Efficient Asynchronous Verifiable Secret Sharing and Byzantine Agreement with Optimal Resilience

Arpita Patra **Ashish Choudhary** **C. Pandu Rangan**
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai India 600036
Email:{ arpita,ashishc }@cse.iitm.ernet.in, rangan@iitm.ernet.in

Abstract

Consider a completely asynchronous network consisting of n players where every two players are connected by a private channel. An adversary $\mathcal{A}_t$ with *unbounded computing power* actively controls at most $t = (\lceil \frac{n}{3} \rceil - 1)$ out of n players in Byzantine fashion. In this setting, we present a new *asynchronous verifiable secret sharing* (AVSS) protocol which privately communicates $\mathcal{O}((\ell n^3 + n^6\kappa)\kappa)$ bits, A-Cast $\mathcal{O}(n^4 \log(n))$ bits during *sharing phase* and privately communicates $\mathcal{O}((\ell n^4 + n^6\kappa)\kappa)$ bits during *reconstruction phase*, to share a secret S containing ℓ secret elements from a finite field $\mathbb{F}$ of size 2^κ, where all the honest players terminate the protocol with probability at least $1 - 2^{-\mathcal{O}(\kappa)}$. Conditioned on the event that all non-faulty players have completed the execution of the AVSS protocol, the non-faulty players terminates in constant expected time. Our protocol is to be compared with the AVSS protocol of [7] in the same settings and having the same properties. The AVSS protocol of [7] privately communicates $\mathcal{O}(\ell n^9 \kappa^4)$ bits, A-Cast $\mathcal{O}(\ell n^9 \kappa^2 \log(n))$ bits during *sharing phase* and privately communicates $\mathcal{O}(\ell n^6 \kappa^3)$ bits, A-Cast $\mathcal{O}(\ell n^6 \kappa \log(n))$ bits during *reconstruction phase*. Thus our new AVSS protocol significantly improves over the communication complexity of the AVSS protocol of [7]. As an application of our new AVSS protocol, we present a new *asynchronous Byzantine Agreement* (ABA) protocol, with $n = 3t + 1$, which significantly improves over the communication complexity of the ABA protocol of [7] and [1]. To design our AVSS protocol, we use several interesting and new techniques, which are of independent interest.

Keywords: Unbounded Computing Power, VSS, Byzantine Agreement, Asynchronous Networks.

1 Introduction

In *secret sharing* [20], a dealer **D** wants to share a secret s among a set of n players, such that no set of t players can reconstruct s while any set of $t+1$ or more players can reconstruct s by pooling their shares. Verifiable secret sharing (VSS) [8] extends secret sharing to work against active corruption. It is a stronger notion than secret sharing and provides robustness against t malicious players (possibly including **D**), having unbounded computing power. VSS is one of the fundamental tools in secure distributed computing and is used as a black box in several other distributed computing tasks such as multiparty computation (MPC), Byzantine Agreement (BA), etc. The important parameters of any VSS scheme are round complexity, fault tolerance and communication complexity. In the past two decades, a lot of research has been carried out to design efficient and optimal VSS protocols, which try to optimize these parameters (see [14, 15, 13] and their references). However, all these results assume that the underlying network is synchronous. Though theoretically impressive, these results do not fit too well in the real life scenario like internet due to synchrony assumed in the network.

Asynchronous Networks: In asynchronous networks, messages are delayed arbitrarily. As a worst case assumption, the adversary is given the power to schedule the delivery of messages. Such networks models real life networks like the Internet much better than their synchronous counterpart. However, protocols for asynchronous networks are much more involved than their synchronous counterparts. This is so because if a player does not receive an expected message then he cannot decide whether the sender is corrupt (and did not send the message at all) or the message is just delayed in the network.

Thus in a fully asynchronous settings, it is impossible to consider the inputs of all uncorrupted players. So input of up to t (potentially honest) players have to be ignored because waiting for them could turn out to be endless. Also, the existing protocols and techniques used in synchronous settings cannot be trivially extended in asynchronous settings.

Asynchronous Verifiable Secret Sharing (AVSS): Though VSS in synchronous settings [14, 15, 13, 9, 19, 3] has been studied extensively, *asynchronous verifiable secret sharing* (AVSS) has drawn very little attention. It is known that perfectly secure (i.e., information theoretic security with zero error probability) AVSS is possible iff $n \geq 4t + 1$ [6]. Similarly AVSS achieving information theoretic security with negligible error probability is possible iff $n \geq 3t + 1$ [6]. Moreover, there exists efficient perfect AVSS protocol with $n = 4t + 1$ [6]. However, with $n = 3t + 1$, the *only* known AVSS protocol is due to [7], which privately communication $\mathcal{O}(n^9\kappa^4)$ bits, A-Cast $\mathcal{O}(n^9\kappa^2 \log(n))$ bits during *sharing phase* and privately communicates $\mathcal{O}(n^6\kappa^3)$ bits, A-Cast $\mathcal{O}(n^6\kappa \log(n))$ bits during *reconstruction phase* for sharing a single secret s, where all the honest players terminate the protocol with probability at least $1 - 2^{-\mathcal{O}(\kappa)}$ and κ is the error parameter. Here A-Cast is a primitive in asynchronous world, allowing a player to send the same value to all the other players. Hence A-Cast in asynchronous would is the parallel notion of broadcast in synchronous world. The AVSS protocol of [7] has the following property: conditioned on the event that all non-faulty players have completed the execution of the AVSS protocol, the non-faulty players terminate in constant expected time. Now in order to share ℓ secrets, the AVSS protocol of [7] need to be executed ℓ times, resulting in a private communication of $\mathcal{O}(\ell n^9\kappa^4)$ bits, A-Cast of $\mathcal{O}(\ell n^9\kappa^2 \log(n))$ bits during *sharing phase* and private communication of $\mathcal{O}(\ell n^6\kappa^3)$ bits, A-Cast of $\mathcal{O}(\ell n^6\kappa \log(n))$ bits during *reconstruction phase*.

Asynchronous Byzantine Agreement (ABA) and Application of AVSS in ABA: Roughly speaking the problem of ABA is as follows: a set of n players are connected with each other by direct and secure asynchronous channels. An adversary $\mathcal{A}_t$ with *unbounded computing power* can control at most t of n nodes in Byzantine fashion. By Byzantine means that the adversary takes full control of the player and forces it to behave arbitrarily during the protocol execution. Each player has a binary input value. The goal is for all honest players to agree on a consensus value that is an input value of one of the honest players. The challenge is to design a protocol, which achieves this goal even in the presence of $\mathcal{A}_t$. This problem was first formulated by Pease, Shostak and Lamport [18]. It is known that ABA tolerating $\mathcal{A}_t$ is possible iff $n = 3t + 1$ [16]. The first ABA protocol with $n = 3t + 1$ was designed in [7], who used their AVSS protocol (with $n = 3t + 1$) to implement a global coin, a standard technique introduced by [11]. Using the AVSS scheme of [7], the global coin protocol of [7] will privately communicate $\mathcal{O}(n^{11}\kappa^4)$ bits and A-Cast $\mathcal{O}(n^{11}\kappa^2 \log(n))$ bits. Now in the ABA protocol of [7], the global coin protocol may be called expected constant number, say c times. Thus the ABA protocol of [7] privately communicates $\mathcal{O}(cn^{11}\kappa^4)$ bits and A-Cast $\mathcal{O}(cn^{11}\kappa^2 \log(n))$ bits. The ABA protocol of [7] has the following property: each honest player will terminate the protocol with probability at least $1 - 2^{-\mathcal{O}(\kappa)}$, where κ is the error parameter. Moreover, conditioned on the event that all non-faulty players have completed the execution of the ABA protocol, the honest players terminates in constant expected time. For a comprehensive description of the reduction from AVSS to ABA, see [6].

From the above description, we find that in the ABA protocol of [7], the honest players may not terminate the protocol with negligible error probability. However, this is unavoidable as implied by the results of Fischer, Lynch and Paterson [12], which states that in asynchronous settings, any BA protocol must have some non-terminating runs. Thus, for asynchronous BA protocol with $n = 3t + 1$, the best we can hope for is that the set of nonterminating executions has probability 0. We say such protocols as *almost-surely terminating* following [1]. Thus though the ABA protocol of [7] has optimal fault tolerance and expected constant running time, it is not *almost-surely terminating*. Recently in [1], the authors have designed an ABA protocol with $n = 3t + 1$ which is almost surely terminating, but terminates in polynomial expected time. The authors of [1] first introduced a new primitive called *shunning VSS* (SVSS), which is a slightly weaker notion of AVSS in the sense that if all the players (including corrupted players) behave correctly then SVSS satisfies the properties of AVSS. Otherwise it does not satisfy the properties of AVSS but will enable some honest player to identify at least one

corrupted player whom the honest player shuns from then onwards. Then in [1], the authors have used their SVSS protocol to implement shunning global coin protocol. Finally in the ABA protocol, the shunning global coin protocol is called $(c+n^2)$ times, where $c > 0$ is some constant, resulting in a private communication of $\mathcal{O}((c+n^2)n^6\log(n))$ bits and A-Cast of $\mathcal{O}((c+n^2)n^6\log(n))$ bits.

Motivation for Our Work: From the above description, it is clear that AVSS is a fundamental and important distributed computing primitive. In addition to ABA, it also finds application in asynchronous multiparty computation (AMPC) [4]. Though the AVSS and ABA scheme of [7] with $n = 3t+1$ has polynomial communication complexity and first of their kind, they involve very high communication complexity. Reducing the communication complexity of AVSS and ABA protocol with optimal fault tolerance; i.e., with $n = 3t+1$, is a challenging and interesting problem which is yet to be attempted for their hardness. Thus motivated by the practical utility of communication efficient AVSS and ABA schemes, we design communication efficient AVSS and ABA schemes in this work.

Principle Used in the AVSS Scheme of [7]: In [7], the authors have clearly explained the inherent difficulties encountered in designing an AVSS scheme with $n = 3t+1$. To overcome these difficulties, the authors in [7] used the following approach to design their AVSS scheme. They first designed a protocol called *Asynchronous Recoverable Sharing* (A-RS). To design A-RS protocol, they used a tool called *Information Checking* (IC) protocol, which was introduced in [19]. Using A-RS as a primitive, the authors have designed a primitive called *Asynchronous Weak Secret Sharing* (AWSS). Then the authors presented a variation of AWSS scheme called *Two & Sum AWSS*. Finally using their *Two & Sum AWSS*, the authors in [7] designed their AVSS scheme. Thus pictorially, the route taken by [7] to design their AVSS scheme is as follows:*IC* $\rightarrow$ *A-RS* $\rightarrow$ *AWSS* $\rightarrow$ *Two & Sum AWSS* $\rightarrow$ *AVSS*. Since the final AVSS scheme is designed on the top of so many sub-protocols, it results in significant communication overhead. In addition, the final AVSS protocol becomes very involved.

Our Contribution and Its Impact Factor: We present a new AVSS protocol with $n = 3t+1$, which privately communicates $\mathcal{O}((\ell n^3 + n^5\kappa)\kappa)$ bits, A-Cast $\mathcal{O}(n^4\log(n))$ bits during *sharing phase* and privately communicates $\mathcal{O}((\ell n^4 + n^5\kappa)\kappa)$ bits during *reconstruction phase*, to share a secret S containing ℓ field elements, where all the honest players terminate the protocol with probability at least $1 - 2^{-\mathcal{O}(\kappa)}$. Conditioned on the event that all non-faulty players have completed the execution of the AVSS protocol, the non-faulty players terminate in constant expected time. This significantly improves the communication complexity of the AVSS scheme of [7], with same properties. We follow the following route to design our AVSS: *IC* $\rightarrow$ *AWSS* $\rightarrow$ *AVSS*, thus avoiding many sub-protocols used in [7].

As an application of our new AVSS protocol, we design a new ABA protocol with $n = 3t+1$, which privately communicates $\mathcal{O}(cn^6\kappa)$ bits and A-Cast $\mathcal{O}(cn^5\log(n))$ bits ($c > 0$ is some constant), thus significantly improving the ABA protocol of [7], while maintaining the same properties.Also comparing our ABA protocol with the ABA protocol of [1], we find that our ABA protocol though not *almost-surely terminating*, terminates in constant expected time with less communication overhead. The following table gives the comparison of our results with the results of [7, 1] in terms of bits.

