High-Threshold AVSS with Optimal Communication Complexity

Nicolas AlHaddad[1], Mayank Varia[1], and Haibin Zhang[2]

[1] Boston University, {nhaddad,varia}@bu.edu
[2] Shandong Institute of Blockchain, bchainzhang@aliyun.com

Abstract. Asynchronous verifiable secret sharing (AVSS) protocols protect a secret that is distributed among n parties. Dual-threshold AVSS protocols guarantee consensus in the presence of t Byzantine failures and privacy if fewer than p parties attempt to reconstruct the secret. In this work, we construct a dual-threshold AVSS protocol called Haven that is optimal along several dimensions. First, it is a *high-threshold* AVSS scheme, meaning that it is a dual-threshold AVSS with optimal parameters $t < n/3$ and $p < n - t$. Second, it has $O(n^2)$ message complexity, and for large secrets it achieves the optimal $O(n)$ communication overhead, without the need for a public key infrastructure or trusted setup. While these properties have been achieved individually before, to our knowledge this is the first protocol that achieves all of the above simultaneously. The core component of Haven is a high-threshold AVSS scheme for small secrets based on polynomial commitments that achieves $O(n^2 \log(n))$ communication overhead, as compared to prior schemes that require $O(n^3)$ overhead with $t < n/4$ Byzantine failures or $O(n^4)$ overhead for the recent high-threshold protocol of Kokoris-Kogias et al (CCS 2020). Using standard amortization methods based on erasure coding, we can reduce the communication complexity to $O(n|s|)$ for a large secret s.

1 Introduction

Broadcast protocols are a core component in the design of fault-tolerant systems; for example, they enable replica servers to coordinate their actions in state machine replication, and they contribute toward the finality of cryptocurrencies. Reliable broadcast protocols between n servers ensure both that a message is delivered to all servers and that the delivered messages are identical. While there exist many broadcast protocols that assume strict or partial synchrony (i.e., an upper bound on message delivery times), *asynchronous* reliable broadcast protocols do not rely on any timing assumptions and are inherent more robust against denial-of-service and performance attacks. Bracha's asynchronous reliable broadcast protocol has $O(n^2)$ total message complexity and achieves reliability for up to $t < n/3$ Byzantine failures [11], which is optimal for protocols without setup that provide correctness, liveness, and agreement [24].

Asynchronous verifiable secret sharing (AVSS) protocols [19] introduce a fourth guarantee: privacy of the message against any coalition of up to p servers

comprising the t Byzantine servers plus $p - t$ honest servers that unintentionally cooperate with the adversary. Combining asynchronous broadcast with a Shamir secret sharing scheme [36] with threshold p, AVSS protocols proceed in two phases: a sharing phase in which the initial holder or *dealer* of a secret message s distributes secret shares of s to all servers, and a reconstruction phase in which any collection of $p + 1$ servers can recover s. This is the asynchronous version of verifiable secret sharing [22] because correct reconstruction is required even against a malicious dealer. While many AVSS protocols consider $p = t$, a subset called *dual-threshold AVSS* protocols consider $p > t$.

In this paper, we explore *high-threshold asynchronous verifiable secret sharing* (HAVSS) protocols, which are a special case of dual-threshold AVSS that can achieve any possible consensus threshold $t < n/3$ and privacy threshold $p < n-t$. These match the known upper bounds for consensus [34] and privacy (the honest servers must be able to reconstruct even if the Byzantine servers refuse to do so). HAVSS enables the generation of an asynchronous fair coin tossing protocol that can be used to remove the trusted dealer assumption needed in many distributed computations, such as efficient asynchronous Byzantine agreement, distributed key generation, threshold signatures, and threshold encryption [14–16, 28].

Our contributions. In this work, we contribute an HAVSS protocol called Haven that is optimal along several dimensions:

- Haven achieves any consensus threshold of $t < n/3$ and privacy threshold of $p < n - t$.
- Haven has $O(n^2)$ message complexity during sharing and reconstruction. Concretely, every server sends 2 messages to each party during sharing (3 for the dealer), and 1 message to each party during reconstruction.
- For a short secret s sampled randomly from a finite field, its communication overhead (i.e., number of field elements sent) is $O(n^2 \log n)$ without trusted setup. If trusted setup is permissible, this can be reduced to $O(n^2)$ in some cases.
- For a long secret s, its communication complexity is $O(n|s|)$.
- Haven does not require trusted setup or a public key infrastructure (PKI).

All of these parameters improve upon the recent breakthrough by Kokoris-Kogias et al. [28], the first HAVSS protocol with optimal resilience. Our communication complexity even beats many existing AVSS schemes that were not striving for dual-threshold. Table 1 shows a comparison of our work to several related protocols, which we describe in more detail below.

Why is HAVSS possible? Suppose there are $n = 3t + 1$ parties, where the dealer is one of the t Byzantine servers, and the honest servers are split into two camps:

- $t+1$ *informed* servers that always receive valid messages from the Byzantine servers (i.e., what honest servers would have sent), and
- t *clueless* servers that never receive any messages from Byzantine servers.

Works	threshold		complexity				avoiding setup		crypto
	dual	high	message	comm.	amortized	rounds	no trust?	no PKI?	assumption
Cachin et al. [15]	✓	✗	$O(n^2)$	$O(\kappa n^3)$	$O(\kappa n^2)$	3	✓	✓	DL
Backes et al. [2]	✗	✗	$O(n^2)$	$O(\kappa n^2)$	$O(\kappa n^2)$	3	✗	✓	t-SDH
Kate et al. [25]	✗	✗	$O(n^2)$	$O(\kappa n^3)$	$O(\kappa n)$	> 4	✗	✗	t-SDH
Kokoris-Kogias et al. [28]	✓	✓	$O(n^2)$	$O(\kappa n^4)$	$O(\kappa n^3)$	4	✓	✗	DL
HAVEN option 1	✓	✓	$O(n^2)$	$O(\kappa n^2)$	$O(\kappa n)$	3	✗	✓	t-SDH
HAVEN option 2	✓	✓	$O(n^2)$	$\tilde{O}(\kappa n^2)$	$O(\kappa n)$	3	✓	✓	DL + ROM

Table 1. Comparison of our HAVEN protocol with several prior AVSS protocols. Note that HAVEN's communication complexity, computational assumption, and reliance on trusted setup depend on the polynomial commitment scheme used; the contents of this table are predicated on the use of Bulletproofs [13] (cf. Sec. 3.3).

One might wonder: is it even possible to achieve $p = 2t$ privacy? Intuitively, it seems that we run into a paradox. The informed servers can complete the sharing phase because they receive valid messages from $2t + 1$ servers (i.e., from their perspective, the clueless servers appear Byzantine). Ergo, the $2t+1$ informed and clueless servers must collectively be able to recover the secret, even if the Byzantine servers refuse to participate in reconstruction. However, the clueless servers cannot contribute anything meaningful toward the reconstruction because they have only received messages from the informed servers. Hence, if the $t + 1 < p$ informed servers can learn the secret with the clueless servers, then they must have been able to learn the secret without them, breaking privacy.

Fortunately, there is one flaw in the above argument that enables HAVSS (and which schemes like HAVEN must exploit): even if the $t+1$ informed servers collectively possess enough data to learn the secret, they might not actually transmit this data during reconstruction. Ergo, they might still rely on the clueless servers during the actual reconstruction protocol, even if the clueless servers are relying information that the informed servers collectively know.

