J. Cryptol. (2015) 28: 49–109
DOI: 10.1007/s00145-013-9172-7

Journal of
CRYPTOLOGY

Efficient Asynchronous Verifiable Secret Sharing and Multiparty Computation*

Arpita Patra and Ashish Choudhury

Department of Computer Science, University of Bristol, Bristol, UK
arpita.patra@bristol.ac.uk; arpitapatra10@gmail.com
partho31@gmail.com; ashish.choudhary@bristol.ac.uk

C. Pandu Rangan

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

Communicated by Tal Rabin

Received 29 November 2010
Online publication 11 December 2013

Abstract. Secure Multi-Party Computation (MPC) providing information-theoretic security allows a set of n parties to securely compute an agreed function over a finite field, 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 $t < n/4$, while statistical AMPC is possible if and only if $t < n/3$. 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 protocols. In this paper, we design two AVSS schemes 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 with the degree of sharing at most d, where $t \leq d \leq 2t$. In contrast, the existing AVSS schemes allow the degree of sharing to be at most t. The new property of our AVSS schemes simplifies the degree-reduction step for the evaluation of multiplication gates in an AMPC protocol.
2. Using our statistical AVSS scheme, we design a statistical AMPC protocol with $n = 4t + 1$ which requires an amortized communication of $\mathcal{O}(n^2)$ field elements per multiplication gate. Though this protocol has non-optimal resilience,

* A preliminary version of this paper appeared in [38]. The work was initiated when the first two authors were PhD. students in Department of Computer Science and Engineering at IIT Madras.

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 incurs an amortized communication of $\mathcal{O}(n^2)$ field elements per multiplication gate. This protocol improves on our statistical AMPC protocol as it has optimal resilience. This is the most communication efficient, optimally-resilient, perfect AMPC protocol.

Key words. Unconditional security, Fault tolerance, Communication complexity

1. Introduction

Threshold Multi-Party Computation (MPC) [8,16,28,41,44] 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 an active adversary $\mathcal{A}_t$, who can behave in any arbitrary manner during the execution of a protocol. 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 [2,5–9,16,22, 28,40,44]. In any general MPC protocol, the function $\mathcal{F}$ is expressed as an arithmetic circuit over $\mathbb{F}$, consisting of input, linear (i.e. addition), non-linear (i.e. multiplication), random and output gates over $\mathbb{F}$; the protocol then allows the parties to "evaluate" each gate of the circuit in a distributed fashion. The evaluation of multiplication gates require the maximum communication among the parties and so the focus is on measuring the communication complexity (namely the total number of bits communicated by the honest parties), required to evaluate the multiplication gates in the circuit.

The MPC problem has been studied extensively over the synchronous network model, 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 the real-world networks like the Internet. Thus a new line of research was initiated and dedicated for MPC in the asynchronous network model [5,7,9,37,43], where the messages are allowed to be delayed arbitrarily.

Unlike synchronous MPC protocols, designing asynchronous MPC (AMPC) protocols has received less attention due to its 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,[1] 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 (to avoid endless waiting) and so the communication from t (potentially slow but honest) parties may have to be ignored. In this paper, our focus is on the AMPC protocols providing information-theoretic security (that is security against a computationally unbounded $\mathcal{A}_t$). Such protocols can be categorized into two types:

1. *Perfectly-Secure AMPC or Perfect AMPC*: The protocols of this type do not involve any error in the computation. In [7], it was shown that perfectly-secure AMPC is possible if and only if $t < n/4$. Thus any perfectly-secure AMPC protocol designed with exactly $n = 4t + 1$ parties is called an *optimally-resilient*

[1] A party is called corrupted if it is under the control of the adversary, otherwise it is called honest.

perfectly-Secure AMPC protocol. Such AMPC protocols are reported in [5,7,43]. Among these, the AMPC protocol of Ref. [5] is the most communication efficient, with an amortized[2] communication complexity of $\mathcal{O}(n^3 \log |\mathbb{F}|)$ 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*: These protocols may involve a negligible error (specified by an error parameter ϵ) in the computation. From [9] it is known that statistically-secure AMPC is possible if and only if $t < n/3$. Thus any statistical AMPC protocol designed with exactly $n = 3t + 1$ parties is called an *optimally-resilient statistically-secure* AMPC protocol. Such AMPC protocols are reported in [9,37]. Among these, the AMPC protocol of Patra et al. [37] is the most efficient, with an amortized communication complexity of $\mathcal{O}(n^5 \log |\mathbb{F}|)$ bits per multiplication gate, where the computation is performed over a field $\mathbb{F} = GF(q)$, with $q > \max(n, 2^{\kappa})$, 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 a negligible error, in comparison to the error-free protocols. There are two reasons behind this anomaly: First, the corruption threshold is different for optimally-resilient statistical and perfect protocols. Namely, the perfect protocols can only tolerate up to $t < n/4$ corruptions, while in comparison, the statistical protocols have to tolerate more corruptions, namely up to $t < n/3$. 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 a statistical AVSS scheme with $n = 3t + 1$ parties, whose communication complexity matches the communication complexity of a perfect AVSS scheme with $n = 4t + 1$. An excellent informal description of this difficulty is outlined in [15].

An interesting approach used to obtain a communication efficient statistical AMPC protocol 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 setting to achieve efficiency (see, for example, [21,22]). In the asynchronous setting, this approach was reported in [39], where the authors presented a statistical AMPC protocol with $n = 4t + 1$. Following this trend in [30], the authors presented a statistical AMPC protocol with $n = 4t + 1$ (we will show later that this protocol is flawed). The (amortized) communication complexity (per multiplication gate) of the known information-theoretically secure AMPC protocols is summarized in Table 1.

In [5,18], communication efficient MPC protocols over hybrid networks that exhibit "partial synchrony" were presented, where one round of communication is assumed to be synchronous.[3] In another work, Damgård et al. [20] have reported an efficient MPC protocol over a network that assumes the concept of a "synchronization point";

[2] The amortized communication complexity is derived under the assumption that the circuit is large enough so that the terms that are independent of the circuit size can be ignored.

[3] An advantage of these protocols is that they facilitate input provision; namely the inputs of all the honest parties are considered for the evaluation of the circuit, which, otherwise, is impossible to achieve in a completely asynchronous protocol.

Table 1. Communication complexity (CC) in bits per multiplication gate of the known information-theoretically secure 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
[7,14]	Perfect	$t < n/4$ (optimal)	$\mathcal{O}(n^6 \log \lvert\mathbb{F}\rvert)$
[43]	Perfect	$t < n/4$ (optimal)	$\Omega(n^5 \log \lvert\mathbb{F}\rvert)$
[5]	Perfect	$t < n/4$ (optimal)	$\mathcal{O}(n^3 \log \lvert\mathbb{F}\rvert)$
[9]	Statistical	$t < n/3$ (optimal)	$\Omega(n^{11}(\log \lvert\mathbb{F}\rvert)^4)$
[37]	Statistical	$t < n/3$ (optimal)	$\mathcal{O}(n^5 \log \lvert\mathbb{F}\rvert)$
[39]	Statistical	$t < n/4$ (non-optimal)	$\mathcal{O}(n^4 \log \lvert\mathbb{F}\rvert)$
[30]	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)	$\mathcal{O}(n^2 \log \lvert\mathbb{F}\rvert)$

i.e. the network is asynchronous before and after the synchronization point. We will not consider the protocols of Refs. [5,18,20] 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 protocols 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 Huang et al. [30]. However, we show that this protocol is flawed. We further design a new statistically-secure AMPC protocol with $n = 4t + 1$, having an amortized communication complexity of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits per multiplication gate.[4] Our protocol achieves its goal without using the player-elimination framework of Ref. [29], which was used in [30]. We note that our statistical AMPC protocol has non-optimal resilience.
2. We present a perfectly-secure AMPC protocol with $n = 4t + 1$, with an amortized communication complexity of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits per multiplication gate. Our perfect AMPC protocol has optimal resilience. From Table 1, the best known perfect AMPC protocol with optimal resilience [5] incurs a communication of $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits per multiplication gate. Hence our AMPC protocol provides the best communication complexity among all the known optimally-resilient, perfect AMPC protocols.

To design our 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 protocol), while the second one is perfectly-secure (has optimal resilience and is used in our perfect AMPC protocol). Both these AVSS schemes achieve some common interesting properties, which were not achieved by the previous schemes. Even

[4] We note that after the submission of this article, a more efficient statistical AMPC protocol with $n = 4t + 1$ and an amortized communication complexity of $\mathcal{O}(n \log |\mathbb{F}|)$ bits per multiplication gate was presented by Choudhury et al. in [18]; the techniques used in their protocol are different from ours and discussing their protocol is out of scope of this article.

though our statistical and perfect AVSS schemes have the same communication complexity (more on this later), we present both of them, as 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) [1,15, 24,32,36], 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 the n 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 [4,6,8,9,15–17,19,23,25,27,31,34,40,41] 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 schemes 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 network model was initiated. VSS schemes for the asynchronous networks are called Asynchronous VSS (AVSS) schemes.

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 number of parties does not get any information about the shared 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. Typically, such an AVSS scheme is used in the information-theoretically secure AMPC protocols, as it allows the parties to locally perform any linear computation of shared values. Information-theoretically secure AVSS schemes (for the threshold access structure) can be categorized into two classes:

1. *Perfectly-Secure AVSS or Perfect AVSS:* A scheme of this type satisfies the requirements of AVSS[5] without any error. Perfectly-secure AVSS tolerating $\mathcal{A}_t$ is possible if and only if $t < n/4$ [7,14]. Hence, we call a perfectly-secure AVSS scheme designed with exactly $n = 4t + 1$ parties an *optimally-resilient, perfectly-secure* AVSS scheme. Such AVSS schemes are proposed in [5,7,14].
2. *Statistically-Secure AVSS or Statistical AVSS:* A scheme of this type satisfies the requirements of AVSS except with a negligible error (specified by an error parameter ϵ). Statistical AVSS is possible if and only if $t < n/3$ [9,15]. To the best

[5] See Definition 1 for the properties of AVSS.

of our knowledge, the AVSS schemes of Refs. [9,15,36,37] are the only known optimally-resilient statistical AVSS schemes (i.e. with $n = 3t + 1$).

The AVSS schemes based on polynomial interpolation are the most popular ones and they have been used in almost all the existing information-theoretically secure AMPC protocols. 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 an AMPC protocol, it is important to see how AVSS schemes are used in the AMPC protocols; the rest of this section is dedicated for the same.

The polynomial based AVSS schemes are an important building block for designing AMPC protocols tolerating $\mathcal{A}_t$. The sharing phase of such an AVSS scheme enforces the dealer to t-share a value (even if the dealer is corrupted). Informally, a value 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 for two purposes: first to make the parties commit and share their inputs and second to generate several shared 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-shares its input x_i, where x_i is P_i's input for the computation. Then the parties agree on a common subset of $n - t$ parties, say 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 an 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 Ben-Or et al. [8]. 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 shares (of the final output) 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 an AMPC protocol 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 evaluation 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 $[d]_t$, a 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 $[c]_t \cdot [d]_t = [e]_{2t}$, instead of $[e]_t$. If $[e]_{2t}$ is not converted to $[e]_t$ then further multiplication of (shared) 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 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 Beaver's circuit-randomization method [3], where the multiplication gates are evaluated using pre-computed, t-shared random multiplication triples (which can be generated in a pre-processing stage, prior to the beginning of the computation), unknown to the adversary.[6] This approach is used in almost all the MPC protocols (both synchronous and asynchronous) proposed in the recent years [4,6,10,22]. An alternative to the above approach, proposed in [5] and also used by us in this paper is to evaluate the multiplication gates using pre-computed $(t, 2t)$-sharing of random values, unknown to the adversary. A $(t, 2t)$-sharing [5] of a value $r \in \mathbb{F}$ consists of a t-sharing and a $2t$-sharing of r via independent polynomials of degree at most t and $2t$, respectively; so both $[r]_t$ and $[r]_{2t}$ will be available to the parties. Given a $(t, 2t)$-sharing of a pre-computed random value r unknown to the adversary, 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 latter computation follows from the linearity property of $2t$-sharing. This 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 and unknown to the adversary. 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 multiplication triples. The evaluation cost of a multiplication gate in both the approaches is nearly the same. For the multiplication 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 efficiently generating $(t, 2t)$-sharing of random values. 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 invoking an AVSS scheme $3t + 1$ 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 single t-sharing and a single $(2t - 1)$-sharing (more on this in the

[6] Such shared triples are of the form $([x]_t, [y]_t, [z]_t)$, where x and y are random and unknown to the adversary, with $z = x \cdot y$.

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

Reference	Type	Number of secrets shared	Degree of the sharing	Communication complexity in bits
[7]	Perfect	1	t	$\mathcal{O}(n^3 \log \lvert\mathbb{F}\rvert)$
[5]	Perfect	ℓ, where $\ell \geq 1$	t	$\mathcal{O}(\ell n^2 \log \lvert\mathbb{F}\rvert)$
This article	Statistical	ℓ, where $\ell \geq 1$	d, 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, for any given d, where $t \leq d \leq 2t$	$\mathcal{O}(\ell n^2 \log \lvert\mathbb{F}\rvert)$

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 they do so with the same communication complexity as the best known existing AVSS scheme of Ref. [5], which generates only t-sharing.

Our Contributions for AVSS We present two AVSS schemes with $4t + 1$ parties; one is statistically-secure (with non-optimal resilience) and the other one is perfectly-secure (with optimal resilience). These schemes share an interesting property: the sharing phase of these schemes allow 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 $v \in \mathbb{F}$ to be shared[7] 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 party 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 $\ell \geq 1$) concurrently, such that each individual value is d-shared. The advantage of the enhanced schemes that generate concurrent sharing of ℓ values, over ℓ instances of the basic schemes for individually generating each sharing is that the former allows us to combine the broadcast (public) communication (needed in the protocol) for all the ℓ values and therefore the broadcast communication remains independent of ℓ (namely the number of values shared). This is an important feature since implementing the broadcast primitive by a protocol in the asynchronous setting [13] is expensive and we must aim to keep the broadcast communication independent of ℓ. Table 2 gives a comparison of our AVSS schemes with the existing AVSS schemes (with $4t + 1$ parties) in the literature.

We next highlight the following two different perspectives of our AVSS schemes:

(a) They generate d-sharing of ℓ values, with communication complexity $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits.
(b) They share $\ell(d + 1 - t)$ values in the sense of "packed secret-sharing" [26] where $\ell \geq 1$ and $t \leq d \leq 2t$, with communication complexity $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits.

The two different perspectives have two different implications. The first perspective allows us to design a method for generating $(t, 2t)$-sharing of random values with a better

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

communication complexity than the existing method of Ref. [5]. The second perspective implies that the amortized cost of sharing a single value tolerating an active adversary is $\mathcal{O}(n)$ field elements, which matches the complexity of sharing a single value tolerating a passive adversary (e.g. Shamir's secret-sharing [42]). 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 Ref. [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 $P(x) = r^{(0)} + r^{(1)} \cdot x + \cdots + r^{(t)} \cdot x^t$ and $Q(x) = r^{(0)} + r^{(t+1)} \cdot x + \cdots + r^{(3t)} \cdot x^{2t}$ 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 an additional "randomness extraction" step. Using the AVSS scheme of Ref. [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[8] for n parties.

