Simple and Efficient Asynchronous 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 parties where every two parties 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 parties in Byzantine fashion. In these settings, we say that π is a t-resilient, $(1-\epsilon)$-terminating *Asynchronous Byzantine Agreement* (ABA) protocol, if π satisfies all the properties of Byzantine Agreement (BA) in asynchronous settings tolerating $\mathcal{A}_t$ and terminates (i.e every honest party terminates π) with probability at least $(1-\epsilon)$. In this work, we present a new t-resilient, $(1-\epsilon)$-terminating ABA protocol which *privately* communicates $\mathcal{O}(\mathcal{C}n^5\kappa)$ bits and A-casts[1] $\mathcal{O}(\mathcal{C}n^5\kappa)$ bits, where $\epsilon = 2^{-\mathcal{O}(\kappa)}$ and $\mathcal{C}$ is the *expected running time* of the protocol. Moreover, conditioned on the event that our ABA protocol terminates, it does so in constant expected time; i.e., $\mathcal{C} = \mathcal{O}(1)$. Our ABA protocol is to be compared with the *only known* t-resilient, $(1-\epsilon)$-terminating ABA protocol of [5] in the same settings, which *privately* communicates $\mathcal{O}(\mathcal{C}n^{11}\kappa^4)$ bits and A-casts $\mathcal{O}(\mathcal{C}n^{11}\kappa^2\log(n))$ bits, where $\epsilon = 2^{-\mathcal{O}(\kappa)}$ and $\mathcal{C} = \mathcal{O}(1)$. So our ABA achieves a huge gain in communication complexity in comparison to the ABA of [5], while keeping all other properties in place. In another landmark work, in PODC 2008, Abraham et. al [1] proposed a t-resilient, 1-terminating (called as *almost-surely terminating* in [1]) ABA protocol which communicates $\mathcal{O}(\mathcal{C}n^6\log n)$ bits and A-casts $\mathcal{O}(\mathcal{C}n^6\log n)$ bits. But ABA protocol of Abraham et. al. takes polynomial ($\mathcal{C} = \mathcal{O}(n^2)$) expected time to terminate. Hence the merits of our ABA protocol over the ABA of Abraham et. al. are: (i) For any $\kappa < n^3 \log n$, our ABA is better in terms of communication complexity (ii) conditioned on the event that our ABA protocol terminates, it does so in constant expected time (the constant is independent of n, t and κ), whereas ABA of Abraham et. al. takes polynomial expected time. However, it should be noted that our ABA is only $(1-\epsilon)$-terminating whereas ABA of Abraham et al. is *almost-surely terminating*. Summing up, in a practical scenario where a faster and communication efficient ABA protocol is required, our ABA fits the bill better than ABA protocols of [5, 1].

For designing our ABA protocol, we present a novel and simple *asynchronous verifiable secret sharing* (AVSS) protocol which significantly improves the communication complexity of the only known AVSS protocol of [5] in the same settings. We believe that our AVSS can be used in many other applications for improving communication complexity and hence is of independent interest.

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

[1] A-Cast is a primitive in asynchronous world, allowing a party to send the same value to all the other parties. Hence A-Cast in asynchronous world is the parallel notion of broadcast in synchronous world.

1 Introduction

The problem of reaching agreement in the presence of faults is one of the most fundamental problems in distributed computing. A particularly interesting variant of this problem, called Byzantine Agreement (BA) was introduced by Pease et.al [16] in 1980. Roughly speaking, the BA problem is as follows: there are n parties, each having an input binary value; the goal is for all honest parties to agree on a consensus value that is the input value of one of the honest parties. The challenge lies in reaching agreement despite the presence of faulty parties, who may deviate from the protocol arbitrarily.

Since 1980, the BA problem has been investigated extensively in various models, characterized by the synchrony of the network, privacy of the channels, computational power of the faulty parties and many other parameters. The interested reader may go through [10, 6, 4, 15] to survey the long chain of works carried out in this area. An interesting and practically motivated variant of BA is *Asynchronous* BA (ABA) tolerating *unbounded powerful* malicious adversary. In ABA, the communication channels between the parties have arbitrary, yet finite delay (i.e the messages are guaranteed to reach eventually) and an adversary $\mathcal{A}_t$ with *unbounded computing power* can control at most t of n parties in Byzantine fashion. By Byzantine we mean that the adversary can take full control of the party and force it to behave arbitrarily during the protocol execution. It is known that ABA tolerating $\mathcal{A}_t$ is possible iff $n \geq 3t + 1$ [15, 16] (this is true even in synchronous network). Any ABA protocol designed with $n = 3t + 1$ is therefore called as *optimally resilient*. By the seminal result of [11], any optimally resilient ABA protocol must have *non-terminating* runs, where some honest party(ies) may not output any value. So we say an ABA protocol to be $(1 - \epsilon)$-terminating, if in the protocol, each honest party eventually terminates with probability $(1 - \epsilon)$, where ϵ is the probability of not terminating. As a special case, we call an ABA protocol to be *almost-surely terminating*, a term coined by Abraham et. al in [1], if the set of non-terminating executions in the protocol has *probability zero*. The BA problem has been studied extensively over the past three decades and tight bounds have been established on resilience, round complexity and communication complexity for several variants of the problem in synchronous settings (see [15]). However, very little attention has been paid in designing fast, communication efficient, optimally resilient ABA protocol. Our results in this paper marks a significant progress in this direction.

Existing Results: Bracha [3] provides an optimally resilient, almost-surely terminating ABA protocol, which runs in $\Theta(2^n)$ expected time and requires exponential communication (message) complexity. Feldman and Micali [8] present an almost-surely terminating ABA protocol which runs in polynomial time and requires polynomial communication complexity (they actually extend their BA protocol in synchronous settings [8] to asynchronous settings). However, the ABA protocol of Feldman and Micali [8] is not optimally resilient (requires $4t + 1$ parties). So it remained an open question whether there exists an optimally resilient ABA with polynomial running time and communication complexity. Canetti and Rabin [5, 4] answered this question in affirmative and provided an ABA protocol which is optimally resilient, runs in expected *constant* time and incurs polynomial communication complexity. But Canetti and Rabin's ABA protocol is $(1 - \epsilon)$-terminating and is not almost-surely terminating. For the first time in the literature, Abraham et. al. [1] provides an optimally resilient, almost-surely terminating ABA protocol which runs in polynomial time and requires polynomial communication complexity. Nevertheless, Canetti and Rabin's ABA protocol achieves an important property which is not achieved by any of the previously mentioned protocols, namely the constant expected running time. Though the protocols of Canetti and Rabin [5, 4] and Abraham et. al. [1] provide polynomial communication complexity, they involve fairly very high communication complexity. So, designing optimally resilient, communication efficient ABA protocol which runs in constant expected time is a natural next important and interesting problem. So, in this paper we focus on designing optimally resilient ABA which is communication efficient as well as runs in expected constant time. Generally, in a distributed network, fast and communication efficient protocols find lot of application. Our ABA protocol, though $(1 - \epsilon)$-terminating, is fast (in a sense that it runs in constant expected time) and communication efficient (in a sense that it provides the best known communication complexity so far).

Our Contribution: In this work, we present an optimally resilient, $(1 - \epsilon)$-terminating ABA pro-

tocol which privately communicates $\mathcal{O}(\mathcal{C}n^5\kappa)$ bits and A-casts $\mathcal{O}(\mathcal{C}n^5\kappa)$ bits, where $\epsilon = 2^{-\mathcal{O}(\kappa)}$ and $\mathcal{C}$ is the *expected running time* of the protocol. Moreover, conditioned on the event that our ABA protocol terminates, it does so in constant expected time; i.e., $\mathcal{C} = \mathcal{O}(1)$. Our ABA protocol is to be compared with the *only known* optimally resilient, $(1-\epsilon)$-terminating ABA protocol of [5] with same properties, which privately communicates $\mathcal{O}(\mathcal{C}n^{11}\kappa^4)$ bits and A-casts $\mathcal{O}(\mathcal{C}n^{11}\kappa^2\log(n))$ bits, where $\epsilon = 2^{-\mathcal{O}(\kappa)}$ and $\mathcal{C} = \mathcal{O}(1)$. So our ABA achieves a huge gain in communication complexity in comparison to the ABA of [5], while keeping all other properties in place. The optimally resilient, almost-surely terminating ABA protocol of Abraham et. al [1] privately communicates $\mathcal{O}(\mathcal{C}n^6\log n)$ bits and A-casts $\mathcal{O}(\mathcal{C}n^6\log n)$ bits. But as mentioned earlier, ABA protocol of Abraham et. al. takes polynomial ($\mathcal{C} = \mathcal{O}(n^2)$) expected time to terminate. Hence our ABA enjoys the following merits over the ABA of Abraham et. al.: (i) For any $\kappa < n^3\log n$, our ABA is better in terms of communication complexity (ii) our ABA runs in constant expected time. However, we stress that our ABA is only $(1-\epsilon)$-terminating whereas ABA of Abraham et al. is *almost-surely terminating*. So in a practical scenario where an usually fast and communication efficient ABA protocol is required, our ABA fits the bill more appropriately than any other existing ABA protocols.

Our construction of ABA protocol, employs an *asynchronous verifiable secret sharing* (AVSS) scheme with $n = 3t+1$. Roughly speaking, an AVSS scheme consists of a sharing phase and reconstruction phase. Informally, the goal of the AVSS scheme is to share a secret among n parties during the sharing phase in a way that would later allow for *unique* reconstruction of this secret in the reconstruction phase. In many applications one treats AVSS as a form of *commitment*, where the commitment information is held in a distributed fashion among the parties. In this paper we present a novel and simple AVSS protocol which is far better in terms of communication complexity than the previously reported AVSS protocol of [5] in the same setting and having the same properties. We believe that our AVSS and the tools we use for designing AVSS can be used in many other applications for improving communication complexity and hence are of independent interest.

A Brief Discussion on the Approaches used in the ABA Protocols of [5] and [1] and Cur rent Article: Almost all the ABA protocols used Bracha's [3] idea of reducing the problem of ABA to that of implementing a *shared coin*. Feldman and Micali [8, 9] reduced the problem of efficiently implementing a shared coin to that of efficiently implementing AVSS. Roughly speaking, the secrets committed by AVSS are used to generate a shared coin. For a comprehensive account on the reduction from AVSS to ABA, reader may refer Canetti's Thesis [4].

1. The ABA protocol of Canetti and Rabin [5, 4] also uses the reduction from AVSS to ABA. Hence they have first designed an AVSS scheme with $n = 3t+1$ parties. There are well known inherent difficulties in designing an AVSS scheme with $n = 3t+1$ (as clearly pointed out in [5, 4]). To overcome these difficulties, the authors in [5] used the following approach to design their AVSS scheme. They first designed a tool called *Information Checking Protocol* (ICP), which was introduced by Rabin et al. in [18] (over a synchronous network). Then a protocol called *Asynchronous Recoverable Sharing* (A-RS) was designed using ICP as a black box. Subsequently, using A-RS as a primitive, the authors have designed another well known 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*, an AVSS scheme was presented. Thus pictorially, the route taken by [5] to design their AVSS scheme is as follows:*ICP* $\rightarrow$ *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 becomes highly communication intensive as well as very much involved. The AVSS protocol of [5] requires private communication of $\mathcal{O}(n^9\kappa^4)$ bits, A-Cast $\mathcal{O}(n^9\kappa^2\log(n))$ bits during *sharing phase* and private communication of $\mathcal{O}(n^6\kappa^3)$ 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 $1-2^{-\mathcal{O}(\kappa)}$. The communication complexity analysis of the AVSS protocol of [5] is not done earlier and for the sake of completeness, we carry out the same in **Appendix A**.

