Asynchronous Computational VSS with Reduced Communication Complexity*

Michael Backes[1], Amit Datta[2], and Aniket Kate[3]

[1] Saarland University and MPI-SWS, Germany
backes@mpi-sws.org
[2] Carnegie Mellon University, U.S.A.
amitdatta@cmu.edu
[3] MMCI, Saarland University, Germany
aniket@mmci.uni-saarland.de

Abstract. Verifiable secret sharing (VSS) is a vital primitive in secure distributed computing. It allows an untrusted dealer to verifiably share a secret among n parties in the presence of an adversary controlling at most t of them. VSS in the synchronous communication model has received tremendous attention in the cryptographic research community. Nevertheless, recent interest in deploying secure distributed computing over the Internet requires going beyond the synchronous model and thoroughly investigating VSS in the asynchronous communication model.

In this work, we consider the communication complexity of asynchronous VSS in the computational setting for the optimal resilience of $n = 3t + 1$. The best known asynchronous VSS protocol by Cachin et al. has $O(n^2)$ message complexity and $O(\kappa n^3)$ communication complexity, where κ is a security parameter. We close the linear complexity gap between these two measures for asynchronous VSS by presenting two protocols with $O(n^2)$ message complexity and $O(\kappa n^2)$ communication complexity. Our first protocol satisfies the standard VSS definition, and can be used in stand-alone VSS scenarios as well as in applications such as Byzantine agreement. Our second and more intricate protocol satisfies a stronger VSS definition, and is useful in all VSS applications including multiparty computation and threshold cryptography.

Keywords: Verifiable Secret Sharing, Asynchronous Communication Model, Communication Complexity, Polynomial Commitments.

1 Introduction

The notion of secret sharing was introduced independently by Shamir [24] and Blakley [6] in 1979. For integers n and t such that $n > t \geq 0$, an (n, t)-*secret sharing* scheme is a method used by a *dealer* to share a secret s among a set of n parties in such a way that any subset of $t+1$ or more parties can compute the secret s, but subsets of size t or fewer cannot.

* An extended version of this paper is available [3].

E. Dawson (Ed.): RSA 2013, LNCS 7779, pp. 259–276, 2013.

In many applications of secret sharing, parties may need to verify the correctness of the values dealt in order to prevent malicious behavior by the dealer. To satisfy this requirement, Chor et al. [12] introduced the concept of *verifiable secret sharing* (VSS). With its applicability to Byzantine agreement, multiparty computation (MPC) and threshold cryptography, VSS has remained an important area of cryptographic research for the last two decades [9, 13, 15, 16, 21, 22].

Although the literature for VSS is vast, the notion of VSS in the asynchronous communication setting (no bounds on message transfer delays) has not yet received the deserved attention in terms of practical efficiency or theoretical lower bounds. Asynchronous VSS schemes with unconditional security have been developed [1, 5, 11, 20]; however, these schemes are prohibitively expensive for any realistic use as they need $\Omega(\kappa n^5)$ bits of communication for κ-bit secrets. In the computational security setting, Cachin et al. [9], Zhou et al. [25], and recently Schultz et al. [23] suggested more practical asynchronous VSS schemes: asynchronous verifiable secret sharing (AVSS), asynchronous proactive secret sharing (APSS) and mobile proactive secret sharing (MPSS), respectively. Of these, AVSS [9] is the most generic and practical asynchronous VSS scheme and it forms the basis for many practical threshold cryptographic protocols such as [17]. AVSS assimilates a bivariate polynomial into Bracha's deterministic reliable broadcast protocol [8], which results into its $O(n^2)$ message complexity (number of messages transferred) and $O(\kappa n^4)$ communication complexity (number of bits transferred) for the optimal resiliency condition of $n = 3t+1$. Cachin et al. [9] further refined the AVSS protocol to reduce the communication complexity to $O(\kappa n^3)$. Nevertheless, a further reduction in the communication complexity is not possible using similar techniques, and a linear complexity gap between the message complexity and the communication complexity still remains.

In this work, we bridge this gap. We present two *efficient* asynchronous VSS schemes (eAVSS and eAVSS-SC) with different properties (and correspondingly different utilities) with $O(n^2)$ message complexity and $O(\kappa n^2)$ communication complexity.

1.1 Our Contributions

Kate, Zaverucha and Goldberg [18] define the concept of commitments to polynomials, and devise two schemes $\mathsf{PolyCommit}_{\mathrm{DLog}}$ and $\mathsf{PolyCommit}_{\mathsf{Ped}}$ that commit to a univariate polynomial of degree t (or less) using a single element of size $O(\kappa)$. Their schemes work in the bilinear pairing setting under the t-strong Diffie–Hellman (t-SDH) assumption [7]. We use their $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme and a collision-resistant hash function to achieve our goal of asynchronous VSS with $O(\kappa n^2)$ communication complexity. Although we choose the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme that provides unconditional hiding (secrecy) instead of the much simpler $\mathsf{PolyCommit}_{\mathrm{DLog}}$ scheme that provides computational hiding against the discrete logarithm (DLog) assumption, our protocols work with the $\mathsf{PolyCommit}_{\mathrm{DLog}}$ scheme with no modification.

Nevertheless, the schemes we present are not a straightforward adaptation of the PolyCommit schemes to the bivariate polynomial-based AVSS scheme [9],

and not surprisingly, Kate et al. [18] left the applicability of PolyCommit to asynchronous VSS as an open problem. The reason for that, as we elaborate in Section 2.4, is that modifying the PolyCommit schemes to a scheme providing constant-size commitments to bivariate-polynomials used in asynchronous VSS seems difficult if not impossible.

We achieve our goal by taking an entirely different path, bypassing the open problem of obtaining constant-size commitments to bivariate polynomials. We realize asynchronous VSS in two steps: We first present a univariate polynomial-based asynchronous VSS scheme (eAVSS), which guarantees that at least $t+1$ honest parties receive proper shares of the secret, while the remaining honest parties are assured that at least $t+1$ honest parties have received correct shares and can reconstruct the shared secret. This construction is sufficient for stand-alone VSS and for applications such as asynchronous Byzantine agreement (ABA). For applications such as MPC and threshold cryptographic constructions, we then design an efficient stronger asynchronous VSS scheme (eAVSS-SC), which guarantees that every honest party receives its share during the sharing phase. In principle, this is possible by running $n+1$ instances of eAVSS; however, it asks for a broadcast of commitment vectors of size $O(\kappa n)$ which increases the communication complexity to $O(\kappa n^3)$. In eAVSS-SC, we overcome this barrier by aptly modifying the AVSS protocol flow and by hashing the commitments in the vector using a collision-resistant hash function and running a PolyCommit instance over the hashed values.

Our schemes have direct implications to the efficiency of all asynchronous VSS applications. Most prominently, using our eAVSS protocol in the modular ABA construction by Canetti and Rabin [11] it is possible to obtain the first $O(\kappa n^3)$ communication complexity ABA protocol, which is secure against the *adaptive* adversary in the *standard* model.

Organization. In Section 2, we describe our system model and provide a brief overview of the concepts of VSS, polynomial commitments and asynchronous VSS. In Section 3, we define and prove our basic asynchronous VSS protocol (eAVSS), while in Section 4, we define our main asynchronous VSS protocol (eAVSS-SC). In Section 5, we discuss a few interesting applications. An in-depth discussion of the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme and corresponding computational assumptions have been added in Appendix A.

2 Preliminaries

Our schemes work in the computational security setting. The adversary $\mathcal{A}$ is a probabilistic polynomial time (PPT) algorithm with respect to a security parameter κ unless stated otherwise. A function $\epsilon(\cdot) : \mathbb{N} \to \mathbb{R}^+$ is called *negligible* if for all $c > 0$ there exists a κ_0 such that $\epsilon(\kappa) < 1/\kappa^c$ for all $\kappa > \kappa_0$. Throughout the rest of this paper, $\epsilon(\cdot)$ denotes a negligible function.