Overview of the construction. The core of HAVEN is a construction for small, randomly chosen secrets. Like other (dual-threshold) AVSS schemes, it broadly follows a "two-layer secret sharing" approach. The dealer begins by constructing a degree p polynomial R that is a Shamir secret sharing with her secret s encoded at location $R(0)$. We call R the *recovery polynomial* because the reconstruction phase consists of each party P_i revealing $R(i)$ so that everyone can interpolate R and learn the secret. Next for $i \in [1, n]$, the dealer constructs the degree t polynomial S_i that is a Shamir secret sharing with the secret encoded at location $R(i) = S_i$. This creates a diagonal pattern as shown on Fig. 1.

The sharing phase of HAVEN follows a 3-message (send, echo, ready) format, just like Bracha's reliable broadcast. There are two goals that are achieved concurrently:

– Each party P_i must learn S_i, which we call its *share polynomial*. To do this privately, the dealer sends each party one share on everyone else's share

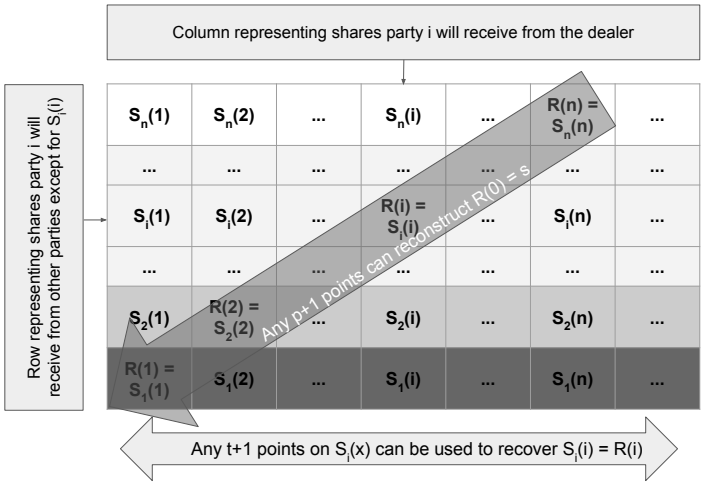

Fig. 1. Overview of the data transmitted during Haven. Each party P_i receives column i of this matrix from the dealer; note that this includes $S_i = R(i)$. If P_i never received $R(i)$, then $t + 1$ other parties will send their points on row i, from which P_i can interpolate the polynomial S_i and learn the value $S_i(i) = R(i)$. At the reconstruction stage, any $p + 1$ points on the polynomial (1 per party) suffice to interpolate R and learn the secret $s = R(0)$.

polynomial, which they can disperse in the echo stage. Polynomial commitments (and their succinct proofs/arguments) enable each party P_i to verify the integrity of everyone else's claimed point on her share polynomial.
– The parties must collectively reach consensus about which share polynomials to use (against a Byzantine dealer). To do this, the dealer produces a vector commitment of the $n+1$ polynomial commitments and uses Bracha's reliable broadcast to disseminate this value (which is not sensitive).

Because we provide each server with 1 share on everyone else's S_i, observe that $t + 1$ servers collectively hold enough data to reconstruct all polynomials and recover the secret (as stated above). This is acceptable because HAVSS only aims to protect against t Byzantine adversaries.

Unlike previous (dual-threshold) AVSS constructions, Haven does not encompass R and S_i into a (larger) bivariate polynomial. Instead, we directly check for consistency of these univariate polynomials by designing a polynomial commitment scheme that is:

– Homomorphic, in order to construct a succinct proof that the polynomials intersect at the correct point (i.e., that $R(i) - S_i(i) = 0$).
– Degree-revealing, meaning that its proofs demonstrate an upper bound on the degree of the committed polynomial (cf. Def. 4) so that parties can consistently reconstruct it .

– Deterministic, so that once party P_i learns S_i, she can now prove the value at new points. This property allows P_i to prove that she is sending the correct value for $S_i(i) = (R(i))$ during reconstruction.

The communication complexity of Haven is dominated by the transmission of a polynomial commitment between each pair of parties. These commitments can be constant-sized with trusted setup, or logarithmic in size otherwise, which leads to the two options shown in Table 1.

Finally, we extend Haven into an HAVSS scheme for large, arbitrarily-chosen secrets with linear communication overhead. The idea is simple: choose a short, ephemeral secret key k, and engage in (1) the above HAVSS protocol to share k and (2) an asynchronous verifiable information dispersal protocol [17] to disseminate the ciphertext $\text{Enc}_k(s)$ without concern for privacy. The linear overhead to disseminate the ciphertext dominates the cost of HAVSS for secrets of length $|s| = \Omega(n \log n)$ (this can be reduced to $|s| = \Omega(n)$ with trusted setup).

Related work. Several AVSS protocols were proposed in the 1990s with unconditional security [5,18,19], but at the expense of huge communication complexity. The first practical AVSS was achieved by Cachin et al. [15] using computational assumptions (namely, the discrete logarithm assumption). Their protocol achieves an optimal message complexity of $O(n^2)$ and resilience of $n > 3t$, but their $O(\kappa n^3)$ communication complexity is suboptimal. Cachin et al. [15] also constructed the first dual-threshold AVSS with consensus $t < n/4$ and privacy $p < n/2$, with the same message and communication complexity as above.

Recently, Kokoris-Kogias et al. [28] constructed the first HAVSS protocol. As described above, breaking the privacy barrier from $p < n/2$ to $p < 2n/3$ is a challenging accomplishment. Nevertheless, the improved privacy comes at a price of $O(\kappa n^4)$ communication complexity with 4 rounds of communication, plus the need for a public key infrastructure (PKI) so that any server can pass along digitally-signed messages from other senders to their intended destination. Kokoris-Kogias et al. use their HAVSS in a black-box manner to construct randomness beacons, distributed key generation, threshold signatures, and more; using Haven reduces the communication of those constructions too.

Several other works focused on providing linear (amortized) communication overhead of AVSS for large secrets, including Cachin and Tessaro [17] and more recently Basu et al. [4] and Kate et al. [25]. The latter two works use polynomial commitments [26] that require a trusted setup to achieve an optimal communication complexity of $O(\kappa n)$, but for short messages their communication complexity is $O(\kappa n^3)$ in the worst case.

Our construction makes extensive use of polynomial commitments, which were introduced by Kate et al. [26] and subsequently used by Backes et al. [2] to design an AVSS protocol (with a single threshold). We also leverage recent works that construct polynomial commitments without trusted setup; we use Bulletproofs [13] in our construction, but our techniques are amenable to other polynomial commitment schemes (e.g., [9,10,12]).

Finally, we focus exclusively on worst-case metrics for message and (amortized) communication complexity in this paper. There exist several works that contain a "fast path" and "slow path" approach whose typical communication complexity is comparable to Haven and can provide benefits like lower computational cost or higher thresholds for some security properties, even if the worst-case metrics are identical or worse (e.g., [1, 3, 24, 33]).

