Succinct Erasure Coding Proof Systems

Nicolas Alhaddad
nhaddad@bu.edu

Sisi Duan
duansisi@mail.tsinghua.edu.cn

Mayank Varia
varia@bu.edu

Haibin Zhang
bchainzhang@aliyun.com

***Abstract*—Erasure coding is a key tool to reduce the space and communication overhead in fault-tolerant distributed computing. State-of-the-art distributed primitives, such as asynchronous verifiable information dispersal (AVID), reliable broadcast (RBC), multi-valued Byzantine agreement (MVBA), and atomic broadcast, all use erasure coding.**

This paper introduces an erasure coding proof (ECP) system, which allows the encoder to prove succinctly and non-interactively that an erasure-coded fragment is consistent with a constant-sized commitment to the original data block. Each fragment can be verified independently of the other fragments. Our proof system is based on polynomial commitments, with new batching techniques that may be of independent interest. To illustrate the benefits of our ECP system, we show how to build the first AVID protocol with optimal message complexity, word complexity, and communication complexity.

I. Introduction

Erasure coding is a fundamental building block in fault-tolerant distributed computing. Erasure coding is used in distributed systems to reduce the *space* and *communication* overheads. An (m, n) erasure code encodes a block of data into n fragments such that any $m < n$ of them can be used to reconstruct the original block.

This paper focuses on information dispersal and agreement between n participants, or *replicas*, in completely asynchronous settings with Byzantine failures. In this area, erasure coding is used in almost all state-of-the-art fault-tolerant distributed computing protocols, e.g., asynchronous verifiable information dispersal (AVID) [1,2], reliable broadcast (RBC) [1, 3], multi-valued (validated) Byzantine agreement (MVBA) [4, 5], atomic broadcast (aka Byzantine fault tolerant, or simply BFT) [6], read/write storage [7]–[9], and BFT storage [10].

Despite the popularity of using erasure coding in these systems, there does not exist an efficient, generic way to apply erasure coding as a building block in all of the above protocols. The challenge is due to the difficulty of determining if a given fragment corresponds to the original block. If the condition is not guaranteed, then reconstructing from different set of erasure-coded fragments lead to different blocks. (Note this is not a problem for replication-based distributed systems, as one can easily use majority voting to discover faulty fragments.)

Instead, various customized approaches have been introduced to tackle this problem. Consider the case for AVID consisting of a dispersal protocol and a retrieval protocol. In the dispersal protocol, a client disperses a block among a set of replicas and eventually replicas will receive a consistent erasure coded fragment. In the retrieval protocol, a client can retrieve fragments from replicas and reconstruct the original block. The AVID protocol of Cachin and Tessaro (CT) [1] generates for each fragment a Merkle tree. The fragment verification, however, is interactive and expensive. CT AVID uses an *all-to-all* broadcast of *all* fragments and to verify a single fragment, it needs each replica to recover the *full* block and verify the consistency of *all* fragments. The HGR AVID protocol due to Hendricks, Ganger, and Reiter [2] improves the approach of Cachin and Tessaro by building fingerprinted cross-checksum, but the checksum includes $O(n)$ cryptographic elements for each fragment. The overhead incurs a large word complexity [11][1] and would easily dominate for a large n.

A. Our Contributions

In this work, we make several contributions about erasure coding and its application to distributed computing protocols.

Erasure coding proof system. First, we define and construct a general-purpose erasure coding proof (ECP) system from any linear erasure code. Consider an (m, n) erasure code that encodes a message M into a set of n fragments $d_1, d_2, \ldots, d_n$. Our proof system is designed to allow for efficient dispersal of these fragments. A proof contains two parts: a constant-sized commitment c plus a per-replica witness π_i that is about as long as d_i (namely, about $|M|/m$). Together, c and π_i convince replica i that d_i is the correct data fragment for the message committed to by c. That is, reconstruction from any subset of m valid fragments corresponding to the same commitment c would lead to the same original message M.

We contribute two instantiations of ECP systems in this work: ECP-1 provides optimal proof size but requires trusted setup, whereas ECP-2 avoids the need for trusted setup at the expense of a logarithmic overhead in proof size. In order to achieve an optimal proof size for arbitrarily-large messages, we also contribute a new batching technique for polynomial commitment schemes that may be of independent interest.

While ECP systems do not strive to achieve agreement on the large message M, they do ensure that it suffices to achieve consensus over the short commitment c. For this reason, ECP systems are a valuable building block in distributed protocols.

Novel constructions based on ECP systems. With an ECP system, we can recast and improve almost all Byzantine fault-tolerant distributed computing protocols that involve bulk data, or alternatively, improve all protocols using erasure coding in

[1]A word is a constant number of domain values, signatures, and hashes, etc.

protocol	trusted setup	dispersal communication	dispersal word	retrieval communication	retrieval word
CT AVID [1]	no	$O(Ln + \lambda n^2 \log n)$	$O(n^2 \log n)$	$O(L + \lambda n \log n)$	$O(n \log n)$
HGR AVID [2]	no	$O(L + \lambda n^3)$	$O(n^3)$	$O(L + \lambda n^2)$	$O(n^2)$
Our AVID-1	yes	$\boldsymbol{O(L + \lambda n^2)}$	$\boldsymbol{O(n^2)}$	$\boldsymbol{O(L + \lambda n)}$	$\boldsymbol{O(n)}$
Our AVID-2	no	$\boldsymbol{O(L + \lambda n^2)}$	$\boldsymbol{O(n^2)}$	$O(L + \lambda n \log n)$	$O(n \log n)$

TABLE I: Comparison of AVID constructions. L is the input length and λ is the security parameter.

the presence of Byzantine failures, either asymptotically or concretely.

To illustrate the benefits using ECP, we provide two new asynchronous verifiable information dispersal (AVID) protocols from our ECP constructions: AVID-1 and AVID-2. AVID-1 is the first AVID protocol that has optimal word complexity for both the dispersal and retrieval protocols. AVID-1 outperforms CT AVID and HGR AVID consistently in terms of all known complexity measures. AVID-2 is strictly better than HGR AVID in all aspects; AVID-2 outperforms CT AVID for the dispersal protocol, while sharing the same complexities as CT AVID for the retrieval protocol.

Discussion on trusted setup. We provide two instantiations for our ECP system, one of which relies on trusted setup. One could, however, run a distributed protocol to avoid the trusted setup. In fact, almost all efficient, high-level asynchronous fault-tolerant protocols use trusted setup, as these protocols would rely on threshold common-coin protocols for common coins or use threshold signatures to reduce the communication and authenticator complexities (e.g., HotStuff [12], SBFT [13], all MVBA protocols except [14], and all known completely asynchronous BFT protocols implemented except [15]).

Our contributions. In summary, this paper makes the following three contributions.

- We introduce a formal notion of *erasure coding proof* (ECP) systems. We provide a new batching technique for additively homomorphic polynomial commitments, which we use to instantiate two efficient ECP constructions.
- We leverage ECP to recast asynchronous Byzantine resilient protocols with asymptotically and concretely improved efficiency. For instance, we show how to use ECP to build asymptotically better AVID protocols (see Table I).

