2022-07-20

Brief announcement: asynchronous verifiable information dispersal with near-optimal communication

Asynchronous Verifiable Information Dispersal with Near-Optimal Communication*

NICOLAS ALHADDAD, Boston University, USA
SOURAV DAS, University of Illinois at Urbana-Champaign, USA
SISI DUAN, Tsinghua University, China
LING REN, University of Illinois at Urbana-Champaign, USA
MAYANK VARIA, Boston University, USA
ZHUOLUN XIANG, University of Illinois at Urbana-Champaign, USA
HAIBIN ZHANG, Beijing Institute of Technology, China

We present a near-optimal asynchronous verifiable information dispersal (AVID) protocol. The total dispersal cost of our AVID protocol is $O(|M|+\kappa n^2)$, and the retrieval cost per client is $O(|M|+\kappa n)$. Unlike prior works, our AVID protocol only assumes the existence of collision-resistant hash functions. Also, in our AVID protocol, the dispersing client incurs a communication cost of $O(|M|+\kappa n)$ in comparison to $O(|M|+\kappa n \log n)$ of prior best. Moreover, each node in our AVID protocol incurs a storage cost of $O(|M|/n + \kappa)$ bits, in comparison to $O(|M|/n + \kappa \log n)$ bits of prior best. Finally, we present lower bound results on communication cost and show that our AVID protocol has near-optimal communication costs – only a factor of $O(\kappa)$ gap from the lower bounds.

1 INTRODUCTION

Verifiable information dispersal (VID), introduced by Rabin [12], is a primitive with emerging applications in fault-tolerant replication [14], distributed storage [10]. VID lets a *client*, here on referred to as the *dispersing* client, disperse a message among a set of nodes during the *dispersal phase*, such that during the *retrieval phase* the message can be later retrieved by any node or any other client, which we refer to as the *retrieving* client. A VID protocol immediately implies a RBC protocol, where the broadcaster acts as the dispersing client, and each node retrieves the data by acting as a retrieving client. In this paper, we consider the VID problem in asynchronous networks (AVID), and we assume Byzantine faults that may deviate arbitrarily from the protocols. We consider the unauthenticated setting where the protocol does not use digital signatures, but we assume collision-resistant hash functions.

Existing works. Cachin and Tessaro [6] presented the first AVID protocol, with dispersal phase cost $O(n|M|+\kappa n^2 \log n)$ and retrieval phase cost $O(|M|+\kappa n \log n)$. It is then improved by Hendricks et al. [9], and very recently, by Alhaddad et al. and Yang et al. [2, 14] to $O(|M|+\kappa n^2)$ for the dispersal and $O(|M|+\kappa n \log n)$ for the retrieval phase. In their protocols, both the dispersing client and the retrieving client incur a cost of $O(|M|+\kappa n \log n)$, and each node incurs $O(|M|/n + \kappa \log n)$ storage cost. We summarize the existing works on AVID in Table 1 and describe them in more detail in §5.

Our contributions. Our main contribution is an AVID protocol that does not require any trusted setup and has a communication cost of $O(|M|+\kappa n^2)$ during the dispersal phase. Moreover, in our AVID protocol, both dispersing and retrieving clients incur a communication cost of $O(|M|+\kappa n)$. We also reduce the per node storage to $O(|M|/n + \kappa)$, and the communication cost of the retrieval phase to $O(|M|+\kappa n)$ by designing a novel retrieval phase. We also present a

*This publication is a merge of the following ePrint papers [3, 8]. All authors contributed equally to this work and are listed alphabetically. Corresponding authors: Sisi Duan and Haibin Zhang.

Authors' addresses: Nicolas Alhaddad, Boston University, USA, nhaddad@bu.edu; Sourav Das, University of Illinois at Urbana-Champaign, USA, souravd2@illinois.edu; Sisi Duan, Tsinghua University, China, duansisi@tsinghua.edu.cn; Ling Ren, University of Illinois at Urbana-Champaign, USA, renling@illinois.edu; Mayank Varia, Boston University, USA, varia@bu.edu; Zhuolun Xiang, University of Illinois at Urbana-Champaign, USA, xiangzl@illinois.edu; Haibin Zhang, Beijing Institute of Technology, China, haibin@bit.edu.cn.

Manuscript submitted to ACM

Table 1. Comparison with existing AVID protocols. The following accronyms are used in the table; DL: Discrete Logarithm, CRS: Common Reference String, q-SDH: q-Strong Diffie-Hellman.

Scheme	Dispersal Cost (client)	Dispersal Cost (total)	Retrieval Cost (total)	Storage Cost (total)	Cryptographic Assumption	Setup
Cachin-Tessaro [6]	$O(\|M\|+\kappa n \log n)$	$O(n\|M\|+\kappa n^2 \log n)$	$O(\|M\|+\kappa n \log n)$	$O(\|M\|+\kappa n \log n)$	Hash	None
Hendricks et al. [9]	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n^3)$	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n^2)$	Hash	None
Alhaddad et al. [2]	$O(\|M\|+\kappa n \log n)$	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n \log n)$	$O(\|M\|+\kappa n \log n)$	DL	CRS
Alhaddad et al. [2]	$O(\|M\|+\kappa n)$	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n)$	$O(\|M\|+\kappa n)$	q-SDH+Hash	Trusted
DisperseLedger [14]	$O(\|M\|+\kappa n \log n)$	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n \log n)$	$O(\|M\|+\kappa n \log n)$	Hash	None
This work	$O(\|M\|+\kappa n)$	$O(\|M\|+\kappa n^2)$	$O(\|M\|+\kappa n)$	$O(\|M\|+\kappa n)$	Hash	None
Lower bound	$\Omega(\|M\|+n)$	$\Omega(\|M\|+n^2)$	$\Omega(\|M\|+n)$	$\Omega(\|M\|)$	—	—

lower bound result on the communication cost of any AVID protocol. In particular, we prove that in any deterministic AVID protocol, the dispersal phase has a communication cost of $\Omega(|M|+n^2)$ and the retrieval phase has a communication cost of $\Omega(|M|+n)$. Hence, our AVID protocol above has near-optimal communication costs – only a factor of $O(\kappa)$ gap from the lower bounds.