Organization. The rest of this paper is organized as follows. In Section 2, we define (high-threshold) asynchronous verifiable secret sharing as well as many of the building blocks that we need in this work. In Section 3, we construct Haven for "short" secrets that are approximately equal in length to the security parameter. Finally, in Section 4, we amortize Haven to achieve lower communication complexity for secrets that are substantially larger than the security parameter.

2 Definitions

In this section, we provide definitions for polynomial and vector commitments that we will use within Haven as well as a definition for asynchronous verifiable secret sharing. In the definitions below, κ denotes the security parameter, "negligible" refers to a function vanishing faster than any inverse polynomial, "overwhelming" refers to $1 - \epsilon$ for a negligible function ϵ, and PPT is an abbreviation for probabilistic polynomial time.

2.1 Commitment schemes

In this work, we consider non-interactive commitment schemes for polynomials and vectors. We begin by defining a polynomial commitment scheme [26]. In this work, we exclusively consider schemes that are homomorphic, and our definition is similar to the notion of "linear combination schemes" from Boneh et al. [8] except that we restrict our attention to *deterministic* schemes.

Definition 1. *A* polynomial commitment scheme $\mathcal{P}$ *comprises four algorithms* Setup, Com, Eval, Verify *and an optional fifth algorithm* Hom *that act as follows:*

- $\text{Setup}(1^\kappa, \mathbb{F}, D) \to \mathsf{pp}$ *is given a security parameter* κ*, a finite field* $\mathbb{F}$*, and an upper bound* D *on the degree of any polynomial to be committed. It generates public parameters* pp *that are required for all subsequent operations.*
- $\text{Com}(\mathsf{pp}, \phi(x), d) \to \hat{\phi}$ *is given a polynomial* $\phi(x) \in \mathbb{F}[x]$ *of degree* $d \leq D$*. It outputs a commitment string* $\hat{\phi}$ *(throughout this work, we use the hat notation to denote a commitment to a polynomial).*
- $\text{Eval}(\mathsf{pp}, \phi, i) \to \langle i, \phi(i), w \rangle$ *is given a polynomial* ϕ *as well as an index* $i \in \mathbb{F}$*. It outputs a 3-tuple containing* i*, the evaluation* $\phi(i)$*, and witness string* w_i*.*
- $\text{Verify}(\mathsf{pp}, \hat{\phi}, y, d) \to$ *True/False takes as input a commitment* $\hat{\phi}$*, a 3-tuple* $y = \langle i, j, w \rangle$*, and a degree* d*. It outputs a Boolean.*

– $\text{Hom}(\mathsf{pp}, \hat{\phi}_1, \hat{\phi}_2, a) \to \widehat{\phi_1 + a\phi_2}$ *takes in commitments to two polynomials* ϕ_1 *and* ϕ_2 *of degree at most* D*, as well as a field element* $a \in \mathbb{F}$*. Outputs the commitment* $\text{Com}(\mathsf{pp}, \phi, \max\{d_1, d_2\})$ *to the polynomial* $\phi = \phi_1 + a\phi_2$*.*

Informally, the Verify method of a polynomial commitment scheme should return True if and only if $\phi(i) = j$, w is a witness previously created by Eval, and the degree of ϕ is at most d. We formalize this guarantee in Definitions 2-5 below.

Definition 1 is mostly similar to prior works that have defined and constructed polynomial commitments [9,10,13,25,26]. There are three differences. First, the verifier learns an upper bound on the degree of the polynomial. Second, we restrict our attention to deterministic Com algorithms, and consequently the witness generation process in Eval is well-specified purely from the polynomial (i.e., without requiring the randomness string used earlier in the Com stage). These changes have consequences for our security definitions (see Definitions 4 and 5) and constructions of polynomial commitments (see Section 3.3). Third, we don't include a method to open the entire polynomial ϕ; this is without loss of generality since revealing ϕ is equivalent to revealing enough evaluations to interpolate ϕ.

There are two predominant styles of security definitions for polynomial commitments: game-based definitions that resemble the binding and hiding properties of traditional commitments [25, 26], or simulation-based definitions in the vein of zero knowledge proofs of knowledge [9, 10, 13].

The weaker indistinguishability style suffices for this work, and we use it in the definitions below. Definitions 2, 3, and 5 are nearly identical to their counterparts in [25,26], whereas Definition 4 is a new security guarantee that we require in this work (cf. Section 3.3 for constructions). All of these definitions apply equally whether or not the commitment scheme is homomorphic.

Definition 2 (Strong correctness). *Let* $\mathsf{pp} \leftarrow \text{Setup}(1^\kappa, \mathbb{F}, D)$*. For any polynomial* $\phi(x) \in \mathbb{F}[x]$ *of degree* d *with associated commitment* $\hat{\phi} = \text{Com}(\mathsf{pp}, \phi, d)$*:*

– *If* $d \leq D$*, then for any* $i \in \mathbb{F}$ *the output* $y \leftarrow \text{Eval}(\mathsf{pp}, \hat{\phi}, i)$ *of evaluation is successfully verified by* $\text{Verify}(\mathsf{pp}, \hat{\phi}, y, d)$*.*
– *If* $d > D$*, then no adversary can succeed with non-negligible probability at creating a commitment* $\tilde{\phi}$ *that is successfully verified at* $d+1$ *randomly chosen indices.*

Definition 3 (Evaluation binding). *Let* $\mathsf{pp} \leftarrow \text{Setup}(1^\kappa, \mathbb{F}, D)$*. For any PPT adversary* $\mathcal{A}(\mathsf{pp})$ *that outputs a commitment* $\tilde{\phi}$*, a degree* d*, and two evaluations* $y = \langle i, j, w\rangle$ *and* $y' = \langle i', j', w'\rangle$*, there exists a negligible function* $\varepsilon(\kappa)$ *such that:*

$$\Pr[(\tilde{\phi}, y, y', d) \leftarrow \mathcal{A}(\mathsf{pp}) : i = i' \wedge j \neq j' \wedge \text{Verify}(\mathsf{pp}, \tilde{\phi}, \tilde{y}, d) \wedge \text{Verify}(\mathsf{pp}, \tilde{\phi}, \tilde{y}', d)] < \varepsilon(\kappa).$$

Definition 4 (Degree binding). *Let* $\mathsf{pp} \leftarrow \text{Setup}(1^\kappa, \mathbb{F}, D)$*. For any PPT adversary* $\mathcal{A}$ *that outputs a polynomial* ϕ *of degree* $\deg(\phi)$*, evaluation* $\tilde{y}$*, and integer* d*, there exists a negligible function* $\varepsilon(\kappa)$ *such that:*

$$\Pr[(\phi, \tilde{y}, d) \leftarrow \mathcal{A}(\mathsf{pp}), \hat{\phi} = \text{Com}(\mathsf{pp}, \phi, \deg(\phi)) : \text{Verify}(\mathsf{pp}, \hat{\phi}, \tilde{y}, d) \wedge \deg(\phi) > d] < \varepsilon(\kappa).$$

Definition 5 (Hiding for random polynomials). *Let* $\mathsf{pp} \leftarrow \text{Setup}(1^\kappa, \mathbb{F}, D)$*, d be an arbitrary integer less than D, and $I \subset \mathbb{F}$ be an arbitrary set of indices with $|I| \leq d$. Randomly choose a $\phi \leftarrow \mathbb{F}[x]$ of degree d and construct its commitment $\hat{\phi} = \text{Com}(\mathsf{pp}, \phi, d)$. For all PPT adversaries $\mathcal{A}$, there exists a negligible polynomial $\varepsilon(\kappa)$ such that:*

$$\Pr[(x, y) \leftarrow \mathcal{A}(\mathsf{pp}, \hat{\phi}, \{\text{Eval}(\mathsf{pp}, \phi, i)\}_{i \in I}) : y = \phi(x) \wedge x \notin I] < \varepsilon(\kappa),$$

where the probability is taken over $\mathcal{A}$'s coins and the random choice of ϕ.

