Erasure Coding in Distributed Protocols

Tutorial at PODC 2026

Vivien Bammert
Mariarosaria Barbaraci
Annalisa Cimatti
Christian Cachin

University of Bern
Institute of Computer Science
Cryptology and Data Security Group

July 10, 2026

Erasure Codes

Tutorial on Erasure Coding in Distributed Protocols

Vivien Bammert

Mariarosaria Barbaraci
Annalisa Cimatti
Christian Cachin

University of Bern
Institute of Computer Science
Cryptology and Data Security Group

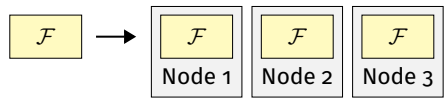
July 10, 2026

Motivation: Distributed Storage

Setting

- File $\mathcal{F}$ (Data) is distributed among multiple nodes
- Failures are unavoidable $\rightarrow$ data must remain available

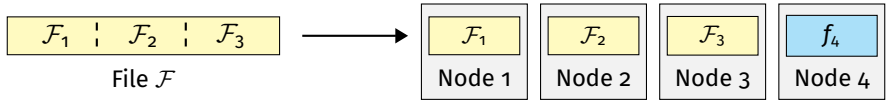
Naive Solution: $3\times$ Replication



- Tolerates up up to 2 node failures
- But 200% storage overhead

Scale Problem: Storage of tremendous amounts of data $\longrightarrow$ Replication is inefficient

RAID-5 [PGK88]



where $f_4 = \mathcal{F}_1 \oplus \mathcal{F}_2 \oplus \mathcal{F}_3$.

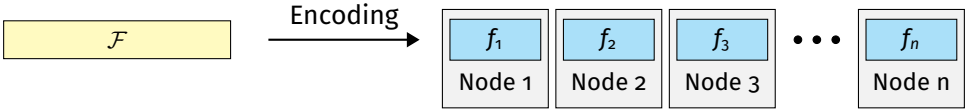
- Tolerates up to 1 failure
- $\approx$ 33% storage overhead

Key Question: Can we tolerate $t > 1$ failures while storing data with *small* storage overhead?

Erasure Codes

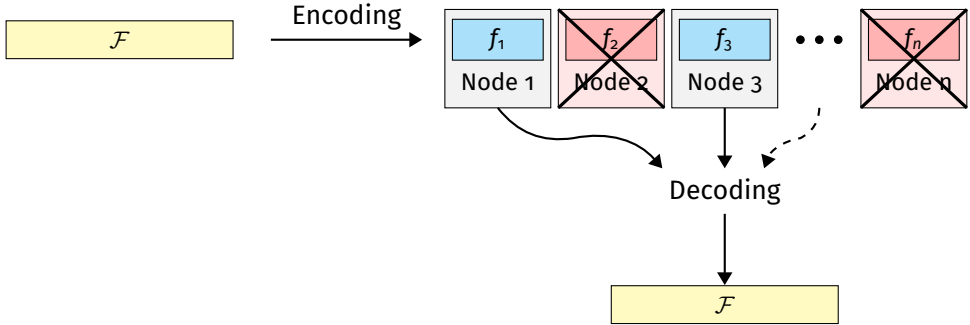
Erasure Codes

An **erasure code** splits data $\mathcal{F}$ into fragments so that the original data can still be recovered even if some fragments are lost.



Erasure Codes

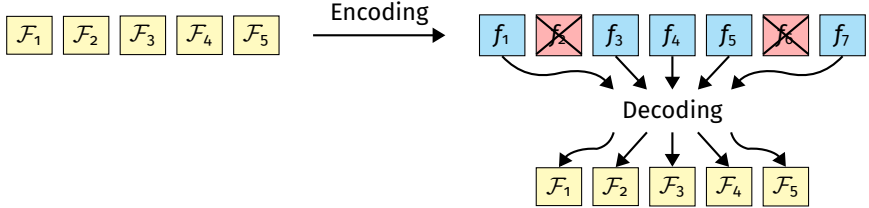
An **erasure code** splits data $\mathcal{F}$ into fragments so that the original data can still be recovered even if some fragments are lost.



Erasure Codes

Maximum-distance separable (MDS) codes (e.g., Reed-Solomon codes [RS60])

- Data is divided into k pieces and encoded into $n > k$ fragments of equal size
- Reconstruction of data is possible with any k out of n fragments

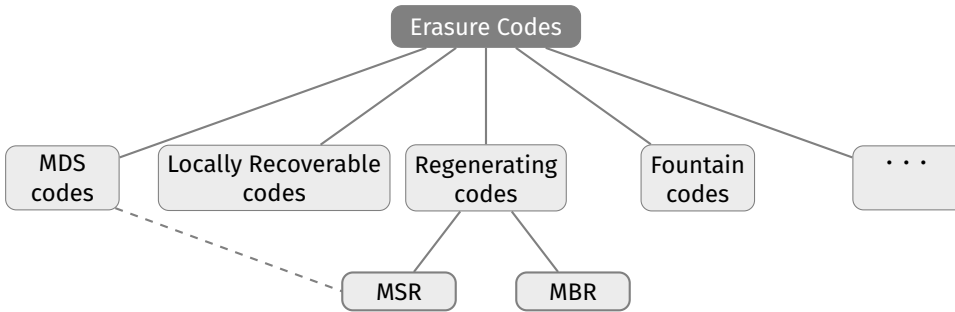


	Storage Overhead	Fault Tolerance
3× Replication	200%	2 failures
Reed-Solomon code ($n = 7, k = 5$)	40%	2 failures

Overview: Code Families

There is no universally “optimal” code $\rightarrow$ the choice depends on the system’s priorities.

- How much storage/memory is available?
- How many failures must be tolerated?
- How fast/efficient must repair of a lost fragment be?