II. System Model and Problem Statement

We consider distributed computing protocols consisting of n replicas, where f out of them replicas may fail arbitrarily (Byzantine failures). Furthermore, we assume the existence of secure point-to-point channels between each pair of replicas. We consider completely asynchronous systems making no timing assumptions on message processing or transmission delays.

This paper considers adaptive corruption, where the adversary can choose its set of corrupted replicas at any moment during the execution of the protocol, based on the information it accumulated thus far. The static corruption is weaker, as the adversary is restricted to choose its set of corrupted replicas at the start of the protocol and cannot change this set later on. All protocols we consider assume that f is a constant fraction of n with $f \leq \lfloor \frac{n-1}{3} \rfloor$, which is optimal. A (Byzantine) *quorum* is a set of $\lceil \frac{n+f+1}{2} \rceil$ replicas. Without loss of generality, this paper may assume $n = 3f + 1$ and a quorum size of $2f + 1$.

Asynchronous verifiable information dispersal (AVID). AVID is introduced by Cachin and Tessaro [1] and used to disperse a data block among a set of replicas.

AVID is the most fundamental (and arguably the most simple) primitive in reliable distributed computing. AVID can be used to build other primitives such as RBC, MVBA, and atomic broadcast.

An AVID scheme consists of a dispersal protocol and a retrieval protocol. The dispersal protocol is specified by *avid-disperse* and *avid-deliver*. A client (may it be a replica) may *avid-disperse* (id, M), and replicas complete the dispersal protocol and *avid-deliver* M for id. The retrieval protocol is defined by *avid-retrieve* and *avid-output*. In the retrieval protocol, a client ct may trigger *avid-retrieve* and eventually *avid-output* the full block M.

An AVID scheme with tag id should satisfy the following with overwhelming probability:

- **Termination**: If *avid-disperse* for id is initiated by a correct client, then *avid-disperse* for id is eventually completed by all correct replicas.
- **Agreement**: If a correct replica completes *avid-disperse* for id, then all correct replicas eventually complete *avid-disperse* for id.
- **Availability**: If $f+1$ correct replicas complete *avid-disperse* for id, then a correct client that initiates *avid-retrieve* for id eventually reconstructs some block B'.
- **Correctness**: After $f + 1$ correct replicas complete *avid-disperse* for id, all correct clients that initiate *avid-retrieve* for id eventually retrieve the same block B'. If the client that initiated *avid-disperse* for id was correct, then $B' = B$.

AVID can be used to build, e.g., reliable broadcast protocol [1], BFT protocols [16], and BFT storage [10].

III. Review of Existing AVID Constructions and Obstacles

We use AVID as an example to illustrate the challenge of using erasure coding. In particular, we discuss existing techniques for AVID and their "limitations" (i.e., why they are not "ideal") and then motivates the design of an "ideal" ECP system.

A. Challenge of Using Erasure Coding

Let us consider an erasure coding scheme with $n = 4$ and $f = 1$. Given a block m, one could apply a (2, 4) erasure coding scheme to generate four fragments m_1, m_2, m_3, and m_4. The system can allow us to tolerate two benign failures, because any two fragments can be used to reconstruct the original block m. Without loss of generality, we consider a systematic erasure coding, where m_1 and m_2 are data fragments and $m = (m_1, m_2)$, while m_3 and m_4 are redundant fragments. Crucially, m_1, m_2, m_3, and m_4 are "correlated:" fixing an erasure coding scheme, m_3 and m_4 are generated according to the specific generator matrix using m_1 and m_2 as input. If we consider linear erasure coding, m_3 and m_4 are simply a linear combination of m_1 and m_2 with the columns (or rows) of the generator matrix as coefficients.

To disperse m reliably in AVID with n servers, the most *intuitive and efficient* way is to send erasure-coded fragments to individual replicas. However, erasure coding creates a fundamental (and unique) challenge. That is, in erasure-coded systems, one must determine if a given fragment, for instance, m_3, corresponds to the original data block m. If this is not guaranteed, then reconstructing from different subsets of fragments may lead to different data blocks. For instance, if a replica p_3 provides an incorrect share m'_3 (that is not a linear combination of m_1 and m_2), the block reconstructed from m_1 and m_2 is different from the block from m'_3 and m_4.

Note this is not at all an issue for replication-based fault-tolerated distributed systems, as assuming a minority of faulty servers, one can use, for instance, majority voting to identify inconsistent data from faulty servers.

B. CT AVID using Cross-Checksum and Merkle Tree

CT AVID has two AVID protocols: AVID based on cross-checksum [17, 18] (a list of n hash functions) and more efficient AVID based on Merkle tree. Both AVID protocols follow the three-step communication pattern of Bracha's broadcast [19].

To write a block m of size L, a client applies an $(f+1, n)$ erasure coding scheme to generate n fragments (of size $\frac{L}{f+1}$) and computes the hash of each fragment, forming a cross-checksum with n hashes. In the SEND phase, the client sends each replica its fragment and the cross-checksum. In the ECHO phase, each replica echoes the its fragment and cross-checksum to all replicas. If receiving $2f + 1$ fragments and matching cross-checksum, a replica first decodes the original block, encodes the block *again* to generate all n fragments, computes the hashes of n fragments, and finally verifies if all hashes matches the cross-checksum. In the READY stage, each replica broadcasts the cross-checksum and its fragment to all replicas. As in Bracha's broadcast, if a correct replica receives $f+1$ READY messages, it broadcasts the cross-checksum and its fragment to all replicas.

The more efficient AVID construction proposed by Cachin and Tessaro is simply to replace cross-checksum with Merkle tree. Each message sent now carries a $O(\log n)$ hashes instead of n hashes.

For both protocols, the complexities are dominated by the all-to-all broadcast phases, namely ECHO and READY phases. Note $O(n) = O(3f + 1) = O(f)$. The communication complexity for them is $O(n^2(\frac{L}{f+1} + \lambda n)) = O(nL + n^2\lambda)$ and $O(n^2(\frac{L}{f+1} + \lambda \log n)) = O(nL + n \log n\lambda)$, respectively. Above, λ is the security parameter and here denotes the output length of the hash function. The word complexity for the two schemes are $O(n^3)$ and $O(n^2 \log n)$, respectively.

C. Using Vector Commitment

We observe that one could extend the Merkle tree based AVID protocol of Cachin and Tessaro for a more general AVID using vector commitments [20], as Merkle tree can be viewed a special case of vector commitment. Assuming trusted setup, there exists vector commitment with a constant-size commitment and a constant-size proof [20]. If directly using the constant-size vector commitment, one would obtain an AVID protocol with slightly better communication complexity of $O(n^2(\frac{L}{f+1} + \lambda)) = O(nL + n\lambda)$. The resulting protocol, however, needs interactive verification of erasure-coded fragment, just as Merkle tree based AVID. Hence, the protocol needs all-to-all communication of fragments.