   We generate $(t, 2t)$-sharing of a random value from t-sharing of a single random value and $(2t - 1)$-sharing of another 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 P_i can locally compute its share corresponding to $[r]_{2t}$ by computing $h(i) = f(i) + i \cdot g(i)$, where $f(i)$ and $g(i)$ are the shares for P_i corresponding to $[r]_t$ and $[s]_{2t-1}$. 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 t-share a random value and $(2t - 1)$-share another random value. This step is followed by an additional randomness-extraction step. 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[9] for n parties. Thus, we note a reduction of $\Theta(n)$ over Ref. [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 just because of our different way of generating a $(t, 2t)$-sharing. The approach used by us is not applicable to [5] because neither the AVSS scheme of [5], nor any prior AVSS scheme, can be used to $(2t - 1)$-share a value.
2. *Packed secret-sharing in the asynchronous setting*: 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

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

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

the adversary. Intuitively, this implies that from the view-point of the adversary, there exists $d+1-t$ "degree of freedom". This further implies that using a single polynomial of degree at most d, an honest dealer can share $d+1-t$ secrets. This is the reminiscent of packed secret-sharing, introduced in [26] for the synchronous setting. Our constructions provide packed secret-sharing scheme in the asynchronous setting 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, Shamir secret-sharing scheme [42]).

Our schemes are useful in applications where a party needs to share multiple values. For example, *common coin* [14] is 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 an AVSS scheme. Using our packed secret-sharing, a party can share n values using $\ell = n/(d+1-t)$ polynomials, each of degree at most d (through a single polynomial, the party can share $d+1-t$ values). Substituting the maximum value of $d = 2t$ and using the fact that $t = \Theta(n)$, we find that $\ell = n/(d+1-t) = \mathcal{O}(1)$. This implies that each party can now share n values by invoking our AVSS schemes a constant number of times, by setting $\ell = n/(d+1-t)$ and $d = 2t$. This overall reduces the communication complexity of the ABA protocol.[10]

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 Schemes Our AVSS schemes have the following common properties:

1. Designed with $n = 4t+1$.
2. Generate 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, the schemes are completely independent. We further believe that some of the techniques may lead to an improved AVSS scheme, which may further lead to a more efficient AMPC and ABA 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 protocol by using the approach outlined earlier. It is the underlying AVSS, which makes the resulting AMPC protocol either statistical or perfect. We next discuss the ideas used in our AVSS schemes.

1.3. *Overview of 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 bivariate polynomial,

[10] Giving the exact details of the common coin and ABA is out of scope of the current article and so we avoid discussing them.

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

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

as used in several existing schemes [19,25,27,31,35]. In the existing schemes, the dealer D selects a random bivariate polynomial $F(x, y)$ of degree at most t in x and y, subject to the condition that $F(0, 0) = s$, where s is the secret to be shared. We observe that given $n = 4t + 1$, the dealer can use a bivariate polynomial of degree at most d in x and t in y for all d with $t \leq d \leq 2t$. This of course does not come for free and calls for new ideas on top of the existing schemes. By being able to hide the secret in a bivariate polynomial of degree-(d, t) (we use this notation to denote bivariate 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 a 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 bivariate polynomial $F(x, y)$ of degree-(d, t) with $F(0, 0) = s$ and giving 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 ith row and the ith 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 bivariate polynomial $F(x, y)$ facilitates *two-level sharing* of s (see Fig. 1): at the top level, s is d-shared through the polynomial $f_0(x)$ and then at 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 its 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.

Although the second level sharing of the shares of the secret does not seem to serve any purpose for an honest dealer D, it is required for two different reasons to deal with a corrupted D. First, it ensures that D indeed d-shares (i.e. the underlying sharing polynomial has degree at most d) the secret. Second, it is required to "complete" d-sharing of s, since a corrupted D may not give 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 compute $Sh_i = g_i(0)$. 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 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 bivariate 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 follows from the theory of error-correcting codes.

We now explain how the above idea is implemented in our schemes. After the dealer distributes the univariate polynomials, the parties try to identify and agree on a common subset of $3t + 1$ parties, say CORE, such that the $f_i(x)$ polynomials of the honest parties in CORE lie on a unique bivariate polynomial,[11] 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 bivariate polynomial $\overline{F}(x, y)$ of degree-(d, t), which will be considered as D's committed bivariate 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 $\overline{s}$ will be 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 $|\text{CORE}| \geq 3t + 1$, the parties in CORE can send their share-share of $\overline{Sh}_i$ to P_i and enable P_i to robustly reconstruct $\overline{g}_i(y)$ 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 a 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 in a statistical AVSS scheme. The second method identifies a CORE without any error and results in a perfect AVSS scheme.

[11] A univariate polynomial $f_i(x)$ is said to lie on a bivariate polynomial $F(x, y)$ if $f_i(x) = F(x, i)$.

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 Sect. 3. This is followed by the discussion on the modifications required to extend these schemes to share multiple values concurrently in Sect. 4. The protocols for generating $(t, 2t)$-sharing using our AVSS schemes are presented in Sect. 5. In Sect. 6, we present our AMPC protocols, followed by a brief discussion on the application of our AVSS schemes in packed secret-sharing in Sect. 7. In Sect. 8, we discuss the proposed statistical AMPC protocol of [30] and show that it is flawed.

2. Definitions and Preliminaries

2.1. *Model*

We consider a completely asynchronous network, where we have a set of $n = 4t + 1$ parties, say $\mathcal{P} = \{P_1, \ldots, P_n\}$, connected by pairwise secure and authentic channels; each party is modeled as a probabilistic polynomial time Turing machine. We assume that there exists a computationally unbounded adversary $\mathcal{A}_t$, who can actively corrupt at most t out of the n parties and make them behave in any arbitrary manner during the execution of a protocol.

The underlying network is asynchronous, where the communication channels between the parties have arbitrary, yet finite delay (i.e. the messages are guaranteed to reach their destinations eventually). Moreover, the order in which the messages reach their destinations may be different from the order in which they were sent. 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.

As in [14], we consider a computation (namely a protocol execution) in the asynchronous model as a sequence of *atomic steps*, where a single party is *active* in each such step. A party gets activated by receiving a message after which it performs an internal computation and then possibly sends messages on its outgoing channels. The order of the atomic steps are controlled by a "scheduler", which is controlled by $\mathcal{A}_t$. At the beginning of the computation, each party will be in a special *start* state. We say a party has *terminated/completed* the computation if it reaches a *halt* state, after which it does not perform any further computation. A protocol execution is said to be *complete* if each (honest) party terminates the protocol. Notice that the executions that complete do so after a finite number of steps.

For simplicity, we assume the adversary to be static, who decides the set of t parties to be corrupted at the beginning of the execution of a 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 a protocol, provided that more than t parties are not under the control of the adversary.

2.2. *Definitions*

The computation in our protocols is performed over a finite field $\mathbb{F}$; 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 $n = \text{poly}(\kappa)$; so every field element can be represented by $\log |\mathbb{F}|$ bits.

We next recall the definition of AVSS from [7,14].

Definition 1 (Asynchronous Verifiable Secret Sharing (AVSS) [7,14]). Let (Sh, Rec) be a pair of protocols for the n parties, where a dealer $\mathsf{D} \in \mathcal{P}$ has a private input $s \in \mathbb{F}$ for Sh. We say that (Sh, Rec) is an AVSS scheme if the following requirements hold for every possible $\mathcal{A}_t$:

1. **Termination**:
   (a) If D is honest and all the honest parties participate in the protocol Sh, then each honest party eventually terminates the protocol Sh.
   (b) If some honest party terminates Sh, then irrespective of the behavior of D, each honest party eventually terminates Sh.
   (c) If all the honest parties invoke Rec, then each honest party eventually terminates Rec.
2. **Correctness**: If some honest party terminates Sh, then there exists a fixed value $\overline{s} \in \mathbb{F}$, such that the following requirements hold[12]:
   (a) If D is *honest* then $\overline{s} = s$ and each honest party upon completing the protocol Rec, outputs s.
   (b) Even if D is *corrupted*, each honest party upon completing Rec outputs $\overline{s}$, irrespective of the behavior of the corrupted parties. This property is also known as *strong commitment*.
3. **Secrecy**: If D is honest then the adversary's view during Sh reveals no information about s in the information-theoretic sense; i.e. the adversary's view is identically distributed for all possible 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 present 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[13] at least $(1 - \epsilon)$, for a given error parameter ϵ, then such a scheme is called a statistical AVSS scheme. 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 scheme.

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

[12] We often say that D has committed/shared $\overline{s}$ during Sh.

[13] In the rest of the paper, all probabilities are taken over the random coins of the honest parties.

Definition 3 (d-Sharing and $(t, 2t)$-sharing [5]). A value $s \in \mathbb{F}$ is said to be d-shared among $\mathcal{P}$ if there exists a polynomial over $\mathbb{F}$, say $f(x)$, 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.

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 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 ith 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 ith share of $y^{(1)}, \ldots, y^{(m')}$, respectively. We say that the *parties (locally) compute/generate* $([y^{(1)}]_d, \ldots, [y^{(m')}]_d) = H([x^{(1)}]_d, \ldots, [x^{(m)}]_d)$ to mean the above.

Throughout the paper, we say that a bivariate 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 [7]. Later [9] followed the same definition; though they presented the definition in the style of a "property based" definition. In the information-theoretic world, the definition of [9] and [7] are in essence "equivalent." Since then, all the papers on information-theoretic AMPC follow the definition presented in [9] 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 [9] presented below. However using standard techniques, our protocols can be proved secure according to the real-world/ideal-world definition of [7], without affecting their efficiency.

Definition 4 (Secure Asynchronous Multi-Party Computation (AMPC) [9]). Let $\mathcal{F} : \mathbb{F}^n \rightarrow \mathbb{F}^n$ be a publicly known function and let party P_i have a private input $x_i \in \mathbb{F}$. Any asynchronous multiparty computation consists of three stages. In the first stage, each party P_i commits its input. Even if P_i is faulty, if it completed this step, then it 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 common subset C of at least $n - t$ committed inputs. In the last stage the parties compute $\mathcal{F}(y_1, \ldots, y_n)$, where $y_i = x_i'$ if $P_i \in C$, otherwise $y_i = 0$.

An asynchronous protocol Π among the n parties for computing the function $\mathcal{F}$ is called an AMPC protocol if it satisfies the following conditions for every possible $\mathcal{A}_t$:

1. **Termination**: If all the honest parties participate in the protocol, then every honest party eventually terminates Π.

2. **Correctness**: Every honest party outputs $\mathcal{F}(y_1, \ldots, y_n)$ after completing Π, irrespective of the behavior of the corrupted parties.
3. **Secrecy**: The adversary obtains no additional information (in the information-theoretic sense) about the inputs of the honest parties during Π, other than what is inferred from the input and the output of the corrupted parties.

Based on whether the above properties are achieved with a 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 protocol. 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 protocol.

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

2.3. *Primitives Used*

Asynchronous Broadcast In our protocols, we use the asynchronous broadcast primitive, which was introduced and elegantly implemented by Bracha [13]; the primitive allows a special party $\mathsf{S} \in \mathcal{P}$, called sender, to send a message identically to all the parties. More formally:

Definition 6 (Asynchronous broadcast [15]). Let Π be an asynchronous protocol for the n parties initiated by a special party $\mathsf{S} \in \mathcal{P}$, having input m (the message to be broadcast). We say that Π is an asynchronous broadcast protocol if the following hold, for every possible $\mathcal{A}_t$:

1. **Termination**:
   (a) If S is honest and all the honest parties participate in the protocol, then each honest party eventually terminates the protocol.
   (b) Irrespective of the behavior of S, if any honest party terminates the protocol then each honest party eventually terminates the protocol.
2. **Correctness**: If the honest parties terminate the protocol then they do so with a common output $m^\star$. Furthermore, if the sender is honest then $m^\star = m$.

Bracha presented a protocol called A-cast, for realizing the asynchronous broadcast primitive; the protocol can actually tolerate up to $t < n/3$ corruptions. For the sake of completeness, we recall the Bracha's A-cast protocol from Canetti [14] and present it in Fig. 2.

Theorem 1 ([14]). *Protocol* A-cast *incurs a communication of* $\mathcal{O}(\ell n^2)$ *bits to broadcast an* ℓ *bit message.*

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

A-cast(m)

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

1. Send the message $(\texttt{MSG}, m)$ to all the parties.

CODE FOR THE PARTY P_i — every party in $\mathcal{P}$ (including S) executes this code:

1. Upon receiving a message $(\texttt{MSG}, m)$ from S, send the message $(\texttt{ECHO}, m)$ to all the parties.
2. Upon receiving $n - t$ messages $(\texttt{ECHO}, m^\star)$ that agree on the value of $m^\star$, send the message $(\texttt{READY}, m^\star)$ to all the parties.
3. Upon receiving $t + 1$ messages $(\texttt{READY}, m^\star)$ that agree on the value of $m^\star$, send the message $(\texttt{READY}, m^\star)$ to all the parties.
4. Upon receiving $n - t$ messages $(\texttt{READY}, m^\star)$ that agree on the value of $m^\star$, send the message $(\texttt{OK}, m^\star)$ to all the parties, output $m^\star$ and terminate the protocol.

Fig. 2. Bracha's asynchronous broadcast protocol tolerating $t < n/3$ corruptions.

Terminology 1 (Terminologies for using the A-cast protocol). *We say that*:

1. *P_i broadcasts m: this means that P_i acts as an* S *and invokes an instance of* A-cast *to broadcast m.*
2. *P_j receives m from the broadcast of P_i: this means that P_j (as a receiver) completes the execution of P_i's* A-cast *(namely the instance of the* A-cast *protocol where P_i is the sender), with m as the output.*

*Agreement on a Common Subset (*ACS*)* The ACS protocol [7,9], is used in all the existing AMPC protocols (including ours). It allows the (honest) parties to agree on a common subset 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 is no restriction 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 (possibly 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 [30] that stems from the fact that Huang et al. [30] 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 broadcast a value and the property Q is whether a party has broadcasted or not. The termination property of broadcast (namely the A-cast protocol) ensures that if some honest P_j finds some party, say P_α, to satisfy Q (that is P_j received a value from the broadcast of P_α), 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 subset of $n - t$ parties who have broadcast some value.

The idea behind the ACS protocol is to execute n instances of an *asynchronous Byzantine Agreement* (ABA) protocol [14], one on the behalf of each party to decide whether it will be in the common subset. For the sake of completeness, we present the description of the protocol ACS (taken from Ben-Or et al. [9]) in Fig. 3.

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 an ABA protocol executed for $P_j \in \mathcal{P}$ to decide whether P_j will be in the common subset.
2. Upon terminating $n - t$ instances of ABA protocol 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 instances of ABA protocol, let your $\mathcal{S}_i$ 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 $\mathcal{S}_i$ and terminate.

Fig. 3. Protocol for the agreement on a common subset.

Theorem 2 ([9]). *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 a constant number of times in our AMPC protocols, we choose not to be explicit on its communication complexity.

*Online Error Correction (*OEC*) ([5,14])* The next protocol we discuss is OEC, which can be viewed as the method of applying the Reed–Solomon (RS) error-correction [33] in the asynchronous setting. Given a value which is d-shared among a set of parties $\overline{\mathcal{P}} \subseteq \mathcal{P}$ with $d < (|\overline{\mathcal{P}}| - 2t)$, the goal is to make some designated party, say P_R, reconstruct the value robustly (actually OEC allows P_R to reconstruct the entire polynomial through which the value is d-shared). In the synchronous setting, this can be achieved by asking every party in $\overline{\mathcal{P}}$ to send its share to P_R, who can apply the RS error-correction to reconstruct the value. Given the condition $d < (|\overline{\mathcal{P}}| - 2t)$, the reconstruction will be robust. In the asynchronous setting, achieving the same goal requires a bit of trick.