2 PRELIMINARIES

System Model. We consider a network of n nodes where every pair of nodes is connected via a pairwise authenticated channel. We consider the presence of a malicious adversary $\mathcal{A}$ that can corrupt up to t nodes in the network. The corrupted nodes can behave arbitrarily, and we call a node honest if it remains non-faulty for the entire protocol execution. We assume the network is asynchronous, i.e., $\mathcal{A}$ can arbitrarily delay any message but must eventually deliver all messages sent between honest nodes. We use $|S|$ to denote the size of a set S. Let $\mathbb{F}$ be a finite field. For any integer a, we use $[a]$ to denote the set $\{1, 2, \ldots, a\}$. We use κ to denote the size of the output of the collision-resistant hash function. Naturally, we assume that $\kappa > \log n$.

Problem Formulations. An AVID protocol has two functions: `DISPERSE`(M), which a dispersing client invokes to disperse a message M to n nodes, and `RETRIEVE`, which a (possibly different) retrieving client invokes to retrieve the message M.

Definition 1 (Asynchronous Verifiable Information Dispersal [6])**.** An asynchronous verifiable information dispersal (AVID) scheme for a message M consists of a pair of protocols `DISPERSE` and `RETRIEVE` which satisfy the following requirements under asynchrony:

- *Termination:* If an honest client invokes `DISPERSE`(M) and no other client invokes `DISPERSE` on the same instance, then every honest node eventually finishes the dispersal phase.
- *Agreement*: If any honest node finishes the dispersal phase, all honest nodes eventually finish the dispersal phase.
- *Availability*: If an honest node has finished the dispersal phase, and some honest client initiates `RETRIEVE`, then the client eventually reconstructs some message M'.
- *Correctness*: If an honest node has finished the dispersal phase, then honest clients always reconstruct the same message M' when invoking `RETRIEVE`. Furthermore, if an honest client invoked `DISPERSE`(M) and no other client invokes `DISPERSE` on the same instance, then $M' = M$.

2.1 Primitives

Error Correcting Code. We use error-correcting codes, such as Reed-Solomon (RS) codes [13]. Let RSEnc(M, m, k) be the encoding algorithm. Briefly, the RSEnc takes as input a message M consisting of k symbols, treats it as a polynomial of degree $k-1$, and outputs m evaluations of the corresponding polynomial. Let RSDec(k, r, T) be the RS decoding procedure. RSDec takes as input a set of symbols T (some of which may be incorrect), and outputs a degree $k-1$ polynomial, i.e., k symbols, by correcting up to r errors (incorrect symbols) in T. It is well-known that RSDec can correct up to r errors in T and output the original message provided that $|T| \geq k + 2r$.

Online Error Correction. The OEC [4] takes a set T consisting of tuples (j, a_j) where j is an index $j \in [n]$ and a_j is a symbol of a Reed-Solomon codeword. The OEC algorithm then tries to decode a message M such that Reed-Solomon encoding of M matches with at least $2t+1$ elements in T. We summarize the OEC in Algorithm 1.

Collision-resistant Hash Function. A cryptographic collision-resistant hash function guarantees that a computationally bounded adversary cannot come up with two inputs that hash to the same value, except for a negligible probability.

Algorithm 1 Information Theoretic Online Error-correcting (IT-OEC) protocol

1: **Input:** T *// T consisting of tuples (j, a_j) where $j \in [n]$ and a_j is a symbol*
2: **for** $0 \leq r \leq t$ **do** *// online error correction*
3: Wait till $|T| \geq 2t + r + 1$
4: Let $p_r(\cdot) :=$ RSDec$(t+1, r, T)$
5: **if** $2t+1$ elements $(j, a) \in T$ satisfy $p_r(j) = a$ **then**
6: let M be the coefficients of $p_r(\cdot)$
7: return M

3 AVID

Challenges and Our Approaches. In the state-of-the-art AVID protocol with no trusted setup, both the dispersing and retrieving clients incur communication cost of $O(|M| + \kappa n \log n)$. This is due to sending a Merkle path and an encoded symbol of the input message to each node. Nodes use the Merkle path to check and prove the consistency of the encoded symbol. The cost can be improved to $O(|M| + \kappa n)$ using a trusted setup phase and polynomial commitments [2]. Omitting the Merkle tree or commitment in a naïve manner introduces the challenge that, during the dispersal phase, nodes cannot check the consistency of the symbol they receive. Moreover, the lack of a consistency proof allows Byzantine nodes to equivocate to different retrieving clients, violating Correctness.

