Statistical Asynchronous Weak Commitment Scheme: A New Primitive to Design Statistical Asynchronous Verifiable Secret Sharing Scheme

Ashish Choudhury[1]* **Arpita Patra**[2]

[1]Applied Statistics Unit,
Indian Statistical Institute, Kolkata India
partho_31@yahoo.co.in, partho31@gmail.com

[2]Department of Computer Science
Aarhus University, Denmark
arpitapatra10@gmail.com, arpita@cs.au.dk

Abstract

In this paper, we consider the problem of statistical *Asynchronous Verifiable Secret Sharing* (AVSS) with *optimal resilience*, involving $n = 3t + 1$ parties over a completely asynchronous network, who are pair-wise connected by secure channel, such that t out of the n parties are under the control of a *computationally unbounded* active adversary. In this setting, the best known AVSS protocols are proposed by Patra et al. in [18] and [16], which are used to design *Asynchronous Byzantine Agreement* (ABA) and statistical *Asynchronous Multiparty Computation* (AMPC) respectively. The AVSS of [16] achieves slightly stronger properties than that of [18]. However, both the AVSS protocols of [18] and [16] use a well known asynchronous primitive, namely *Asynchronous Weak Secret Sharing* (AWSS) as a black-box. The AWSS protocol of [18] as well as [16] is based on the idea of sharing a secret using bivariate polynomial of degree-t in x and y and invokes n^2 instances of another well known asynchronous primitive, namely *Asynchronous Information Checking Protocol* (AICP).

In this paper, we propose a new asynchronous primitive called *asynchronous weak commitment scheme* (AWCS), which has weaker requirements than AWSS and which is first of its kind. Due to its weaker requirements, we find that it is conceptually simpler to construct AWCS than AWSS. Specifically, AWCS can be designed using simple Shamir secret sharing scheme [21] (based on univariate polynomial of degree-t in x), instead of using bivariate polynomials. Moreover, our AWCS scheme invokes only n instances of AICP, in comparison to n^2 instances of AICP, as done in the AWSS scheme of [18] and [16]. Furthermore, we find that we can substitute our AWCS scheme in place of AWSS scheme in the AVSS protocols of [18] and [16], without affecting the properties of the AVSS protocols. As a result, the communication complexity of the AVSS protocols of [18] and [16] are improved by a factor of $\Omega(n)$. This further reduces the communication complexity of the ABA and AMPC protocol of [18] and [16] by a factor of $\Omega(n)$. This is a positive step towards solving the open problem posed in [18] and [16], urging to improve the communication complexity of their ABA and AMPC protocol.

Keywords: Secret Sharing, Asynchronous Network, Information Theoretic Security.

1 Introduction

Verifiable Secret Sharing (VSS) [7] is a fundamental primitive in secure distributed computing. VSS extends the notion of *secret sharing* [21] against malicious/active corruption. VSS is a two phase (share, reconstruct) protocol, carried out among n parties. In a VSS protocol, there exists a special party called *dealer*, denoted by D, who has a secret s which he wants to share among the n parties. There exists a *computationally unbounded* adversary, who can *actively* corrupt at most t parties, out of the n parties (possibly including D). The goal of the VSS protocol is to allow D to share s during sharing phase, such that the secrecy of s is maintained during sharing phase if D is *honest*. Moreover, even if D is *corrupted*, there should exist a *unique* value to which D is committed to during sharing phase and the same unique value should be reconstructed in the reconstruction phase, irrespective of behavior of the corrupted parties.

*Financial support from Department of Information Technology, Govt. of India acknowledged.

1.1 Motivation of Our Work

Due to its importance in secure distributed computing, VSS has been studied extensively over the past three decades [2, 20, 8, 11, 10, 12, 9, 15]. However, almost all existing VSS protocols are designed in *synchronous* network, where it is assumed that there exists a global clock in the system and the delay in the transmission of any message is bounded by a constant. Though theoretically impressive, they do no model real life scenario appropriately, as a single delay in the transmission of a message will affect the overall security of synchronous VSS protocols. Motivated by this, researchers have studied VSS in *asynchronous* network [6, 3, 1, 17, 18, 16]. However unlike synchronous VSS protocols, existing asynchronous VSS (AVSS) protocols involve huge communication complexity. This motivates us to reduce the communication complexity of AVSS.

1.2 Asynchronous Verifiable Secret Sharing

We follow the network model of [6]. Specifically, there is a set of n parties, say $\mathcal{P} = \{P_1, \ldots, P_n\}$, where every two parties are directly connected by a secure and authentic channel and t out of the n parties can be under the influence of a *computationally unbounded Byzantine (active) adversary*, denoted as $\mathcal{A}_t$. The adversary $\mathcal{A}_t$, completely dictates the parties under its control and can force them to deviate from the protocol in any arbitrary manner. The parties not under the influence of $\mathcal{A}_t$ are called *honest or uncorrupted*.

The underlying network is asynchronous, where the communication channels between the parties have *arbitrary*, yet finite delay (i.e the messages are guaranteed to reach eventually). To model the worst case scenario, $\mathcal{A}_t$ is given the power to schedule the delivery of *all* messages in the network. However, $\mathcal{A}_t$ can *only schedule* the messages communicated between honest parties, without having any access to them. *In asynchronous network, the inherent difficulty in designing a protocol comes from the fact that when a party does not receive an expected message then he cannot decide whether the sender is corrupted (and did not send the message at all) or the message is just delayed.* So a party can not wait to consider the values sent by all parties, as waiting for all of them could turn out to be endless. Hence the values of up to t (potentially honest) parties may have to be ignored. Due to this, protocols in asynchronous network are generally involved in nature and require new set of primitives. For a good introduction to asynchronous protocols, see [5].

In the literature, there are two types of statistical AVSS protocols, called strong-AVSS and weak-AVSS [14], both of which are used for different purpose. We now present both these definitions.