We assume that the shared secret s lies over a finite field $\mathbb{F}_p$, where p is a κ-bit long prime. We use Shamir's *polynomial-based* secret sharing approach [24], where our polynomials belong to $\mathbb{F}_p[x]$ or $\mathbb{F}_p[x, y]$.

2.1 Asynchronous System Model

Following the adversary and communication model of AVSS given by Cachin et al. [9], we assume an asynchronous fully-connected network of n parties $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$, where every pair of parties is connected by an authenticated and private communication link. A special party $P_d \in \mathcal{P}$ works as a dealer. The indices for the parties are chosen from $\mathbb{F}_p$. Without loss of generality, we assume these indices to be $\{1, \ldots, n\}$.

The adversary $\mathcal{A}$ is *t-bounded* and it can coordinate the actions of up to t out of n parties. The adversary $\mathcal{A}$ is further assumed to be *adaptive*, and may corrupt a party of its choice at any instance during a protocol execution as long as its total number of corruptions is bounded by t. A party is said to be *honest* if the adversary has not corrupted it. In our asynchronous setting, the adversary $\mathcal{A}$ controls the network and may delay messages between any two honest parties. However, it cannot read or modify these messages, and it also has to eventually deliver all the messages by honest parties.

2.2 Verifiable Secret Sharing—VSS

In many secret sharing applications, a dealer may behave maliciously. This led to the conception of VSS [12].

Definition 1. *An* (n, t)-VSS *scheme among* n *parties in* $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ *with a distinguished party* $P_d \in \mathcal{P}$ *consists of two phases: the* sharing *(*Sh*) phase and the* reconstruction *(*Rec*) phase.*

Sh phase. *A dealer* P_d *distributes a secret* $s \in \mathbb{F}_p$ *among parties in* $\mathcal{P}$*. At the end of the* Sh *phase, each honest party* P_i *holds a share* s_i *of the distributed secret* s*.*

Rec phase. *In this phase, each party* P_i *sends its secret share* s'_i *to every party in* $\mathcal{P}$ *and a reconstruction function is applied in order to compute the secret* $s = \mathsf{Rec}(s'_1, s'_2, \ldots, s'_n)$ *or output* $\perp$ *indicating that* P_d *is malicious. For honest parties* $s'_i = s_i$*, while for malicious parties* s'_i *may be different from* s_i *or even absent.*

An (n, t)-VSS scheme has the following security requirements:

Secrecy. *If the dealer is honest, the adversary who can compromise* t *parties does not have any more information about* s *except what is implied by the public parameters.*

Correctness. *If* P_d *is honest, the reconstructed value should be equal to the dealer's secret* s*.*

Commitment. *Even if* P_d *is dishonest, there exists a value* $s^* \in \mathbb{F}_p \cup \{\perp\}$ *at the end of the* Sh *phase, such that all honest parties output* s^* *at the end of the* Rec *phase.*

In this paper, we consider VSS schemes where any malicious behaviour by P_d can be identified by the honest parties in the Sh phase itself and the commitment

property simplifies to the following: the reconstructed value z should be equal to a shared secret $s \in \mathbb{F}_p$ that gets fixed at the end of the Sh phase.

Many VSS applications (e.g., threshold cryptography and MPC) avoid participation by all parties once the Sh phase is over. It is required that messages from any $t+1$ honest parties (or any $2t+1$ parties) are sufficient to reconstruct the shared secret s. For these applications, we require a stronger commitment property that we refer as the *strong commitment* requirement.

Strong Commitment. *Even if P_d is dishonest, there exists a value $s^* \in \mathbb{F}_p$ at the end of the Sh phase, such that s^* is reconstructed regardless of the subset of parties (of size greater than $2t$) chosen by the adversary in the Rec phase.*

Some VSS schemes achieve a weaker (computational) secrecy guarantee.

Weak Secrecy. *A* t-limited *adversary who can compromise t parties cannot compute s during the Sh phase.*

We also give the following definitions for the complexity measures.

Definition 2 (Message Complexity). *The message complexity is defined as the total number of messages exchanged between the parties participating in a scheme.*

Definition 3 (Communication Complexity). *The communication complexity is defined as the total number of bits exchanged between the parties taking into consideration every message that has been transmitted.*

A variant of VSS considers dealer P_d to be an external party (i.e., $P_d \notin \mathcal{P}$) and allows the adversary to corrupt P_d and up to t additional parties in $\mathcal{P}$. All our protocols also work in this stronger setting.

Assuming a broadcast channel, Feldman [15] gave the first non-interactive and efficient VSS scheme and Pedersen [21, 22] presented a modification to it. Both protocols obtain the strong commitment property. In terms of secrecy, Feldman VSS achieves the weak secrecy property, while Pedersen VSS achieves the stronger form.

2.3 Use of Commitments in VSS

A verification mechanism for a consistent dealing is fundamental to VSS. It is achieved using distributed computing techniques in the information-theoretic security setting. In the computational setting that we focus in this paper, the commitment schemes provide an efficient alternative.

A commitment scheme allows an entity, the *committer*, to publish a value, called the *commitment* (say $\mathcal{C}$), which binds her to a message s (*binding*) without revealing it (*hiding*). Later, she may *open* the commitment $\mathcal{C}$ and reveal the committed message m to a verifier, who can check that the message is consistent

with the commitment. In particular, the computational VSS schemes utilize the commitments to the shared polynomials. Kate et al. [18] formalize this concept of polynomial commitments. Here, we present a refined version of their polynomial commitment (PolyCommit) definition for polynomial of degree $\leq t$.

Definition 4. *A* PolyCommit *scheme consists of the following algorithms:*

Setup$(1^\kappa, t)$ *generates system parameters* SP *to commit to a polynomial of degree* $\leq t$*. In these system parameters, let* $\mathbb{G}$ *be an algebraic structure for commitments.* Setup *is run by a trusted or distributed authority.* SP *can also be standardized for repeated use.*

Commit$(\mathbf{SP}, \phi(x))$ *outputs a commitment* $\mathcal{C}$ *to a polynomial* $\phi(x)$ *for the system parameters* SP*, and some associated decommitment information* d*. (In some constructions,* d *can be null.)*

Open$(\mathbf{SP}, \mathcal{C}, \phi(x), [d])$ *outputs the polynomial* $\phi(x)$ *used while creating the commitment, with decommitment information* d*.*

VerifyPoly$(\mathbf{SP}, \mathcal{C}, \phi(x), [d])$ *verifies that* $\mathcal{C}$ *is a commitment to* $\phi(x)$*, created with decommitment information* d*. If so, the algorithm outputs* 1*, otherwise it outputs* 0*.*

CreateWitness$(\mathbf{SP}, \phi(x), i, [d])$ *outputs* $\langle i, \phi(i), w_i, d_i\rangle$*, where* w_i *is a witness and* d_i *is the decommitment information for the evaluation* $\phi(i)$ *of* $\phi(x)$ *at the index* i*. This algorithm is optional.*

VerifyEval$(\mathbf{SP}, \mathcal{C}, i, \phi(i), [d_i, w_i])$ *verifies that* $\phi(i)$ *is indeed the evaluation at the index* i *of the polynomial committed in* $\mathcal{C}$*. If so, the algorithm outputs* 1*, otherwise it outputs* 0*.*

Given $\mathsf{SP} \leftarrow \mathsf{Setup}(1^\kappa, t)$, a PolyCommit scheme satisfies the following properties:

Correctness. *Let* $\mathcal{C} \leftarrow$ *Commit*$(SP, \phi(x))$*. For* $\mathcal{C}$ *generated by* *Commit*$(SP, \phi(x))$*, and all* $\phi(x) \in \mathbb{Z}_p[x]$*, any* $\langle i, \phi(i), w_i, d_i\rangle$ *generated using* *CreateWitness*$(SP, \phi(x), i)$ *is correctly verified by* *VerifyEval*$(SP, \mathcal{C}, i, \phi(i), d_i, , w_i)$*.*

Strong Correctness. $\forall \mathcal{A} : \Pr\{(\mathcal{C}, \langle \phi(x), d\rangle) \leftarrow \mathcal{A}(SP) : \deg(\phi(x)) > t\} = \epsilon(\kappa)$.