The intuition behind OEC is that P_R keeps waiting till it receives $d + t + 1$ values, all of which lie on a unique polynomial of degree d. This step requires applying the RS error-correction repeatedly. We denote an RS error-correcting procedure as RS-Dec(d, r, W) that takes as input a vector W of shares (possibly incorrect) of a d-shared value (that we would like to reconstruct) and tries to output a polynomial of degree d, by correcting at most r errors in W. Coding theory [33] says that RS-Dec can correct r errors in W and correctly interpolate the original polynomial provided that $|W| \geq d + 2r + 1$. There are several efficient implementations of RS-Dec (for example, the Berlekamp–Welch algorithm [33]). Once P_R receives $d + t + 1$ values that lie on a unique polynomial of degree d (returned by RS-Dec), then that unique polynomial is the actual polynomial, say $Q(x)$, of degree d that defines d-sharing of $Q(0)$. This is because at least $d + 1$ values out of the $d + t + 1$ values are from the honest parties, which uniquely define the original polynomial $Q(x)$. Note that the corrupted parties in $\overline{\mathcal{P}}$ may send wrong values to P_R. But there are at least $|\overline{\mathcal{P}}| - t \geq d + t + 1$ honest parties in the set $\overline{\mathcal{P}}$ whose values will be eventually received by P_R and so P_R will eventually terminate the process. The above procedure is nothing but applying the RS error-correction algorithm in an "online" fashion.

OEC

Setting: A value is d-shared among a set of parties $\overline{\mathcal{P}} \subseteq \mathcal{P}$, where $Q(x)$ is the underlying sharing polynomial of degree at most d, where $d < (|\overline{\mathcal{P}}| - 2t)$; each party $P_i \in \overline{\mathcal{P}}$ holds the share $v_i = Q(i)$ of $Q(0)$. A designated party $P_R \in \mathcal{P}$, is supposed to reconstruct the polynomial $Q(x)$.

CODE FOR THE PARTY P_i — Every party in $\overline{\mathcal{P}}$ executes the following code:

1. Send the share v_i to the party P_R and terminate.

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

1. Let $\mathcal{W}$ denote the set of parties in $\overline{\mathcal{P}}$ from whom P_R has received the shares and I_r denote the values received from the parties in $\mathcal{W}$, when $\mathcal{W}$ contains exactly $d + t + 1 + r$ parties.
2. Wait until $|\mathcal{W}| \geq d + t + 1 + r$. Execute RS-Dec$(d, r, I_r)$ to get a polynomial $Q_r(x)$ of degree d. If no polynomial is output, then skip the next step and proceed to the next iteration.
3. If for at least $d + t + 1$ values $v_i \in I_r$ it holds that $Q_r(i) = v_i$, then output $Q_r(x)$ and terminate. Otherwise, proceed to the next iteration.

Fig. 4. Protocol for online error-correction.

The steps for the OEC are now presented in Fig. 4. The current description is inspired from Canetti [14] (skipping several other formal details).

Theorem 3 ([5,14]). *Let a value be d-shared among a set of parties $\overline{\mathcal{P}} \subseteq \mathcal{P}$ where $d < (|\overline{\mathcal{P}}| - 2t)$ and let $Q(x)$ be the underlying sharing polynomial. Moreover, let $P_R \in \mathcal{P}$ be a designated party, who is supposed to reconstruct $Q(x)$. Then protocol* OEC *achieves the following properties for every possible $\mathcal{A}_t$:*

1. ***Termination***:
   (a) *Every honest party in $\overline{\mathcal{P}}$ eventually terminates the protocol.*
   (b) *If P_R is honest then it eventually terminates the protocol.*
1. ***Correctness***: *Party P_R upon terminating outputs $Q(x)$.*
2. ***Secrecy***: *If P_α is honest then $\mathcal{A}_t$ obtains no additional information about $Q(0)$.*
3. ***Communication complexity***: *The protocol incurs a total communication of $\mathcal{O}(n \log |\mathbb{F}|)$ bits.*

Proof. The termination property is argued as follows. The honest parties in $\overline{\mathcal{P}}$ will terminate the protocol trivially after sending their shares to P_R. We now argue that (an honest) P_R will terminate the protocol, as well. Let $\hat{r}$ parties in $\overline{\mathcal{P}}$ be corrupted, where $\hat{r} \leq t$. Further assume that $\hat{r}_1$ corrupted parties send wrong values and $\hat{r}_2$ corrupted parties send nothing ever, subject to $\hat{r}_1 + \hat{r}_2 = \hat{r}$. Consider the $(t - \hat{r}_2)$th iteration; since $\hat{r}_2$ parties in $\overline{\mathcal{P}}$ never send any value, P_R will receive $d + t + 1 + t - \hat{r}_2$ distinct values on the polynomial $Q(x)$, of which $\hat{r_1}$ are corrupted. Since $|I_{t-\hat{r_2}}| = d + t + 1 + t - \hat{r_2} \geq d + 2\hat{r_1} + 1$, the algorithm RS-Dec will correct $\hat{r_1}$ errors and will return $Q_{t-\hat{r}_2}(x) = Q(x)$ during the $(t - \hat{r_2})$th iteration. Therefore the protocol will terminate at the latest after the $(t - \hat{r_2})$th iteration.

To argue correctness, assume that P_R terminates during the rth iteration and outputs $Q_r(x)$ such that the polynomial $Q_r(x)$ is consistent with $d + t + 1$ values from I_r. To prove the correctness, we now show that $Q_r(x) = Q(x)$. However, the equality follows from the fact that at least $d + 1$ values in I_r belong to the honest parties and thus they

lie on $Q(x)$, as well. In other words, these $d+1$ values are the common points of the two polynomials which are of degree at most d.

The secrecy is argued as follows. It is easy to see that if P_R is honest, then $\mathcal{A}_t$ gets no additional information about $Q(0)$. Since the (honest) parties in $\overline{\mathcal{P}}$ privately send their shares to P_R, no additional information about $Q(0)$ or the values of $Q(x)$ is revealed to $\mathcal{A}_t$ during OEC. In the protocol, each party in $\overline{\mathcal{P}}$ sends its share to P_R, incurring a communication of $\mathcal{O}(n)$ field elements. □

Randomness Extraction Here, we discuss about a well-known method for randomness-extraction in the information-theoretic setting. 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 goal is to compute K values, say $b_1, \ldots, b_K$, from $a_1, \ldots, a_N$, which are uniformly distributed over $\mathbb{F}$ and are information-theoretically secure. This is achieved through the following well-known method introduced in [11,12]: 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 $b_1 = f(N), \ldots, b_K = f(N+K-1)$. We call this algorithm Ext and invoke it as $(b_1, \ldots, b_K) = \mathsf{Ext}(a_1, \ldots, a_N)$. It is easy to see that $b_1, \ldots, b_K$ computed as above will be information-theoretically secure; this is because there exists a one-to-one mapping between the K information-theoretically secure values in $a_1, \ldots, a_N$ and $b_1, \ldots, b_K$. 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. Looking ahead, we exploit some interesting properties of (n,t)-star in order to build our perfect AVSS scheme. An (n,t)-star is defined as follows:

Definition 7 ((n,t)-star [7,14]). Let G be an undirected graph with the n parties in $\mathcal{P}$ as its vertex set. We say that a pair $(\mathbf{C}, \mathbf{D})$ of sets with $\mathbf{C} \subseteq \mathbf{D} \subseteq \mathcal{P}$ is an (n,t)-star in G, if the following hold:

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

In [7], the authors 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)-$\overline{\text{star}}$ in $\overline{G}$ where an (n,t)-$\overline{\text{star}}$ is a pair $(\mathbf{C}, \mathbf{D})$ of sets with $\mathbf{C} \subseteq \mathbf{D} \subseteq \mathcal{P}$, satisfying the following conditions:

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

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

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

Clearly, a pair $(\mathbf{C}, \mathbf{D})$ representing an (n,t)-$\overline{\text{star}}$ in $\overline{G}$, is an (n,t)-star in G. Recasting the task of Find-Star in terms of the complementary graph $\overline{G}$, we say that Find-Star outputs either an (n,t)-$\overline{\text{star}}$, 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)-$\overline{\text{star}}$. For simple notation, we denote $\overline{G}$ by H. The algorithm Find-Star is presented in Fig. 5.

Theorem 4 ([14]). *If* Find-Star *outputs* $(\mathbf{C}, \mathbf{D})$ *on input graph* H, *then* $(\mathbf{C}, \mathbf{D})$ *is an* (n,t)-$\overline{\text{star}}$ *in* H.

3. AVSS for Sharing a Single Secret

In this section, we present AVSS schemes that allow a dealer $\mathsf{D} \in \mathcal{P}$ (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 and the perfect scheme. The names of such common protocols are not prefixed by 'S' or 'P'.

Structurally, the sharing protocol (S-Sh and P-Sh) 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 also eventually terminates 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 Sect. 1.3, D selects a random bivariate polynomial $F(x,y)$ of degree-(d,t), with s as the constant term. In protocol S-Distr, D hands the ith 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 ith row and column polynomial, respectively (in connection with Fig. 1).

2. *Verification & Agreement on* CORE: The protocols for this phase are S-Ver-Agree and P-Ver-Agree, respectively. Though the goal is the 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 "sufficient" 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, $\overline{f}_i(x) = \overline{F}(x, i)$ holds 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 a 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 "pairwise consistent", then all these polynomials lie on a unique bivariate polynomial of degree-(d, t). More formally:

Lemma 1. *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 bivariate polynomial over $\mathbb{F}$ of degree-(d, t), say $\overline{F}(x, y)$, 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 in [14]. 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 its 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 bivariate 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 $\overline{s}$

Gen

Setting: The parties in $\mathcal{P}$ have agreed on a CORE $\subseteq \mathcal{P}$, where $|\text{CORE}| \geq 3t+1$. Party $P_i \in$ 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 bivariate polynomial, say $\overline{F}(x,y)$ of degree-(d,t). The goal is to enable every party $P_i \in \mathcal{P}$ reconstruct the polynomial $\overline{g}_i(y) = \overline{F}(i,y)$ and compute its share of the committed secret $\overline{F}(0,0)$.

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

1. If $P_i \in$ CORE, then for $j = 1, \ldots, n$, send $\overline{f}_i(j)$ to the party P_j [a].
2. Apply OEC on the $\overline{f}_j(i)$ values received from the parties P_j belonging to CORE to reconstruct the polynomial $\overline{g}_i(y)$ of degree at most t, output $Sh_i = \overline{g}_i(0)$ and terminate.

[a] 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.

Fig. 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.

with $Sh_i = \overline{f}_0(i)$ being the ith 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 computes its 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.

We state the following lemma for the protocol Gen.

Lemma 2. *Let the honest parties have agreed upon a* CORE, *satisfying the properties discussed above. Then protocol* Gen *satisfies the following properties for every possible* $\mathcal{A}_t$:

1. *It generates d-sharing of $\overline{s} = \overline{F}(0,0)$. Moreover, if* D *is honest, then $\overline{s} = s$ where s is* D*'s secret.*
2. *The protocol requires a 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) $P_i \in$ CORE lies on a bivariate polynomial $\overline{F}(x,y)$ of degree-(d,t). Moreover, if D is honest then $\overline{F}(x,y)$ is the same bivariate 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 $|\text{CORE}| \geq 3t+1$. Substituting $\overline{\mathcal{P}} = \text{CORE}$ 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 $\overline{f}_j(i)$ values (which are the same as $\overline{g}_i(j)$ values) 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 communication of n elements from $\mathbb{F}$. So this requires a total 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 the appropriate protocols for a phase

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 a 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 a CORE is generated and agreed upon in the previous phase, then the parties execute the protocol Gen and terminate.

Protocol Rec **for the Reconstruction Phase**

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

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

Fig. 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.

(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 3. *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 incurs a communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.*

Proof. The fact that every honest party will eventually reconstruct s follows from the properties of OEC (see Theorem 3) by noting that $|\overline{\mathcal{P}}| = |\mathcal{P}| = 4t + 1$ and the maximum degree of sharing of s is $2t$, which is strictly less than $|\mathcal{P}| - 2t$. 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.

3.1.1. *Protocol* S-Distr

Here D on having a secret s for sharing, selects a random bivariate polynomial $F(x, y)$ of degree-(d, t) with the constant term s and sends to P_i the ith row polynomial. In addition, D also distributes some "additional" information which will be used later to preserve the secrecy of s, during the probabilistic checks performed during the 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 masking polynomials indexed as $(P_i, \star)$ are *associated* with party P_i and $t + 1$ masking polynomials are associated with each P_i. The reason for selecting $(t + 1)n$ masking polynomials will be clear when we present the protocol S-Ver-Agree. Now the protocol S-Distr is presented in Fig. 8.

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 and terminate:
   (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.

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

We make the following claim about S-Distr, that trivially follows from the fact that $d \leq 2t$.

Claim 1. *In protocol* S-Distr, D *communicates* $\mathcal{O}((nd + n^3) \log |\mathbb{F}|) = \mathcal{O}(n^3 \log |\mathbb{F}|)$ *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 bivariate 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; moreover let $\overline{m}_{(P_j,1)}(i), \ldots, \overline{m}_{(P_j,t+1)}(i)$ be the shares of the masking polynomials received by P_i, for $j = 1, \ldots, n$. If D is honest then $\overline{f}_i(x) = f_i(x)$ and $\overline{m}_{(P_j,k)}(i) = m_{(P_j,k)}(i)$, for $k = 1, \ldots, t + 1$. The properties of bivariate polynomials of degree-(d, t) say that if indeed a CORE exists then the points $\{\overline{f}_i(j) : P_i \in \text{CORE}\}$ 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 jth 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 a 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 univariate polynomials has degree at most t, then with very high probability, each individual univariate polynomial in the set has also degree at most t. Formally:

Lemma 4. *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 $h_{\text{com}}(y) \stackrel{def}{=} h_0(y) + rh_1(y) + \cdots + r^l h_l(y)$ 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. □

Based on the above lemma, the core idea behind probabilistically checking the consistency of the row polynomials distributed by D is as follows: consider a set of at least $3t+1$ parties, say ReceivedSet, who claim to receive their respective row polynomials and their shares of the masking polynomials from D. To verify whether ReceivedSet has a subset of parties constituting a potential CORE, we proceed as follows. Let the points $\{\overline{f}_i(j) : P_i \in \text{ReceivedSet}\}$ define some polynomial $g^\star_j(y)$, for $j = 1, \ldots, n$. Similarly, let the shares $\{\overline{m}_{(P_j,k)}(i) : P_i \in \text{ReceivedSet}\}$ define some masking polynomial $m^\star_{(P_j,k)}(y)$, for $k = 1, \ldots, t+1$. Then we publicly verify if the polynomial $E(y) = m^\star(y) + r g^\star_1(y) + \cdots + r^n g^\star_n(y)$ is of degree at most t. Here $m^\star(y)$ is one of the masking polynomials (among the $(t+1)n$ masking polynomials $m^\star_{(P_j,k)}(y)$); and r is a random combiner, which is made public, after D's delivery of the row polynomials and shares of the masking polynomials to the (honest) parties in ReceivedSet (we will discuss in the sequel how such an r will be available and which masking polynomial among the $(t+1)n$ masking polynomials should be considered as $m^\star(y)$). We ask D to publish $E(y)$ and every $P_i \in$ ReceivedSet to publish the corresponding random linear combination $e_i = m^\star(i) + r\overline{f}_i(1) + \cdots + r^n \overline{f}_i(n)$. If $E(y)$ has degree at most t (which should be ideally the case) and if there are at least $3t+1$ parties in ReceivedSet, say AgreeSet, who "agrees" with D in the sense that $E(i) = e_i$ holds for every $P_i \in$ AgreeSet, then with high probability, AgreeSet constitutes a candidate for CORE. More specifically, let the points $\{\overline{f}_i(j) : P_i \in \text{AgreeSet}\}$ define some polynomial $\overline{g}_j(y)$, for $j = 1, \ldots, n$; then except with probability $2^{-\Omega(\kappa)} \approx \epsilon$, the polynomials $\overline{g}_j(y)$ are of degree at most t, for every $j = 1, \ldots, n$. This holds since (a possibly corrupted) D had no idea about the random r, when it distributed the row polynomials and the shares of the masking polynomials to the (honest) parties in AgreeSet. The secrecy of the row polynomials of the honest parties in ReceivedSet (for an honest D) will be preserved during the above check, thanks to the masking polynomial $m^\star(y)$.