Definition 1 (Statistical Weak-AVSS [14, 18]) *Let (Sh, Rec) be a pair of protocols in which a dealer $D \in \mathcal{P}$ shares a secret s, selected from a finite field $\mathbb{F}$. Moreover, let ϵ be the error parameter, where ϵ is given and $\epsilon > 0$. We say that (Sh, Rec) is a statistical weak-AVSS scheme for n parties if all the following hold for every possible behavior of $\mathcal{A}_t$:*

- **Termination***: All the following requirements hold except with probability ϵ:*
  1. *If D is honest and all honest parties participate in the protocol, then each honest party will eventually terminate protocol Sh.*
  2. *If some honest party has terminated protocol Sh, then irrespective of the behavior of D, each honest party will eventually terminate Sh.*
  3. *If all honest parties have terminated Sh and invoked Rec, then each honest party will eventually terminate Rec.*
- **Correctness***: All the following requirements hold except with probability ϵ:*
  1. *If D is honest then each honest party upon terminating Rec, outputs the shared secret s.*
  2. *If D is faulty and some honest party has terminated Sh, then there exists a unique $s^\star \in \mathbb{F} \cup \{NULL\}$, such that D is committed to $s^\star$ and each honest party upon terminating Rec will output only $s^\star$. This property is also called as strong-commitment.*
- **Secrecy***: If D is honest and no honest party has begun executing protocol Rec, then $\mathcal{A}_t$ has no information about s.*

Definition 2 (Statistical Strong-AVSS [14, 16]) *It is same as weak-AVSS, except that* **Correctness 2** *property is strengthened as follows:*

- **Correctness 2***: If D is faulty and some honest party has terminated Sh, then there exists a unique $s^\star \in \mathbb{F}$ (and not $\mathbb{F} \cup \{NULL\}$), such that D is committed to $s^\star$ and each honest party upon terminating Rec will output only $s^\star$.*

Remark 1 (Difference Between Weak-AVSS and Strong-AVSS) *The only difference between Weak-AVSS and Strong-AVSS is that in the former, a corrupted D's committed value can be NULL, while in later the committed value should be from $\mathbb{F}$. Even in synchronous world, we have Strong-VSS and Weak-VSS. While the VSS protocols of [15, 13] are Weak-VSS (corrupted D's commitment can be NULL), the VSS protocols of [12, 2, 8] are Strong-VSS (corrupted D's commitment belongs to $\mathbb{F}$). The fundamental difference between these two type of VSS protocols is that in Weak-VSS scheme, we cannot ensure that the values distributed by a corrupted D to the honest parties during sharing phase are "pair-wise" consistent and hence satisfy certain condition. As a result of this, we say that D has committed NULL. On the other hand, Strong-VSS scheme ensures that the values distributed by a corrupted D to the honest parties during sharing phase are "pair-wise" consistent and hence satisfy certain condition. However, ensuring the same does require additional "resources", like more number of communication rounds, more communication complexity and even additional complex computational steps, in comparison to Weak-VSS protocols. Nevertheless, Weak-VSS protocols find several applications, like for designing Byzantine Agreement [18, 19, 5] and studying the round complexity of VSS [13, 15]. On the other hand, Strong-VSS protocols are used for designing for Multiparty Computation (MPC) protocols [12, 16].* □

The above definitions of AVSS can be extended for secret S containing ℓ element(s) from $\mathbb{F}$.

1.3 Approach Used in Designing Existing Statistical AVSS Protocols

Statistical AVSS (both strong and weak) is possible iff $n \geq 3t+1$ [6]. Hence any statistical AVSS protocol (whether strong or weak) designed with $n = 3t+1$ is said to have *optimal resilience*. In the literature, there are only three *optimally resilient* statistical AVSS protocols, presented in [6], [18] and [16]. We now discuss these protocols and the approach used in designing them.

1. Canetti and Rabin [6] presented the first statistical AVSS protocol with optimal resilience. There are well known inherent difficulties in designing AVSS with $n = 3t+1$ (see [6, 5]). To overcome these difficulties, the authors in [6] used the following route to design their AVSS scheme: *AICP* $\rightarrow$ *A-RS* $\rightarrow$ *AWSS* $\rightarrow$ *Two & Sum AWSS* $\rightarrow$ *AVSS*, where $X \rightarrow Y$ means that protocol Y is designed using protocol X as a black-box. Since the final AVSS scheme is designed on the top of so many sub-protocols, it is highly communication intensive as well as very much involved. The protocol privately communicates[1] $\mathcal{O}(n^9(\log \frac{1}{\epsilon})^4)$ bits, A-casts[2] $\mathcal{O}(n^9(\log \frac{1}{\epsilon})^2 \log(n))$ bits during *sharing phase* and privately communicates $\mathcal{O}(n^6(\log \frac{1}{\epsilon})^3)$ bits, A-casts $\mathcal{O}(n^6(\log \frac{1}{\epsilon}) \log(n))$ bits during *reconstruction phase* for sharing a single secret s. Using their AVSS, [6] designed the first ABA protocol with $n = 3t+1$.

2. After the seminal result of [6], there was al most no attempt to improve the communication complexity of AVSS protocol with $n = 3t+1$. The first significant attempt in this direction was made by Patra et al. [18], who designed a Weak-AVSS with $n = 3t+1$. To design their AVSS, they used the following shorter route: *AICP* $\rightarrow$ *AWSS* $\rightarrow$ *AVSS*. Moreover, they used new ideas in designing each of the underlying primitives. Furthermore, each of their underlying primitive is designed to deal with ℓ elements from $\mathbb{F}$ *concurrently*. The AWSS (asynchronous weak secret sharing) scheme[3] of [18] is adapted from the WSS scheme of [8] with $n = 2t+1$ in synchronous settings, with several additional steps to deal with the asynchrony of the network. Informally, the AWSS scheme of [18] uses the idea of sharing the secret using symmetric bivariate polynomial of degree-t in x and y, where the constant term of the polynomial is the secret. This idea (of using bivariate polynomial) is used in al most all existing WSS protocols in synchronous settings [8, 10, 12, 15]. The AWSS scheme of [18] invokes n^2 instances of AICP[4] scheme of [18]. Finally, the AVSS scheme of [18] invokes n instances of AWSS scheme of [18]. *This implies that the communication complexity of the AVSS scheme of [18] is that of $n \cdot n^2 = n^3$ instances of AICP scheme of [18].* Specifically, to share ℓ secrets, the AVSS scheme of [18] requires private communication as well as A-cast communication of $\mathcal{O}((\ell n^3 + n^4) \log \frac{1}{\epsilon})$ bits during sharing phase and A-cast communication of $\mathcal{O}((\ell n^3 + n^4) \log \frac{1}{\epsilon})$ bits during reconstruction phase. This significantly improves the AVSS scheme of [6]. Using their AVSS scheme, [18] presented a communication efficient ABA protocol, which invokes n instances of Weak-AVSS scheme of [18] and hence $n \cdot n^3 = n^4$ instances of AICP of [18].

