j, check if $\overline{f_i}(j) \stackrel{?}{=} \overline{g_{ji}}$ and $\overline{g_i}(j) \stackrel{?}{=} \overline{f_{ji}}$. If both the equalities hold, then A-cast the signal $\texttt{OK}(P_i, P_j)$.
3. Construct the undirected consistency graph G_i with $\mathcal{P}$ as the vertex set. Add an edge (P_j, P_k) in G_i upon receiving the $\texttt{OK}(P_k, P_j)$ signal and the $\texttt{OK}(P_j, P_k)$ signal from the A-cast of P_k and P_j respectively.

ii. CODE FOR D (FOR GENERATING CORE). Only D executes this code: Let G_D denote the consistency graph constructed by D.

1. After every new receipt of some $\texttt{OK}(\star, \star)$ signal, 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) If an (n, t)-star is found from the current execution of Find-STAR($\overline{G_D}$) that is distinct from all the previous α (n, t)-stars (obtained earlier), do the following:
      i. Call the new (n, t)-star as $(\mathsf{C}_{\alpha+1}, \mathsf{D}_{\alpha+1})$.
      ii. Construct a set $\mathsf{F}_{\alpha+1}$ as follows: Add P_j to $\mathsf{F}_{\alpha+1}$ if P_j has at least $2t + 1$ neighbours in $\mathsf{C}_{\alpha+1}$ in G_D.
      iii. Construct a set $\mathsf{E}_{\alpha+1}$ as follows: Add P_j to $\mathsf{E}_{\alpha+1}$ if P_j has at least $d + t + 1$ neighbours in $\mathsf{F}_{\alpha+1}$ in G_D.
      iv. For $\beta = 1, \ldots, \alpha$, update the existing F_β and E_β as follows:
         A. Add P_j to F_β, if $P_j \notin \mathsf{F}_\beta$ and P_j has now at least $2t + 1$ neighbours in C_β in G_D.
         B. Add P_j to E_β, if $P_j \notin \mathsf{E}_\beta$ and P_j has now at least $d + t + 1$ neighbours in F_β in G_D.
   (b) If an (n, t)-star that has been already found in the past is obtained, then execute the step (a).iv(A-B) to update the existing F_β's and E_β's.

   Let $(\mathsf{E}_\gamma, \mathsf{F}_\gamma)$ be the first pair among the generated $(\mathsf{E}_\beta, \mathsf{F}_\beta)$'s such that $|\mathsf{E}_\gamma| \geq 3t + 1$ and $|\mathsf{F}_\gamma| \geq 3t + 1$. Assign CORE $= \mathsf{E}_\gamma$, A-cast $((\mathsf{C}_\gamma, \mathsf{D}_\gamma), (\mathsf{E}_\gamma, \mathsf{F}_\gamma))$ and terminate.

iii. CODE FOR P_i (FOR VERIFYING THE CORE): Every party (including D) executes this code.

1. Wait to receive $((\mathsf{C}_\gamma, \mathsf{D}_\gamma), (\mathsf{E}_\gamma, \mathsf{F}_\gamma))$ from the A-cast of D, such that $|\mathsf{E}_\gamma| \geq 3t + 1$ and $|\mathsf{F}_\gamma| \geq 3t + 1$.
2. Wait until $(\mathsf{C}_\gamma, \mathsf{D}_\gamma)$ becomes an (n, t)-star in the consistency graph G_i. For this, wait to receive the corresponding $\texttt{OK}$ signals from the parties in C_γ and D_γ.
3. Wait until every party $P_j \in \mathsf{F}_\gamma$ has at least $2t + 1$ neighbours in C_γ in G_i.
4. Wait until every party $P_j \in \mathsf{E}_\gamma$ has at least $d + t + 1$ neighbours in F_γ in G_i.

Once the above conditions are satisfied, accept CORE $= \mathsf{E}_\gamma$ and terminate.

We now prove the properties of the protocol P-Ver-Agree.

Lemma 13. *Let* (C, D) *be any* (n,t)-**star** *in the consistency graph* G_k *of an honest* P_k. *Then the row polynomials held by the honest parties in* C *define a unique bivariate polynomial, say* $\overline{F}(x,y)$*, of degree-*(d,t)*, such that* $\overline{F}(x,i) = \overline{f_i}(x)$ *and* $\overline{F}(j,y) = \overline{g_j}(y)$ *will hold for every honest* P_i *and* P_j *in* C *and* D *respectively. Moreover, if* D *is honest then* $\overline{F}(x,y) = F(x,y)$.

PROOF: For any (n,t)-star (C, D), we know that $|\mathsf{C}| \geq n - 2t$ and $|\mathsf{D}| \geq n - t$. So C and D contains at least $n - 3t \geq t + 1$ and $n - 2t \geq 2t + 1$ honest parties, respectively. Moreover, every honest party P_i in C will be pair-wise consistent with every honest party in D. That is, $\overline{f_i}(j) = \overline{g_j}(i)$ and $\overline{f_j}(i) = \overline{g_i}(j)$ will hold for every honest $P_i \in \mathsf{C}$ and every honest $P_j \in \mathsf{D}$. Furthermore, the row and column polynomials of the honest parties will have degree at most d (where $d \leq 2t$) and t respectively. The proof now follows from Lemma 2. It is very easy to see that if D is honest, then $\overline{F}(x,y) = F(x,y)$. □

The next two lemmas are very crucial as they show that if D is honest then eventually every honest party will agree on CORE and terminate the protocol P-Ver-Agree.

Lemma 14. *If* D *is honest then eventually an* (n,t)-**star** $(\mathsf{C}_\beta, \mathsf{D}_\beta)$ *will be generated by* D*, such that* C_β *will contain at least* $2t+1$ *honest parties.*

PROOF: If D is honest then eventually the edges between each pair of honest parties will vanish in the complementary graph $\overline{G_D}$. So each edge in $\overline{G_D}$ will be eventually either (a) between an honest and a corrupted party OR (b) between two corrupted parties. Moreover, the set of honest parties will form an independent set of size at least $(n-t)$. Let $(\mathsf{C}_\beta, \mathsf{D}_\beta)$ be the (n,t)-star which is obtained while applying the Find-STAR algorithm on $\overline{G_D}$, when $\overline{G_D}$ contains edges of only the above two types. Now, by the construction of C_β (see Algorithm Find-STAR), it excludes the parties in N (the set of parties that are associated with the maximum matching M) and T (the set of parties that are associated with the triangle-heads). 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 we are considering the instance when $\overline{G_D}$ does not have any 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_i, 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 C_β, at least one (if P_i belongs to N, then one; if P_i belongs to T, then two) corrupted party also remains outside C_β. As there are at most t corrupted parties, C_β may exclude at most t honest parties. So C_β is bound to contain at least $2t+1$ honest parties.

To complete the proof, we now have to show that $\overline{G_D}$ will contain the edges of the above two types after finite number of steps. We observe that an honest D may compute $\mathcal{O}(n^2)$ distinct (n,t)-stars in G_D. This is because D applies Find-STAR on $\overline{G_D}$ every time when an edge is added to G_D and we know that there can be $\mathcal{O}(n^2)$ edges in G_D. Now $(\mathsf{C}_\beta, \mathsf{D}_\beta)$ with C_β containing at least $2t+1$ honest parties will occur among these $\mathcal{O}(n^2)$ (n,t)-stars. □

Lemma 15. *In protocol* ***P-Ver-Agree****, if* D *is honest, then eventually* CORE *will be generated by* D *and every honest party will accept* CORE *and terminate the protocol* ***P-Ver-Agree****.*

PROOF: By Lemma 14, if D is honest then eventually it will obtain an (n,t)-star $(\mathsf{C}_\beta, \mathsf{D}_\beta)$ in the consistency graph G_D, such that C_β will contain at least $2t+1$ honest parties. Moreover, every honest party will eventually have an edge with every other honest party in G_D. So every honest party in $\mathcal{P}$ will eventually have an edge with all the honest parties in C_β. This implies that every honest party in $\mathcal{P}$ will eventually have at least $2t+1$ neighbours in C_β and so they will be included in F_β. Following similar argument, every honest party in $\mathcal{P}$ will eventually have at least $d+t+1$ neighbours in F_β and so they will be included in E_β. So D will find that $|\mathsf{E}_\beta| \geq (3t+1)$ and $|\mathsf{F}_\beta| \geq (3t+1)$ and will take E_β as CORE and A-cast $((\mathsf{C}_\beta, \mathsf{D}_\beta), (\mathsf{E}_\beta, \mathsf{F}_\beta))$ and terminate. By the property of the A-cast, every honest party P_i will receive these sets correctly from D and will eventually find that $(\mathsf{C}_\beta, \mathsf{D}_\beta)$ is an (n,t)-star in the consistency graph G_i. This is because if an honest D has included the edges between the parties in C_β and D_β in his consistency graph G_D, then the same edges will also be eventually included by every honest P_i in his consistency graph G_i. Due to the same reason, an honest P_i will find that every party in F_β has at least $(2t+1)$ neighbours in C_β in the graph G_i and similarly, every party in E_β has at least $d+t+1$ neighbours in F_β in the graph G_i eventually. So P_i will accept CORE and terminate the protocol P-Ver-Agree. $\square$

The previous two lemmas ascertained that the honest parties will eventually terminate the protocol P-Ver-Agree if D is honest. The next lemma shows that even if D is corrupted and some honest party has terminated the protocol P-Ver-Agree then every honest party will also eventually do the same.

Lemma 16. *If D is corrupted and some honest party P_i terminates the protocol P-Ver-Agree after accepting CORE, then every other honest party P_j will also eventually do the same.*

PROOF: If an honest P_i has accepted CORE, then this implies that it has received $((\mathsf{C}_\gamma, \mathsf{D}_\gamma), (\mathsf{E}_\gamma, \mathsf{F}_\gamma))$ from the A-cast of D and verified the following in his consistency graph G_i: (a) $(C_\gamma, \mathsf{D}_\gamma)$ is an (n,t)-star; (b) every party in F_γ has at least $(2t+1)$ neighbors in C_γ and (c) every party in E_γ has at least $(d+t+1)$ neighbors in F_γ. From the properties of the A-cast, every other honest party P_j will also receive the same $((\mathsf{C}_\gamma, \mathsf{D}_\gamma), (\mathsf{E}_\gamma, \mathsf{F}_\gamma))$ from the A-cast of D. Moreover, P_j will also find that eventually the above three conditions are also met in his consistency graph G_j. So P_j will also eventually accept CORE and terminate the protocol P-Ver-Agree. $\square$

The next lemma shows that if a CORE is generated then indeed the row polynomials of the honest parties in CORE define a unique bi-variate polynomial of degree-(d,t).