Codes: Formal Definitions

Codes: Formal Definitions

A **linear** $[n, k]_q$-code $\mathcal{C}$ is a k-dimensional subspace of $\mathbb{F}_q^n$, where

- n: length of $\mathcal{C}$
- k: dimension of $\mathcal{C}$
- k/n: rate of $\mathcal{C}$

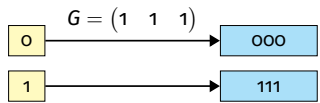
$\mathcal{C}$ is specified by a **generator matrix** $G \in \mathbb{F}_q^{k \times n}$, whose rows form a basis for $\mathcal{C}$:

$$\mathcal{C} = \{ mG \ : \ m \in \mathbb{F}_q^k \}$$

Example (Binary Repetition Code)

- $q = 2, n = 3, k = 1$
- $\mathcal{C} = \{000, 111\}$

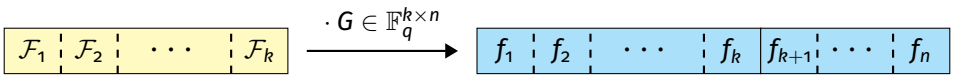
$\rightarrow 3\times$ Replication



Encoding of a File

Represent file $\mathcal{F}$ as a **vector** of length k over $\mathbb{F}_q$:

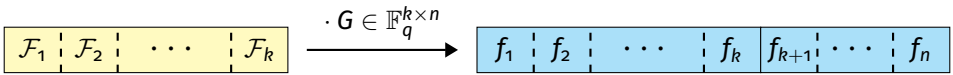
$$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k) \in \mathbb{F}_q^k$$



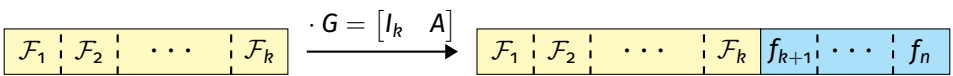
Encoding of a File

Represent file $\mathcal{F}$ as a **vector** of length k over $\mathbb{F}_q$:

$$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k) \in \mathbb{F}_q^k$$



Systematic form of G:



where

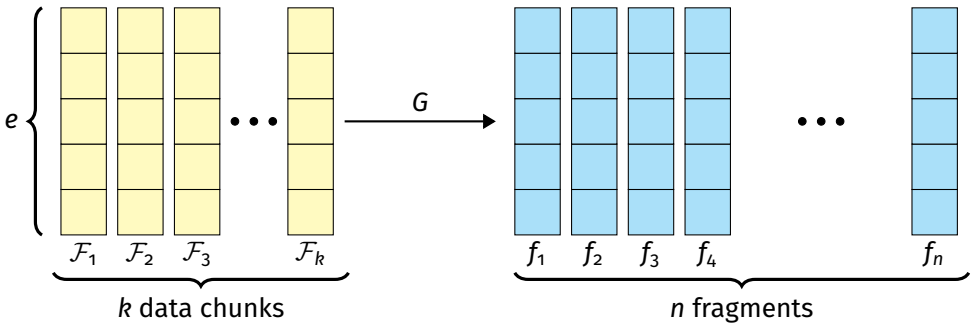
- I_k is the identity matrix of size k
- $A \in \mathbb{F}_q^{k \times (n-k)}$

Arrange Files as Matrices

Problem: Encoding an entire M-byte file at once requires memory $\geq M$ bytes

$\longrightarrow$ infeasible for GB/TB files

Encoding File as Matrix

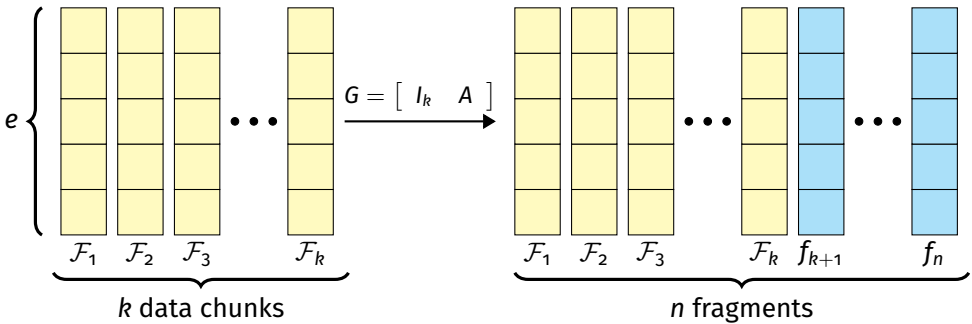


where

- $\mathcal{F}_i \in \mathbb{F}_q^e$ are **data chunks**

$\mathcal{C} \subseteq (\mathbb{F}_q^e)^n$

Systematic Encoding

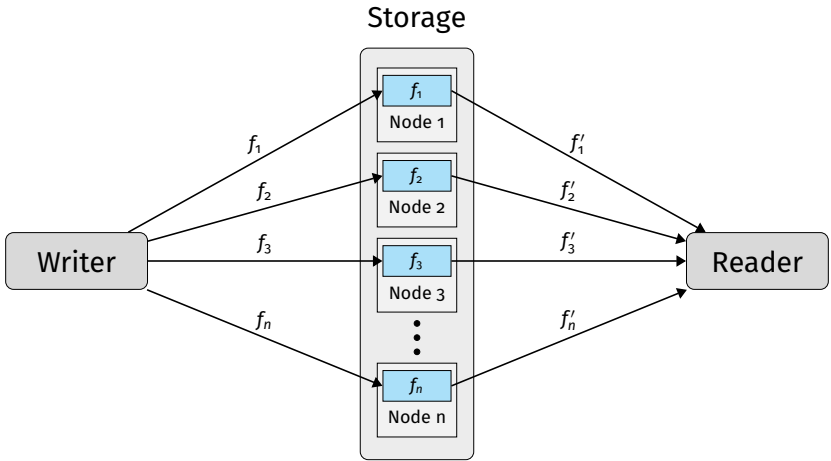


where

- $\mathcal{F}_i \in \mathbb{F}_q^e$ are **data chunks**

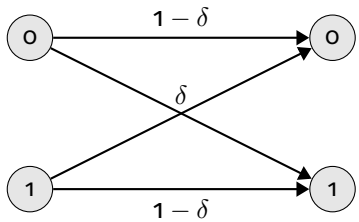
$\mathcal{C} \subseteq (\mathbb{F}_q^e)^n$