	AVSS		ABA
	Sharing Phase	Reconstruction Phase	
[7]	Private — $\mathcal{O}(\ell n^9\kappa^4)$ A-Cast — $\mathcal{O}(\ell n^9\kappa^2\log(n))$	Private — $\mathcal{O}(\ell n^6\kappa^3)$ A-Cast — $\mathcal{O}(\ell n^6\kappa\log(n))$	Private — $\mathcal{O}(cn^{11}\kappa^4)$ A-Cast — $\mathcal{O}(cn^{11}\kappa^2\log(n))$
[1]			Private — $\mathcal{O}((c+n^2)n^6\log(n))$ A-Cast — $\mathcal{O}((c+n^2)n^6\log(n))$
This paper	Private — $\mathcal{O}((\ell n^3 + n^6\kappa)\kappa)$ A-Cast — $\mathcal{O}(n^4\log(n))$	Private — $\mathcal{O}((\ell n^4 + n^6\kappa)\kappa)$	Private — $\mathcal{O}(cn^6\kappa)$ A-Cast — $\mathcal{O}(cn^5\log(n))$

2 Network Model, Definitions and Notations

Model: We follow the asynchronous network model of [7], where there are n players, denoted by the set $\mathcal{P}$, who are connected by pairwise reliable and secure channel. Messages sent on a channel may have arbitrary (but finite) delays. An adversary $\mathcal{A}_t$ with *unbounded computing power* can corrupt at most t

players during the protocol in Byzantine fashion. Once a player is corrupted, he remains so throughout the protocol. As a worst case assumption, we assume that $\mathcal{A}_t$ controls the delay in transmission of the messages flowing through different channels and hence he can arbitrarily (but finitely) delay their transmission. However, $\mathcal{A}_t$ can only delay the message sent by the honest players and will have no information about these messages. The error probability of our protocols is expressed in terms of an error parameter $\kappa > 0$, where the error probability of the protocols is $2^{-O(\kappa)}$. To bound the error probability by $2^{-O(\kappa)}$, all our protocols work over a finite field $\mathbb{F}$ where $\mathbb{F} = GF(2^\kappa)$. Thus each field element can be represented by κ bits. Moreover, we assume that n is polynomial in κ.

Asynchronous Weak Secret Sharing (AWSS) [7]: Let (Sh, Rec) be a pair of protocols in which a dealer $\mathbf{D} \in \mathcal{P}$ shares a secret S containing $\ell \geq 1$ field elements. We say that (Sh, Rec) is a t-resilient AWSS scheme for n players if the following hold for every possible $\mathcal{A}_t$.

• **Termination**: With probability at least $1 - 2^{-O(\kappa)}$, the following requirements hold:

1. If $\mathbf{D}$ is honest then each player will eventually terminate protocol Sh.
2. If some honest player has completed protocol Sh, then irrespective of the behavior of $\mathbf{D}$, each honest player will eventually terminate Sh.
3. If an honest player has completed Sh and all the honest players invoke protocol Rec, then each honest player will terminate Rec.

• **Correctness**:

1. If $\mathbf{D}$ is honest then with probability at least $1-2^{-O(\kappa)}$, each honest player upon completing protocol Rec, outputs the shared secret.
2. Once the first honest player completes protocol Sh, then there exists an $r \in \mathbb{F}^\ell \cup NULL$, such that with probability at least $1 - 2^{-O(\kappa)}$, each honest player upon completing Rec, will output either r or $NULL$.

• **Secrecy**: If $\mathbf{D}$ is honest and no honest player has begun executing protocol Rec, then the corrupted players have no information about the shared secret.

As mentioned in [7], we stress that in the case of a corrupted $\mathbf{D}$, even if $r \neq NULL$, then some of the uncorrupted players may output r and some may output $NULL$. The adversary can decide which players will output $NULL$ during the execution of reconstruction protocol.

Asynchronous Verifiable Secret Sharing (AVSS) [7]: The **Termination** and **Secrecy** conditions for AVSS is same as in AWSS. The only difference is in the **Correctness** 2 property:

Correctness 2: Once the first honest player has completed Sh, there exists an unique $r \in \mathbb{F}^\ell$, such that with probability at least $1 - 2^{-O(\kappa)}$, each honest player upon completing Rec, will output r.
Thus the difference between the **Correctness** property of AWSS and AVSS is that in AWSS, if $\mathbf{D}$ is corrupted then he may change the committed secret from r to $NULL$ during reconstruction protocol, while in AVSS, a corrupted $\mathbf{D}$ cannot change his commitment from r to any other value during reconstruction protocol.

Asynchronous Byzantine Agreement (ABA)[7]: Let Π be an asynchronous protocol in which each player has a binary input and let $\kappa > 0$ be the error parameter. We say that Π is a $(1 - 2^{-O(\kappa)})$-terminating, t-resilient Byzantine Agreement protocol if the following hold, for every possible $\mathcal{A}_t$ and input:

• **Termination**: With probability at least $1 - 2^{-O(\kappa)}$ all the honest players complete the protocol (i.e., terminate locally).

• **Correctness**: All the honest players who have terminated have identical outputs. Furthermore, if all the honest players have the same input, say ρ, then all the honest players output ρ.

A-Cast[7]: It is an asynchronous broadcast primitive, which was introduced and elegantly implemented by Bracha [5] with $n = 3t + 1$. Let Π be an asynchronous protocol initiated by a special player (called the sender), having input m (the message to be broadcast). We say that Π is a t-resilient A-cast protocol if the following hold, for every possible $\mathcal{A}_t$ and input:

• **Termination**:

1. If the sender is honest and all the honest players participate in the protocol, then each honest player will eventually complete the protocol.

2. Irrespective of the behavior of the sender, if any honest player completes the protocol then each honest player will eventually complete the protocol.

• **Correctness**: If the honest players complete the protocol then they terminate with a common output m^*. Furthermore, if the sender is honest then $m^* = m$.

Notice that the termination property of A-Cast is much weaker than the termination property of BA because for A-Cast, we do not require that the honest players complete the protocol when the sender is faulty. In the sequel, we use the following convention: we say that player P_j listens the A-Cast of player P_i with value m, meaning that P_j completes the execution of P_i's A-Cast and has locally set the value of the A-Cast to m.

Random Value Generation: We now design a simple protocol called RandomGenerator, which allows the players to jointly generate a random number $r \in \mathbb{F}$. The idea behind RandomGenerator is very simple: Each $P_i \in \mathcal{P}$ with input $x_i \in_R \mathbb{F}$ participates in an instance of asynchronous multiparty computation (AMPC) protocol described in [4], where the function to be computed is $(y_1, \ldots, y_n) = f(x_1, \ldots, x_n)$ and each $y_i = (x_1 + x_2 + \ldots + x_n)$. Now r is nothing but $(x_1 + x_2 + \ldots + x_n)$. Thus at the termination of the AMPC instance, every player in $\mathcal{P}$ will have a value r which is completely random.

Theorem 1 *RandomGenerator correctly generates random r by communicating $O(poly(n\kappa))$ bits.*

3 Communication Complexity Analysis of the AVSS Protocol of [7]

The communication complexity analysis of the AVSS protocol of [7] was never done and we have carried out the same. We now try to briefly give the communication complexity analysis of the AVSS protocol of [7]. To do so, we have considered the detailed description of the AVSS protocol of [7] given in [6]. To begin with, in the **IC** protocol of [7], **D** gives $\mathcal{O}(n\kappa)$ field elements to INT and $\mathcal{O}(\kappa)$ field elements to each of the n receivers/verifiers (for formal definition of INT and receivers/verifiers, see Section 4). So the **IC** protocol of [7] involves a communication complexity of $\mathcal{O}(n\kappa^2)$ bits.

Now by incorporating their **IC** protocol in Shamir secret sharing [20], the authors in [7] designed an asynchronous primitive called **A-RS**, which consists of two sub-protocols, namely **A-RS-Share** and **A-RS-Rec**. In the **A-RS-Share** protocol, **D** generates n shares (Shamir shares) of a secret s and for each of the n shares, **D** executes an instance of **IC** protocol. So the **A-RS-Share** protocol of [7] involves a communication complexity of $\mathcal{O}(n^2\kappa^2)$ bits. In addition to this, the **A-RS-Share** protocol involves an A-Cast of $\mathcal{O}(\log(n))$ bits. In the **A-RS-Rec** protocol, the IC signatures given by **D** in **A-RS-Share** are revealed, which involves a communication complexity of $\mathcal{O}(n^2\kappa^2)$ bits. In addition, the **A-RS-Rec** protocol involves A-Cast of $\mathcal{O}(n^2 \log(n))$ bits.

Proceeding further, by incorporating their **A-RS** protocol, the authors in [7] designed an AWSS protocol. The AWSS protocol consists of two sub-protocols, namely **AWSS-Share** and **AWSS-Rec**. In the **AWSS-Share** protocol, **D** generates n shares (Shamir shares [20]) of the secret and instantiate n instances of the **IC** protocol for each of the n shares. Now each individual player **A-RS-Share** all the values that it has received in the n instances of the **IC** protocol. This involves a communication complexity of $\mathcal{O}(n^4\kappa^3)$ bits and A-Cast of $\mathcal{O}(n^2\kappa \log(n))$ bits. In the **AWSS-Rec** protocol, each player P_i tries to reconstruct the values which are **A-RS-Shared** by each player P_j in a set $\mathcal{E}_i$. Here $\mathcal{E}_i$ is a set which is defined is the **AWSS-Share** protocol. In the worst case, the size of each $\mathcal{E}_i$ is $\mathcal{O}(n)$. So in the worst case, the **AWSS-Rec** protocol communicates $\mathcal{O}(n^5\kappa^3)$ bits and A-Cast $\mathcal{O}(n^5\kappa \log(n))$ bits.

The authors in [7] then further extended their **AWSS-Share** protocol to **Two&Sum AWSS-Share** protocol, where each player P_i has to **A-RS-Share** $\mathcal{O}(n\kappa^2)$ field elements. So the communication complexity of **Two&Sum AWSS-Share** protocol is $\mathcal{O}(n^4\kappa^4)$ bits and A-Cast of $\mathcal{O}(n^2\kappa^2 \log(n))$ bits.

Finally using their **Two&Sum AWSS-Share** and **AWSS-Rec** protocol, the authors in [7] have deigned their **AVSS** protocol, which consists of two sub-protocols, namely **AVSS-Share** and **AVSS-Rec**. In the **AVSS-Share** protocol, the most communication expensive step is the one where each player has to **AWSS-Rec** $\mathcal{O}(n^3\kappa)$ values. So in total, the **AVSS-Share** protocol of [7] involves a communication complexity of $\mathcal{O}(n^9\kappa^4)$ bits and A-Cast $\mathcal{O}(n^9\kappa^2\log(n))$ bits. The **AVSS-Rec** protocol involves n instances of **AWSS-Rec**, resulting in a communication complexity of $\mathcal{O}(n^6\kappa^3)$ bits and A-cast of $\mathcal{O}(n^6\kappa\log(n))$ bits.

4 Information Checking Protocol and IC Signature

The Information Checking (IC) scheme is a tool for authenticating messages in the presence of computationally unbounded corrupted players. In synchronous settings, the notion of IC scheme was first introduced by Rabin et.al [19] who have designed an IC scheme with $n \geq 2t+1$. The IC scheme of Rabin et. al. was extended and adapted in asynchronous settings by Canetti et. al. [7]. We now recall the definition of IC scheme, its properties and its outcome.

Information Checking (IC) Scheme and IC Signatures [19, 7]: Typically, in an IC scheme, the entities involved are the dealer $\mathbf{D} \in \mathcal{P}$, ($\mathbf{D}$ should not be confused with the $\mathbf{D}$ of AWSS and AVSS; but following the standard definition of IC scheme, we overload this notation), an intermediary player $INT \in \mathcal{P}$ and the verifiers in $\mathcal{P}$ (note that $\mathbf{D}$ and INT also belongs to the set of verifiers). IC scheme provides an information theoretically secure method for authenticating data and is used to generate IC signatures on $\mathbf{D}$'s secret S, consisting of ℓ field elements. Once a player $INT \in \mathcal{P}$ receives an IC signature on S from $\mathbf{D} \in \mathcal{P}$, then INT can later produce the signature and have the players in $\mathcal{P}$ verify that it is a valid signature (handed over by $\mathbf{D}$ to INT). An IC scheme consists of a sequence of three protocols[1]:

1. Distr$(\mathbf{D}, INT, \mathcal{P}, S)$ is initiated by the dealer $\mathbf{D}$, who hands over the secret S to *intermediary* INT. In addition, $\mathbf{D}$ hands some **authentication information** to INT and **verification information** to individual players in $\mathcal{P}$, also called as *verifiers*.

2. AuthVal$(\mathbf{D}, INT, \mathcal{P}, S)$ is carried out by INT and the set of verifiers $\mathcal{P}$. Here INT decides whether to continue or abort the protocol depending upon the prediction whether in protocol RevealVal, secret S held by INT will be (eventually) accepted/will be considered as valid by all the (honest) verifiers in $\mathcal{P}$. We denote continuation (resp. abortion) by Auth $= 1$ (resp., 0). If Auth $= 1$, then the **authentication information**, along with S, which is held by INT at the end of AuthVal is called $\mathbf{D}$'s IC signature on S, obtained by INT. We denote the IC signature by $\mathsf{ICSig}_{\mathbf{D},\mathbf{INT}}(S)$.

3. RevealVal$(\mathbf{D}, INT, \mathcal{P}, S)$ is carried out by INT and the verifiers in $\mathcal{P}$. RevealVal can be presented in two flavors: (i) *Private verification* of IC signature by player P_α and (ii) *Public verification* of IC signature by every player in $\mathcal{P}$. In the *private verification* by player P_α, INT sends the IC signature to P_α and the individual verifiers in $\mathcal{P}$ send **verification information** to P_α. With the received information, P_α either accepts S or rejects it. We denote the acceptance, (resp., rejection) by Reveal $= S$ (resp., $NULL$). *Public verification* towards every player in $\mathcal{P}$ is achieved by executing *private verification* for each $P_\alpha \in \mathcal{P}$.