Briefly, we address these challenges with the following ideas. First, we replace the Merkle tree with a vector of hashes of the encoded symbols and then let the dispersing client reliably broadcast the entire vector of hashes to every node. The nodes then use this vector to check the consistency of the symbols they receive from the dispersing client. Note that, during the retrieval phase, the retrieving client does not have the vector of hashes and thus cannot directly validate the symbols it receives from other nodes. Furthermore, asking each sender to send the entire vector would result in $O(\kappa n^2)$ communication cost. We address this by first letting the client retrieve the vector of hashes, and then use the retrieved vector for the rest of the retrieval phase.

Design of AVID. In the dispersal phase, first, the dispersing client encodes the message M using a $(n, t+1)$ Reed-Solomon code (line 2) and computes a hash vector H for the encoded message (line 3). The dispersing client then sends the i-th symbol m_i to the i-th node, and reliably broadcasts H using a balanced reliable broadcast protocol BalBRB from [1] (line 4). During the BalBRB, as per the predicate, the i-th node checks whether the i-th element of the hash vector that

Algorithm 2 Pseudocode for AVID

// the dispersing client invokes DISPERSE*(M)*
1: **input** M
2: Let $M' := [m_1, m_2, \ldots, m_n] := \mathsf{RSEnc}(M, n, t+1)$
3: Let $H := [\mathsf{hash}(m_1), \mathsf{hash}(m_2), \ldots, \mathsf{hash}(m_n)]$
4: Send m_i to node i for each $i \in [n]$, and invoke BalBRB(H) with predicate $P(\cdot)$ described below

// additional predicate $P(\cdot)$ for node i to check in BalBRB
5: **procedure** $P(H)$
6: **upon** receiving m_i from the dispersing client **do**
7: **return** true iff $\mathsf{hash}(m_i) = H[i]$

// code for node i during the dispersal phase
8: Wait till BalBRB($\cdot$) terminate
9: Let $H = [h_1, h_2, \ldots, h_n]$ be the output of BalBRB($\cdot$)
10: Let $h = \mathsf{hash}(H)$
11: $[h'_1, h'_2, \ldots, h'_n] := \mathsf{RSEnc}(H, n, t+1)$
12: Output and store $\langle m_i, h'_i, h \rangle$ for the dispersal phase

// the retrieving client invokes RETRIEVE
13: **send** $\langle$RETRIEVE$\rangle$ to all nodes

// retrieving H
14: Let $T_h := \{\}$ and $T_M := \{\}$
15: For every $\langle \mathsf{HASH}, h'_j, h \rangle$ received from j, add (j, h'_j) to T_h
16: For every $\langle \mathsf{SYMBOL}, m_j \rangle$ received from j, add (j, m_j) to T_M
17: Run IT-OEC using T_h
18: Let $H :=$ IT-OEC(T_h)

// retrieving M after retrieving H
19: **for** each $(j, a) \in T_M$ **do**
20: **if** $\mathsf{hash}(a) = H[j]$ **then**
21: add (j, a) to T
22: Wait till $|T| = t+1$
23: Interpolate T as a degree-t polynomial
24: Let M' be the interpolated polynomial evaluated at every element in $[n]$
25: **if** $\exists j \in [n]$ such that $\mathsf{hash}(M'[j]) \neq H[j]$ **then**
26: **output** $\perp$ and **return**
27: **else**
28: **output** $\mathsf{RSDec}(t+1, 0, M')$ and **return**

// code for node i during the retrieval phase
29: **upon** receiving $\langle$RETRIEVE$\rangle$ from the retrieving client **do**
30: Wait till the dispersal phase outputs $\langle m_i, h'_i, h \rangle$
31: send $\langle \mathsf{HASH}, h'_i, h \rangle$ to the retrieving client
32: **if** $m_i \neq \perp$ **then**
33: send $\langle \mathsf{SYMBOL}, m_i \rangle$ to the retrieving client

is being reliably broadcast, is equal to the hash of the symbol it received from the dispersing client (line 5-7). Let H be the output of the validated RBC. Each node then encodes H and compute $h = \mathsf{hash}(H)$. At the end of the dispersal phase, the i-th node outputs $\langle m_i, h'_i, h \rangle$ where $m_i = \perp$ if the i-th node did not receive a valid symbol from the dispersing client, and h'_i is the i-th encoding symbol of H.

The main idea of the retrieval phase is to let the retrieving client first recover the vector H and then use it to validate symbols sent by nodes. More specifically, during the retrieval phase, the retrieving client sends RETRIEVE request to all nodes (line 13). Upon receiving RETRIEVE request from the retrieving client, each node waits till the dispersal phase

terminates (line 30). The i-th node then sends the message $\langle \mathsf{HASH}, h'_i, h \rangle$ to the retrieving client (line 31). Additionally, if node i received a symbol m_i during the dispersal phase such that $\mathsf{hash}(m_i) = H[i]$, it sends a $\langle \mathtt{SYMBOL}, m_i \rangle$ to the retrieving client (line 32-33). The retrieving client upon receiving messages stores the symbols in T_h and T_M. The retrieving client then uses T_h and the standard online error correction to recover H (line 17-18). After recovering H, the retrieving client uses it to retrieve the message. In particular, for every tuple $(j, a) \in T_M$, it first checks whether $\mathsf{hash}(a) = H[j]$ and adds the tuple (j, a) to the set T (line 19-21). The retrieving client waits till $|T| = t + 1$ and then interpolates the tuples in T into a polynomial of degree at most t (line 22-23). Let M' be the interpolated polynomial. The client then checks if there exists any $j \in [n]$ such that $\mathsf{hash}(M[j]) \neq H[j]$. If such j exists, then the client outputs $\perp$ and returns. Otherwise, the client outputs the Reed-Solomon decoding of M' (line 28).