Storage System as Communication Channel



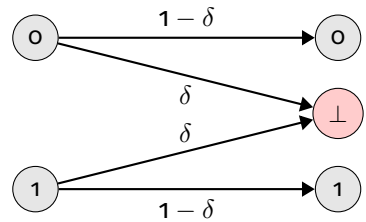
Failure Model

Binary Symmetric Channel



- δ error probability
- Error position **unknown**

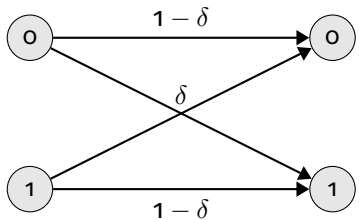
Binary Erasure Channel



- δ failure probability
- Erasure position **known**, content lost

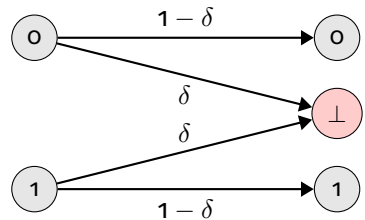
Failure Model

Binary Symmetric Channel



- δ error probability
- Error position **unknown**

Binary Erasure Channel



- δ failure probability
- Erasure position **known**, content lost

Error Correction Capability

Hamming Distance

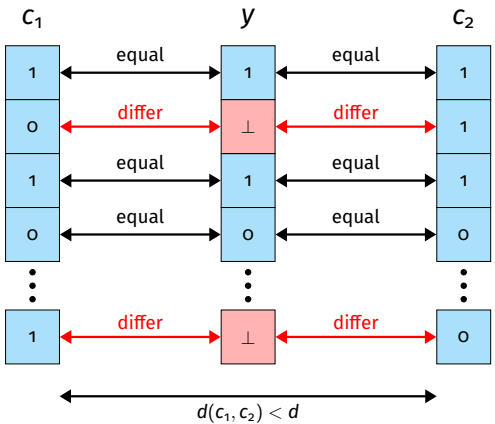
$$d(u, v) = |\{i : u_i \neq v_i\}|$$

where $u, v \in \mathbb{F}_q^n$.

Minimum Distance of a linear code $\mathcal{C}$

$$d(\mathcal{C}) = \min_{c_1 \neq c_2 \in \mathcal{C}} d(c_1, c_2)$$

Error Correction Capability



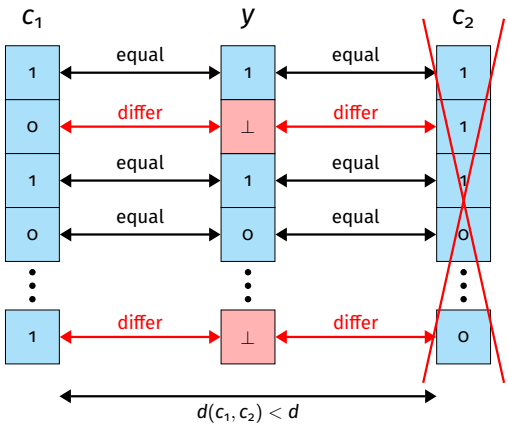
Erasure Model: An $[n, k]_q$-linear code can correct up to

$$d - 1 \text{ erasures}$$

$\Leftrightarrow$ reconstruction threshold κ is

$$\kappa = n - d + 1$$

Error Correction Capability



Erasure Model: An $[n, k]_q$-linear code can correct up to

$$d - 1 \text{ erasures}$$

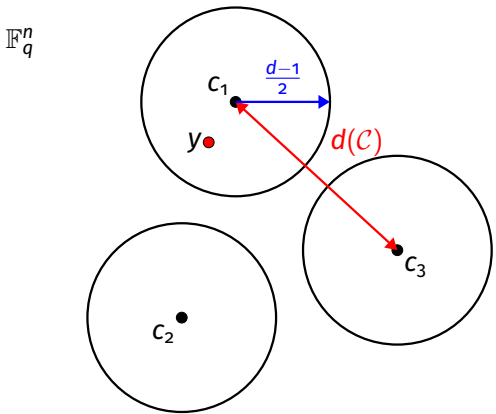
$\Leftrightarrow$ reconstruction threshold κ is

$$\kappa = n - d + 1$$

Example (Binary Repetition Code)

- $q = 2, n = 3, k = 1,\ \mathcal{C} = \{000, 111\}$
- Minimum Distance $d = 3 \longrightarrow \kappa = 1$

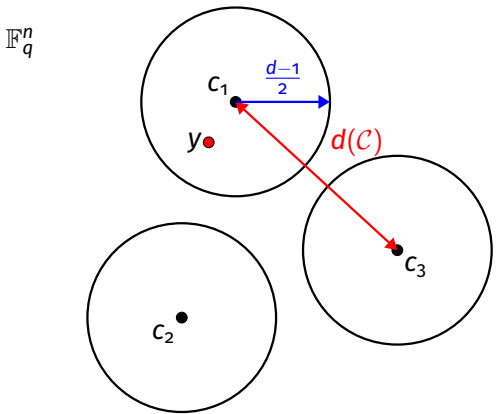
Error Correction Capability



Error Correction Model: An $[n, k]_q$-linear code with distance d can correct up to:

$$\left\lfloor \frac{d-1}{2} \right\rfloor \text{ errors.}$$

Error Correction Capability



Error Correction Model: An $[n, k]_q$-linear code with distance d can correct up to:

$$\left\lfloor \frac{d-1}{2} \right\rfloor \text{ errors.}$$

Example (Binary Repetition Code)