Lemma 17. *If an honest P_i has accepted CORE, then the row polynomials 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 $\overline{F}(x,y) = F(x,y)$.*

PROOF: If an honest P_i has accepted CORE, then it implies that he has received $((\mathsf{C}_\gamma, \mathsf{D}_\gamma), (\mathsf{E}_\gamma, \mathsf{F}_\gamma))$ from the A-cast of D and checked their validity with respect to his own consistency graph G_i. This means that $(\mathsf{C}_\gamma, \mathsf{D}_\gamma)$ is an (n,t)-star in G_i. Lemma 13 implies that the row polynomials of the honest parties in C_γ define a unique bivariate polynomial, say $\overline{F}(x,y)$, of degree-(d,t). Recall that E_γ is obtained by expanding the set C_γ. To complete the proof we need to show that even the row polynomials of the honest parties in $\mathsf{E}_\gamma \setminus \mathsf{C}_\gamma$ lie on $\overline{F}(x,y)$. We do so in two stages: we first claim that the column polynomial $\overline{g_j}(y)$ of every honest P_j in F_γ lie on $\overline{F}(x,y)$. This is because by the construction of F_γ, every honest $P_j \in \mathsf{F}_\gamma$ has at least $2t+1$ neighbors in C_γ, which implies that $\overline{f_{kj}} = \overline{f_k}(j) = \overline{g_j}(k)$ for at least $2t+1$ P_k's in C_γ. Moreover the degree of $\overline{g_j}(y)$ is at most t. Now out of the $(2t+1)$ P_k's in C_γ (with whom P_j has an edge), at least $(t+1)$ are honest. Also the row polynomials of those P_k's lie on $\overline{F}(x,y)$. This clearly implies that $\overline{g_j}(y) = \overline{F}(j,y)$.

Next we claim that the row polynomial $\overline{f_j}(x)$ of every honest party $P_j \in \mathsf{E}_\gamma$ also lies on $\overline{F}(x,y)$. By the construction of E_γ, every such P_j has at least $d+t+1$ neighbors in F_γ, which means that $\overline{f_j}(k) = \overline{g_{kj}} = \overline{g_k}(j)$ for at least $(d+t+1)$ P_k's in F_γ. Moreover the degree of $\overline{f_j}(x)$ is at most d. Now out of the $(d+t+1)$ P_k's in F_γ (with whom P_j has an edge), at least $(d+1)$ are honest. Also the column polynomials of those P_k's lie on $\overline{F}(x,y)$. This clearly implies that $\overline{f_j}(x) = \overline{F}(x,j)$. Hence the row polynomials of the honest parties in CORE define $\overline{F}(x,y)$. $\square$

The next two lemmas are related to the privacy and the communication complexity of the protocol P-Ver-Agree.

Lemma 18. *In protocol P-Ver-Agree if D is honest then s will remain information theoretically secure.*

PROOF: Without loss of generality, let $P_1, \ldots, P_t$ be under the control of the adversary. So $\mathcal{A}_t$ will know the row polynomials $f_1(x), \ldots, f_t(x)$ and the column polynomials $g_1(y), \ldots, g_t(y)$. The adversary will also receive from the honest parties the common points on their row and column polynomials. However, these points do not add any new information to the view of the adversary about $F(x,y)$, as they can be computed from the knowledge of $f_1(x), \ldots, f_t(x), g_1(y), \ldots, g_t(y)$. The adversary is completely oblivious of the communication done between the honest parties. So he has no information about the common points exchanged between the honest parties. The knowledge of CORE does not add any information about $F(x, y)$ to the view of the adversary. So overall, $\mathcal{A}_t$ has $f_1(x), \ldots, f_t(x), g_1(y), \ldots, g_t(y)$. From these polynomials, he obtains $t(d+1)+t$ distinct points on $F(x, y)$. However $F(x, y)$ is of degree-(d, t). So the adversary lacks $(t+1)(d+1) - t(d+1) - t = d+1-t$ points to uniquely recover $F(x, y)$. This implies information theoretic security for the secret s. □

Lemma 19. *Protocol P-Ver-Agree requires a private communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits and A-cast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.*

PROOF: In the protocol, the parties privately exchange the common points on their row and column polynomials, which requires a private communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. In addition, the parties also A-cast $\mathtt{OK}(\star, \star)$ signals, which requires A-cast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. Furthermore, the A-casting of $((\mathsf{C}_\gamma, \mathsf{D}_\gamma), (\mathsf{E}_\gamma, \mathsf{F}_\gamma))$ requires A-cast communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.

In the next section, we present the perfect AVSS scheme and prove its properties.

3.2.3 Perfect AVSS Scheme for a Single Secret

The sharing protocol PAVSS-Share for the perfect scheme PAVSS is presented in Fig. 14.

Figure 14: Protocol for the sharing phase of the perfect AVSS scheme

Protocol PAVSS-Share

1. D executes P-Distr.
2. The parties execute P-Ver-Agree.
3. If CORE is generated and agreed upon, then the parties execute Gen and terminate.[a]

[a] We note that in PAVSS-Share the parties in CORE need not have to communicate the values on their column polynomials during the protocol Gen (unlike in SAVSS-Share). This is because the parties already exchange the common values on their row and column polynomials during the protocol P-Ver-Agree. So once CORE is identified, every party can apply the OEC on the values received from the parties in CORE and reconstruct their share of the secret as described in the protocol Gen. On the contrary, in protocol S-Ver-Agree, the parties were not provided with their column polynomials and so to reconstruct their column polynomials, the parties in CORE need to communicate values to the parties in the protocol Gen, after CORE is identified.

Theorem 5. *Protocols (PAVSS-Share, Rec) constitute a perfect AVSS scheme, which generates d-sharing of s. During PAVSS-Share, the parties privately communicate $\mathcal{O}(n^2 \log(|\mathbb{F}|))$ bits and A-cast $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. During Rec, the parties privately communicate $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.*

PROOF: If D is honest then every honest party will terminate the protocol PAVSS-Share. This follows from the Lemma 15 and Lemma 3(1). If D is corrupted but some honest party terminates PAVSS-Share, then every other honest party will also eventually terminate the protocol PAVSS-Share. This follows from Lemma 16 and Lemma 3(1). Moreover, if the honest parties invoke the protocol Rec, then every honest party will eventually terminate Rec. This follows from the Lemma 4. This completes the proof of termination.

If D is honest then at the end of PAVSS-Share, every honest party will have his share of s. This follows from Lemma 17 and Lemma 3(1). Moreover, every honest party on terminating the protocol Rec will output s. This follows from Lemma 4. On the other hand, even if D is corrupted and some honest party terminates PAVSS-Share, then it implies that CORE is generated and agreed upon, which from Lemma 17 further implies that D has committed the polynomial $\overline{F}(x,y)$ and hence the value $\overline{s} = \overline{F}(0,0)$ to the honest parties in CORE. The property of Gen (Lemma 3(1)) ensures that every honest party will have the share of $\overline{s}$. Moreover, the honest parties on terminating the Rec will output $\overline{s}$. This proves the correctness property.

Information theoretic security of s for an honest D follows from the Lemma 18. Finally the communication complexity follows from Claim 2, Lemma 19, Lemma 3(2) and Lemma 4. □

This completes our discussion on the AVSS schemes for sharing a single secret.

4 AVSS for Sharing Multiple Secrets

The AVSS schemes that we discussed so far allow to d-share a single element from $\mathbb{F}$. Now consider a situation where we have to d-share $S = (s_1, \ldots, s_\ell) \in \mathbb{F}^\ell$, where $\ell > 1$ (indeed in our AMPC protocols, every party has to share multiple values). One simple way to d-share S is to individually d-share each $s_l \in S$ by executing an instance of SAVSS (resp. PAVSS). This will require a communication complexity which is ℓ times the communication complexity of SAVSS (resp. PAVSS). We now show how to d-share all the elements of S simultaneously, such that the private communication depends on ℓ but the A-cast communication is independent of ℓ. Since the A-cast is an expensive protocol[5], we save a lot of communication in our AMPC protocols by using our new AVSS schemes for sharing multiple secrets together.

The main idea behind making the A-cast communication independent of ℓ is the following: we observe that in the sub-protocols dealing with a single secret, the steps which involve private communication among the parties can be extended in a "natural" way to deal with ℓ values. For example, instead of taking a single bi-variate polynomial, D now selects ℓ such polynomials and accordingly every party receives ℓ row and column polynomials. However, we need not have to extend the steps involving broadcast in the same way to deal with ℓ secrets. Instead, those steps can be "modified" to deal with all the ℓ values *concurrently* to keep the A-cast communication independent of ℓ. In the sequel we elaborate on this. We do not present the complete protocols, as this calls for un-necessary repetition; instead we only discuss the key steps that are modified in the earlier sub-protocols for a single value to deal with ℓ values. We also do not present the proofs for the new sub-protocols, as they trivially follow from the properties of the sub-protocols dealing with a single value. The new sub-protocols have "MS" in their names, indicating that they deal with multiple secrets. We first discuss the sub-protocols for the statistical scheme.

4.1 Sub-Protocols for the Statistical Scheme to Share ℓ Values

The statistical scheme is called SAVSS-MS, which consists of the protocol SAVSS-MS-Share for the sharing phase and protocol Rec-MS for the reconstruction phase (this protocol is also the protocol for the reconstruction phase of the perfect scheme for ℓ values). Now the sharing protocol SAVSS-MS-Share consists of a sequence of three stages (similar to the protocol SAVSS-Share), each implemented by a specific sub-protocol discussed below:

1. *Protocol* S-MS-Distr: This protocol implements the distribution by D phase. Here for each $s_l \in S$, the dealer D selects a random bi-variate polynomial $F_l(x,y)$ of degree-(d,t) with the constant term as s_l and distributes the i^{th} row polynomial $f_{l,i}(x) = F_l(x,i)$ to P_i. Thus each P_i receives ℓ row polynomials. In addition, D shares $(t+1)n$ masking polynomials, each of degree at most t, as in the protocol S-Distr (Fig.

[5]The best known perfect A-cast protocol is due to [14]. It communicates $\mathcal{O}(n^2)$ bits to broadcast a single bit.

8). D does not distribute the column polynomials $g_{l,i}(y) = F_l(i, y)$ to P_i as in the protocol S-Distr.