Any IC scheme must satisfy following properties:

1. If $\mathbf{D}$ and INT are honest, then S will be accepted in RevealVal by each honest player.

2. If INT is honest and Auth =1, then S held by INT will be accepted in RevealVal by each honest players, except with probability $2^{-O(\kappa)}$.

3. If $\mathbf{D}$ is honest, then during RevealVal, with probability at least $1 - 2^{-O(\kappa)}$, every $S' \neq S$ produced by a corrupted INT will be rejected by an honest player.

[1]The definition given here is slightly different from the one given in [7], where an IC protocol is executed among $\mathbf{D}$, INT and a single verifier R. However, in the their AWSS and AVSS protocols, the IC protocol is parallely executed with every player in $\mathcal{P}$ acting as verifier.

4. If $\mathbf{D}$ and INT are honest, then at the end of $\mathsf{AuthVal}$, S is information theoretically secure.

We now present an IC protocol called IC, which allows $\mathbf{D}$ to sign on a secret S containing ℓ field elements, with $n \geq 3t+1$. Our IC protocol is motivated by the Information Checking Protocol (with dispute control) presented in [2] in synchronous settings. Though [7] have presented an IC protocol with $n = 3t+1$ in asynchronous settings, it is designed to get signature on only a single secret. In our AVSS protocol, we require to generate IC signature on multiple secrets at once. In that case, using the IC protocol of [7] will result in high communication overhead. On the other hand, using our IC protocol, signature on multiple secrets can be generated at once, with reduced communication complexity. Also, though there exists several IC protocol in synchronous settings [19, 9, 17], the IC protocol of [2] is designed to handle multiple secrets at once and also can be adapted in asynchronous settings efficiently. So we adapt the IC protocol of [2] in asynchronous settings, as shown in Table 1.

$\mathsf{Distr}(\mathbf{D}, INT, \mathcal{P}, S)$: Let $S = (s_1, \ldots, s_\ell)$. For every verifier $P_j \in \mathcal{P}$, $\mathbf{D}$ computes and communicates the following: $\mathbf{D}$ selects uniformly at random κ authentication tags (values) $y_1^j, \ldots, y_\kappa^j \in_R \mathbb{F}^\kappa$, κ elements $u_1^j, \ldots, u_\kappa^j \in_R (\mathbb{F} \backslash \{0, \ldots, \ell\})^\kappa$, and computes $v_1^j, \ldots, v_\kappa^j$ such that for each $i \in \{1, \ldots, \kappa\}$, the $\ell+2$ points $(0, y_i^j), (1, s_1), \ldots, (\ell, s_\ell), (u_i^j, v_i^j)$ lie on a polynomial of degree ℓ. $\mathbf{D}$ sends the authentication tags $y_1^j, \ldots, y_\kappa^j$ to INT and the verification tags $z_1^j = (u_1^j, v_1^j), \ldots, z_\kappa^j = (u_\kappa^j, v_\kappa^j)$ to P_j. Finally, $\mathbf{D}$ sends the secret $S = (s_1, \ldots, s_\ell)$ to INT.

$\mathsf{AuthVal}(\mathbf{D}, INT, \mathcal{P}, S)$:

1. Every verifier P_j randomly partitions the index set $\{1, \ldots, \kappa\}$ into two sets I^j and $\overline{I^j}$ of (almost) equal size, and sends I^j, $\overline{I^j}$ and z_i^j for all $i \in I^j$ to INT.

2. For every verifier P_j, INT checks whether for every $i \in I^j$, the points $(0, y_i^j), (1, s_1), \ldots, (\ell, s_\ell), z_i^j$ lie on a polynomial of degree ℓ. If for at least $2t+1$ verifiers, the above condition is satisfied, then INT sets $\mathsf{Auth} = 1$. and includes these verifiers in a set called $HappySetINT$ ($HappySetP_\beta$ when P_β acts as INT resp.). Otherwise INT sets $\mathsf{Auth} = 0$. If $\mathsf{Auth} = 1$, then the information held by INT is denoted as $\mathsf{ICSig}_{\mathbf{D}, INT}(S)$.

$\mathsf{RevealVal}(\mathbf{D}, INT, \mathcal{P}, S)$

$\mathsf{RevealVal\text{-}Private}(\mathbf{D}, INT, \mathcal{P}, S, P_\alpha)$: *Private verification* of IC signature, by P_α.

1. INT sends the following to player P_α: the secret $S = (s_1, \ldots, s_\ell)$ and the remaining authentication tags y_i^j corresponding to the index set $\overline{I^j}$ (i.e $i \in \overline{I^j}$) along with index set $\overline{I^j}$, for each verifier P_j.
2. To player P_α, every verifier P_j sends the index set $\overline{I^j}$ and all z_i^j such that $i \in \overline{I^j}$.
3. Corresponding to each P_j, player P_α checks whether for **any** $i \in \overline{I^j}$, the points $(0, y_i^j), (1, s_1), \ldots, (\ell, s_\ell), z_i^j$ lie on a polynomial of degree ℓ. If for at least $t+1$ players the above condition is satisfied, then P_α sets $\mathsf{Reveal} = S$. Otherwise he sets $\mathsf{Reveal} = NULL$.

$\mathsf{RevealVal\text{-}Public}(\mathbf{D}, INT, \mathcal{P}, S)$: *Public Verification* of the IC signature by every player in $\mathcal{P}$

1. INT sends the following to every verifier $P_j \in HappySetINT$: the secret $S = (s_1, \ldots, s_\ell)$ and the remaining authentication tags y_i^j corresponding to the index set $\overline{I^j}$ (i.e $i \in \overline{I^j}$) along with index set $\overline{I^j}$.
2. Verifier $P_j \in HappySetINT$ checks whether for **any** $i \in \overline{I^j}$, the points $(0, y_i^j), (1, s_1), \ldots, (\ell, s_\ell), z_i^j$ lie on a polynomial of degree ℓ. If yes P_j outputs $\mathsf{Reveal}_j = S$. Otherwise he sets $\mathsf{Reveal}_j = NULL$. Now P_j sends Reveal_j to every player $P_\alpha \in \mathcal{P}$.
3. Now player $P_\alpha \in \mathcal{P}$ (including players in $HappySetINT$)waits for at least $t+1$ Reveal_j's which are non-NULL and same. Upon receiving the same, P_α sets Reveal_α as one of the Reveal_j's received by him. Otherwise he sets $\mathsf{Reveal}_\alpha = NULL$.

Table 1: Protocol $\mathsf{IC}(\mathbf{D}, INT, \mathcal{P}, S)$

Lemma 1 *If $\mathbf{D}$ and INT are honest, then S will be accepted in RevealVal-Private by every honest P_α.*

Proof(sketch): For an honest $\mathbf{D}$ and honest INT, $\mathsf{Auth} = 1$ at the end of $\mathsf{AuthVal}$ and every (honest) player P_α will eventually output $\mathsf{Reveal} = S$ at the end of $\mathsf{RevealVal}$. □

Lemma 2 *If INT is honest and $\mathsf{Auth} = 1$ at the end of AuthVal, then S held by INT will be accepted in RevealVal-Private by every honest P_α, except with probability $2^{-O(\kappa)}$.*

Proof: Since INT is honest and $\mathsf{Auth} = 1$ at the end of $\mathsf{AuthVal}$, the condition specified in step 2 of $\mathsf{AuthVal}$ is satisfied corresponding to at least $2t+1$ verifiers. Among the $2t+1$ verifiers at least $t+1$ verifiers are honest. Now to prove the above lemma, we prove that corresponding to each of these

honest verifiers, the condition stated in step 3 of RevealVal-Private will be satisfied with very high probability. We first note that corresponding to an honest verifier P_j, the condition stated in step 3 of RevealVal-Private will fail if for all $i \in \overline{I^j}$, the points $(0, y_i^j), (1, s_1), \ldots, (\ell, s_\ell), z_i^j$ does not lie on a polynomial of degree ℓ. This implies $\mathbf{D}$ (which is corrupted in this case) must have distributed y_i^j (to INT) and z_i^j (to P_j) inconsistently for all $i \in \overline{I^j}$ and it so happens that P_j has partitioned $\{1, \ldots, \kappa\}$ into two sets I^j and $\overline{I^j}$ such that $\overline{I_j}$ contains only inconsistent tuples (z_i^j's). Thus corresponding to an honest verifier P_j, the probability that the condition stated in step 3 of RevealVal-Private fails is same as the probability of selecting all consistent tuples in I^j (or selecting all inconsistent tuples in $\overline{I^j}$) which is at most $2^{-O(\kappa)}$. □

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

PROOF: For corrupted INT producing $S' \neq S$ in RevealVal-Private, t corrupted verifiers may produce verification information such that the condition stated in step 3 of RevealVal-Private gets satisfied for all of them. So INT's signature on S' will be validated if the condition stated in step 3 of RevealVal-Private gets satisfied corresponding to at least one honest verifier (in addition to t corrupted verifiers). So we have to analyze: what is the probability that corresponding to an honest verifier P_j, the condition stated in step 3 of RevealVal-Private gets satisfied. The probability of above event is same as the probability of a corrupted INT to guess a verification tag z_i^j for $i \in \overline{I^j}$ (and produce corresponding correct authentication tag) which is at most $\frac{\kappa}{2^{\kappa-\ell-1}} = 2^{-O(\kappa)}$. □

Lemma 4 *If* $\mathbf{D}$ *and* INT *are honest, then at the end of AuthVal,* S *is information theoretically secure from* $\mathcal{A}_t$ *controlling* t *verifiers in* $\mathcal{P}$*.*

PROOF: Trivial, as every verification tuple is statistically independent from the secret. □

Lemma 5 *Protocol Distr, AuthVal and RevealVal-Private communicate* $O((\ell+n\kappa)\kappa)$ *bits each. Protocol RevealVal-public communicate* $O((\ell n^2 + n^2\kappa)\kappa)$ *bits.*

LINEARITY PROPERTY OF IC PROTOCOL: A very important property of our IC protocol IC is that it generates an IC signature satisfying *linearity* property. By *linearity* property, we mean that if INT has got signatures (from $\mathbf{D}$) on $S^1 = (s_1^1, \ldots, s_\ell^1)$ and $S^2 = (s_1^2, \ldots, s_\ell^2)$ separately in two different instances of IC, then it is possible for INT to generate (and produce) a signature on linear combination of the secrets $r_1S^1 + r_2S^2 = (r_1s_1^1 + r_2s_1^2, \ldots, r_1s_\ell^1 + r_2s_\ell^2)$ without further communications. Similarly the set of verifiers can also adjust their verification tags accordingly. This is clearly possible subject to the following two conditions: (a) If for both the instances of IC, the κ random elements $u_1^j, \ldots, u_\kappa^j$ chosen (by $\mathbf{D}$) for verifier P_j remains same, and (b) INT must complete AuthVal for both the set of secrets with at least $2t+1$ verifiers in common in $HappySetINT$. Subject to the above two conditions, INT has to do the appropriate linear combination of the secrets and authentication tags to obtain the new signature and the verifiers can do the same linear combination on the v components of verification tuples z to obtain new verification tuples. The intuition for condition (b) is very clear. If INT completes AuthVal for first set of secrets with $HappySetINT^1$ and for the second set of secrets with $HappySetINT^2$ such that $|HappySetINT^1 \cap HappySetINT^2| \leq 2t$, then it may be possible that out of these $2t$ players, t are corrupted, who can purposely send incorrect verification tag(s) during RevealVal-Private. And the honest players which are not in $HappySetINT^1 \cap HappySetINT^2$ can not compute the new verification tag corresponding to the linear combination of the secrets. Thus even if condition (a) is satisfied, the new signature computed by an honest INT may be rejected in RevealVal-Private (or RevealVal-Public). Also notice that if $\mathbf{D}$ is honest then condition (b) will be eventually satisfied. The linearity property of the IC protocol can be extended for any linear combination of any number of secrets, provided the above two conditions are satisfied. Our AVSS protocol presented in this paper uses the linearity property of IC protocol.

5 Asynchronous Weak Secret Sharing

In this section, we present a novel AWSS protocol called AWSS with $n = 3t + 1$ to share a secret S containing ℓ field elements from $\mathbb{F}$. The reconstruction of AWSS is presented in two flavors: private reconstruction and public reconstruction. While in private reconstruction, only a specific player is allowed to do reconstruction with the help of all the players, public reconstruction allows every player to perform reconstruction. For ease of understanding, we first present an AWSS protocol, called AWSS-Single-Secret which shares a single secret s and then prove that AWSS-Single-Secret satisfies all the properties of an AWSS protocol. Later we show how to extend protocol AWSS-Single-Secret to obtain our AWSS protocol AWSS for sharing ℓ secrets at once.

AWSS-Single-Secret$(\mathbf{D}, \mathcal{P}, s)$

AWSS-Single-Secret-Share$(\mathbf{D}, \mathcal{P}, s)$:

1. Distribution - Code for $\mathbf{D}$: $\mathbf{D}$ selects a degree-t symmetric bivariate polynomial $F(x, y)$ such that $F(0, 0) = s$ and delivers $f_i(x) = F(x, i)$ to P_i with his authentication tags by initiating Distr$(\mathbf{D}, P_i, \mathcal{P}, f_i(j))$ for every $j \in \{1, \ldots, n\}$.