3.1 Analysis of AVID

We next analyze the properties of our AVID protocol and its performance.

Theorem 1 (Termination and Agreement)**.** *If an honest dispersing client invokes* $\mathtt{DISPERSE}(M)$ *and no other client invokes* DISPERSE *on the same instance, then every honest node eventually finishes the dispersal phase. If any honest node finishes the dispersal phase, all honest nodes eventually finish the dispersal phase.*

Proof. An honest dispersing client sends the correct symbols $[m_1, m_2, \ldots, m_n] = \mathsf{RSEnc}(M, n, t + 1)$, and reliably broadcasts the hash vector $H = [\mathsf{hash}(m_1), \mathsf{hash}(m_2), \ldots, \mathsf{hash}(m_n)]$. By the Validity property of the RBC, the RBC will terminate at all honest nodes. Hence, every honest node will finish the dispersal phase.

A nodes terminates the dispersal phase if and only if the RBC protocol terminates. Thus, by the Totality property of the RBC every node will terminate the RBC and thus terminate the dispersal phase. □

Lemma 1. *If the dispersal phase terminates at an honest node, then every honest node will output the same vector of hashes* $H = [h_1, h_2, \ldots, h_n]$ *for RBC. Furthermore, at least* $t + 1$ *honest nodes have received a symbol that matches with the corresponding location of* H.

Proof. Nodes terminate the dispersal phase if and only if the RBC protocol terminates. Thus by the Totality and Agreement property of the RBC every node will receive the same message H. Furthermore, when any honest node finishes the RBC, it has received READY messages from at least $2t + 1$ nodes, among which at least $t + 1$ are honest. Thus, at least one honest node receives ECHO messages from at least $2t + 1$ nodes, among which at least $t + 1$ are honest. Before these honest nodes send ECHO messages, they have the predicate evaluated to be true, which implies that each honest node j above has received m_j from the client such that $\mathsf{hash}(m_j) = H[j]$. □

Lemma 2. *If an honest node has finished the dispersal phase with* H *as the output of the RBC, then any honest client can reconstruct the same* H *after invoking* RETRIEVE.

Proof. By Agreement of AVID, all honest nodes eventually finish the dispersal phase once an honest node has finished the dispersal phase. Also, due to Lemma 1 all honest nodes output the same H for RBC when the dispersal phase terminates. Therefore, when an honest client invokes RETRIEVE, all $2t + 1$ honest nodes will send the correct $\langle \mathsf{HASH}, h'_i, \mathsf{hash}(H) \rangle$ to the client. By OEC (line 17-20) and the collision resistance property of the hash function, the client can successfully decode the same hash vector H. □

Theorem 2 (Availability and Correctness). *If an honest node has finished the dispersal phase, and some honest clients invoke* RETRIEVE*, then they eventually output the same message M'. Furthermore, if an honest client invoked* DISPERSE(M) *and no other client invokes* DISPERSE *on the same instance, then $M' = M$.*

Proof. By Lemma 2, the honest client reconstructs the same hash vector H as the one output by any honest node during the dispersal phase, where H corresponds to some message M'. Moreover, at least $t + 1$ honest nodes have received m_j such that $\mathsf{hash}(m_j) = H[j]$. Therefore, when an honest client invokes RETRIEVE, all these $t + 1$ honest nodes will send the correct $\langle \mathtt{SYMBOL}, m_i \rangle$ where $\mathsf{hash}(m_i) = H[i]$ to the retrieving client, enabling it to reconstruct the message M'.

Let $f_u(\cdot)$ denote the polynomial of degree t or less a retrieving client u obtains via interpolation. The client u then uses $f_u(\cdot)$ to recover the message M_u only if $\mathsf{hash}(f_u(i)) = H[i]$ for all $i \in [n]$. Hence, due to collision resistance property of the $\mathsf{hash}(\cdot)$, if two retrieving client, u and v outputs messages $M_u \neq \perp$ and $M_v \neq \perp$, respectively, then $M_u = M_v$.

Also, if any honest retrieving client outputs $\perp$, then every honest retrieving client outputs $\perp$. For the sake of contradiction, let us assume that a retrieving client u outputs $\perp$ but another retrieving client $v \neq u$ outputs $M_v \neq \perp$. Let T_u be the set of indices used by retrieving client u to interpolate f_u. Then, there exists a $k \in [n] \setminus T_u$ such that $H[k] \neq \mathsf{hash}(f_u[k])$. Since retrieving client v outputs $M_v \neq \perp$, this implies $f_u[k] = f_v[k]$ for all $k \in T_u$. However, both f_u and f_v have degree t or less and agrees on $t + 1$ distinct points. This implies f_u and f_v matches as polynomial and $f_u[k] = f_v[k]$ for all $k \in [n]$, which is a contradiction.