2. The ABA protocol of Abraham et al. [1] used the same reduction from AVSS to ABA as in [5], except that the use of AVSS is replaced by a variant of AVSS that the authors called *shunning* (asynchronous) VSS (SVSS), where each party is guaranteed to terminate *almost-surely*. SVSS is a

slightly weaker notion of AVSS in the sense that if all the parties (including corrupted parties) behave correctly then SVSS satisfies the properties of AVSS *perfectly* (i.e without any error probability). Otherwise it does not satisfy the properties of AVSS but will enable some honest party to identify at least one corrupted party, whom the honest party shuns from then onwards. In order to implement the SVSS protocol, the authors in [1] use a weaker (than SVSS) protocol called *moderated weak shunning* (asynchronous) VSS (MW-SVSS). The use of SVSS instead of AVSS in generating shared coin causes the ABA protocol of [1] to run for $\mathcal{O}(n^2)$ expected time. The SVSS protocol requires private communication of $\mathcal{O}(n^6 \log(n))$ bits and A-Cast of $\mathcal{O}(n^6 \log(n))$ bits. The description of MW-SVSS and SVSS is fairly simple and hence their communication complexity analysis can be done easily.

3. Our ABA protocol also follows the same reduction from AVSS to ABA as in [5]. In the course of designing our ABA protocol, our first step is to design a communication efficient AVSS protocol. Instead of following a fairly complex route taken by [5] to design an AVSS scheme, we follow a shorter route for designing our AVSS: *ICP* $\rightarrow$ *AWSS* $\rightarrow$ *AVSS*. Beside this, we significantly improve each of the building blocks (underlying primitives) of our ABA protocol, namely ICP, AWSS and AVSS. The improvements are contributed by key factors like: (a) new design approach, (b) harnessing the advantages offered by dealing with multiple secrets *concurrently*. To emphasize on the second factor, we remark that our protocols dealing with ℓ secrets *concurrently* are far better than ℓ repeated applications of protocols dealing with single secret. Together, they lead to our efficient AVSS and ABA protocols which we believe are much simpler than the AVSS and ABA of [5].

2 Network Model, Definitions and Notations

Model: We follow the asynchronous network model of [5], where there are n parties, denoted by the set $\mathcal{P} = \{P_1, \ldots, P_n\}$, who are connected by pairwise reliable and secure channel. An adversary $\mathcal{A}_t$ with *unbounded computing power* can corrupt at most t parties during the protocol in Byzantine fashion. Once a party is corrupted, he remains so throughout the protocol. Messages sent on a channel may have arbitrary (but finite) delays. To model this, 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 parties 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 $2^{-\mathcal{O}(\kappa)}$. To bound the error probability by $2^{-\mathcal{O}(\kappa)}$, 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 where without loss of generality, $\kappa = poly(n)$.

Asynchronous Weak Secret Sharing (AWSS) [5]: Let (Sh, Rec) be a pair of protocols in which a dealer $D \in \mathcal{P}$ shares a secret $S = (s^1 \ldots s^\ell) \in \mathbb{F}^\ell$ containing $\ell \geq 1$ field elements. We say that (Sh, Rec) is a t-resilient AWSS scheme for n parties if the following hold for every possible $\mathcal{A}_t$.

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

1. If D is honest then each party will eventually terminate protocol Sh.
2. If some honest party has completed protocol Sh, then irrespective of the behavior of D, each honest party will eventually terminate Sh.
3. If at least one honest party has completed Sh and if all the honest parties invoke protocol Rec, then each honest party will eventually terminate Rec.

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

1. If D is *honest* then each honest party upon completing protocol Rec, outputs the shared secret S.
2. If D is *faulty* and some honest party has completed Sh, then there exists a unique $S' = (s'^1 \ldots s'^\ell) \in (\mathbb{F} \cup NULL)^\ell$, such that each honest party upon completing Rec, will output either S' or $(NULL)^\ell$. Here $(NULL)^\ell$ means an ℓ tuple where each entry is $NULL$.

• **Secrecy**: If D is honest and no honest party has begun executing protocol Rec, then the corrupted parties have no information about the shared secret.

As mentioned in [5], we stress that in the case of a *corrupted* D, even if $S' \neq (NULL)^\ell$, it may happen

that some honest parties output S' and some may output $(NULL)^\ell$. The adversary can decide which parties will output $(NULL)^\ell$ during the execution of Rec.

Asynchronous Verifiable Secret Sharing (AVSS) [5]: The **Termination** and **Secrecy** conditions for AVSS is same as in AWSS. The only difference is in the **Correctness** 2 property:

Correctness 2: If D is *faulty* and some honest party has completed Sh, then there exists a unique $S' = (s'^1 \dots s'^\ell) \in (\mathbb{F} \cup NULL)^\ell$, such that with probability at least $1 - 2^{-\mathcal{O}(\kappa)}$, each honest party upon completing Rec, will output *only* S'.

The difference between the **Correctness 2** property of AWSS and AVSS is that in AWSS, when $S' \neq (NULL)^\ell$, then D may change the committed secret from S' to $(NULL)^\ell$, while in AVSS, D cannot change his commitment from S' to any other value during Rec protocol.

Asynchronous Byzantine Agreement (ABA)[5]: Let Π be an asynchronous protocol in which each party has a binary input and let $\kappa > 0$ be the error parameter. We say that Π is a $(1 - 2^{-\mathcal{O}(\kappa)})$-terminating, t-resilient Byzantine Agreement protocol if the following hold, for every possible $\mathcal{A}_t$:

• **Termination**: With probability at least $1 - 2^{-\mathcal{O}(\kappa)}$ all honest parties terminate the protocol.

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

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

• **Termination**:

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

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

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

From [3], A-Cast of b bits incurs a private communication of $\mathcal{O}(n^2 b)$ bits. Notice that the termination property of A-Cast is much weaker than the termination property of ABA because for A-Cast, it is not required that the honest parties complete the protocol when the sender is faulty. In the sequel, we use the following convention: *we say that P_j listens the A-Cast of P_i with value m, indicating that P_j completes the execution of P_i's A-Cast and has locally set the value of the A-Cast to m.*

3 Information Checking Protocol and IC Signature

The Information Checking Protocol (ICP) is a tool for authenticating messages in the presence of computationally unbounded corrupted parties. The notion of ICP was first introduced by Rabin et.al [18, 17] who have designed an ICP in synchronous settings. The ICP of Rabin et. al. was also used as a tool by Canetti et. al. [5] for designing ABA with optimal resilience (i.e $n = 3t + 1$).

As described in [18, 17, 5, 7], an ICP is executed among three parties: a dealer D, an intermediary INT and a verifier R. The dealer D hands over a secret value s to INT. At a later stage, INT is required to hand over s to R and convince R that s is indeed the value which INT received from D. The basic definition of ICP involves only a *single* verifier R and deals with *only one* secret s [17, 7, 5]. We extend this notion to *multiple* verifiers, where all the n parties in $\mathcal{P}$ act as verifiers. Thus our ICP is executed among three entities: the dealer $D \in \mathcal{P}$, an intermediary $INT \in \mathcal{P}$ and entire set $\mathcal{P}$ acting as verifiers. This will be later helpful in using ICP as a tool in our AWSS/AVSS protocol. Moreover, when appropriate, we run our ICP on *multiple* secrets, denoted by S, which contains $\ell \geq 1$ secret values. Note that, as opposed to the case of a single verifier, when multiple verifiers *simultaneously* participate in ICP, we need to distinguish between synchronity and asynchronity of the network. Our

ICP is executed in asynchronous settings and is denoted by A-ICP(D, INT, P, S). As in [18, 17, 5], our A-ICP is also structured in three phases:

1. **Generation Phase**: is initiated by the dealer D. Here D hands over the secret S to *intermediary* INT. In addition, D hands some *authentication information* to INT and *verification information* to individual parties (verifiers) in $\mathcal{P}$.

2. **Verification Phase**: 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 **Revelation Phase**, secret S held by INT will be (eventually) accepted/will be considered as valid by all the (honest) verifiers in $\mathcal{P}$. INT achieves this by setting a boolean variable $\mathsf{Ver} = 0/1$, where $\mathsf{Ver} = 0$ implies abortion of the protocol, while $\mathsf{Ver} = 1$ implies the continuation of the protocol. If $\mathsf{Ver} = 1$, then the *authentication information*, along with S, which is held by INT at the end of **Verification Phase** is called D's IC *signature* on S.

3. **Revelation Phase**: is carried out by INT and the verifiers in $\mathcal{P}$. Here INT reveals his secret S along with the *authentication information*. The verifiers publish their responses with respect to the revealed information of INT. Depending upon the responses by the verifiers, a verifier $P_i \in \mathcal{P}$ either accepts INT's secret S or rejects it. We denote the acceptance by verifier P_i, (resp., rejection) by $\mathsf{Reveal}_i = S$ (resp., $\mathsf{Reveal}_i = NULL$).

Protocol A-ICP satisfies almost the same properties of the ICP defined in [5], which are as follows:

1. If D and INT are honest, then S will be accepted in Reveal by each honest verifier.

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

3. If D is honest, then during Reveal, with probability at least $1 - 2^{-\mathcal{O}(\kappa)}$, every $S' \neq S$ produced by a corrupted INT will not be accepted by an honest verifier.

4. If D and INT are honest and INT has not started Reveal, then S is information theoretically secure.

Notice that unlike other asynchronous primitives (e.g. AWSS, AVSS, ABA), A-ICP need not have to satisfy any termination property. The reason is that A-ICP will never be executed as a stand alone application in our ABA. Rather, A-ICP will act as a tool to design AWSS and AVSS, which have their own termination properties. This is in line with [5], where ICP is defined without termination property and is used as a tool in AWSS/AVSS protocol. We now present our novel ICP called A-ICP, which allows D to deal with secret S containing $\ell \geq 1$ secret field elements, where $n = 3t + 1$.

The high level idea of the protocol is as follows: D sets the secret as the ℓ lower order coefficients of a polynomial $F(x)$ of degree $\ell + t - 1$ and gives it to INT. D also hands over some *secret evaluation points* and the value of $F(x)$ at those points to individual verifiers. This ensures that at a later stage, a corrupted INT cannot produce an incorrect $F(x)$ during **Revelation Phase**, without being caught by honest verifiers. This ensures property 3 of ICP. [2] Now to satisfy property 2 of ICP, an honest INT has to ensure that his polynomial $F(x)$ will be accepted by honest verifiers during **Revelation Phase**. For this, INT interacts with the verifiers and finds whether his polynomial is 'consistent' with the secret evaluation points and the values possessed by the verifiers. In order to do so, INT and verifiers follow a zero-knowledge strategy. Then in **Revelation Phase**, INT's secret is accepted on the basis of the consistency between the responses of verifiers during **Verification Phase** and the responses of verifiers in **Revelation Phase**. Accepting INT's secret based upon such *consistency checking of responses* is done for the first time in the literature. We now present our ICP protocol (given in Table. 1) which addresses the above described delicate issues. *In the sequel, whenever we say that D gives his IC signature on S to INT, we mean that D starts the* **Generation Phase** *of A-ICP by executing Gen.* For the proof of the properties of our A-ICP, see **APPENDIX B**.

Lemma 1 *In A-ICP, Gen incurs private communication of $\mathcal{O}((\ell + n)\kappa)$ bits. Both Ver and Reveal*

[2] Secret evaluation points were used by Tompa and Woll [20] to protect against faulty actions in synchronous settings.

incur A-cast of $\mathcal{O}((\ell+n)\kappa)$ bits and private communication of $\mathcal{O}(n)$ bits.