2. Code for P_i:

   1. Player P_i waits until Distr$(\mathbf{D}, P_i, \mathcal{P}, f_i(j))$ is completed for every $j \in \{1, \ldots, n\}$. Then P_i and players in $\mathcal{P}$ executes AuthVal$(\mathbf{D}, P_i, \mathcal{P}, f_i(j))$ for every $j \in \{1, \ldots, n\}$.
   2. If P_i completes AuthVal$(\mathbf{D}, P_i, \mathcal{P}, f_i(j))$ with Auth $= 1$ for every $j \in \{1, \ldots, n\}$, he hands over $f_i(j)$ to P_j with his signature by initiating Distr$(P_i, P_j, \mathcal{P}, f_i(j))$ for all $j \in \{1, \ldots, n\}$. In addition, P_i participates in Distr$(P_j, P_i, \mathcal{P}, f_j(i))$ for all $j \in \{1, \ldots, n\}$.
   3. P_i waits until Distr$(P_j, P_i, \mathcal{P}, f_j(i))$ is completed. Then P_i checks whether $f_i(j) \stackrel{?}{=} f_j(i)$. If yes then P_i and players in $\mathcal{P}$ executes AuthVal$(P_j, P_i, \mathcal{P}, f_j(i))$.
   4. If P_i completes AuthVal$(P_j, P_i, \mathcal{P}, f_j(i))$ with Auth $= 1$, then P_i A-casts OK(P_i, P_j).

3. CORE Construction:

   1. $\mathbf{D}$ maintains a set $OKSetP_j = \{P_i | \mathbf{D}$ *listens* OK$(P_i, P_j)\}$. If $|OKSetP_j| \geq n - t$, then $\mathbf{D}$ adds P_j in $CORE$ (which is initially empty).
   2. $\mathbf{D}$ waits until $|CORE| \geq n - t$. Then $\mathbf{D}$ A-casts the set $CORE$ and current $OKSetP_j$ for all $P_j \in CORE$.
   3. Once a player P_i listens $CORE'$ and $OKSetP'_j$ for all $P_j \in CORE'$ from $\mathbf{D}$, he himself waits to listen OK(P_k, P_j) for all $P_k \in OKSetP'_j$ and $P_j \in CORE'$. Once P_i listens all the required OK's, he A-casts $CORE'$ and $OKSetP'_j$ for all $P_j \in CORE'$.
   4. Now P_i waits to listen same $CORE''$ and $OKSetP''_j$ for all $P_j \in CORE''$ from at least $n - t$ players' A-cast. Upon listening, he sets $CORE = CORE''$ and $OKSetP_j = OKSetP''_j$ for all $P_j \in CORE$ and terminates.

AWSS-Single-Secret-Rec-Private$(\mathbf{D}, \mathcal{P}, s, P_\alpha)$: Private Reconstruction towards P_α:

1. For every $P_j \in CORE$ and every $P_k \in OKSetP_j$, every player participates in RevealVal-Private$(\mathbf{D}, P_k, \mathcal{P}, f_k(j), P_\alpha)$ and RevealVal-Private$(P_j, P_k, \mathcal{P}, f_j(k), P_\alpha)$. /*Thus each $P_k \in OKSetP_j$ reveals $\mathbf{D}$'s and P_j's IC signature on $f_k(j)$ and $f_j(k)$ respectively.*/

2. For every $P_j \in CORE$, P_α constructs a set $ValidSetP_j$. P_α adds a player $P_k \in OKSetP_j$ to $ValidSetP_j$ if P_α completes RevealVal-Private$(\mathbf{D}, P_k, \mathcal{P}, f_k(j), P_\alpha)$ and RevealVal-Private$(P_j, P_k, \mathcal{P}, f_j(k), P_\alpha)$ with Reveal $= \overline{f_k(j)}$ and Reveal $= \overline{f_j(k)}$ respectively and $\overline{f_k(j)} = \overline{f_j(k)}$ is satisfied. P_α waits until $|ValidSetP_j| = n - 2t = t + 1$. Once this is satisfied, P_α constructs a polynomial $\overline{f_j(x)}$ passing through the points $(k, \overline{f_j(k)})$ where $P_k \in ValidSetP_j$. If $\overline{f_j(x)}$ is of degree t, then P_α associates $\overline{f_j(x)}$ with player $P_j \in CORE$. Otherwise P_α associates $NULL$ with P_j.

3. If for at least one $P_j \in CORE$, the associated polynomial is $NULL$, then P_α reconstructs $\overline{s} = NULL$ and terminates. Other wise P_α waits for all $P_j \in CORE$ to be associated with some polynomial $\overline{f_j(x)}$. Then for every pair $(P_\gamma, P_\delta) \in CORE$, P_α checks whether $\overline{f_\gamma(\delta)} \stackrel{?}{=} \overline{f_\delta(\gamma)}$. If the condition is satisfied for every pair in $CORE$, P_α reconstructs the bivariate polynomial $\overline{F(x, y)}$ using the polynomials $\overline{f_j(x)}$ associated with $P_j \in CORE$ and gets $\overline{s} = \overline{F(0, 0)}$ and terminates. Otherwise again P_α reconstructs $\overline{s} = NULL$ and terminates.

AWSS-Single-Secret-Rec-Public$(\mathbf{D}, \mathcal{P}, s, \mathcal{P})$: Public Reconstruction towards $\mathcal{P}$:

1. Same as in AWSS-Single-Secret-Rec-Private, except that every RevealVal-Private in AWSS-Single-Secret-Rec-Private is replaced by RevealVal-Public. Now every player $P_\alpha \in \mathcal{P}$ proceeds in the same way as in AWSS-Single-Secret-Rec-Private and reconstructs $\overline{s}$.

Protocol AWSS-Single-Secret uses the bivariate polynomial approach of Feldman [11]. The high level description of AWSS-Single-Secret is as follows: $\mathbf{D}$ selects a degree-t symmetric bivariate polynomial

$F(x,y)$ such that $F(0,0) = s$ and delivers $f_i(x) = F(x,i)$ to P_i with his authentication tags by initiating $\mathsf{Distr}(\mathbf{D}, P_i, \mathcal{P}, f_i(j))$ for every $j \in \{1,\ldots,n\}$. Once player P_i completes $\mathsf{AuthVal}(\mathbf{D}, P_i, \mathcal{P}, f_i(j))$ with $\mathsf{Auth} = 1$ for every $j =\in \{1,\ldots,n\}$, he hands over $f_i(j)$ to P_j (by the property of symmetric bivariate polynomial $f_i(j) = f_j(i)$) with his signature by executing $\mathsf{Distr}(P_i, P_j, \mathcal{P}, f_i(j))$ for every $j \in \{1,\ldots,n\}$. Once player P_i verifies $f_i(j) \stackrel{?}{=} f_j(i)$ and completes $\mathsf{AuthVal}(P_j, P_i, \mathcal{P}, f_j(i))$ with $\mathsf{Auth} = 1$, he $\mathsf{A\text{-}casts}\ \mathsf{OK}(P_i, P_j)$. This indicates that P_i has verified that $f_i(j) = f_j(i)$. Moreover, P_i has $\mathbf{D}$'s valid signature on $f_i(j)$, as well as P_j's valid signature on $f_j(i) = f_i(j)$.

Once $\mathbf{D}$ listens at least $n-t$ $\mathsf{A\text{-}casts}$ of the form $\mathsf{OK}(P_i, P_j)$ for P_j, $\mathbf{D}$ includes P_j in a set $CORE$ (initially $CORE = \emptyset$). $\mathbf{D}$ also maintains a set $OKSetP_j = \{P_i | \mathbf{D}\ listens\ \mathsf{OK}(P_i, P_j)\}$. So for $P_j \in CORE$, $|OKSetP_j| \geq n-t$. Once $CORE \geq n-t$, $\mathbf{D}$ $\mathsf{A\text{-}casts}$ the set $CORE$ and current $OKSetP_j$ for all $P_j \in CORE$. Notice that an honest $\mathbf{D}$ will always end up with such a $CORE$ set. Now each honest player P_i waits to listen a $CORE$ set from $\mathbf{D}$ of size $n-t$, with $OKSetP_j$ of size $n-t$ for each P_j in $CORE$. Once a player P_i listens $CORE'$ and $OKSetP'_j$ for all $P_j \in CORE'$ from $\mathbf{D}$, he tries to check the validity of the received information. For this, P_i himself waits to listen $\mathsf{OK}(P_k, P_j)$ for all $P_k \in OKSetP'_j$ and $P_j \in CORE'$. Once P_i listens all the required OK's, he $\mathsf{A\text{-}casts}$ $CORE'$ and $OKSetP'_j$ for all $P_j \in CORE'$.

However, in our protocol, we require that the honest players should agree on the same $CORE$ and corresponding $OKSet$, as sent by $\mathbf{D}$. For this, every (honest) P_i waits to listen *same* $CORE''$ and $OKSetP''_j$ for all $P_j \in CORE''$ from the $\mathsf{A\text{-}casts}$ of at least $n-t$ players. Upon listening, he sets $CORE = CORE''$ and $OKSetP_j = OKSetP''_j$ for all $P_j \in CORE$ and terminates the sharing phase of $\mathsf{AWSS\text{-}Single\text{-}Secret}$. Since there are $n = 3t+1$ players, the above steps ensure that two different honest players can not have different $CORE$ and corresponding $OKSet$s. Thus if sharing phase terminates then all honest players must have agreed on $CORE$ as well as $OKSetP_j$ for all $P_j \in CORE$. Then in the reconstruction phase irrespective of whether $\mathbf{D}$ is honest or not, for every honest $P_i \in CORE$, the $f_i(x)$ polynomial will be reconstructed correctly. For honest $\mathbf{D}$, correct $f_i(x)$ polynomial will be reconstructed for a corrupted $P_i \in CORE$ as well. But if $\mathbf{D}$ is dishonest, then for a corrupted $P_i \in CORE$ a wrong $\overline{f_i(x)}$ may get reconstructed. Thus for an honest $\mathbf{D}$, correct secret s will be recovered (since bivariate $F(x,y)$ can be reconstructed) with very high probability. But for a corrupted $\mathbf{D}$ either the committed secret or $NULL$ will be recovered. Note that in our AWSS protocol if $\mathbf{D}$ is corrupted then it is possible that some honest player reconstructs the committed secret while some other honest player reconstructs $NULL$.

Lemma 6 *Protocol AWSS-Single-Secret satisfies termination property.*

Proof: Termination 1: When $\mathbf{D}$ is honest, every honest P_i will complete $\mathsf{AuthVal}(\mathbf{D}, P_i, \mathcal{P}, f_i(j))$ with $\mathsf{Auth} = 1$. Also for every pair of honest players P_i, P_j, $\mathsf{AuthVal}(P_i, P_j, \mathcal{P}, f_i(j))$ and $\mathsf{AuthVal}(P_j, P_i, \mathcal{P}, f_j(i))$ will be successful. Hence $\mathbf{D}$ will eventually listen $n-t = 2t+1$ $\mathsf{OK}(P_i, P_j)$ for every honest P_j. Since there are at least $2t+1$ honest players in $\mathcal{P}$, $\mathbf{D}$ gets $CORE$ of size at least $n-t$ and $\mathsf{A\text{-}casts}$ $CORE$ and $OKSetP_j$ for every $P_j \in CORE$. Since $\mathbf{D}$ is honest every honest P_i eventually listens $CORE$ and $OKSetP_j$ for every $P_j \in CORE$ from $\mathbf{D}$. Again by the property of $\mathsf{A\text{-}cast}$, since (honest) $\mathbf{D}$ has listened $\mathsf{OK}(P_k, P_j)$ for all $P_j \in CORE$ and $P_k \in OKSetP_j$, all other honest players will also eventually listen them. Finally since all the honest players $\mathsf{A\text{-}casts}$ same $CORE$ and $OKSetP_j$'s, an honest player will eventually listen at least $n-t = 2t+1$ $\mathsf{A\text{-}casts}$ of same $CORE$ and $OKSetP_j$'s. Thus every honest player will terminate $\mathsf{AWSS\text{-}Single\text{-}Secret\text{-}Share}$.

Termination 2: If an honest player P_i has completed $\mathsf{AWSS\text{-}Single\text{-}Secret\text{-}Share}$, then he must have listened at least $n-t = 2t+1$ $\mathsf{A\text{-}casts}$ of same $CORE$ and $OKSetP_j$'s. By the definition of $\mathsf{A\text{-}cast}$, each honest player will also listen the same. Since there are total $n = 3t+1$ players, there can not be two different version of $CORE$ and $OKSetP_j$'s. Hence, every honest player will eventually terminate $\mathsf{AWSS\text{-}Single\text{-}Secret\text{-}Share}$.