In words, the hiding definition states that even given evaluations at the indices in I, no adversary can find a new point on ϕ with non-negligible probability. Note that the hiding property is only achievable for randomly-chosen ϕ because we defined Eval deterministically.

Finally, we provide the syntax for (static) vector commitments, which are succinct encodings of finite, ordered lists in such a way that one can later open a value at a specific location [20,31]. (We use 0-indexing throughout this work.)

Definition 6. *A static* vector commitment scheme $\mathcal{V} = (\text{vSetup}, \text{vCom}, \text{vGen}, \text{vVerify})$ *comprises four algorithms that operate as follows:*

- $\text{vSetup}(1^\kappa, U, L) \to \bar{\mathsf{pp}}$ *is given a security parameter κ, a set U, and a maximum vector length L. It generates public parameters* $\bar{\mathsf{pp}}$.
- $\text{vCom}(\bar{\mathsf{pp}}, \boldsymbol{v}) \to C$ *is given a vector $\boldsymbol{v} \in U^\ell$ where $\ell \leq L$. It outputs a commitment string C.*
- $\text{vGen}(\bar{\mathsf{pp}}, \boldsymbol{v}, i) \to w_i$ *is given a vector $\boldsymbol{v}$ and an index i. It outputs a witness string w_i.*
- $\text{vVerify}(\bar{\mathsf{pp}}, C, u, i, w) \to$ *True/False takes as input a vector commitment C, an element $u \in U$, an index i, and a witness string w_e. It outputs a Boolean value that should only equal True if $u = \boldsymbol{v}[i]$ and w is a witness to this fact.*

This is a special case of a polynomial commitment scheme, and indeed throughout this work we assume that vector commitments are instantiated using our polynomial commitments (see Section 3.3) although other instantiations are possible like Merkle trees [32]. Vector commitments also have analogous binding and hiding security guarantees to Definitions 2-5 [20,31]. Without loss of generality, we can consider $U = \{0,1\}^*$ by hashing strings before running the vector commitment algorithms.

2.2 Dual-threshold asynchronous verifiable secret sharing

In this section, we define dual-threshold asynchronous verifiable secret sharing (DAVSS) protocols that are the focus of this work. Our definition is consistent with the works of Cachin et al. [15] and Kokoris-Kogias et al. [28]. Recall that, informally, an AVSS scheme is an interactive protocol between n servers that allows one server (the "dealer") to split a secret among all servers in such a way that they obtain consensus over the shared secret while also protecting the

privacy of the secret until reconstruction time, even though some of the servers are adversarial. The term "dual-threshold" means that the number of parties required for reconstruction of the secret may be different than the number of Byzantine failures that can be withstood.

Definition 7. *A* (n, p, t) dual-threshold asynchronous verifiable secret sharing *(DAVSS) protocol involves* n *servers interacting in two stages.*

- Sharing stage. *This stage begins when a special party, called the "dealer"* P_d*, is activated on an input message of the form* $(ID.d, \mathsf{in}, \mathsf{share}, s)$*. Here, the value ID.d is a tag identifying the session, and* s *is the dealer's secret.* P_d *begins the protocol to* share s using $\mathrm{ID}.d$*. A server* P_i *has* completed the sharing *for ID.d when it generates a local output of the form* $(ID.d, \mathsf{out}, \mathsf{shared})$*.*
- Reconstruction stage. *After server* P_i *has completed the sharing stage, it may* start reconstruction *for ID.d when activated on a message* $(ID.d, \mathsf{in}, \mathsf{reconstruct})$*. Eventually, the server halts with output* $(ID.d, \mathsf{out}, \mathsf{reconstructed}, z_i)$*. In this case, we say that* P_i reconstructs z_i *for ID.d.*

An (n, p, t)*-DAVSS satisfies the following security guarantees in the presence of an adversary* $\mathcal{A}$ *who can adaptively and maliciously corrupt up to* t *servers.*

- Liveness. *If* $\mathcal{A}$ *initializes all honest servers on a sharing ID.d, delivers all associated messages, and the dealer* P_d *is honest throughout the sharing stage, then with overwhelming probability all honest servers complete the sharing.*
- Privacy. *If an honest dealer shared* s *using ID.d and at most* $p - t$ *honest servers started reconstruction for ID.d, then* $\mathcal{A}$ *has no information about* s*.*
- Agreement. *Provided that* $\mathcal{A}$ *initializes all honest servers on a sharing ID.d and delivers all associated messages: (1) if some honest server completes the sharing for ID.d, then all honest servers complete the sharing for ID.d, and (2) if all honest servers start reconstruction for ID.d, then with overwhelming probability every honest server* P_i *reconstructs some* s_i *for ID.d.*
- Correctness. *Once* $p + 1$ *honest servers have completed the sharing for ID.d, there exists a fixed value* $z \in \mathbb{F}$ *such that the following holds with overwhelming probability: (1) if the dealer shared* s *using ID.d and is honest throughout the sharing stage, then* $z = s$ *and (2) if an honest server* P_i *reconstructs* z_i *for ID.d, then* $z_i = z$*.*

A high-threshold asynchronous verifiable secret sharing *(HAVSS) protocol is a* (n, p, t)*-DAVSS that supports any choice of* $t < n/3$ *and* $p < n - t$*.*

3 Haven for Short, Uniformly Random Secrets

In this section, we show how to use polynomial commitments to construct an HAVSS protocol called Haven for a short secret s that is uniformly sampled from a finite field $\mathbb{F}$. Then, we demonstrate that our construction achieves all the security properties for an HAVSS. Finally, we show how to modify two existing polynomial commitment schemes so that they meet our Definitions 2-5.

3.1 Construction

Like all AVSS protocols, Haven proceeds in two phases: a sharing phase in which the dealer distributes shares of her secret s, and a reconstruction phase in which the servers collectively reconstruct the secret. We formally present the two phases of Haven in Algorithms 1 and 2, respectively. Here, we present the concepts behind the construction.

Sharing phase. The sharing phase of Haven follows the same communication pattern as Bracha's asynchronous reliable broadcast [11] and the AVSS protocol of Cachin et al. [15]. First, the dealer transmits a $\tilde{O}(n)$-size send message to all parties. Then, everyone sends a $\tilde{O}(1)$-size echo and ready messages to all parties.

Lines 1-14 show the dealer's initial work, culminating in the send message. This is the most complex part of the protocol, and we describe it in detail.