Polynomial Binding. $\forall \mathcal{A} : \Pr\{(\mathcal{C}, \langle \phi(x), d\rangle, \langle \phi'(x), d'\rangle) \leftarrow \mathcal{A}(SP)\} = \epsilon(\kappa)$ *given* $\big[(\textit{VerifyPoly}(SP, \mathcal{C}, \phi(x), d) = 1) \wedge (\textit{VerifyPoly}(SP, \mathcal{C}, \phi'(x), d') = 1) \wedge (\phi(x) \neq \phi'(x))\big]$

Evaluation Binding. $\forall \mathcal{A} : \Pr\{(\mathcal{C}, \langle i, \phi(i), d_i, w_i\rangle, \langle i, \phi(i)', d_i', w_i'\rangle) \leftarrow \mathcal{A}(SP)\} = \epsilon(\kappa)$ *given* $\big[(\textit{VerifyEval}(SP, \mathcal{C}, i, \phi(i), d_i, w_i) = 1) \wedge (\textit{VerifyEval}(SP, \mathcal{C}, i, \phi(i)', d_i', w_i') = 1) \wedge (\phi(i) \neq \phi(i)')\big]$

(Unconditional) Hiding. *For* $\phi(x) \in_R \mathbb{Z}_p[x]$*, given* $\langle SP, \mathcal{C}\rangle$ *and* $\{\langle i_j, \phi(i_j), d_{i_j}, w_{\phi_{i_j}}\rangle : j \in [1, \deg(\phi)]\}$ *such that* *VerifyEval* $(SP, \mathcal{C}, i_j, \phi(i_j), d_{i_j}, w_{\phi_{i_j}}) = 1$ *for each* j*, a computationally unbounded adversary* $\hat{\mathcal{A}}$ *has no information about* $\phi(\hat{j})$ *for any unqueried index* $\hat{j}$*.*

The above strong correctness property is not present in the original PolyCommit definition. We include it as restricting degree of the committed polynomial by a threshold t is required for VSS. Further, a weaker form of hiding is also possible,

where a computationally bounded adversary $\mathcal{A}$ cannot compute $\phi(\hat{j})$ for any unqueried index $\hat{j}$. We consider the unconditional hiding property in the paper.

In literature, VSS protocols utilized commitments to the *coefficients* or *evaluations* of shared polynomials as polynomial commitments. They used two commitment schemes. Given g and h as two random generators of a multiplicative group of order p, Feldman VSS and its variants use a commitment scheme of the form g^s with computational hiding under the discrete logarithm (DLog) assumption and unconditional binding. Pedersen [21] presented another commitment of the form $g^s h^r$ with unconditional hiding but computational binding under the DLog assumption. The hiding property of the commitment scheme leads to the secrecy property of VSS, while the binding property leads to the correctness property of VSS. Both of these commitment schemes also trivially satisfy the commitment property of VSS by the fact that the size of a commitment to a polynomial $\phi(x) \in \mathbb{Z}_p[x]$ is equal to $\deg(\phi) + 1$. In the complexity terms, the size of commitment is $O(n)$ (since for optimal resiliency, $\deg(\phi) = t = \lfloor \frac{n-1}{2} \rfloor$). However, the commitment to a shared polynomial has to be broadcast to all parties, which results in a linear-size broadcast for Feldman VSS, and a linear complexity gap between the message and the communication complexities.

Kate et al. [18] close this gap for Feldman VSS and its variants using a commitment that commits to the entire univariate polynomial using a single element. In particular, they define two polynomial commitment ($\mathsf{PolyCommit}$) schemes: $\mathsf{PolyCommit}_{\mathrm{DLog}}$ and $\mathsf{PolyCommit}_{\mathsf{Ped}}$, both of which works in the bilinear pairing setting with $\Theta(t)$ system parameters. $\mathsf{PolyCommit}_{\mathrm{DLog}}$ attains hiding under the DLog assumption, binding under the t-strong Diffie-Hellman (t-SDH) assumption [7], and strong correctness under the t-polynomial Diffie-Hellman assumption (refer to Appendix A for references to any assumption). Using a technique similar to Pedersen commitments, they also define $\mathsf{PolyCommit}_{\mathsf{Ped}}$, which attains unconditional hiding and computational binding under the t-SDH assumption. These constructions are based on an algebraic property of polynomials $\phi(x) \in \mathbb{F}_p[x]$ that $(x - i)$ *perfectly* divides the polynomial $\phi(x) - \phi(i)$ for any $i \in \mathbb{F}_p$.

In this work, we extend the utility of the $\mathsf{PolyCommit}$ concept to asynchronous VSS. We choose the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme for our protocol as it provides unconditional hiding and include the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ construction in Appendix A.

2.4 Asynchronous VSS

The asynchronous communication setting places no bounds on message delays. Consequently, there is no trivially available broadcast channel, and Feldman VSS and its variants do not guarantee a correct completion. This gives rise to the concept of asynchronous VSS for optimal resilience of $n = 3t + 1$.

An asynchronous VSS protocol requires the liveness and agreement properties along with the secrecy, correctness and commitment properties defined in Section 2.2

Definition 5. *An* asynchronous VSS *protocol having $n \geq 3t + 1$ parties with a t-limited Byzantine adversary satisfies the following conditions:*

Liveness. *If the dealer P_d is honest in the* Sh *phase, then all honest parties complete the* Sh *phase.*

Agreement. *If some honest party completes the* Sh *phase, then all honest parties will eventually complete the* Sh *phase. If all honest parties subsequently start the* Rec *phase, then all honest parties will complete the* Rec *phase.*

Correctness, Commitment and Secrecy. *as defined in Section 2.2.*

For VSS applications such as MPC, we need VSS that has identical secrecy, correctness, liveness and agreement properties as in Definition 5, but a stronger *commitment* property as defined in Section 2.2. In other words, there exists a t-degree polynomial $f(x)$ such that a share s_i held by every honest party P_i at the end of the sharing phase is equal to $f(i)$.

As discussed in the introduction, three computational VSS schemes have been suggested for the asynchronous setting: AVSS [9], APSS [25], and MPSS [23]. Of these, AVSS [9] provides the first and the most practical asynchronous VSS scheme. In the AVSS methodology, secret sharing is integrated into a reliable broadcast primitive [8]. This results into its $O(n^2)$ messages complexity. Here, the commitments to the secret and its shares are broadcast, and the shares themselves are appropriately appended to the broadcast commitments so that parties receive their shares while maintaining their secrecy. To overcome an adversarial dealer that does not provide some honest party with its correct share, parties send sub-shares to each other along with the broadcasted commitment. The victim party then computes its share from the received sub-shares. AVSS implements this using bivariate polynomial-based secret sharing, which leads to a commitment (or broadcast) of size $\Theta(\kappa n^2)$ and correspondingly $O(\kappa n^4)$ bits of communication. In the same paper, Cachin et al. improve their AVSS scheme by reducing the commitment-size to $\Theta(\kappa n)$, which results in $O(\kappa n^3)$ bits of communication. A linear gap between the message complexity and the communication complexity still remains.