D. HGR AVID using FPCC

Apparently, cross-checksum, Merkle tree, and vector commitment are generic approaches to compressing any data blocks. They are not specifically invented for erasure coding, and the verification is non-interactive. Realizing the problem, Hendricks, Ganger, and Reiter (HGR) introduce fingerprinted cross-checksum (fpcc) that allows each replica to independently verify that its fragment was generated from the original block. As in CT AVID, HGR AVID needs to associate a fragment with a proof. In HGR AVID, each proof for a fragment (which is fpcc) now contains m universal hashes of data fragments and n hashes of all fragments. Intuitively, the n hashes serves as a commitment for all fragments, and m universal hashes preserve the linearity of data fragments. The independent verification property in HGR AVID makes it possible to send bulk erasure-coded fragment only in the SEND phase (which involves just a one-to-all communication). In contrast, the ECHO and READY phases only need to agree on the fpcc. This approach avoids all-to-all communication of bulk erasure-coded fragments. Since fpcc itself contains $m+n$ words, HGR AVID has significantly larger word complexity of $O(n^3)$. The corresponding communication complexity is of $O(L + \lambda n^3)$. When n becomes large or the input size L is small, the λ term easily dominates the communication.

E. What's Desired for an Ideal Erasure-Coded System?

So far, we argue that none of the existing AVID protocols and techniques seem ideal to build an AVID protocol, or more generally, erasure-coded fault-tolerant distributed systems.

To directly solve the problem, we desire a proof system specifically ensuring the consistency of the erasure-coded fragment. Namely, such a proof must be able to prove that an erasure-coded fragment algebraically corresponds to some

original block. Moreover, the proof should be independent of n or λ. Moreover, we desire the proof to be publicly verifiable such that the recipient of the proof can verify its correctness without using any pre-shared secret and the proof is transferable.

IV. Erasure Coding Proof System

In this section, we define and construct a general erasure coding proof system based on any linear erasure code and for any additively homomorphic polynomial commitment. We provide two instantiations based on the Reed-Solomon code, and we compare the difference in proof size for state-of-the-art transparent vs non-transparent polynomial commitments.

A. Building Blocks

We begin by describing the erasure coding schemes upon which we build proof system, plus the polynomial commitments and hash functions that we will use in our construction.

Erasure coding scheme. An (m, n) erasure coding scheme over an alphabet $\mathcal{M}$ is a pair of algorithms $(\textit{encode}, \textit{decode})$, where $\textit{encode} : \mathcal{M}^m \to \mathcal{M}^n$ and $\textit{decode} : \mathcal{M}^m \to \mathcal{M}^m$. The *encode* algorithm takes as input a data block, consisting of m data fragments, and outputs $n > m$ coded fragments. The *decode* algorithm takes as input any m-size subset of coded fragments and outputs the original data block containing m data fragments. Namely, if $[d_1, \ldots, d_n] \leftarrow \textit{encode}(M)$, then $\textit{decode}(d_{i_1}, \ldots, d_{i_m}) = M$ for any distinct $i_1, \ldots, i_m \in [1..n]$. A (m, n) erasure coding scheme is *linear* if each coded fragment d_i ($i \in [1..n]$) is a linear combination of the first m data fragments, i.e., $d_i = \sum_{j=1}^{m} b_{ij} d_j$, where b_{ij}'s are coding coefficients. The coding coefficients form a *generator matrix* for the linear code. An (n, m) erasure coding scheme is *systematic*, if the first m coded fragments are the original m data fragments.

Reed-Solomon code. A (m, n) Reed-Solomon code is an (m, n) linear erasure code whose generator matrix is the transpose of the Vandermonde matrix. We present an alternative but equivalent way to describe the *encode* and *decode* algorithms that highlights the relationship between the Reed-Solomon code and polynomial commitments. The *encode* algorithm takes m data fragments and uses those as m coefficients to produce a polynomial P of degree $m - 1$. The *encode* algorithm produces n points by evaluating f on n different evaluation points. The *decode* algorithm takes any m points, interpolates P and produces the original m fragments.

Polynomial commitments. In this work, we use *deterministic* homomorphic polynomial commitments following the definitions used by Haven [21]. Specifically, this commitment scheme contains seven algorithms (Setup, Com, Eval, VerifyEval, Open, VerifyOpen, Hom) such that:

- $\mathrm{Setup}(1^\lambda, D) \to \mathsf{pp}$ takes a security parameter λ and an upper bound D on the degree of any polynomial to be committed, and generates public parameters pp.
- $\mathrm{Com}_{\mathsf{pp}}(P(x), d) \to \hat{P}$ is given a polynomial $P(x)$ of degree $d \leq D$. It outputs a commitment string $\hat{P}$.
- $\mathrm{Eval}_{\mathsf{pp}}(P, i) \to (P(i), w_i)$ is given a polynomial P as well as an index i. It outputs the evaluation $P(i)$ and a corresponding witness string w_i.
- $\mathrm{VerifyEval}_{\mathsf{pp}}(\hat{P}, i, P(i), w_i, d) \to$ True/False is given a commitment $\hat{P}$ as well as the inputs and outputs of Eval, and it outputs a Boolean.
- $\mathrm{Open}_{\mathsf{pp}}(P, i) \to (P_i, \bar{w}_i)$ is given a polynomial P as well as an index i. It outputs the i^{th} coefficient P_i and a corresponding witness string $\bar{w}_i$.
- $\mathrm{VerifyOpen}_{\mathsf{pp}}(\hat{P}, i, P_i, \bar{w}_i) \to$ True/False determines whether P_i is the i^{th} coefficient of the polynomial whose commitment is $\hat{P}$.
- $\mathrm{Hom}_{\mathsf{pp}}(a, \hat{P}_1, \hat{P}_2) \to \widehat{P^*}$ receives a scalar a and commitments to two polynomials P_1 and P_2 of degrees d_1 and d_2. It outputs a commitment $\mathrm{Com}_{\mathsf{pp}}(P^*, \max\{d_1, d_2\})$ to the polynomial $P^* = aP_1 + P_2$.

Polynomial commitment schemes satisfy three properties for any polynomial P and corresponding commitment $\hat{P}$.

- Correctness states that the witness produced by Eval is successfully verified by VerifyEval if $\deg P \leq D$.
- Evaluation binding states that for each input i, there exists only one opening $P(i)$ that can be verified.
- Degree polynomial says that if $\deg P > D$ then it is computationally infeasible to find any point $(i, P(i))$ that causes VerifyEval to succeed.

All of these properties are required both for commitments generated directly from Com and those produced homomorphically using Hom. We remark that this work only considers non-hiding commitments.

Hash. We also use a collision-resistant hash function $hash$ mapping a message of arbitrary length to a fixed-length output.

B. Defining ECP systems

In this section, we formally define an erasure coding proof (ECP) system based on an erasure coding scheme. We also describe its efficiency goals and security properties.

Syntax of an ECP system. An (m, n) ECP system, with an associated (m, n) erasure coding scheme (*encode*, *decode*), consists of three algorithms ($\textit{ecsetup}$, $\textit{ecprove}_{\mathsf{pp}}$, $\textit{ecverify}_{\mathsf{pp}}$).

1) The *ecsetup* algorithm is optional. It receives a security parameter λ and sets up the system parameters pp.
2) The $\textit{ecprove}_{\mathsf{pp}}$ algorithm takes as input a block of data M and outputs $(c, \boldsymbol{d}, \boldsymbol{\pi})$ where $|\boldsymbol{d}| = |\boldsymbol{\pi}| = n$. Here, c is a binding commitment to all erasure-coded fragments $\boldsymbol{d} \leftarrow \textit{encode}(M)$, and each π_i is intended to serve as a proof that the corresponding d_i is the i^{th} data fragment with respect to the commitment c.
3) The $\textit{ecverify}_{\mathsf{pp}}$ algorithm takes as input (c, d_i, π_i) and outputs a bit. If $\textit{ecverify}_{\mathsf{pp}}(c, d_i, \pi_i) = 1$, then we say d_i is a valid fragment with respect to c.