- In lines 2-4, the dealer samples a degree-p *recovery polynomial* R and along with n different degree-t *share polynomials* $S_1, \ldots, S_n$, all in $\mathbb{F}[x]$. They satisfy $R(0) \triangleq s$ and $S_i(i) \triangleq R(i)$, but are otherwise uniformly sampled. Figure 1 pictorially shows the relationships between these polynomials; we stress that they need not be consistent with any low-degree bivariate polynomial.
- In lines 5-6, the dealer computes polynomial commitments of R and all S_i.
- Recall that an evaluation contains one (x, y) coordinate as well as a proof that this coordinate is on the committed polynomial. In line 8, we form a vector $\boldsymbol{y}_i^S$ containing n evaluations, but in a transposed order: this vector contains the evaluation of one point on each share polynomial $S_1, \ldots, S_n$.
- In lines 9-10, we construct the n *test polynomials* $T_i \triangleq R - S_i$ along with evaluations proving that $T_i(i) = 0$ for all i. This proves consistency between the share and recovery polynomials, as shown in Figure 1. (Interestingly, even though we have committed to R, we never evaluate it directly.)
- In lines 11-12, we build the *root commitment* C; this is a vector commitment to all of the polynomial commitments. Looking ahead, we will run Bracha's reliable broadcast protocol on C, and servers will only believe a polynomial (commitment) if it can be linked back to C. Abusing notation, we assume each polynomial commitment contains the witness to its own inclusion in C (this witness is ignored when running a polynomial Verify check).
- Finally, in lines 13-14, the dealer sends to party P_i the root commitment, all $n + 1$ polynomial commitments, one evaluation on everybody's share polynomial, and all n evaluations of the test polynomial.

When a party P_i receives the send message from the dealer, it performs several checks to ensure that the message is internally consistent (lines 16-18):

- All polynomial evaluations received are verifiably part of S_j and T_j.
- The degrees of the S_j and T_j polynomials are at most t and p, respectively.
- The recovery and share polynomials are equal at $R(i) = S_i(i)$.
- All polynomial commitments link back to the root commitment.

Algorithm 1 Sharing phase of HAVEN, for server P_i and tag ID.d

UPON RECEIVING (ID.d, in, share, s): ▷ only if party is the dealer P_d
  randomly choose recovery polynomial $R \in \mathbb{F}[x]$ of degree p s.t. $R(0) = s$
  for $i \in [1, n]$ **do**
    randomly choose share polynomial $S_i \in \mathbb{F}[x]$ of degree t s.t. $S_i(i) = R(i)$
  compute $\hat{R} = \mathrm{Com}(\mathsf{pp}, R, p)$ ▷ make polynomial commitments
  compute each $\hat{S}_i = \mathrm{Com}(\mathsf{pp}, S_i, t)$ and let $\hat{\boldsymbol{S}} = \langle \hat{S}_1, \hat{S}_2, \ldots, \hat{S}_n \rangle$
  for $i \in [1, n]$ **do** ▷ evaluate and create witnesses
    compute $\boldsymbol{y}_i^S = [\mathrm{Eval}(\mathsf{pp}, S_j, i) \text{ for } j \in [1, n]]$ ▷ one point on each S_j
    compute $\hat{T}_i = \mathrm{Hom}(\hat{R}, \hat{S}_i, -1)$ ▷ $T_i(x) = R(x) - S_i(x)$
  compute $\boldsymbol{y}^T = [\mathrm{Eval}(\mathsf{pp}, T_i, i) \text{ for } i \in [1, n]]$ ▷ tests if all $S_i(i) = R(i)$
  compute $C = \mathrm{vCom}(\bar{\mathsf{pp}}, \langle \hat{R}, \hat{S}_1, \hat{S}_2, \ldots, \hat{S}_n \rangle)$ ▷ root commitment
  append to $\hat{R}$ and each $\hat{S}_i$ a witness of inclusion in C at the right location
  for $i \in [1, n]$ **do**
    send "ID.d, send, set_i" to party P_i, where $\mathsf{set}_i = \{C, \hat{R}, \hat{\boldsymbol{S}}, \boldsymbol{y}_i^S, \boldsymbol{y}^T\}$

UPON RECEIVING (ID.d, send, set_j) from P_d for the first time: ▷ echo stage
  if all $\mathrm{Verify}(\mathsf{pp}, \hat{S}_j, \boldsymbol{y}_i^S[j], t)$ and $\mathrm{Verify}(\mathsf{pp}, \hat{T}_j, \boldsymbol{y}^T[j], p)$ are true **then**
    if $\hat{R}$ and all $\hat{\boldsymbol{S}}$ are in C at the expected locations **then**
      if $T_j(j) = 0$ for all $j \in [1, n]$ **then** ▷ dealer's message is consistent
        for $j \in [1, n]$ **do** ▷ send message to each party P_j
          send "ID.d, echo, $\mathsf{info}_{i,j}$" to P_j, where $\mathsf{info}_{i,j} = \{C, \hat{S}_j, \boldsymbol{y}_i^S[j]\}$

UPON RECEIVING (ID.d, echo, $\mathsf{info}_{j,i}$) from P_m for the first time: ▷ ready stage
  if $\hat{S}_m$ is in C at location m and $\mathrm{Verify}(\mathsf{pp}, \hat{S}_m, \boldsymbol{y}_i^S[m], t) = \mathrm{True}$ **then**
    if not yet sent ready and received $2t + 1$ valid echo with this C **then**
      send "ID.d, ready, C" to all parties ▷ Bracha consensus on C

UPON RECEIVING (ID.d, ready, C) from P_m for the first time:
  if not yet sent ready and received $t + 1$ ready with this C **then**
    send "ID.d, ready, C" to all parties ▷ Bracha consensus on C
  if received $2t + 1$ ready with this C **then**
    wait to receive $t + 1$ valid echo with this C ▷ must happen eventually
    interpolate S_i from the $t + 1$ valid $\boldsymbol{y}_m^S[i]$ in the received echo
    compute $y_i^* = \mathrm{Eval}(\mathsf{pp}, S_i, i)$ ▷ evaluation of $S_i(i)$
    output (ID.d, out, shared) ▷ locally halt

Algorithm 2 Reconstruction phase of HAVEN, for server P_i and tag ID.d

1: UPON RECEIVING (ID.d, in, reconstruct):
2: **for** j in $[1, n]$ **do**
3: send (ID.d, reconstruct-share, $\hat{S}_j$, y_j^*) ▷ to Party P_j

4: UPON RECEIVING (ID.d, reconstruct-share, $\hat{S}_m, y_m^*$): ▷ from Party P_m
5: **if** $\hat{S}_m$ in C and Verify(pp, $\hat{S}_m, y_m^*, t$) **then**
6: **if** received $p+1$ valid reconstruct-share messages **then**
7: interpolate R from the $p+1$ valid points ▷ assume that $R(j) = S_j(j)$
8: output (ID.d, out, reconstructed, $R(0)$)

If all checks pass: party P_i sends an echo message to each party P_j containing what it believes to be the root commitment C (as part of Bracha's broadcast) along with two pieces of information about party j's share polynomial: its commitment $\hat{S}_j$ and the evaluation at one point $S_j(i)$ (to help P_j interpolate the polynomial). We stress that when party P_i sends an echo message, she may not yet be able to tell whether her polynomial (commitments) will become the consensus ones, because the Bracha broadcast protocol on C might not be complete.

When party P_i receives an echo message from another party, she will disregard the message if either the received polynomial commitment or evaluation cannot link back to the received root commitment (all of which may be different from her local state from the earlier send message). The remainder of the protocol proceeds as in Bracha's broadcast.