If an honest dispersing client invoked DISPERSE(M) and no other dispersing client invokes DISPERSE on the same instance, by the Termination property of AVID, all honest nodes eventually finish the dispersal phase with RBC output $H = [\mathsf{hash}(m_1), \ldots, \mathsf{hash}(m_n)]$ where $[m_1, \ldots, m_n] = \mathsf{RSEnc}(M, n, t + 1)$. Also, from Lemma 2 the retrieving client will receive H during the retrieval phase. Then, by collision-resistant property of the hash function, the honest retrieving client use correct symbol to recover the message, hence will recover and output M. □

Theorem 3 (Performance). *The communication cost of a dispersing client during the dispersal phase is $O(|M|+\kappa n)$ and the total communication cost of the dispersal phase is $O(|M|+\kappa n^2)$. Also, each node incurs a storage cost of $O(|M|/n + \kappa)$. Furthermore, the total communication cost for retrieval at a client is $O(|M|+\kappa n)$.*

Proof. During the dispersal phase, the dispersing client only sends a symbol of size $O(|M|/n)$ to each node and reliably broadcasts a message of size κn. Hence, the total communication cost of the dispersing client is $O(|M|+\kappa n)$ from Lemma 8. Also, each node receives a symbol of size $O(|M|/n)$ and participates in the RBC of a message of size $O(\kappa n)$. Hence, using Lemma 8, the total communication cost of the dispersal phase is $O(|M|+\kappa n^2)$. At the end of the dispersal phase, each node stores two symbol of size $O(|M|/n)$ and $O(\kappa)$, respectively, and a hash output of size κ. Thus, the total storage cost of our AVID protocol is $O(|M|+\kappa n)$.

During retrieval, each node sends at most two symbols of size $O(\kappa)$ and $O(|M|/n)$ to the retrieving client. Hence, the communication cost of a single retrieving client is $O(|M|+\kappa n)$. □

4 LOWER BOUNDS

For AVID (or even synchronous VID), the communication cost of the dispersing client during dispersal phase is lower bounded by $\Omega(|M|+n)$ by the following simple argument – the dispersing client needs to send $\Omega(|M|)$ bits, and needs to send messages to at least $t + 1 = \Omega(n)$ nodes otherwise it is possible that no honest node receives any information from the client, and the Termination property of VID can be violated. For the total communication cost during dispersal, we will show the following $\Omega(|M|+n^2)$ lower bound for AVID.

Theorem 4. *Any deterministic protocol that solves AVID must incur a communication cost of* $\Omega(|M|+n^2)$ *during the dispersal phase in at least one execution.*

Proof. Recall that the communication cost of the dispersing client during dispersal phase is lower bounded by $\Omega(|M|+n)$. Now we will show that any deterministic protocol that solves AVID must uses $\geq (t/2)^2$ messages during the dispersal phase in at least one execution. Suppose it is not true, and there exists a deterministic AVID protocol that uses $< (t/2)^2$ messages during the dispersal phase for all executions. Consider the following two executions.

(1) *Execution $E1$:* Let the dispersing client be honest. The adversary corrupts the set of nodes B where $|B| = t/2$. Let A denote the rest of the nodes. For each node $b \in B$, b does not send messages to each other, and ignores the first $t/2$ messages received.

    One observation is that $\exists p \in B$ such that p receives $< t/2$ messages, since the protocol uses $< (t/2)^2$ messages during the dispersal phase and $|B| = t/2$. Also, due to Termination, all honest nodes in A terminate the dispersal phase in $E1$.

(2) *Execution $E2$ same as $E1$ except the following differences:* Let $A(P)$ denote the set of nodes that (attempt to) send p messages in $E1$. Since p receives $< t/2$ messages, $|A(p)| < t/2$. The adversary corrupts nodes in $A(p)$ and $B \setminus \{p\}$. The corrupted nodes in $B \setminus \{p\}$ behaves like in $E1$ and ignores all messages from node p. The corrupted nodes in $A(p)$ behave like in $E1$ but do not send messages to p.

Claim: $E1$ and $E2$ are distinguishable to honest nodes in $A \setminus A(p)$, and the honest nodes in $A \setminus A(p)$ will terminate the dispersal phase in $E2$ as well.

The claim can be shown by examining how every node behaves in $E1, E2$. Nodes in $B \setminus \{p\}$ behave the same to all nodes. Nodes in $A(p)$ behave the same to $A \setminus A(p)$. Node p behaves the same since in both executions it does not receive any message from others. Thus the claim is true. □

Now, consider two cases.

- If p never terminates the dispersal phase in $E2$, then Agreement property is violated since other honest nodes terminates the dispersal phase in $E2$, contradiction.
- If p terminates the dispersal phase in $E2$ without receiving any messages, suppose p terminates at time τ. Then consider another execution $E3$ where the dispersing client is Byzantine and remains silent, and all nodes are honest. All messages from any node to p are delayed after τ. Since p cannot distinguish $E2$ and $E3$, it terminates the dispersal phase in $E3$ at time τ as well. Due to the Agreement property, all honest nodes will eventually terminate the dispersal phase in $E3$. Now consider another execution $E4$ where the dispersing client with message M is honest but its messages are delayed, and all nodes are honest. For all honest nodes, they cannot distinguish $E3$ and $E4$ before receiving any message from the dispersing client. Therefore, these honest nodes will terminate the dispersal phase in $E4$ before receiving any message from the dispersing client. In the retrieval phase of $E4$, according to the Availability property, the retrieving client reconstructs some message eventually at some time τ'. Suppose the messages of the dispersing client are delayed beyond time τ' in $E4$. Since no honest node receives any message from the dispersing client, during the retrieval phase of $E4$ the dispersed message cannot be reconstructed, violating the Correctness property of AVID, hence a contradiction.