3. Though Weak-AVSS scheme of [18] is efficient, it is not suitable for designing statistical *asynchronous multiparty computation* (AMPC) (see [16] for the details). Motivated by this, Patra et al. [16] designed a Strong-AVSS scheme in [16]. The AVSS scheme of [16] follows the same route as used in the AVSS scheme

[1] Communication done over secure channels.

[2] See Sec. 2 for details about A-cast

[3] AWSS satisfies the same properties as Weak-AVSS, except that **Correctness 2** property is *weakened* as follows: If D is *faulty* and some honest party has terminated Sh, then there exists a *unique* $s^\star \in \mathbb{F} \cup \{NULL\}$, such that D is committed to $s^\star$ and each honest party upon terminating Rec will output *either* $s^\star$ or $NULL$.

[4] See Sec. 2 for details about AICP

of [18], namely *AICP* $\rightarrow$ *AWSS* $\rightarrow$ *AVSS*. The AICP of [16] is different from the AICP of [18]. On the other hand, the AWSS scheme of [16] is same as that of [18] (based on the idea of bivariate polynomials), except that now it uses the AICP of [16] instead of AICP of [18]. Thus, AWSS scheme of [16] invokes n^2 instances of AICP of [16]. Finally, the AVSS scheme of [16] invokes n instances of AWSS scheme of [16]. However, the AVSS scheme of [16] is much more involved than the AVSS scheme of [18] because of being a Strong-AVSS scheme. Nevertheless, *the communication complexity of the AVSS scheme of [16] is that of* $n \cdot n^2 = n^3$ *instances of AICP scheme of [16].* Specifically, to share ℓ secrets, the AVSS scheme of [16] requires private communication of $\mathcal{O}((\ell n^3 + n^4 \log \frac{1}{\epsilon}) \log \frac{1}{\epsilon})$ bits and A-cast communication of $\mathcal{O}(n^3 \log n)$ bits during sharing phase and A-cast communication of $\mathcal{O}(n^3 \log n)$ bits during reconstruction phase. So as compared to Weak-AVSS of [18], the A-cast communication of Strong-AVSS of [16] is *independent* of ℓ. Now using their Strong-AVSS scheme, [16] presented a new asynchronous primitive called *asynchronous complete secret sharing scheme* (ACSS) [16], which invokes n instances of AVSS scheme of [16]. Finally using their ACSS scheme, [16] presented a new and efficient statistical *asynchronous multiparty computation* (AMPC) protocol, which invokes n instances of ACSS scheme of [16]. So the communication complexity of the AMPC protocol of [16] is that of n^5 instances of AICP of [16].

1.4 Our Result and Its Applications

In [18] as well as in [16], the authors have left it as an open problem to further reduce the communication complexity of their AVSS scheme. In this article, we make a positive step towards this direction. Specifically, we introduce a new asynchronous primitive called *asynchronous weak commitment scheme* (AWCS), which can act as a *substitute* for AWSS. We find that AWCS has weaker requirements than AWSS and hence it is conceptually easier to design AWCS than AWSS. Specifically, instead of bivariate polynomials used for designing AWSS scheme [18, 16], we find that we can design AWCS scheme using simple Shamir secret sharing [21] (based on univariate polynomial) along with AICP. As a result, we can design AWCS scheme using only n instances of AICP, as compared to n^2 instances of AICP used for designing AWSS scheme. Moreover, we only need to slightly modify the existing AVSS protocols of [18] and [16], so that it can use our AWCS scheme instead of AWSS. As a result, we get a Strong-AVSS and a Weak-AVSS scheme that uses only $n \cdot n = n^2$ instances of AICP, instead of n^3 instances of AICP. So the resultant AVSS schemes gain a factor of $\Omega(n)$ over the AVSSs of [18] and [16]. When further used for designing ABA and AMPC, this results in a gain of $\Omega(n)$ over ABA and AMPC of [18] and [16] respectively. *The novelty of this article is the specific design of AWCS scheme and the way we use it in the AVSS scheme of [18] and [16].*

2 Primitives Used

We now present the primitives which are used in this paper. Our AVSS protocols have a negligible error probability of ϵ, where $\epsilon > 0$ and is called the error parameter. To bound the error probability by ϵ, all our protocols work over a finite field $\mathbb{F}$ where $\mathbb{F} = GF(2^\kappa)$ and $\epsilon = 2^{-\Omega(\kappa)}$, for some non-zero κ. Thus each field element can be represented by $\mathcal{O}(\kappa) = \mathcal{O}(\log \frac{1}{\epsilon})$ bits. Moreover, as assumed in [18, 16], without loss of generality, we assume $n = poly(\kappa)$. That is, n is polynomial in κ. Thus $n = poly(\log \frac{1}{\epsilon})$.

2.1 Asynchronous Broadcast: A-cast

A-cast is the parallel notion of *broadcast* from the synchronous world and it allows a party to send some information identically to all the parties.

Definition 3 (A-cast [6, 18]) *Let* Π *be an asynchronous protocol initiated by a special party (called the sender), having input* m *(the message to be broadcast). We say that* Π *is a* t-*resilient A-cast protocol if the following hold:*

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

Bracha [4] gave an elegant implementation of A-cast with $n = 3t + 1$. For details, see [5]. The following theorem states the communication complexity of Bracha's A-cast protocol.