Reconstruction phase. If a party P_i completes the sharing and starts the reconstruction stage, then this party knows the share polynomial S_i and a witness that it links back to the broadcast root commitment. She sends all parties an evaluation of $S_i(i)$ along with the witness linking S_i to the root commitment; observe that P_i can construct this evaluation because the commitment is deterministic. Everyone verifies this evaluation and interprets this point as $R(i)$ instead; this is acceptable because at least $t+1$ honest parties have verified that $R(j) = S_j(j)$ for all j during the sharing phase (line 18). Given $p+1$ valid messages from other parties, P_i can interpolate R and recover the secret s.

3.2 Analysis

In this section, we prove that our HAVEN protocol is a high-threshold AVSS.

Theorem 1. *Assuming that the underlying polynomial and vector commitment schemes satisfy Definitions 2-6, then* HAVEN *protocol is a high threshold AVSS with $O(n^2)$ message complexity and $O(\kappa n^2 c)$ communication complexity, where κ is the security parameter and c is the size of the underlying commitments and evaluations.*

Below, we provide proofs for each of the four security properties in Definition 7 and the efficiency claim.

Proof (Liveness). If the dealer P_d is honest and all messages are delivered, then P_d will send everyone the same root commitment, polynomial commitments. Also, each party receives 1 point on every share polynomial. All of the checks on lines 16-18 pass, so the honest parties can echo these points, from which everyone will be able to send ready messages and interpolate their own share polynomial. If any dishonest party tries to send a malformed commitment or evaluation in their echo message, then evaluation binding (Def. 3) ensures that it will not link back to the root commitment, so honest parties will disregard this message.

Proof (Privacy). We focus first on the reconstruction stage, and we assume without loss of generality that the adversary $\mathcal{A}$ knows her own share polynomials. As a result, she knows t points on the recovery polynomial R and will receive $p-t$ additional points from honest parties. From this information alone, Shamir secret sharing guarantees that $\mathcal{A}$ learns nothing about the secret unless she can distinguish at least 1 more point on R from random.

Next, we consider the information available to the adversary $\mathcal{A}$ during the sharing stage. The dealer's send messages give $\mathcal{A}$ a total of t evaluations on each share polynomial S_i, but the hiding property (Def. 5) guarantees that this is insufficient to distinguish any other point on S_i from random with non-negligible probability. The subsequent echo and ready messages are of no help because they only contain information about $\mathcal{A}$'s share polynomials, not those of other parties.

Proof (Agreement). Suppose that an honest party P_i has completed the sharing for ID.d. We must show that another (arbitrary) honest P_j will also complete the sharing.

- Since P_i completed the sharing, she heard $2t+1$ ready messages with the same root commitment C^* and have confirmed that the dealer correctly split the large secret s into proper shares and fingerprinted them correctly in the root commitment C^*. At least $t+1$ of those senders are honest and will send ready messages to everyone. Due to line 27, this will cause all honest parties to send ready messages if they have not yet done so. Ergo, party P_j will eventually hear $2t+1$ ready messages with root commitment C^*, thereby satsifying the conditional on line 28.
- Since P_i completed the sharing, she must have sent a ready message in line 24 or line 27 (this is an xor since honest parties only send one ready message).
- The condition for line 27 cannot be satisfied until $t+1$ parties sent a ready due to line 24, at least one of whom must be honest (say, party P_m). For this to occur, party P_m must have observed $2t+1$ echo messages that are internally consistent and link to the same root commitment C^*.
- At least $t+1$ of those echo message senders are honest, so they will also send consistent echo messages to party P_j. Once this happens, P_j can complete the wait step on line 29. Also, the echo messages contain enough information for P_j to compute lines 30-31 and complete the sharing.

Next, suppose all honest servers start reconstruction for ID.d (note that there are at least $p+1$ honest servers). Because these parties completed the sharing,

they each have a share polynomial that can be linked back to the common root commitment C^*. Hence, they can construct an evaluation at $S_i(i)$ that will be accepted by others. Once $p+1$ such points are received, each honest server can recover a secret.

Proof (Correctness). First, assume that an honest dealer shared a secret s. Then, the share polynomial evaluations at all $\{S_i(i)\}_{i\in[1,n]}$ lie on a degree p polynomial that will recover s. By the Agreement property, once an honest server completes the sharing, the parties will have a common root commitment C^*. Thus, the only way to deviate from a reconstruction of s is to reveal an invalid evaluation of $S_i(i)$, which occurs with negligible probability by evaluation binding (Def. 3).

The second correctness property requires evaluation and degree binding. Even with a dishonest dealer, still the parties reconstruct the secret using $p+1$ of the n points $S_1(1), \ldots, S_n(n)$. This reconstruction is unique if and only if that polynomial is of degree p. Recall that at least $t+1$ honest parties verified during the sharing stage that $R(j) - S_j(j) = 0$ for all j, where this polynomial is of degree p (line 18). Evaluation binding ensures that the $S_i(i)$ evaluations revealed during reconstruction match the values tested earlier, and degree binding ensures that the points lie on a degree p polynomial as desired.

Proof (Efficiency). The protocol achieves a message complexity of $O(n^2)$ because every party sends n echo, ready, and reconstruct-share messages (plus n send messages for the dealer). Also assuming a field size $|\mathbb{F}| = O(\kappa)$, every stage of Haven has $O(\kappa n^2 c)$ communication complexity. In the send stage the dealer sends n messages of size $O(\kappa n c)$, and in all other stages each party sends n messages of size $O(\kappa c)$.

3.3 Constructing the Underlying Commitments

To complete our Haven construction for short secrets, it remains only to construct deterministic polynomial commitment schemes that satisfy Definitions 2-5 for random polynomials with short commitments and evaluations. In this section, we present two such constructions. The first construction is based on Bulletproofs [13], and it provides constant-size commitments and logarithmic-size evaluations without trusted setup. The second construction is based on the scheme of Kate et al. [26], and its commitments and evaluations are constant-sized at the expense of requiring trusted setup.

Deterministic Bulletproofs. Bulletproofs [13] are constant-sized vector commitments that support logarithmic-sized arguments of the result of an inner product operation applied to two (committed) vectors. One can construct a polynomial commitment from Bulletproofs as follows: $\mathrm{Com}(\mathsf{pp}, \phi, d)$ commits to the vector $\boldsymbol{\phi} = \langle \phi_j \rangle_{j\in[0,d]}$ of coefficients of the polynomial, $\mathrm{Eval}(\mathsf{pp}, \phi, i)$ constructs the vector $\boldsymbol{i} = \langle 1, i, i^2, \ldots, i^d \rangle$ (padding with 0s if needed) and produces an argument to the value of $\boldsymbol{\phi} \cdot \boldsymbol{i} = \phi(i)$, and Verify checks this argument.

The commitment scheme in Bulletproofs is deterministic: $\mathrm{Setup}(1^\kappa, D)$ uniformly samples $D+1$ group elements $g_0, \ldots, g_D \leftarrow G_\kappa$, and then $\mathrm{Com}(\phi) =$

$\prod_{j=0}^{d} g_j^{\phi_j}$. If the discrete log assumption holds for the family $\mathcal{G} = \{G_\kappa\}_{\kappa\in\mathbb{N}}$, then this commitment scheme is binding, and it is also hiding when the polynomial ϕ is chosen uniformly at random (even though it is not hiding otherwise) Bulletproofs are also homomorphic, and its argument reveals an upper bound d on the degree of the committed polynomial (it's the number of non-zero entries in the public vector $\boldsymbol{i}$).