Remark: Note that, had we executed ℓ times the protocol A-ICP for single secret, the communication complexity would turn out to be $\mathcal{O}(\ell n\kappa)$ bits (both private and A-cast). However, the communication complexity of A-ICP treating all the ℓ secrets simultaneously is $\mathcal{O}((\ell+n)\kappa)$ bits (both private and A-cast). This clearly shows that executing a *single instance* of A-ICP dealing with *multiple secrets* is advantageous over executing *multiple instances* of A-ICP dealing with *single secret*. The same principle holds for AWSS and AVSS which are described in the sequel.

Protocol A-ICP(D,INT,$\mathcal{P}$,S)

Generation Phase: Gen($D, INT, \mathcal{P}, S$)

1. The dealer D, having secret $S = (s^1, \ldots, s^\ell)$, selects a random $\ell + t - 1$ degree polynomial $F(x)$ over $\mathbb{F}$, such that for $0 \leq i \leq \ell - 1$, the coefficient of x^i in $F(x)$ is s^{i+1}.
2. D selects another random $\ell + t - 1$ degree polynomial $R(x)$ over $\mathbb{F}$.
3. D selects n distinct random secret evaluation points $\alpha_1, \alpha_2, \ldots, \alpha_n$ from $\mathbb{F} \setminus \{0, 1, \ldots, n-1\}$.
4. D sends $F(x)$ and $R(x)$ to INT. To verifier $P_i \in \mathcal{P}$, D sends α_i, v_i and r_i, where $v_i = F(\alpha_i)$ and $r_i = R(\alpha_i)$. We call α_i, v_i and r_i as *verification information* for P_i.

Verification Phase: Ver($D, INT, \mathcal{P}, S$)

1. Verifier P_i sends a `Received-From-D` signal to INT after receiving *verification information* from D.
2. INT waits for $2t + 1$ `Received-From-D` signal. Upon receiving them, INT puts the identity of the verifiers (from whom it has obtained `Received-From-D` signal) in a list, say *ReceivedSet* with $|ReceivedSet| = 2t + 1$. INT then chooses a random $d \in \mathbb{F} \setminus \{0\}$ and A-casts d and $B(x) = dF(x) + R(x)$, along with *ReceivedSet*.
3. A verifier $P_i \in ReceivedSet$, on receiving the A-cast values sent by INT, checks whether $B(\alpha_i) \stackrel{?}{=} dv_i + r_i$. If yes (no) then P_i A-casts `Accept` (`Reject`).
4. INT sets $Ver = 1$, if he receives $t + 1$ `Accept`s from A-casts of $t + 1$ verifiers who belong to *ReceivedSet*. In this case, we say that INT has 'successfully' completed the **Verification Phase**. Moreover, ($F(x)$ and $R(x)$) held by INT is called D's IC *signature* on secret S. On the other hand, INT sets $Ver = 0$, if he receives $t+1$ `Reject`s from A-casts of $t + 1$ verifiers who belong to *ReceivedSet*.

Revelation Phase: Reveal($D, INT, \mathcal{P}, S$)

1. INT A-casts $F(x)$ and $R(x)$.
2. A verifier $P_i \in ReceivedSet$, on receiving $F(x)$ and $R(x)$, A-casted by INT, checks whether $F(\alpha_i) \stackrel{?}{=} v_i$ and $R(\alpha_i) \stackrel{?}{=} r_i$. Now to ensure that INT has indeed set `Ver` $= 1$ during **Verification Phase**, P_i waits to listen $t + 1$ A-casts of `Accept` from $t + 1$ verifiers who belong to *ReceivedSet*. Once he listens the same, P_i A-casts `Re-accept` if both the above test passes. If either one of the test fails, P_i A-casts `Re-reject`.
3. A verifier $P_i \in ReceivedSet$ is called as *consistent* if
   (a) he A-casted `Accept` in **Verification Phase** and `Re-accept` in **Revelation Phase**, Or
   (b) he A-casted `Reject` in **Verification Phase** and `Re-reject` in **Revelation Phase**.
4. Similarly, a verifier $P_i \in ReceivedSet$ is called as *inconsistent* if
   (a) he A-casted `Accept` in **Verification Phase** and `Re-reject` in **Revelation Phase**, Or
   (b) he A-casted `Reject` in **Verification Phase** and `Re-accept` in **Revelation Phase**.
5. If a verifier $P_i \in \mathcal{P}$ finds $(t+1)$ *consistent* verifiers from *ReceivedSet*, then P_i sets $\mathsf{Reveal}_i = S$ and terminates, where S is the set of lower order ℓ coefficients of $F(x)$ which is A-casted by INT in the first step of current phase. We say that INT is 'successful' in producing *IC signature* to P_i.
6. If a verifier $P_i \in \mathcal{P}$ finds $(t + 1)$ *inconsistent* verifiers from *ReceivedSet*, then P_i sets $\mathsf{Reveal}_i = NULL$ and terminates. We say that INT fails to correctly produce IC signature to P_i.

Table 1: **Information Checking Protocol,** $n = 3t + 1$

4 Asynchronous Weak Secret Sharing

We now present a novel AWSS protocol called AWSS-Multiple-Secret with $n = 3t+1$, which allows D to share a secret S containing $\ell \geq 1$ field elements from $\mathbb{F}$. 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 present AWSS-Multiple-Secret which is a simple extension of AWSS-Single-Secret.

We follow the general idea of [2, 7, 13, 12, 14] in synchronous settings for sharing the secret s with a degree-t symmetric bivariate polynomial $F(x, y)$, where each party P_i gets the univariate polynomial $f_i(x) = F(x, i)$. In particular, AWSS-Single-Secret is somewhat inspired by the WSS protocol of [7] in synchronous settings, with several new ideas incorporated in it.

Briefly, the high-level idea of AWSS-Single-Secret is as follows: In the sharing phase of AWSS-Single-Secret, first D hands over n points on $f_i(x)$ (implies passing of $f_i(x)$) to P_i with his IC signature on these values. Then D, in conjunction with all other parties, perform a sequences of communication and computation. As a result of this, at the end of the sharing phase, every party agrees on a set of $2t + 1$ parties, called $WCORE$, such that every party $P_j \in WCORE$ is *confirmed* by a set of $2t + 1$ parties, called as $OKSetP_j$. A party $P_k \in OKSetP_j$ provides the *confirmation* to P_j, only when it possesses proper IC signature of D on $f_k(j)$ (j^{th} point on polynomial $f_k(x)$, which P_k is entitled to receive from D) as well as IC signature of P_j on the point $f_j(k)$ (k^{th} point on polynomial $f_j(x)$, which P_j is entitled to receive from D), such that $f_j(k) = f_k(j)$ holds (which should ideally hold due to the selection and distribution of symmetric bivariate polynomial). In some sense, we may view these checkings as every $P_j \in WCORE$ is attempting to commit his polynomial $f_j(x)$ among the parties in $OKSetP_j$ (by giving his *IC Signature* on one point of $f_j(x)$ to each party) and the parties in $OKSetP_j$ allowing him to do so after verifying that they have got D's IC signature on the same value of $f_j(x)$. We will refer this commitment as P_j's *IC-Commitment* on $f_j(x)$.

Achieving the agreement (among the parties) on $WCORE$ and corresponding $OKSet$s is a bit tricky in asynchronous network. Even though the *confirmations* are A-casted by parties, parties may end up with different versions of $WCORE$ and $OKSet$'s while attempting to generate them locally, due to the asynchronous nature of the network. We solve this problem by asking D to construct $WCORE$ and $OKSet$s after listening *confirmations* and ask D to A-cast the same. After listening $WCORE$ and $OKSet$s from the A-cast of D, individual parties ensure the validity of (verifies) these sets by listening the same *confirmations* from the parties in the received $OKSet$s. Once the verification is done, every honest party agree on these sets. A similar approach was used in the protocols of [1].

In the reconstruction phase, the parties in $WCORE$ and corresponding $OKSet$'s are used for reconstructing the secret. Precisely, in the reconstruction phase, P_j's *IC-Commitment* on $f_j(x)$ is revealed by reconstructing it with the help of the parties in $OKSetP_j$ for every $P_j \in WCORE$. Then the polynomials $f_j(x)$'s are used to construct the symmetric bivariate polynomial (if possible) $F(x, y)$ that is committed by D during sharing phase. Since $f_j(x)$ is a degree-t polynomial, any $t + 1$ points on it are enough to interpolate $f_j(x)$. The points on $f_j(x)$ are obtained by requesting each party P_k in $OKSetP_j$ to reveal IC signature of D on $f_k(j)$ and IC signature of P_j on $f_j(k)$ such that $f_j(k) = f_k(j)$ holds. Asking $P_k \in OKSetP_j$ to reveal D's signature ensures that when D is *honest*, then even for a *corrupted* $P_j \in WCORE$, the reconstructed polynomial $f_j(x)$ will be same as the one handed over by D to P_j in sharing phase. This helps our AWSS protocol to satisfy CORRECTNESS 1 property of AWSS. Now asking P_k in $OKSetP_j$ to reveal P_j's signature ensures that even if D is *corrupted*, for an *honest* $P_j \in WCORE$, the reconstructed polynomial $f_j(x)$ will be same as the one received by P_j from D in the sharing phase. This ensure CORRECTNESS 2 property. Summing up, when at least one of D and P_j is honest, P_j's *IC-Commitment* on $f_j(x)$ is revealed properly. But when both D and P_j are corrupted, P_j's *IC-Commitment* on $f_j(x)$ can be revealed as any $\overline{f_j(x)} \neq f_j(x)$. It is the later property that makes our protocol to qualify as a AWSS protocol rather than a AVSS protocol. Protocol AWSS-Single-Secret is now given in Table 2.

Lemma 2 *Protocol AWSS-Single-Secret satisfies termination property.*

PROOF: **Termination 1:** If D is honest then he will eventually include all honest parties $(2t + 1)$ in $WCORE$ as well as in their respective $OKSet$'s and will A-Cast the same. By property of A-Cast, every honest party will listen the same, confirm their validity and will terminate the sharing phase.

Termination 2: If an honest P_i has completed AWSS-Single-Secret-Share, then he must have listened $WCORE$ and $OKSetP_j$'s from the A-cast of D and verified their validity. By properties of A-cast, each honest party will also listen the same and will eventually terminate AWSS-Single-Secret-Share.

Termination 3: By Lemma 10, if P_i is honest and sets $\mathsf{Ver} = 1$, then IC signature produced by P_i will be accepted in Reveal, except with probability $2^{-\mathcal{O}(\kappa)}$. For every $P_j \in WCORE$, $|OKSetP_j| = 2t+1$. So there are at least $t+1$ honest parties in $OKSetP_j$ who will be present in $ValidSetP_j$ with high probability. So for every $P_j \in WCORE$, P_j's *IC-Commitment* will be reconstructed. Thus with very high probability, each honest party will terminate AWSS-Single-Secret-Rec. □

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

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

DISTRIBUTION: CODE FOR D

1. Select a random degree-t symmetric bivariate polynomial $F(x, y)$ such that $F(0, 0) = s$.
2. For $1 \leq i \leq n$, deliver $f_i(x) = F(x, i)$ to P_i, along with *IC signature* on each of $f_i(j)$ by considering P_i as INT and executing $\mathsf{Gen}(D, P_i, \mathcal{P}, f_i(j))$ for every $j \in \{1, \ldots, n\}$.