Theorem 1 ([6, 18, 16]) *Bracha's A-cast protocol privately communicates $\mathcal{O}(|m|n^2)$ bits.*

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

2.2 Asynchronous Information Checking Protocol: AICP

Information Checking Protocol (ICP) is a tool, introduced in [20] to authenticate data in synchronous world even in the presence of computationally unbounded adversary. In [18], the authors extended this notion to asynchronous world. We now recall the definition of AICP from [18].

Definition 4 (Asynchronous Information Checking Protocol (AICP) [18]) *Let a dealer $D \in \mathcal{P}$ and D has a secret $S = (s^1, \ldots, s^\ell)$, containing ℓ element(s) from $\mathbb{F}$. D wants to give S to an intermediary $INT \in \mathcal{P}$, such that later when INT reveals S, the entire set $\mathcal{P}$ can act as verifiers and verify that S was indeed received by INT from D. Then any AICP protocol to achieve this task is a sequence of following three phases:*

1. **Generation Phase***: initiated by D, where D privately sends S, along with some authentication information to INT and some verification information to individual verifiers.*
2. **Verification Phase***: initiated by INT where INT interacts with D and the verifiers in $\mathcal{P}$ to ensure the consistency of S and authentication information possessed by INT and verification information possessed by individual verifiers. Specifically, INT ensures that INT possesses an S obtained from D, which will be later accepted by each (honest) verifier in $\mathcal{P}$. The secret S, along with the authentication information, which is finally possessed by INT at the end of* **Verification Phase** *is called as D's IC signature on S given to INT and is denoted by $ICSig(D, INT, \mathcal{P}, S)$.*
3. **Revelation Phase***: carried out by INT and the verifiers in $\mathcal{P}$. Here INT reveals $ICSig(D, INT, \mathcal{P}, S)$: that is INT reveals the secret S along with the authentication information. The verifiers then publish their responses after verifying S with respect to their verification information. Depending upon the responses by the verifiers, every individual verifier $P_i \in \mathcal{P}$ either accepts S (indicating that P_i is convinced that S was indeed obtained by INT from D) or rejects it (indicating that P_i is not convinced that S was indeed obtained by INT from D). Upon acceptance (resp., rejection), verifier P_i sets $\mathsf{Reveal}_i = S$ (resp., $\mathsf{Reveal}_i = NULL$).*

Any AICP should satisfy the following properties:

1. **AICP-Correctness1:** *If D and INT are honest, then S will be accepted in* **Revelation Phase** *by each honest verifier without any error.*
2. **AICP-Correctness2:** *At the end of* **Verification Phase***, an honest INT will possess an S which will be accepted in* **Revelation Phase** *by each honest verifier, except with probability ϵ.*
3. **AICP-Correctness3:** *If D is honest, then during* **Revelation Phase***, with probability at least $(1-\epsilon)$, every $S' \neq S$ revealed by a corrupted INT will not be accepted by an honest verifier.*
4. **AICP-Secrecy:** *If D and INT are honest and INT has not started* **Revelation Phase***, then $\mathcal{A}_t$ will have no information about S.*

In [18] and [16], the authors have presented two different AICP protocols with $n = 3t + 1$ having different communication complexities. However, we do not go into the details of these AICP protocols because we will use AICP as a black-box in our AWCS scheme. Now depending upon which AICP is used in the AWCS scheme, we will get different communication complexities. Instead of giving the exact communication complexity of our AWCS scheme, we only measure the number of instances of underlying AICP protocol, invoked in the AWCS scheme.

In both the AICP protocol of [18] as well as [16], the **Generation Phase** is denoted as **Gen(**$D, INT, \mathcal{P}, S, \epsilon$**)**, the **Verification Phase** is denoted as **Ver(**$D, INT, \mathcal{P}, S, \epsilon$**)** and the **Revelation Phase** is denoted as **Reveal-Public(**$D, INT, \mathcal{P}, S, \epsilon$**)**. We now recall the following notation of using AICP protocol from [18, 16], which will be used while describing our AWCS scheme.

Notation 2 ([18]) *Recall that D and INT can be any party from $\mathcal{P} = \{P_1, \ldots, P_n\}$. We say that:*

1. *"P_i gives $ICSig(P_i, P_j, \mathcal{P}, S)$ to P_j" to mean that P_i as a dealer executes $\mathsf{Gen}(P_i, P_j, \mathcal{P}, S, \epsilon)$, considering P_j as INT to give his IC signature on S to P_j.*
2. *"P_i receives $ICSig(P_j, P_i, \mathcal{P}, S)$ from P_j" to mean that P_i as INT has completed $\mathsf{Ver}(P_j, P_i, \mathcal{P}, S, \epsilon)$ with the help of the verifiers in $\mathcal{P}$ and finally possess $ICSig(P_j, P_i, \mathcal{P}, S)$, where P_j is the dealer.*

3. *"P_i reveals $ICSig(P_j, P_i, \mathcal{P}, S)$" to mean P_i as INT executes* **Reveal-Public**$(P_j, P_i, \mathcal{P}, S, \epsilon)$ *along with the participation of the verifiers in $\mathcal{P}$ to reveal S, which he has obtained from P_j.*

4. *"P_k completes revelation of $ICSig(P_j, P_i, \mathcal{P}, S)$ with $\mathsf{Reveal}_k = \overline{S}$ (resp. $\mathsf{Reveal}_k = NULL$)" to mean that P_k as a verifier has completed* **Reveal-Public**$(P_j, P_i, \mathcal{P}, S, \epsilon)$ *with $\mathsf{Reveal}_k = \overline{S}$ (resp. $\mathsf{Reveal}_k = NULL$).*

We are now ready to present our AWCS scheme, which is the main contribution of this paper.

3 Asynchronous Weak Commitment Scheme (AWCS)

We now present the definition of statistical AWCS. We first present the definition with respect to a single secret s. The definition can be extended in a straightforward way for a secret S, containing ℓ element(s).