The only remaining issue is with Definition 5, since Bulletproof arguments are not hiding. We can resolve the issue of hiding using the blinding technique previously used by [9, 12, 13, 21]. Concretely, because $\text{Eval}(\mathsf{pp}, \phi, i)$ cannot show an argument for the inner product $\boldsymbol{\phi}\cdot\boldsymbol{i}$ directly, instead the evaluator can: sample and commit to a random ephemeral polynomial $\psi \in \mathbb{F}[x]$ of degree d, send this commitment along with the two field elements $\phi(i)$ and $\psi(i)$ to the verifier, query the (public coins) verifier for a challenge $c \in \mathbb{F}$, and use the homomorphic property to construct a non-hiding argument proving that $(\boldsymbol{\psi} + c\boldsymbol{\phi}) \cdot \boldsymbol{i}$ equals $\psi(i)+c\phi(i)$. It is acceptable for this argument to reveal information about $\psi+c\phi$ because the polynomial ψ serves as a one-time pad that hides ϕ from the verifier.

We believe that this construction can be adapted to construct deterministic versions (for random polynomials) of other linear combination schemes [8] such as DARK [12], Dory [30], and the post-quantum polynomial commitment schemes [7, 27, 37] based on FRI [6]. We leave this as an open question for future work.

Deterministic KZG commitments. Kate et al. construct two polynomial commitment schemes, the first of which (called $\mathsf{PolyCommit}_{\text{DL}}$ in their work [26, §3.2]) is already deterministic and was shown to meet Definitions 2, 3, and 5 based on the t-bilinear strong Diffie-Hellman assumption. The only discrepancy between our requirements and their construction is Definition 4. We must verify that the commitments of the share polynomials S_i are of degree at most t. However, the $\mathsf{PolyCommit}_{\text{DL}}$ construction is predicated upon using trusted setup to generate powers of a generator element $\mathsf{pp} = \langle g, g^2, \ldots, g^D\rangle$, and once this information is public, it is impossible to verify whether a committer has committed to a polynomial of the maximum degree or a smaller one.

If there exist constant integers α and β such that $p = \alpha t + \beta$ (such as the case where $n = 3t+1$ and $p = 2t$), then there is a simple resolution to this issue: always construct polynomial commitments of maximum degree so that we can rely on strong correctness (Def. 2) instead. Observe that throughout Algorithm 1, there only exist commitments to polynomials of two different degrees: R of degree $p = \alpha t$ (in line 5) and each S_i of degree t (in line 6). In our construction, party P_i must be able to (a) interpolate S_i when given evaluations from $t + 1$ honest parties and (b) verify that the share polynomials are constructed in this fashion. Ergo, we can set the maximum degree $D = p$ during setup, sample S_i as a polynomial of degree D, and adjust line 6 of the Eval method to provide α distinct evaluations of the polynomial to each party (say, party i receives evaluations at the points i, $i + t$, $i + 2t$, ...) as well as β evaluations of the polynomial in common to all parties (say, at points $\alpha t + 1$, $\alpha t + 2$, ..., $\alpha t + \beta$). Each test polynomial T_i is now the difference of two degree-D polynomials, so it is also of degree D with overwhelming probability. Finally, while each party receives

α points on each share polynomial S_i, it suffices that only one of those points intersect with R for the test polynomial to guarantee correct reconstruction.

4 Amortizing Haven for Long Secrets

In this section, we show how to extend Haven to share a large secret s using $O(\kappa n)$ communication complexity such that any $p+1$ people can reconstruct the secret. Following the techniques used by Krawczyk [29] and Cachin and Tessaro [17], the core idea is to use a communication-efficient protocol for asynchronous reliable broadcast of the ciphertext corresponding to the long secret s, alongside Algorithm 1 to share the ephemeral symmetric key.

Building blocks. In more detail, our construction uses a (t, n)-information dispersal algorithm IDA [35], a semantically secure symmetric key encryption scheme, and a collision-resistant hash function H. By comparison to Shamir secret sharing, an IDA scheme is similar in that it contains algorithms to split and reconstruct an object to and from shares, respectively, but it differs in two ways:

1. Shares of an IDA might leak information about the original object while Shamir shares do not.
2. Shares of an IDA are smaller in size than the original object while Shamir shares are of the same size.

More formally, an IDA consists of the following two algorithms.

1. $\mathsf{split}(f, t, n)$: Splits an object f into n shares such that any t can reconstruct the original object f where each share has size $|f|/t$.
2. $\mathsf{reconstruct}(s)$: Takes a vector of t shares and combine them to reconstruct the original object f.

Also, we define an encryption scheme as containing a key generation algorithm, an encryption method $\mathrm{Enc} : k, m \mapsto c$, and a decryption method $\mathrm{Dec} : k, c \mapsto m$ such that no probabilistic polynomial time adversary can distinguish ciphertexts belonging to two arbitrarily-chosen plaintexts m_0 or m_1 with noticeable probability. A hash function $H : \{0,1\}^* \to \mathbb{F}$ is called collision resistant if no polynomial time adversary can find two inputs x and $x' \neq x$ such that $H(x) = H(x')$ with non-negligible probability. We refer readers to [23] for formal definitions.

Our new construction. To support long secrets, our Amortized Haven protocol makes the following additions to the sharing phase in Algorithm 1.

– At the start of the protocol, the dealer generates a random key $k \in \mathbb{F}$ and encrypts the large secret s using k by running $c = \mathrm{Enc}_k(s)$. Using $\mathsf{split}(c, t+1, n)$, the ciphertext is then encoded in n pieces $c_1, \ldots, c_n$ of length $|s|/t$. Additionally, we add $h_i = H(c_i)$ to the vector that forms the root commitment (line 11). Finally, the send message to party P_i also includes the ciphertext c and all witnesses to different hashes of every piece $h_i = H(c_i)$ that is included in the root commitment (line 14).

- Upon receiving (ID.d, send, set$_j$): party P_i adds one more consistency check to the list of requirements for her to produce an echo response. Namely, P_i checks that the every piece c_j is linked back to the root commitment using the witness provided by the dealer. To acquire each piece c_j, P_i runs split$(c, t+1, n)$ the same way the dealer did. P_i then adds to info$_{i,j}$ the corresponding c_i along with it's witness.
- Upon receiving(ID.d, echo, info$_{j,i}$): party P_i adds one more consistency check to the list of requirements for her to produce a ready response. Namely, P_i checks that every c_m is linked back to the root commitment using the witness provided by P_m. The rest of the protocol proceeds as normal.

At the start of reconstruction, each honest party recovers the large ciphertext by running reconstruct IDA on $t+1$ pieces that are linked to the root commitment that we have agreement on. The reconstruction phase continues as before and reconstructs the key k. The parties use the symmetric key k to decrypt the ciphertext and recover the original large secret s.