By abuse of notation, we sometimes omit the parameter pp when no ambiguity arises in describing our algorithms.

Efficiency goals. For efficiency, we desire ECP constructions whose commitment c is constant size — more precisely, $O(\lambda)$

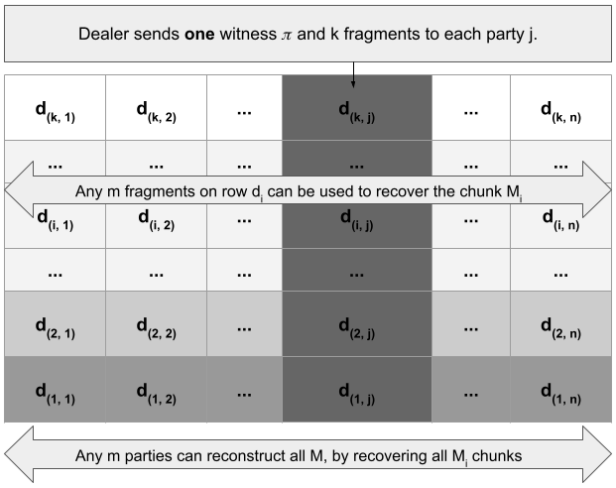

Fig. 1: Dealer distributing fragments of a message M

but independent of $|M|$ — while still being sufficient to verify the (d_i, π_i) fragments sent to each of the n replicas. We also desire that $|\pi_i|$ be equal or close to $|d_i| = |M|/m$ so that the proofs have low communication overhead. In particular, we require that the proof size not scale multiplicatively in the number of replicas n. Finally, we levy no restrictions on who may run the *ecverify*$_{\mathsf{pp}}$ algorithm; the algorithm is publicly verifiable.

Definitions of security. An ECP system (*ecsetup*, *ecprove*$_{\mathsf{pp}}$, *ecverify*$_{\mathsf{pp}}$) is secure if the following properties hold with overwhelming probability:

1) EC correctness: If a correct encoder runs *ecprove*$_{\mathsf{pp}}$ (M, pp) and obtains $(c, \boldsymbol{d}, \boldsymbol{\pi})$ and a correct decoder reconstructs M' from a set of m valid fragments (d_i, π_i) with respect to c, then $M = M'$.
2) EC consistency: If a correct decoder reconstructs M_1 from a set of m valid fragments with respect to c and another correct decoder reconstructs M_2 from a set of valid fragments with respect to c, then $M_1 = M_2$.

Both properties defined above, EC correctness and EC consistency, are intuitive. The former requires if an encoder correctly generates proofs for all fragments, then reconstructing from any m valid fragments will lead to the original block. The latter ensures reconstructing from different subsets of valid fragments will lead to the same block, even if the encoder is faulty.

C. Overview of our construction

In this section, we build up in stages the ideas for an efficient ECP scheme. Our construction is based upon any (m, n) systematic linear erasure coding scheme whose fragments are of size equal to the security parameter, i.e. $\lambda = \lceil \log |\mathcal{M}| \rceil$.

Warm-up: fixed-length messages. We begin with a warm-up exercise about constructing an erasure coded proof system for messages M of a fixed size $|M| = m\lambda$. As a result, the message can be partitioned into m fragments $M = d_1 \parallel d_2 \parallel \cdots \parallel d_m$, where $\parallel$ denotes concatenation. These fragments can be encoded into $d_1, \cdots, d_n \leftarrow \mathit{encode}(M)$ due to the systematic property.

The dealer wants to produce n witnesses, each of which will convince one of n verifiers $p_1, \ldots, p_n$ that one of the corresponding fragments $d_1, \cdots, d_n$ has been generated correctly. Specifically, the dealer needs to prove that $d_i = \sum_{j=1}^{m} b_{ij} d_j$ for each i, where b_{ij}'s are public coding coefficients specific to the linear erasure coding scheme used. Written differently, $d_i = \langle \boldsymbol{v}, \boldsymbol{b^i} \rangle$ is the inner product of $\boldsymbol{v} = [d_1, \cdots, d_m]$ and $\boldsymbol{b^i}$, the i^{th} column in the public matrix of coefficients.

Notice that $\boldsymbol{v}$, the vector encoding the message, is the same for all n verifiers. So, one way for the prover to convince the verifier about the value of the inner product of $\boldsymbol{v}$ with a public vector $\boldsymbol{b^i}$ is for the prover to send $\boldsymbol{v}$ to every verifier, who can then compute the inner product herself. But this approach is inefficient in communication because $|\boldsymbol{v}| = |M|$.

A better approach is for the prover to form a succinct (constant or log-sized) commitment to the vector $\boldsymbol{v}$ in a manner that allows for succinct proofs of inner products. The DARK transformation by Bünz et al. [22] provides a way to do so, based on polynomial commitments. The transformation begins by creating new polynomials $V(x)$ and $B^i(x)$ whose coefficients are equal to the elements of the vectors $\boldsymbol{v}$ and $\boldsymbol{b^i}$, respectively. Then, the inner product of the two length-m vectors $\langle \boldsymbol{v}, \boldsymbol{b^i} \rangle$ is equal to the m^{th} coefficient of the polynomial $Z(x) = V(x) * B^i(x)$. This property can be proved after providing the verifier with the polynomial commitment $\hat{V}$, the m^{th} coefficient and the constant size witness $\bar{w}_m$. Note that B^i doesn't have to be sent to the verifier because it's public. The resulting proof of correctness of the inner product achieves our efficiency goals: it only requires a constant overhead over the witness size of the underlying polynomial commitment scheme.

Scaling to variable-length messages. This proof that d_i has been generated correctly does work for a fixed-size message M, but it relied crucially on the fact that M fits precisely within the vector $\boldsymbol{v}$. As a result, it is tricky to scale this proof to handle larger messages, say of size $|M| = km\lambda$ that are a factor k larger than in our warm-up example because there is no place to store M within the current vector $\boldsymbol{v}$ for a fixed fragment alphabet λ bits.

To support large messages, we must either: (1) switch to an erasure coding scheme with a larger fragment size $k\lambda$, (2) create a longer vector $\boldsymbol{v}$ of length km, or (3) partition M across k vectors that are each of length m. We consider each approach in turn, and we will ultimately adopt the last one.

Approach 1: We could increase the alphabet $\mathcal{M}$ for the erasure coding scheme so that its size grows linearly with k. With this change, all of the analysis from the warm-up example would continue to hold. However, efficiency would be ruined: the commitments and proofs of the resulting scheme would now also scale linearly in the message length.

Approach 2: Alternatively, we could partition the long message M into a single vector $\boldsymbol{v}$ of length km. This approach

requires a different change to the erasure code: each fragment remains of length λ as in the warm-up, but we now require a (km, kn) erasure code in order to encode $\boldsymbol{v}$. With this change, the technique from the warm-up would produce a valid proof, but our efficiency is harmed in a different way. Committing to an unbounded-length $\boldsymbol{v}$ makes it difficult to use polynomial commitment schemes that require (bounded) trusted setup [23]. Also, the length and proving time for each witness grows with the size of M, making the scheme less practical.