Definition 5 (Statistical Asynchronous Weak Commitment Scheme (AWCS)) *In an AWCS scheme, there exists a dealer $D \in \mathcal{P}$, who has a secret s, which he wants to commit to the parties in $\mathcal{P}$. The scheme consists of two phases as follows:*

1. **Commit phase***: In this phase, D tries to commit s.*

2. **Decommit phase***: In this phase, D tries to decommit s. If the decommit is successful, then s is accepted as authentic. Otherwise, D is discarded.*

A pair of protocols $(\mathsf{Com}, \mathsf{DeCom})$ is said to be a valid AWCS scheme, if it satisfies the following conditions for every possible behavior of $\mathcal{A}_t$:

- **Termination***: All the following requirement should hold:*

  1. *If D is honest and all honest parties participate in Com, then all honest parties will eventually terminate Com.*
  2. *If some honest party has terminated Com, then irrespective of the behavior of D, all honest parties will eventually terminate Com.*
  3. *If all honest parties have terminated Com and if D invokes DeCom, then all honest parties will eventually terminate DeCom.*

- **Correctness***: All the following requirement should hold:*

  1. *If D is honest then s will be accepted as authentic by all honest parties during DeCom, except with error probability ϵ.*
  2. *If D is corrupted and some honest party has terminated Com, then there exists a unique $s^\star \in \mathbb{F} \cup \{NULL\}$, such that D has committed $s^\star$ during Com. Moreover, if some honest party terminates DeCom and if some s' is accepted as authentic, then $s' = s^\star$, except with error probability ϵ.*

- **Secrecy***: If D is honest and has not invoked DeCom, then $\mathcal{A}_t$ gets no information about s.*

Remark 2 (Difference between AWCS and AWSS) *In the first look, the properties of AWCS looks similar to AWSS. However, there is a conceptual difference. In AWSS, even if D is corrupted and does not participate in the reconstruction phase of AWSS, the reconstruction phase will terminate if all the honest parties participate in the reconstruction phase. However, this is not the case for AWCS. In AWCS, it is D who has to invoke the decommit phase and if he is corrupted then he may not do so, in which case the decommit phase will not terminate. This means that AWSS has more stringent requirements than AWCS. Also in AWSS, all the honest parties participate to reconstruct D's committed secret. On the other hand, in AWCS, D is asked to reveal his committed secret.*

We now present our AWCS scheme with $n = 3t + 1$, consisting of protocols **Com** and **DeCom**. The protocol is given for a single secret. However, it can be extended easily for ℓ secrets. We first give an informal description of the protocol. During **Com**, D generates Shamir shares [21] of secret s. Then to each party P_i, D gives the share s_i and ask P_i to give his IC signature on s_i to D. Due to asynchronity of the network, D cannot wait to receive IC signature from all n parties. So D stops as soon as he gets IC signature on s_i's from $2t + 1$ P_i's. Now here D ensures that he receives the IC signature from P_i on the same s_i which D has given to P_i. As soon as D receives $2t + 1$ IC signatures, he A-cast the identity of the $2t + 1$ parties from whom he has received the IC signature. Now every party verifies that D has indeed received $2t + 1$ IC signatures from these $2t + 1$ parties (one from each party) and if so, they terminate protocol **Com**. In short, the goal of the **Com** protocol is to allow D to get IC signatures from $2t + 1$ parties on respective

Shamir shares. Notice that if D is *corrupted*, then he may not generate Shamir shares and instead can get IC signatures on any value from respective parties. This is because no P_i can verify whether s_i (on which he is asked to give his IC signature) is a valid Shamir share of some secret. There is no checking mechanism for this in **Com**. However still our protocol is an AWCS scheme.

Now during **DeCom**, D is asked to reveal $2t+1$ IC signatures on $2t+1$ values, which he has received during **Com**. Now the parties verify whether the $2t+1$ values revealed by D are valid Shamir shares. If yes, then the secret corresponding to these shares is accepted as authentic. Otherwise D is discarded. Now our AWCS scheme is formally given in Fig. 1. We now prove the properties of our AWCS scheme.

Figure 1: AWCS Scheme for $n = 3t+1$ with Error Parameter ϵ

Protocol Com(s, ϵ)

SHARE GENERATION : CODE FOR D – Only D executes this code:

1. Select a random degree-t polynomial $f(x)$ over $\mathbb{F}$, such that $f(0) = s$.
2. Privately send $f(i)$ to party P_i, for $i = 1, \ldots, n$ and request P_i to give his IC signature on s_i to D.

SIGNING THE SHARES : CODE FOR P_j – Every party including D executes this code

1. Wait to receive s_j from D. Then give $ICSig(P_j, D, \mathcal{P}, s_j, \epsilon')$ to D by acting as a dealer and considering D as INT, where $\epsilon' = \frac{\epsilon}{n}$.
2. Once $ICSig(P_j, D, \mathcal{P}, s_j, \epsilon')$ is received by D then A-cast a signal `Signature-Sent-From-`P_j.

PREPARING FOR THE COMMITMENT : CODE FOR D — Only D executes this code

1. If $ICSig(P_j, D, \mathcal{P}, s'_j, \epsilon')$ is received from P_j and if $s'_j = s_j$, then add P_j to a list WCORE, which is initially $\emptyset$.
2. Wait till $|\mathsf{WCORE}| = 2t+1$. Then A-cast a signal `Commitment-Over` and the set WCORE.

VERIFICATION OF D'S CLAIM : CODE FOR P_j — Every party including D executes this code

1. If `Commitment-Over` and WCORE are received from D's A-cast where $|\mathsf{WCORE}| = 2t+1$, then wait to receive `Signature-Sent-From-`P_i signal from the A-cast of every $P_i \in$ WCORE.
2. If `Signature-Sent-From-`P_i signals are received from A-cast of each $P_i \in$ WCORE then conclude that D's commitment is over and terminate **Com**.