Termination 3: By Lemma 2, if P_i (acting as INT) is honest and $\mathsf{Auth} = 1$ at the end of $\mathsf{AuthVal}$, then P_i's secret (with signature given by any dealer) will be accepted in $\mathsf{RevealVal\text{-}Private}$, except with probability $2^{-O(\kappa)}$. Since for every $P_j \in CORE$, $|OKSetP_j| \geq n-t$, there are at least $t+1$ honest players in $OKSetP_j$ who will be present in $ValidSetP_j$ of P_α with very high probability. Hence

$|ValidSetP_j| \geq n - 2t \geq t + 1$. Thus every honest P_α will eventually terminate AWSS-Single-Secret-Rec-Private$(\mathbf{D}, \mathcal{P}, s, P_\alpha)$ after executing the remaining steps specified in the same. Similarly, we can prove that every $P_\alpha \in \mathcal{P}$ will terminate AWSS-Single-Secret-Rec-Public eventually. □

Lemma 7 *Protocol AWSS-Single-Secret satisfies secrecy property.*

PROOF: Follows from the secrecy of IC and properties of symmetric bivariate polynomial. □

Lemma 8 *Protocol AWSS-Single-Secret satisfies correctness property.*

PROOF: CORRECTNESS 1: We first prove that if $\mathbf{D}$ is honest, then the polynomial $\overline{f_j(x)}$ associated with $P_j \in CORE$ (by an honest P_α) is same as the polynomial $f_j(x)$ distributed by $\mathbf{D}$. From the property of IC protocol, for an honest $P_j \in CORE$, a corrupted $P_k \in OKSetP_j$ can produce a valid signature on $\overline{f_j(k)} \neq f_j(k)$ with negligible probability (see Lemma 3). Hence with very high probability $\overline{f_j(k)}$ is same as $f_j(k)$ for all $k \in ValidSetP_j$. Thus the polynomial associated with P_j is $f_j(x)$ distributed by honest $\mathbf{D}$. For a corrupted $P_j \in CORE$, a corrupted $P_k \in OKSetP_j$ can produce a valid signature on $\overline{f_j(k)} \neq f_j(k)$ but P_k will fail to produce $\mathbf{D}$'s signature on $\overline{f_k(j)} = \overline{f_j(k)}$ for an honest $\mathbf{D}$. Hence P_k will not be included in $ValidSetP_j$. Hence again the polynomial associated with P_j is $f_j(x)$ distributed by honest $\mathbf{D}$. So P_α will reconstruct $F(x, y)$ and hence the secret $s = F(0, 0)$ with very high probability.

CORRECTNESS 2: Since in AWSS-Single-Secret-Share, every honest player agrees on a common $CORE$ and $OKSetP_j$ for $P_j \in CORE$, an unique secret $s' \in \mathbb{F} \cup NULL$ is defined by the $(n-t)-t = n-2t \geq (t+1)$ honest players in $CORE$. If for every two *honest* players P_γ and P_δ in $CORE$, $f_\gamma(\delta) = f_\delta(\gamma)$ holds then s' is the secret defined by the bivariate polynomial $F'(x, y)$, induced by the polynomials of honest players. Otherwise $\mathbf{D}$'s committed secret $s' = NULL$. Let $\mathbf{D}$ has committed $s' \in \mathbb{F}$. Now we show that an honest player P_α either reconstructs s' or $NULL$. To prove this, we first claim that the polynomial $\overline{f_j(x)}$ associated with an *honest* $P_j \in CORE$ (by P_α) is same as the polynomial $f_j(x)$ distributed by $\mathbf{D}$. This claim follows from the argument given in CORRECTNESS 1. But now for a corrupted P_j, a corrupted $P_k \in OKSetP_j$ can produce a valid signature of P_j on $\overline{f_j(k)} \neq f_j(k)$ as well as a valid signature of $\mathbf{D}$ (who is corrupted in this case) on $\overline{f_k(j)} = \overline{f_j(k)}$. Also adversary can delay the messages such that the values of all corrupted $P_k \in OKSetP_j$'s reach P_α before the values of honest players in $OKSetP_j$. Thus the polynomial associated with a corrupted P_j can be $\overline{f_j(x)} \neq f_j(x)$. Thus the polynomials corresponding to all the players in $CORE$ will not define a valid bivariate polynomial $F'(x, y)$. Thus $NULL$ will be reconstructed by P_α. Notice that since all honest players of $CORE$ are associated with original $f_j(x)$, no other secret (other than s') can be reconstructed.

Now let $\mathbf{D}$'s committed secret $s' = NULL$. In this case irrespective of the behavior of corrupted players, $NULL$ will be reconstructed. □

We now extend protocol AWSS-Single-Secret (which shares a single secret) to protocol AWSS (given in Table 5) which shares ℓ secrets at once. The properties of AWSS follows from the properties of AWSS-Single-Secret.

Lemma 9 *Protocol AWSS-Share communicates $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits and A-casts $\mathcal{O}(n^3 \log(n))$ bits. Protocol AWSS-Rec-Private and AWSS-Rec-Public communicates $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ and $\mathcal{O}((\ell n^4 + n^4\kappa)\kappa)$ bits, respectively.*

PROOF: Protocol AWSS-Share executes at most $n + n^2 = \Theta(n^2)$ instances of Distr and AuthVal. Hence by Lemma 5, AWSS-Share communicates $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits. Also every player A-casts $CORE$ and $OKSetP_j$'s which contain $\mathcal{O}(n^2)$ numbers from $\{1, \ldots, n\}$, each of which can be represented by $\log(n)$ bits. Protocol AWSS-Rec-Private executes $\Theta(n^2)$ instances of RevealVal-Private and hence by Lemma 5, AWSS-Rec-Private communicates $\mathcal{O}((\ell n^2 + n^3\kappa)\kappa)$ bits. Protocol AWSS-Rec-Public executes $\Theta(n^2)$ instances of RevealVal-Public and hence by Lemma 5, AWSS-Rec-Public communicates $\mathcal{O}((\ell n^4 + n^4\kappa)\kappa)$ bits. □

Before ending this section, we present another interesting protocol called AWSS-Share-MultiSet, which shares $\mathcal{M}$ sets of ℓ secrets at once such that later any linear combination of the $\mathcal{M}$ sets can be

AWSS($\mathbf{D}, \mathcal{P}, S$): Protocol to Share $S = [s^1\ s^2\ \ldots\ s^\ell]$

AWSS-Share($\mathbf{D}, \mathcal{P}, S$):

1. Distribution - Code for $\mathbf{D}$: $\mathbf{D}$ selects ℓ degree-t symmetric bivariate polynomials $F^1(x,y), \ldots, F^\ell(x,y)$ such that for $l = 1, \ldots, \ell$, $F^l(0,0) = s^l$. He then delivers $f_i^l(x) = F^l(x,i)$ for $l = 1, \ldots, \ell$ to P_i with his authentication tags by initiating Distr($\mathbf{D}, P_i, \mathcal{P}, (f_i^1(j), \ldots, f_i^\ell(j))$) for every $j \in \{1, \ldots, n\}$.

2. Code for P_i:

   1. Player P_i waits until Distr($\mathbf{D}, P_i, \mathcal{P}, (f_i^1(j), \ldots, f_i^\ell(j))$) for every $j \in \{1, \ldots, n\}$ are completed. Then P_i and players in $\mathcal{P}$ executes AuthVal($\mathbf{D}, P_i, \mathcal{P}, (f_i^1(j), \ldots, f_i^\ell(j))$) for every $j \in \{1, \ldots, n\}$.
   2. If P_i completes AuthVal($\mathbf{D}, P_i, \mathcal{P}, (f_i^1(j), \ldots, f_i^\ell(j))$) with Auth $= 1$ for every $j \in \{1, \ldots, n\}$, he hands over $(f_i^1(j), \ldots, f_i^\ell(j))$ to P_j with authentication tags by initiating Distr($P_i, P_j, \mathcal{P}, (f_i^1(j), \ldots, f_i^\ell(j))$) for all $j \in \{1, \ldots, n\}$. In addition, P_i participates in Distr($P_j, P_i, \mathcal{P}, (f_j^1(i), \ldots, f_j^\ell(i))$) for every $j \in \{1, \ldots, n\}$.
   3. P_i waits until Distr($P_j, P_i, \mathcal{P}, (f_j^1(i), \ldots, f_j^\ell(i))$) is completed. Then P_i checks whether $f_i^l(j) \stackrel{?}{=} f_j^l(i)$ for $l = 1, \ldots, \ell$. If yes then P_i and players in $\mathcal{P}$ executes AuthVal($P_j, P_i, \mathcal{P}, (f_j^1(i), \ldots, f_j^\ell(i))$).
   4. If P_i completes AuthVal($P_j, P_i, \mathcal{P}, (f_j^1(i), \ldots, f_j^\ell(i))$) with Auth $= 1$, then he A-casts OK(P_i, P_j).

3. CORE Construction: This is identically same as in Protocol AWSS-Single-Secret-Share.

AWSS-Rec-Private($\mathbf{D}, \mathcal{P}, S, P_\alpha$): Private Reconstruction towards P_α:

1. For every $P_j \in CORE$ and every $P_k \in OKSetP_j$, every player initiates RevealVal-Private($\mathbf{D}, P_k, \mathcal{P}, (f_k^1(j), \ldots, f_k^\ell(j)), P_\alpha$) and RevealVal-Private($P_j, P_k, \mathcal{P}, (f_j^1(k), \ldots, f_j^\ell(k)), P_\alpha$).

2. For every $P_j \in CORE$, P_α constructs a set $ValidSetP_j$. P_α adds a player $P_k \in OKSetP_j$ to $ValidSetP_j$ if P_α completes RevealVal-Private($\mathbf{D}, P_k, \mathcal{P}, (f_k^1(j), \ldots, f_k^\ell(j)), P_\alpha$) and RevealVal-Private($P_j, P_k, \mathcal{P}, (f_j^1(k), \ldots, f_j^\ell(k)), P_\alpha$) with Reveal $= \overline{(f_k^1(j), \ldots, f_k^\ell(j))}$ and Reveal $= \overline{(f_j^1(k), \ldots, f_j^\ell(k))}$ respectively and for each $l = 1, \ldots, \ell$, $\overline{f_k^l(j) = f_j^l(k)}$ is satisfied. P_α waits until $|ValidSetP_j| = n - 2t = t + 1$. Once this is satisfied, P_α constructs polynomials $\overline{f_j^l(x)}$ passing through the points $(k, \overline{f_j^l(k)})$ such that $P_k \in ValidSetP_j$. If for all $l = 1, \ldots, \ell$, $\overline{f_j^l(x)}$ is of degree t, then P_j associates all $\overline{f_j^l(x)}$'s with player $P_j \in CORE$. Otherwise associate $NULL$ with P_j.

3. If for at least one $P_j \in CORE$, the associated polynomial is $NULL$, then P_α reconstructs $\overline{S} = NULL$ and terminates. Other wise P_α waits for each $P_i \in CORE$ to be associated with a set of ℓ polynomials $\overline{f_i^l(x)}$. Then for every pair $(P_\gamma, P_\delta) \in CORE$, P_α checks whether $\overline{f_\gamma^l(\delta)} \stackrel{?}{=} \overline{f_\delta^l(\gamma)}$ for $l = 1, \ldots, \ell$. If the condition is satisfied for every pair of players in $CORE$, P_α reconstructs bivariate polynomials $\overline{F^1(x,y)}, \ldots, \overline{F^\ell(x,y)}$ from the univariate polynomials associated with the players in $CORE$ and gets $\overline{s_l} = \overline{F^l(0,0)}$. Otherwise again P_α reconstructs $\overline{S} = NULL$.

AWSS-Rec-Public($\mathbf{D}, \mathcal{P}, S, \mathcal{P}$): Private Reconstruction towards $\mathcal{P}$:

1. Same as in AWSS-Rec-Private, except that every RevealVal-Private in AWSS-Rec-Private is replaced by RevealVal-Public. Now every player $P_\alpha \in \mathcal{P}$ proceeds in the same way as in AWSS-Rec-Private and reconstructs $\overline{S}$.

Table 2: AWSS($\mathbf{D}, \mathcal{P}, S$): Protocol to Share $S = [s^1\ s^2\ \ldots\ s^\ell]$

reconstructed. The purpose of presenting this protocol will be clear in the next section. More specifically, AWSS-Share-MultiSet can share $S^m = (s^{(m,1)}, \ldots, s^{(m,\ell)})$ for $m = 1, \ldots, \mathcal{M}$ at once such that later $S = \sum_{m=1}^{\mathcal{M}} r_m S^m = (\sum_{m=1}^{\mathcal{M}} r_m s^{(m,1)}, \ldots, \sum_{m=1}^{\mathcal{M}} r_m s^{(m,\ell)})$ (or NULL, if $\mathbf{D}$ is corrupted) can be reconstructed using Protocol AWSS-Rec-Private or AWSS-Rec-Public for some $(r_1, \ldots, r_{\mathcal{M}}) \in \mathbb{F}^{\mathcal{M}}$. Protocol AWSS-Share-MultiSet goes in the same line of AWSS-Share, with the following restrictions to be ensured. In Code for P_i of AWSS-Share, P_i must ensure that he completes AuthVal($\mathbf{D}, P_i, \mathcal{P}, (f_i^1(j), \ldots, f_i^\ell(j))$) for all the $\mathcal{M}$ set of secrets and have $2t+1$ common verifiers in each corresponding $HappySetP_i$. Similarly, the same must be ensured for AuthVal($P_j, P_i, \mathcal{P}, (f_j^1(i), \ldots, f_j^\ell(i))$) for all the $\mathcal{M}$ set of secrets. The intuition of these steps are very clear from the idea given in Linearity Property of IC.

It is easy to see that AWSS-Share-MultiSet satisfies termination and secrecy properties. The correctness follows from the correctness of AWSS and the linearity of IC and bivariate polynomials.