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 setting. The main issues are when and how to generate the random combiner r, which masking polynomial to consider, who decides a ReceivedSet and how many such candidate ReceivedSet need to be examined to finally get a CORE. As explained above, we require an r that remains secret from D during its 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 to the parties in ReceivedSet. We solve this issue by asking a designated party $V \in \mathcal{P}$ to act as a *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 clearly 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 play the role of a verifier in parallel. So we first construct a sub-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 in parallel 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 the sub-protocol for a single 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. For this, we let the parties inform V when they receive their values from D and let V to construct the set ReceivedSet, based on the received responses, before V 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 lead to 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 (namely the $e(i)$ values). Even when D is honest, we can only guarantee that the honest parties in ReceivedSet (say exactly $2t + 1$) respond correctly by "agreeing" 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 may not 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 dynamically, after receiving confirmations from additional parties about their receipt of row polynomial and shares of the masking polynomials. After every expansion of ReceivedSet, V generates a new random challenge r and makes public the updated ReceivedSet and the newly generated challenge r. When a ReceivedSet and a random challenge is made public by V, 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 broadcast the corresponding 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, generates the challenge and then asks D and ReceivedSet to play the game. In the game, D wishes to convince everyone that the information that it handed to the parties in ReceivedSet are consistent (without violating the secrecy 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; as mentioned earlier, we denote by AgreeSet the set of such "agreeing" parties.

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 cannot cheat when V is honest, since V selects r only after getting the confirmation of the receipt of the row polynomials and the shares of the masking polynomials from the parties in ReceivedSet and more importantly, a random r is chosen for every instance of ReceivedSet. Our final observation is that there can be at most $t + 1$ different instances of ReceivedSet, since initially ReceivedSet may have $3t + 1$ parties and finally it may have all the $4t + 1$ parties. So V may need to generate a random challenge $t + 1$ times and so the checking game will be performed at most $t + 1$ times. Each time the game is played between D and the parties in a distinct instance of ReceivedSet, using the associated random challenge r, published along with the instance of ReceivedSet. This clearly implies that in order to maintain the secrecy of the row polynomials of the honest parties during the probabilistic checks, every time a distinct masking polynomial is to be used. Thus we may require $t + 1$ masking polynomials

Protocol Single-Verifier(V)

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 $\overline{m}_{(V,1)}(i), \ldots, \overline{m}_{(V,t+1)}(i)$ of the masking polynomials from D [a].
2. If $\overline{f}_i(x)$ is a polynomial of degree at most d, then send the (ECHO, i) message to V.

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

1. Wait to receive the (ECHO, i) message from $3t+1$ parties. Then include these $3t+1$ parties in the set $\mathsf{ReceivedSet}_{(V,1)}$. Select a random, non-zero value $r_{(V,1)}$ from $\mathbb{F}$ and broadcast $(r_{(V,1)}, \mathsf{ReceivedSet}_{(V,1)})$ [b].
2. After the previous step, for every new receipt of the (ECHO, i) message from a party $P_i \notin \mathsf{ReceivedSet}_{(V,\beta-1)}$, where $\beta \in \{2, \ldots, t+1\}$, 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 broadcast $(r_{(V,\beta)}, \mathsf{ReceivedSet}_{(V,\beta)})$ [c].

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

1. If $(r_{(V,\beta)}, \mathsf{ReceivedSet}_{(V,\beta)})$ with $\beta \in \{1, \ldots, t+1\}$ is received from the broadcast of V, then broadcast the linear combination $E_{(V,\beta)}(y)$ of the masking polynomial $m_{(V,\beta)}(y)$ and the column polynomials $g_1(y), \ldots, g_n(y)$, where [d]

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

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

1. If $(r_{(V,\beta)}, \mathsf{ReceivedSet}_{(V,\beta)})$ with $\beta \in \{1, \ldots, t+1\}$ is received from the broadcast of V, then do the following:
   (a) If $P_i \in \mathsf{ReceivedSet}_{(V,\beta)}$, then broadcast 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

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

   (b) If corresponding to the pair $(r_{(V,\beta)}, \mathsf{ReceivedSet}_{(V,\beta)})$, the polynomial $E_{(V,\beta)}(y)$ and the point $e_{(V,\beta,j)}$ is received from the broadcast of D and P_j respectively, then P_j is said to *agree with* D *with respect* to $(r_{(V,\beta)}, \mathsf{ReceivedSet}_{(V,\beta)})$ if the following holds:
      i. $E_{(V,\beta)}(y)$ has degree at most t;
      ii. $P_j \in \mathsf{ReceivedSet}_{(V,\beta)}$ and $e_{(V,\beta,j)} = E_{(V,\beta)}(j)$.
   (c) With respect to the pair $(r_{(V,\beta)}, \mathsf{ReceivedSet}_{(V,\beta)})$, when there are $3t+1$ parties P_j who agree with D, add such parties to the set $\mathsf{AgreeSet}_{(V,\beta)}$ [e].

Fig. 9. Verification of the row polynomials of the parties navigated by a verifier $V \in \mathcal{P}$.

on the behalf of a single V (and total $(t+1)n$ masking polynomials for n verifiers); the masking polynomials with index $(V, \star)$ are *associated* with V, for the game played under the control of V.[14] We now present the protocol Single-Verifier in Fig. 9 that captures the above discussion for a designated verifier $V \in \mathcal{P}$. We stress that there is no explicit terminating condition for Single-Verifier; and the parties do not terminate after finding an AgreeSet. The terminating condition will be specified in the protocol S-Ver-Agree, where several instances of Single-Verifier are executed and a CORE is selected based on several AgreeSets generated in those instances (more on this later).

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 5). This property 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 jth point on the row polynomials of the honest parties in $\mathsf{AgreeSet}_{(V,\beta)}$ indeed define polynomials of degree at most t (Lemma 6). This will further imply that the row polynomials

[14] For example, if $V = P_j$, then the masking polynomials $m_{(P_j,1)}(y), \ldots, m_{(P_j,t+1)}(y)$ are deployed.

of the honest parties in $\mathsf{AgreeSet}_{(V,\beta)}$ lie on a unique bivariate polynomial of degree-(d,t) (Lemma 7), implying that $\mathsf{AgreeSet}_{(V,\beta)}$ is a candidate for CORE. We then show that if D is honest, then the secret s remains information-theoretically secure during Single-Verifier, even if V is corrupted. This will ensure information-theoretic security for s in protocol S-Ver-Agree. Finally we show that protocol Single-Verifier involves a total broadcast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.

Lemma 5. *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 6. *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 jth 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 (ECHO, $\star$) message 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:

$$e_{(V,\beta,i)} = \overline{m}_{(V,\beta)}(i) + r_{(V,\beta)}\overline{g}_1(i) + \cdots + r^n_{(V,\beta)}\overline{g}_n(i).$$

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 $E_{\text{def}}(y) \stackrel{def}{=} \overline{m}_{(V,\beta)}(y) + r_{(V,\beta)}\overline{g}_1(y) + \cdots + r^n_{(V,\beta)}\overline{g}_n(y)$ 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 4.
2. We next claim that $E_{\min}(y) = E_{\text{def}}(y)$. 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 $E_{\text{def}}(y)$ (this condition has to be satisfied for P_i to be in $\mathsf{AgreeSet}_{(V,\beta)}$). Now consider the difference polynomial $dp(y) = E_{\text{def}}(y) - E_{\min}(y)$. 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 $E_{\text{def}}(y)$ is defined by the points on the row polynomials held by the parties in $H_{(V,\beta)}$ and so the maximum degree of $E_{\text{def}}(y)$ can be $|H^{(V,\beta)}| - 1$. These two facts together imply that $dp(y)$ is the zero polynomial, implying that $E_{\text{def}}(y) = E_{\min}(y)$ and so $E_{\min}(y)$ will have degree more than t.

□

Lemma 7. *In protocol* Single-Verifier, *if V is honest and some* $\mathsf{AgreeSet}_{(V,\beta)}$ *(containing at least $3t+1$ parties) is generated, then with probability at least $(1-\epsilon)$, there exists a unique bivariate 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 $\mathsf{AgreeSet}_{(V,\beta)}$ contain 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 6, 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 1, there is a unique bivariate 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 8. *If* D *is honest then s remains information-theoretically secure during the 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 the knowledge of $(t+1)(d+1)$ 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 $(t+1)(d+1) - t(d+1) = d+1$ distinct 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 $E_{(V,\beta)}(y) = m_{(V,\beta)}(y) + r_{(V,\beta)}g_1(y) + \cdots + r^n_{(V,\beta)}g_n(y)$, 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 $m_{(V,\beta)}(y)$ 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 $s = F(0, 0)$. □

Lemma 9. *Protocol* Single-Verifier *requires a total broadcast of* $\mathcal{O}(n^2 \log |\mathbb{F}|)$ *bits.*

Proof. In the protocol, the parties have to do the following communication at most $t+1$ times: broadcast of a random challenge by V; broadcast of the linear combination of its column polynomials and a masking polynomial by D; broadcast 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 broadcast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. □

Towards the Identification of a 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 (Lemmas 5–7). Since we do not know the identity of the honest parties, we cannot place our confidence on any particular party and ask it to play the role of the verifier. Thus we parallely execute 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 cannot pick any arbitrary $\mathsf{AgreeSet}_{(\star,\star)}$ as a 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 AgreeSet define the same bivariate 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 bivariate polynomial of degree-(d, t) (Lemma 11).

Lemma 10. *Let V be an honest verifier and let* $\mathsf{AgreeSet}_{(V,\gamma)}$ *and* $\mathsf{AgreeSet}_{(V,\delta)}$ *are generated during* Single-Verifier(V), *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 bivariate polynomial of degree-(d, t).*

Protocol S-Ver-Agree

VERIFICATION AND CORE CONSTRUCTION

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

1. Act as a verifier and invoke Single-Verifier(P_i).
2. For $j = 1, \ldots, n$, participate in Single-Verifier(P_j).
3. Add a verifier P_α to the set VerifierSet$_i$ if some AgreeSet$_{(P_\alpha,\beta)}$, where $\beta \in \{1, \ldots, t+1\}$, is generated during Single-Verifier(P_α), executed on behalf of the verifier P_α.
4. If $|\text{VerifierSet}_i| \geq t+1$, then do the following:
   (a) For every $P_\alpha \in$ VerifierSet$_i$, compute AgreeSet$_{P_\alpha} = \bigcup_\beta$ AgreeSet$_{(P_\alpha,\beta)}$ [a];
   (b) Compute CORE$_i = \{P_j \mid P_j$ belongs to AgreeSet$_{P_\alpha}$ of at least $t+1$ parties P_α in VerifierSet$_i\}$.
   (c) Wait for the new updates (such as the generation of new sets AgreeSet$_{(P_k,\star)}$, expansion of the existing sets AgreeSet$_{(P_k,\star)}$, 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. Let CORE$_\text{D}$ denote the instance of CORE$_i$ for D. Assign CORE = CORE$_\text{D}$, as soon as $|\text{CORE}_\text{D}| = 3t+1$ and broadcast CORE.

AGREEMENT ON CORE — Every party in $\mathcal{P}$ including D executes this code:

1. Wait to receive a CORE from the broadcast of D, such that $|\text{CORE}| = 3t+1$.
2. Wait until CORE $\subseteq$ CORE$_i$. Then output CORE and terminate.

[a] This denotes taking the union of all AgreeSet$_{(P_\alpha,\star)}$, generated during Single-Verifier(P_α).

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

Proof. By Lemma 7, if V is honest, then the row polynomials held by the honest parties in AgreeSet$_{(V,\gamma)}$, as well as in AgreeSet$_{(V,\delta)}$, define unique bivariate 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 AgreeSet$_{(V,\gamma)}$ and AgreeSet$_{(V,\delta)}$, whose row polynomials (which are of degree at most d) define a unique bivariate polynomial of degree-(d,t). □

Lemma 11. *Let V_α and V_δ be two different honest verifiers. Then the row polynomials of the honest parties in any* AgreeSet$_{(V_\alpha,\star)}$ *and* AgreeSet$_{(V_\delta,\star)}$, *generated during* Single-Verifier(V_α) *and* Single-Verifier(V_δ), *respectively, define the same bivariate 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 AgreeSet$_{(V_\alpha,\star)}$ and AgreeSet$_{(V_\delta,\star)}$, whose row polynomials define a single bivariate polynomial of degree-(d,t). □

Using Lemmas 10 and 11, we suggest to check the presence of a 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 a 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 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 bivariate polynomial of degree-(d,t). This intuition is captured in the protocol S-Ver-Agree, presented in Fig. 10.

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

Lemma 12. *Protocol* S-Ver-Agree *achieves the following for every possible* $\mathcal{A}_t$:

1. *The protocol incurs a total broadcast of* $\mathcal{O}(n^3 \log |\mathbb{F}|)$ *bits.*
2. *If* D *is honest, then* s *remains information-theoretically secure.*
3. *If* D *is honest then eventually every honest party outputs a* CORE, *such that the row polynomials of the honest parties in* CORE *lie on the bivariate polynomial* $F(x, y)$.
4. *If* D *is corrupted and some honest party outputs a* CORE, *then every other honest party eventually does the same. Moreover, except with probability* ϵ, *the row polynomials of the honest parties in* CORE *lie on a unique bivariate polynomial of degree-*(d, t).

Proof. From Lemma 9, a single instance of Single-Verifier incurs a total broadcast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. In protocol S-Ver-Agree, n instances of Single-Verifier are executed which will incur a total broadcast of $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits.

The information-theoretic security for s follows from Lemma 8 and the fact that in each instance of Single-Verifier, independent masking polynomials are used.

If D is honest, then the set of $3t + 1$ honest parties will be eventually present in every $\mathsf{AgreeSet}_{P_\alpha}$, corresponding to every verifier P_α. Moreover, every honest verifier will be eventually included in the set $\mathsf{VerifierSet}_i$ of every honest P_i. If D is honest, then D will eventually construct a CORE_D of size $3t + 1$ and broadcasts the same and by the property of the broadcast, the set will be received by every honest party. Moreover, every honest P_i will find that $\mathsf{CORE}_\mathsf{D} \subseteq \mathsf{CORE}_i$ and will output 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 output a CORE, then it implies that P_i has received CORE from the broadcast of D. Moreover, P_i must have found the condition $\mathsf{CORE} \subseteq \mathsf{CORE}_i$ to hold. From the properties of the broadcast, every other honest party P_j will also eventually receive the same CORE from the broadcast 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 output CORE. We now show that except with probability ϵ, the row polynomials of the honest parties in CORE lie on a unique bivariate 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 bivariate 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 sets $\mathsf{AgreeSet}_{P_\alpha}$, corresponding to the honest parties P_α, 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 $\overline{F}(x, y)$. □

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

3.1.3. *Statistical AVSS Scheme for a Single Secret*

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

Protocol S-Sh

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

Fig. 11. Protocol for the sharing phase of the statistical AVSS scheme.

Theorem 5. *Protocols* (S-Sh, Rec) *constitute a statistical AVSS scheme that generates d-sharing of s. Protocol* S-Sh *incurs a communication of $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits and a total broadcast of $\mathcal{O}(n^3 \log |\mathbb{F}|)$ bits. Protocol* Rec *has communication complexity $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits.*

Proof. If D is honest, then every honest party will eventually terminate S-Sh with its share of the secret s. This follows from Lemmas 12(3) and 2(1). From Lemma 12(4), if D is corrupted and some honest party has output a CORE, then every other honest party will output the same CORE. Moreover, except with probability ϵ, the row polynomials of the honest parties in CORE will lie on a unique bivariate polynomial of degree-(d, t). So from Lemma 2(1), executing the protocol Gen generates d-sharing. If the honest parties execute Rec, then s will be reconstructed correctly. This follows from Lemma 3. 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 Lemmas 12(2) and 8, 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 it 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, Lemmas 12(1), 2(2) and 3. □

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 does not share any masking polynomial. Moreover, it distributes both row and 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.