Protocol DeCom(s, ϵ)

OPENING THE COMMITMENT : CODE FOR D — Only D executes this code

1. Corresponding to each $P_i \in$ WCORE, reveal $ICSig(P_i, D, \mathcal{P}, s_i, \epsilon')$.

VERIFICATION OF THE COMMITMENT : CODE FOR P_j — Every party including D executes this code

1. Wait to complete revelation of $ICSig(P_i, D, \mathcal{P}, s_i, \epsilon')$ corresponding to every $P_i \in$ WCORE.
2. If $Reveal_j = NULL$, corresponding to any $ICSig(P_i, D, \mathcal{P}, s_i, \epsilon')$ for any $P_i \in$ WCORE, then *discard* D and terminate **DeCom**.
3. If $Reveal_j = \overline{s_i}$, corresponding to every $ICSig(P_i, D, \mathcal{P}, s_i, \epsilon')$ for every $P_i \in$ WCORE, then check whether all $\overline{s_i}$'s lie on a degree-t polynomial. If not then *discard* D and terminate **DeCom**. Otherwise let $\overline{s_i}$'s lie on degree-t polynomial $\overline{f(x)}$. Then accept $\overline{s} = \overline{f(0)}$ as *authentic* and terminate **DeCom**.

Lemma 1 (Termination) *Protocols* **(Com, DeCom)** *satisfies* **Termination** *condition of Def. 5.*

PROOF (SKETCH): **Termination 1**: It is easy to see that if D is honest then eventually all honest P_i's will give $ICSig(P_i, D, s'_i, \mathcal{P})$ to D with $s'_i = s_i$ and there are at least $2t+1$ honest parties. The rest follows from the protocol steps.

Termination 2: Let P_j be an honest party who has terminated **Com**. This implies that P_j must have received WCORE and `Commitment-Over` signal from the A-cast of D, where $|\mathsf{WCORE}| = 2t+1$. In addition, P_j also must have received `Signature-Sent-From-`P_i signal from every $P_i \in$ WCORE. Now from the properties of A-cast, every other honest party will also eventually receive these signals and terminate **Com**.

Termination 3: Suppose D invokes **DeCom**. This implies that corresponding to each $P_i \in$ WCORE, D reveals $ICSig(P_i, D, s_i, \mathcal{P})$. So eventually all the signatures will be revealed and either D will be discarded or some secret will accepted as authentic. But whatever may be the case, every honest party will eventually terminate **DeCom**. □

Lemma 2 (Secrecy) *Protocol* **Com** *satisfies* **Secrecy** *condition of Def. 5.*

PROOF (SKETCH): If D is *honest* then $\mathcal{A}_t$ will get at most t shares of s during **Com**. The rest follows from the properties of Shamir secret sharing [21] and secrecy property of **AICP**. □

Lemma 3 (Correctness) *Protocols* **(Com, DeCom)** *satisfies* **Correctness** *condition of Def. 5.*

PROOF: **Correctness 1**: Here we have to consider an *honest* D. Without loss of generality, let $P_1, \ldots, P_{2t+1}$ be the $2t+1$ parties in WCORE. Then D as INT has received $ICSig(P_i, D, \mathcal{P}, s_i)$ from P_i, for $i = 1, \ldots, 2t+1$. So during **DeCom**, D as INT will be able to reveal each IC signature successfully, except with error probability ϵ' (follows from **AICP-Correctness2**). Thus, except with error probability $(2t+1) \cdot \epsilon' \approx \epsilon$, all s_i's will be correctly revealed by honest D. Moreover, all s_i's will lie on $f(x)$. Thus $s = f(0)$ will be accepted as authentic except with error probability ϵ.

Correctness 2: Here we have to consider a *corrupted* D. If D is corrupted and some honest party, say P_j, has terminated Com, then it implies that P_j has received WCORE from A-cast of D, where $|\mathsf{WCORE}| = 2t+1$. In addition, P_j also must have received `Signature-Sent-From-`P_i signals from every $P_i \in$ WCORE. Now there can be *corrupted* P_i's in WCORE, who may not have given their IC signature on their respective shares to D and still may have A-casted the signal `Signature-Sent-From-`P_i. However, the *honest* P_i's in WCORE must have given their IC signature on their respective shares to D and then only they have A-casted the signal `Signature-Sent-From-`P_i. Also note that there are at least $t+1$ honest parties in WCORE. Now we define D's committed value as follows: *if the s'_i values corresponding to honest parties in WCORE lie on a degree-t polynomial, say $f^\star(x)$, then D's committed value $s^\star = f^\star(0)$. Otherwise, D's committed value $s^\star = NULL$.* Now during **DeCom**, D has to reveal $ICSig(P_i, D, \mathcal{P}, s'_i)$, corresponding to every $P_i \in$ WCORE. Since there are at least $t+1$ honest parties in WCORE, from **AICP-Correctness3** property, except with error probability $(t+1) \cdot \epsilon' \approx \epsilon$, D will not be able to reveal $ICSig(P_i, D, \mathcal{P}, s''_i)$, where $s''_i \neq s'_i$, corresponding to any *honest* P_i in WCORE. So if some value s' is accepted as authentic, then it implies that all the $2t+1$ values revealed by D lie on a degree-t polynomial. This further implies that $s' = s^\star$, except with error probability ϵ. This is because there can be only one degree-t polynomial, namely $f^\star(x)$, passing through the s'_i's corresponding to honest parties in WCORE. □

Theorem 2 *Protocols* **(Com, DeCom)** *constitutes a valid AWCS scheme, which invokes n instances of AICP scheme, each dealing with a single value.*

PROOF: The proof that **(Com, DeCom)** constitutes a valid AWCS scheme follows from Lemma 1, 2 and 3. It is easy to see that in the AWCS scheme, there are n instances of **Gen**, **Ver** and **Reveal-Public**. Thus there are n instances of AICP scheme used in the AWCS scheme. □

3.1 AWCS for Dealing with ℓ Secrets

The AWCS scheme given in the previous section deals with only one value. If we want to design an AWCS scheme to deal with ℓ values, then we can parallely execute the previous scheme ℓ times, once for each of the ℓ values. However, this will require $\ell \cdot n = \ell n$ invocations of AICP scheme. We now informally discuss how we can design an AWCS scheme, dealing with all the ℓ values *concurrently* and using only n instances of AICP. The formal details will appear in the full version of the paper.