2. *Protocol* S-MS-Ver-Agree: This protocol allows the parties to verify the presence of CORE and to agree on a CORE of size at least $3t + 1$ if it exists, where CORE has the following property: for $l = 1, \ldots, \ell$, the row polynomials $\{\overline{f_{l,i}}(x) : P_i \in \text{CORE} \text{ and } P_i \text{ is honest}\}$ define a unique bi-variate polynomial, say $\overline{F_l}(x, y)$, of degree-(d, t). Moreover, if D is honest then $\overline{F_l}(x, y) = F_l(x, y)$. Here $\overline{f_{l,i}}(x)$ denotes the row polynomials received by P_i from D. The protocol uses another sub-protocol Single-MS-Verifier as a black-box. This protocol is almost the same as the protocol Single-Verifier (Fig. 9) with the following modifications: In step i, P_i waits to receive ℓ row polynomials $\overline{f_{1,i}}(x), \ldots, \overline{f_{\ell,i}}(x)$, each of degree at most d from D. In step iii, D broadcasts the linear combination of ℓn column polynomials (instead of n column polynomials) and a masking polynomial. Specifically, D broadcasts $E_{(V,\beta)}(y)$, where

$$E_{(V,\beta)}(y) \stackrel{def}{=} r^0_{(V,\beta)} m_{(V,\beta)}(y) + r^1_{(V,\beta)} g_{1,1}(y) + \ldots + r^n_{(V,\beta)} g_{1,n}(y) + \ldots + r^{(\ell-1)n+1}_{(V,\beta)} g_{\ell,1}(y) + \ldots + r^{\ell n}_{(V,\beta)} g_{\ell,n}(y).$$

Here $g_{l,i}(y) = F_l(i, y)$. Accordingly, in step iv.1, party P_i will broadcast a linear combination of the share of a masking polynomial and n points on each of his ℓ row polynomial. Specifically, P_i broadcasts $e_{(V,\beta,i)}$, where

$$e_{(V,\beta,i)} \stackrel{def}{=} r^0_{(V,\beta)} \overline{m}_{(V,\beta)}(i) + r^1_{(V,\beta)} \overline{f_{1,i}}(1) + \ldots + r^n_{(V,\beta)} \overline{f_{1,i}}(n) + \ldots + r^{(\ell-1)n+1}_{(V,\beta)} \overline{f_{\ell,i}}(1) + \ldots + r^{\ell n}_{(V,\beta)} \overline{f_{\ell,i}}(n).$$

The rest of the steps for the protocol Single-MS-Verifier are same as in the protocol Single-Verifier. Now protocol S-MS-Ver-Agree is exactly the same as protocol S-Ver-Agree(Fig. 10), except that all instances of Single-Verifier in S-Ver-Agree are now replaced with the instances of Single-MS-Verifier.

3. *Protocol* Gen-MS: If a CORE is generated and agreed upon then this protocol is invoked to complete the d-sharing of the secrets in S. This protocol is a simple extension of the protocol Gen (Fig. 6): each party P_i in CORE sends the j^{th} points $\overline{f_{1,i}}(j), \ldots, \overline{f_{\ell,i}}(j)$ on his row polynomials to P_j, who then applies the OEC on these points to reconstruct the column polynomials $\overline{g_{1,j}}(y), \ldots, \overline{g_{\ell,j}}(y)$ and hence the share $Sh_{l,j} = \overline{g_{l,j}}(0)$ of $s_l \in S$, for $l = 1, \ldots, \ell$.

The reconstruction protocol Rec-MS is a straight forward extension of the protocol Rec (Fig. 7), where for every $s_l \in S$, every party P_i simply sends his share $Sh_{l,i}$ of s_l to every other party P_j and then by applying the OEC, every party reconstructs s_l. We now state the following theorem which follows from the properties of the statistical scheme for sharing a single secret.

Theorem 6. *Protocols (SAVSS-MS-Share, Rec-MS) constitute a statistical AVSS scheme SAVSS-MS, which generates d-sharing of $S = (s_1, \ldots, s_\ell)$. In SAVSS-MS-Share, the parties privately communicate $\mathcal{O}(\ell n d + n^3 \log(|\mathbb{F}|)) = \mathcal{O}(\ell n^2 + n^3 \log(|\mathbb{F}|))$ bits and A-cast $\mathcal{O}(n^3 \log(|\mathbb{F}|))$ bits. During Rec-MS, the parties privately communicate $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits.*

We next discuss the sub-protocols for the perfect AVSS scheme to share ℓ values.

4.2 Sub-Protocols for the Perfect Scheme to Share ℓ Values

The extension of the perfect scheme PAVSS to PAVSS-MS is very simple. PAVSS-MS consists of the protocol PAVSS-MS-Share for the sharing phase and protocol Rec-MS (discussed in the previous section) for the reconstruction phase. Now the sharing protocol PAVSS-MS-Share consists of a sequence of three stages (similar to the protocol PAVSS-Share), each implemented by a specific sub-protocol described below:

1. *Protocol* P-MS-Distr: This protocol implements the distribution by D phase. Here for each $s_l \in S$, the dealer D selects a random bi-variate polynomial $F_l(x, y)$ of degree-(d, t) with s_l as the constant term and distributes the i^{th} row polynomial $f_{l,i}(x) = F_l(x, i)$ and the i^{th} column polynomial $g_{l,i}(y) = F_l(i, y)$ to P_i. Thus each P_i receives ℓ row and column polynomials.

2. *Protocol* P-MS-Ver-Agree: This protocol allows the parties to agree on a CORE and it is almost same as the protocol P-Ver-Agree (Fig. 13), except that step i is extended to deal with ℓ values as follows: first, each P_i waits to receive ℓ row polynomials $\overline{f_{1,i}}(x), \ldots, \overline{f_{\ell,i}}(x)$, each of degree at most d and ℓ column polynomials $\overline{g_{1,i}}(y), \ldots, \overline{g_{\ell,i}}(y)$, each of degree at most t from D. After receiving, P_i proceeds to check the pair-wise consistency of ℓ row and ℓ column polynomials with each P_j. Specifically, P_i sends ℓ values $\overline{f_{l,i,j}} = \overline{f_{l,i}}(j)$, for $l = 1, \ldots, \ell$ on his row polynomials and another ℓ values $\overline{g_{l,i,j}} = \overline{g_{l,i}}(j)$, for $l = 1, \ldots, \ell$ on his column polynomials to P_j. Now on receiving the ℓ values $\overline{f_{l,j,i}}$, for $l = 1, \ldots, \ell$ and the ℓ values $\overline{g_{l,j,i}}$, for $l = 1, \ldots, \ell$ from P_j, party P_i checks $\overline{f_{l,i}}(j) \stackrel{?}{=} \overline{g_{l,j,i}}$ and $\overline{g_{l,i}}(j) \stackrel{?}{=} \overline{f_{l,j,i}}$, for all $l = 1, \ldots, \ell$. If the test passes for every $l = 1, \ldots, \ell$, then P_i A-cast the signal $\texttt{OK}(P_i, P_j)$. The rest of the steps for P-MS-Ver-Agree will be now same as in the protocol P-Ver-Agree.

3. *Protocol* Gen-MS: This protocol is the same as discussed in the previous section. The footnote mentioned in the protocol PAVSS-Share (see the footnote in Fig. 14) applies here as well. That is, the parties in CORE are not required to communicate in the protocol Gen-MS in the perfect AVSS scheme.

We now state the following theorem that follows from the properties of the perfect scheme for sharing a single secret.

Theorem 7. *Protocols (PAVSS-MS-Share, Rec-MS) constitute a perfect AVSS scheme PAVSS-MS, which generates d-sharing of $S = (s_1, \ldots, s_\ell)$. In PAVSS-MS-Share, the parties privately communicate $\mathcal{O}(\ell n^2 \log(|\mathbb{F}|))$ bits and A-cast $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. During Rec-MS, the parties privately communicate $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits.*

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

Once we have an AVSS scheme that can d-share ℓ values for any given d, where $d \leq 2t$, generating $(t, 2t)$-sharing of ℓ values can be done using the following simple idea (outlined earlier in the introduction): To $(t, 2t)$-share $S = (s_1, \ldots, s_\ell)$, the dealer D first t-share S. In addition, he also $(2t-1)$-share ℓ random values, denoted by $R = (r_1, \ldots, r_\ell)$. This implies that each s_l and r_l is shared through polynomials, say $f_l(x)$ and $g_l(x)$, of degree at most t and $(2t-1)$ respectively, with every honest party holding its shares $f_l(i)$ and $g_l(i)$ of s_l and r_l respectively. Now consider the polynomial $h_l(x) = f_l(x) + x \cdot g_l(x)$. It has degree at most $2t$ with the constant term as s_l. Moreover, every party can locally compute $h_l(i) = f_l(i) + i \cdot g_l(i)$. It is easy to see that each s_l is $(t, 2t)$-shared through the polynomials $f_l(x)$ and $h_l(x)$. To implement this idea, the dealer has to invoke two instances of the sharing phase of our AVSS scheme (dealing with ℓ values). Now depending upon whether he invokes the statistical AVSS scheme SAVSS-MS-Share or the perfect scheme PAVSS-MS-Share, the resulting protocol will either have a negligible error or no error in the correctness and in the termination. We call the resulting protocols as S-(t, 2t)-Share and P-(t, 2t)-Share respectively. We present the protocol in Fig. 15.

We now state the properties of the protocol S-(t,2t)-Share and P-(t,2t)-Share, that follow from the properties of the protocols SAVSS-MS-Share and PAVSS-MS-Share respectively.

Theorem 8. *Protocol S-(t,2t)-Share achieves the following properties:*

1. *Termination: (a) If D is honest, then all honest parties will eventually terminate S-(t,2t)-Share.*

Figure 15: Protocol for generating $(t, 2t)$-sharing of $S = (s_1, \ldots, s_\ell)$.

Protocol S-(t,2t)-Share / P-(t,2t)-Share

CODE FOR D (FOR SHARING S): Only D executes this code

1. Select $R = (r_1, \ldots, r_\ell) \in \mathbb{F}^\ell$ uniformly and randomly. In S-(t,2t)-Share, invoke two instances of SAVSS-MS-Share to t-share and $(2t-1)$-share S and R respectively. On the other hand, in P-(t,2t)-Share, invoke two instances of PAVSS-MS-Share to t-share and $(2t-1)$-share S and R respectively.

CODE FOR P_i (TO OBTAIN THE SHARES OF S): Every party in $\mathcal{P}$ executes this code

1. In S-(t,2t)-Share, participate in the two instances of SAVSS-MS-Share and wait to terminate these two instances. On the other hand, in P-(t,2t)-Share, participate in the two instances of PAVSS-MS-Share and wait to terminate these two instances.
2. Let $(\varphi_{1,i}, \ldots, \varphi_{\ell,i})$ and $(\chi_{1,i}, \ldots, \chi_{\ell,i})$ be the i^{th} shares obtained from the two instances of SAVSS-MS-Share/PAVSS-MS-Share.
3. For $l = 1, \ldots, \ell$, locally compute $\psi_{l,i} = \varphi_{l,i} + i \cdot \chi_{l,i}$. Output $(\varphi_{1,i}, \ldots, \varphi_{\ell,i})$ and $(\psi_{1,i}, \ldots, \psi_{\ell,i})$ as the i^{th} share of $(t, 2t)$-sharing of S and terminate.