VERIFICATION: CODE FOR P_i

1. Wait until $\mathsf{Gen}(D, P_i, \mathcal{P}, f_i(j))$ is completed for every $j \in \{1, \ldots, n\}$.
2. Check whether the points $(f_i(1), \ldots, f_i(n))$ lie on a unique t-degree polynomial. If yes, then acting as INT, execute $\mathsf{Ver}(D, P_i, \mathcal{P}, f_i(j))$ for every $j \in \{1, \ldots, n\}$.
3. For all $j \in \{1, \ldots, n\}$, if $\mathsf{Ver}(D, P_i, \mathcal{P}, f_i(j))$ is completed with $\mathsf{Ver} = 1$, then hand over $f_i(j)$ to P_j, along with *IC signature* by acting as dealer and executing $\mathsf{Gen}(P_i, P_j, \mathcal{P}, f_i(j))$ for all $j \in \{1, \ldots, n\}$. In addition, participate in $\mathsf{Gen}(P_j, P_i, \mathcal{P}, f_j(i))$ for all $j \in \{1, \ldots, n\}$ by acting as INT.
4. Wait until $\mathsf{Gen}(P_j, P_i, \mathcal{P}, f_j(i))$ is completed. If $f_i(j) = f_j(i)$ then execute $\mathsf{Ver}(P_j, P_i, \mathcal{P}, f_j(i))$ as an INT. If $\mathsf{Ver}(P_j, P_i, \mathcal{P}, f_j(i))$ is completed with $\mathsf{Ver} = 1$, then A-cast $\mathtt{OK}(P_i, P_j)$. Here $j \in \{1, \ldots, n\}$.

CORE CONSTRUCTION : CODE FOR D

1. For each P_j, build a set $OKSetP_j = \{P_i | D \textit{ listens } \mathtt{OK}(P_i, P_j)\}$. When $|OKSetP_j| = 2t+1$, then conclude that P_j's *IC-Commitment* on $f_j(x)$ is over and add P_j in $WCORE$ (which is initially empty).
2. Wait until $|WCORE| = 2t + 1$. Then A-cast $WCORE$ and $OKSetP_j$ for all $P_j \in WCORE$.

CORE VERIFICATION & AGREEMENT ON CORE : CODE FOR P_i

1. Wait to obtain $WCORE$ and $OKSetP_j$ for all $P_j \in WCORE$ from D's A-cast, such that $|WCORE| = 2t + 1$ and $|OKSetP_j| = 2t + 1$ for each $P_j \in WCORE$.
2. Wait to receive $\mathtt{OK}(P_k, P_j)$ for all $P_k \in OKSetP_j$ and $P_j \in WCORE$. After receiving, accept the $WCORE$ and $OKSetP_j$'s and terminate **AWSS-Single-Secret-Share**.

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

SIGNATURE REVELATION: CODE FOR P_i

1. If P_i belongs to $OKSetP_j$ for some $P_j \in WCORE$, then participate in $\mathsf{Reveal}(D, P_i, \mathcal{P}, f_i(j))$ and $\mathsf{Reveal}(P_j, P_i, \mathcal{P}, f_j(i))$ as an INT.
2. Participate in $\mathsf{Reveal}(D, P_k, \mathcal{P}, f_k(j))$ and $\mathsf{Reveal}(P_j, P_k, \mathcal{P}, f_j(k))$ for all $P_k \in OKSetP_j$ and $P_j \in WCORE$ as a verifier.

LOCAL COMPUTATION: CODE FOR P_i

1. For every $P_j \in WCORE$, reconstruct P_j's *IC-Commitment* on $f_j(x)$ as follows:
   (a) Construct a set $ValidSetP_j = \emptyset$.
   (b) Add party $P_k \in OKSetP_j$ to $ValidSetP_j$ if both the following conditions are true:
      i. $\mathsf{Reveal}(D, P_k, \mathcal{P}, f_k(j))$ and $\mathsf{Reveal}(P_j, P_k, \mathcal{P}, f_j(k))$ are successfully completed, with outputs, $\mathsf{Reveal}_i = \overline{f_k(j)}$ and $\mathsf{Reveal}_i = \overline{f_j(k)}$ respectively; and
      ii. $\overline{f_k(j)} = \overline{f_j(k)}$.
   (c) Wait until $|ValidSetP_j| = t+1$. Construct a polynomial $\overline{f_j(x)}$ passing through the points $(k, \overline{f_j(k)})$ where $P_k \in ValidSetP_j$. Associate $\overline{f_j(x)}$ with $P_j \in CORE$.
2. Wait until for each P_j in $WCORE$, *IC-Commitment* is reconstructed with a polynomial $\overline{f_j(x)}$.
3. For each $(P_\gamma, P_\delta) \in WCORE$, check $\overline{f_\gamma(\delta)} \stackrel{?}{=} \overline{f_\delta(\gamma)}$. If the test passes for each pair of parties in $WCORE$, then construct the bivariate polynomial $\overline{F(x, y)}$ using the polynomials $\overline{f_j(x)}$ associated with $P_j \in WCORE$, compute $\overline{s} = \overline{F(0, 0)}$ and terminate. Else set $\overline{s} = NULL$ and terminate.

Table 2: **AWSS for Sharing a Single Secret s with $n = 3t + 1$**

Lemma 3 *Protocol AWSS-Single-Secret satisfies secrecy property.*

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

Lemma 4 *Protocol AWSS-Single-Secret satisfies correctness property.*

PROOF: **Correctness 1:** Here we have to consider the case when D is *honest*. We first prove that if D is honest, then with very high probability, for every $P_j \in WCORE$, P_j's *IC-Commitment* will be reconstructed correctly. In other words, $\overline{f_j(x)}$ associated with *every* $P_j \in WCORE$ during reconstruction phase, is same as $f_j(x)$ selected by D. From the property of ICP, for an *honest* $P_j \in WCORE$, a corrupted $P_k \in OKSetP_j$ can produce P_j's valid signature on $\overline{f_j(k)} \neq f_j(k)$ with negligible probability (from Lemma 11). Similarly, for a *corrupted* $P_j \in WCORE$, a corrupted $P_k \in OKSetP_j$ can produce P_j's valid signature on $\overline{f_j(k)} \neq f_j(k)$ but P_k will fail to produce honest D's signature on $\overline{f_k(j)} = \overline{f_j(k)}$ with very high probability. Thus with very high probability, corresponding to each $P_j \in WCORE$, the parties in $ValidSetP_j$ will produce correct points on $f_j(x)$. So everybody will reconstruct $F(x, y)$ and hence the secret $s = F(0, 0)$ with very high probability.

Correctness 2: Here we have to consider the case, when D is *corrupted*. Since in AWSS-Single-Secret-Share, every honest party agrees on a $WCORE$ and $OKSetP_j$ for $P_j \in WCORE$, a unique secret $s' \in \mathbb{F} \cup NULL$ is defined by (at least $t + 1$) honest parties in $WCORE$ at the end of sharing phase. We say that s' is D's *committed secret*. If for every two *honest* parties P_γ and P_δ in $WCORE$, $f_\gamma(\delta) = f_\delta(\gamma)$ holds then s' is the constant term of the degree-t symmetric bivariate polynomial $F'(x, y)$, that is defined by the univariate polynomials of honest parties in $WCORE$. Otherwise $s' = NULL$. We show that every honest party will reconstruct either s' or $NULL$.

We first consider the case when $s' = F'(0, 0)$. This implies that $f_j(x)$'s corresponding to honest P_j's in $WCORE$ define a symmetric t-degree bivariate polynomial $F'(x, y)$. We claim that with very high probability, for an honest $P_j \in WCORE$, P_j's *IC-Commitment* will be reconstructed correctly. In other words, $\overline{f_j(x)}$ associated with an *honest* $P_j \in WCORE$ during reconstruction phase, is same as $f_j(x)$ which was received by P_j. This claim follows from the argument given in CORRECTNESS 1. But for a *corrupted* P_j in $WCORE$, P_j's *IC-Commitment* can be revealed as any t-degree polynomial $\overline{f_j(x)}$. This is because a corrupted $P_k \in OKSetP_j$ can produce a valid signature of P_j on any $\overline{f_j(k)}$ as well as a valid signature of D (who is corrupted as well) on $\overline{f_k(j)} = \overline{f_j(k)}$. Also the adversary can delay the messages such that the values of corrupted $P_k \in OKSetP_j$ are revealed (to parties) before the values of honest parties in $OKSetP_j$. Now if reconstructed $\overline{f_j(x)}$'s corresponding to *corrupted* P_j's in $WCORE$, along with reconstructed $f_j(x)$'s corresponding to *honest* P_j's in $WCORE$ defines $F'(x, y)$, then s' will be reconstructed. Otherwise, $NULL$ will be reconstructed. However, since for all the honest parties of $WCORE$, *IC-Commitment* will be reconstructed correctly with $f_j(x)$ (who in tern define $F'(x, y)$), no other secret (other than s') can be reconstructed with very high probability.

On the other hand, let D's committed secret $s' = NULL$. In this case irrespective of the behavior of corrupted parties, $NULL$ will be reconstructed by each honest party. □

Theorem 1 *The pair (AWSS-Single-Secret-Share, AWSS-Single-Secret-Rec) constitutes a valid AWSS scheme for $n = 3t + 1$ parties, which shares a single secret and satisfies the properties of AWSS.*

IMPORTANT NOTE: In AWSS-Single-Secret, the degree-t univariate polynomial $F(x, 0) = f_0(x)$ is used to share the secret s. *In the following we will say that D shares a degree-t polynomial $f(x)$ using AWSS-Single-Secret* by executing AWSS-Single-Secret-Share$(D, \mathcal{P}, f(x))$. For this D selects a t-degree symmetric bivariate polynomial $F(x, y)$, such that $F(x, 0) = f(x)$ and execute the protocol. It should be noted that $f(x)$ is not completely random but only preserves the secrecy of the constant term which is the secret $s = f(0)$. Yet, this distribution of polynomials is sufficient to provide the secrecy requirements needed by our AVSS where AWSS is used as a building block. If D indeed selects the bivariate polynomial in the above way and follows the protocol steps correctly, then as a result of the above execution, P_i in $WCORE$ will hold the i^{th} share $f(i) = F(i, 0) = F(0, i)$, the polynomial $f_i(x) = F(x, i)$ and the share-share $f_j(i) = F(i, j) = f_i(j)$ corresponding to every other P_j. Similarly, AWSS-Single-Secret-Rec$(D, \mathcal{P}, f(x))$ can be used for the reconstruction of $f(x)$ and secret $s = f(0)$.

We now extend AWSS-Single-Secret to AWSS-Multiple-Secret which shares secret $S = (s^1 \dots s^\ell)$, containing ℓ field elements concurrently. Since AWSS-Multiple-Secret is a simple extension of AWSS-Single-Secret for multiple secrets, we present AWSS-Multiple-Secret and its properties in **APPENDIX C**. For ease of reference, the communication complexity of AWSS-Multiple-Secret is given below.

Lemma 13: *Both AWSS-Multiple-Secret-Share and AWSS-Multiple-Secret-Rec privately communicates* $\mathcal{O}((\ell n^2 + n^3)\kappa)$ *bits and A-cast* $\mathcal{O}((\ell n^2 + n^3)\kappa)$ *bits.*

5 Asynchronous Verifiable Secret Sharing

In this section, we present our novel AVSS scheme called AVSS-Multiple-Secret which shares secret S containing $\ell \geq 1$ elements from $\mathbb{F}$. However, if D is *corrupted*, then each element of S can be either from $\mathbb{F}$ or $NULL$ (in a sense explained in the sequel). As in AWSS, we first present an AVSS protocol, called AVSS-Single-Secret which shares a single secret s. Later we present AVSS-Multiple-Secret which is a simple extension of AVSS-Single-Secret.