Approach 3: Finally, the approach we actually take in this work is for the dealer to break M into multiple messages $M_1 \dots M_k$, each of size $m\lambda$ as in the warm-up example. This approach requires no change to the erasure coding scheme itself. The dealer encodes each message into a separate vector $\boldsymbol{v^1}, \dots, \boldsymbol{v^k}$, as shown in the rows of Figure 1. Each verifier j receives a fragment containing k field elements, as shown in the columns of Figure 1. We stress that k only depends on the message size $|M|$ and is independent of n.

It remains only to design a new method to prove that each fragment has been generated correctly, based upon all k polynomial commitments $\hat{V}^1, \dots, \hat{V}^k$. If done naively, this would result in each verifier receiving k witnesses. By exploiting additively homomorphic polynomial commitments, we show in the next section how to reduce the cost to a single witness.

D. Succinct Multi-Point Proofs

Consider a set of n polynomial commitments $\{\hat{P}_1 \dots \hat{P}_n\}$ and a single evaluation point j. If a prover wants to prove that $P_i(j) = y_i$ for all $i \in \{1 \dots n\}$, then naively the prover has to send n witnesses, one for each $(y_i, \hat{P}_i)$. In this section, we construct a generic, succinct proof that exploits the additive homomorphism of the polynomial commitment scheme so that a verifier can be convinced using only one witness.

We first explain the scheme in terms of a public coin interactive argument system between a prover and a verifier. The scheme contains three steps: 1) a commitment 2) a public coin challenge and 3) a response.

1) Commitment: prover commits to n different polynomials $\{P_1 \dots P_n\}$ with the same degree d. For each $i \in \{1, \dots, n\}$, the prover runs $\hat{P}_i = \mathrm{Com}_{\mathsf{pp}}(P_i, d)$ and sends the pair $(P_i(j), \hat{P}_i)$.
2) Challenge: verifier generates a random point $c \in \mathbb{F}$
3) Response: prover interpolates P_c from $\{P_1 \dots P_n\}$ and sends the witness w after running $\mathrm{Eval}_{\mathsf{pp}}(\hat{P}_c, j)$

The verifier computes $P_c(j)$ by interpolating all $P_i(j)$ where $i \in \{1 \dots n\}$. The verifier then computes $\hat{P}_c = \sum_{i=1}^{n} \hat{P}_i * l_i(c)$ where l_i is the Lagrange basis polynomial. The verifier can compute $\hat{P}_c$ because of the addititive homomorphic property of the polynomial commitment used. The Verifier accepts, if $\mathrm{VerifyEval}_{\mathsf{pp}}(\hat{P}_c, (j, P_c(j), w), d)$ returns True and rejects otherwise.

The completeness of this protocol follows from Lagrange interpolation and the correctness of the underlying polynomial commitment scheme Π_{pc}. We call this scheme $\mathrm{BatchProof}_{\mathsf{pp}}$ using the same public parameters pp as the underlying polynomial commitment scheme, and we make it non-interactive by using the Fiat-Shamir heuristic.

Theorem 1. *The $\mathrm{BatchProof}_{\mathsf{pp}}$ protocol has a negligible soundness error of at most $\frac{n-1}{|\mathbb{F}|} + n \cdot Adv^{evalBind}_{polycom}$.*

Proof. Consider a statement consisting of the n points $\{y_1, \dots, y_n\}$, n polynomial commitments $\{\hat{P}_1 \dots \hat{P}_n\}$, and a single evaluation point j. This statement also serves as the contents of step 1 of the interactive $\mathrm{BatchProof}_{\mathsf{pp}}$ protocol.

We observe that if the individual polynomial commitments would have been arguments of knowledge, then this claim would be straightforward to prove. One could extract the polynomials $\{P_1, \dots, P_n\}$ from the underlying commitments, compute the polynomials corresponding to both the claimed y_i points and the real $P_i(j)$ evaluations, and use the Schwartz-Zippel lemma to argue that these polynomials are unlikely to intersect.

Without extractable commitments, we require a more detailed reduction to learn the "real" evaluations embedded inside the prover's responses. For any prover who succeeds with noticeable probability q at its task of producing a response proof in step 3 that is successfully verified, our goal is to show how to extract n independent witnesses demonstrating whether all y_i are contained in the polynomials $\hat{P}_i$ with probability $q - \frac{n-1}{|\mathbb{F}|} + n \cdot \mathrm{Adv}^{\mathrm{evalBind}}_{\mathrm{polycom}}$, thereby proving the theorem.

Let c_1 be the challenge used in the actual received proof and $w_1 = \mathrm{Eval}_{\mathsf{pp}}(P_{c_1}, j)$ be the corresponding witness received in step 3 of the interactive protocol. Next, our reduction will rewind and rerun the prover on steps 2 and 3 until it receives $n-1$ additional challenges and witnesses of the form (c_i, w_i) for $i \in \{2, \dots, n\}$. This will take $1/q^{n-1}$ time in expectation, which is acceptable since q is noticeable.

From these n total polynomial commitments $\hat{P}_{c_i}$ and associated witnesses w_i, it is possible to compute the witnesses associated with any other row using Lagrange interpolation: $P_l(x) = \sum_{i=1}^{n} P_{c_i}(x) * l_i(l)$. In particular, it is possible to compute the evaluations and associated witnesses for each of the original polynomials $P_1, \dots, P_n$ at evaluation point j.

Next, we compare these evaluations $P_i(j)$ with the corresponding claimed evaluations y_i from the prover's original statement. If all of them are equal, then the witnesses $\{w_i\}_{i \in [1..n]}$ demonstrate that the statement is true, up to the evaluation binding error of these n witnesses.

However if there exists at least one index i^* such that $P_{i^*}(j) \neq y_{i^*}$, then the statement is false and must be caught in order for the $\mathrm{BatchProof}_{\mathsf{pp}}$ to be sound. In this case, we compute the probability that the prover broke soundness and managed to provide a challenge witness that was verified.

Let Y denote the vertical polynomial formed by interpolating the claimed points (i, y_i), and let E denote the vertical polynomial formed by interpolating the known evaluations $(c_i, P_{c_i}(j))$. We know that $E \neq Y$ because these polynomials disagree at the point i^*. If these polynomials also disagree at the challenge point c_1, then the prover has provided witnesses that $P_{c_1}(j)$ equals both $E(c_1)$ and $Y(c_1)$ and thus broken

evaluation binding. (For the remaining $n-1$ queries to the prover, we obtained witnesses at $E(c_i)$ but not $Y(c_i)$ so there is no issue with evaluation binding.) Finally, the polynomials E and Y of degree $n-1$ intersect at the randomly-chosen challenge point with probability $\frac{n-1}{|\mathbb{F}|}$ by the Schwartz-Zippel lemma. By a union bound, the prover succeeds with probability at most $\frac{n-1}{|\mathbb{F}|} + n \cdot \text{Adv}^{\text{evalBind}}_{\text{polycom}}$ as desired. □

E. Our ECP construction

Our ECP construction in Algorithm 1 is built from a $\text{BatchProof}_{\mathsf{pp}}$ scheme, which in turn requires an additively homomorphic polynomial commitment scheme Π_{pc}. We use Reed-Solomon erasure coding and polynomial commitments, so that the proof of correctness of each fragment reduces to just a polynomial evaluation (see definition: IV-A). As such, we drop the need for the DARK transformation [22].