A Mismatch between AVSS and PolyCommit. It is tempting to consider filling this gap for AVSS using a bivariate PolyCommit scheme that commits to an entire bivariate polynomial using a constant-size commitment; however, this does not seem to be possible with the existing PolyCommit methodology. PolyCommit schemes use the algebraic property that, for $\phi(x) \in \mathbb{F}_p[x]$, $(x - i)$ perfectly divides the polynomial $\phi(x) - \phi(i)$ for any $i \in \mathbb{F}_p$. However, such a perfect and direct relation is not known between a bivariate polynomial $\phi(x, y)$ and its evaluations $\phi(i, j)$ for any $i, j \in \mathbb{F}_p$.[1] Therefore, we will have to use two-stage properties involving univariate polynomials (e.g., $(x-i)(y-j)$ perfectly divides the polynomial $\phi(x, y)-\phi(i, y)-\phi(x, j)+\phi(i, j)$ for any $i, j \in \mathbb{F}_p$). However, this does not work either because even though the t-SDH problem to find $\langle c, g^{\frac{1}{\alpha+c}} \rangle$ for any value of $c \in \mathbb{Z}_p$ given $\langle g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^t} \rangle$ is conjectured to be hard, its exponential version to find a pair $\langle g^c, g^{\frac{1}{\alpha+c}} \rangle$ is easy.

[1] This is equivalent to derivatives in calculus, where complete derivation of a multivariable equation is not possible and partial derivatives are employed.

A closer look at AVSS reveals that further reducing the commitment-size in the hash-based approach of Cachin et al. using a univariate PolyCommit scheme also does not work: Cachin et al. hash the shares (or the univariate polynomials) for n parties and the secret. These $n+1$ hashed values sent to each party constitute a polynomial of degree n instead of degree t of the underlying bivariate polynomial. This requires an honest party to wait for (constant-size) messages from all n parties in AVSS, which is impossible in the asynchronous setting.

As a result, we have to work towards our goal of asynchronous VSS with $O(\kappa n^2)$ in a different way. In the next section, we provide an asynchronous VSS that satisfies the basic VSS definition, and extend it to a stronger version with applicability in all known VSS applications in Section 4.

3 eAVSS: Asynchronous VSS Protocol

In this section, we present a protocol (eAVSS) with $O(n^2)$ message complexity and $O(\kappa n^2)$ communication complexity and that satisfies Definition 5 of asynchronous VSS. The eAVSS protocol guarantees that at least $t+1$ honest parties receive proper shares of the secret committed using a t-degree univariate polynomial during the Sh phase, while the remaining honest parties are assured that there are at least $t+1$ honest parties that have received correct shares and can complete the Rec phase. The protocol is sufficient for applications such as Byzantine agreement and stand-alone VSS. The protocol construction is significantly simpler than the AVSS protocol [9] and it has a protocol flow similar to a VSS protocol for non-homomorphic commitments [4].

3.1 Construction

We assume a $\mathsf{PolyCommit}_{\mathsf{Ped}}$ commitment Setup instance $\mathsf{SP} \leftarrow \mathsf{Setup}(1^\kappa, t)$. We choose $\mathsf{PolyCommit}_{\mathsf{Ped}}$ due to its unconditional hiding property and the constant size of the commitments. It can, however, be replaced by any polynomial commitment scheme.

The dealer P_d starts off the protocol by choosing a *univariate* polynomial $\phi(x)$ with $\phi(0) = s$, and computing a commitment $\langle \mathcal{C}, d \rangle \leftarrow \mathsf{Commit}(\mathsf{SP}, \phi(x))$ and corresponding witnesses $w_j \leftarrow \mathsf{CreateWitness}(\mathsf{SP}, \phi(x), d, j)$ for $j \in [1, n]$. In $\mathsf{PolyCommit}_{\mathsf{Ped}}$, the decommitment information d is a t-degree polynomial, which is represented as $\widehat{\phi}(x)$ in the rest of the paper. P_d then sends $(\mathsf{send}, \mathcal{C}, \phi(j), \widehat{\phi}(j), w_j)$ messages to all parties and the parties verify their shares against the received commitment $\mathcal{C}$. In the rest of the protocol, the parties try to agree on $\mathcal{C}$. Unlike AVSS, the parties in eAVSS do not exchange their common evaluations of a bivariate polynomial; they only verify consistency of the received shares (if any) with $\mathcal{C}$ locally. If the dealer is dishonest, some honest parties may not receive shares consistent with $\mathcal{C}$; however, they still help to reach an agreement on $\mathcal{C}$ once they are assured that at least $t+1$ honest parties have received shares and witnesses consistent with $\mathcal{C}$. We describe the protocol in Figure 1. Note that commitment $\mathcal{C}$ is set to $\perp$ initially. An honest party accepts only one message

Sh Phase

Dealer P_d with the secret s

- Select a polynomial $\phi(x)$ of degree t such that $\phi(0) = s$.
- Compute a commitment $\langle \mathcal{C}, \widehat{\phi}(x) \rangle \leftarrow \mathsf{Commit}(\mathsf{SP}, \phi(x))$ and witnesses $w_j \leftarrow \mathsf{CreateWitness}\ (\mathsf{SP}, \phi(x), \widehat{\phi}(x), j)$ for $j \in [1, n]$.
- Send $(\mathsf{send}, \mathcal{C}' = \mathcal{C}, \phi(j), \widehat{\phi}(j), w_j)$ to every party P_j.

Every party P_i

- On receiving a message $(\mathsf{send}, \mathcal{C}', \phi(i), \widehat{\phi}(i), w_i)$ from P_d, run $\mathsf{VerifyEval}(\mathsf{SP}, \mathcal{C}', i, \phi(i), \widehat{\phi}(i), w_i)$. If the verification succeeds, set $\mathcal{C} = \mathcal{C}'$ and send $(\mathsf{echo}, \mathcal{C}')$ to all parties.
- On receiving $(\mathsf{echo}, \mathcal{C}')$ messages from $(n - t)$ parties:
  - For $\mathcal{C}' = \mathcal{C}$, send $(\mathsf{ready}, \mathsf{share}, \mathcal{C}')$ to all parties;
  - For $\mathcal{C}' \neq \mathcal{C}$, discard $\langle \phi(j), \widehat{\phi}(j), w_j \rangle$, set $\mathcal{C} = \mathcal{C}'$, and send $(\mathsf{ready}, \mathsf{no\text{-}share}, \mathcal{C}')$ to all parties.
- If a ready message has not been sent, then on receiving $(\mathsf{ready}, *, \mathcal{C}')$ messages from $t + 1$ parties:
  - For $\mathcal{C}' = \mathcal{C}$, send $(\mathsf{ready}, \mathsf{share}, \mathcal{C}')$ to all parties;
  - For $\mathcal{C}' \neq \mathcal{C}$, discard $\langle \phi(j), \widehat{\phi}(j), w_j \rangle$, set $\mathcal{C} = \mathcal{C}'$, and send $(\mathsf{ready}, \mathsf{no\text{-}share}, \mathcal{C}')$ to all parties.
- On receiving $(\mathsf{ready}, *, \mathcal{C}')$ messages from at least $(n-t)$ parties such that share flags are set in at least $t+1$ of those, complete the Sh phase with commitment $\mathcal{C} = \mathcal{C}'$ (and $\langle \phi(i), \widehat{\phi}(i), w_i \rangle$ if present).

Rec Phase

Every party P_i

- Send $(\mathsf{rec\text{-}share}, \phi(i), \widehat{\phi}(i), w_i)$ to all parties, if it has sent a $(\mathsf{ready}, \mathsf{share}, \mathcal{C})$ message in the Sh phase.
- On receiving $t + 1$ rec-share messages verified using $\mathsf{VerifyEval}(\mathsf{SP}, \mathcal{C}, j, \phi(j), \widehat{\phi}(j), w_j)$, interpolate the secret as $s = \phi(0)$.

Fig. 1. Protocol eAVSS for Asynchronous VSS ($n \geq 3t + 1$)

of a kind from any other party, and without loss of generality, we assume that every party chooses only the first message.

The protocol requires $O(n^2)$ messages as decided by its echo and ready messages. Use of PolyCommit ensures that all messages are of a constant size, and results in $O(\kappa n^2)$ communication complexity.

3.2 Analysis

Theorem 1. *Given a PolyCommit scheme that satisfies Definition 4, eAVSS is an asynchronous VSS protocol that satisfies Definition 5.*

Proof. To prove the theorem, we show that protocol eAVSS satisfies liveness, agreement, correctness, commitment, secrecy properties of asynchronous VSS

according to Definition 5. Our analysis is based on the properties of the polynomial commitment scheme used.