(b) If D is corrupted and some honest party terminates S-(t,2t)-Share, then all honest parties will eventually terminate the protocol, except with probability ϵ.

2. *Correctness: (a) If D is honest, then S will be $(t, 2t)$-shared among the parties in $\mathcal{P}$. (b) If D is corrupted and the honest parties terminate S-(t,2t)-Share, then there exist ℓ values, which are $(t, 2t)$-shared among the parties in $\mathcal{P}$, except with probability ϵ.*

3. *Communication Complexity: Protocol S-(t,2t)-Share requires a private communication of $\mathcal{O}((\ell n^2 + n^3) \log |\mathbb{F}|)$ bits and A-cast of $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits.*

PROOF: The proof follows from the properties of the protocol SAVSS-MS-Share (Theorem 6). □

The proof of the following theorem follows using the same arguments as used in the previous theorem, except that we now depend on the properties of PAVSS-MS-Share instead of SAVSS-MS-Share.

Theorem 9. *Protocol P-(t,2t)-Share achieves the following properties:*

1. *Termination: (a) If D is honest, then all honest parties will eventually terminate P-(t,2t)-Share. (b) If D is corrupted and some honest party terminates P-(t,2t)-Share, then all honest parties will eventually terminate P-(t,2t)-Share.*

2. *Correctness: (a) If D is honest, then S will be $(t, 2t)$-shared among the parties in $\mathcal{P}$. (b) If D is corrupted and the honest parties terminate P-(t,2t)-Share, then there exist ℓ values, which will be $(t, 2t)$-shared among the parties in $\mathcal{P}$.*

3. *Communication Complexity: Protocol P-(t,2t)-Share incurs 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.*

6 AMPC Protocol with $n = 4t + 1$

Once we have an efficient protocol for generating $(t, 2t)$-sharing, we can design AMPC protocol following the approach of [5]. Structurally, both our statistical and perfect AMPC protocol are divided into a sequence of three phases. Depending upon the type of sub-protocols (with negligible error or without any error) used in these phases, we get either a statistical AMPC or a perfect AMPC protocol. Let $\mathcal{F}$ be a publicly known function over $\mathbb{F}$, which is represented by an arithmetic circuit over $\mathbb{F}$, consisting of input gates, linear gates, multiplication gates, random gates and output gates of bounded fan-in. Without loss of generality, we assume that the multiplication gates have fan-in two and the random gates have fan-in one. It is well known that any arithmetic circuit can be represented like this. Let c_I, c_L, c_M, c_R and c_O denote the number of input, linear, multiplication, random and output gates respectively in the circuit representing $\mathcal{F}$. We denote by IGate, LGate, MGate, RGate and OGate the input, linear, multiplication, random and output gates respectively. For simplicity, we assume that $\mathcal{F} : \mathbb{F}^n \rightarrow \mathbb{F}^n$, where each party P_i has the input $x_i \in \mathbb{F}$ for the computation and all the n parties receive the function output $\mathcal{F}(x_1, \ldots, x_n)$. This implies that $c_I = n$. The three phases of our AMPC protocols are as follows:

1. *Preparation Phase*: The goal of this phase is to prepare the "raw material" to be used later during the evaluation of the circuit. Specifically, in this phase, the parties interact to generate $(t, 2t)$-sharing of $c_M + c_R$ uniformly random values from $\mathbb{F}$, that are information theoretically secure.

2. *Input Phase*: In this phase, the parties share their actual inputs for the function $\mathcal{F}$. For this, every party t-share his input and then the parties agree on a common set of at least $(n - t)$ parties, who t-shared their inputs. Every honest party will eventually get shares of the inputs of the parties in this common set.

3. *Computation Phase*: Here, based on the inputs of the parties in the common set (agreed in the previous phase), the circuit will be evaluated gate by gate in a shared fashion, such that the output of each gate remains t-shared among the parties.

We now present the protocols for each of the above phases.

6.1 Preparation Phase

Here the parties interact to generate $(t, 2t)$-sharing of $c_M + c_R$ uniformly random values from $\mathbb{F}$. The shared values should also remain information theoretically secure. For this, every party in $\mathcal{P}$ acts as a dealer and $(t, 2t)$-shares $\frac{c_M + c_R}{n-2t}$ uniformly random values from $\mathbb{F}$. The parties then agree on a common set of at least $(n - t)$ parties who indeed $(t, 2t)$-shared $\frac{c_M + c_R}{n-2t}$ values. Out of these $(n - t)$ parties, at least $(n - 2t)$ are honest, who have indeed shared random values. The random values shared by the honest parties are unknown to $\mathcal{A}_t$. But the identities of the honest parties are unknown. So, we apply the information theoretic randomness extraction function Ext (see Section 2.3) on the sharings done by the parties in the common set to obtain $(t, 2t)$-sharing of $c_M + c_R$ uniformly random values. In Fig. 16, we present the protocol for this phase. Now depending upon whether the parties invoke the protocol S-(t, 2t)-Share (having negligible error) or P-(t, 2t)-Share (having no error), we get the protocol S-Preparation or P-Preparation respectively for the preparation phase.

We now prove the properties of the protocol S-Preparation and P-Preparation, which follows from the properties of S-(t, 2t)-Share and P-(t, 2t)-Share respectively, along with the properties of Ext.

Lemma 20. *Protocol S-Preparation satisfies the following properties:*

1. *Termination: All honest parties will eventually terminate the protocol, except with probability ϵ.*

2. *Correctness: The protocol outputs $(t, 2t)$-sharing of $c_M + c_R$ uniformly random values, except with probability ϵ.*

3. *Secrecy: For $i = 1, \ldots, n-2t$ and $j = 1, \ldots, \frac{c_M + c_R}{n-2t}$, the values $r_{i,j}$ will be information theoretically secure.*

Figure 16: Protocol for the Preparation Phase.

Protocol S-Preparation / P-Preparation

SHARING RANDOM VALUES:CODE FOR P_i: Every party executes this code

1. Select $L = \frac{c_M + c_R}{n-2t}$ random elements $S_i = (s_{i,1}, \ldots, s_{i,L})$ from $\mathbb{F}$. In S-Preparation, invoke S-(t, 2t)-Share as a dealer, to $(t, 2t)$-share S_i. Let this instance of S-(t, 2t)-Share be denoted as S-(t, 2t)-Share$_i$. On the other hand, in P-Preparation, invoke P-(t, 2t)-Share as a dealer, to $(t, 2t)$-share S_i and denote this instance as P-(t, 2t)-Share$_i$
2. For $j = 1, \ldots, n$, participate in the instance S-(t, 2t)-Share$_j$ / P-(t, 2t)-Share$_j$, depending upon whether it is S-Preparation or P-Preparation.

AGREEMENT ON THE COMMON SET: CODE FOR P_i: Every party executes this code

1. Create an accumulative set $C_i = \emptyset$. Upon terminating the instance S-(t,2t)-Share$_j$ / P-(t,2t)-Share$_j$, add P_j in C_i.
2. Participate in an instance of ACS with accumulative set C_i as the input.

Let C be the common set of size $(n - t)$ obtained as the output of the ACS and without loss of generality, let $C = \{P_1, \ldots, P_{n-t}\}$. For every $k \in \{1, \ldots, L\}$, let $(r_{1,k}, \ldots, r_{n-2t,k}) = \mathsf{Ext}(s_{1,k}, \ldots, s_{n-t,k})$. The parties then obtain their shares corresponding to $(t, 2t)$-sharing of $(r_{1,k}, \ldots, r_{n-2t,k})$ as follows:

GENERATION OF THE RANDOM $(t, 2t)$-SHARINGS: CODE FOR P_i: Every party executes this code

1. For every P_j in the common set C, obtain the i^{th} shares $(\varphi_{j,1,i}, \ldots, \varphi_{j,L,i})$ and $(\psi_{j,1,i}, \ldots, \psi_{j,L,i})$, corresponding to $(t, 2t)$-sharing of S_j.
2. For every $k \in \{1, \ldots, L\}$, locally compute $([r_{1,k}]_t, \ldots, [r_{n-2t,k}]_t) = \mathsf{Ext}([s_{1,k}]_t, \ldots, [s_{n-t,k}]_t)$ and $([r_{1,k}]_{2t}, \ldots, [r_{n-2t,k}]_{2t}) = \mathsf{Ext}([s_{1,k}]_{2t}, \ldots, [s_{n-t,k}]_{2t})$. That is, locally compute the i^{th} shares $(\varsigma_{1,k,i}, \ldots, \varsigma_{n-2t,k,i}) = \mathsf{Ext}(\varphi_{1,k,i}, \ldots, \varphi_{n-t,k,i})$ and $(\sigma_{1,k,i}, \ldots, \sigma_{n-2t,k,i}) = \mathsf{Ext}(\psi_{1,k,i}, \ldots, \psi_{n-t,k,i})$ and terminate.

The values $r_{1,1}, \ldots, r_{n-2t,1}, \ldots, r_{1,L}, \ldots, r_{n-2t,L}$ denote the $c_M + c_R$ random values which are now $(t, 2t)$-shared.

4. *Communication Complexity: The protocol privately communicates $\mathcal{O}(((c_M + c_R)n^2 + n^4)\log|\mathbb{F}|)$ bits, incurs A-cast of $\mathcal{O}(n^4 \log|\mathbb{F}|)$ bits and requires one invocation of ACS.*

PROOF: For the termination property, we first notice that the invocation of ACS will indeed output a common set C of $3t + 1$ parties. This is because there are at least $3t + 1$ honest parties, who will invoke an instance of S-(t, 2t)-Share and these instances will be eventually terminated by every honest party. We next claim that every honest party will eventually terminate the S-(t, 2t)-Share instance of every party (dealer) in C, except with probability ϵ. If C contains only honest parties then the claim is trivially true. We consider the worst case, when C can contain t corrupted parties. The termination property of S-(t, 2t)-Share ensures that the S-(t, 2t)-Share instance of each of these t corrupted dealers will be terminated except with probability ϵ. So the error probability that the S-(t, 2t)-Share instance of at least one corrupted dealer in C is not terminated is at most $t \cdot \epsilon$. Assuming $t \cdot \epsilon \approx \epsilon$ ensures that the S-(t, 2t)-Share instances of all the parties in C will eventually terminate, except with probability ϵ. Alternatively, by appropriately setting the parameters (the size of the field), we can execute each instance of S-(t, 2t)-Share to have an error probability of at most $\frac{\epsilon}{t}$. This will bound the error probability in the termination property of S-Preparation by at most ϵ.