- $q = 2, n = 3, k = 1,\ \mathcal{C} = \{000, 111\}$
- Minimum Distance $d = 3 \longrightarrow 1$ error correction

Singleton Bound and MDS codes

Singleton Bound and MDS codes

Singleton Bound: For any $[n, k]_q$-linear code with minimum distance d it holds

$$\boxed{d \leq n - k + 1}$$

Codes achieving equality are called **Maximum-distance separable** (MDS) codes.

Singleton Bound and MDS codes

Singleton Bound: For any $[n, k]_q$-linear code with minimum distance d it holds

$$\boxed{d \leq n - k + 1}$$

Codes achieving equality are called **Maximum-distance separable** (MDS) codes.

MDS Codes Applications

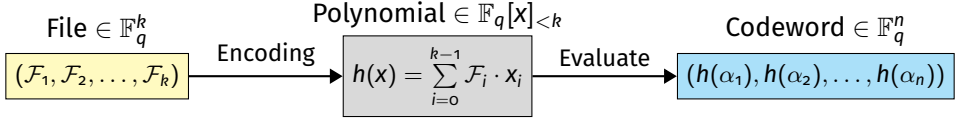
- Asynchronous Verifiable Information Dispersal (AVID) [CT05]
- Reliable Broadcast: MiniCast [LS25]
- BFT Protocols: DispersedLedger [YPAKT22]
- Consensus and Blockchain: DispersedSimplex [Sho24], Solana Alpenglow [KSW25]
- Data Availability Sampling [HSW24]

Reed-Solomon Codes

Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ be pairwise distinct, where $q \geq n$. The code defined by

$$RS(n, k) = \{(h(\alpha_1), \ldots, h(\alpha_n)) \mid h \in \mathbb{F}_q[x]_{<k}\}$$

is called **Reed-Solomon code** with length n and dimension k.

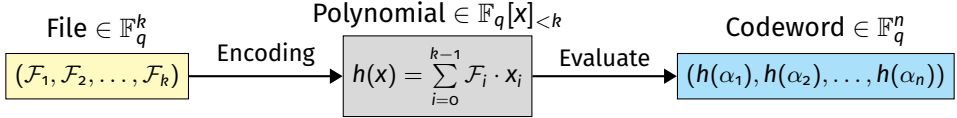


Reed-Solomon Codes

Let $\alpha_1, \ldots, \alpha_n \in \mathbb{F}_q$ be pairwise distinct, where $q \geq n$. The code defined by

$$RS(n,k) = \{(h(\alpha_1), \ldots, h(\alpha_n)) \mid h \in \mathbb{F}_q[X]_{<k}\}$$

is called **Reed-Solomon code** with length n and dimension k.



$\kappa = k \longrightarrow$ Tolerates $t = n - k$ erasures

Reed-Solomon Codes

Generator matrix: $k \times n$- Vandermonde matrix over $\mathbb{F}_q$, where $\alpha_i \in \mathbb{F}_q$ are pairwise distinct

$$G = \begin{pmatrix} 1 & 1 & \dots & 1 \\ \alpha_1 & \alpha_2 & \dots & \alpha_n \\ \vdots & & & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & \dots & \alpha_n^{k-1} \end{pmatrix} \in \mathbb{F}_q^{k \times n}.$$

Applications in Storage

	n	k	Storage Overhead	Fault Tolerance
SSRI [GGJR00]	n	$n-2t$	$2t/(n-2t)$	$2t$
Google Colossus [Fik10]	9	6	50%	3
Pelican [Bal+14]	18	15	20%	3
Facebook f4 [Sub+14]	14	10	40%	4

Advanced Properties of Codes

University of Bern

Outline

Fixed-Rate Codes:

- Locally Recoverable Codes [PD14]
- Regenerating Codes [DGWWR10]

Rateless Codes:

- Fountain Codes [Mac05]
- LT-Codes [Lub02]
- Raptor Codes [Sho06]

The Repair Bottleneck

What if a node failed? How to efficiently “*repair*” its fragment?

The Repair Bottleneck

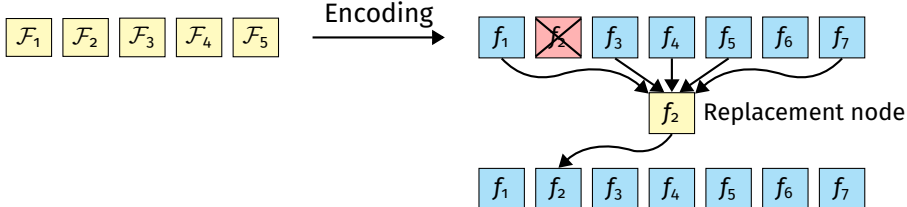
What if a node failed? How to efficiently *"repair"* its fragment?