AWSS-Share-MultiSet($\mathbf{D}, \mathcal{P}, S^1, \ldots, S^{\mathcal{M}}$)

1. Distribution - Code for $\mathbf{D}$: $\mathbf{D}$ executes CODE for $\mathbf{D}$ of AWSS-Share for every set S^m. Specifically, for every set S^m, $\mathbf{D}$ does the following: $\mathbf{D}$ selects ℓ degree-t symmetric bivariate polynomials $F^{(m,1)}(x,y), \ldots, F^{(m,\ell)}(x,y)$ such that for $l = 1, \ldots, \ell$, $F^{(m,l)}(0,0) = s^{(m,l)}$. He then delivers $f_i^{(m,l)}(x) = F^{(m,l)}(x,i)$ for $l = 1, \ldots, \ell$ to P_i with his authentication tags by initiating Distr($\mathbf{D}, P_i, \mathcal{P}, (f_i^{(m,1)}(j), \ldots, f_i^{(m,\ell)}(j))$) for every $j \in \{1, \ldots, n\}$.

2. Code for P_i:

   1. For every set S^m, player P_i waits until Distr($\mathbf{D}, P_i, \mathcal{P}, (f_i^{(m,1)}(j), \ldots, f_i^{(m,\ell)}(j))$) for every $j \in \{1, \ldots, n\}$ are completed. Then P_i and players in $\mathcal{P}$ executes AuthVal($\mathbf{D}, P_i, \mathcal{P}, (f_i^{(m,1)}(j), \ldots, f_i^{(m,\ell)}(j))$) for every $j \in \{1, \ldots, n\}$.
   2. P_i waits to complete AuthVal($\mathbf{D}, P_i, \mathcal{P}, (f_i^{(m,1)}(j), \ldots, f_i^{(m,\ell)}(j))$) for all $m \in \{1, \ldots, \mathcal{M}\}$ with Auth $= 1$ such that $HappySetP_i$ for all the $\mathcal{M}$ instances have common $n - t$ verifiers. If P_i completes the above for every $j \in \{1, \ldots, n\}$, then he hands over $(f_i^{(m,1)}(j), \ldots, f_i^{(m,\ell)}(j))$ to P_j with authentication tags by initiating Distr($P_i, P_j, \mathcal{P}, (f_i^{(m,1)}(j), \ldots, f_i^{(m,\ell)}(j))$). In addition, he participates in Distr($P_j, P_i, \mathcal{P}, (f_j^{(m,1)}(i), \ldots, f_j^{(m,\ell)}(i))$) for all $m \in \{1, \ldots, \mathcal{M}\}$ and $j \in \{1, \ldots, n\}$.
   3. P_i waits until Distr($P_j, P_i, \mathcal{P}, (f_j^{(m,1)}(i), \ldots, f_j^{(m,\ell)}(i))$) is completed. Then P_i checks whether $f_i^{(m,l)}(j) \stackrel{?}{=} f_j^{(m,l)}(i)$ for $l = 1, \ldots, \ell$ and $m = 1, \ldots, \mathcal{M}$. If yes then P_i and players in $\mathcal{P}$ executes AuthVal($P_j, P_i, \mathcal{P}, (f_j^{(m,1)}(i), \ldots, f_j^{(m,\ell)}(i))$).
   4. If P_i completes AuthVal($P_j, P_i, \mathcal{P}, (f_j^{(m,1)}(i), \ldots, f_j^{(m,\ell)}(i))$) for all $m \in \{1, \ldots, \mathcal{M}\}$ with Auth $= 1$ such that $HappySetP_i$ for all the $\mathcal{M}$ instances have common $n - t$ verifiers, then he A-casts OK(P_i, P_j).

3. CORE Construction: This is identically same as in Protocol AWSS-Share($\mathbf{D}, \mathcal{P}, s$).

Lemma 10 *AWSS-Share-Multiset communicates* $\mathcal{O}((\ell n^2 + n^3\kappa)\mathcal{M}\kappa)$ *bits and A-casts* $\mathcal{O}(n^3 \log(n))$ *bits.*

6 Asynchronous Verifiable Secret Sharing

In this section, we present our novel AVSS scheme called AVSS to share a secret containing ℓ field elements from $\mathbb{F}$; i.e., $S \in \mathbb{F}^\ell$. As in the case of AWSS, the reconstruction of AVSS is also presented in two flavors: private reconstruction and public reconstruction. Prior to the presentation of our AVSS protocol, we first present a protocol called AVSS-Weak which allows a $\mathbf{D}$ to commit some secret $S \in \mathbb{F}^\ell \cup NULL$ (in a sense explained in the sequel). Then we incorporate a *verification step* in AVSS-Weak to obtain AVSS protocol, where the honest players terminate their sharing phase, only after ensuring that $\mathbf{D}$ has committed secret $S \in \mathbb{F}^\ell$ (not $NULL$).

We explain the idea used in AVSS-Weak with a single secret s shared by $\mathbf{D}$. The modifications required to share multiple secrets at once will follow. The idea of AVSS-Weak is as follows: $\mathbf{D}$ hides the secret as the constant term of a symmetric degree-t bivariate polynomial $F(x,y)$ and then distribute the univariate polynomial $f_i(x) = F(x,i)$ to player P_i. Now each player P_i acting as a dealer initiates an instance of AWSS protocol by selecting a degree-t symmetric bivariate polynomial $F^{P_i}(x,y)$, such that $f_0^{P_i}(x) = F^{P_i}(x,0) = f_i(x)$. By doing so, P_i facilitates player P_j to check the equality of the shares $f_i(j)$ and $f_j(i)$ (ideally for an honest $\mathbf{D}$, $f_i(j) = f_j(i)$ should hold) as follows: P_j on receiving $f_j^{P_i}(x) = F^{P_i}(x,j)$ from P_i (as part of the AWSS instance initiated by P_i) checks whether $f_j^{P_i}(0) \stackrel{?}{=} f_j(i)$. This is because ideally $f_j^{P_i}(0) = F^{P_i}(0,j)$ should be same as $F^{P_i}(j,0)$ since $F^{P_i}(x,y)$ is symmetric. And the way P_i has selected $F^{P_i}(x,y)$, $F^{P_i}(j,0) = f_i(j)$ should hold. So if $f_j^{P_i}(0) \stackrel{?}{=} f_j(i)$ fails then P_j can guess that either $\mathbf{D}$ is corrupted or P_i is corrupted. Now player P_j proceed further to participate in the AWSS instance initiated by P_i only if the above consistency check passes. Now $\mathbf{D}$ waits for the termination of AWSS instance of at least $n - t = 2t + 1$ players and include them in a set $TCORE$. $\mathbf{D}$ keeps on including new players whose instance of AWSS is terminated in $TCORE$ until he gets a subset of player $CORE \subseteq TCORE$ with $|CORE| \geq n - t$, such that for each $P_i \in CORE$, the set $|CORE \cap CORE^{P_i}| \geq n - t$, where $CORE^{P_i}$ is the $CORE$ set corresponding to the instance of AWSS initiated by P_i. Note that for an honest $\mathbf{D}$ this will eventually happen. $\mathbf{D}$ then A-casts his version of $CORE$ and for each $P_i \in CORE$, the set $CORE^{P_i}$ and the corresponding sets $OKSetP_j^{P_i}$ for each $P_j \in CORE^{P_i}$. Now using the same procedure as in AWSS protocol, all the honest players agree on *common* $CORE$, the set $CORE^{P_i}$ for each $P_i \in CORE$ and the corresponding sets $OKSetP_j^{P_i}$ for each $P_j \in CORE^{P_i}$. Note that for an honest $\mathbf{D}$, this will happen eventually.

We say that **D**'s committed secret is the constant term of the symmetric degree-t bivariate polynomial $\overline{F(x,y)}$ which is induced by the univariate polynomials $f_i(x) = F^{P_i}(x,0)$ of the honest players in $CORE$. If **D** is honest, then $\overline{F(x,y)} = F(x,y)$. But if **D** is corrupted, then $\overline{F(x,y)}$ may or may not be symmetric degree-t bivariate polynomial. We say that **D**'s committed secret is *meaningful* (resp. $NULL$) if $\overline{F(x,y)}$ is (resp. not) a symmetric degree-t bivariate polynomial. Our reconstruction protocol ensures the reconstruction of the committed secret (irrespective of whether it is *meaningful* or $NULL$). The protocol AVSS-Weak is presented in Table 3.

So AVSS-Weak satisfies all the properties of AVSS except that it does not force **D** to commit some *meaningful* secret. Hence a corrupted **D** may commit $S \in \mathbb{F}^\ell \cup NULL$.

Lemma 11 *Protocol AVSS-Weak satisfies termination property.*

Proof: Termination 1: When **D** is honest, every honest P_j's instance of AWSS-Share i.e. AWSS-Share^{P_j} will eventually terminate with the $n-t$ honest players in $CORE^{P_j}$. Thus eventually all the $n-t$ honest players will be included in $TCORE$. Also since $TCORE$ and $CORE^{P_j}$ for every $P_j \in TCORE$ has all the $n-t$ honest players, $|TCORE \cap CORE^{P_j}|$ will satisfy. Thus **D** will eventually get $CORE = TCORE$ such that $|CORE| \geq n-t$ and $|CORE \cap CORE^{P_j}| \geq (n-t)$ for $P_j \in CORE$. **D** finally A-casts $CORE$, $CORE^{P_j}$ for $P_j \in CORE$ and $OKSetP_k^{P_j}$ for every $P_k \in CORE^{P_j}$. Now the proof follows from the similar argument given in Termination 1 of Lemma 6. Thus eventually all honest players will agree on $CORE$, $CORE^{P_j}$ for $P_j \in CORE$ and $OKSetP_k^{P_j}$.

Termination 2: If an honest player P_i has completed AWSS-Single-Secret-Share, then he must have listened at least $n-t = 2t+1$ A-casts of same $CORE$, $CORE^{P_j}$ for $P_j \in CORE$ and $OKSetP_k^{P_j}$ for every $P_k \in CORE^{P_j}$. By the definition of A-cast, each honest player will also eventually listen the same. Since there are total $n = 3t+1$ players, there can not be two different version of those sets, as each honest player A-casts only one $CORE$, $CORE^{P_j}$ for $P_j \in CORE$ and $OKSetP_k^{P_j}$ for every $P_k \in CORE^{P_j}$.

Termination 3: By Termination 3 of Lemma 6, for every player $P_j \in CORE$ whose AWSS-Share^{P_j} has terminated, AWSS-Rec-Private^{P_j} will eventually terminate. Hence P_α will always terminate after doing the steps mentioned in AVSS-Weak-Rec-Private. Similarly, we can prove that every $P_\alpha \in \mathcal{P}$ will terminate AVSS-Weak-Rec-Public eventually. □

Lemma 12 *Protocol AVSS-Weak satisfies correctness property.*

Proof: Correctness 1: When **D** is honest we prove that for every $P_i \in FINAL$ (recall that $FINAL$ contains all players in $CORE$ whose AWSS-Rec-Private is successful), AWSS-Rec-Private^{P_i} must have disclosed $\overline{f_i^l(x)}$ which is the same polynomial distributed by honest **D**. For every honest player $P_i \in FINAL$ this is true. We have to prove the above statement for a corrupted $P_i \in FINAL$. A corrupted $P_i \in FINAL$ implies AWSS-Rec-Private^{P_i} is successful (i.e., the output is a symmetric degree-t bivariate polynomial and not $NULL$) and AWSS-Share^{P_i} had terminated in AVSS-Weak-Share. Moreover termination of AVSS-Weak-Share ensures $|CORE \cap CORE^{P_i}| \geq n-t$. The above statements have the following implications together: (a) P_i must have given consistent polynomials to the honest players in $CORE^{P_i}$ (during AWSS-Share^{P_i}) such that they induce valid degree-t symmetric bivariate polynomials (see Correctness 2 of Lemma 8). (b) P_i must have agreed with the honest players of $CORE^{P_i}$ with respect to the common shares given by **D**. This means that as a part of AWSS-Share^{P_i}, P_i handed over $f_j^{(P_i,l)}(x)$ to an honest P_j (in $CORE^{P_i}$) satisfying $f_j^l(i) = f_j^{(P_i,l)}(0)$ for all $l \in \{1,\ldots,\ell\}$. The statements in (a) and (b) together implies P_i must have committed (to the honest players in $CORE^{P_i}$ which are at least $t+1$) some bivariate polynomials $F^{(P_i,l)}(x,y)$ satisfying $F^{(P_i,l)}(x,0) = f_i^l(x)$. Thus if AWSS-Rec-Private^{P_i} is successful, then $\overline{F^{(P_i,l)}(x,y)} = F^{(P_i,l)}(x,y)$ and hence $\overline{f_i^l(x)} = f_i^l(x)$. Since **D** is honest, $\overline{f_i^l(x)}$ for $P_i \in FINAL$ will define $\overline{F^l(x,y)} = F^l(x,y)$. Thus $s^l = \overline{F^l(0,0)}$ will be recovered.