If the common set C contains only honest parties then the correctness property holds trivially without any error. This is because each honest party indeed does $(t, 2t)$-sharing of random values. We now consider the worst case, when C can contain t corrupted parties (dealers). Even in this case, there will be $(n - 2t)$ honest parties in C and they will $(t, 2t)$-share random values. The correctness property of S-(t, 2t)-Share ensures that even a corrupted party in C does $(t, 2t)$-sharing of L values (probably non-random), except with probability ϵ in his instance of S-(t, 2t)-Share. This implies that except with probability at most $t \cdot \epsilon$, every corrupted party in C has done $(t, 2t)$-sharing of L values. Again, assuming that either $t \cdot \epsilon \approx \epsilon$ or by invoking each instance of S-(t, 2t)-Share to have an error probability of at most $\frac{\epsilon}{t}$, we can ensure that except with probability at most ϵ, every party in C has done $(t, 2t)$-sharing of L values. Moreover, at least $(n-2t)\cdot L = c_M + c_R$ of these $|C|\cdot L$ values will be random. Now the property of Ext ensures that the protocol outputs $(t, 2t)$-sharing of random values.

The secrecy property of S-(t, 2t)-Share ensures that the L values which are $(t, 2t)$-shared by the honest parties in C are information theoretically secure. This implies that out of the total $|C|\cdot L$ values which are shared by the parties in C, at least $(|C|-t)\cdot L = c_M + c_R$ values are information theoretically secure. The property of Ext ensures that for $i = 1, \ldots, n-2t$ and $j = 1, \ldots, \frac{c_M+c_R}{n-2t}$, the values $r_{i,j}$ are information theoretically secure. In the protocol, other than the execution of the instances of S-(t, 2t)-Share, there is no interaction among the parties. The function Ext is applied locally on the shares of the parties in C. This implies that $r_{i,j}$'s remain information theoretically secure.

In the protocol, each party executes an instance of S-(t,2t)-Share to $(t, 2t)$ share $L = \frac{c_M+c_R}{n-2t}$ values. Substituting $\ell = L$ in Theorem 8, the total private communication of the protocol is $\mathcal{O}(\left(Ln^3 + n^4\right)\log|\mathbb{F}|)$ bits. Since $L = \frac{c_M+c_R}{n-2t}$ and $n-2t = \Theta(n)$, the total private communication is $\mathcal{O}(((c_M+c_R)n^2+n^4)\log|\mathbb{F}|)$ bits. Moreover, the protocol A-casts $\mathcal{O}(n^4\log|\mathbb{F}|)$ bits and requires one invocation of ACS. □

The proof of the following lemma follows using the same arguments as used in the previous lemma, except that we now depend on the properties of P-(t,2t)-Share instead of S-(t,2t)-Share.

Lemma 21. *Protocol P-Preparation satisfies the following properties:*

1. *Termination: All honest parties will eventually terminate the protocol.*
2. *Correctness: The protocol outputs $(t, 2t)$-sharing of $c_M + c_R$ uniformly random values.*
3. *Secrecy: For $i = 1, \ldots, n-2t$ and $j = 1, \ldots, \frac{c_M+c_R}{n-2t}$, the values $r_{i,j}$ will be information theoretically secure.*
4. *Communication Complexity: the protocol privately communicates $\mathcal{O}((c_M+c_R)n^2\log|\mathbb{F}|)$ bits, incurs A-cast of $\mathcal{O}(n^3\log|\mathbb{F}|)$ bits and requires one invocation of the ACS.*

6.2 Input Phase

In this phase, each party t-share his input x_i (for the computation), by executing an instance of our AVSS schemes. If the parties invoke the statistical protocol SAVSS-MS-Share, then the resultant protocol for the input phase is called S-Input. On the other hand, if the parties use the perfect protocol PAVSS-MS-Share to share their inputs, then the resultant protocol is called P-Input. The asynchrony of the network does not allow the parties to wait for the termination of the SAVSS-MS-Share / PAVSS-MS-Share instances of more than $(n-t)$ parties. In order to agree on a common set C (this should not be confused with the common set C of the previous phase) of parties whose instances of SAVSS-MS-Share / PAVSS-MS-Share have terminated, one instance of the ACS is invoked. The parties then consider t-sharing of the inputs shared by the parties in the common set C and substitute a default t-sharing of 0 corresponding to the inputs of the parties not in C. The protocol for this phase is given in Fig. 17.
We now prove the properties of the protocol S-Input and P-Input, which follows from the properties of the protocol SAVSS-MS-Share and PAVSS-MS-Share respectively.

Lemma 22. *Protocol S-Input satisfies the following properties:*

1. *Termination: All honest parties will eventually terminate the protocol, except with probability ϵ.*
2. *Correctness: The protocol correctly outputs t-sharing of the inputs of the parties in the agreed common set C, except with probability ϵ.*

Figure 17: Protocol for the Input phase.

Protocol S-Input / P-Input

SHARING THE INPUTS: CODE FOR P_i: Every party executes this code

1. On having the input $x_i \in \mathbb{F}$, invoke SAVSS-MS-Share / PAVSS-MS-Share as a dealer, to t-share x_i. Let this instance be denoted as SAVSS-MS-Share$_i$ / PAVSS-MS-Share$_i$.
2. For $j = 1, \ldots, n$, participate in the instance SAVSS-MS-Share$_j$ / PAVSS-MS-Share$_j$.

AGREEMENT ON THE COMMON SET: CODE FOR P_i: Every party executes this code

1. Create an accumulative set $C_i = \emptyset$. Upon terminating the instance SAVSS-MS-Share$_j$ / PAVSS-MS-Share$_j$, add P_j in C_i.
2. Participate in an instance of ACS with the accumulative set C_i as input.

OUTPUT GENERATION: CODE FOR P_i: Every party executes this code

1. Wait until the ACS instance terminates with output C containing $n - t$ parties. Output the shares corresponding to t-sharing of the inputs of the parties in C. Substitute a default t-sharing of 0 for the inputs of the parties not in C and terminate.

3. *Secrecy: The inputs of the honest parties in the set C will remain information theoretically secure.*

4. *Communication Complexity: The protocol privately communicates $\mathcal{O}((c_I n^2 + n^4)\log|\mathbb{F}|)$ bits, incurs A-cast of $\mathcal{O}(n^4 \log|\mathbb{F}|)$ bits and requires one invocation of ACS.*

PROOF: Every honest party will t-share his input and his instance of SAVSS-MS-Share will be eventually terminated by every honest party. Moreover, there are at least $(n - t)$ honest parties. This implies that the instance of ACS will eventually terminate with output C. To show the termination property, we require to show that the SAVSS-MS-Share instance of the corrupted parties in C will be eventually terminated by every honest party. However, this follows from the termination property of SAVSS-MS-Share.

Every honest party in C will correctly t-share his input in his instance of SAVSS-MS-Share. The correctness property of SAVSS-MS-Share also ensures that even a corrupted $P_i \in C$ will t-share a value x_i (which may or may not be his actual input; but the value shared by a party is considered as his intended input). So the inputs of each party in C will be correctly t-shared.

The secrecy property of SAVSS-MS-Share ensures that the input x_i of every honest P_i in C remains information theoretically secure in the instance SAVSS-MS-Share$_i$. Apart from the execution of the instances of SAVSS-MS-Share, the protocol does not involve any communication among the parties. This implies that the inputs of the honest parties in the set C will remain information theoretically secure.

In the protocol, each party executes an instance of SAVSS-MS-Share to t-share his input $x_i \in \mathbb{F}$. From Theorem 6, we find that this requires total private communication of $\mathcal{O}((c_I n^2 + n^4)\log|\mathbb{F}|)$ bits, incurs A-cast of $\mathcal{O}(n^4 \log|\mathbb{F}|)$ bits and requires one invocation of ACS. □

The proof of the following lemma follows using similar arguments as used in the previous lemma, except that we now depend upon the properties of PAVSS-MS-Share, instead of SAVSS-MS-Share.

Lemma 23. *Protocol P-Input satisfies the following properties:*

1. *Termination: All honest parties will eventually terminate the protocol.*

2. *Correctness: The protocol correctly outputs t-sharing of the inputs of the parties in the agreed common set C.*

3. *Secrecy: The inputs of the honest parties in the set C will remain information theoretically secure.*

4. *Communication Complexity: the protocol privately communicates $\mathcal{O}(c_I n^2 \log |\mathbb{F}|)$ bits, incurs A-cast of $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits and requires one invocation of ACS.*

6.3 Computation Phase

The protocol for this phase is the same for both the statistical as well as the perfect AMPC scheme. Here the circuit is evaluated gate by gate, where all intermediate values during the computation remain t-shared. As soon as a party holds his shares of the input values of a gate, he joins the evaluation of the gate. Due to the linearity of the used t-sharing, linear gates can be evaluated locally by simply applying the linear function to the shares of the inputs of the gate. With every random gate, one random $(t, 2t)$-sharing (from the preparation phase) is associated. The t-sharing of the associated $(t, 2t)$-sharing is directly used as the outcome of the random gate. With every multiplication gate, one random $(t, 2t)$-sharing is associated, which is then used to compute t-sharing of the product, following the idea outlined earlier (in the introduction). This approach of evaluating a multiplication gate is also used in the AMPC protocol of the [5]. The protocol for this phase is called Computation, which is presented in Fig. 18.

Figure 18: Protocol for the computation phase to evaluate the circuit of $\mathcal{F}$.

Protocol Computation

Let $[r_1]_{t,2t}, \ldots, [r_{c_M+c_R}]_{t,2t}$ be the $c_M + c_R$ $(t, 2t)$-sharings which have been generated during the preparation phase.