We now explain the high level idea used in AVSS-Single-Secret. D uses the bi-variate polynomial approach, as used in the synchronous VSS protocols of [2, 13, 12, 14], where D selects a degree-t symmetric bivariate polynomial $F(x, y)$, such that $F(0, 0) = s$ and sends $f_i(x) = F(x, i)$ to party P_i. Now the parties communicate with each other to perform what we say *commitment upon verification*. Here each party P_i is asked to *commit* his received polynomial $f_i(x)$. However, P_i is allowed to commit $f_i(x)$, only after the parties have verified that they have received same points on $f_i(x)$ from D as well as P_i. More formally, to achieve *commitment upon verification*, party P_i, acting as a dealer, shares his polynomial $f_i(x)$ by initiating an instance of our AWSS protocol AWSS-Single-Secret-Share with a degree-t symmetric bivariate polynomial $Q^{P_i}(x, y)$, such that $Q^{P_i}(x, 0) = f_i(x)$. Since party P_j receives $q_j^{P_i}(x) = Q^{P_i}(x, j)$ from P_i as part of the AWSS, he can check whether $q_j^{P_i}(0) \stackrel{?}{=} f_j(i)$, as ideally $q_j^{P_i}(0) = f_i(j) = f_j(i)$ should hold in case of honest D, P_i and P_j. A party P_j participates in the remaining steps of the AWSS instance of P_i, only if $q_j^{P_i}(0) = f_j(i)$ holds. Once *commitment upon verification* is over, the parties want to agree on a set of at least $2t + 1$ parties, denoted as $VCORE$ such that for every party $P_j \in VCORE$, the instance of AWSS initiated by P_j, terminates with a $WCORE$ set, denoted as $WCORE^{P_j}$ and $|VCORE \cap WCORE^{P_j}| \geq 2t + 1$. Informally, this means that each party $P_j \in VCORE$ has 'successfully' committed his polynomial $f_j(x)$ to at least $2t + 1$ parties in $VCORE$, who have verified that they have received correct points on $f_j(x)$. We will refer this commitment as P_j's *AWSS-Commitment* on $f_j(x)$. It should be noted that *AWSS-Commitment* is strictly stronger commitment than *IC-Commitment* that was enforced in our AWSS protocols. These two commitments can be distinguished only when both D and P_j are corrupted. In this case, while *AWSS-Commitment* ensures that reconstruction of *AWSS-Commitment* can not be changed to some other polynomial other than $NULL$, *IC-Commitment* can not ensure the same.

As in our AWSS protocol, the agreement on $VCORE$ and corresponding $WCORE^{P_j}$'s is achieved by letting D to first construct $VCORE$ after obtaining $WCORE^{P_j}$'s from the A-cast of P_j's and then A-cast $VCORE$ and corresponding $WCORE^{P_j}$'s. Later, on receiving $VCORE$ and $WCORE^{P_j}$'s from the A-cast of D, a party checks the validity of $VCORE$ and $WCORE^{P_j}$ by listening the required A-casts and accepts them after verification.

In the reconstruction phase, D's committed secret is recovered with the help of the parties in $VCORE$ and $WCORE^{P_j}$'s. Precisely, in the reconstruction phase, for every $P_j \in VCORE$, *AWSS-Commitment* on $f_j(x)$ is revealed by reconstructing it with the help of the parties in $WCORE^{P_j}$. This is done by executing an instance of AWSS-Single-Secret-Rec with the parties in $WCORE^{P_j}$. This results in the reconstruction of either $f_j(x)$ or NULL depending on whether P_j is honest or corrupted. Since $|VCORE| \geq 2t + 1$, for (at least $t + 1$) honest parties, $f_j(x)$'s will be recovered correctly. Now with the $f_j(x)$'s, the bivariate polynomial $F(x, y)$ will be reconstructed.

A Note on NULL Commitment in Our AVSS Scheme: In our AVSS scheme, we say that D's committed secret is the constant term of the symmetric degree-t bivariate polynomial $F'(x, y)$ which is defined by the univariate polynomials of the honest parties in $VCORE$. For an *honest* D, the polynomials of the honest parties in $VCORE$ will always define a symmetric bivariate polynomial

and thus the committed secret is always a *valid field element* and hence considered as *meaningful*. But for a *corrupted* D, this may not hold. In this case, we say that D has committed $NULL$. The reconstruction phase of our AVSS protocol ensures the reconstruction of *only committed secret*.

Protocol AVSS-Single-Secret$(D, \mathcal{P}, s)$

AVSS-Single-Secret-Share$(D, \mathcal{P}, s)$

DISTRIBUTION: CODE FOR D

1. Select a degree-t random symmetric bivariate polynomial $F(x, y)$ such that $F(0, 0) = s$.
2. Deliver $f_i(x) = F(x, i)$ to P_i.

COMMITMENT UPON VERIFICATION: CODE FOR P_i

1. Wait to obtain $f_i(x)$ from D.
2. If $f_i(x)$ is a t-degree polynomial then as a dealer, execute AWSS-Single-Secret-Share$(P_i, \mathcal{P}, f_i(x))$ by selecting a degree-t symmetric bivariate polynomial $Q^{P_i}(x, y)$ such that $Q^{P_i}(x, 0) = q_0^{P_i}(x) = f_i(x)$. We call this instance of AWSS-Single-Secret-Share initiated by P_i as AWSS-Single-Secret-Share^{P_i}.
3. As a part of the execution of AWSS-Single-Secret-Share^{P_j}, wait to receive $q_i^{P_j}(x) = Q^{P_j}(x, i)$ from P_j. Then check $f_i(j) \stackrel{?}{=} q_i^{P_j}(0)$. If the test passes then participate in AWSS-Single-Secret-Share^{P_j} and act according to the remaining steps of AWSS-Single-Secret-Share^{P_j}.

CORE CONSTRUCTION : CODE FOR D

1. Wait to terminate AWSS-Single-Secret-Share^{P_j} with $WCORE^{P_j}$ and $OKSetP_k^{P_j}$ for every $P_k \in WCORE^{P_j}$. Then add P_j in a set $TempCORE$ (initially empty). Here $j \in \{1, \ldots, n\}$.
2. Even after terminating AWSS-Single-Secret-Share^{P_j}, update (include new parties in) $WCORE^{P_j}$ and $OKSetP_k^{P_j}$'s upon receiving new A-casts of the form OK$(.,.)$ as a part of AWSS-Single-Secret-Share^{P_j}. Also update $TempCORE$ upon terminating AWSS-Single-Secret-Share for new parties.
3. After every update, perform the following computations:
   (a) Assign $VCORE = TempCORE$ and check whether $|VCORE \cap WCORE^{P_j}| \geq 2t + 1$ for every $P_j \in VCORE$. If not then remove P_j from $VCORE$ and keep on repeating this until no more party can be removed from $VCORE$.
   (b) Check whether $|VCORE| \geq 2t + 1$. If not, then delete $VCORE$ and wait for more updates. Otherwise, A-cast $VCORE$, $WCORE^{P_j}$ for $P_j \in VCORE$ and $OKSetP_k^{P_j}$ for every $P_k \in WCORE^{P_j}$. In this case, each $P_j \in VCORE$ has *AWSS-committed* to $f_j(x)$.

CORE VERIFICATION & AGREEMENT ON CORE : CODE FOR P_i

1. Wait to listen $VCORE$, $WCORE^{P_j}$ for $P_j \in VCORE$ and $OKSetP_k^{P_j}$ for every $P_k \in WCORE^{P_j}$ from D's A-cast, such that $|VCORE| \geq 2t + 1$ and for each $P_j \in VCORE$, $|VCORE \cap WCORE^{P_j}| \geq 2t + 1$ and $|OKSetP_k^{P_j}| \geq 2t + 1$ for every $P_k \in WCORE^{P_j}$.
2. Wait to terminate AWSS-Single-Secret-Share^{P_j} corresponding to every P_j in $VCORE$.
3. For every $P_j \in VCORE$, wait to listen OK(P_m, P_k) for every $P_m \in OKSetP_k^{P_j}$ and $P_k \in WCORE^{P_j}$. After listening all OKs, accept the $VCORE$, $WCORE^{P_j}$ for $P_j \in VCORE$ and $OKSetP_k^{P_j}$ for every $P_k \in WCORE^{P_j}$ and terminate **AVSS-Single-Secret-Share**.

AVSS-Single-Secret-Rec$(D, \mathcal{P}, s)$: Public reconstruction of s towards each party:

SECRET RECONSTRUCTION: CODE FOR P_i

1. For every $P_j \in VCORE$, reconstruct P_j's *AWSS-Commitment* on $f_j(x)$ as follows:
   (a) Participate in AWSS-Single-Secret-Rec$(P_j, \mathcal{P}, f_j(x))$ with $WCORE^{P_j}$ and $OKSetP_k^{P_j}$ for every $P_k \in WCORE^{P_j}$. We call this instance of AWSS-Single-Secret-Rec as AWSS-Single-Secret-Rec^{P_j}.
   (b) Wait for the termination of AWSS-Single-Secret-Rec^{P_j} with output either $\overline{Q^{P_j}(x, y)}$ or $NULL$.
2. Wait for the reconstruction of P_j's *AWSS-Commitment* for every $P_j \in VCORE$.
3. Add $P_j \in VCORE$ to $FINAL$ if AWSS-Single-Secret-Rec^{P_j} gives a non-$NULL$ output. Now for $P_j \in FINAL$, assign $\overline{f_j(x)} = \overline{Q^{P_j}(x, 0)}$.
4. For every pair $(P_\gamma, P_\delta) \in FINAL$ check $\overline{f_\gamma(\delta)} \stackrel{?}{=} \overline{f_\delta(\gamma)}$. If the test passes for every pair of parties then recover $\overline{F(x, y)}$ using $\overline{f_j(x)}$'s corresponding to each $P_j \in FINAL$ and reconstruct $\overline{s} = \overline{F(0, 0)}$. Else reconstruct $\overline{s} = NULL$. Finally output $\overline{s}$ and terminate.

Lemma 5 *Protocol AVSS-Single-Secret satisfies termination property.*

PROOF: **Termination 1:** If D is honest, then corresponding to every honest P_j, AWSS-Single-Secret-Share^{P_j} will eventually terminate with $2t+1$ honest parties in $WCORE^{P_j}$. Thus eventually all the $2t+1$ honest parties will be included in $TempCORE$ and D will eventually get $VCORE = TempCORE$, such that $|VCORE| \geq 2t+1$ and $|VCORE \cap WCORE^{P_j}| \geq 2t+1$ for $P_j \in VCORE$. From similar argument given in **Termination 1** of Lemma 2, all honest parties will eventually agree on $VCORE$, $WCORE^{P_j}$ for $P_j \in VCORE$ and $OKSetP_k^{P_j}$ and will terminate AVSS-Single-Secret-Share.

Termination 2: The proof follows from the similar argument given in **Termination 2** of Lemma 2.