Protocol P-Distr

CODE FOR D — Only D executes this code:

1. On having a secret s, 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 and terminate.

Fig. 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.

Claim 2. *Protocol* P-Distr *incurs a communication of* $\mathcal{O}((nd + n^2)\log|\mathbb{F}|) = \mathcal{O}(n^2 \log|\mathbb{F}|)$ *bits by* D.

3.2.2. *Protocol* P-Ver-Agree

The goal of the protocol P-Ver-Agree is to enable the (honest) parties identify the presence of a CORE in an error-free fashion. For this, 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 messages if the consistency check passes. Using these messages, we construct a consistency graph with the edges representing pairwise consistency and check for the presence of an (n, t)-star (see Sect. 2.3). The intuition is that if D is honest, then eventually every honest party will receive its row and column polynomial, which will be pairwise 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 $(\mathbf{C}, \mathbf{D})$ be such a star. Our first observation is that the row polynomials of the honest parties in $\mathbf{C}$ and the column polynomials of the honest parties in $\mathbf{D}$ will be pairwise consistent and thus they lie on a unique bivariate polynomial, say $\overline{F}(x, y)$, of degree-(d, t). This is due to Lemma 1 and the fact that there will be at least $t + 1$ and $2t + 1$ honest parties in $\mathbf{C}$ and $\mathbf{D}$, respectively. Moreover, if D is honest then $\overline{F}(x, y) = F(x, y)$.

The next obvious question is: does the presence of $(\mathbf{C}, \mathbf{D})$ implies the existence of a CORE? Recall that we want CORE to be of size $3t + 1$. Clearly $\mathbf{C}$ is not qualified to be a CORE. On the other hand, even though $\mathbf{D}$ is of size $3t + 1$, it cannot be considered as a CORE. This is because we want the *row* polynomials of the honest parties in CORE to lie on a unique bivariate polynomial; whereas an (n, t)-star ensures that the *column* polynomials of the honest parties in $\mathbf{D}$ lie on a unique bivariate polynomial. If we consider $\mathbf{D}$ as a CORE, then we cannot "complete" the d-sharing by executing the protocol Gen on $\mathbf{D}$. So we cannot directly confirm the presence of a CORE from the presence of an (n, t)-star. However, we observe that if indeed D is honest then there will be "additional" honest parties, apart from the honest parties in $\mathbf{C}$, whose row polynomials 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 $\mathbf{D}$ by identifying additional parties not in $\mathbf{D}$ whose column polynomial also lie on $\overline{F}(x, y)$. The expanded set, denoted by $\mathbf{F}$, includes

all the parties having edges with at least $2t + 1$ parties from $\mathbf{C}$ in the consistency graph. The parties in $\mathbf{D}$ will be automatically included in $\mathbf{F}$. It is easy to note that the column polynomial of an honest $P_j \in \mathcal{P} \setminus \mathbf{D}$ 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 an edge with at least $2t + 1$ parties from $\mathbf{C}$, this implies that its column polynomial is pairwise consistent with the row polynomial of at least $t + 1$ honest parties from $\mathbf{C}$, which lie on $\overline{F}(x, y)$.

- We then try to "expand" the set $\mathbf{C}$. Specifically, we search for the parties P_j, who have an edge with at least $d + t + 1$ parties from $\mathbf{F}$ 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 from $\mathbf{F}$ (with whom P_j has an edge), at least $d + 1$ will be honest. Thus, the row polynomial of P_j will be pairwise consistent with the column polynomials of at least $d + 1$ honest parties from $\mathbf{F}$, 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 parties P_j in a set $\mathbf{E}$. Notice that all the parties in $\mathbf{C}$ will be included in $\mathbf{E}$.

If we find $\mathbf{E}$ to be of size $3t + 1$, then $\mathbf{E}$ is taken as a CORE. It is easy to see that indeed the row polynomials of all the honest parties in $\mathbf{E}$ 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 $\mathbf{D}$ and $\mathbf{C}$ sets until $\mathbf{E}$ admits a size of $3t + 1$. Consider the case when $d = 2t$ and $\mathbf{C}$ and $\mathbf{D}$ are exactly of size $2t + 1$ and $3t + 1$, respectively, such that they contain t corrupted parties. If the corrupted parties in $\mathbf{C}$ choose to be inconsistent with the parties outside $\mathbf{D}$, then the honest parties outside $\mathbf{D}$ will have edges with only $t + 1$ parties from $\mathbf{C}$ and will not be included in the set $\mathbf{F}$. So $\mathbf{F}$ will remain the same as $\mathbf{D}$ and will not include any additional party. Similarly, if the corrupted parties in $\mathbf{F}$ choose to be inconsistent with the parties outside $\mathbf{C}$, then the honest parties outside $\mathbf{C}$ will have edges with only $2t + 1$ parties from $\mathbf{F}$ and will be never included in the set $\mathbf{E}$. So $\mathbf{C}$ 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 the $\mathbf{C}$ component of the obtained (n, t)-star will have at least $2t + 1$ honest parties. When $\mathbf{C}$ contains at least $2t + 1$ honest parties, then eventually the set $\mathbf{D}$ will expand to the set $\mathbf{F}$, which will contain all the $3t + 1$ honest parties and eventually the set $\mathbf{C}$ will expand to the set $\mathbf{E}$ 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 the expansion of $\mathbf{C}$ and $\mathbf{D}$ for every instance of the consistency graph. In a more detail, after every update in the consistency graph (on receiving new OK messages), we check for the presence of a new (n, t)-star in the graph (which was not found earlier) along with the corresponding $\mathbf{F}$ and $\mathbf{E}$ sets and update the existing $\mathbf{F}$ and $\mathbf{E}$ sets (corresponding to all the previously generated (n, t)-stars). This is continued till we find an instance of $\mathbf{E}$ of size $3t + 1$. Such an $\mathbf{E}$ will be considered as a CORE. Surely

Protocol P-Ver-Agree

i. CODE FOR P_i — Every party in $\mathcal{P}$ (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 broadcast the message $\text{OK}(P_i, P_j)$.
3. Construct the undirected consistency graph G_i with $\mathcal{P}$ as the vertex set. Add an edge (P_j, P_k) to G_i upon receiving the $\text{OK}(P_k, P_j)$ message and the $\text{OK}(P_j, P_k)$ message from the broadcast 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 $\text{OK}(\star, \star)$ message, update G_{D}. If a new edge is added to G_{D}, then execute Find-Star($\overline{G}_{\mathsf{D}}$). Assume there are $\alpha \geq 0$ distinct (n,t)-stars that are found in the past, from different executions of Find-Star($\overline{G}_{\mathsf{D}}$).
   (a) If an (n,t)-star is found from the current execution of Find-Star($\overline{G}_{\mathsf{D}}$) that is distinct from all the previous α (n,t)-stars (obtained earlier), then do the following:
      i. Call the new (n,t)-star as $(\mathbf{C}_{\alpha+1}, \mathbf{D}_{\alpha+1})$.
      ii. Construct a set $\mathbf{F}_{\alpha+1}$ as follows: Add P_j to $\mathbf{F}_{\alpha+1}$ if P_j has an edge with at least $2t+1$ parties from $\mathbf{C}_{\alpha+1}$ in G_{D}.
      iii. Construct a set $\mathbf{E}_{\alpha+1}$ as follows: Add P_j to $\mathbf{E}_{\alpha+1}$ if P_j has an edge with at least $d+t+1$ parties from $\mathbf{F}_{\alpha+1}$ in G_{D}.
      iv. For $\beta = 1, \ldots, \alpha$, update the existing $\mathbf{F}_\beta$ and $\mathbf{E}_\beta$ as follows:
         A. Add P_j to $\mathbf{F}_\beta$, if $P_j \notin \mathbf{F}_\beta$ and P_j has an edge with at least $2t+1$ parties from $\mathbf{C}_\beta$ in G_{D}.
         B. Add P_j to $\mathbf{E}_\beta$, if $P_j \notin \mathbf{E}_\beta$ and P_j has an edge with at least $d+t+1$ parties from $\mathbf{F}_\beta$ 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 sets $\mathbf{F}_\beta$ and $\mathbf{E}_\beta$.

   Let $(\mathbf{E}_\gamma, \mathbf{F}_\gamma)$ be the first pair among the generated pairs $(\mathbf{E}_\beta, \mathbf{F}_\beta)$ such that $|\mathbf{E}_\gamma| \geq 3t+1$ and $|\mathbf{F}_\gamma| \geq 3t+1$. Assign CORE $= \mathbf{E}_\gamma$ and broadcast $((\mathbf{C}_\gamma, \mathbf{D}_\gamma), (\mathbf{E}_\gamma, \mathbf{F}_\gamma))$.

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

1. Wait to receive $((\mathbf{C}_\gamma, \mathbf{D}_\gamma), (\mathbf{E}_\gamma, \mathbf{F}_\gamma))$ from the broadcast of D, such that $|\mathbf{E}_\gamma| \geq 3t+1$ and $|\mathbf{F}_\gamma| \geq 3t+1$.
2. Wait until $(\mathbf{C}_\gamma, \mathbf{D}_\gamma)$ becomes an (n,t)-star in the consistency graph G_i. For this, wait to receive the corresponding OK messages from the broadcast of the parties in $\mathbf{C}_\gamma$ and $\mathbf{D}_\gamma$.
3. Wait until every party $P_j \in \mathbf{F}_\gamma$ has an edge with at least $2t+1$ parties from $\mathbf{C}_\gamma$ in G_i.
4. Wait until every party $P_j \in \mathbf{E}_\gamma$ has an edge with at least $d+t+1$ parties from $\mathbf{F}_\gamma$ in G_i.

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

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

if D is honest, then we will get an $\mathbf{E}$ with the desired size. We let D moderate these repetitions by asking it to repeat the star-finding process and the expansion of $\mathbf{C}$ and $\mathbf{D}$ for every instance of the consistency graph. Upon finding a CORE, D makes all the parties agree on CORE by broadcasting the star and the corresponding $\mathbf{E}$ and $\mathbf{F}$ 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 a 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 $(\mathbf{C}_\beta, \mathbf{D}_\beta)$ denotes the βth instance of an (n,t)-star and the pair

$(\mathbf{E}_\beta, \mathbf{F}_\beta)$ denotes the corresponding $\mathbf{E}$ and $\mathbf{F}$ sets. After every update in the consistency graph, $(\mathbf{E}_\beta, \mathbf{F}_\beta)$ may be updated. In the sequel, we will show that there will be a finite number of instances of (n,t)-star (and the corresponding $\mathbf{E}$ and $\mathbf{F}$ sets) that can result from the consistency graph. We will also prove the following three key observations on which the protocol P-Ver-Agree is based upon:

1. If D is honest, then eventually some (n,t)-star $(\mathbf{C}_\beta, \mathbf{D}_\beta)$ will be generated, where $\mathbf{C}_\beta$ 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 $\mathbf{C}$ component of an (n,t)-star $(\mathbf{C}_\beta, \mathbf{D}_\beta)$ contains at least $2t+1$ honest parties, then a CORE will be eventually generated from $(\mathbf{C}_\beta, \mathbf{D}_\beta)$ (Lemma 15).
3. For any (n,t)-star $(\mathbf{C}_\beta, \mathbf{D}_\beta)$, the row polynomials of the honest parties in $\mathbf{C}_\beta$ define a unique bivariate polynomial of degree-(d,t), irrespective of D (Lemma 13). Moreover, if a CORE is generated from this $(\mathbf{C}_\beta, \mathbf{D}_\beta)$, then the row polynomials of the honest parties in CORE define the same bivariate polynomial (Lemma 17).

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

Lemma 13. *Let* $(\mathbf{C}, \mathbf{D})$ *be any* (n,t)-star *in the consistency graph* G_k *of an honest party* P_k. *Then the row polynomials held by the honest parties in* $\mathbf{C}$ *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* $\overline{F}(j,y) = \overline{g}_j(y)$ *holds for every honest* P_i *and* P_j *in* $\mathbf{C}$ *and* $\mathbf{D}$, *respectively. Moreover, if* D *is honest then* $\overline{F}(x,y) = F(x,y)$.

Proof. For any (n,t)-star $(\mathbf{C}, \mathbf{D})$, we know that $|\mathbf{C}| \geq n-2t$ and $|\mathbf{D}| \geq n-t$. So $\mathbf{C}$ and $\mathbf{D}$ contains at least $n-3t \geq t+1$ and $n-2t \geq 2t+1$ honest parties, respectively. Moreover, every honest party P_i in $\mathbf{C}$ will be pairwise consistent with every honest party in $\mathbf{D}$. That is, $\overline{f}_i(j) = \overline{g}_j(i)$ and $\overline{f}_j(i) = \overline{g}_i(j)$ will hold for every honest $P_i \in \mathbf{C}$ and every honest $P_j \in \mathbf{D}$. 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 1. 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 outputs a CORE and terminates the protocol P-Ver-Agree.

Lemma 14. *If* D *is honest then eventually an* (n,t)-star $(\mathbf{C}_\beta, \mathbf{D}_\beta)$ *will be generated by* D, *such that* $\mathbf{C}_\beta$ *contains 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 $\overline{G}_\mathsf{D}$. So each edge in $\overline{G}_\mathsf{D}$ 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 $(\mathbf{C}_\beta, \mathbf{D}_\beta)$ be the (n,t)-star which is obtained while applying the Find-Star algorithm on $\overline{G}_\mathsf{D}$, when $\overline{G}_\mathsf{D}$ contains edges of only the above two types. Now, by the construction of $\mathbf{C}_\beta$ (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 $\overline{G}_{\mathsf{D}}$ 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 $\overline{G}_{\mathsf{D}}$. This clearly implies that both P_j and P_k are surely corrupted. So for every honest P_i outside $\mathbf{C}_\beta$, at least one (if P_i belongs to N, then one; if P_i belongs to T, then two) corrupted party also remains outside $\mathbf{C}_\beta$. As there are at most t corrupted parties, $\mathbf{C}_\beta$ may exclude at most t honest parties. So $\mathbf{C}_\beta$ is bound to contain at least $2t + 1$ honest parties.

To complete the proof, we now have to show that $\overline{G}_{\mathsf{D}}$ will contain the edges of the above two types after a 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 $\overline{G}_{\mathsf{D}}$ 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 $(\mathbf{C}_\beta, \mathbf{D}_\beta)$ with $\mathbf{C}_\beta$ 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 a* CORE *will be generated by* D *and every honest party will output* 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 $(\mathbf{C}_\beta, \mathbf{D}_\beta)$ in the consistency graph G_{D}, such that $\mathbf{C}_\beta$ 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 $\mathbf{C}_\beta$. This implies that every honest party in $\mathcal{P}$ will eventually have at least $2t + 1$ neighbors in $\mathbf{C}_\beta$ and so they will be included in $\mathbf{F}_\beta$. Following a similar argument, every honest party in $\mathcal{P}$ will eventually have at least $d + t + 1$ neighbors in $\mathbf{F}_\beta$ and so they will be included in $\mathbf{E}_\beta$. So D will find that $|\mathbf{E}_\beta| \geq 3t + 1$ and $|\mathbf{F}_\beta| \geq 3t + 1$ and will assign $\mathbf{E}_\beta$ as CORE and broadcast $((\mathbf{C}_\beta, \mathbf{D}_\beta), (\mathbf{E}_\beta, \mathbf{F}_\beta))$. By the property of the broadcast, every honest party P_i will receive these sets correctly from D and will eventually find that $(\mathbf{C}_\beta, \mathbf{D}_\beta)$ 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 $\mathbf{C}_\beta$ and $\mathbf{D}_\beta$ in its consistency graph G_{D}, then the same edges will also be eventually included by every honest P_i in its consistency graph G_i. Due to the same reason, an honest P_i will find that every party in $\mathbf{F}_\beta$ has at least $2t + 1$ neighbors in $\mathbf{C}_\beta$ in the graph G_i and similarly, every party in $\mathbf{E}_\beta$ has at least $d + t + 1$ neighbors in $\mathbf{F}_\beta$ in the graph G_i eventually. So P_i will output CORE $= \mathbf{E}_\beta$ and terminate the protocol P-Ver-Agree. □

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 terminates the protocol P-Ver-Agree then every honest party also eventually does the same.