Problem: To repair a **single** failed node in an $[n, k]_q$-MDS code

- Must contact k surviving nodes
- Download their fragments
- Reconstruct original data and re-encode missing fragment

$\longrightarrow$ very expensive for large k

Example $RS(7, 5)$

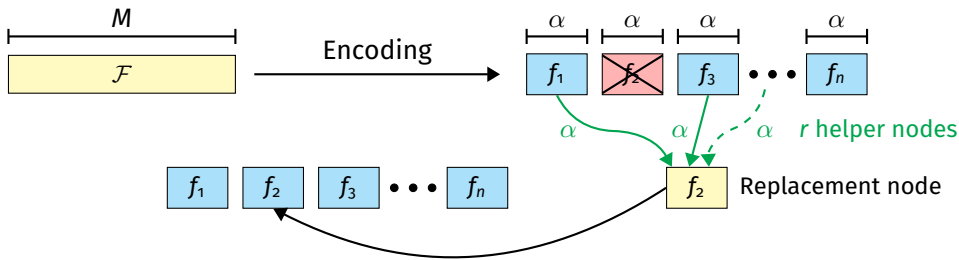


Locally Recoverable Codes [PD14]

Locally Recoverable Codes [PD14]

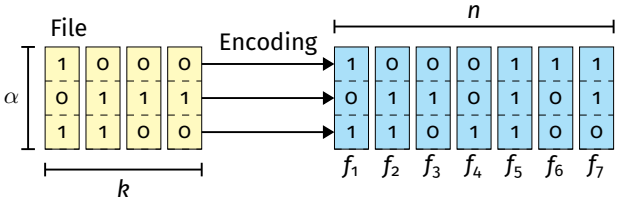
An $[n, k]_q$-code $\mathcal{C}$ encoding data $\mathcal{F}$ of size M into n **fragments** of size α is a **locally recoverable code (LRC)** with repair degree r and reconstruction threshold $\kappa > r$ if

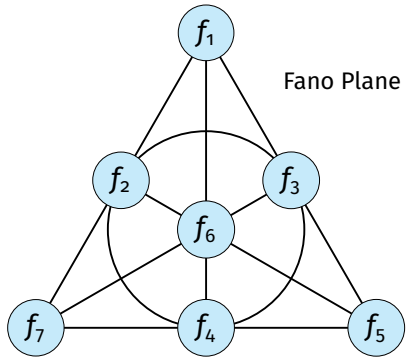
- Any κ fragments suffice to reconstruct $\mathcal{F}$
- Any lost fragment can be repaired by accessing at most r other fragments.



Example

LRC with parameters $n = 7, M = 12, k = 4, \alpha = 3$, threshold $\kappa = 5$ and repair degree $r = 2$





$f_i = f_j \oplus f_k$, if f_i, f_j, f_k are on the same line

Distance – Repair Degree Trade-off

For an LRC with repair degree r and parameters n, M, k, α, it holds

$$d \leq n - \left\lceil \frac{M}{\alpha} \right\rceil - \left\lceil \frac{M}{r \cdot \alpha} \right\rceil + 2$$

Trade-Off

Code	Repair degree r	Distance d
MDS (n, k)	$\kappa = k$	$n - k + 1$
LRC $(M = k \cdot \alpha)$	$\ll \kappa$	$\leq n - k - \lceil k/r \rceil + 2$

Applications

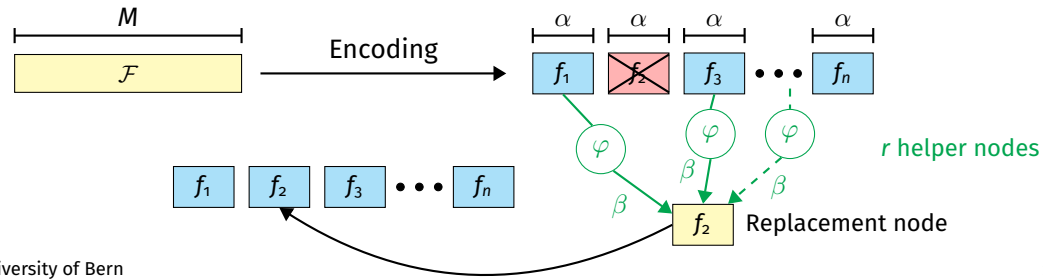
Microsoft Azure [Hua+12]

Regenerating Codes [DGWWR10]

Regenerating Codes [DGWWR10]

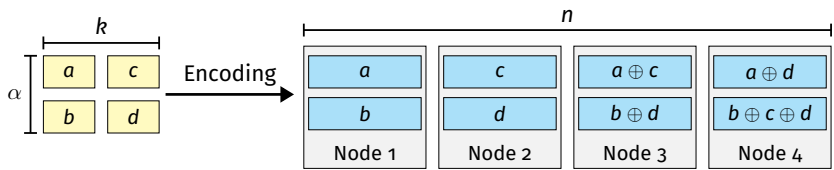
An $[n, k]_q$-code $\mathcal{C}$ encoding data $\mathcal{F}$ of size M into n fragments of size α is a **regenerating code** with repair degree r and reconstruction threshold κ if

- Any κ fragments suffice to reconstruct $\mathcal{F}$.
- A lost fragment can be repaired by downloading information of size $\beta \leq \alpha$ from r other nodes.



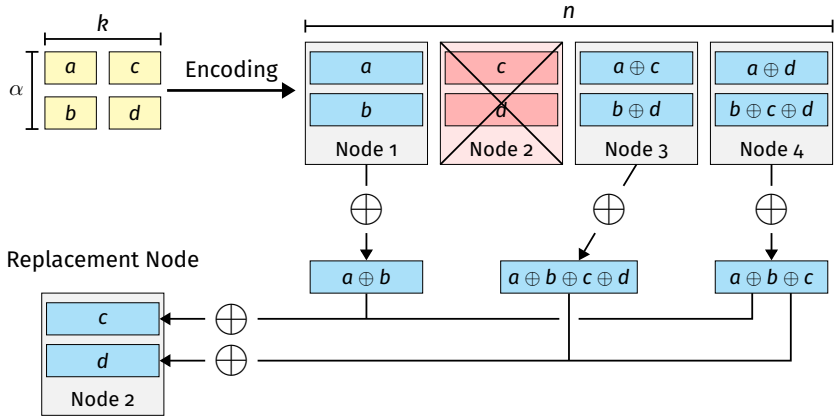
Example

Regenerating code with parameters $n = 4, M = 4, k = 2, \alpha = 2, \beta = 1$, threshold $\kappa = 2$ and repair degree $r = 3$