Correctness 2: Since in AVSS-Weak-Share, every honest player agrees on a common $CORE$, an unique secret $S' \in \mathbb{F}^\ell \cup NULL$ is defined by the $n-2t$ honest players in $CORE$. If for every two *honest*

AVSS-Weak($\mathbf{D}, \mathcal{P}, S$)

AVSS-Weak-Share($\mathbf{D}, \mathcal{P}, S$):

1. Distribution - Code for $\mathbf{D}$: $\mathbf{D}$ selects ℓ degree-t symmetric bivariate polynomials $F^1(x,y), \ldots, F^\ell(x,y)$ such that for $l = 1, \ldots, \ell$, $F^l(0,0) = s^l$. He then delivers $f_i^l(x) = F^l(x,i)$ for $l = 1, \ldots, \ell$ to P_i.

2. Code for P_i:

   1. Player P_i waits to obtain all $f_i^l(x)$ from $\mathbf{D}$. Then P_i as a dealer initiates AWSS-Share$(P_i, \mathcal{P}, (f_i^1(0), \ldots, f_i^\ell(0)))$. We call this instance of AWSS-Share initiated by P_i as AWSS-Share^{P_i}. As a part of AWSS-Share^{P_i}, P_i selects ℓ degree-t symmetric bivariate polynomials $F^{(P_i,1)}(x,y), \ldots, F^{(P_i,\ell)}(x,y)$ such that for $l = 1, \ldots, \ell$, $F^{(P_i,l)}(x,0) = f_0^{(P_i,l)}(x) = f_i^l(x)$. This ensures $F^{(P_i,l)}(0,0) = f_i^l(0)$. Also when P_i distributes $F^{(P_i,l)}(x,j) = f_j^{(P_i,l)}(x)$ to P_j (as a part of AWSS-Share^{P_i}), P_j can check $f_j^l(i) \stackrel{?}{=} f_j^{(P_i,l)}(0) = f_0^{(P_i,l)}(j) = f_i^l(j)$ (P_j obtains $f_j^l(i)$ from $\mathbf{D}$ and $f_j^{(P_i,l)}(0) = f_0^{(P_i,l)}(j) = f_i^l(j)$ from P_i). Thus in AWSS-Share^{P_i}, the selection of bivariate polynomials (by P_i) enables the consistency checking of the common shares given by $\mathbf{D}$ (P_i's share $f_i^l(j)$ given by $\mathbf{D}$ should be same as P_j's share $f_j^l(i)$ given by $\mathbf{D}$) between every pair (P_i, P_j) for $j = 1, \ldots, n$.

   2. As a part of the execution of AWSS-Share^{P_j}, if P_i receives $f_i^{(P_j,l)}(x)$, he checks $f_i^l(j) \stackrel{?}{=} f_i^{(P_j,l)}(0)$ for all $l \in \{1, \ldots, \ell\}$. If the test passes then P_i participates in AWSS-Share^{P_j} and act according to the remaining steps of AWSS-Share^{P_j}.

3. CORE Construction:

   1. $\mathbf{D}$ adds a player P_j in a set $TCORE$ (initially empty) when $\mathbf{D}$ terminates AWSS-Share^{P_j} with a set $CORE^{P_j}$ and $OKSetP_k^{P_j}$ for every $P_k \in CORE^{P_j}$.

   2. Even though $\mathbf{D}$ terminates AWSS-Share^{P_j}, he updates (and includes new players in) $CORE^{P_j}$ and $OKSetP_k^{P_j}$'s upon receiving new A-casts of the form OK(.,.) as a part of the execution of AWSS-Share^{P_j}. He also updates $TCORE$ upon terminating AWSS-Share for new players.

   3. For every update $\mathbf{D}$ becomes active and does the following computations: He assigns $CORE = TCORE$ and checks whether $|CORE \cap CORE^{P_j}| \geq n - t$ for every $P_j \in CORE$. If not then he removes P_j from $CORE$ (but not from $TCORE$) and keeps on repeating this until no more player can be removed from $CORE$. He then check whether $|CORE| \geq n - t$. If not, then he deletes $CORE$ and waits for more update. Otherwise, he A-casts $CORE$, $CORE^{P_j}$ for $P_j \in CORE$ and $OKSetP_k^{P_j}$ for every $P_k \in CORE^{P_j}$.

   4. Once a player P_i listens $\overline{CORE}$, $\overline{CORE^{P_j}}$ for $P_j \in \overline{CORE}$ and $\overline{OKSetP_k^{P_j}}$ for every $P_k \in \overline{CORE^{P_j}}$ from $\mathbf{D}$, he waits to terminate every instance of AWSS-Share^{P_j}. He also waits to listen OK(P_k, P_j) for all $P_k \in \overline{OKSetP_k^{P_j}}$ and $P_j \in \overline{CORE}$. Once P_i listens all the required OK's, he A-casts $\overline{CORE}$, $\overline{CORE^{P_j}}$ for $P_j \in \overline{CORE}$ and $\overline{OKSetP_k^{P_j}}$ for every $P_k \in \overline{CORE^{P_j}}$.

   5. Now P_i waits to listen same copy of the above sets from at least $n - t$ players' A-cast. Upon listening, he sets them as his final $CORE$, $CORE^{P_j}$ for $P_j \in CORE$ and $OKSetP_k^{P_j}$ for every $P_k \in CORE^{P_j}$ and terminates.

AVSS-Weak-Rec-Private($\mathbf{D}, \mathcal{P}, S, P_\alpha$): Private Reconstruction towards P_α:

1. For every $P_j \in CORE$, AWSS-Rec-Private$(P_j, \mathcal{P}, (f_j^1(0), \ldots, f_j^\ell(0)), P_\alpha)$ is executed with $CORE^{P_j}$ and $OKSetP_k^{P_j}$ for every $P_k \in CORE^{P_j}$. We call this instance of AWSS-Rec-Private as AWSS-Rec-Private^{P_j}. According to the property of AWSS, P_α either reconstructs $\overline{F^{(P_j,l)}(x,y)}$ for $l = 1, \ldots, \ell$ (which is committed by P_j to the honest players in $CORE^{P_j}$ in AVSS-Weak-Share) or reconstructs $NULL$.

2. P_α waits for the termination of all AWSS-Rec-Private$(P_j, \mathcal{P}, (f_j^1(0), \ldots, f_j^\ell(0)), P_\alpha)$ for $P_j \in CORE$. He then adds $P_j \in CORE$ to $FINAL$ if AWSS-Rec-Private^{P_j} is successful (i.e., the output is not $NULL$). Now for every player $P_j \in FINAL$, P_α assigns $\overline{f_j^l(x)} = \overline{F^{(P_j,l)}(x,0)}$.

3. For every pair $P_\gamma, P_\delta \in FINAL$ and for every $l \in \{1, \ldots, \ell\}$, P_α checks $\overline{f_\gamma^l(\delta)} \stackrel{?}{=} \overline{f_\delta^l(\gamma)}$. If the condition fails, then P_α outputs $\overline{S} = NULL$. Otherwise, P_α recovers $\overline{F^l(x,y)}$ and gets $\overline{s^l} = \overline{F^l(0,0)}$. Finally $\overline{S} = (\overline{s^1}, \ldots, \overline{s^\ell})$.

AVSS-Weak-Rec-Public($\mathbf{D}, \mathcal{P}, S, \mathcal{P}$): Public Reconstruction towards $\mathcal{P}$:

1. Same as in AVSS-Weak-Rec-Private, except that every AWSS-Rec-Private in AVSS-Weak-Rec-Private is replaced by AWSS-Rec-Public. Now every player $P_\alpha \in \mathcal{P}$ proceeds in the same way as in AVSS-Weak-Rec-Private and reconstructs $\overline{S}$.

Table 3: AVSS-Weak($\mathbf{D}, \mathcal{P}, S$)

players P_γ and P_δ in $CORE$ and for every $l = 1, \ldots, \ell$, $f_\gamma^l(\delta) = f_\delta^l(\gamma)$ holds then S' is the secrets defined by the constant term of the bivariate polynomials $F^l(x,y)$'s, induced by the univariate polynomials of honest players. Otherwise $\mathbf{D}$'s committed secret $S' = NULL$. We now show irrespective of whether S' is meaningful or $NULL$, the committed S' will be recovered in AVSS-Weak-Rec-Private.

If committed secret $S' = NULL$, then recovered secrets $\overline{S}$ will be $NULL$. This is because, all honest player in $CORE$ will be added to $FINAL$ and irrespective of the remaining players included in $FINAL$, the consistency checking (i.e., $\overline{f_\gamma^l(\delta)} \stackrel{?}{=} \overline{f_\delta^l(\gamma)}$) will fail and $NULL$ will be reconstructed.

Now let $\mathbf{D}$ has committed $S' \in \mathbb{F}^\ell$. Let $F'^1(x,y), \ldots, F'^\ell(x,y)$ are the corresponding committed bivariate polynomials defined by the honest players in $CORE$. Every honest $P_i \in CORE$ will enter in $FINAL$. We now show that if a corrupted P_i is present in $FINAL$, then his reconstructed univariate polynomials are consistent with polynomials of the honest players in $FINAL$ (i.e. they satisfy the pair-wise checking done in step 3 of AVSS-Weak-Rec-Private). Following the argument given in CORRECTNESS 1 of this lemma for a corrupted $P_i \in FINAL$, we conclude that P_i must have committed some bivariate polynomials $F^{(P_i,l)}(x,y)$ satisfying $F^{(P_i,l)}(x,0) = F'^l(x,i)$. Hence AWSS-Rec-Private^{P_i} will recover $F^{(P_i,l)}(x,y)$ and $F'^l(x,i)$. Hence the lemma. □

Lemma 13 *Protocol AVSS-Weak satisfies secrecy property.*

PROOF: Follows from Lemma 7, Lemma 4 and properties of symmetric bivariate polynomial. □

Lemma 14 *Protocol AVSS-Weak-Share communicates $O((\ell n^3 + n^4\kappa)\kappa)$ bits and A-casts $n^4 \log(n)$ bits. Protocol AVSS-Weak-Rec-Private and AVSS-Weak-Rec-Public communicates $O((\ell n^3 + n^4\kappa)\kappa)$ and $O(\ell n^5 + n^5\kappa)\kappa$ bits, respectively.*

PROOF: Protocol AVSS-Weak-Share and Protocol AVSS-Weak-Rec-Private execute n instances of Protocol AWSS-Share and Protocol AWSS-Rec-Private, respectively. Hence communication complexity of AVSS-Weak-Share and AVSS-Weak-Rec-Private follows from Lemma 9. Since AVSS-Weak-Rec-Public executes AVSS-Weak-Rec-Private n times, it communicates $O((\ell n^5 + n^5\kappa)\kappa)$ bits. □

As stated earlier, our AVSS-Weak-Share can not ensure that $\mathbf{D}$'s committed secrets are *meaningful*. In the sequel, we incorporate a *verification step* in AVSS-Weak to obtain AVSS protocol, where the honest players terminate their sharing phase, only after ensuring that $\mathbf{D}$ has committed something 'meaningful'. Prior to the presentation of AVSS-Share, we present another protocol called AVSS-Weak-Share-MultiSet whose goal is similar to the goal of AWSS-Share-MultiSet. Precisely, AVSS-Weak-Share-MultiSet shares $\mathcal{M}$ sets of ℓ secrets at once such that later any linear combination of the $\mathcal{M}$ sets can be reconstructed. More clearly, AVSS-Weak-Share-MultiSet can share $S^m = (s^{(m,1)}, \ldots, s^{(m,\ell)})$ for $m = 1, \ldots, \mathcal{M}$ at once such that later $S = \sum_{m=1}^{\mathcal{M}} r_m S^m = (\sum_{m=1}^{\mathcal{M}} r_m s^{(m,1)}, \ldots, \sum_{m=1}^{\mathcal{M}} r_m s^{(m,\ell)})$ can be reconstructed using Protocol AVSS-Weak-Rec-Private or AWSS-Weak-Rec-Public for some $(r_1, \ldots, r_{\mathcal{M}}) \in \mathbb{F}^{\mathcal{M}}$. It is easy to see that AVSS-Share-MultiSet satisfies termination and secrecy properties. The

AVSS-Weak-Share-MultiSet($\mathbf{D}, \mathcal{P}, S^1, \ldots, S^{\mathcal{M}}$)

1. DISTRIBUTION - CODE FOR $\mathbf{D}$: $\mathbf{D}$ executes CODE FOR $\mathbf{D}$ of AVSS-Weak-Share for every set S^m. Specifically, for every set S^m, $\mathbf{D}$ does the following: $\mathbf{D}$ selects ℓ degree-t symmetric bivariate polynomials $F^{(m,1)}(x,y), \ldots, F^{(m,\ell)}(x,y)$ such that for $l = 1, \ldots, \ell$, $F^{(m,l)}(0,0) = s^{(m,l)}$. He then delivers $f_i^{(m,l)}(x) = F^{(m,l)}(x,i)$ for $l = 1, \ldots, \ell$ to P_i.
2. CODE FOR P_i:
   1. This is same as CODE FOR P_i of AVSS-Weak-Share. The only difference is that P_i as a dealer initiates AWSS-Share-MultiSet($P_i, \mathcal{P}, S^1, \ldots, S^{\mathcal{M}}$) where $S^m = (f_i^{(m,1)}(0), \ldots, f_i^{(m,\ell)}(0))$. As described in CODE FOR P_i of AVSS-Weak-Share, P_i maintains the properties of bivariate polynomials selected in AWSS-Share-MultiSet.
3. CORE CONSTRUCTION: This is identically same as in Protocol AVSS-Weak-Share($\mathbf{D}, \mathcal{P}, s$).

correctness follows from the correctness of AVSS-Weak-Share and AWSS-Weak-Share-Multiset.