EVALUATION OF THE CIRCUIT: CODE FOR P_i — Every party executes this code

For every gate in the circuit, wait until the i^{th} share of each input of the gate is available. Now depending on the type of gate, proceed as follows:

1. **Input Gate:** $[s]_t = \mathsf{IGate}([s]_t)$: Simply output Sh_i, the i^{th} share of s.
2. **Linear Gate:** $[e]_t = \mathsf{LGate}([c]_t, [d]_t, \ldots)$: Compute and output $e_i = \mathsf{LGate}(c_i, d_i, \ldots)$, the i^{th} share of $e = \mathsf{LGate}(c, d, \ldots)$, where $c_i, d_i, \ldots$ denotes the i^{th} share of $c, d, \ldots$.
3. **Multiplication Gate:** $[e]_t = \mathsf{MGate}([c]_t, [d]_t)$: If this is the k^{th} multiplication gate in the circuit, then the $(t, 2t)$-sharing $[r_k]_{t,2t}$ is associated with this gate. Let $(\varphi_{k,1}, \ldots, \varphi_{k,n})$ and $(\psi_{k,1}, \ldots, \psi_{k,n})$ denote the corresponding t-sharing and $2t$-sharing of r_k, respectively.
   (a) Locally compute $[\delta]_{2t} = [c]_t \cdot [d]_t - [r]_{2t}$. For this, compute $\delta_i = c_i \cdot d_i - \psi_{k,i}$, where c_i and d_i are the i^{th} shares of c and d respectively and δ_i is the i^{th} share, corresponding to $2t$-sharing of δ.
   (b) To reconstruct δ, privately send the share δ_i to every party in $\mathcal{P}$. Apply OEC on the received δ_j's to privately reconstruct δ.
   (c) Locally compute $[e]_t = [r]_t + [\delta]_t$, where $[\delta]_t = (\delta, ..., \delta(n \text{ times}))$. For this, compute and output $e_i = \delta + \varphi_{k,i}$, the i^{th} share of e.
4. **Random Gate:** $[R]_t = \mathsf{RGate}(\cdot)$: If this is the k^{th} random gate in the circuit, then the $(t, 2t)$-sharing $[r_{c_M+k}]_{t,2t}$ is associated with this gate. Let $(\varphi_{c_M+k,1}, \ldots, \varphi_{c_M+k,n})$ and $(\psi_{c_M+k,1}, \ldots, \psi_{c_M+k,n})$ denote the corresponding t-sharing and $2t$-sharing of r_{c_M+k}, respectively. Output $R_i = \varphi_{c_M+k,i}$ as the i^{th} share of R.
5. **Output Gate:** $x = \mathsf{OGate}([x]_t)$: Privately send x_i, the i^{th} share of x to every party in $\mathcal{P}$. Apply OEC on the received x_j's and output x.

Once all the output gates in the circuit are evaluated, terminate the protocol.

We now prove the properties of the protocol Computation.

Lemma 24. *Given that $c_M + c_R$ information theoretically secure random values are $(t, 2t)$-shared among the parties and the inputs of all the parties are t-shared, protocol Computation satisfies the following properties:*

1. *Termination: All honest parties will eventually terminate the protocol.*

2. *Correctness: The protocol correctly computes the output of the function $\mathcal{F}$.*

3. *Secrecy: The adversary obtains no additional information about the intermediate values in the computation (in the information theoretic sense), other than what is inferred from the input and the output of the corrupted parties.*

4. *Communication Complexity: The protocol privately communicates $\mathcal{O}(n^2(c_M + c_O)\log|\mathbb{F}|)$ bits.*

PROOF: The circuit representing the function $\mathcal{F}$ is finite. To prove the termination property, we claim that each honest party will eventually evaluate each gate of the circuit. The claim is trivially true for the input gates and the random gates. The linearity property of t-sharing ensures that the claim is also true for the linear gates. Now consider a multiplication gate: the property of OEC (Theorem 3) implies that every honest party will eventually reconstruct δ during the evaluation of the multiplication gate. After this, the evaluation of the multiplication gate involve local computation and so it will be done eventually by every honest party. Similarly, the property of OEC ensures that each honest party will eventually obtain the value of each output gate.

The linearity of t-sharing ensures that each linear gate is evaluated correctly by the honest parties. Now consider a multiplication gate with inputs c, d and let r be the random value, whose $(t, 2t)$-sharing is associated with the multiplication gate. It is easy to see that $e = c\dot{d} = (c\dot{d} - r) + r = \delta + r$, where $\delta = (c\dot{d} - r)$, which also implies that $[e]_t = \delta + [r]_t$, if δ is publicly known. The property of OEC ensures that every honest party will correctly reconstruct δ, which implies that the multiplication gates will also be evaluated correctly by the honest parties. The random gates will be evaluated correctly due to the assumption that the associated $(t, 2t)$-sharing is correct. Now if all the gates in the circuit are evaluated correctly, it implies that each honest party will have the correct share corresponding to t-sharing of the function output (namely the output gates). So by the property of OEC, each honest party will correctly reconstruct the value of each output gate and hence the function output.

To prove the secrecy, we claim the following for every intermediate gate (i.e. other than the output gates) in the circuit: the evaluation of the gate reveals no additional information to the adversary (in the information theoretic sense) about the sharings associated with the input(s) of the gate (the sharings of the output gates are reconstructed by all the parties and hence they will be known to everyone). Our claim is trivially true for the random gates, as to evaluate a random gate, no communication is done among the parties; the parties simply associate t-sharing of a random value (which is already proven to be information theoretically secure) with the gate. The claim is also true for any linear gate; the evaluation of the linear gates require only local computation and no interaction among the parties. Now consider a multiplication gate, with inputs c and d and let r be the random, information theoretically secure value, associated with the multiplication gate, which is $(t, 2t)$-shared. Since r is $(t, 2t)$-shared, it implies that r is t-shared and $2t$-shared through independent polynomials of degree atmost t and $2t$ respectively, with the adversary knowing at most t points on each polynomial. During the evaluation of the multiplication gate, the $2t$-sharing of $\delta = (c \cdot d - r)$ is revealed to the adversary (since it is reconstructed by every party). However, since r is random and information theoretically secure, the reconstruction of δ does not add any extra information to the view of the adversary. Specifically, from the viewpoint of the adversary, the reconstructed polynomial and its constant term (which is δ) is completely random. Once δ is known, the evaluation of the multiplication gate involves only local computation and so the adversary gains no extra information. This shows that during the evaluation of the circuit, the adversary obtains no additional information about the intermediate values, other than what is inferred from the input and the output of the corrupted parties [6].

[6] As mentioned earlier, we can prove the secrecy in the framework of real world/ideal world paradigm of [8]. However, we avoid doing so, as it requires additional technicalities which will make the paper complicated.

The communication complexity follows from the fact that $c_M + c_O$ values are reconstructed in the protocol (one value per multiplication gate and one value per output gate) and to reconstruct a value, every party sends his share to every other party, incurring a private communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. $\square$

In the next section, we finally present our statistical AMPC protocol.

6.4 Statistical AMPC Protocol with $n = 4t + 1$

The statistical AMPC protocol SAMPC consists of the following three steps:

1. Invoke S-Preparation.
2. Invoke S-Input.
3. Invoke Computation.

We next state the properties of the protocol SAMPC.

Theorem 10. *Protocol SAMPC is a statistical AMPC protocol, satisfying the Def. 5. The protocol privately communicates $\mathcal{O}(((c_I + c_M + c_R + c_O)n^2 + n^4) \log |\mathbb{F}|)$ bits, incurs A-cast of $\mathcal{O}(n^4 \log |\mathbb{F}|)$ bits and requires two invocations of ACS.*

PROOF: The proof follows from the properties of the protocol S-Preparation (Lemma 20), protocol S-Input (Lemma 22) and the protocol Computation (Lemma 24). $\square$

6.5 Perfect AMPC Protocol with $n = 4t + 1$

The perfect AMPC protocol PAMPC consists of the following three steps:

1. Invoke P-Preparation.
2. Invoke P-Input.
3. Invoke Computation.

We next state the properties of the protocol PAMPC.

Theorem 11. *Protocol PAMPC is a perfect AMPC protocol, satisfying the Def. 5. The protocol privately communicates $\mathcal{O}((c_I + c_M + c_R + c_O)n^2 \log |\mathbb{F}|)$ bits, incurs A-cast of $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits and requires two invocations of ACS.*

PROOF: The proof follows from the properties of the protocol P-Preparation (Lemma 21), protocol P-Input (Lemma 23) and the protocol Computation (Lemma 24). $\square$

7 Packed Secret Sharing: Another Perspective of Our AVSS Schemes

We now briefly discuss another important perspective of our AVSS schemes. For simplicity and concreteness, we refer to our perfect AVSS scheme in the discussion below (although the discussion holds for the statistical AVSS scheme as well). Consider the protocol PAVSS-Share that can d-share a single secret: if D is honest in the protocol, then the following holds at the end of protocol; there exists a polynomial $f_0(x) = F(x, 0)$ of degree at most d and every party P_i holds $Sh_i = f_0(i)$. Furthermore, the adversary $\mathcal{A}_t$ knows at most t distinct points on $f_0(x)$ and he lacks $(d - t) + 1$ additional distinct points on $f_0(x)$ to uniquely interpolate the polynomial $f_0(x)$. This fact suggests that from the view point of the adversary,

$(d-t)+1$ coefficients of the polynomial $f_0(x)$ are "free" and hence random. So D can share $(d-t)+1$ secrets using the single polynomial $f_0(x)$. This concept, known as the packed secret sharing was introduced in [27], but for the synchronous settings [7]. In what follows we show how the protocol PAVSS-Share can be used as a packed secret sharing scheme where D can share $(d-t)+1$ secrets simultaneously in the information theoretic sense. Moreover, even if D is corrupted, there exist $(d-t)+1$ values, to which D is committed to at the end of the protocol.

Let $s_1, \ldots, s_k$ be the k values, which D wants to share among the parties, such that $k = (d-t)+1$. D selects a polynomial $f(x)$ over $\mathbb{F}$ of degree at most d. The polynomial $f(x)$ is an otherwise random polynomial such that $f(0) = s_1, f(-1) = s_2, \ldots, f(-k+1) = s_k$. D then selects a bi-variate polynomial $F(x, y)$ over $\mathbb{F}$ of degree-(d, t), which is an otherwise random polynomial such that $F(x, 0) = f(x)$. This implies that $f_0(x) \stackrel{def}{=} F(x, 0) = f(x)$. D then invokes the protocol PAVSS-Share using the bi-variate polynomial $F(x, y)$ selected as above and the parties participate in this instance of PAVSS-Share. If D is honest, then by the termination property of PAVSS-Share, every honest party P_i will eventually terminate the protocol, with his share $Sh_i = f(i)$. Notice that Sh_i is the share for the multi-secret $s_1, \ldots, s_k$. Moreover, $s_1, \ldots, s_k$ are information theoretically secure, since $f(x)$ has degree at most d and the adversary $\mathcal{A}_t$ gets at most t points on $f(x)$. Interestingly, even if D is corrupted, there are k values, say $\overline{s}_1, \ldots, \overline{s}_k$, to which D is committed to at the end of PAVSS-Share. To recover $s_1, \ldots, s_k$, the parties execute the protocol Rec. From the property of Rec, every honest party will eventually reconstruct $f(x)$ correctly and will obtain the secrets $s_1, \ldots, s_k$.