Example

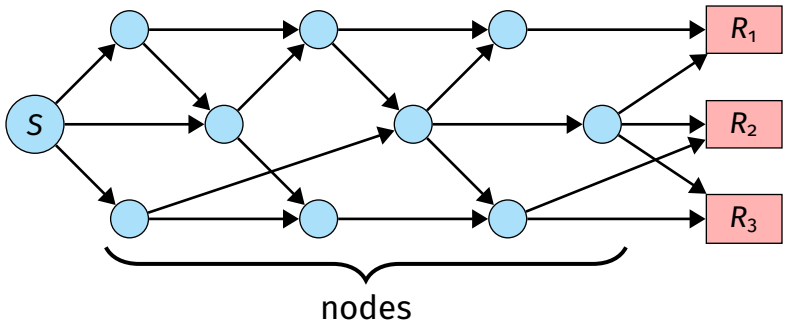
Regenerating code with parameters $n = 4, M = 4, k = 2, \alpha = 2, \beta = 1$, threshold $\kappa = 2$ and repair degree $r = 3$



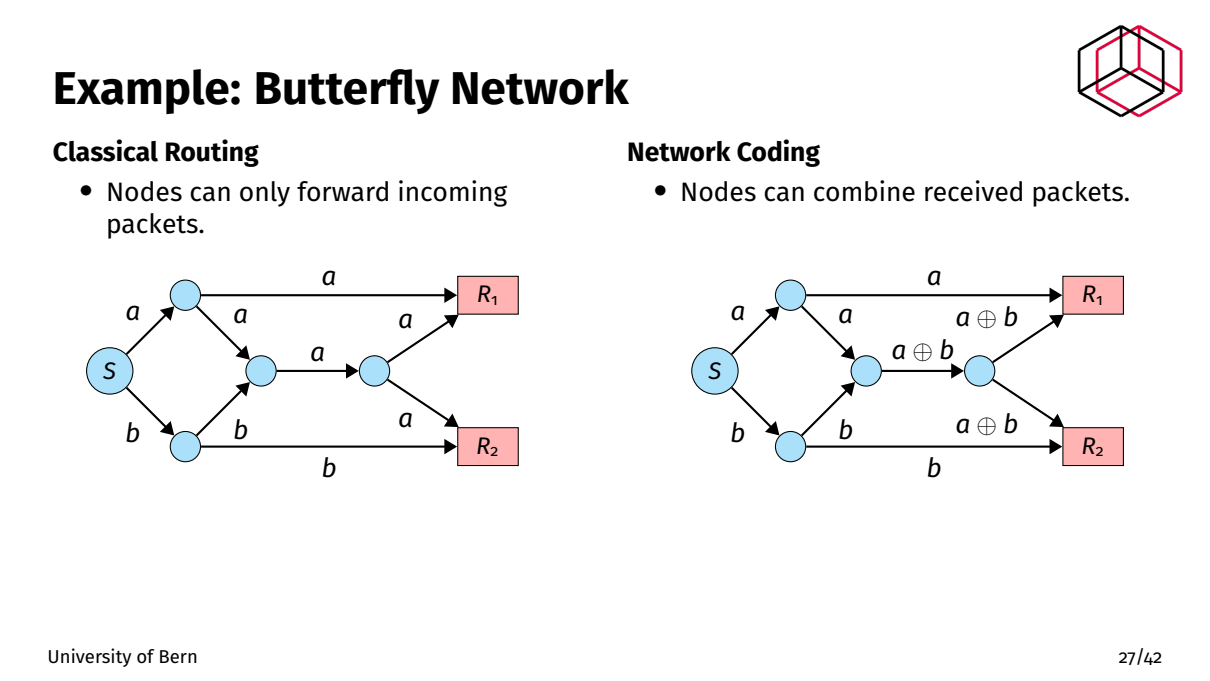
Network Coding: Multicast Model

Network Coding: Multicast Model

Goal: All receivers want to get the same data at the same time.



Example: Butterfly Network



Classical Routing

- Nodes can only forward incoming packets.

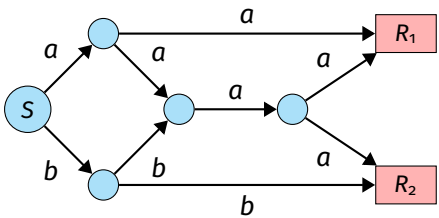
Network Coding

- Nodes can combine received packets.

Example: Butterfly Network

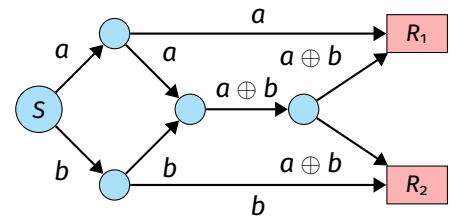
Classical Routing

- Nodes can only forward incoming packets.



Network Coding

- Nodes can combine received packets.

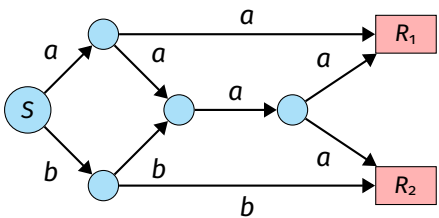


Network coding is a method of attaining maximum information flow in a multicast network.