Lemma 15 *Protocol AVSS-Weak-Share-MultiSet communicates $O((\ell n^3 + n^4\kappa)\mathcal{M}\kappa)$ bits and A-casts $O(n^4 \log(n))$ bits.*

Now here is a very important observation, which is the basis of our *verification step*, ensuring that $\mathbf{D}$ has not committed $NULL$: Let r be a random number jointly generated by all (honest) players after the execution of AVSS-Weak-Share-MultiSet. If we reconstruct the linear combination $S^* = \sum_{m=1}^{\mathcal{M}} r^m S^m$ then with very high probability it will be $NULL$ if a least one of the set S^m shared by $\mathbf{D}$ was $NULL$ (the reason goes in the same line of the argument as given in Section 4.4 of [10]). Thus with very

high probability, we can detect a corrupted $\mathbf{D}$ who did not share all the sets meaningfully. Also by the reconstructing S^*, the information theoretic security on set S^1 is lost. But the remaining sets $S^2, \ldots, S^{\mathcal{M}}$ remains secure. To avoid this, we can make AVSS-Weak-Share-MultiSet to share $\mathcal{M}+1$ sets where the first set S^0 can be used for padding the remaining sets of secrets, namely $S^1, \ldots, S^{\mathcal{M}}$. This provides a good clue of how AVSS-Weak-Share-MultiSet can be used to ensure that $\mathbf{D}$'s committed secrets are all meaningful. We implement the above idea in our protocol AVSS-Share which is given in the sequel. We now present our AVSS protocol AVSS which shares ℓ secrets. For that $\mathbf{D}$ first divides ℓ secrets equally into n^2 sets, namely $S^1, \ldots, S^{n^2}$. So every set S^m contains $\lceil \frac{\ell}{n^2} \rceil$ secrets. If n^2 does not divide ℓ, then $\mathbf{D}$ adds random numbers in the last set so that it contains $\lceil \frac{\ell}{n^2} \rceil$ secrets. $\mathbf{D}$ also selects $\lceil \frac{\ell}{n^2} \rceil$ random numbers from $\mathbb{F}$ to put in a set S^0. Now step 1,2 and 3 in CODE FOR P_i

AVSS$(\mathbf{D}, \mathcal{P}, S)$

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

DISTRIBUTION - CODE FOR $\mathbf{D}$:

1. $\mathbf{D}$ first divides ℓ secrets equally into n^2 sets, namely $S^1, \ldots, S^{n^2}$. So every set S^m contains $\lceil \frac{\ell}{n^2} \rceil$ secrets. If n^2 does not divide ℓ, then add random numbers in the last set so that it contains $\lceil \frac{\ell}{n^2} \rceil$ secrets. $\mathbf{D}$ also selects $\lceil \frac{\ell}{n^2} \rceil$ random numbers from $\mathbb{F}$ to put in a set S^0.
2. $\mathbf{D}$ executes AVSS-Weak-Share-MultiSet$(\mathbf{D}, \mathcal{P}, S^0, S^1, \ldots, S^{n^2})$.

CODE FOR P_i: VERIFICATION STEP

1. Upon completion of AVSS-Weak-Share-MultiSet$(\mathbf{D}, \mathcal{P}, S^0, S^1, \ldots, S^{n^2})$, player P_i participates in protocol RandomGenerator given in Section 2 to generate a random number r.
2. Once r is generated, P_i locally computes the sharings of $S^* = \sum_{m=0}^{n^2} r^m S^m$.
3. P_i invokes AVSS-Weak-Rec-Public$(\mathbf{D}, \mathcal{P}, S^*, \mathcal{P})$ to reconstruct S^* towards every player in $\mathcal{P}$. P_i terminates AVSS-Share if he does not reconstruct $NULL$.If an honest P_i terminates, then $\mathbf{D}$ has meaningfully shared the secrets in $S^1, \ldots, S^{n^2}$ with very high probability.

AVSS-Rec-Private$(\mathbf{D}, \mathcal{P}, S, P_\alpha)$: Private Reconstruction towards P_α:

1. Every S^m is reconstructed by P_α by executing AVSS-Weak-Rec-Private$(\mathbf{D}, \mathcal{P}, S_m, P_\alpha)$. But here P_α will be able to reconstruct some $\overline{S^m}$ which is not NULL. This is ensured by the *verification step* used in AVSS-Share.

2. After recovering each $\overline{S^m}$, P_α concatenates them to get $\overline{S}$.

AVSS-Rec-Public$(\mathbf{D}, \mathcal{P}, S, \mathcal{P})$: Public Reconstruction towards $\mathcal{P}$:

1. Same as in AVSS-Rec-Private, except that every AVSS-Weak-Rec-Private in AVSS-Rec-Private is replaced by AVSS-Weak-Rec-Public. Now every player $P_\alpha \in \mathcal{P}$ proceeds in the same way as in AVSS-Rec-Private and reconstructs $\overline{S}$.

incorporate the *verification step* to ensure that $\mathbf{D}$ has shared meaningful secrets. Notice that random r is generated only after the completion of the sharing of $n+1$ sets $S^0, S^1, \ldots, S^{n^2}$. This is crucial as it ensures the following: if at least one of the sets $S^0, S^1, \ldots, S^{n^2}$ is not shared meaningfully (i.e. some $S^m = NULL$), then the sharing of the linear combination $S^* = \sum_{m=0}^{n^2} r^m S^m$ will not be meaningful. When we try to reconstruct S^* using AVSS-Weak-Rec-Public, $S^* = NULL$ will be reconstructed with very high probability. In the sequel, we prove the above statement.

Theorem 2 *If at least one of the sets $S^0, S^1, \ldots, S^{n^2}$ is not shared meaningfully (i.e. some $S^m = NULL$), then the sharing of the linear combination $S^* = \sum_{m=0}^{n^2} r^m S^m$ will not be meaningful with very high probability.*

PROOF: Let one of the set S^k is not shared meaningfully. Recall that every secret in S^k is supposed to be shared by a degree-t symmetric bivariate polynomial. More specifically, the honest players in $CORE$ must define a degree-t symmetric bivariate polynomial for each secret in S^k. If at least one of the bivariate polynomials corresponding to some secret in S^k is not symmetric, then we say that the corresponding committed secret is $NULL$ and this makes the entire bunch of secrets in S^k to be $NULL$. Thus a set $S^k = NULL$ implies at least one secret in S^k is not shared by a symmetric degree-t bivariate polynomial. Let that secret be l^{th} secret in S^k. Recall that every set S^m has $\lceil \frac{\ell}{n^2} \rceil$ secrets. Now consider l^{th} secret from every set S^m for $m = 0, \ldots, n^2$. Let these secrets

be shared by polynomials $F^{(0,l)}(x,y),\ldots,F^{(n^2,l)}(x,y)$ respectively. According to our assumption, $F^{(k,l)}(x,y)$ is not a symmetric degree-t bivariate polynomial. Thus there exists at least two honest players, say P_i and P_j in $CORE$ such that $F^{(k,l)}(i,j) \neq F^{(k,l)}(j,i)$. Now we show that the polynomial $F^{(*,l)}(x,y) = \sum_{m=0}^{n^2} F^{(m,l)}(x,y)r^m$ will be asymmetric with respect to same $P_i, P_j \in CORE$ with very high probability. Notice that $F^{(*,l)}(x,y)$ will be symmetric with respect to P_i, P_j iff $\sum_{m=0}^{n^2} F^{(m,l)}(i,j)r^m = \sum_{m=0}^{n^2} F^{(m,l)}(j,i)r^m$. Now $\sum_{m=0}^{n^2} F^{(m,l)}(i,j)r^m$ can be viewed as $g_1(r)$ where $g_1(x)$ is a n^2 degree polynomial defined as $g_1(x) = \sum_{m=0}^{n^2} F^{(m,l)}(i,j)x^m$. Similarly $\sum_{m=0}^{n^2} F^{(m,l)}(j,i)r^m$ can be viewed as $g_2(r)$ where $g_2(x)$ is a n^2 degree polynomial defined as $g_2(x) = \sum_{m=0}^{n^2} F^{(m,l)}(j,i)x^m$. It is clear that $g_1(x) \neq g_2(x)$, as $F^{(l,k)}(i,j) \neq F^{(l,k)}(j,i)$. But $F^{(*,l)}(x,y)$ will be symmetric with respect to P_i, P_j iff $g_1(r) = g_2(r)$. Since r is computed only after $\mathbf{D}$ has shared all the n^2 sets, the probability that a corrupted $\mathbf{D}$ can correctly guess r before its generation such that $g_1(r) = g_2(r)$ is at most $\frac{n^2}{|\mathbb{F}|} = 2^{-O(\kappa)}$. This implies that with very high probability, $F^{(l,*)}(x,y)$ will be asymmetric with respect to P_i, P_j. Hence the sharing of S^* will not be meaningful with very high probability. □

Theorem 3 *Protocol AVSS-Share communicates $O((\ell n^3 + n^6\kappa)\kappa)$ bits and A-casts $n^4\log(n)$ bits.*

PROOF: Protocol AVSS-Share calls AVSS-Weak-Share-MultiSet with n^2+1 sets $S^0,\ldots,S^{n^2}$ each containing $\frac{\ell}{n^2}$ secrets. Hence by Lemma 15, it incurs a communication cost $O(\ell n^3 + n^6\kappa)\kappa$ bits and A-casts of $n^4\log(n)$ bits. Generation of random number requires communication complexity independent of secret size ℓ. Finally AVSS-Share calls AVSS-Weak-Rec-Public with S^*, which requires $O(\ell n^3 + n^5\kappa)\kappa$ bits of communication. □

7 Asynchronous BA with $n = 3t+1$

We now briefly sketch how we use our new AVSS scheme AVSS to design efficient $(1-2^{-O(\kappa)})$-terminating, t-resilient ABA with $3t+1$ players. We follow the same line of Canetti et. al. [7] to design our ABA protocol. We proceed in two steps. The first step is to get a global coin. In [7], the authors have shown how to implement global coin which terminates with probability $1-2^{-O(\kappa)}$ provided there exists an AVSS scheme which terminates with probability $1-2^{-O(\kappa)}$. In their implementation of global coin, each player begins by sharing n random secrets using the AVSS scheme of [7]. Since the AVSS scheme given in [7] shares only one secret at a time, in order to share n secrets simultaneously, each player has to parallely execute n separate instances of AVSS scheme, resulting in high communication overhead. However, using our new AVSS scheme, each player can simultaneously share n secrets with comparatively very less communication complexity. Now the rest of the steps of our global coin primitive will be same as in [7].

The second step is to use the common coin protocol to get ABA protocol. The author of [7] use their common coin protocol that terminates with probability $(1-2^{-O(\kappa)})$ to get a $(1-2^{-O(\kappa)})$-terminating, t-resilient ABA protocol. We replace their common coin by our common coin to obtain our efficient $(1-2^{-O(\kappa)})$-terminating, t-resilient ABA with $3t+1$ players. We avoid giving the details as it will be a repetition of the steps given [7]. Thus we have the following theorem:

Theorem 4 *Let $n \geq 3t+1$. Then for every $\kappa > 0$, there exists an $(1-2^{-O(\kappa)})$-terminating, t-resilient ABA protocol for n players, which terminates in constant expected time. Moreover, the protocol privately communicates $\mathcal{O}(cn^6\kappa)$ bits and A-casts $\mathcal{O}(cn^5\log(n))$ bits, where $c>0$ is a constant.*

PROOF: The communication complexity is easy. Derivation of the constant expected time follows the same argument as given in [7, 6]. □

8 Conclusion

In this paper, we have presented a novel AVSS protocol with optimal resilience whose communication cost is significantly better than the previous AVSS protocol of [7]. Moreover using our AVSS protocol, we have designed a new expected constant time $(1-2^{-O(\kappa)})$-terminating, t-resilient ABA protocol whose communication complexity is significantly better than existing ABA protocols in the same settings [7, 1]

(though the ABA protocol of [1] has a strong property of being *almost surely terminating*). Finally, our AVSS protocol can be further extended to make it suitable for designing efficient asynchronous multiparty computation (MPC) with $n = 3t + 1$. However, we avoid giving the exact details as it is out of scope of this paper.

References

[1] I. Abraham, D. Dolev, and J. Y. Halpern. An almostsurely terminating polynomial protocol for asynchronous Byzantine agreement with optimal resilience. In *PODC*, pages 405–414, 2008.

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

[3] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

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

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

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

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

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

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

[10] I. Damgård and J. B. Nielsen. Scalable and unconditionally secure multiparty computation. In *Proc. of CRYPTO*, volume 4622 of *LNCS*, pages 572–590. Springer Verlag, 2007.

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

[12] M. J. Fischer, N. A. Lynch, and M. Paterson. Impossibility of distributed consensus with one faulty process. *JACM*, 32(2):374–382, 1985.

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

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

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

[16] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[17] Arpita Patra, Ashish Choudhary, AshwinKumar B.V, and C. Pandu Rangan. On Round Complexity of Unconditional VSS. Cryptology ePrint Archive, Report 2008/172, 2008.

[18] M. Pease, R. E. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *JACM*, 27(2):228–234, 1980.

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

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