The above idea is also applicable for the protocol PAVSS-MS-Share, where D can use ℓ independent bi-variate polynomials, each of degree-(d, t) and using each polynomial, he can share $(d-t)+1$ values. This implies that he can share total $\ell \cdot ((d-t)+1)$ values. The total cost for sharing these values will be $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits of private communication and A-cast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits (Theorem 7). Setting $d = 2t$ (the maximum allowed value of d), we see that PAVSS-MS-Share can share $\ell(t+1) = \Theta(\ell n)$ values by privately communicating $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits and incurring A-cast communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ 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 PAVSS-MS-Share is $\mathcal{O}(n \log |\mathbb{F}|)$ bits. The best known perfect AVSS of [5] requires an amortized cost of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits for sharing a single secret. Hence PAVSS-MS-Share shows a clear improvement over the AVSS of [5] when both are interpreted as a packed secret sharing scheme. We further note that the amortized cost of sharing a single secret from $\mathbb{F}$ in PAVSS-MS-Share tolerating active adversary matches the cost of sharing a single element in the presence of a passive adversary (for example, the Shamir secret sharing scheme [43]).

Notice that the above discussion holds for the statistical protocol SAVSS-MS-Share as well. However, the protocol SAVSS-MS-Share may involve a negligible error. On the other hand, protocol PAVSS-MS-Share is perfect in all respect and does not involve any error.

8 Flaw in the Statistical AMPC of Huang et al.

We now recall the statistical AMPC protocol of [31] and show that it does not satisfy the correctness and the termination condition. The AMPC protocol of [31] is divided into a sequence of three phases: the Preparation Phase, the Input Phase and the Computation & Output Phase. We concentrate on the Preparation Phase and show that it fails to satisfy the correctness and the termination property which are claimed in [31] to hold. This will further imply that the AMPC of [31] does not satisfy the correctness and the termination property.

Recall that c_M is the number of multiplication gates in the circuit expressing the function $\mathcal{F}$. The goal

[7] In [27], the concept was introduced in a slightly different way but the essence was the same.

of the Preparation Phase is to generate c_M random multiplication triples $(a_1, b_1, c_1), \ldots, (a_{c_M}, b_{c_M}, c_{c_M})$, where for $k = 1, \ldots, c_M$, each a_k, b_k and c_k are t-shared among the parties in $\mathcal{P}$ with a_k and b_k being random and c_k satisfying $c_k = a_k \cdot b_k$. For this, the authors used the batch secret sharing scheme (BSS) from [46]. In [46], the authors claimed that their BSS protocol correctly generates c_M random multiplication triples over $\mathbb{F}$. Moreover, every honest party will eventually terminate BSS. However, we now show that their BSS scheme does not satisfy the correctness property as well as the termination property. As a result, the AMPC protocol of [31] (which uses the BSS scheme as a black box) does not satisfy the correctness and the termination condition.

The BSS scheme of [46] is based on the player elimination framework [30], where the computation is divided into a sequence of segments. To show the weakness in the BSS scheme of [46], we do not need to get into the details of the player elimination framework. We concentrate only on the crucial steps (presented in a simplified form for the ease of presentation) which are executed in a segment to generate t-sharing of one multiplication triple (a, b, c). The main steps in the generation of such a triple are as follows:

1. The parties in $\mathcal{P}$ jointly generate t-sharing of a random a and b.
2. The parties in $\mathcal{P}$ then jointly generate t-sharing of $c = ab$.

Now a t-sharing of a and b in the BSS scheme of [46] is generated by executing the steps presented in Fig. 19.

Figure 19: Steps for generating t-sharing of a random a and b in the BSS scheme of Huang et al.

1. GENERATION OF t-SHARING OF a AND b: Code for the Party $P_i \in \mathcal{P}$: Every party executes this code
   (a) Select two random polynomials $f_i(x)$ and $g_i(x)$ of degree at most t and send $f_i(j), g_i(j)$ to every $P_j \in \mathcal{P}$. After sending, A-cast 1 to indicate that you have finished the sharing.
   (b) Participate in the ACS protocol and input 1 in ABA_j (in ACS) if you have received 1 from the A-cast of P_j and if you have privately received $f_j(i), g_j(i)$ from P_j.
   (c) Let C be the common set which is output by the ACS protocol, where $|C| \geq (n - t)$.
   (d) Compute $a_i = \sum_{P_j \in C} f_j(i)$ and $b_i = \sum_{P_j \in C} g_j(i)$, as the i^{th} share of a and b respectively.
2. VERIFYING WHETHER a AND b ARE t-SHARED: Here the parties perform some computation to verify whether a and b are indeed shared through polynomials of degree at most t. If it is not the case then the segment fails and parties execute another protocol for the fault localization (for details see [46]). However, the verification is carried out under the assumption that every (honest) party $P_i \in \mathcal{P}$ will eventually possess the share a_i and b_i of a and b respectively.

From Fig. 19, we find that step 2 that verifies whether a and b are indeed t-shared among the parties in $\mathcal{P}$, will work if every (honest) $P_i \in \mathcal{P}$ holds a_i and b_i eventually. Clearly, this is possible if every (honest) party $P_i \in \mathcal{P}$ eventually receives $f_j(i)$ and $g_j(i)$ from every $P_j \in C$. In [46], the authors claimed that by executing the step 1 in Fig. 19, every (honest) $P_i \in \mathcal{P}$ will eventually receive $f_j(i)$ and $g_j(i)$ from *every* $P_j \in C$ and hence will be able to compute a_i and b_i. However, we now show that $\mathcal{A}_t$ may behave (specially schedules the messages) in such a way that every honest P_i have to wait indefinitely to compute a_i and b_i.

Without loss of generality, let the first $(n - t)$ parties in $\mathcal{P}$ (i.e. $P_1, \ldots, P_{n-t}$) be honest and the last t parties in $\mathcal{P}$ be corrupted. Now consider the following behavior of a corrupted $P_j \in \mathcal{P}$: P_j selects $f_j(x)$ and $g_j(x)$ of degree higher than t and gives points on $f_j(x), g_j(x)$ to only the first $(n - 2t)$ honest parties and to the t corrupted parties (but not to the remaining t honest parties in $\mathcal{P}$). But still P_j A-casts 1 to indicate that he has sent the points to every party in $\mathcal{P}$. Moreover, $\mathcal{A}_t$ schedules the messages of P_j such that they reach to their respective receivers immediately, without any delay. Now the first $(n - 2t)$ honest parties and the t

corrupted parties will input 1 in ABA_j in ACS, as they will receive points on $f_j(x)$ and $g_j(x)$ from P_j and will also receive 1 from the A-cast of P_j. So in ABA_j, there are $(n-t)$ inputs, with value 1. Now assuming that all the parties including the corrupted parties behave properly in ABA_j, the property of ABA ensures that every party in $\mathcal{P}$ will terminate ABA_j with output 1 and hence P_j will be present in the common set C. However, notice that the last t honest parties (to whom P_j has not sent the points on $f_j(x)$ and $g_j(x)$) did not feed any input in ABA_j. In fact, these honest parties will never receive their respective points on $f_j(x)$ and $g_j(x)$, despite terminating ABA_j with output 1. So even though a (corrupted) P_j is present in C, potentially t honest parties may never receive their respective points on $f_j(x)$ and $g_j(x)$.

Now using a similar strategy, another corrupted $P_k \in C$ (where $P_k \neq P_j$) may bar another set of t honest parties in $\mathcal{P}$, say the first t honest parties, from receiving their respective points on $f_k(x)$ and $g_k(x)$. In the worst case, there can be t corrupted parties in C, who may follow a similar strategy as explained above and can ensure that every honest party in $\mathcal{P}$ may wait indefinitely to receive their respective points on the polynomials, corresponding to some corrupted party (ies) in C. Thus every honest P_i in $\mathcal{P}$ may wait indefinitely to compute a_i and b_i.

The Technical Problem and a Possible Solution: From the description of ACS (see section 2.3), it follows that the primitive ACS can be used to agree on a set of parties who will eventually satisfy the property Q, where Q should have the following characteristic: "if some honest P_i finds some party P_j to satisfy Q, then every other honest party will also eventually find P_j to satisfy Q". However, in the steps given in Fig. 19, the parties use an instance of ACS to agree on a set of parties satisfying some property Q which does not satisfy the above characteristic. Specifically, in this case the property Q with respect to a party P is as follows: P has delivered a point on each of his two polynomials to every party and A-casted 1. Now as explained above, a corrupted P_j may not give points to all the honest parties and can still A-cast 1. So even if some honest party may receive points on the polynomials from P_j and concludes that P_j satisfies Q, it does not mean that every other honest party will also conclude the same, as they may never receive the values from P_j. It is this subtle property of Q in ACS, which causes the BSS scheme of [46] and hence the AMPC of [31] to fail to satisfy the termination (and the correctness) property.

A simple way to fix the above problem is to ask each $P_j \in \mathcal{P}$ to share two random values, say a_j and b_j using the Sh protocol of some AVSS and then use the ACS primitive to agree on a common set of $(n-t)$ parties C whose instances of the Sh protocol will be eventually terminated by all the (honest) parties in $\mathcal{P}$. Then each party P_i can locally compute $a_i = \sum_{P_j \in C} a_{j,i}$ and $b_i = \sum_{P_j \in C} b_{j,i}$, where $a_{j,i}$ and $b_{j,i}$ are the i^{th} share of a_j and b_j respectively. Now by the termination property of AVSS, every (honest) $P_i \in \mathcal{P}$ will eventually terminate the Sh and thus will receive $a_{j,i}, b_{j,i}$ corresponding to every $P_j \in C$ and can compute a_i and b_i finally. However, the current best AVSS protocol with $n = 4t+1$ (prior to our work) is due to [5], which requires a communication cost of $\mathcal{O}(\ell n^2 \log(|\mathbb{F}|))$ bits for concurrent sharing of ℓ values. If this AVSS is used then the resultant AMPC protocol will have a communication complexity of $\Omega(n^3 \log(|\mathbb{F}|)$ bits per multiplication gate, which is clearly more than the communication complexity of our AMPC protocols.