Example: Butterfly Network

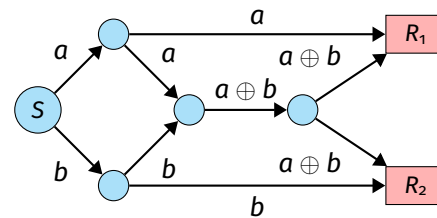
Classical Routing

- Nodes can only forward incoming packets.



Network Coding

- Nodes can combine received packets.



Applications

- Random Linear Network Coding: Gossiping in P2P Networks [Nic+25]

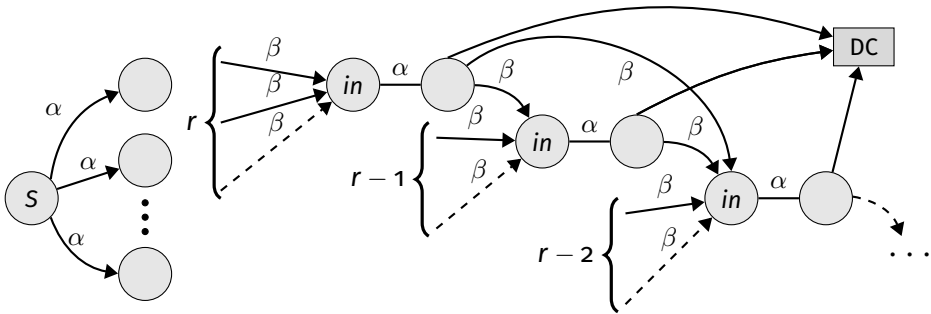
Storage – Bandwidth Trade-off for Regenerating Codes

Goal: Find relation between

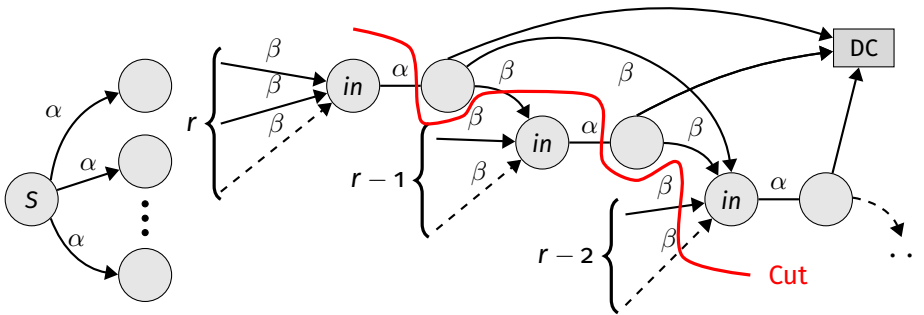
- File size M
- Storage per node α
- Repair bandwidth $r\beta$

$\longrightarrow$ Consider repair problem as an information flow graph in the multicast setting.

Information Flow Graph



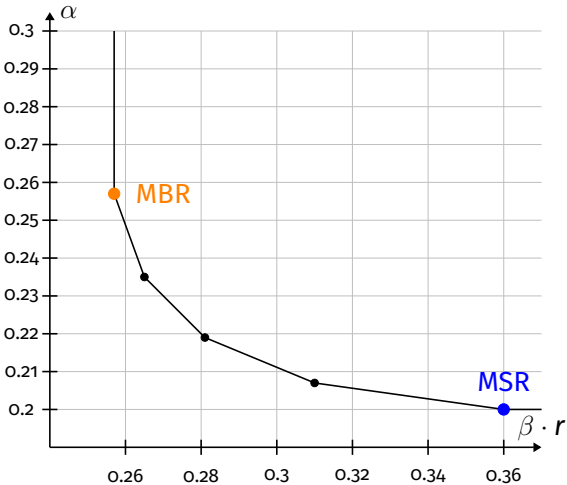
Information Flow Graph



Fundamental Bound:

$$M \leq \sum_{i=0}^{\kappa-1} \min\{\alpha, (r-i)\beta\} \longrightarrow \text{Network Coding can achieve this bound}$$

Trade-off Curve for Regenerating Codes



$n = 10, M = 1, \kappa = 5$

Minimum Storage Regenerating (MSR)

- $\alpha = \frac{M}{\kappa}$
- $\beta = \frac{M}{\kappa(r-\kappa+1)}$ where $r = n - 1$

Minimum Bandwidth Regenerating (MBR)

- $\alpha = \frac{2Mr}{2\kappa r - \kappa^2 + \kappa}$
- $\beta = \frac{\alpha}{r}$ where $r = n - 1$

Limitation of Fixed-Rate Codes

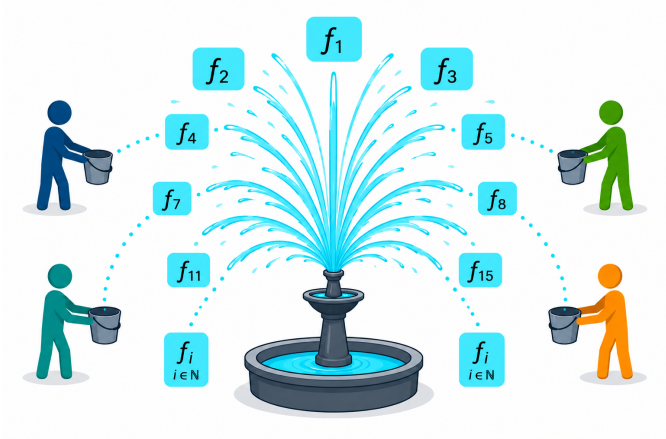
Limitation of Fixed-Rate Codes

Problem:

- A sender transmits data to many receivers at once (e.g, broadcast, multicast, streaming).
- Depending on the failure probability, classical codes fix n and k in advance.
  - Too little redundancy: Some receivers fail
  - Too much: Inefficient

Fountain Codes [Mac05]