- *ecsetup*: Initializes the polynomial commitment and sets the maximum degree of the polynomial commitment to be $m-1$.
- *ecprove*$_{\mathsf{pp}}$: Takes the messsage M and breaks it into k polynomials $P(1)\dots P_k$ each of degree $m-1$. For each fragment d_i consisting of k points $P_1(i)\dots P_k(i)$, *ecprove*$_{\mathsf{pp}}$ batches a proof π_i by calling $\text{BatchProof}_{\mathsf{pp}}(P_1(i)\dots P_k(i), i)$. Since each π_i contains the same k polynomial commitments $\hat{P}_1 \dots \hat{P}_k$, the construction picks arbitrarily π_1 and extracts them from it. The polynomial commitments are then hashed to produce c. c is a constant size commitment with respect to the security parameter λ. The method returns the ordered list of fragments $\boldsymbol{d}$ and their corresponding ordered lists of proofs and c the hash.
- *ecverify*$_{\mathsf{pp}}$: Takes a fragment d_i, a proof π_i and a hash c. The method first extracts the fragments and polynomial commitments from the proof pi_i and checks that they are equal to the ones provided. If both of these checks pass the $\text{BatchProof}_{\mathsf{pp}}.\text{verify}(\pi_i)$ is called to verify the proof π_i.

Theorem 2. *Given a secure polynomial commitment scheme Π_{pc}, the ECP system described in Algorithm 1 is secure.*

Proof. EC-correctness follows directly from the completeness and soundness property of the $\text{BatchProof}_{\mathsf{pp}}$ scheme. For any arbitrary message M, a correct prover can run *ecprove*$_{\mathsf{pp}}$ on M, which produces k polynomials that encode M. In Algorithm 1, *ecprove*$_{\mathsf{pp}}$ calls $\text{BatchProof}_{\mathsf{pp}}.\text{prove}$ to produce n valid proofs for n fragments with the same c. Any d_i fragment with a valid proof would imply having one point on each of k different polynomials whose hash is c. Since each polynomial is of degree $m-1$, then any m fragments with valid proofs would reconstruct the same k polynomials and therefore the same message M.

To show EC-consistency, consider any two sets of fragments s_1 and s_2 with cardinality m together with two sets of valid proofs π_1 and π_2 with respect to the same c. Validity means that all fragments of s_1 and s_2 can pass *ecverify*$_{\mathsf{pp}}$ and therefore also $\text{BatchProof}_{\mathsf{pp}}.\text{verify}$. Since c is the same for all fragments in s_1 and s_2, then by the collision resistance property of the hash function we have the same set of k polynomials being passed to $\text{BatchProof}_{\mathsf{pp}}.\text{verify}$ with overwhelming probability. By the soundness property of $\text{BatchProof}_{\mathsf{pp}}.\text{verify}$ (cf. Theorem 1), all points in s_1 and s_2 must be contained in the same set of k polynomials of degree $m-1$, which means that $\textit{decode}(s_1) = \textit{decode}(s_2)$, so consistency is satisfied. □

Algorithm 1 ECP based on a polynomial commitment scheme Π_{pc} an (m, n) Reed-Solomon code and a message M that is a multiple of m.

▷ *ecsetup* (1^λ)
  return $\Pi_{pc}.\text{Setup}(1^\lambda, m-1)$
▷ ***ecprove***$_{\mathsf{pp}}$ (M)
  $k \leftarrow \frac{M}{(m)*\lambda}$
  for $0 \le i < k$
    $P_i \leftarrow M[i*m, (i+1)*m]$ {coefficient polynomial of degree $m-1$}
  for $1 \le i < n$
    $d_i \leftarrow P_1(i)\dots P_k(i)$
    $\pi_i \leftarrow \text{BatchProof}_{\mathsf{pp}}.\text{prove}([P_1, \dots, P_k], i)$
  $c \leftarrow hash(\hat{P}_1, \dots, \hat{P}_k)$ {recall that $\hat{P}_i = \text{Com}_{\mathsf{pp}}(P_i, d)$}
  return $(c, \boldsymbol{d}, \boldsymbol{\pi})$
▷ ***ecverify***$_{\mathsf{pp}}$ (c, d_i, π_i)
  return $(c \stackrel{?}{=} hash(extract_c(\pi_i))$ and {extract commitments from π_i}
    $d_i \stackrel{?}{=} extract_{\boldsymbol{d}_i}(\pi_i)$ and {extract points from π_1 step 1}
    $\text{BatchProof}_{\mathsf{pp}}.\text{verify}(\pi_i))$

F. Two instantiations of ECP

In this work, we consider two instantiations of Algorithm 1 based on different types of polynomial commitments.

1) Our first instantiation, ECP-1, achieves better efficiency in scenarios that allow for trusted setup. Specifically, we instantiate our construction with KZG polynomial commitments [23], which have constant size commitments and witnesses. This will make the size of each π_i in ECP-1 to be $O(d_i) = O(M/n)$. Hence, the cumulative cost of sending all π_i witnesses is absorbed completely by d_i and **does not depend on either** n **or** λ.
2) ECP-2 is an alternate instantiation of Algorithm 1 based on polynomial commitment schemes with transparent setup and logarithmic-sized witnesses, such as Bulletproofs [24] or the DARK polynomial commitment scheme [22]. The size of each witness π_i in ECP-2 is $O(M/n + \lambda \log n)$, which is $O(\lambda \log n)$ larger in bits than the witnesses in ECP-1.

Notice that both instantiations have the $\pi_i = O(M/n)$ term. This is because our constructions use the $\text{BatchProof}_{\mathsf{pp}}$ scheme (cf. §IV-D), in which the prover sends $\lceil \frac{M}{m\lambda} \rceil$ commitments and each polynomial commitment is of size $O(\lambda)$.

V. AVID with Optimal Word Complexity

In this section, we present a new asynchronous verifiable information dispersal (AVID) protocol based on the (m, n) ECP proof system. We let $m = f+1$ and $n \ge m + 2f$. AVID consists of two phases: a dispersal and retrieval phase.

Algorithm 2 AVID with identifier id and sender p_k. Code is shown for replica p_i, who might or might not be the sender.