Lemma 16. *If* D *is corrupted and some honest party* P_i *terminates the protocol* P-Ver-Agree *outputting a* CORE, *then every other honest party* P_j *eventually does the same.*

Proof. If an honest P_i has output a CORE, then this implies that it has received $((\mathbf{C}_\gamma, \mathbf{D}_\gamma), (\mathbf{E}_\gamma, \mathbf{F}_\gamma))$ from the broadcast of D, with $\mathsf{CORE} = \mathbf{E}_\gamma$, and verified the following in its consistency graph G_i: (a) $(\mathbf{C}_\gamma, \mathbf{D}_\gamma)$ is an (n,t)-star; (b) every party in $\mathbf{F}_\gamma$ has at least $2t+1$ neighbors in $\mathbf{C}_\gamma$ and (c) every party in $\mathbf{E}_\gamma$ has at least $d+t+1$ neighbors in $\mathbf{F}_\gamma$. From the properties of the broadcast, every other honest party P_j will also receive the same $((\mathbf{C}_\gamma, \mathbf{D}_\gamma), (\mathbf{E}_\gamma, \mathbf{F}_\gamma))$ from the broadcast of D. Moreover, P_j will also find that eventually the above three conditions are also satisfied in its consistency graph G_j. So P_j will also eventually output CORE and terminate the protocol P-Ver-Agree. □

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

Lemma 17. *If an honest P_i has output a* 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 output a CORE, then it implies that it has received $((\mathbf{C}_\gamma, \mathbf{D}_\gamma), (\mathbf{E}_\gamma, \mathbf{F}_\gamma))$ from the broadcast of D and checked their validity with respect to its own consistency graph G_i. This means that $(\mathbf{C}_\gamma, \mathbf{D}_\gamma)$ is an (n,t)-star in G_i. Lemma 13 implies that the row polynomials of the honest parties in $\mathbf{C}_\gamma$ define a unique bivariate polynomial, say $\overline{F}(x,y)$, of degree-(d,t). Recall that $\mathbf{E}_\gamma$ is obtained by expanding the set $\mathbf{C}_\gamma$. To complete the proof we need to show that even the row polynomials of the honest parties in $\mathbf{E}_\gamma \setminus \mathbf{C}_\gamma$ 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 $\mathbf{F}_\gamma$ lies on $\overline{F}(x,y)$. This is because by the construction of $\mathbf{F}_\gamma$, every honest $P_j \in \mathbf{F}_\gamma$ has at least $2t+1$ neighbors in $\mathbf{C}_\gamma$, which implies that $\overline{f}_{kj} = \overline{f}_k(j) = \overline{g}_j(k)$ for at least $2t+1$ parties P_k in the set $\mathbf{C}_\gamma$. Moreover, the degree of $\overline{g}_j(y)$ is at most t. Now out of these $2t+1$ parties P_k, at least $t+1$ are honest. Also the row polynomials of those P_k 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 $P_j \in \mathbf{E}_\gamma$ also lies on $\overline{F}(x,y)$. By the construction of $\mathbf{E}_\gamma$, every such P_j has at least $d+t+1$ neighbors in $\mathbf{F}_\gamma$, which means that $\overline{f}_j(k) = \overline{g}_{kj} = \overline{g}_k(j)$ for at least $d+t+1$ parties P_k in $\mathbf{F}_\gamma$. Moreover, the degree of $\overline{f}_j(x)$ is at most d. Now out of the $d+t+1$ parties P_k in $\mathbf{F}_\gamma$ (with whom P_j has an edge), at least $d+1$ are honest. Also the column polynomials of such P_k 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 $\overline{F}(x,y)$. □

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

Lemma 18. *In protocol* P-Ver-Agree *if* D *is honest then s remains 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

Protocol P-Sh

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

[a] We note that in P-Sh the parties in CORE need not have to communicate the values on their column polynomials during the protocol Gen (unlike in S-Sh). This is because the parties already exchange the common values on their row and column polynomials during the protocol P-Ver-Agree. So once a 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 a CORE is identified.

Fig. 14. Protocol for the sharing phase of the perfect AVSS scheme.

$g_1(y), \ldots, g_t(y)$. 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 $f_1(x), \ldots, f_t(x), g_1(y), \ldots, g_t(y)$. The adversary is completely oblivious of the communication done between the honest parties. So it 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, it 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 *incurs a communication of* $\mathcal{O}(n^2 \log |\mathbb{F}|)$ *bits and a total broadcast of* $\mathcal{O}(n^2 \log |\mathbb{F}|)$ *bits.*

Proof. In the protocol, the parties exchange the common points on their row and column polynomials, which requires a communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. In addition, the parties also broadcast the $\mathtt{OK}(\star, \star)$ messages, which requires a broadcast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. Furthermore, broadcasting of $((\mathbf{C}_\gamma, \mathbf{D}_\gamma), (\mathbf{E}_\gamma, \mathbf{F}_\gamma))$ requires a broadcast 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 P-Sh for the perfect scheme PAVSS is presented in Fig. 14.

Theorem 6. *Protocols* (P-Sh, Rec) *constitute a perfect AVSS scheme, which generates* d*-sharing of* s*. Protocol* P-Sh *requires a communication of* $\mathcal{O}(n^2 \log |\mathbb{F}|)$ *bits and a total broadcast* $\mathcal{O}(n^2 \log |\mathbb{F}|)$ *bits. Protocol* Rec *has communication complexity* $\mathcal{O}(n^2 \log |\mathbb{F}|)$ *bits.*

Proof. If D is honest then every honest party eventually terminates the protocol P-Sh. This follows from Lemmas 15 and 2(1). If D is corrupted and some honest party terminates P-Sh, then every other honest party will also eventually terminate the protocol P-Sh. This follows from Lemmas 16 and 2(1). Moreover, if the honest parties invoke the protocol Rec, then every honest party will eventually terminate Rec. This follows from Lemma 3. This completes the proof of termination.

If D is honest then at the end of P-Sh, every honest party will have its share of s. This follows from Lemmas 17 and 2(1). Moreover, every honest party on terminating the protocol Rec will output s. This follows from Lemma 3. On the other hand, even if D is corrupted and some honest party terminates P-Sh, then it implies that a 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 2(1)) ensures that every honest party will have the share of $\overline{s}$. Moreover, the honest parties on terminating Rec will output $\overline{s}$. This proves the correctness property.

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

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 concurrently, such that the point-to-point communication depends on ℓ, but the broadcast communication is independent of ℓ. Since the broadcast is an expensive protocol,[15] we save a lot of communication in our AMPC protocols by using our new AVSS schemes for sharing multiple secrets concurrently.

The main idea behind making the broadcast communication independent of ℓ is the following: we observe that in the sub-protocols dealing with a single secret, the steps which involve point-to-point communication among the parties can be extended in a "natural" way to deal with ℓ values. For example, instead of taking a single bivariate 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 broadcast 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

[15] Recall that the best known broadcast protocol due to [13] incurs a communication of $\mathcal{O}(n^2)$ bits to broadcast a single bit.

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 S-MS-Sh 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 S-MS-Sh consists of a sequence of three phases (similar to the protocol S-Sh), 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 bivariate polynomial $F_l(x, y)$ of degree-(d, t) with the constant term as s_l and distributes the ith 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. 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 a 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 $\{\overline{f}_{l,i}(x) : P_i \in \text{CORE}$ and P_i is honest$\}$ define a unique bivariate 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, party 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

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

   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 its ℓ row polynomial. Specifically, P_i broadcasts $e_{(V,\beta,i)}$, where

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

   The rest of the steps for the protocol Single-MS-Verifier are the 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 jth points $\overline{f}_{1,i}(j), \ldots, \overline{f}_{\ell,i}(j)$ on its 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 its 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 7. *Protocols* (S-MS-Sh, Rec-MS) *constitute a statistical AVSS scheme* SAVSS-MS, *which generates d-sharing of $S = (s_1, \ldots, s_\ell)$. In* S-MS-Sh, *the parties communicate $\mathcal{O}((\ell nd + n^3)\log|\mathbb{F}|) = \mathcal{O}((\ell n^2 + n^3)\log|\mathbb{F}|)$ bits and broadcast $\mathcal{O}(n^3 \log|\mathbb{F}|)$ bits. Protocol* Rec-MS *incurs a communication of $\mathcal{O}(\ell n^2 \log|\mathbb{F}|)$ 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 P-MS-Sh for the sharing phase and protocol Rec-MS (discussed in the previous section) for the reconstruction phase. Now the sharing protocol P-MS-Sh consists of a sequence of three phases (similar to the protocol P-Sh), 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 bivariate polynomial $F_l(x, y)$ of degree-(d, t) with s_l as the constant term and distributes the ith row polynomial $f_{l,i}(x) = F_l(x, i)$ and the ith 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 the 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 pairwise 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 its row polynomials and another ℓ values $\overline{g}_{l,i,j} = \overline{g}_{l,i}(j)$, for $l = 1, \ldots, \ell$ on its 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 if $\overline{f}_{l,i}(j) \stackrel{?}{=} \overline{g}_{l,j,i}$ 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 broadcasts the message $\texttt{OK}(P_i, P_j)$. The rest of the steps for P-MS-Ver-Agree will be now the same as in the protocol P-Ver-Agree.

Protocol S-$(t, 2t)$-Sh/P-$(t, 2t)$-Sh

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-$(t, 2t)$-Sh, invoke two instances of S-MS-Sh to t-share and $(2t-1)$-share S and R respectively. On the other hand, in P-$(t, 2t)$-Sh, invoke two instances of P-MS-Sh to t-share and $(2t-1)$-share S and R respectively.

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

1. In S-$(t, 2t)$-Sh, participate in the two instances of S-MS-Sh and wait to terminate these two instances. On the other hand, in P-$(t, 2t)$-Sh, participate in the two instances of P-MS-Sh 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 at the end of the two instances of S-MS-Sh/P-MS-Sh.
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.

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

3. *Protocol* Gen-MS: This protocol is the same as discussed in the previous section. The footnote mentioned in the protocol P-Sh (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 8. *Protocols* (P-MS-Sh, Rec-MS) *constitute a perfect AVSS scheme* PAVSS-MS, *which generates d-sharing of $S = (s_1, \ldots, s_\ell)$. In* P-MS-Sh, *the parties communicate $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits and broadcast $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. During* Rec-MS, *the parties communicate $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits.*

5. Protocols 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 performs the t-sharing of S. In addition, it also does the $(2t-1)$-sharing of ℓ random values, say $R = (r_1, \ldots, r_\ell)$. 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 protocol of our AVSS schemes (dealing with ℓ values). Now depending upon whether it invokes the protocol S-MS-Sh or P-MS-Sh, the resulting protocol will either have a negligible error or no error in the correctness. We call the resulting protocols as S-$(t, 2t)$-Sh and P-$(t, 2t)$-Sh, respectively. We present the protocol in Fig. 15.

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

Theorem 9. *Let* $\mathsf{D} \in \mathcal{P}$ *have the input* $S = (s_1, \ldots, s_\ell)$. *Then protocol* S-$(t, 2t)$-Sh *achieves the following properties for every possible* $\mathcal{A}_t$:

1. ***Termination:*** (a) *If* D *is honest, then all the honest parties eventually terminate* S-$(t, 2t)$-Sh. (b) *If* D *is corrupted and some honest party terminates* S-$(t, 2t)$-Sh, *then all the honest parties eventually terminate the protocol.*
2. ***Correctness:*** *If some honest party terminates the protocol, then there exist* ℓ *values, say* $\overline{S} = (\overline{s}_1, \ldots, \overline{s}_\ell)$, *which will be eventually* $(t, 2t)$*-shared among the parties in* $\mathcal{P}$, *except with probability at most* ϵ. *Moreover, if* D *is honest, then* $\overline{S} = S$.
3. ***Secrecy:*** *If* D *is honest then* S *remains information-theoretically secure during the protocol.*
4. ***Communication Complexity***: *The protocol requires a communication of* $\mathcal{O}((\ell n^2 + n^3) \log |\mathbb{F}|)$ *bits and a total broadcast of* $\mathcal{O}(n^3 \log |\mathbb{F}|)$ *bits.*

Proof. The proof follows from the properties of the protocol S-MS-Sh (Theorem 7). □

The proof of the following theorem follows from the properties of P-MS-Sh.

Theorem 10. *Let* $\mathsf{D} \in \mathcal{P}$ *have the input* $S = (s_1, \ldots, s_\ell)$. *Then protocol* P-$(t, 2t)$-Sh *achieves the following properties for every possible* $\mathcal{A}_t$:

1. ***Termination:*** (*a*) *If* D *is honest, then all the honest parties eventually terminate the protocol.* (*b*) *If* D *is corrupted and some honest party terminates* P-$(t, 2t)$-Sh, *then all the honest parties eventually terminate* P-$(t, 2t)$-Sh.
2. ***Correctness:*** *If some honest party terminates the protocol, then there exist* ℓ *values, say* $\overline{S} = (\overline{s}_1, \ldots, \overline{s}_\ell)$, *which will be eventually* $(t, 2t)$*-shared among the parties in* $\mathcal{P}$. *Moreover, if* D *is honest, then* $\overline{S} = S$.
3. ***Secrecy:*** *If* D *is honest then* S *remains information-theoretically secure during the protocol.*
4. ***Communication Complexity***: *Protocol* P-$(t, 2t)$-Sh *incurs a communication of* $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ *bits and a total broadcast of* $\mathcal{O}(n^2 \log |\mathbb{F}|)$ *bits.*

6. AMPC Protocols with $n = 4t + 1$

Once we have an efficient protocol for generating $(t, 2t)$-sharing, we can design an AMPC protocol following the approach of Ref. [5]. Structurally, both our statistical and perfect AMPC protocols are divided into a sequence of three phases. Depending upon the type of sub-protocols (with a 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 the publicly known function over $\mathbb{F}$ (which the parties want to compute), 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 zero. 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 $c_I = n$ and $c_O = n$. 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-shares its input and then the parties agree on a common subset of at least $n - t$ parties, whose inputs will be eventually t-shared among the parties.
3. *Computation Phase*: Here, based on the inputs of the parties in the common subset (agreed in the previous phase), the circuit is 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 subset C of at least $n - t$ parties, who correctly $(t, 2t)$-shared $\frac{c_M + c_R}{n-2t}$ values. The parties then apply the randomness-extraction function Ext (see Sect. 2.3) on the values shared by the parties in C to output $c_M + c_R$ $(t, 2t)$-shared values; since the values shared by the honest parties in C are indeed random and information-theoretically secure, the output $(t, 2t)$-shared values will be indeed random and information-theoretically secure. In Fig. 16, we present the protocol for this phase. Now depending upon whether the parties invoke the protocol S-$(t, 2t)$-Sh (having a negligible error) or P-$(t, 2t)$-Sh (having no error) for sharing the values, 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-$(t, 2t)$-Sh and P-$(t, 2t)$-Sh, respectively, along with the properties of Ext.

Lemma 20. *Protocol* S-Preparation *satisfies the following properties for every possible* $\mathcal{A}_t$:

1. ***Termination***: *All the honest parties eventually terminate the protocol.*
2. ***Correctness***: *Except with probability at most* ϵ, *the parties output* $(t, 2t)$*-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 shared values* $r_{i,j}$ *remain information-theoretically secure.*
4. ***Communication Complexity***: *The protocol incurs a communication of* $\mathcal{O}(((c_M + c_R)n^2 + n^4) \log |\mathbb{F}|)$ *bits, a total broadcast of* $\mathcal{O}(n^4 \log |\mathbb{F}|)$ *bits and requires one invocation of* ACS.