We start by proving the following two claims.

Claim. If some honest party has agrees on $\mathcal{C}$, then every honest party will eventually agree on $\mathcal{C}$.

Proof. We first prove by contradiction that if P_i be the first honest party to send ready message containing $\mathcal{C}$, then a ready message sent by every other honest party P_j will contain $\mathcal{C}$. Assume an honest party P_j sends a ready message with $\overline{\mathcal{C}}$ such that $\mathcal{C} \neq \overline{\mathcal{C}}$. Being first honest party to send a ready message with $\mathcal{C}$ party P_i must have received $(\mathsf{echo}, \mathcal{C})$ from at least $n-t$ parties of which at least $n-2t$ were honest. P_j can send $\overline{\mathcal{C}}$ only after one of the following two events and in both cases we arrive at a contradiction:

1. P_j can send $(\mathsf{ready}, \overline{\mathcal{C}})$ after receiving $(\mathsf{echo}, \overline{\mathcal{C}})$ from at least $n-t$ parties. As $n \geq 3t+1$, $(n-t)+(n-t)-n = n-2t \geq t+1$ parties must have sent echo with both $\mathcal{C}$ and $\overline{\mathcal{C}}$. This implies that at least one *honest* party sent echo messages of two types, which is impossible.
2. P_j can also send $(\mathsf{ready}, \overline{\mathcal{C}})$ after receiving $n-2t$ $(\mathsf{ready}, *, \overline{\mathcal{C}})$ messages. For $n \geq 3t+1$, $n-2t \geq t+1$. Therefore, there is at least one honest party (say P_k), who sent $\overline{\mathcal{C}}$ in its ready message to P_j. This means that one of the events (1) or (2) must have occurred with the honest party P_k. If we argue in a recursive manner, we reach some honest party who must have experienced event (1), which is a contradiction.

Therefore, no two honest parties will send ready messages containing different commitments.

A honest party agrees on $\mathcal{C}$ only after receiving at least $n-t$ ready messages such that at least $t+1$ contain share. Therefore, $n-2t \geq t+1$ honest parties must have sent ready message and at least one honest party must have sent a ready message containing share. ready messages from $t+1$ or more parties will eventually reach all remaining honest parties and they will send ready messages with the same $\mathcal{C}$, as discussed above. As the number of honest parties is at least $n-t$, every honest party will receive at least $n-t$ ready messages.

It, however, remains to be shown that every honest party will eventually receive at least $t+1$ ready messages with the share flag. From the above paragraph, we know that at least one honest party must have sent a ready message for $\mathcal{C}$ after receiving $n-t$ echo messages for $\mathcal{C}$ and, out of those, at least $n-2t \geq t+1$ are sent by honest parties. As an honest party sends an echo message only after receiving a verified send message from the dealer, at least $t+1$ honest parties must have received their shares from the dealer. As every honest party eventually sends a ready message, these $t+1$ parties will also certainly send ready messages and importantly, they will contain the share flag. Therefore, every honest party will eventually receive $n-t$ ready messages for $\mathcal{C}$ and at least $t+1$ among them will have share flags and thereby agree on $\mathcal{C}$.

Claim. If some honest party agrees on $\mathcal{C}$, then there exists a subset of at least $n - 2t \geq t + 1$ honest parties such that each of those holds an evaluation of a degree-t polynomial consistent with $\mathcal{C}$.

Proof. From the proof of Claim 3.2, $n - 2t$ honest parties will eventually send out ready messages for $\mathcal{C}$ with share; these $n - 2t$ honest parties have received verifiable send messages for $\mathcal{C}$ from the dealer P_d. Note that these honest parties never update $\mathcal{C}$, and eventually agree on the same $\mathcal{C}$ by Claim 3.2. Due to the strong correctness and polynomial binding properties of $\mathsf{PolyCommit}_{\mathsf{Ped}}$, there is a unique t-degree of polynomial $\phi(x)$ committed by $\mathcal{C}$. Therefore, evaluations available with these $n-2t \geq t+1$ parties implicitly defines $\phi(x)$ that is consistent with $\mathcal{C}$.

Liveness. If the dealer P_d is honest, then every honest party will eventually receive verifiable send message sent by P_d and will send an echo message and then a ready message. As there are $n-t \geq 2t+1$ honest parties, they will finally agree on $\mathcal{C}$ and complete the Sh phase.

Agreement. A party completes its Sh phase as soon as it agrees on a commitment $\mathcal{C}$. Claim 3.2 suggests that if an honest party agrees on $\mathcal{C}$, then every honest party will eventually agree on $\mathcal{C}$. Therefore, if one honest party completes the Sh phase, then every honest party will complete its Sh phase.

For agreement in the Rec phase, Claim 3.2 shows that there is a subset of at least $t + 1$ honest parties each holding an evaluation of a degree-t polynomial $\phi(x)$ that is consistent with $\mathcal{C}$. As every honest party participates in the Rec phase, $t + 1$ correct evaluations of $\phi(x)$ associated with $\mathcal{C}$ are available in the Rec phase, and the secret $s = \phi(0)$ can be interpolated by every honest party.

Correctness. Assume that the dealer has shared a secret s using a polynomial $\phi(x)$, and has remained honest throughout the execution of the Sh phase. Let $\mathcal{C}$ be the commitment to $\phi(x)$ sent by the dealer. Given correctness of the polynomial commitment scheme, all honest parties will receive correct shares of the secret s that is consistent with $\mathcal{C}$. Therefore, as we discussed above for agreement, the same secret s will be reconstructed by the parties.

Commitment. We prove the commitment by contradiction. Assume that two different honest parties P_i and P_j reconstruct different s' and s'' such that $s' \neq s''$, The maximum possible degree of the committed polynomial is t due to strong correctness of $\mathsf{PolyCommit}_{\mathsf{Ped}}$. Therefore, each of them must have agreed upon different commitments (say) $\mathcal{C}'$ and $\mathcal{C}''$ in the Sh phase. However, this contradicts with Claim 3.2. Therefore, a unique value $s^* \in \mathbb{F}_p$ will be reconstructed by all honest parties.

Secrecy. To prove secrecy, we need to show that if dealer P_d is honest, then the adversary $\mathcal{A}$ gains no information about the secret s. A t-limited adversary will be able to obtain t messages of the form (send, $\mathcal{C}$, w_i, $\phi(i)$). Due to the hiding property for polynomial commitments, given only t such messages it is impossible to reconstruct polynomial $\phi(x)$ (of degree t) and correspondingly the dealer's secret $s = \phi(0)$.

4 eAVSS-SC: AVSS Protocol with Strong Commitment

Although protocol eAVSS in Section 3 does not attain the strong commitment property, it can be used as a component of a VSS protocol that satisfies it. The most intuitive way to realize such a VSS scheme is to make P_d execute $(n+1)$ correlated instances of eAVSS, where the secret s is shared using the first instance (say) eAVSS_0 and the associated shares or polynomial evaluations for all n parties in eAVSS_0 are themselves shared using n instances eAVSS_j for $j \in [1, n]$. Once all eAVSS_j instances complete their Sh phases, a subset of $t+1$ or more honest parties provide every P_j its share in eAVSS_0 by running the Rec phase of eAVSS_j, and by sending their verifiable shares of eAVSS_j to only P_j. It is possible to combine send, echo and ready messages for all $n+1$ instances to keep the message complexity the same as that of AVSS and eAVSS, i.e., $O(n^2)$. However, to broadcast all associated commitments, the communication complexity becomes $O(\kappa n^3)$, which is no better than that of AVSS [9]. In protocol eAVSS-SC, we overcome this drawback using a collision-resistant hash function.