Initialization
  $\mathsf{pp} \leftarrow$ *ecsetup*(1^λ) {initialize ECP scheme (optional)}
  $fragments \leftarrow \perp$ {dictionary id $\mapsto$ d_i, π_i, c}
▷ **dispersal**
upon *avid-disperse*(id, M) and party is p_k {Step 1: send}
  $(c, \boldsymbol{d}, \boldsymbol{\pi}) \leftarrow$ *ecprove*$_{\mathsf{pp}}(M)$
  for $1 \le j \le n$
    send $(id, \text{SEND}, c, d_j, \pi_j)$ to p_j
upon receiving $(id, \text{SEND}, c, \pi_i, d_i)$ from p_k for first time {Step 2: echo}
  if *ecverify*$(c, d_i, \pi_i) \stackrel{?}{=} 1$
    $fragments[id] \leftarrow (d_i, \pi_i, c)$
    send (id, ECHO, c) to all replicas
upon receiving (id, ECHO, c) from p_j for first time {Step 3: ready}
  if not yet send (id, READY, c) and received $2f + 1$ Echo with same c
    send (id, READY, c) to all replicas
upon receiving (id, READY, c) from p_j for the first time
  if not yet send (id, READY, c) and received $f + 1$ Ready with same c
    send (id, READY, c) to all replicas {amplification step}
  if received $2f + 1$ READY with the same c
    if $fragments[id]$ contains a different commitment
      $fragments[id] \leftarrow \perp$
    avid-deliver(id)
▷ **retrieval**
upon *avid-retrieve*(id, p) {broadcast fragments}
  if $fragments[id] \stackrel{?}{\neq} \perp$
    $(d_i, \pi_i, c) \leftarrow fragments[id]$
    send $(id, \text{RETRIEVE}, d_i, \pi_i, c, i)$ to p
upon receiving $(id, \text{RETRIEVE}, d_j, \pi_j, c, j)$ {verify and decode}
  if *ecverify*$(c, d_j, \pi_j) \stackrel{?}{=} 1$
    $B.add(d_j)$
  if $len(B) \stackrel{?}{=} f + 1$
    $M \leftarrow$ *decode*(B)
    avid-output(M)

We present the protocol formally in Algorithm 2 and describe both phases below.

In the dispersal phase for a given message M with an associated ID tag of id, a correct dealer p_k will use the ECP proof system in order to generate fragments of M that are consistent with a commitment c that is also dispersed to all replicas. The protocol begins with an initialization step where every replica runs the *ecsetup* algorithm of the ECP proof system and creates an empty dictionary $fragments$ whose keys are ids of the messages. $fragments$ will hold the fragment of each message, together with its proof and commitment. Our protocol follows the three phases of Bracha's RBC protocol [19] and CT AVID protocol.

- In the SEND phase, the dealer p_k breaks the message M into fragments, their proofs and commitment vector using $(c, \boldsymbol{d}, \boldsymbol{\pi}) \leftarrow$ *ecprove*$_{\mathsf{pp}}(M)$. The dealer then broadcasts each piece c, d_i, π_i to each replica p_i.
- In the ECHO phase, every replica p_i checks that the fragment d_i is consistent with the vector c and π using *ecverify*(c, d_i, π_i). If the fragment is valid, then replica p_i saves the data and echoes c to everyone else with the goal of reaching agreement over it.
- In the READY phase, the protocol behaves similar to Bracha's reliable broadcast in the sense that a correct replica will send a READY message only if it receives $2f+1$ ECHO messages, or if $f+1$ READY messages for the same c and if the replica has not done so already.
- Finally, a correct replica delivers only if it receives $2f+1$ READY messages for the same c. If a replica receives $2f + 1$ READY messages with some c that is different from the commitment it received from the dealer in the broadcast phase, then that replica destroys its own fragment d_i because it is no longer useful to contribute toward retrieval.

The retrieval for a given message M with an associated ID tag of id provides the message to a retrieving party p. It operates as follows. If an honest party has some data for a given message tag stored in $fragments[id]$ then that party must send that data to the party with id p. The party p will wait for fragments from replicas. Once it receives valid fragments with a matching c from at least $f + 1$ replicas, it begins to decode from the fragments and reconstruct M.

Our new AVID protocol illustrated above carries fragment data only in the first SEND phase. The new AVID protocols from our ECP constructions ECP-1 and ECP-2 are called AVID-1 and AVID-2, respectively.

We compare AVID-1 and AVID-2 with existing AVID constructions in Table I (in the introduction). AVID-1 is the first AVID protocol that has optimal word complexity for both the dispersal and retrieval protocols. It is easy to see that AVID-1 outperforms CT AVID and HGR AVID consistently in all complexity measures. AVID-2 is strictly better than HGR AVID in all aspects. AVID-2 outperforms CT AVID for the dispersal protocol, while sharing the same complexities as CT AVID for the retrieval protocol. Moreover, compared to prior constructions, our constructions gain in modularity: our AVID protocols simply use ECP systems in a black-box manner.

VI. PROOF OF AVID

Proof. We now prove that our AVID protocol satisfies termination, agreement, availability, and correctness. Without loss of generality we assume optimal resiliency of $n = 3f + 1$.

Termination. On a high level, termination follows from Bracha's reliable broadcast and the correctness property of the ECP proof system. If a correct replica runs *avid-disperse* for id then all correct replicas will pass the check *ecverify*$(c, d_i, \pi_i) = 1$ because of the correctness property of the ECP system. Hence, all correct replicas will echo the same message c. Therefore, all $2f + 1$ correct replicas will receive enough ECHO messages to send a READY of their own. Therefore, all correct replicas will receive $2f+1$ READY messages for the same c and they can all *avid-deliver* for id.

Agreement. If a correct replica p_i completes the dispersal protocol, then it must have received $2f + 1$ READY messages with some c. At least $f + 1$ READY messages of c come from correct replicas. Hence, all correct replicas will receive $f + 1$ READY messages for the same c. Therefore, all correct replicas will send a READY message due to the amplification step. Since we have $2f+1$ correct replicas, all correct replicas will receive $2f + 1$ consistent READY messages for the same

c. As a result, all correct replicas will eventually complete the dispersal protocol.

Availability. If $f+1$ correct replicas complete *avid-disperse* for id, then at least one correct replica p_i has completed the dispersal protocol. Hence, p_i must have received $2f+1$ READY messages of c. Since the maximum number of faulty replicas is f, at least $f+1$ messages with the same c must have came from correct replicas. Thus, at least one correct replica has sent a READY message for c. (If not, at most f READY messages would be received by any correct replica.) Again, since f is the maximum number of malicious replicas, $f+1$ echo messages must have came from correct replicas. So at least $f+1$ correct replicas have received proofs that are consistent with their fragments that are associated with the same c. In particular, every replica p_j of those correct replicas has passed the check *ecverify*$(c, d_j, \pi_j) = 1$. Therefore, if any arbitrary client p were to initiate *avid-retrieve* for id, at least $f+1$ replicas will pass the check "$fragment \neq \perp$" and will send $f+1$ valid fragments and proofs with a matching c that will allow p to decode a message B.

Correctness. We first prove the first part of the property. To prove this, we first show correct replicas will store the same c. Suppose, on the contrary, that some correct replica stores c_1, while some other correct replicas stores c_2. Suppose $2f+1$ replicas have echoed c_1 and among these replicas, at least $f+1$ were correct. Meanwhile, we know at least $2f+1$ replicas have echoed c_2 and among these replicas, at least $f+1$ were correct. Thus, at least one correct replica has echoed more than once, which is prohibited from our protocol. Hence, any block decoded must have the same c. Due to the EC consistency property, all blocks decoded must be the same with overwhelming probability. For the second part of the property, if a correct client initiates the dispersal protocol, the decoded block must be equal to the original block dispersed with overwhelming probability, due to the EC correctness property. (Note the proof for correctness is more modular and concise than prior constructions, hiding the concrete implementation details.) □

VII. Applications

We argue ECP is indeed what is needed in erasure-coded distributed system. ECP can be used in many other fault-tolerant distributed computing settings. We briefly mention a few applications below.