Termination 3: Follows from the fact that corresponding to each $P_j \in VCORE$, an honest P_i will eventually terminate AWSS-Single-Secret-Rec^{P_j} (from **Termination 3** of Lemma 2). □

Lemma 6 *Protocol AVSS-Single-Secret satisfies correctness property.*

PROOF: **Correctness 1:** We have to consider the case when D is honest. If D is *honest* then we prove that with very high probability, for every $P_i \in FINAL$, P_i's *AWSS-Commitment* will be reconstructed correctly. In other words, AWSS-Single-Secret-Rec^{P_i} will disclose $\overline{f_i(x)}$ which is same as $f_i(x)$ selected by honest D. For every *honest* $P_i \in FINAL$ this is trivially true. We have to prove the above statement for a corrupted $P_i \in FINAL$. If a corrupted P_i belongs to $FINAL$, it implies AWSS-Single-Secret-Rec^{P_i} is successful (i.e., the output is a symmetric degree-t bivariate polynomial) and AWSS-Single-Secret-Share^{P_i} had terminated during AVSS-Single-Secret-Share. Moreover termination of AVSS-Single-Secret-Share ensures $|VCORE \cap WCORE^{P_i}| \geq 2t+1$. The above statements have the following implications together: (a) With very high probability, P_i must have given consistent polynomials to the honest parties in $WCORE^{P_i}$ (during AWSS-Single-Secret-Share^{P_i}) such that they induce valid degree-t symmetric bivariate polynomial (see **Correctness 2** of Lemma 4). (b) P_i must have agreed with the honest parties of $WCORE^{P_i}$ with respect to the common values given by D. This means that as a part of AWSS-Single-Secret-Share^{P_i}, P_i handed over $q_j^{P_i}(x)$ to an honest P_j (in $WCORE^{P_i}$) satisfying $f_j(i) = q_j^{P_i}(0)$. The statements in (a) and (b) together implies that P_i must have committed (to the honest parties in $WCORE^{P_i}$ which are at least $t+1$) some bivariate polynomials $Q^{P_i}(x,y)$ satisfying $Q^{P_i}(x,0) = f_i(x)$. Thus if AWSS-Single-Secret-Rec^{P_i} is successful, then $\overline{Q^{P_i}(x,y)} = Q^{P_i}(x,y)$ and hence $\overline{f_i(x)} = f_i(x)$. Since D is honest, $\overline{f_i(x)}$'s corresponding to $P_i \in FINAL$ will define $\overline{F(x,y)} = F(x,y)$. Thus $s = \overline{F(0,0)} = F(0,0)$ will be recovered.

Correctness 2: Since in AVSS-Single-Secret-Share, every honest party agrees on a common $VCORE$, a unique secret $s' \in (\mathbb{F} \cup NULL)$ is defined by (at least $t+1$) honest parties in $VCORE$. If for every two *honest* parties P_γ and P_δ in $VCORE$, $f_\gamma(\delta) = f_\delta(\gamma)$ holds then s' is the constant term of the bivariate polynomial $F(x,y)$, which is defined by the univariate polynomials $f_\gamma(x)$'s, held by the honest parties in $VCORE$. Otherwise $s' = NULL$. In this case, we say that committed s' is not *meaningful*. We show that irrespective of whether s' is meaningful or $NULL$, s' will be recovered in AVSS-Single-Secret-Rec with very high probability.

Let $s' = NULL$. Now for every honest $P_i \in VCORE$, AWSS-Single-Secret-Rec^{P_i} will disclose $\overline{f_i(x)}$ which is same as $f_i(x)$ received by P_i in sharing phase. Hence, all honest parties from $VCORE$ will be added to $FINAL$ with very high probability. Now irrespective of the remaining (corrupted) parties included in $FINAL$, the consistency checking (i.e., $\overline{f_\gamma(\delta)} \stackrel{?}{=} \overline{f_\delta(\gamma)}$) will fail for some honest parties (P_γ, P_δ) and $NULL$ will be reconstructed.

On the other hand, let s' be *meaningful* and defined by the bivariate polynomial $F'(x,y)$. Hence, $F'(x,y)$ is defined by the $f_i(x)$'s of the honest parties in $VCORE$. Now this case completely resembles with the case when D is honest and hence the proof follows from the proof of **Correctness 1**. □

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

PROOF: Follows from Lemma 3, Lemma 12 and properties of symmetric bivariate polynomial. □

IMPORTANT NOTE: Protocol AVSS-Single-Secret does not force *corrupted* D to commit some *meaningful* secret (i.e., an element from $\mathbb{F}$). Hence, the secret s, committed by a *corrupted* D can be either from $\mathbb{F}$ or $NULL$. We may assume that if D's committed secret is $NULL$, then D has committed some predefined value $s^* \in \mathbb{F}$, which is known publicly. Hence in AVSS-Single-Secret-Rec, whenever

$NULL$ is reconstructed, every honest party replaces $NULL$ by the predefined secret s^*. Interpreting this way, we say that our AVSS scheme allows D to *commit* secret $s \in \mathbb{F}$.

We now present AVSS-Multiple-Secret which is a simple extension of AVSS-Single-Secret. For its similarity with AVSS-Single-Secret, we defer the description of AVSS-Multiple-Secret in **APPENDIX D**. For ease of reference, the communication complexity of AVSS-Multiple-Secret is presented below:

Lemma 14: *Both Protocol AVSS-Multiple-Secret-Share and Protocol AVSS-Multiple-Secret-Rec communicate* $\mathcal{O}((\ell n^3 + n^4)\kappa)$ *bits and A-cast* $\mathcal{O}((\ell n^3 + n^4)\kappa)$ *bits.*

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

Once we have an AVSS protocol in our hand, we can get an ABA protocol with optimal resilience, following the ideas outlined in [4]. The first step is to get a *common coin* [4]. In [4], it is shown that given an AVSS protocol we can construct a Common-Coin protocol that generates a common coin. We replace the AVSS scheme of [5, 4] with our AVSS protocol AVSS-Multiple-Secret to get an efficient common coin protocol, which we call as Efficient-Common-Coin. Efficient-Common-Coin is exactly the common coin protocol described in Figure 5-9 in [4], with the following difference, which has no effect on the final outcome of common coin protocol. In the common coin protocol of [4], every party shares n random secrets using n different instances of AVSS protocol of [5, 4]. But in our Efficient-Common-Coin, a party shares n random secrets using a single instance of our AVSS protocol AVSS-Multiple-Secret. The immediate result is enormous gain in terms of communication complexity.

Lemma 8 *Protocol Efficient-Common-Coin communicates* $\mathcal{O}(n^5\kappa)$ *bits and A-cast* $\mathcal{O}(n^5\kappa)$ *bits.*

PROOF: Follows from the fact that Efficient-Common-Coin executes at most n instances of AVSS-Multiple-Secret-Share and Protocol AVSS-Multiple-Secret-Rec with $\ell = n$ secrets. □

The proof that Efficient-Common-Coin will satisfy the same properties as common coin protocol of [4] follows from Lemmas 5.27-5.31 of [4]. The next step is to use the common coin protocol to get ABA protocol. In [4], Canetti has used common coin protocol of [5, 4] that terminates with probability $(1 - 2^{-O(\kappa)})$ to get a $(1 - 2^{-O(\kappa)})$-terminating, t-resilient ABA protocol (see Figure 5-11 of [4]). We replace the common coin protocol of [4] by our Efficient-Common-Coin protocol to obtain our efficient $(1 - 2^{-O(\kappa)})$-terminating, t-resilient ABA with $3t + 1$ parties which we call as Efficient-ABA. Similar to the ABA protocol of [5], conditioned on the event that Efficient-ABA terminates, it does so in constant expected time. The proof for this follows from same arguments given in [4]. We avoid giving the details of Efficient-ABA as it will be a repetition of the steps given in the ABA protocol of [5]. So:

Theorem 2 *Protocol Efficient-ABA communicates* $\mathcal{O}(\mathcal{C}n^5\kappa)$ *bits and A-casts* $\mathcal{O}(\mathcal{C}n^5\kappa)$ *bits, where* $\mathcal{C} = \Theta(1)$ *is the expected running time of the protocol.*

PROOF: Since Efficient-ABA terminates in $\mathcal{C} = \Theta(1)$ expected time, protocol Efficient-Common-Coin will be called expected constant number of times in Efficient-ABA. This follows from the similar argument as given in [5, 4]). Hence the theorem. □

7 Conclusion and Open Problems

We have presented a novel expected constant time $(1 - 2^{-O(\kappa)})$-terminating, optimally resilient ABA protocol whose communication complexity is significantly better than existing ABA protocols of [5, 1] (though the ABA protocol of [1] has a strong property of being *almost surely terminating*). Here we summarize the key factors that have contributed to the gain in the communication complexity of our ABA protocol: (a) a shorter route: $ICP \rightarrow AWSS \rightarrow AVSS \rightarrow ABA$, (b) Improving each of the building blocks by introducing new techniques and by exploiting the advantages of dealing with multiple secrets concurrently in each of these blocks. It is to be mentioned that our new AVSS scheme significantly outperforms the existing AVSS schemes in the same settings in terms of communication complexity. An interesting open problem is to further improve the communication complexity of ABA protocols. Also one can try to provide an *almost surely terminating*, optimally resilient, expected constant time ABA protocol whose communication complexity is less than the ABA protocol of [1].

Acknowledgements: The authors would like to thank the anonymous referees of TCC 2009 for pointing out some subtle flaws in the earlier version 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. ACM Press, 2008.

[2] 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.

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

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

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

[6] B. Chor and C. Dwork. Randomization in byzantine agreement. *Advances in Computing Research*, 5::443–497, 1989.

[7] 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.

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

[9] P Feldman and S. Micali. An optimal probabilistic protocol for synchronous byzantine agreement. *SIAM Journal of Computing*, 26(4):873–933, 1997.

[10] M. Fischer. The consensus problem in unreliable distributed system. Technical Report, Department of ComputerScience, Yale University, 1983.

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

[12] 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.

[13] 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.

[14] 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.

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

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

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

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

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

[20] M. Tompa and H. Woll. How to share a secret with cheaters. In *CRYPTO*, pages 261–265, 1986.

APPENDIX A: Communication Complexity Analysis of AVSS of [5]

The communication complexity analysis of the AVSS protocol of [5] was not reported anywhere so far. So we have carried out the same at this juncture. To do so, we have considered the detailed description of the AVSS protocol of [5] given in Canetti's Thesis [4]. To begin with, in the **ICP** protocol of [5], D gives $\mathcal{O}(\kappa)$ field elements to INT and $\mathcal{O}(\kappa)$ field elements to verifier R. Though the **ICP** protocol of [4] is presented with a *single* verifier, it is executed with n verifiers in protocol **A-RS**. In order to execute **ICP** with n verifiers, D gives $\mathcal{O}(n\kappa)$ field elements to INT and $\mathcal{O}(\kappa)$ field elements to each of the n verifiers. So the communication complexity of **ICP** of [4] when executed with n verifiers is $\mathcal{O}(n\kappa)$ field elements and hence $\mathcal{O}(n\kappa^2)$ bits.

Now by incorporating their **ICP** protocol with n verifiers in Shamir secret sharing [19], the authors in [5] 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 **ICP** protocol with n verifiers. So the **A-RS-Share** protocol of [5] involves a private communication of $\mathcal{O}(n^2\kappa^2)$ bits. In addition to this, the **A-RS-Share** protocol also 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 private communication 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 [5] 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 [19]) of the secret and instantiate n instances of the **ICP** protocol for each of the n shares. Now each individual party **A-RS-Share** all the values that it has received in the n instances of the **ICP** protocol. Since each individual party receives a total of $\mathcal{O}(n\kappa)$ field elements in the n instances of **ICP**, the above step incurs a private communication 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 party P_i tries to reconstruct the values which are **A-RS-Shared** by each party P_j in a set $\mathcal{E}_i$. Here $\mathcal{E}_i$ is a set which is defined in 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 privately communicates $\mathcal{O}(n^5\kappa^3)$ bits and A-Cast $\mathcal{O}(n^5\kappa\log(n))$ bits.