Hence, any deterministic protocol that solves AVID must uses $\geq (t/2)^2$ messages during the dispersal phase in at least one execution, and thus must incur a communication cost of $\Omega(\max\{|M|+n, (t/2)^2\}) = \Omega(|M|+n^2)$. □

Theorem 5. *Any deterministic protocol that solves AVID must incur a communication cost of $\Omega(|M|+n)$ during the retrieval phase in at least one execution.*

Proof. Consider any execution with an honest retrieving client. The client needs to receive at least $\Omega(|M|)$ bits from the honest nodes to obtain M, and the client needs to send messages to at least $t + 1 = \Omega(n)$ nodes for retrieval otherwise it could be the t nodes are Byzantine and ignore the message. Hence, the retrieval phase has cost $\Omega(|M|+n)$. □

5 RELATED WORK

The first AVID protocol is due to Cachin and Tessaro [6]. In their protocol, during the dispersal phase, each node, including the dispersing client, incurs a communication cost of $O(|M|+\kappa n \log n)$, leading to a total dispersal cost of $O(n|M|+\kappa n^2 \log n)$. This cost arises because every node needs to send a symbol and its associated Merkle path proof to all nodes. Finally, during retrieval, each retrieving client incurs a communication cost of $O(|M|+\kappa n \log n)$. Again, the $\log n$ factor is due to nodes sending Merkle path proofs to the retrieving client.

Hendricks et al. in [9] propose an alternate AVID protocol where during dispersal, the dispersing client incurs a communication cost of $O(|M|+\kappa n^2)$. They improve the communication cost using a non-interactive verification scheme with fingerprinted cross-checksum. In their protocol, only the dispersing client sends the symbols to all nodes, and the nodes perform an RBC on the fingerprinted cross-checksum but not the symbols. As a result, the remaining nodes incur a communication cost of $O(\kappa n^2)$. Hence, the total communication cost of their protocol during the dispersal phase is $O(|M|+\kappa n^3)$. Also, the total retrieval cost for a single retrieving client is $O(|M|+\kappa n^2)$, as each node sends a $O(\kappa n)$ size finger-printed cross-checksum and an encoded symbol to the retrieving client.

Very recently, Yang et al. [14] presents a new AVID protocol in which, during the dispersal phase, the dispersing client incurs a communication cost of $O(|M|+\kappa n \log n)$. Furthermore, the total communication cost of their dispersal phase is $O(|M|+\kappa n^2)$. The main innovation of the AVID protocol of [14] is that they remove the need for nodes to gossip symbols and Merkle path proofs during the dispersal phase. They do so by designing a novel retrieval protocol and a RBC on the root of the associated Merkle tree. Nevertheless, during the dispersal phase, the dispersing client still needs to send a Merkle path proof to every node. Moreover, during retrieval, each node still sends an encoded symbol and the associated Merkle path proof to the retrieving client, leading to a communication cost of $O(|M|+\kappa n \log n)$. Our protocol improves the communication costs of both these steps by a factor of $\log n$ using a vector of hashes instead of a Merkle tree, along with our balanced RBC protocol for long messages.

With trusted setup and assuming hardness of q-SDH [11], the recent work by Alhaddad et al. [2] achieves the dispersing client cost to $O(|M|+\kappa n)$ and the total communication to $O(|M|+\kappa n^2)$ using the KZG [11] polynomial commitment scheme. Our protocol achieves the same cost using only collision-resistant hash function without any trusted setup or additional cryptographic assumptions other than collision-resistant hash functions.

6 CONCLUSION

In this paper, we present an asynchronous verifiable information dispersal (AVID) protocol with near-optimal communication complexity, as well as some lower bounds for AVID.

ACKNOWLEDGMENTS

The authors would like to thank our shepherd Elad Schiller and PODC reviewers for their helpful comments on the paper.

Nicolas Alhaddad and Mayank Varia are supported by NSF Grants No. 1718135, 1801564, 1915763, and 1931714, by the DARPA SIEVE program under Agreement No. HR00112020021, and by DARPA and the Naval Information Warfare Center (NIWC) under Contract No. N66001-15-C-4071. Sisi Duan was supported in part by Tsinghua Independent Research Program, National Financial Cryptography Research Center, and National Key Research and Development Program of China under grant No. 2018YFA0704701. Ling Ren is supported by NSF Grant No. 2143058. Haibin Zhang was supported in part by Shandong Provincial Key Research and Development Program (2021CXGC010106) and Teli Youth Scholarship.

REFERENCES

[1] Nicolas Alhaddad, Sourav Das, Sisi Duan, Ling Ren, Mayank Varia, Zhuolun Xiang, and Haibin Zhang. Balanced byzantine reliable broadcast with near-optimal communication and improved computation. In *PODC*, 2022.

[2] Nicolas Alhaddad, Sisi Duan, Mayank Varia, and Haibin Zhang. Succinct erasure coding proof systems. *Cryptology ePrint Archive*, 2021.

[3] Nicolas Alhaddad, Sisi Duan, Mayank Varia, and Haibin Zhang. Practical and improved byzantine reliable broadcast and asynchronous verifiable information dispersal from hash functions. Cryptology ePrint Archive, Paper 2022/171, 2022.

[4] Michael Ben-Or, Ran Canetti, and Oded Goldreich. Asynchronous secure computation. In *STOC*, pages 52–61, 1993.

[5] Gabriel Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2):130–143, 1987.

[6] Christian Cachin and Stefano Tessaro. Asynchronous verifiable information dispersal. In *SRDS*, pages 191–201. IEEE, 2005.

[7] Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination and its applications. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 2705–2721, 2021.

[8] Sourav Das, Zhuolun Xiang, and Ling Ren. Near-optimal balanced reliable broadcast and asynchronous verifiable information dispersal. Cryptology ePrint Archive, Paper 2022/052, 2022.

[9] James Hendricks, Gregory R Ganger, and Michael K Reiter. Verifying distributed erasure-coded data. In *PODC*, pages 139–146, 2007.