BFT storage. In one example, BEAT3 is a BFT storage system in the BEAT family [10]. It uses HGR AVID and has a word complexity of $O(n^4)$. If using our AVID protocol, BEAT3 only needs $O(n^3)$ words.

AVID with fast read. Duan, Reiter, and Zhang proposed an AVID protocol with fast read. In such a protocol, a non-MDS (maximum distance separable) erasure coding scheme is used such that the retrieval is possible using less than $f+1$ erasure-coded fragments [10]. Our ECP system equally applies to this protocol, reducing the word complexity by a factor of $O(n)$.

Read/Write (R/W) storage. In another example, one can directly replace fingerprinted cross-checksum in the Byzantine-resilient atomic register system by Hendricks, Ganger, and Reiter [25] using our ECP, yielding a system that reduces the word complexity by an order of magnitude.

RBC. It is also easy to build a 3-step Byzantine reliable broadcast protocol using our technique, achieving a communication complexity of $O(nL + \lambda n^2)$. This result matches the asymptotic complexity of a recent protocol of Das, Xiang, and Ren [3], and slightly improves their protocol concretely ($6nL$ vs. $7nL$ for the nL terms).

VIII. Additional Related Work

Much related work has been discussed in the course of the paper. The section discusses additional related work on commitments and batch proofs.

Commitments and batch proofs. Several works have constructed polynomial [23], vector [20], or inner product [26] commitment schemes. Some of them also considered batch proofs for the same point z across different commitments, but all were done for specific constructions as opposed to our generic approach. Maller et al. [27] were the first to introduce the concept which they applied to their own polynomial commitments that are based on KZG [23] and which they used for building their own SNARK. Bünz et al. [22] defined batching for their own multivariate polynomial commitments, and Yurek et al. [28] showed how to do it with Bulletproofs [24] which they used to build their AVSS. Boneh et al. [29] generalized batching from a fixed point z to arbitrary subset of points across multiple polynomials, for their own variant commitment scheme of KZG.

IX. Conclusion

In this work, we introduce the concept of erasure coding proofs (ECP) and present efficient ECP constructions using polynomial commitments together with a novel batching procedure. We argue ECP is exactly the tool that we want in the distributed computing community: it allows us to build simple distributed protocols that improve (asymptotically or concretely) over the state of the art in almost all Byzantine resilient distributed computing protocols that carry bulk data.

References

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

[2] James Hendricks, Gregory R. Ganger, and Michael K. Reiter. Verifying distributed erasure-coded data. In *PODC*, 2007.

[3] Sourav Das, Zhuolun Xiang, and Ling Ren. Asynchronous data dissemination and its applications. *IACR Cryptol. ePrint Arch.*, 2021:777, 2021.

[4] Y. Lu, Z. Lu, Q. Tang, and G. Wang. Dumbo-mvba: Optimal multi-valued validated asynchronous byzantine agreement, revisited. In *PODC*, 2020.

[5] Kartik Nayak, Ling Ren, Elaine Shi, Nitin H. Vaidya, and Zhuolun Xiang. Improved extension protocols for byzantine broadcast and agreement. In Hagit Attiya, editor, *34th International Symposium on Distributed Computing, DISC 2020*.

[6] Idit Keidar, Eleftherios Kokoris-Kogias, Oded Naor, and Alexander Spiegelman. All you need is DAG. In *PODC*, pages 165–175. ACM, 2021.

[7] Dan Dobre, Ghassan O. Karame, Wenting Li, Matthias Majuntke, Neeraj Suri, and Marko Vukolic. Proofs of writing for robust storage. *IEEE Trans. Parallel Distributed Syst.*, 30(11):2547–2566, 2019.

[8] Elli Androulaki, Christian Cachin, Dan Dobre, and Marko Vukolic. Erasure-coded byzantine storage with separate metadata. In *Principles of Distributed Systems - 18th International Conference*, 2014.

[9] James Hendricks, Gregory R. Ganger, and Michael K. Reiter. Low-overhead byzantine fault-tolerant storage. In Thomas C. Bressoud and M. Frans Kaashoek, editors, *SOSP*, pages 73–86, 2007.

[10] Sisi Duan, Michael K Reiter, and Haibin Zhang. Beat: Asynchronous bft made practical. In *CCS*, pages 2028–2041. ACM, 2018.

[11] Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. Asymptotically optimal validated asynchronous byzantine agreement. In *PODC*, pages 337–346. ACM, 2019.

[12] Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan Gueta, and Ittai Abraham. Hotstuff: Bft consensus with linearity and responsiveness. In *38th ACM symposium on Principles of Distributed Computing (PODC)*, 2019.

[13] Guy Golan Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. Sbft: a scalable and decentralized trust infrastructure. In *2019 49th Annual IEEE/IFIP international conference on dependable systems and networks (DSN)*, pages 568–580. IEEE, 2019.

[14] Ittai Abraham, Philipp Jovanovic, Mary Maller, Sarah Meiklejohn, Gilad Stern, and Alin Tomescu. Reaching consensus for asynchronous distributed key generation. In *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*.

[15] Henrique Moniz, Nuno Ferreria Neves, Miguel Correia, and Paulo Verissimo. Ritas: Services for randomized intrusion tolerance. *IEEE transactions on dependable and secure computing*, 8(1):122–136, 2008.

[16] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of BFT protocols. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 31–42. ACM, 2016.

[17] Hugo Krawczyk. Distributed fingerprints and secure information dispersal. In *Proceedings of the twelfth annual ACM symposium on Principles of distributed computing*, pages 207–218, 1993.

[18] Li Gong. Securely replicating authentication services. In *[1989] Proceedings. The 9th International Conference on Distributed Computing Systems*, pages 85–91. IEEE, 1989.

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

[20] Dario Catalano and Dario Fiore. Vector commitments and their applications. In *International Workshop on Public Key Cryptography*, pages 55–72. Springer, 2013.

[21] Nicolas Alhaddad, Mayank Varia, and Haibin Zhang. High-threshold AVSS with optimal communication complexity. In *Financial Cryptography*, Lecture Notes in Computer Science. Springer, 2021.

[22] Benedikt Bünz, Ben Fisch, and Alan Szepieniec. Transparent snarks from dark compilers. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 677–706, Cham, 2020. Springer International Publishing.

[23] 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.

[24] 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.

[25] James Hendricks, Gregory R. Ganger, and Michael K. Reiter. Low-overhead byzantine fault-tolerant storage. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles 2007, SOSP 2007, Stevenson, Washington, USA, October 14-17, 2007*.

[26] 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.

[27] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge snarks from linear-size universal and updateable structured reference strings. *IACR Cryptol. ePrint Arch.*, 2019:99, 2019.

[28] Thomas Yurek, Licheng Luo, Jaiden Fairoze, Aniket Kate, and Andrew Miller. hbacss: How to robustly share many secrets. Cryptology ePrint Archive, Report 2021/159, 2021. https://ia.cr/2021/159.

[29] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. Efficient polynomial commitment schemes for multiple points and polynomials. *IACR Cryptol. ePrint Arch.*, 2020:81, 2020.