The authors in [5] then further extended their **AWSS-Share** protocol to **Two&Sum AWSS-Share** protocol, where each party P_i has to **A-RS-Share** $\mathcal{O}(n\kappa^2)$ field elements. So the communication complexity of **Two&Sum AWSS-Share** 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 [5] 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 party has to **AWSS-Rec** $\mathcal{O}(n^3\kappa)$ field elements. So in total, the **AVSS-Share** protocol of [5] 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.

APPENDIX B: Proof of the Properties of A-ICP

Lemma 9 *If D and INT are honest, then S will be accepted in Reveal by every honest verifier.*

PROOF: If D and INT are honest, then INT will set $\mathsf{Ver} = 1$ at the end of **Verification Phase**. This is because at least $t+1$ honest verifiers from *ReceivedSet* will A-cast `Accept`. In Reveal, at least $t+1$

honest verifiers from *ReceivedSet* will A-cast `Re-accept` and hence will be considered as *consistent*. Hence every (honest) verifier P_i will eventually output $\mathsf{Reveal}_i = S$ at the end of Reveal. □

Lemma 10 *If INT is honest and $\mathsf{Ver} = 1$ at the end of* **Verification Phase**, *then S held by INT will be accepted in Reveal by every honest verifier, except with probability $2^{-\mathcal{O}(\kappa)}$.*

PROOF: Since INT is honest and $\mathsf{Ver} = 1$ at the end of **Verification Phase**, INT must have listened $t+1$ A-casts of Accept from $t+1$ verifiers of *ReceivedSet*. Moreover, there are $2t+1$ verifiers in *ReceivedSet*, of which at least $t+1$ are honest. If we can show that the *honest verifiers* in *ReceivedSet* will be eventually considered as *consistent* during Reveal with very high probability, then the proof will go through, as every honest verifier will then accept INT's secret S. We now prove the same.

Let P_i be an honest verifier in *ReceivedSet* who A-casts `Accept` during **Verification Phase**. This implies that $B(\alpha_i) = dv_i + r_i$ holds, which further implies that either (a) $F(\alpha_i) = v_i$ and $R(\alpha_i) = r_i$ OR (b) $F(\alpha_i) \neq v_i$ and $R(\alpha_i) \neq r_i$ (for other combinations, $B(\alpha_i) = dv_i + r_i$ will not hold). If (a) holds, then P_i will A-cast `Re-accept` and will be considered as *consistent* during Reveal. But on the other hand if (b) holds, then P_i will A-cast `Re-reject` and will be considered as *inconsistent* during Reveal. We now show that P_i will be considered as *inconsistent* during Reveal with negligible probability. The reason is that for every $F(x), R(x)$ held by an honest INT and v_i, r_i held by an honest verifier P_i in *ReceivedSet*, there is *only one* non-zero d for which $B(\alpha_i) = dv_i + r_i$, even if (b) is true. For otherwise, assume there exists another non-zero element $e \neq d$, for which $B(\alpha_i) = ev_i + r_i$ holds, even if (b) holds. This implies that $(d-e)F(\alpha_i) = (d-e)v_i$ or $F(\alpha_i) = v_i$, which is a contradiction to (b). Now since d is randomly chosen by INT only after D hands over $F(x), R(x)$ to INT and v_i, r_i to P_i, a corrupted D has to guess d in advance during Gen, in order to make P_i *inconsistent* during Reveal. However, D can guess d with probability at most $\frac{1}{|\mathbb{F}|-1} \approx 2^{-\mathcal{O}(\kappa)}$. Thus if P_i has A-cast `Accept` during **Verification Phase**, then it will be considered as *consistent* during Reveal with very high probability.

On the other hand, let P_i be an honest verifier in *ReceivedSet* who has A-cast `Reject` during **Verification Phase**. This implies that $B(\alpha_i) \neq dv_i + r_i$ which further implies either $F(\alpha_i) \neq v_i$ or $R(\alpha_i) \neq r_i$ or both. What ever may be the case, P_i will A-cast `Re-reject` during Reveal and will be always considered as *consistent*. Hence the lemma. □

Lemma 11 *If D is honest, then during Reveal, with probability at least $1 - 2^{-\mathcal{O}(\kappa)}$, every $S' \neq S$ produced by a corrupted INT will not be accepted by an honest verifier.*

PROOF: A corrupted INT may produce incorrect $B'(x) \neq B(x)$ during **Verification Phase**. But since D is honest, a corrupted INT will have no knowledge about α_i's corresponding to honest verifiers in *ReceivedSet*. So all the honest verifiers in *ReceivedSet* (at least $t+1$) will A-cast `Reject` with very high probability during **Verification Phase**. In order that an honest verifier P_i in *ReceivedSet* A-casts `Accept` for $B'(x)$, a corrupted INT has to ensure that $B'(\alpha_i) = B(\alpha_i)$. For this, he has to correctly guess α_i, which he can do with probability with at most $\frac{\ell+t-1}{|\mathbb{F}-n-1|} \approx 2^{-\mathcal{O}(\kappa)}$. So, in this case, none of the honest verifiers in *ReceivedSet* will ever respond during Reveal, irrespective of the values produced by INT during Reveal, as they will be waiting indefinitely for the A-cast of $t+1$ `Accept` signals from the verifiers in *ReceivedSet*.

So assume that a corrupted INT has produced correct $B(x)$ during **Verification Phase**. All honest verifiers from *ReceivedSet* will then A-cast `Accept` eventually. But if INT A-casts $F'(x) \neq F(x)$ to produce a different $S' \neq S$ during Reveal, then all honest verifiers in *ReceivedSet* will eventually A-cast `Re-reject` with very high probability. To make an honest verifier P_i A-cast `Re-accept` during Reveal, INT must ensure $F'(\alpha_i) = F(\alpha_i)$. For this, he has to correctly guess α_i which he can do with probability at most $\frac{\ell+t-1}{|\mathbb{F}-n-1|} \approx 2^{-\mathcal{O}(\kappa)}$. Thus with very high probability, all honest verifiers from *ReceivedSet* will eventually A-cast `Re-reject` and hence will eventually become *inconsistent*. Thus every honest verifier P_i will eventually set $\mathsf{Reveal}_i = NULL$. □

Lemma 12 *If D and INT are honest and INT has not started Reveal, then S is information theoretically secure from $\mathcal{A}_t$ controlling at most t verifiers in $\mathcal{P}$.*

PROOF: At the end of **Verification Phase**, $\mathcal{A}_t$ will know t distinct points on $F(x)$ and $R(x)$ from t corrupted verifiers. $\mathcal{A}_t$ will also learn d and $B(x) = dF(x) + R(x)$. Since $F(x)$ and $R(x)$ are

polynomials of degree $\ell + t - 1$, the secrets (which are the lower order ℓ coefficient of $F(x)$) will still remain information theoretically secure. □

APPENDIX C: Protocol AWSS-Multiple-Secret and Proof of Its Properties

Protocol AWSS-Multiple-Secret is given in Table. 3.

Protocol AWSS-Multiple-Secret$(D, \mathcal{P}, S)$

AWSS-Multiple-Secret-Share$(D, \mathcal{P}, S)$

DISTRIBUTION: CODE FOR D

1. Select ℓ degree-t random symmetric bivariate polynomials $F^1(x,y), \ldots, F^\ell(x,y)$ such that for $l = 1, \ldots, \ell$, $F^l(0,0) = s^l$.
2. For $l = 1, \ldots, \ell$, deliver $f_i^l(x) = F^l(x,i)$ to P_i, along with *IC signature* by considering P_i as INT and executing $\mathsf{Gen}(D, P_i, \mathcal{P}, \Gamma_{ij})$ for every $j \in \{1, \ldots, n\}$ where $\Gamma_{ij} = (f_i^1(j), \ldots, f_i^\ell(j))$.

VERIFICATION: CODE FOR P_i

1. Wait until $\mathsf{Gen}(D, P_i, \mathcal{P}, \Gamma_{ij})$ is completed for every $j \in \{1, \ldots, n\}$
2. For each $l = 1, \ldots, \ell$, check whether the points $(f_i^l(1), \ldots, f_i^l(n)$ lie on a unique t-degree polynomial. If yes, then acting as INT, execute $\mathsf{Ver}(D, P_i, \mathcal{P}, \Gamma_{ij})$ for every $j \in \{1, \ldots, n\}$.
3. If for all $j \in \{1, \ldots, n\}$, $\mathsf{Ver}(D, P_i, \mathcal{P}, \Gamma_{ij})$ gets over with $\mathsf{Ver} = 1$, then hand over Γ_{ij} to P_j, along with *IC signature* by acting as dealer and considering P_j as INT and executing $\mathsf{Gen}(P_i, P_j, \mathcal{P}, \Gamma_{ij})$ for all $j \in \{1, \ldots, n\}$. In addition, participate in $\mathsf{Gen}(P_j, P_i, \mathcal{P}, \Gamma_{ji})$ for all $j \in \{1, \ldots, n\}$ by acting as INT and considering P_j as dealer.
4. Wait until $\mathsf{Gen}(P_j, P_i, \mathcal{P}, \Gamma_{ji})$ is completed. If $\Gamma_{ij} = \Gamma_{ji}$ (i.e. for $l = 1, \ldots, \ell$, $f_i^l(j) = f_j^l(i)$), then execute $\mathsf{Ver}(P_j, P_i, \mathcal{P}, \Gamma_{ji})$. If $\mathsf{Ver}(P_j, P_i, \mathcal{P}, \Gamma_{ji})$ gets over with $\mathsf{Ver} = 1$, then A-cast $\mathtt{OK}(P_i, P_j)$. Here $j \in \{1, \ldots, n\}$.

CORE CONSTRUCTION : CODE FOR D– Same as in Protocol AWSS-Single-Secret-Share

CORE VERIFICATION & AGREEMENT ON CORE : CODE FOR P_i– same as in Protocol AWSS-Single-Secret-Share

AWSS-Multiple-Secret-Rec$(D, \mathcal{P}, S)$**:** Public reconstruction of S towards each party:

SIGNATURE REVELATION: CODE FOR P_i

1. If P_i belongs to $OKSetP_j$ for some $P_j \in WCORE$, then participate in $\mathsf{Reveal}(D, P_i, \mathcal{P}, \Gamma_{ij})$ and $\mathsf{Reveal}(P_j, P_i, \mathcal{P}, \Gamma_{ji})$ as an INT.
2. Participate in $\mathsf{Reveal}(D, P_k, \mathcal{P}, \Gamma_{kj})$ and $\mathsf{Reveal}(P_j, P_k, \mathcal{P}, \Gamma_{jk})$ for all $P_k \in OKSetP_j$ and $P_j \in WCORE$ as a verifier.