[10] Ari Juels and Burton S Kaliski Jr. Pors: Proofs of retrievability for large files. In *CCS*, pages 584–597, 2007.

[11] Aniket Kate, Gregory M Zaverucha, and Ian Goldberg. Constant-size commitments to polynomials and their applications. In *International conference on the theory and application of cryptology and information security*, pages 177–194. Springer, 2010.

[12] Michael O Rabin. The information dispersal algorithm and its applications. In *Sequences*, pages 406–419. Springer, 1990.

[13] Irving S Reed and Gustave Solomon. Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, 8(2):300–304, 1960.

[14] Lei Yang, Seo Jin Park, Mohammad Alizadeh, Sreeram Kannan, and David Tse. Dispersedledger: High-throughput byzantine consensus on variable bandwidth networks. In *NSDI*, 2022.

Algorithm 3 BalBRB protocol for long messages

1: *// only broadcaster node*
2: **input** M
3: Let $h := \mathsf{hash}(M)$
4: Let $[m_1, m_2, \ldots, m_n] := \mathsf{RSEnc}(M, n, t+1)$
5: **send** $\langle \mathtt{PROPOSE}, m_j \rangle$ to node j for each $j \in [n]$

// each node i
6: Let $M := \perp$, $T := \{\}$, $T_h := \{\}$
7: **upon** receiving the first $\langle \mathtt{PROPOSE}, m_i \rangle$ from the broadcaster **do**
8: **send** $\langle \mathtt{SHARE}, m_i \rangle$ to all nodes

9: **upon** receiving the first $\langle \mathtt{SHARE}, m_j^* \rangle$ from any node j **do**
10: $T := T \cup \{(j, m_j^*)\}$

11: Run IT-OEC on the set T
12: Let M' be the output of IT-OEC(T)

13: **if** $P(M') = true$ **then** *// $P(\cdot)$ is an external predicate that returns true or false. See protocol description for more details.*
14: Let $h := \mathsf{hash}(M')$
15: **send** $\langle \mathtt{ECHO}, m_j, h \rangle$ to node j for each $j \in [n]$ where m_j is the j-th symbol of $\mathsf{RSEnc}(M', n, t+1)$

16: **upon** receiving $2t+1$ $\langle \mathtt{ECHO}, m_i, h \rangle$ for the same m_i, h and not having sent a READY message **do**
17: **send** $\langle \mathtt{READY}, m_i, h \rangle$ to all

18: **upon** receiving $t+1$ $\langle \mathtt{READY}, *, h \rangle$ for the same h and not having sent a READY message **do**
19: Wait for $t+1$ matching $\langle \mathtt{ECHO}, m_i', h \rangle$
20: **send** $\langle \mathtt{READY}, m_i', h \rangle$ to all

21: **upon** receiving the first $\langle \mathtt{READY}, m_j^*, h \rangle$ from any node j **do**
22: $T_h := T_h \cup \{(j, m_j^*)\}$

23: Run IT-OEC on the set T_h
24: Let M'' be the output of IT-OEC(T_h)
25: **output** M'' and **return**

A BALANCED RELIABLE BROADCAST

In this section, we provide the balanced reliable broadcast protocol named BalBRB proposed in [1]. The BalBRB protocol replaces the standard multicast with the balanced multicast [1] in the reliable broadcast protocol of [7] to obtain balanced communication cost. For completeness, we provide description and analysis of the protocol in this section.

Definition 2 (Reliable Broadcast [5])**.** A protocol for a set of nodes $\{1, \ldots, n\}$, where a designated broadcaster holds an input M, is a reliable broadcast (RBC) protocol, if the following properties hold

- *Agreement:* If an honest node outputs a message M' and another honest node outputs M'', then $M' = M''$.
- *Validity:* If the broadcaster is honest, all honest nodes eventually output the message M.
- *Totality:* If an honest node outputs a message, then every honest node eventually outputs a message.

Protocol description. In order to reduce the cost of the broadcaster node, protocol BalBRB first lets the broadcaster encode its message M into n symbols using a $(n, t+1)$ Reed-Solomon code (line 4) and only send the i-th symbol to node i together with the hash digest of the message M. In particular, let $[m_1, m_2, \ldots, m_n] = \mathsf{RSEnc}(M, n, t)$ be the RS encoding

of M. Then, to node i, the broadcaster sends the message $\langle \texttt{PROPOSE}, m_i \rangle$ (line 5). Note that due to properties RS code, each symbol has size $|M|/(t+1)$, and therefore the cost of the broadcaster is reduced to $O(n \cdot (|M|/(t+1)+\kappa)) = O(|M|+\kappa n)$ where κ is the size of the hash digest.

Next, each node i upon receiving the $\langle \texttt{PROPOSE}, m_i \rangle$ message from the broadcaster sends the $\langle \texttt{SHARE}, m_i \rangle$ to all nodes (line 7-8). When a node receives a SHARE message from other nodes, it adds the corresponding symbol to the set T. Once enough symbols are collected, nodes use the Online Error Correcting (OEC) algorithm (line 11) to decode the message. As described in 2, intuitively, the OEC algorithm performs up to t trials of reconstruction, and during the r-th trial, a node uses $2t + r + 1$ symbols to decode. If the reconstructed message M' has the matches with at least $2t + 1$ tuples in T, a node successfully reconstructs the message; otherwise, it waits for one more symbol and tries again.