Protocol S-Preparation/P-Preparation

SHARING RANDOM VALUES — Every party $P_i \in \mathcal{P}$ 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-$(t, 2t)$-Sh as a dealer, to $(t, 2t)$-share S_i. Let this instance of S-$(t, 2t)$-Sh be denoted as S-$(t, 2t)$-Sh$_i$. On the other hand, in P-Preparation, invoke P-$(t, 2t)$-Sh as a dealer, to $(t, 2t)$-share S_i and denote this instance as P-$(t, 2t)$-Sh$_i$.
2. For $j = 1, \ldots, n$, participate in the instance S-$(t, 2t)$-Sh$_j$/P-$(t, 2t)$-Sh$_j$, invoked by the dealer P_j, depending upon whether it is S-Preparation or P-Preparation.

AGREEMENT ON THE COMMON SUBSET — Every party $P_i \in \mathcal{P}$ executes this code:

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

Let C be the common subset of size $n - t$ obtained as the output of 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 — Every party $P_i \in \mathcal{P}$ executes this code:

1. For every P_j in the common subset C, wait to receive 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 $c_M + c_R$ output random values which are $(t, 2t)$-shared.

Fig. 16. Protocol for the preparation phase.

Proof. For the termination property, we first notice that the invocation of ACS will indeed output a common subset C of $n-t$ parties. This is because there are at least $n-t$ honest parties, who will invoke an instance of S-$(t, 2t)$-Sh and these instances will be eventually terminated by every honest party. We next claim that every honest party will eventually terminate the S-$(t, 2t)$-Sh instance of every party (dealer) in C. The claim is trivially true for every honest party in C. For any corrupted party P_j in C, at least one honest party must have terminated the instance S-$(t, 2t)$-Sh$_j$; the termination property of S-$(t, 2t)$-Sh ensures that the instance S-$(t, 2t)$-Sh$_j$ will be eventually terminated by every other honest party.

If the common subset C contains only honest parties then the correctness property holds trivially without any error. This is because each honest party indeed $(t, 2t)$-shares 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-$(t, 2t)$-Sh ensures that even a corrupted party in C does $(t, 2t)$-sharing of L values (possibly non-random), except with probability ϵ in its instance of S-$(t, 2t)$-Sh. This implies that except with probability at most $t \cdot \epsilon$, every corrupted party in C has done $(t, 2t)$-sharing of L values. Assuming that either $t \cdot \epsilon \approx \epsilon$ or by invoking each instance of S-$(t, 2t)$-Sh 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 uniformly random. Now the property of Ext ensures that the output $(t, 2t)$-shared values are indeed random.

The secrecy property of S-$(t, 2t)$-Sh 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 output shared values $r_{i,j}$ are information-theoretically secure. In the protocol, other than the execution of the instances of S-$(t, 2t)$-Sh, there is no interaction among the parties. The function Ext is applied locally on the shares of the values, shared by the parties in C. This implies that the $r_{i,j}$ values remain information-theoretically secure.

In the protocol, each party executes an instance of S-$(t, 2t)$-Sh to $(t, 2t)$-share $L = \frac{c_M + c_R}{n-2t}$ values. Substituting $\ell = L$ in Theorem 9, this requires a communication of $\mathcal{O}((Ln^3 + n^4) \log |\mathbb{F}|)$ bits and a total broadcast of $\mathcal{O}(n^4 \log |\mathbb{F}|)$ bits. Since $L = \frac{c_M + c_R}{n-2t}$ and $n - 2t = \Theta(n)$, the communication complexity turns out to be $\mathcal{O}(((c_M + c_R)n^2 + n^4) \log |\mathbb{F}|)$ bits. Moreover, the protocol 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-$(t, 2t)$-Sh instead of S-$(t, 2t)$-Sh.

Lemma 21. *Protocol* P-Preparation *satisfies the following properties for every possible* $\mathcal{A}_t$:

1. ***Termination***: *All the honest parties eventually terminate the protocol.*
2. ***Correctness***: *In the protocol, the parties output* $(t, 2t)$*-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 shared values* $r_{i,j}$ *remain information-theoretically secure.*
4. ***Communication Complexity***: *the protocol incurs a communication of* $\mathcal{O}((c_M + c_R)n^2 \log |\mathbb{F}|)$ *bits, a total broadcast of* $\mathcal{O}(n^3 \log |\mathbb{F}|)$ *bits and requires one invocation of* ACS.

6.2. *Input Phase*

In this phase, each party P_i t-shares its input x_i (for the computation), by executing an instance of the sharing protocol of our AVSS schemes. If the parties invoke the statistical protocol S-MS-Sh, then the resultant protocol for the input phase is called S-Input. On the other hand, if the parties use the perfect protocol P-MS-Sh 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 S-MS-Sh/P-MS-Sh instances of more than $n - t$ parties. In order to agree on a common subset C (this should not be confused with the common subset C of the previous phase) of parties whose instance of S-MS-Sh/P-MS-Sh will eventually terminate, one instance of ACS is invoked. The parties then consider t-sharing of the inputs shared by the parties in the common subset 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 S-MS-Sh and P-MS-Sh, respectively.

Protocol S-Input/P-Input

SHARING THE INPUTS — Every party $P_i \in \mathcal{P}$ executes this code:

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

AGREEMENT ON THE COMMON SUBSET — Every party $P_i \in \mathcal{P}$ executes this code:

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

OUTPUT GENERATION — Every party $P_i \in \mathcal{P}$ executes this code:

1. Wait until the instance of ACS terminates with an 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.

Fig. 17. Protocol for the input phase.

Lemma 22. *Protocol* S-Input *satisfies the following properties for every possible* $\mathcal{A}_t$*:*

1. ***Termination***: *All honest parties eventually terminate the protocol.*
2. ***Correctness***: *Except with probability at most* ϵ*, the parties output* t*-sharing of the inputs of the parties in the common subset* C*.*
3. ***Secrecy***: *The inputs of the honest parties in the set* C *remain information-theoretically secure.*
4. ***Communication Complexity***: *The protocol requires a communication of* $\mathcal{O}((c_I n^2 + n^4) \log |\mathbb{F}|)$ *bits, a total broadcast of* $\mathcal{O}(n^4 \log |\mathbb{F}|)$ *bits and requires one invocation of* ACS*.*

Proof. Every honest party will t-share its input and its instance of S-MS-Sh 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 an output C. To show the termination property, we require to show that the S-MS-Sh instance of the corrupted parties in C will be eventually terminated by every honest party. However, this follows from the termination property of S-MS-Sh and the fact that for every corrupted party $P_j \in C$, there exists at least one honest party who terminates the S-MS-Sh instance of P_j.

Every honest party in C will correctly t-share its input in its instance of S-MS-Sh. The correctness property of S-MS-Sh also ensures that even a corrupted $P_i \in C$ will t-share a value x_i (which may or may not be its actual input; but the value shared by the party is considered as its intended input), except with probability at most ϵ. So the inputs of each party in C will be t-shared, except with probability at most $t \cdot \epsilon \approx \epsilon$.

The secrecy property of S-MS-Sh ensures that the input x_i of every honest P_i in C remains information-theoretically secure in the instance S-MS-Sh$_i$. Apart from the execution of the instances of S-MS-Sh, 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 S-MS-Sh to t-share its input x_i. From Theorem 7, we find that this requires a total communication of $\mathcal{O}((c_I n^2 + n^4) \log |\mathbb{F}|)$ bits and a total broadcast of $\mathcal{O}(n^4 \log |\mathbb{F}|)$ bits. In addition, the protocol requires one invocation of ACS. □

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

Lemma 23. *Protocol* P-Input *satisfies the following properties for every possible* $\mathcal{A}_t$:

1. ***Termination***: *All the honest parties eventually terminate the protocol.*
2. ***Correctness***: *The parties correctly output* t*-sharing of the inputs of the parties in the common subset* C.
3. ***Secrecy***: *The inputs of the honest parties in the set* C *remain information-theoretically secure.*
4. ***Communication Complexity***: *the protocol incurs a total communication of* $\mathcal{O}(c_I n^2 \log |\mathbb{F}|)$ *bits, a total broadcast 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 and the perfect AMPC protocols. Here the circuit is evaluated gate by gate, where all intermediate values during the computation remain t-shared. As soon as a party holds its shares of the input values of a gate, it joins the evaluation of the gate. Due to the linearity of 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 output 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 was also used in the AMPC protocol of [5]. The protocol for this phase is called Computation, which is presented in Fig. 18.

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 for every possible* $\mathcal{A}_t$:

1. ***Termination***: *All the honest parties eventually terminate the protocol.*
2. ***Correctness***: *The parties correctly output 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 requires a communication of* $\mathcal{O}(n^2(c_M + c_O) \log |\mathbb{F}|)$ *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

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 — Every party $P_i \in \mathcal{P}$ 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:** $[s]_t = \mathsf{IGate}([s]_t)$: 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, where $k \in \{1, \ldots, c_M\}$. 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) Send the share δ_i to every party in $\mathcal{P}$. Apply OEC on the received shares δ_j to 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, where $k \in \{1, \ldots, c_R\}$. 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:** $x = \mathsf{OGate}([x]_t)$: Send x_i, the i^{th} share of x to every party in $\mathcal{P}$. Apply OEC on the received shares x_j and output x.

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

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

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 involves only 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 assumed 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 at most 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 view-point 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.[16]

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 its share to every other party, incurring a communication of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. □

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

The statistical AMPC protocol S-AMPC 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 S-AMPC.

Theorem 11. *Protocol* S-AMPC *is a statistical AMPC protocol, satisfying Definition 5. The protocol incurs a communication of* $\mathcal{O}(((c_I + c_M + c_R + c_O)n^2 + n^4) \log |\mathbb{F}|)$ *bits, a total broadcast 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 protocol Computation (Lemma 24). □

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

The perfect AMPC protocol P-AMPC consists of the following three steps:

1. Invoke P-Preparation.

[16] As mentioned earlier, we can prove the secrecy in the framework of real-world/ideal-world paradigm of [7]. However, we avoid doing so, as it requires additional technicalities which will make the paper complicated.

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

We next state the properties of the protocol P-AMPC.

Theorem 12. *Protocol* P-AMPC *is a perfect AMPC protocol, satisfying Def. 5. The protocol requires a communication of* $\mathcal{O}((c_I + c_M + c_R + c_O)n^2 \log |\mathbb{F}|)$ *bits, a total broadcast 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 protocol Computation (Lemma 24). □

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 P-Sh 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 it lacks $d + 1 - t$ 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, $f_0(x)$ has $d + 1 - t$ "degree of freedom". So D can share $d + 1 - t$ secrets using the single polynomial $f_0(x)$. This concept, known as the packed secret sharing was introduced in [26], but for the synchronous setting.[17] In what follows we show how the protocol P-Sh can be used as a packed secret-sharing scheme where D can share $d + 1 - t$ secrets simultaneously in the information-theoretic sense. Moreover, even if D is corrupted, there exist $d + 1 - t$ 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 $k = d + 1 - t$. 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 bivariate 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 P-Sh using the bivariate polynomial $F(x, y)$ selected as above and the parties participate in this instance of P-Sh. If D is honest, then by the termination property of P-Sh, every honest party P_i will eventually terminate the protocol, with its 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 P-Sh. 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 $s_1, \ldots, s_k$.

[17] In [26], the concept was introduced in a slightly different way but the essence was the same.

The above idea is also applicable for the protocol P-MS-Sh, where D can use ℓ independent bivariate polynomials, each of degree-(d, t) and using each polynomial, it can share $d + 1 - t$ values. This implies that it can share a total of $\ell \cdot (d + 1 - t)$ values. This requires a communication of $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits and a broadcast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits (Theorem 8). Setting $d = 2t$ (the maximum allowed value of d), we see that P-MS-Sh can share $\ell(t + 1) = \Theta(\ell n)$ values, incurring a communication of $\mathcal{O}(\ell n^2 \log |\mathbb{F}|)$ bits and a broadcast of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits. As the broadcast communication is independent of ℓ, we may ignore it and conclude that the amortized cost of sharing a single secret using P-MS-Sh 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 P-MS-Sh 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 P-MS-Sh tolerating an active adversary matches the cost of sharing a single field element in the presence of a passive adversary (for example, the Shamir's secret-sharing scheme [42]).

Notice that the above discussion holds for the statistical protocol S-MS-Sh, as well. However, the protocol S-MS-Sh may involve a negligible error. On the other hand, protocol P-MS-Sh 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 Huang et al. [30] and show that it does not satisfy the correctness and the termination condition. The AMPC protocol in [30] is divided into a sequence of three phases: the preparation phase, the input phase and the computation and output phase. We concentrate on the preparation phase and show that it fails to satisfy the correctness and the termination property, as claimed in [30]. This further implies that the AMPC protocol of Huang et al. [30] 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 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 (BSS) scheme in [45]. In [45], the authors claimed that their BSS scheme correctly generates c_M shared random multiplication triples over $\mathbb{F}$. Moreover, every honest party will eventually terminate the BSS protocol. 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 in [30] (which uses the BSS scheme as a black box) does not satisfy the correctness and the termination property.

The BSS scheme of Zheng et al. [45] is based on the player-elimination framework [29], where the computation is divided into a sequence of segments. To show the weakness in the BSS scheme of [45], 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. GENERATION OF t-SHARING OF a AND b — Every party $P_i \in \mathcal{P}$ executes this code:
   (a) Select two random polynomials $f_i(x)$ and $g_i(x)$ of degree at most t and send the shares $f_i(j), g_i(j)$ to every $P_j \in \mathcal{P}$. After sending, broadcast 1 to indicate that the sharing of $f_i(x)$ and $g_i(x)$ is done.
   (b) Participate in an instance of ACS and input 1 in ABA_j (during ACS) if 1 is received from the broadcast of P_j and if the shares $f_j(i), g_j(i)$ are received from P_j.
   (c) Let C be the common subset which is output during ACS, 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 [45]). 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.

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

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 random a and b in the BSS scheme of [45] is generated by executing the steps presented in Fig. 19.

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 provided every (honest) P_i eventually holds a_i and b_i. Clearly, this is possible if every (honest) party P_i eventually receives the points $f_j(i)$ and $g_j(i)$ from every $P_j \in C$. In [45], the authors claimed that this will be indeed the case. However, we now show that the adversary may behave (especially schedules the messages) in such a way that every honest P_i has to wait indefinitely to compute a_i and b_i.

Without loss of generality, let the first $n-t$ parties (i.e. $P_1, \ldots, P_{n-t}$) be honest. Now consider the following behavior of a *corrupted* P_j: it selects $f_j(x)$ and $g_j(x)$ of degree higher than t and sends 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 broadcasts 1 to indicate that it has done the sharing of $f_j(x)$ and $g_j(x)$. Moreover, the adversary schedules the messages of P_j such that they reach 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 during ACS, as they will receive points on $f_j(x)$ and $g_j(x)$ from P_j and will also receive 1 from the broadcast 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 honestly 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 subset C. However, notice that the t honest parties $P_{n-2t+1}, \ldots, P_{n-t}$ do 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 $\mathcal{P}$ waits indefinitely to receive its 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 The reason behind the above flaw is that the BSS scheme of [45] (which basically is the Shamir secret-sharing scheme, executed in the asynchronous setting, assuming an honest dealer), does not ensure that if some honest party terminates the protocol with its share, then every other honest party will also eventually do the same. A BSS scheme (in the asynchronous setting) with the "stronger" termination property will certainly avoid the above problem; however we are not aware of any BSS scheme satisfying such a termination property.