4.1 Construction

Here, the dealer P_d shares the secret s using a symmetric bivariate polynomial $\phi(x, y)$ such that $\phi(0, 0) = s$. The dealer commits to this bivariate polynomial using the univariate PolyCommit scheme twice. In Section 2.4, we observed that constant-size commitments to bivariate polynomials seem difficult, if not impossible. Here, we overcome this hurdle using PolyCommit over the hashed univariate PolyCommit values.[2] We provide an expository description of protocol eAVSS-SC in Figure 2. Notice that although we use send, echo, and ready messages similar to AVSS, our message structures and their utilities are significantly different from those of AVSS. These message structures are crucial to adopt a univariate $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme to our asynchronous VSS scheme, which uses bivariate polynomials.

The protocol requires two $\mathsf{PolyCommit}_{\mathsf{Ped}}$ instances: $\mathsf{SP}_1 \leftarrow \mathsf{Setup}(1^\kappa, t)$ and $\mathsf{SP}_2 \leftarrow \mathsf{Setup}(1^\kappa, n)$. P_d runs $n+1$ eAVSS instances with polynomials $\phi(x, 0)$, $\phi(x, 1)$, ..., $\phi(x, n)$. Let $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_n$ be the commitments for these $n+1$ instances. P_d also computes an n-degree polynomial $h_{\mathcal{C}}(x)$ from $H(\mathcal{C}_0), H(\mathcal{C}_1), \ldots, H(\mathcal{C}_n)$, where $H : \mathbb{G} \rightarrow \mathcal{F}_p$ is a collision-resistant hash function and broadcasts a commitment ζ to $h_{\mathcal{C}}(x)$. The dealer cannot cheat with $\phi(x, y)$ as the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme is binding and the hash function is collision-resistant. When all honest parties agree on ζ, they implicitly agree on $\mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_n$. As $t+1$ or more honest parties have received all required shares $\phi_i(x) = \phi(x, i)$ and $\hat{\phi}_i(x) = \hat{\phi}(x, i)$, and commitments $\boldsymbol{\mathcal{C}}$, they can provide all parties their required shares, commitments and witnesses in a verifiable manner using the homomorphic property of $\mathsf{PolyCommit}_{\mathsf{Ped}}$. We optimize this final step by attaching the required shares, witnesses and commitments to the ready messages.

[2] Note that our scheme is not a generic constant-size commitment scheme for bivariate polynomials and some care has to be taken before applying it in other applications; e.g., our scheme cannot be applied to the main as well as the refined AVSS protocols [9] without making their computational complexity exponential $O\binom{n}{t}$.

Sh Phase

Dealer P_d with the secret s

- Choose a symmetric t-degree bivariate polynomial $\phi(x,y)$ such that $\phi(0,0) = s$ and $\phi(i,j) = \phi(j,i)$.
- Commit to $\phi(x,y)$ using a vector $\boldsymbol{\mathcal{C}} = \{\mathcal{C}_j\}_{j\in[0,n]}$, where $\langle \mathcal{C}_j, \hat{\phi}_j(x)\rangle \leftarrow \mathsf{Commit}(\mathsf{SP}_1, \phi_j(x))$, $\phi_j(x) = \phi(x,j)$ and $\hat{\phi}(x,y)$ is symmetric. Also, compute witness vectors $\overrightarrow{\mathcal{W}_j} = \{w_j^k\}_{k\in[0,n]}$ for every party P_j such that $w_j^k \leftarrow \mathsf{CreateWitness}(\mathsf{SP}_1, \phi_k(x), \hat{\phi}_k(x), j)$.
- Compute an n-degree polynomial $h_{\mathcal{C}}(x)$ from $H(\mathcal{C}_0), H(\mathcal{C}_1), \ldots, H(\mathcal{C}_n)$, where $H : G \rightarrow \mathcal{F}_p$ is a collision-resistant hash function and commit to it $\langle \zeta, \widehat{h}_{\mathcal{C}}(x)\rangle \leftarrow \mathsf{Commit}(\mathsf{SP}_2, h_{\mathcal{C}}(x))$
- Send $(\mathsf{send}, \zeta' = \zeta, \boldsymbol{\mathcal{C}'} = \boldsymbol{\mathcal{C}}, \widehat{h}_{\mathcal{C}}(x), \overrightarrow{\mathcal{W}_j}, \phi_j(x), \hat{\phi}_j(x))$ to every party P_j.

Every party P_i

- On receiving $(\mathsf{send}, \zeta', \boldsymbol{\mathcal{C}'}, \widehat{h}_{\mathcal{C}}(x), \overrightarrow{\mathcal{W}_i}, \phi_i(x), \hat{\phi}_i(x))$ from P_d, verify its correctness:
  - interpolate the complete $\boldsymbol{\mathcal{C}'}$ from any of its $t+1$ elements to assert the degree t of the polynomial;
  - compute $h_{\mathcal{C}}(x)$ from $\boldsymbol{\mathcal{C}'}$ and $\mathsf{VerifyPoly}(\mathsf{SP}_2, \zeta', h_{\mathcal{C}}(x), \widehat{h}_{\mathcal{C}}(x))$;
  - $\mathsf{VerifyPoly}(\mathsf{SP}_1, \mathcal{C}'_i, \phi_i(x), \hat{\phi}_i(x))$;
  - $\mathsf{VerifyEval}(\mathsf{SP}_1, \mathcal{C}'_j, i, \phi_j(i)\big[=\phi_i(j)\big], \hat{\phi}_j(i)\big[=\hat{\phi}_i(j)\big], w_i^j)$ for every $j \in [0,n]$.

  Upon a successful verification, set $\zeta = \zeta'$ and $\boldsymbol{\mathcal{C}} = \boldsymbol{\mathcal{C}'}$, compute witnesses $w_j^i \leftarrow \mathsf{CreateWitness}(\mathsf{SP}_1, \phi_i(x), \hat{\phi}_i(x), j)$ for $j \in [1,n]$ and $w_i^{\mathcal{C}} \leftarrow \mathsf{CreateWitness}(\mathsf{SP}_2, h_{\mathcal{C}}(x), \widehat{h}_{\mathcal{C}}(x), i)$. Send a message (echo, ζ') to all parties.
- On receiving (echo, ζ') from at least $(n-t)$ parties:
  - If $\zeta' = \zeta$, send $(\mathsf{ready}, \zeta', \mathsf{share}, \phi_i(j), \hat{\phi}_i(j), w_j^i, \mathcal{C}_i, \widehat{h}_{\mathcal{C}}(i), w_i^{\mathcal{C}})$ to every party P_j;
  - If $\zeta' \neq \zeta$, discard $\langle \boldsymbol{\mathcal{C}}, \overrightarrow{\mathcal{W}_i}, \phi_i(x), \hat{\phi}_i(x)\rangle$, set $\zeta = \zeta'$, and send $(\mathsf{ready}, \zeta', \mathsf{no\text{-}share})$ to all parties.
- If a ready message has not been sent, then on receiving $(\mathsf{ready}, \zeta', *)$ messages from $(t+1)$ parties:
  - If $\zeta' = \zeta$, send $(\mathsf{ready}, \zeta', \mathsf{share}, \phi_i(j), \hat{\phi}_i(j), w_j^i, \mathcal{C}_i, \widehat{h}_{\mathcal{C}}(i), w_i^{\mathcal{C}})$ to every party P_j;
  - If $\zeta' \neq \zeta$, discard $\langle \boldsymbol{\mathcal{C}}, \overrightarrow{\mathcal{W}_i}, \phi_i(x), \hat{\phi}_i(x)\rangle$, set $\zeta = \zeta'$, and send $(\mathsf{ready}, \zeta', \mathsf{no\text{-}share})$ to all parties.
- On receiving $(\mathsf{ready}, \zeta', *)$ messages from at least $(n-t)$ parties such that at least $(t+1)$ of those messages contain $\langle \mathsf{share}, \phi_j(i), \hat{\phi}_j(i), w_i^j, \mathcal{C}_j, \hat{h}_{\mathcal{C}}(j), w_{\mathcal{C}}^j\rangle$ successfully verified using $\mathsf{VerifyEval}(\mathsf{SP}_1, \mathcal{C}_j, i, \phi_j(i), \hat{\phi}_j(i), w_i^j)$ and $\mathsf{VerifyEval}(\mathsf{SP}_2, \zeta', j, H(\mathcal{C}_j), \hat{h}_{\mathcal{C}}(j), w_{\mathcal{C}}^j)$, interpolate
  - shares $\phi_0(i)$ and $\hat{\phi}_0(i)$ from respectively $(t+1)$ $\phi_j(i)$ and $(t+1)$ $\hat{\phi}_j(i)$ values,
  - commitment $\mathcal{C}_0$, witness w_i^0 from respectively $(t+1)$ $\mathcal{C}_j$ and $(t+1)$ w_i^j values.