Theorem 2. *Suppose that $t = O(n)$, the underlying polynomial and vector commitments satisfy Defs. 2-6,* Enc *is a semantically secure encryption scheme, and H is collision resistant hash function. Then, amortized* Haven *for a large secret $s = \Omega(n \log n)$ is a high threshold AVSS that achieves a message complexity of $O(n^2)$ and a communication overhead of $O(\kappa n)$.*

Proof. We first examine communication costs. None of the changes above impact the message complexity: each party still sends $O(n)$ send, echo, ready, and reconstruct-share messages. The communication complexity now has two components: the HAVSS for the short key and the IDA for the long message. These costs sum to $O(\kappa n^2 \log n) + O(((|s|/t) \cdot n) \cdot n) = O(n|s|)$ as desired.

Next, the only two properties of an HAVSS that are directly impacted by the IDA of an encryption of the long secret are privacy and agreement. We argue about each property below in turn. Both properties only require minor adjustments to the arguments made in the proof of Theorem 1.

- Privacy: If the dealer is honest, the privacy argument in Theorem 1 guarantees that the adversary doesn't get hold of the secret key k used to encrypt the large secret s. The only thing that the adversary would get hold of is encrypted shares of the s. By definition of semantic security the attacker will not be able to extract any useful information about s from the ciphertext except with negligible probability.
- Agreement: Since at least $t+1$ honest parties have to verify that all pieces $\{c_i\}$ are part of the polynomial commitment. Then if agreement is reached over the polynomial commitment, then agreement is reached over the pieces. Availability is also guaranteed, since each honest party has heard $t+1$ echo messages from honest parties. Hence each honest party would have available $t+1$ pieces that are consistent with the root commitment, enough to reconstruct the ciphertext c and thus the large secret s.

Acknowledgments

The authors are grateful to Ran Canetti and the anonymous reviewers for their valuable feedback. This material is based upon work supported by the DARPA SIEVE program under Agreement No. HR00112020021 and the National Science Foundation under Grants No. 1414119, 1718135, 1801564, and 1931714.

References

1. Ittai Abraham, Kartik Nayak, Ling Ren, and Nibesh Shrestha. On the optimality of optimistic responsiveness. *IACR Cryptol. ePrint Arch.*, 2020:458, 2020.
2. Michael Backes, Amit Datta, and Aniket Kate. Asynchronous computational vss with reduced communication complexity. In Ed Dawson, editor, *Topics in Cryptology – CT-RSA 2013*, pages 259–276, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
3. Soumya Basu, Dahlia Malkhi, Mike Reiter, and Alin Tomescu. Asynchronous verifiable secret-sharing protocols on a good day. *CoRR*, abs/1807.03720, 2018.
4. Soumya Basu, Alin Tomescu, Ittai Abraham, Dahlia Malkhi, Michael K. Reiter, and Emin Gün Sirer. Efficient verifiable secret sharing with share recovery in bft protocols. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 2387–2402, New York, NY, USA, 2019. Association for Computing Machinery.
5. Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, page 52–61, New York, NY, USA, 1993. Association for Computing Machinery.
6. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In *ICALP*, volume 107 of *LIPIcs*, pages 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
7. Eli Ben-Sasson, Lior Goldberg, Swastik Kopparty, and Shubhangi Saraf. DEEP-FRI: sampling outside the box improves soundness. In *ITCS*, volume 151 of *LIPIcs*, pages 5:1–5:32. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
8. Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Halo infinite: Recursive zk-snarks from any additive polynomial commitment scheme. *IACR Cryptol. ePrint Arch.*, 2020:1536, 2020.
9. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT (2)*, volume 9666 of *Lecture Notes in Computer Science*, pages 327–357. Springer, 2016.
10. Jonathan Bootle and Jens Groth. Efficient batch zero-knowledge arguments for low degree polynomials. In *Public Key Cryptography (2)*, volume 10770 of *Lecture Notes in Computer Science*, pages 561–588. Springer, 2018.
11. Gabriel Bracha. Asynchronous byzantine agreement protocols. *Inf. Comput.*, 75(2):130–143, 1987.
12. Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from DARK compilers. In *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, pages 677–706, 2020.

13. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334, 2018.
14. Christian Cachin. An asynchronous protocol for distributed computation of RSA inverses and its applications. In *Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing, PODC 2003, Boston, Massachusetts, USA, July 13-16, 2003*, pages 153–162, 2003.
15. Christian Cachin, Klaus Kursawe, Anna Lysyanskaya, and Reto Strobl. Asynchronous verifiable secret sharing and proactive cryptosystems. In *ACM Conference on Computer and Communications Security*, pages 88–97. ACM, 2002.
16. Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantipole: practical asynchronous byzantine agreement using cryptography (extended abstract). In *PODC*, pages 123–132. ACM, 2000.
17. Christian Cachin and Stefano Tessaro. Asynchronous verifiable information dispersal. In *DISC*, volume 3724 of *Lecture Notes in Computer Science*, pages 503–504. Springer, 2005.
18. Ran Canetti. *Studies in secure multiparty computation and applications*. PhD thesis, Citeseer, 1996.
19. Ran Canetti and Tal Rabin. Fast asynchronous byzantine agreement with optimal resilience. In *STOC*, pages 42–51. ACM, 1993.
20. Dario Catalano and Dario Fiore. Vector commitments and their applications. In *Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 55–72. Springer, 2013.
21. Alessandro Chiesa, Michael A. Forbes, and Nicholas Spooner. A zero knowledge sumcheck and its applications. *Electron. Colloquium Comput. Complex.*, 24:57, 2017.
22. Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 383–395, 1985.
23. Oded Goldreich. A uniform-complexity treatment of encryption and zero-knowledge. *Journal of Cryptology*, 6, 06 1998.
24. Martin Hirt, Ard Kastrati, and Chen-Da Liu-Zhang. Multi-threshold asynchronous reliable broadcast and consensus. *IACR Cryptol. ePrint Arch.*, 2020:958, 2020.
25. Aniket Kate, Andrew K. Miller, and Tom Yurek. Brief note: Asynchronous verifiable secret sharing with optimal resilience and linear amortized overhead. *CoRR*, abs/1902.06095, 2019.
26. Aniket Kate, Gregory M. Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2010.
27. Assimakis Kattis, Konstantin Panarin, and Alexander Vlasov. Redshift: Transparent snarks from list polynomial commitment iops. *IACR Cryptol. ePrint Arch.*, 2019:1400, 2019.
28. Eleftherios Kokoris-Kogias, Alexander Spiegelman, and Dahlia Malkhi. Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020.
29. Hugo Krawczyk. Secret sharing made short. In *Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '93, page 136–146, Berlin, Heidelberg, 1993. Springer-Verlag.

30. Jonathan Lee. Dory: Efficient, transparent arguments for generalised inner products and polynomial commitments. *IACR Cryptol. ePrint Arch.*, 2020:1274, 2020.
31. Benoît Libert and Moti Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 499–517. Springer, 2010.
32. Ralph C. Merkle. A digital signature based on a conventional encryption function. In *CRYPTO*, volume 293 of *Lecture Notes in Computer Science*, pages 369–378. Springer, 1987.
33. Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. In *DISC*, volume 91 of *LIPIcs*, pages 39:1–39:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.
34. Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.
35. Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM*, 36(2):335–348, 1989.
36. Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.
37. Alexander Vlasov and Konstantin Panarin. Transparent polynomial commitment scheme with polylogarithmic communication complexity. *IACR Cryptol. ePrint Arch.*, 2019:1020, 2019.