Once a node successfully reconstructs the message M', the rest of the protocol is similar to the four-round RBC of Das, Xiang and Ren [7, Algorithm 4]. Similar to Das et al., we also add an *external predicate* $P(\cdot)$ (line 13) to strengthen the validity guarantee of BalBRB, so that any honest node only output M such that $P(M) = true$. This external validity check is useful for many application of RBC, including verifiable secret sharing [7] and AVID. In fact, our AVID protocol in §3 will use such RBC with external validity check. For standard RBC, $P(\cdot)$ always returns true.

Briefly, after checking the predicate $P(\cdot)$, nodes send ECHO messages with their symbols and the hash digest to all nodes (line 15). Also, nodes send READY messages once $2t + 1$ matching ECHO messages are collected (line 16-17) or upon receiving $t + 1$ READY messages (line 18). Note that each node needs to wait for $t + 1$ matching ECHO messages to learn the symbol to be attached in the READY message (line 19-20). Finally, nodes use the OEC algorithm to reconstruct the broadcaster's message once receiving enough READY messages of the same hash digest.

We next analyze the properties of BalBRB protocol and its performance.

Lemma 3. *Assuming a collision resistant hash function, if an honest node sends* $\langle \texttt{READY}, m_i, h \rangle$ *where* $h = \mathsf{hash}(M)$*, then* m_i *is the* i^{th} *symbol of* $\mathsf{RSEnc}(M, n, t+1)$*, and furthermore, no honest node sends a* READY *message for a different hash* $h' \neq h$*.*

Proof. First, no two honest nodes send READY messages for different hash digests, due to quorum intersection of the ECHO messages same as the Bracha's RBC. Now we show if an honest node sends $\langle \texttt{READY}, m_i, h \rangle$ where $h = \mathsf{hash}(M)$, then m_i is the i^{th} symbol of $\mathsf{RSEnc}(M, n, t+1)$. Note that an honest node i sends $\langle \texttt{READY}, m_i, h \rangle$ for $h = \mathsf{hash}(M)$ only upon receiving at least $t + 1$ matching $\langle \texttt{ECHO}, m_i, h \rangle$. At least one of these ECHO message is from an honest node h. Before the honest node h sends the ECHO message, it successfully reconstructed the message M' whose hash digest equals h. Then, by the collision resistance property of the underlying hash function, $M' = M$ and m_i is the i^{th} symbol of $\mathsf{RSEnc}(M, n, t+1)$. □

Lemma 4. *If an honest node* i *receives* $t + 1$ READY *messages with a matching hash* h*, then node* i *will eventually receive* $t + 1$ *matching* $\langle \texttt{ECHO}, m_i, h \rangle$ *messages and hence send* $\langle \texttt{READY}, m_i, h \rangle$*.*

Proof. Let j be the first honest node that sends $\langle \texttt{READY}, *, h \rangle$ message to all. Then, node j must have received at least $2t + 1$ ECHO messages with matching h, among which at least $t + 1$ are from honest nodes. Hence, node i will eventually receive $t + 1$ $\langle \texttt{ECHO}, m_i, h \rangle$ messages from these honest nodes. □

Theorem 6 (Totality and Agreement)**.** *If an honest node outputs a message, then every honest node eventually outputs a message. If an honest node outputs a message* M' *and another honest node outputs* M''*, then* $M' = M''$*.*

Proof. An honest node outputs a message M only upon receiving at least $2t + 1$ READY messages with a matching hash $h = \mathsf{hash}(M)$. At least $t + 1$ of them are sent by an honest node. Hence, all honest nodes will receive at least $t + 1$

READY messages with hash h. By lemma 4, eventually all honest nodes will send READY messages with hash h. Hence, all honest nodes will receive READY messages from all other honest nodes. Furthermore, due to Lemma 3, all these READY message contain correct symbols from the codeword $\mathsf{RSEnc}(M, n, t+1)$. Thus, every honest node will eventually output M such that $h = \mathsf{hash}(M)$. □

Theorem 7 (Validity)**.** *If the broadcaster node is honest, has an input M, and $P(M) = true$, then all honest nodes eventually output the message M.*

Proof. When the broadcaster is honest and has input M, it sends the correct symbols and hash to all nodes. Then, all honest nodes send the SHARE messages with the correct symbols. Thus, after receiving all SHARE message from honest nodes, any honest node can reconstruct M due to OEC and collision resistance of the hash. Also, the predicate $P(M) = true$ at all honest nodes, so at least $2t+1$ honest nodes will send ECHO messages with identical $h = \mathsf{hash}(M)$. Hence, all honest nodes will eventually send READY messages for h. By lemma 3 no honest node will send READY message for $h' \neq h$. As a result, all honest node will receive at least $2t+1$ READY message for h with valid symbols in it, which is sufficient to recover M. □

Next, we will analyze the communication complexity of the protocol.

Theorem 8 (Performance)**.** *Assuming existence of a collision resistant hash function whose outputs are κ bits long, Algorithm 3 solves RBC with total communication cost of $O(n|M|+\kappa n^2)$, and per-node communication cost of $O(|M|+\kappa n)$.*

Proof. In algorithm 3 the broadcaster sends a single PROPOSE to all other nodes. Moreover, each honest node sends a single SHARE, ECHO and READY message. Each message in Algorithm 3 is $O(|M|/n+\kappa)$ bits long, since $|m_i| = |M|/(t+1)$ and hash outputs are κ bits long. Hence, each node incurs a per-node communication cost of $O(|M|+n\kappa)$. Hence, the total communication cost is $O(n|M|+\kappa n^2)$. □