9 Open Problems

This article presents information theoretically secure AMPC protocols that achieve the communication complexity of $\mathcal{O}(n^2 \log(|\mathbb{F}|))$ bits per multiplication gate. Looking at the literature, we still see a gap between the communication complexity of the synchronous and the asynchronous MPC protocols. The best known optimally resilient perfect MPC protocol in the synchronous settings (i.e. with $n = 3t+1$) [6] communicates $\mathcal{O}(n \log(|\mathbb{F}|))$ bits per multiplication gate. So there is a $\Theta(n)$ gap in the communication complexity between the synchronous and asynchronous protocol. The situation for the statistical protocols is worse than the perfect case. The best known optimally resilient statistical MPC protocol in the synchronous settings (i.e. with $n = 2t + 1$) [11] communicates $\mathcal{O}(n \log(|\mathbb{F}|))$ bits per multiplication gate. On the other hand, the best known optimally resilient statistical AMPC protocol (i.e. with $n = 3t + 1$) [38] communicates $\mathcal{O}(n^5 \log(|\mathbb{F}|))$ bits per multiplication gate[8]. So there is a gap of $\Theta(n^3)$. An interesting research direction is to further reduce the gap between the communication complexity of the optimally resilient synchronous and asynchronous MPC protocols.

Acknowledgement: We would like to thank the anonymous referees of the Journal of Cryptology for their valuable comments. The comments indeed helped to improve the overall presentation. We would also like to sincerely thank Tal Rabin for suggesting the method for generating $(t, 2t)$-sharing and for giving her valuable comments on earlier drafts of the paper. We would also like to thank Nigel Smart for giving his insightful remarks on the revised version of the article.

References

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

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

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

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

[5] Z. Beerliová-Trubíniová and M. Hirt. Simple and efficient perfectly-secure asynchronous MPC. In K. Kurosawa, editor, *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security, Kuching, Malaysia, December 2-6, 2007, Proceedings*, volume 4833 of *Lecture Notes in Computer Science*, pages 376–392. Springer Verlag, 2007.

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

[7] Z. Beerliova-Trubiniova, M. Hirt, and J. B. Nielsen. Almost-asynchronous MPC with faulty minority. Cryptology ePrint Archive, Report 2008/416, 2008.

[8] In [19], the authors presented a new AVSS protocol with reduced communication complexity and mentioned that the communication complexity of the AMPC protocol of [38] can be reduced to $\mathcal{O}(n^4 \log(|\mathbb{F}|))$ bits per multiplication gate by incorporating the AVSS scheme of [19].

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

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

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

[11] E. Ben-Sasson, S. Fehr, and R. Ostrovsky. Near-linear unconditionally-secure multiparty computation with a dishonest minority. Cryptology ePrint Archive, Report 2011/629, 2011.

[12] C. H. Bennett, G. Brassard, C. Crépeau, and U. M. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41(6):1915–1923, 1995.

[13] C. H. Bennett, G. Brassard, and J. Robert. Privacy amplification by public discussion. *SIAM J. Comput.*, 17(2):210–229, 1988.

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

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

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

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

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

[19] A. Choudhury and A. Patra. Statistical asynchronous weak commitment scheme: A new primitive to design statistical asynchronous verifiable secret sharing scheme. In *WCC'11. Proceedings of the Seventh International Workshop on Coding and Cryptography, April 11 - 15, Paris, France*, 2011.

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

[21] I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen. Asynchronous multiparty computation: Theory and implementation. volume 5443 of *Lecture Notes in Computer Science*, pages 160–179. Springer Verlag, 2009.

[22] I. Damgård and Y. Ishai. Scalable Secure Multiparty Computation. In C. Dwork, editor, *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, volume 4117 of *Lecture Notes in Computer Science*, pages 501–520. Springer, 2006.

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

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

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

[26] M. Fitzi, J. Garay, S. Gollakota, C. Pandu Rangan, and K. Srinathan. Round-optimal and efficient verifiable secret sharing. In S. Halevi and T. Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 329–342. Springer Verlag, 2006.

[27] M. K. Franklin and M. Yung. Communication Complexity of Secure Computation (Extended Abstract). In S. R. Kosaraju, M. Fellows, A. Wigderson, and J. A. Ellis, editors, *Proceedings of the 24th Annual ACM Symposium on Theory of Computing, May 4-6, 1992, Victoria, British Columbia, Canada*, pages 699–710. acm, 1992.

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

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

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

[31] Z. Huang, W. Qiu, Q. Li, and K. Chen. Efficient Secure Multiparty Computation Protocol in Asynchronous Network. In J. H. Park, H. Chen, M. Atiquzzaman, C. Lee, T. Kim, and S. Yeo, editors, *Proceedings of Advances in Information Security and Assurance, Third International Conference and Workshops, ISA 2009, Seoul, Korea*, volume 5576 of *Lecture Notes in Computer Science*, pages 152–158. Springer Verlag, June 2009.

[32] J. Katz, C. Koo, and R. Kumaresan. Improving the round complexity of VSS in point-to-point networks. In L. Aceto, I. Damgård, L. A. Goldberg, M. M. Halldórsson, A. Ingólfsdóttir, and

I. Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 499–510. Springer Verlag, 2008.

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

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

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

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

[37] A. Patra, A. Choudhary, and C. Pandu Rangan. Efficient asynchronous Byzantine Agreement with optimal resilience. Submitted to Distributed Computing Journal. A preliminary version of this article appeared in Proceedings of the 28th Annual ACM Symposium on Principles of Distributed Computing, PODC 2009, Calgary, Alberta, Canada, August 10-12, pages 92-101, 2009.

[38] A. Patra, A. Choudhary, and C. Pandu Rangan. Efficient statistical asynchronous verifiable secret sharing and multiparty computation with optimal resilience. Cryptology ePrint Archive, Report 2009/492, 2009.

[39] A. Patra, A. Choudhary, and C. Pandu Rangan. Communication Efficient Perfectly Secure VSS and MPC in Asynchronous Networks with Optimal Resilience. In D.J. Bernstein and T. Lange, editors, *Advances in Cryptology - AFRICACRYPT'10, Third International Conference in Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2009, Proceedings*, volume 6055 of *Lecture Notes in Computer Science*, pages 184–202. Springer Verlag, 2010.

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

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

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

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

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

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

[46] H. Zheng, G. Zheng, and L. Qiang. Batch secret sharing for secure multi-party computation in asynchronous network. *Journal of Shanghai Jiaotong Univ. (Sci.)*, 14(1):112–116, 2009.

A Proof of the Technical Lemmas

Lemma 2: *Let $\overline{f_1}(x), \ldots, \overline{f_l}(x)$ be l polynomials of degree at most d over $\mathbb{F}$ and let $\overline{g_1}(y), \ldots, \overline{g_m}(y)$ be m polynomials of degree at most t over $\mathbb{F}$, where $l \geq (t+1)$ and $m \geq (d+1)$, such that for every $1 \leq i \leq l$ and for every $1 \leq j \leq m$, we have $\overline{f_i}(j) = \overline{g_j}(i)$. Then there exists a unique bi-variate polynomial $\overline{F}(x, y)$ over $\mathbb{F}$ of degree-(d, t), such that $\overline{F}(x, i) = \overline{f_i}(x)$ and $\overline{F}(j, y) = \overline{g_j}(y)$, for $1 \leq i \leq l$ and $1 \leq j \leq m$..*

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

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

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

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

Next, we will show that for any $t+1 < i \leq l$, the polynomial $\overline{f_i}(x)$ also lies on $\overline{F}(x, y)$. In other words, $\overline{F}(x, i) = \overline{f_i}(x)$, for $t+1 < i \leq l$. This is easy to show because according to the lemma statement, $\overline{f_i}(j) = \overline{g_j}(i)$, for $j = 1, \ldots, d+1$ and $\overline{g_1}(i), \ldots, \overline{g_{d+1}}(i)$ lie on $\overline{F}(x, i)$ and uniquely define $\overline{F}(x, i)$. Since both $\overline{f_i}(x)$ and $\overline{F}(x, i)$ are of degree at most d, this implies that $\overline{F}(x, i) = \overline{f_i}(x)$, for $t+1 < i \leq l$. Now using similar argument, we can show that $\overline{F}(j, y) = \overline{g_j}(y)$, for $d+1 < j \leq m$. □

Lemma 5: *Let $h_0(y), \ldots, h_l(y)$ be polynomials over where $l \geq 1$ and let r be a random, non-zero element from $\mathbb{F}$. Assuming $\ell = poly(\kappa)$, if the polynomial $h_{com}(y) \stackrel{def}{=} r^0 h_0(y) + \ldots + r^l h_l(y)$ is of degree at most t, then except with probability $2^{-\Omega(\kappa)} \approx \epsilon$, each polynomial $h_0(y), \ldots, h_l(y)$ has also degree at most t.*

PROOF: On the contrary, assume that at least one of the polynomials $h_0(y), \ldots, h_l(y)$ has degree more than t. Without loss of generality, let $h_1(y)$ has the maximal degree among $h_0(y), \ldots, h_l(y)$, with degree t_{max}, where $t_{max} > t$ (in our context t_{max} will be finite). Then we write every $h_i(y)$ as $h_i(y) = c_i y^{t_{max}} + \widehat{h_i}(y)$, where $\widehat{h_i}(y)$ has degree lower than t_{max}. Then $h_{com}(y) \stackrel{def}{=} r^0h_0(y) + \ldots + r^lh_l(y)$ can be written as:

$$\begin{aligned}
h_{com}(y) &= r^0[c_0y^{t_{max}} + \widehat{h_0}(y)] + \ldots + r^l[c_ly^{t_{max}} + \widehat{h_l}(y)] \\
&= y^{t_{max}}(r^0c_0 + \ldots + r^lc_l) + \Sigma_{j=0}^{l} r^j\widehat{h_j}(y) \\
&= y^{t_{max}}c_{com} + \Sigma_{j=0}^{l} r^j\widehat{h_j}(y) \quad \text{where } c_{com} = r^0c_0 + \ldots + r^lc_l
\end{aligned}$$

By our assumption, $c_1 \neq 0$, as $h_1(y)$ has degree t_{max}. It implies that the vector $(c_0, \ldots, c_l)$ is not a complete 0 vector. Hence $c_{com} = r^0c_0 + \ldots + r^lc_l$ will be zero with probability at most $\frac{l}{|\mathbb{F}|} \approx 2^{-\Omega(\kappa)} \approx \epsilon$ (which is negligible). This is because the vector $(c_0, \ldots, c_l)$ can be considered as the set of coefficients of a polynomial, say $\mu(x)$, of degree at most l and hence the value c_{com} is the value of $\mu(x)$ at $x = r$. Now c_{com} will be zero if r happens to be one of the possible l roots of $\mu(x)$ (since the degree of $\mu(x)$ is at most l). So if r is a non-zero element, selected uniformly and randomly from $\mathbb{F}$, then except with probability ϵ, $c_{com} \neq 0$ and so $h_{com}(y)$ will have degree higher than t, which is a contradiction. $\square$