The idea is as follows: during commit phase, for each of the ℓ secrets, D will generate Shamir shares and will give the i^{th} share of all the ℓ secrets to P_i and ask P_i to give his IC signature on all the ℓ i^{th} shares to D. Now instead of executing ℓ instances of **Gen**, to give his signature on ℓ individual i^{th} share, P_i will execute *only one* instance of **Gen** and will give a *single* IC signature on all the ℓ i^{th} shares *collectively*. Consequently, D will invoke *only one* instance of **Ver** to ensure that it has received a valid IC signature from P_i on ℓ i^{th} shares collectively. The rest of the steps like WCORE construction, etc will be same as in protocol **Com**. Similarly, during decommit phase, D will reveal a single IC signature on all ℓ i^{th} shares collectively, corresponding to each $P_i \in$ WCORE. We do not present the complete details here due to space constraints and state only the following theorem. The details will appear in the full version of the paper.

Theorem 3 *Let $n = 3t+1$. Then there exists an AWCS scheme dealing with ℓ secrets concurrently, which invokes n instances of AICP scheme, each dealing with ℓ values.*

3.2 Another View of Our AWCS Scheme

We now present another view of protocols **(Com, DeCom)**, which will be useful while using them as a black-box in our AVSS scheme. Recall that in protocol **Com**, D is asked to select a random degree-t polynomial and get from each party P_i his IC signature on the i^{th} share $s_i = f(i)$. We can view this entire computation as if D is trying to commit a degree-t polynomial $f(x)$ where $f(0) = s$. So in the sequel, we will call protocol **Com** as **Com**$(f(x), \epsilon)$, instead of **Com**(s, ϵ). The steps of the protocol will be exactly same as presented earlier. Similarly, we will call protocol **DeCom** as **DeCom**$(f(x), \epsilon)$, where D decommits $f(x)$.

Here either $f(x)$ and hence $f(0) = s$ is accepted as authentic or otherwise D is discarded. A similar view can be interpreted for our AWCS scheme dealing with ℓ values concurrently. The details will appear in the full version of the paper.

4 Designing AVSS Using AWCS

We now show how to use AWCS as a substitute for AWSS in the existing AVSS schemes of [18] and [16]. However, due to space constraints, we do the same only for the Weak-AVSS scheme of [18] sharing a single value. In the full version of the paper, we will show how AWCS can be used in place of AWSS in the Strong-AVSS scheme of [16].

The high level idea of the Weak-AVSS scheme of [18] is as follows: D selects a random, symmetric bivariate polynomial $F(x, y)$ of degree-t in x and y, where $F(0, 0) = s$ and gives $f_i(x) = F(x, i)$ to party P_i, for $i = 1, \ldots, n$. Party P_i is then asked to AWSS-Share his received polynomial $f_i(x)$ and so on. Now instead of asking P_i to AWSS-Share his $f_i(x)$, we ask P_i to commit his $f_i(x)$ using our AWCS scheme. In addition to this, we add few additional steps in the Weak-AVSS protocol of [18]. The new Weak-AVSS scheme is given in Fig. 2. *In the protocol, all the steps in italics are the new steps, which are added in the AVSS scheme of [18] so that it can use our AWCS scheme, instead of AWSS scheme of [18].*

Due to space constraints, we only state the following theorem. The details will appear in the full version of the paper.

Theorem 4 *The scheme presented in Fig. 2 is a valid statistical Weak-AVSS scheme, which uses n invocations of AWCS scheme and hence n^2 instances of AICP scheme.*

5 Conclusion

In this article, we made a positive step towards reducing the communication complexity of statistical AVSS with optimal resilience. For this, we presented a new asynchronous primitive called AWCS, which can be used as a substitute for AWSS which is used in all existing efficient statistical AVSS schemes. We find that AWCS has weaker requirements than AWSS and hence it is conceptually simpler to design AWCS than AWSS. Our AWCS when further used in existing efficient AVSS schemes of [18] and [16] reduces their communication complexity by a factor of $\Omega(n)$.

References

[1] Z. Beerliová-Trubíniová and M. Hirt. Simple and efficient perfectly-secure asynchronous MPC. In *ASIACRYPT*, LNCS 4833, pages 376–392, 2007.

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

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

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

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

[6] R. Canetti and T. Rabin. Fast asynchronous Byzantine Agreement with Optimal Resilience. In *STOC*, pages 42–51. ACM, 1993.

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

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

[9] R. Cramer, I. Damgård, and U. M. Maurer. General secure multi-party computation from any linear secret-sharing scheme. In *EUROCRYPT*, LNCS 1807, pages 316–334, 2000.

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

[11] 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.

[12] J. Katz, C. Koo, and R. Kumaresan. Improving the round complexity of vss in point-to-point networks. In *ICALP(2)*, LNCS 5126, pages 499–510, 2008.

Figure 2: Weak-AVSS Scheme for Sharing Single Secret s with Error Parameter ϵ

Protocol AVSS-Share(s, ϵ)

DISTRIBUTION: CODE FOR D — Only D executes this code

1. Select a random symmetric bivariate polynomial $F(x, y)$ of degree-t in x and y such that $F(0, 0) = s$ and send $f_i(x) = F(x, i)$ to party P_i, for $i = 1, \ldots, n$.

AWCS COMMITMENT OF $f_i(x)$: CODE FOR P_i — Every party, including D executes this code

1. Wait to obtain $f_i(x)$ from D.
2. *If $f_i(x)$ is a degree-t polynomial then invoke* **Com($f_i(x), \epsilon'$)** *where $\epsilon' = \frac{\epsilon}{n}$. We call this instance of* **Com** *initiated by P_i as* $\mathbf{Com}^{P_i}$*. As a part of* $\mathbf{Com}^{P_i}$*, perform the following additional step: for $j = 1, \ldots, n$, give $ICSig(P_i, P_j, \mathcal{P}, f_i(j))$ as a dealer to P_j as INT.*
3. As a part of the execution of $\mathbf{Com}^{P_j}$, wait to receive $ICSig(P_j, P_i, \mathcal{P}, f_j(i))$ from P_j. Then check whether $f_j(i) \stackrel{?}{=} f_i(j)$. If the test passes *then only* participate in $\mathbf{Com}^{P_j}$ and act according to the remaining steps of $\mathbf{Com}^{P_j}$.