LOCAL COMPUTATION**:** CODE FOR P_i

1. For every $P_j \in WCORE$, reconstruct P_j's *IC-Commitment* on $f_j^1(x), \ldots, f_j^\ell(x)$ as follows:
   (a) Construct a set $ValidSetP_j = \emptyset$.
   (b) Add party $P_k \in OKSetP_j$ to $ValidSetP_j$ if both the following conditions are true:
      i. $\mathsf{Reveal}(D, P_k, \mathcal{P}, \Gamma_{kj})$ and $\mathsf{Reveal}(P_j, P_k, \mathcal{P}, \Gamma_{jk})$ are successfully completed, with outputs, $\mathsf{Reveal}_i = \overline{\Gamma_{kj}} = (\overline{f_k^1(j)}, \ldots, \overline{f_k^\ell(j)})$ and $\mathsf{Reveal}_i = \overline{\Gamma_{jk}} = (\overline{f_j^1(k)}, \ldots, \overline{f_j^\ell(k)})$ respectively;
      ii. $\overline{\Gamma_{kj}} = \overline{\Gamma_{jk}}$ i.e. for $l = 1, \ldots, \ell$, $\overline{f_k^l(j)} = \overline{f_j^l(k)}$.
   (c) Wait until $|ValidSetP_j| = t+1$. Construct a polynomial $\overline{f_j^l(x)}$ passing through the points $(k, \overline{f_j^l(k)})$ where $P_k \in ValidSetP_j$. For $l = 1, \ldots, \ell$, associate $\overline{f_j^l(x)}$ with $P_j \in WCORE$.
2. Wait until for each P_j in $WCORE$, *IC-Commitment* is reconstructed with ℓ polynomials $\overline{f_j^1(x)}, \ldots, \overline{f_j^\ell(x)}$.
3. For every $l \in \{1, \ldots, \ell\}$ do the following: for every pair $(P_\gamma, P_\delta) \in WCORE$ check $\overline{f_\gamma^l(\delta)} \stackrel{?}{=} \overline{f_\delta^l(\gamma)}$. If the test passes for every pair of parties then recover $\overline{F^l(x,y)}$ using the polynomials $\overline{f_j^l(x)}$ associated with $P_j \in CORE$ and reconstruct $\overline{s^l} = \overline{F^l(0,0)}$. Else reconstruct $\overline{s^l} = NULL$. Finally output $\overline{S} = (\overline{s^1}, \ldots, \overline{s^\ell})$ and terminate.

Table 3: **AWSS for Sharing Secret $S = (s^1, \ldots, s^\ell)$ with $n = 3t + 1$**

The properties of AWSS-Multiple-Secret follow from the properties of AWSS-Single-Secret.

Lemma 13 *Both AWSS-Multiple-Secret-Share and AWSS-Multiple-Secret-Rec privately communicates* $\mathcal{O}((\ell n^2 + n^3)\kappa)$ *bits and A-cast* $\mathcal{O}((\ell n^2 + n^3)\kappa)$ *bits.*

PROOF: AWSS-Multiple-Secret-Share executes at most $n + n^2 = \Theta(n^2)$ instances of Gen and Ver. Hence by Lemma 1, AWSS-Multiple-Secret-Share privately communicates $\mathcal{O}((\ell n^2 + n^3)\kappa)$ bits and A-casts $\mathcal{O}((\ell n^2 + n^3)\kappa)$ bits. Protocol AWSS-Multiple-Secret-Rec executes $\Theta(n^2)$ instances of Reveal and hence by Lemma 1, AWSS-Multiple-Secret-Rec privately communicates $\mathcal{O}((\ell n^2 + n^3)\kappa)$ bits and A-casts $\mathcal{O}((\ell n^2 + n^3)\kappa)$ bits. □

Theorem 3 *The pair (AWSS-Multiple-Secret-Share, AWSS-Multiple-Secret-Rec) constitutes a valid AWSS scheme for* $n = 3t + 1$ *parties, which shares* ℓ *secrets simultaneously and satisfies the properties of AWSS except with an error probability of* $2^{-\mathcal{O}(\kappa)}$.

IMPORTANT NOTE: As in AWSS-Single-Secret, in AWSS-Multiple-Secret, the degree-t univariate polynomial $F^l(x, 0) = f_0^l(x)$ is used to share the secret s^l for $l = 1, \ldots, \ell$. *In the following, we will say that* D *shares* ℓ *degree-t polynomials* $f^1(x), \ldots, f^\ell(x)$ *simultaneously using protocol AWSS-Multiple-Secret by executing AWSS-Multiple-Secret-Share*$(D, \mathcal{P}, f^1(x), \ldots, f^\ell(x))$. For this D selects a ℓ t-degree symmetric bivariate polynomials $F^1(x, y), \ldots, F^\ell(x, y)$, such that for $l = 1, \ldots, \ell, F^l(x, 0) = f^l(x)$ and executes the protocol. It should be noted that $f^l(x)$'s are not random but only preserve the secrecy of the constant terms, $s^l = f^l(0)$. Yet, this distribution of polynomials is sufficient to provide the secrecy requirements needed by our AVSS where AWSS is used as a building block. If D indeed selects the bivariate polynomials in the above way and follows the protocol steps correctly, then as a result of the above execution, party P_i in $WCORE$ holds the i^{th} share $f^l(i) = F^l(i, 0) = F^l(0, i)$, the polynomial $f_i^l(x) = F^l(x, i)$ and the share-share $f_j^l(i) = F^l(i, j) = f_i^l(j)$ corresponding to every other party P_j. Similarly, AWSS-Multiple-Secret-Rec$(D, \mathcal{P}, f^1(x), \ldots, f^\ell(x))$ can be used for the reconstruction of $f^1(x), \ldots, f^\ell(x)$ and hence the secrets $s^1 = f^1(0), \ldots, s^\ell = f^\ell(0)$.

APPENDIX D: Protocol AVSS-Multiple-Secret and Proof of its Properties

Protocol AVSS-Multiple-Secret is given in Table 4. As in AVSS-Single-Secret, Protocol AVSS-Multiple-Secret also does not force *corrupted* D to commit some *meaningful* secret (i.e., an element from $\mathbb{F}$). So, we may assume that if any element of D's committed secret is $NULL$, then D has committed some predefined value $s^* \in \mathbb{F}$, which is known publicly. Hence in AVSS-Multiple-Secret-Rec, whenever $NULL$ is reconstructed, every honest party replaces $NULL$ by the predefined secret s^*. In this way, we say that our AVSS scheme allows D to *commit* secret $S \in \mathbb{F}^\ell$.

The proofs of the properties of AVSS-Multiple-Secret follows from the proofs of AVSS-Single-Secret. So we have the following theorem.

Theorem 4 *The pair (AVSS-Multiple-Secret-Share, AVSS-Multiple-Secret-Rec) constitutes a valid AVSS scheme for sharing* $\ell \geq 1$ *secrets with* $n = 3t + 1$ *and satisfies the properties of AVSS except with an error probability of* $2^{-\mathcal{O}(\kappa)}$.

The communication complexity of AVSS-Multiple-Secret is given by the following lemma.

Lemma 14 *Both Protocol AVSS-Multiple-Secret-Share and Protocol AVSS-Multiple-Secret-Rec communicate* $\mathcal{O}((\ell n^3 + n^4)\kappa)$ *bits and A-cast* $\mathcal{O}((\ell n^3 + n^4)\kappa)$ *bits.*

PROOF: AVSS-Multiple-Secret-Share and AVSS-Multiple-Secret-Rec execute n instances of AWSS-Multiple-Secret-Share and AWSS-Multiple-Secret-Rec, respectively. Hence communication complexity of AVSS-Multiple-Secret-Share and AVSS-Multiple-Secret-Rec follows from Lemma 13. □

Protocol AVSS-Multiple-Secret$(D, \mathcal{P}, S)$

AVSS-Multiple-Secret-Share$(D, \mathcal{P}, S)$

DISTRIBUTION: CODE FOR D

1. Select ℓ degree-t random symmetric bivariate polynomials $F^1(x,y), \ldots, F^\ell(x,y)$ such that for $l = 1, \ldots, \ell$, $F^l(0,0) = s^l$.
2. For $l = 1, \ldots, \ell$, deliver $f_i^l(x) = F^l(x,i)$ to P_i.

COMMITMENT UPON VERIFICATION: CODE FOR P_i

1. Wait to obtain $f_i^l(x)$ for $l = 1, \ldots, \ell$ from D.
2. If for each $l = 1, \ldots, \ell$, $f_i^l(x)$ is a t-degree polynomial, then as a dealer, execute AWSS-Multiple-Secret-Share$(P_i, \mathcal{P}, (f_i^1(x), \ldots, f_i^\ell(x)))$ by selecting ℓ degree-t symmetric bivariate polynomials $Q^{(P_i,1)}(x,y), \ldots, Q^{(P_i,\ell)}(x,y)$ such that for $l = 1, \ldots, \ell$, $Q^{(P_i,l)}(x,0) = q_0^{(P_i,l)}(x) = f_i^l(x)$. We call this instance of AWSS-Multiple-Secret-Share initiated by P_i as AWSS-Multiple-Secret-Share^{P_i}.
3. As a part of the execution of AWSS-Multiple-Secret-Share^{P_j}, wait to receive $q_i^{(P_j,l)}(x)$ for $l = 1, \ldots, \ell$ from P_j. Then check $f_i^l(j) \stackrel{?}{=} q_i^{(P_j,l)}(0)$ for all $l \in \{1, \ldots, \ell\}$. If the test passes for all $l \in \{1, \ldots, \ell\}$, then participate in AWSS-Multiple-Secret-Share^{P_j} and act according to the remaining steps of AWSS-Multiple-Secret-Share^{P_j}. Here $j \in \{1, \ldots, n\}$.

CORE CONSTRUCTION : CODE FOR D: Same as in Protocol AVSS-Single-Secret-Share except that AWSS-Single-Secret-Share^{P_j} is replaced by AWSS-Multiple-Secret-Share^{P_j}.

CORE VERIFICATION & AGREEMENT ON CORE : CODE FOR P_i: Same as in Protocol AVSS-Single-Secret-Share except that AWSS-Single-Secret-Share^{P_j} is replaced by AWSS-Multiple-Secret-Share^{P_j}.

AVSS-Multiple-Secret-Rec$(D, \mathcal{P}, S)$: Public reconstruction of S towards each party:

SECRET RECONSTRUCTION: CODE FOR P_i

1. For every $P_j \in VCORE$, reconstruct P_j's *AWSS-Commitment* on $f_j^1(x), \ldots, f_j^\ell(x)$ as follows:
   (a) Participate in AWSS-Multiple-Secret-Rec$(P_j, \mathcal{P}, (f_j^1(x), \ldots, f_j^\ell(x)))$ with $WCORE^{P_j}$ and $OKSetP_k^{P_j}$ for every $P_k \in WCORE^{P_j}$. We call this instance of AWSS-Multiple-Secret-Rec as AWSS-Multiple-Secret-Rec^{P_j}.
   (b) For each $l \in \{1, \ldots, \ell\}$, wait for the termination of AWSS-Multiple-Secret-Rec^{P_j} with output either $\overline{Q^{(P_j,l)}(x,y)}$ or $NULL$.
2. Wait for the reconstruction of P_j's *AWSS-Commitment* for every $P_j \in VCORE$.
3. Add $P_j \in VCORE$ to $FINAL$ if AWSS-Multiple-Secret-Rec^{P_j} gives a non-$NULL$ output for each $l \in \{1, \ldots, \ell\}$. Now for $P_j \in FINAL$, assign $\overline{f_j^l(x)} = \overline{Q^{(P_j,l)}(x,0)}$.
4. For every $l \in \{1, \ldots, \ell\}$ do the following: for every pair $(P_\gamma, P_\delta) \in FINAL$ check $\overline{f_\gamma^l(\delta)} \stackrel{?}{=} \overline{f_\delta^l(\gamma)}$. If the test passes for every pair of parties then recover $\overline{F^l(x,y)}$ and reconstruct $\overline{s^l} = \overline{F^l(0,0)}$. Else reconstruct $\overline{s^l} = NULL$. Finally output $\overline{S} = (\overline{s^1}, \ldots, \overline{s^\ell})$ and terminate.

Table 4: **AVSS for Sharing Secret $S = (s^1, \ldots, s^\ell)$ with $n = 3t + 1$**