  Complete the Sh phase with $(\zeta = \zeta', \mathcal{C}_0, \phi_0(i), \hat{\phi}_0(i), w_i^0)$ as output.

Rec Phase

Every party P_i

- Send a message $(\mathsf{rec\text{-}share}, \phi_0(i), \hat{\phi}_0(i), w_i^0)$ to every party P_j.
- On receiving $t+1$ rec-share messages that have been verified using $\mathsf{VerifyEval}(\mathsf{SP}_1, \mathcal{C}_0, \phi_0(j), \hat{\phi}_0(j), w_j^0)$, interpolate shares $\phi_0(j)$ to obtain secret s.

Fig. 2. Protocol eAVSS-SC for Asynchronous VSS with Stronger Commitment

From the protocol description, it is apparent that the message complexity is $O(n^2)$. As we use the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme that commits to univariate polynomials using a single element, the communication complexity is $O(\kappa n^2)$. Note that although the size of send messages is $O(\kappa n)$, only n such messages are delivered; thus, the communication complexity does not exceed $O(\kappa n^2)$.

For simplicity of the description, we define our protocol with a *symmetric* bivariate polynomial. It is easily possible to avoid this symmetry requirement in the protocol without any asymptotic increase in the complexity measures.

4.2 Analysis

Theorem 2. *Given a PolyCommit scheme that satisfies Definition 4, eAVSS-SC is an asynchronous VSS protocol that satisfies Definition 5 with the strong commitment property.*

Proof (Proof Outline). We have to prove that protocol eAVSS satisfies the asynchronous VSS properties in Definition 5 along with the strong commitment property. Our analysis is based on the following two claims and the properties of the PolyCommit scheme. We present our proof sketch here, while the complete proof appears in [3].

Claim. If some honest party agrees on ζ, then every honest party will eventually agree on ζ.

Claim. All honest servers complete the Sh phase with the same PolyCommit commitment $\mathcal{C}_0$.

Proof. Assume two honest parties terminate with $\mathcal{C}_0{}'$ and $\mathcal{C}_0{}''$ such that $\mathcal{C}_0{}' \neq \mathcal{C}_0{}''$. From Claim 4.2, we know that all honest parties agree on the same ζ. As ζ commits to $h_{\mathcal{C}}(x)$, an n-degree polynomial interpolated by hashing $n+1$ elements of $\boldsymbol{\mathcal{C}}$, the adversary has to break the evaluation binding property of the polynomial commitment or the collision resistance property of hash function to obtain two different $\mathcal{C}_0$ values that culminate the same ζ. This is not possible in PPT and there is a contradiction. Therefore, we prove that all honest servers complete the Sh phase with the same $\mathcal{C}_0$.

Liveness follows from the protocol flow and correctness of the PolyCommit scheme. Agreement in the Sh phase is evident from claims 4.2 and 3.2, while agreement in reconstruction follows from agreement during the Sh phase. Correctness follows directly from correctness of the PolyCommit scheme and collision-resistance of the hash function. Strong Commitment is apparent from agreement of eAVSS-SC and Claim 3.2. Secrecy follows from the hiding property of PolyCommit.

4.3 Lower Bounds

We observe that the $\Omega(n^2)$ message complexity of our eAVSS and eAVSS-SC protocols as well as the AVSS protocol is optimal.[3] This can be proved in two

[3] With the stronger cryptographic assumptions such as PKI or ZK proofs more efficient schemes can be possible; however, we only assume commitment schemes here.

steps: first, it is known that a VSS protocol is sufficient to implement reliable broadcast [19]; next, extending a result by Dolev and Reischuk [14] for Byzantine agreement to reliable broadcast. The latter proves that if a reliable broadcast protocol terminates, the number of messages exchanged by honest parties is lower bounded by $max\{(n-t),(1+t/2)^2\}$ in presence of a commitment scheme. The above two claims show that the message complexity of asynchronous VSS is lower-bounded by $\Omega(n^2)$ for optimal resiliency condition $n = 3t+1$ and $t > 2$. We thoroughly prove this result in the extended version of this paper [3].

Note that when the shared secret is of size κ (the computational security parameter), the lower bound of $\Omega(n^2)$ message complexity *intuitively* transfers to a lower bound of $\Omega(\kappa n^2)$ on the asynchronous VSS communication complexity. Nevertheless, proving this thoroughly presents an interesting challenge. If proven, it will show that our eAVSS and eAVSS-SC protocols are not only optimal in terms of message complexity but also in terms of communication complexity.

5 Applications

Our eAVSS and eAVSS-SC schemes have direct implications to all asynchronous VSS applications. We briefly discuss some important applications here.

Using our eAVSS-SC protocol in proactive VSS [9] reduces its communication complexity by a linear factor to $O(\kappa n^3)$. The same reduction also applies to distributed key generation required for threshold cryptography, and its group and threshold modification primitives [17]. Using our eAVSS protocol in the asynchronous Byzantine agreement (ABA) framework of Canetti and Rabin [10, 11], it is possible to obtain the first $O(\kappa n^3)$ communication complexity ABA protocol, which is secure against the *adaptive* adversary in the *standard* model without the random oracle assumption (see [9, Sec. 3.5] for details).

Finally, our commitment methodology may also find applications in some other bivariate polynomial-based protocols; however, one has to be careful as it is not a full-fledged bivariate polynomial commitment scheme.

References

1. Abraham, I., Dolev, D., Halpern, J.Y.: An Almost-surely Terminating Polynomial Protocol for Asynchronous Byzantine Agreement with Optimal Resilience. In: Proceedings of ACM PODC 2008, pp. 405–414 (2008)
2. Au, M.H., Susilo, W., Mu, Y.: Practical Anonymous Divisible E-Cash from Bounded Accumulators. In: Tsudik, G. (ed.) FC 2008. LNCS, vol. 5143, pp. 287–301. Springer, Heidelberg (2008)
3. Backes, M., Datta, A., Kate, A.: Asynchronous Computational VSS with Reduced Communication Complexity. Cryptology ePrint Archive, Report 2012/619 (2012), http://eprint.iacr.org/2012/619
4. Backes, M., Kate, A., Patra, A.: Computational Verifiable Secret Sharing Revisited. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 590–609. Springer, Heidelberg (2011)

5. Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous Secure Computation. In: Proceedings of ACM STOC 1993, pp. 52–61 (1993)
6. Blakley, G.R.: Safeguarding Cryptographic Keys. In: Proceedings of the National Computer Conference, pp. 313–317 (1979)
7. Boneh, D., Boyen, X.: Short Signatures Without Random Oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
8. Bracha, G.: An Asynchronous [(n-1)/3]-Resilient Consensus Protocol. In: Proceedings of PODC 1984, pp. 154–162 (1984)
9. Cachin, C., Kursawe, K., Lysyanskaya, A., Strobl, R.: Asynchronous Verifiable Secret Sharing and Proactive Cryptosystems. In: Proceedings of ACM CCS 2002, pp. 88–97 (2002)
10. Canetti, R.: Studies in Secure Multiparty Computation and Applications. Ph.D. thesis, The Weizmann Institute of Science (1996)
11. Canetti, R., Rabin, T.: Fast Asynchronous Byzantine Agreement with Optimal Resilience. In: Proceedings of ACM STOC 1993, pp. 42–51 (1993)
12. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults. In: Proceedings of IEEE FOCS 1985, pp. 383–395 (1985)
13. Cramer, R., Damgård, I., Dziembowski, S.: On the Complexity of Verifiable Secret Sharing and Multiparty Computation. In: Proceedings of STOC 2000, pp. 325–334 (2000)
14. Dolev, D., Reischuk, R.: Bounds on Information Exchange for Byzantine Agreement. Journal of ACM 32(1), 191–204 (1985)
15. Feldman, P.: A Practical Scheme for Non-interactive Verifiable Secret Sharing. In: Proceedings of IEEE FOCS 1987, pp. 427–437 (1987)
16. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive Secret Sharing or: How to Cope with Perpetual Leakage. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995)
17. Kate, A., Goldberg, I.: Distributed Key Generation for the Internet. In: Proceedings of 29th IEEE International Conference on Distributed Computing Systems (ICDCS), pp. 119–128 (2009)
18. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-Size Commitments to Polynomials and Their Applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (2010)
19. Katz, J., Koo, C.-Y., Kumaresan, R.: Improving the Round Complexity of VSS in Point-to-Point Networks. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 499–510. Springer, Heidelberg (2008)
20. Patra, A., Choudhary, A., Rangan, C.P.: Efficient Asynchronous Byzantine Agreement with Optimal Resilience. In: Proceedings of ACM PODC 2009, pp. 92–101 (2009)
21. Pedersen, T.P.: Non-interactive and Information-Theoretic Secure Verifiable Secret Sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992)
22. Pedersen, T.P.: A Threshold Cryptosystem without a Trusted Party. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991)
23. Schultz, D.A., Liskov, B., Liskov, M.: MPSS: Mobile Proactive Secret Sharing. ACM Trans. Inf. Syst. Secur. 13(4), 34 (2010)
24. Shamir, A.: How to Share a Secret. Commun. ACM 22(11), 612–613 (1979)
25. Zhou, L., Schneider, F.B., van Renesse, R.: APSS: Proactive Secret Sharing in Asynchronous Systems. ACM Trans. Inf. Syst. Secur (TISSec) 8(3), 259–286 (2005)

A Protocol $\mathsf{PolyCommit}_{\mathsf{Ped}}$

In this section, we instantiate the $\mathsf{PolyCommit}_{\mathsf{Ped}}$ scheme that commits to a univariate polynomial using a single group element. $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is based on the algebraic property of polynomials $\phi(x) \in \mathbb{F}_p[x]$: $(x - i)$ perfectly divides the polynomial $\phi(x) - \phi(i)$ for $i \in \mathbb{F}_p$. Further, it uses an additional random polynomial $\hat{\phi}(x)$ to achieve unconditional hiding.

Setup$(1^\kappa, t)$ computes two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order p (providing κ-bit security) such that there exists a symmetric bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and for which the t-SDH assumption holds. We denote the generated bilinear pairing group as $\mathcal{G} = \langle e, \mathbb{G}, \mathbb{G}_t \rangle$. Choose two generators $g, h \in_R \mathbb{G}$. Let $\alpha \in_R \mathbb{F}_p^*$ be SK, generated by a (possibly distributed) trusted authority. Setup also generates a $(2t + 2)$-tuple $\langle g, g^\alpha, \ldots, g^{\alpha^t}, h, h^\alpha, \ldots, h^{\alpha^t} \rangle \in \mathbb{G}^{2t+2}$ and outputs $\mathsf{SP} = \langle \mathcal{G}, g, g^\alpha, \ldots, g^{\alpha^t}, h, h^\alpha, \ldots, h^{\alpha^t} \rangle$. Note that SK is not required by the other algorithms of the commitment scheme, and it can be discarded by the authority if t is fixed.

Commit$(\mathsf{SP}, \phi(x))$ chooses $\hat{\phi}(x) \in_R \mathbb{F}_p[x]$ of degree t and computes the commitment $\mathcal{C} = g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)} \in \mathbb{G}$ for the polynomial $\phi(x) \in \mathbb{F}_p[X]$ of degree t or less. For $\phi(x) = \sum_{j=0}^{\deg(\phi)} \phi_j x^j$ and $\hat{\phi}(x) = \sum_{j=0}^{\deg(\hat{\phi})} \hat{\phi}_j x^j$, it outputs $\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j} \prod_{j=0}^{\deg(\hat{\phi})} (h^{\alpha^j})^{\hat{\phi}_j}$ as the commitment to $\phi(x)$.

Open$(\mathsf{SP}, \mathcal{C}, \phi(x), \hat{\phi}(x))$ outputs the committed polynomials $\phi(x)$ and $\hat{\phi}(x)$.

VerifyPoly$(\mathsf{SP}, \mathcal{C}, \phi(x), \hat{\phi}(x))$ verifies that $\mathcal{C} \stackrel{?}{=} g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)}$. If $\mathcal{C} = \prod_{j=0}^{\deg(\phi)} (g^{\alpha^j})^{\phi_j} \prod_{j=0}^{\deg(\hat{\phi})} (h^{\alpha^j})^{\hat{\phi}_j}$ for $\phi(x) = \sum_{j=0}^{\deg(\phi)} \phi_j x^j$ and $\hat{\phi}(x) = \sum_{j=0}^{\deg(\hat{\phi})} \hat{\phi}_j x^j$, the algorithm outputs 1, else it outputs 0. Note that this only works when both $\deg(\phi)$ and $\deg(\hat{\phi}) \leq t$.

CreateWitness$(\mathsf{SP}, \phi(x), \hat{\phi}(x), i)$ computes $\psi_i(x) = \frac{\phi(x) - \phi(i)}{(x - i)}$, $\hat{\psi}_i(x) = \frac{\hat{\phi}(x) - \hat{\phi}(i)}{(x - i)}$, and outputs $\langle i, \phi(i), \hat{\phi}(i), w_i \rangle$. Here, the witness $w_i = g^{\psi_i(\alpha)} h^{\hat{\psi}_i(\alpha)}$.

VerifyEval$(\mathsf{SP}, \mathcal{C}, i, \phi(i), \hat{\phi}(i), w_i)$ verifies that $\phi(i)$ is the evaluation at the index i of the polynomial committed to by $\mathcal{C}$. If $e(\mathcal{C}, g) \stackrel{?}{=} e(w_i, g^\alpha / g^i) e(g^{\phi(i)} h^{\hat{\phi}(i)}, g)$, the algorithm outputs 1, else it outputs 0.

Suppose $h = g^\lambda$ for some unknown λ. Then $\mathsf{VerifyEval}$ is correct because

$$\begin{aligned} e(w_i, g^\alpha / g^i) e(g^{\phi(i)} h^{\hat{\phi}(i)}, g) &= e(g^{\psi_i(\alpha) + \lambda \hat{\psi}_i(\alpha)}, g^{(\alpha - i)}) e(g, g)^{\phi(i) + \lambda \hat{\phi}(i)} \\ &= e(g, g)^{(\psi_i(\alpha)(\alpha - i) + \phi(i)) + \lambda(\hat{\psi}_i(\alpha)(\alpha - i) + \hat{\phi}(i))} \\ &= e(g, g)^{\phi(\alpha) + \lambda \hat{\phi}(\alpha)} = e(g^{\phi(\alpha)} h^{\hat{\phi}(\alpha)}, g) = e(\mathcal{C}, g) \end{aligned}$$

The hiding property of $\mathsf{PolyCommit}_{\mathsf{Ped}}$ is unconditional. The polynomial binding property is based on the DLog assumption, while the evaluation binding property is based on the t-Strong Diffie-Hellman (t-SDH) assumption [7]. The strong correctness property follows from the t-polynomial Diffie-Hellman (t-polyDH) assumption [2, 18].