VCORE CONSTRUCTION : CODE FOR D — Only D executes this code

1. If $\mathbf{Com}^{P_j}$ is terminated, then denote the corresponding WCORE by WCORE^{P_j}. Add P_j in a set VCORE (initially empty).
2. Keep updating VCORE and WCORE^{P_j} for every $P_j \in$ VCORE upon receiving new A-casts of the form `Signature-Sent-From-`P_i (during $\mathbf{Com}^{P_j}$s), until for at least $2t + 1$ $P_j \in$ VCORE, the condition $|\mathsf{VCORE} \cap \mathsf{WCORE}^{P_j}| \geq 2t + 1$ is satisfied. Remove (from VCORE) all $P_j \in$ VCORE for whom the above condition is not satisfied.
3. A-cast VCORE, $n' = |\mathsf{VCORE}|$ and WCORE^{P_j} for every $P_j \in$ VCORE.

VCORE VERIFICATION & AGREEMENT ON VCORE: CODE FOR P_i — Every party executes this code

1. Wait to receive VCORE, $n' = |\mathsf{VCORE}|$ and WCORE^{P_j} for every $P_j \in$ VCORE from D's A-cast.
2. Wait to terminate $\mathbf{Com}^{P_j}$ corresponding to every $P_j \in$ VCORE, with WCORE^{P_j} as the corresponding WCORE. After terminating the desired $\mathbf{Com}^{P_j}$'s, consider VCORE and WCORE^{P_j} for every $P_j \in$ VCORE received from D as valid, accept them and terminate AVSS-Share.

Protocol AVSS-Rec(s, ϵ)

AWCS DECOMMITMENT OF $f_i(x)$: CODE FOR P_i — Every party executes this code

1. If $P_i \in$ VCORE, then execute **DeCom($f_i(x), \epsilon'$)**. Denote this instance of **DeCom** as $\mathbf{DeCom}^{P_i}$.
2. Participate in **DeCom($f_j(x), \epsilon'$)** corresponding to every $P_j \in$ VCORE.
3. *If $P_i \in \mathsf{WCORE}^{P_j}$ for any $P_j \in$ VCORE, then execute* **Reveal-Public**$(P_j, P_i, \mathcal{P}, f_j(i), \epsilon')$.
4. *As a verifier, participate in all instances of* **Reveal-Public**$(P_j, P_k, \mathcal{P}, f_j(k), \epsilon')$*, corresponding to any $P_k \in \mathsf{WCORE}^{P_j}$, where $P_j \in$ VCORE.*

SECRET RECONSTRUCTION: CODE FOR P_i — Every party executes this code

1. *Initialize a set $\mathsf{FINAL}_i = \emptyset$. Add $P_j \in$ VCORE in FINAL_i if in $\mathbf{DeCom}^{P_j}$, some degree-t polynomial, say $f'_j(x)$ is accepted as authentic.*
2. Wait till $|\mathsf{FINAL}_i| = |\mathsf{VCORE}| - t$.
3. For every pair $(P_\gamma, P_\delta) \in \mathsf{FINAL}_i$ check $f'_\gamma(\delta) \stackrel{?}{=} f'_\delta(\gamma)$. If not, then output $NULL$ and terminate AVSS-Rec.
4. *If $f'_\gamma(\delta) = f'_\delta(\gamma)$ for every pair $(P_\gamma, P_\delta) \in \mathsf{FINAL}_i$, then construct bivariate polynomial $F'_i(x, y)$ of degree-t in x and y, using $f'_j(x)$'s corresponding to each $P_j \in \mathsf{FINAL}_i$. Then do the following*:
   (a) *Check whether there exists any $P_k \in \mathsf{FINAL}_i$, such that $P_k \in \mathsf{WCORE}^{P_j}$ of any $P_j \in \mathsf{VCORE} \setminus \mathsf{FINAL}_i$.*
   (b) *If there are any such P_k's, then wait till* **Reveal-Public**$(P_j, P_k, \mathcal{P}, f_j(k), \epsilon')$ *is complete with non-$NULL$ output, say $Reveal_i = \overline{f_j(k)}$.*
   (c) *Check whether all the revealed $\overline{f_j(k)}$'s of previous step (if any) lie on $F'_i(x, y)$. If not, then output $NULL$ and terminate AVSS-Rec. Otherwise, output $F'_i(0, 0)$ and terminate AVSS-Rec.*

[13] R. Kumaresan, A. Patra, and C. Pandu Rangan. The round complexity of verifiable secret sharing: The statistical case. In *ASIACRYPT*, LNCS 6477, pages 431–447, 2010.

[14] A. Patra. Studies on verifiable secret sharing, byzantine agreement and multiparty computation. Cryptology ePrint Archive, Report 2010/280, 2010.

[15] A. Patra, A. Choudhary, T. Rabin, and C. Pandu Rangan. The Round Complexity of Verifiable Secret Sharing Revisited. In *CRYPTO*, LNCS 5677, pages 487–504, 2009.

[16] A. Patra, A. Choudhary, and C. Pandu Rangan. Efficient asynchronous multiparty computation with optimal resilience. Cryptology ePrint Archive, Report 2008/425, 2008. A preliminary version of this paper appeared in *ICITS*, LNCS 5973, pages 74–92, 2009.

[17] A. Patra, A. Choudhary, and C. Pandu Rangan. Communication Efficient Perfectly Secure VSS and MPC in Asynchronous Networks with Optimal Resilience. In *AFRICACRYPT*, LNCS 6055, pages 184–202, 2010.

[18] A. Patra, A. Choudhury, and C. Pandu Rangan. Efficient asynchronous byzantine agreement with optimal resilience. Cryptology ePrint Archive, Report 2008/424, 2008. A preliminary version of this paper appeared in *PODC*, pages 92–101, 2009.

[19] A. Patra and C. Pandu Rangan. Brief announcement: communication efficient asynchronous byzantine agreement. In *PODC*, pages 243–244, 2010.

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

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