An alternative to fix the above problem is to ask each P_j to share two random values, say a_j and b_j using the Sh protocol of some AVSS scheme and then use the ACS primitive to agree on a common subset of $n-t$ parties C whose instances of Sh will be eventually terminated by all the (honest) parties. 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 ith share of a_j and b_j, respectively. By the termination property of AVSS, every (honest) P_i will eventually terminate the Sh instance 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 scheme with $n = 4t+1$ (prior to our work) is due to [5], which requires a communication 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

We presented information-theoretically secure AMPC protocols with $n = 4t+1$, that achieve an amortized communication complexity of $\mathcal{O}(n^2 \log |\mathbb{F}|)$ bits per multiplication gate, which improves the communication complexity of the previous best known protocols by a factor of $\Omega(n)$. The key innovation behind our protocols are new AVSS schemes, which allow to verifiably share secrets with the degree of sharing d, with $t \leq d \leq 2t$. While our perfect AMPC (and AVSS) protocol have optimal resilience, the statistical protocols do not achieve optimal resilience. It would be interesting to improve the resilience of our statistical protocols.

Acknowledgements

We would like to thank the anonymous referees for their valuable comments, which helped us to significantly improve the overall presentation of the article. 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 article. Finally we would also like to thank Nigel P. Smart for giving his insightful remarks on the revised version of the article. The work of A. Patra has been supported in part by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO. The work of A. Choudhury has been supported in part by EPSRC via grant EP/I03126X.

Appendix A. Proof of the Technical Lemmas

Proof of Lemma 1. Let $V^{(k)}$ denote the $k \times k$ Vandermonde matrix, with $[i^0, \ldots, i^{k-1}]^T$ as the ith column, 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 $H = ((V^{(t+1)})^T)^{-1} \cdot E$ 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 polynomials, for $i = 1, \ldots, t+1$ and $j = 1, \ldots, d+1$, we have

$$\overline{F}(j, i) = \left(V^{(t+1)}\right)^T \cdot H \cdot V^{(d+1)} = E \cdot V^{(d+1)} = \overline{f}_i(j) = \overline{g}_j(i).$$

This implies that for $i = 1, \ldots, t+1$, the polynomials $\overline{F}(x, i)$ and $\overline{f}_i(x)$ have the 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 the 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 a similar argument, we can show that $\overline{F}(j, y) = \overline{g}_j(y)$, for $d+1 < j \leq m$. □

Proof of Lemma 4. 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 $h_1(y)$ have 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 $h_{\text{com}}(y) \stackrel{def}{=} r^0 h_0(y) + \ldots + r^l h_l(y)$ can be written as:

$$\begin{aligned} h_{\text{com}}(y) &= r^0\left[c_0 y^{t_{\max}} + \widehat{h_0}(y)\right] + \cdots + r^l\left[c_l y^{t_{\max}} + \widehat{h_l}(y)\right], \\ &= y^{t_{\max}}\left(r^0 c_0 + \cdots + r^l c_l\right) + \Sigma_{j=0}^{l} r^j \widehat{h_j}(y), \\ &= y^{t_{\max}} c_{\text{com}} + \Sigma_{j=0}^{l} r^j \widehat{h_j}(y), \text{ where } c_{\text{com}} = r^0 c_0 + \cdots + r^l c_l. \end{aligned}$$

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 $c_{\text{com}} = r^0 c_0 + \cdots + r^l c_l$ 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, $c_{\text{com}} \neq 0$ and so $h_{\text{com}}(y)$ will have degree higher than t, which is a contradiction. □

References

[1] I. Abraham, D. Dolev, J.Y. Halpern, An almost-surely terminating polynomial protocol for asynchronous Byzantine agreement with optimal resilience, in *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing—PODC 2008*, Toronto, Canada, August 18–21, 2008, ed. by R.A. Bazzi, B. Patt-Shamir (ACM, New York, 2008), pp. 405–414

[2] D. Beaver, Efficient multiparty protocols using circuit randomization, in *Advances in Cryptology—CRYPTO '91, Proceedings of 11th Annual International Cryptology Conference*, Santa Barbara, California, USA, August 11–15, 1991, ed. by J. Feigenbaum. Lecture Notes in Computer Science, vol. 576 (Springer, Berlin, 1991), pp. 420–432

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

[4] Z. Beerliová-Trubíniová, M. Hirt, Efficient multi-party computation with dispute control, in *Theory of Cryptography, Proceedings of 3rd Theory of Cryptography Conference—TCC 2006*, New York, NY, USA, March 4–7, 2006, ed. by S. Halevi, T. Rabin. Lecture Notes in Computer Science, vol. 3876 (Springer, Berlin, 2006), pp. 305–328

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

[6] Z. Beerliová-Trubíniová, M. Hirt, Perfectly-secure MPC with linear communication complexity, in *Theory of Cryptography, 5th Theory of Cryptography Conference—TCC 2008*, New York, USA, March 19–21, 2008, ed. by R. Canetti. Lecture Notes in Computer Science, vol. 4948 (Springer, Berlin, 2008), pp. 213–230

[7] M. Ben-Or, R. Canetti, O. Goldreich, Asynchronous secure computation, in *Proceedings of the 25th Annual ACM Symposium on Theory of Computing* (ACM, New York, 1993), pp. 52–61

[8] M. Ben-Or, S. Goldwasser, 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*, Chicago, Illinois, USA, May 2–4, 1988 (ACM, New York, 1988), pp. 1–10

[9] M. Ben-Or, B. Kelmer, T. Rabin, Asynchronous secure computations with optimal resilience, in *Proceedings of the 13th Annual ACM Symposium on Principles of Distributed Computing*, Los Angeles, California, USA, August 14–17 (ACM, New York, 1994), pp. 183–192

[10] E. Ben-Sasson, S. Fehr, R. Ostrovsky, Near-linear unconditionally-secure multiparty computation with a dishonest minority, in *Proceedings 32nd Annual Cryptology Conference of Advances in Cryptology—CRYPTO 2012*, Santa Barbara, CA, USA, August 19–23, 2012, ed. by R. Safavi-Naini, R. Canetti. Lecture Notes in Computer Science, vol. 7417 (Springer, Berlin, 2012), pp. 663–680

[11] C.H. Bennett, G. Brassard, C. Crépeau, U.M. Maurer, Generalized privacy amplification. *IEEE Trans. Inf. Theory* **41**(6), 1915–1923 (1995)

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

[13] G. Bracha, An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol, in *Proceedings of the 3rd Annual ACM Symposium on Principles of Distributed Computing*, Vancouver, B.C., Canada, August 27–29, 1984 (ACM, New York, 1984), pp. 154–162

[14] R. Canetti, Studies in secure multiparty computation and applications. Ph.D. thesis, Weizmann Institute, Israel, 1995

[15] R. Canetti, T. Rabin, Fast asynchronous Byzantine agreement with optimal resilience, in *Proceedings of the 25th Annual ACM Symposium on Theory of Computing* (ACM, New York, 1993), pp. 42–51

[16] D. Chaum, C. Crépeau, I. Damgård, Multiparty unconditionally secure protocols (extended abstract), in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing—ACM 1988*, Chicago, Illinois, USA, May 2–4, 1988 (ACM Press, New York, 1988), pp. 11–19

[17] B. Chor, S. Goldwasser, S. Micali, 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*, Providence, Rhode Island, USA, 6–8 May, 1985 (ACM, New York, 1985), pp. 383–395

[18] A. Choudhury, M. Hirt, A. Patra, Asynchronous multiparty computation with linear communication complexity, in *Distributed Computing—DISC 2013, Proceedings of 27th International Symposium*, Jerusalem, Israel, October 14–18, 2013, ed. by Y. Afek. Lecture Notes in Computer Science, vol. 8205 (Springer, Berlin, 2013), pp. 388–402

[19] R. Cramer, I. Damgård, S. Dziembowski, M. Hirt, T. Rabin, Efficient multiparty computations secure against an adaptive adversary, in *Advances in Cryptology—EUROCRYPT 99, Proceeding of International Conference on the Theory and Application of Cryptographic Techniques*, Prague, Czech Republic, May 2–6, 1999, ed. by J. Stern. Lecture Notes in Computer Science, vol. 1592 (Springer, Berlin, 1999), pp. 311–326

[20] I. Damgård, M. Geisler, M. Krøigaard, J.B. Nielsen, Asynchronous multiparty computation: theory and implementation, in *Public Key Cryptography—PKC 2009, Proceeding of 12th International Conference on Practice and Theory in Public Key Cryptography*, Irvine, CA, USA, March 18–20, 2009, ed. by S. Jarecki, G. Tsudik. Lecture Notes in Computer Science, vol. 5443 (Springer, Berlin, 2009), pp. 160–179

[21] I. Damgård, Y. Ishai, Scalable secure multiparty computation, in *Advances in Cryptology—CRYPTO 2006, Proceedings of 26th Annual International Cryptology Conference*, Santa Barbara, California, USA, August 20–24, 2006, ed. by C. Dwork. Lecture Notes in Computer Science, vol. 4117 (Springer, Berlin, 2006), pp. 501–520

[22] I. Damgård, J.B. Nielsen, Scalable and unconditionally secure multiparty computation, in *Advances in Cryptology—CRYPTO 2007, 27th Annual International Cryptology Conference*, Santa Barbara, CA, USA, August 19–23, 2007, Proceedings, ed. by A. Menezes. Lecture Notes in Computer Science, vol. 4622 (Springer, Berlin, 2007), pp. 572–590

[23] D. Dolev, C. Dwork, O. Waarts, M. Yung, Perfectly secure message transmission. *J. ACM* **40**(1), 17–47 (1993)

[24] P. Feldman, S. Micali, An optimal algorithm for synchronous Byzantine agreement, in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*, Chicago, Illinois, USA, May 2–4, 1988 (ACM, New York, 1988), pp. 639–648

[25] M. Fitzi, J. Garay, S. Gollakota, C. Pandu Rangan, K. Srinathan, Round-optimal and efficient verifiable secret sharing, in *Theory of Cryptography—TCC 2006, Proceedings of 3rd Theory of Cryptography Conference*, New York, NY, USA, March 4–7, 2006, ed. by S. Halevi, T. Rabin. Lecture Notes in Computer Science, vol. 3876 (Springer, Berlin, 2006), pp. 329–342

[26] M.K. Franklin, M. Yung, Communication complexity of secure computation (extended abstract), in *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, Victoria, British Columbia, Canada, May 4–6, 1992, ed. by S.R. Kosaraju, M. Fellows, A. Wigderson, J.A. Ellis (ACM, New York, 1992), pp. 699–710

[27] R. Gennaro, Y. Ishai, E. Kushilevitz, T. Rabin, The round complexity of verifiable secret sharing and secure multicast, in *Proceedings on 33rd Annual ACM Symposium on Theory of Computing*, Heraklion, Crete, Greece, July 6–8, 2001 (ACM, New York, 2001), pp. 580–589

[28] O. Golderich, S. Micali, A. Wigderson, How to play a mental game or a completeness theorem for protocols with honest majority, in *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, New York, NY, USA (ACM, New York, 1987), pp. 218–229

[29] M. Hirt, U. Maurer, B. Przydatek, Efficient secure multiparty computation, in *Advances in Cryptology—ASIACRYPT 2000, Proceedings of 6th International Conference on the Theory and Application of Cryptology and Information Security*, Kyoto, Japan, December 3–7, 2000, ed. by T. Okamoto. Lecture Notes in Computer Science, vol. 1976 (Springer, Berlin, 2000), pp. 143–161

[30] Z. Huang, W. Qiu, Q. Li, K. Chen, Efficient secure multiparty computation protocol in asynchronous network, in *Proceedings of Advances in Information Security and Assurance—ISA 2009, 3rd International Conference and Workshops*, Seoul, Korea, ed. by J.H. Park, H. Chen, M. Atiquzzaman, C. Lee, T. Kim, S. Yeo. Lecture Notes in Computer Science, vol. 5576 (Springer, Berlin, 2009), pp. 152–158

[31] J. Katz, C. Koo, R. Kumaresan, Improving the round complexity of VSS in point-to-point networks, in *Automata, Languages and Programming—ICALP 2008, Proceedings of 35th International Colloquium, Part II, Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography*

Foundations, Reykjavik, Iceland, July 7–11, 2008, ed. by L. Aceto, I. Damgård, L.A. Goldberg, M.M. Halldórsson, A. Ingólfsdóttir, I. Walukiewicz. Lecture Notes in Computer Science, vol. 5126 (Springer, Berlin, 2008), pp. 499–510

[32] J. Katz, C.Y. Koo, On expected constant-round protocols for Byzantine agreement, in *Advances in Cryptology—CRYPTO 2006, Proceedings of 26th Annual International Cryptology Conference*, Santa Barbara, California, USA, August 20–24, 2006, Lecture Notes in Computer Science, ed. by C. Dwork (Springer, Berlin, 2006), pp. 445–462

[33] F.J. MacWilliams, N.J.A. Sloane, *The Theory of Error Correcting Codes* (North-Holland, Amsterdam, 1978)

[34] A. Patra, A. Choudhary, T. Rabin, C. Pandu Rangan, The round complexity of verifiable secret sharing revisited, in *Advances in Cryptology—CRYPTO 2009, Proceedings of 29th Annual International Cryptology Conference*, Santa Barbara, CA, USA, August 16–20, 2009, ed. by S. Halevi. Lecture Notes in Computer Science, vol. 5677 (Springer, Berlin, 2009), pp. 487–504

[35] A. Patra, A. Choudhary, C. Pandu Rangan, Round efficient unconditionally secure multiparty computation protocol, in *Progress in Cryptology—INDOCRYPT 2008, Proceedings of 9th International Conference on Cryptology in India*, Kharagpur, India, December 14–17, 2008, ed. by D.R. Chowdhury, V. Rijmen, A. Das. Lecture Notes in Computer Science, vol. 5365 (Springer, Berlin, 2008), pp. 185–199

[36] A. Patra, A. Choudhary, C. Pandu Rangan, Efficient asynchronous Byzantine agreement with optimal resilience. (accepted for publication) Distr. Comput. J. 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, 10–12 August, pp. 92–101 (2009)

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

[38] A. Patra, A. Choudhary, C. Pandu Rangan, Communication efficient perfectly secure VSS and MPC in asynchronous networks with optimal resilience, in *Advances in Cryptology—AFRICACRYPT'10, Proceedings of 3rd International Conference in Cryptology in Africa*, Stellenbosch, South Africa, May 3–6, 2009, ed. by D.J. Bernstein, T. Lange. Lecture Notes in Computer Science, vol. 6055 (Springer, Berlin, 2010), pp. 184–202

[39] B. Prabhu, K. Srinathan, C. Pandu Rangan, Trading players for efficiency in unconditional multiparty computation, in *3rd International Conference on Security in Communication Networks—SCN 2002*, Amalfi, Italy, September 11–13, 2002, ed. by S. Cimato, C. Galdi, G. Persiano. Lecture Notes in Computer Science, vol. 2576 (Springer, Berlin, 2002), pp. 342–353. Revised papers

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

[41] T. Rabin, 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*, Seattle, Washington, USA, May 14–17, 1989 (ACM, New York, 1989), pp. 73–85

[42] A. Shamir, How to share a secret. *Commun. ACM* **22**(11), 612–613 (1979)

[43] K. Srinathan, C. Pandu Rangan, Efficient asynchronous secure multiparty distributed computation, in *Progress in Cryptology—INDOCRYPT 2000, Proceedings of 1st International Conference in Cryptology in India*, Calcutta, India, December 10–13, 2000, ed. by B.K. Roy, E. Okamoto. Lecture Notes in Computer Science, vol. 1977 (Springer, Berlin, 2000), pp. 117–129

[44] A.C. Yao, Protocols for secure computations, in *Proceedings of 23rd Annual Symposium on Foundations of Computer Science*, Chicago, Illinois, 3–5 November 1982, IEEE Computer Society (1982), pp. 160–164

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