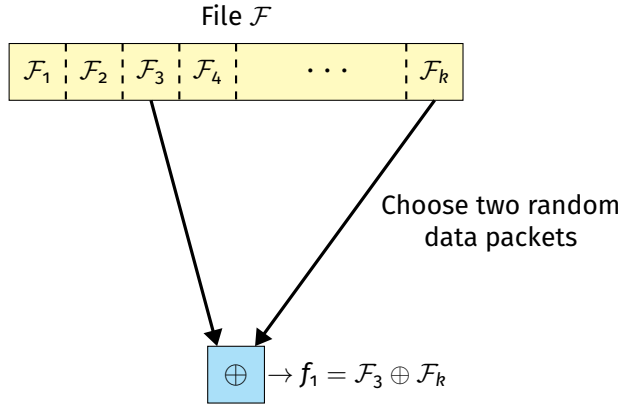
- The number of fragments can be determined on the fly from k data packets.
- Any $k(1+\varepsilon)$ fragments for small $\varepsilon > 0$ allows for reconstruction with high probability.



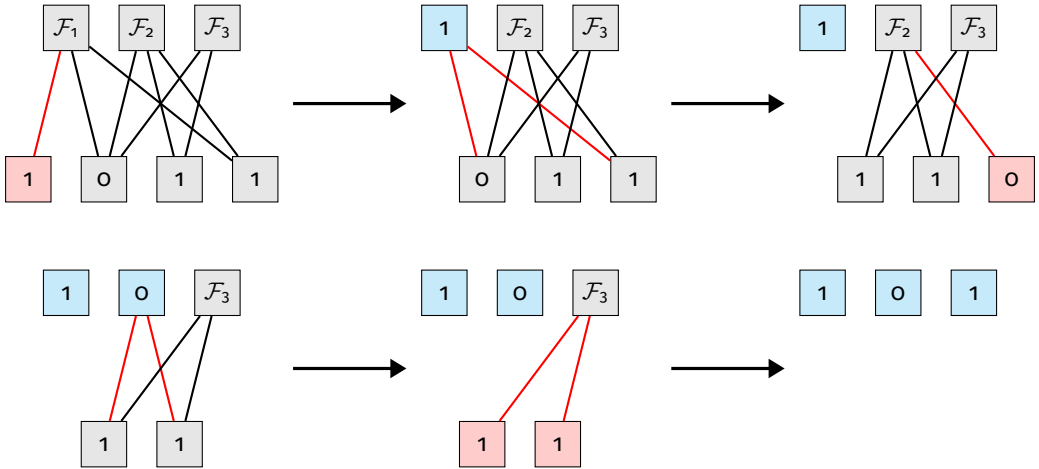
LT Codes [Lubo2]

For simplicity assume $\mathcal{F} \in \mathbb{F}_2^k$

Degree Distribution	
Degree	Probability
1	p_1
2	p_2
3	p_3
4	p_4
5	p_5
6	p_6
⋮	⋮



Decoding LT codes: Bit Flipping



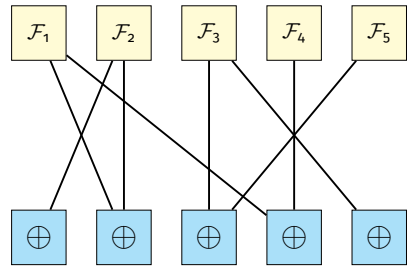
Decoding LT codes

In order that decoding works:

- Many fragments must have low degree
- Some fragments must have high degree so that there are no data packets that are connected to no-one.
- Average degree of fragments should be small for fast encoding and decoding

Complexity Cost: $\mathcal{O}(k \cdot ln(k))$
$\longrightarrow$ using *soliton degree distribution*

Bins: Data Packets
Balls: Linked Edges



Average balls per bin $\approx ln(k)$

Raptor Codes [Sho06]

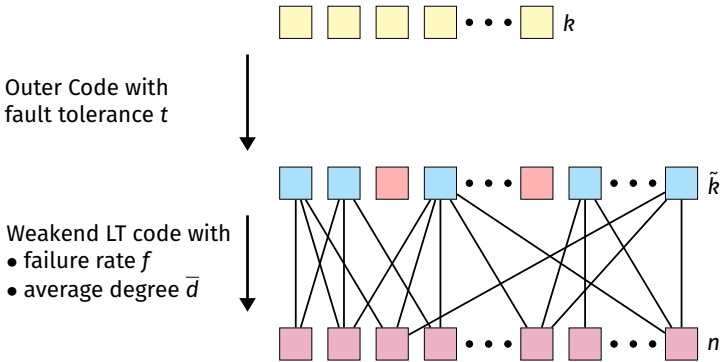
Goal: Reduce Complexity of LT codes $\longrightarrow$ Raptor Codes [Sho06] have linear time encoding and decoding

- Concatenating a *weakened* LT code with an outer fixed-rate code.
- Average degree of weakened LT code is about 3
- A fraction of data packets can have no connections to fragments $\longrightarrow$ can be tolerated by the outer code

Applications

- MonadBFT [JB25]

Raptor Codes [Sho06]



$$t \geq f \cdot \tilde{k}$$

fraction of empty bins:

$$e^{-N/\tilde{k}} \approx e^{-\overline{d}} \approx f$$

- N : # balls/edges
- $\overline{d}$: average degree

Monotone Erasure Codes [BCALC26]

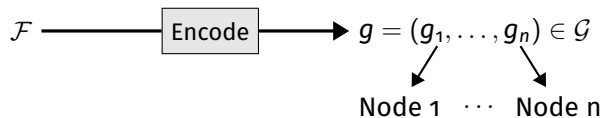
Classical fixed-rate codes:

Threshold Access Structure
$\mathcal{A} \subseteq 2^{\mathcal{P}}$, with $\mathcal{P}$ a set of nodes and $|A| = \kappa$ for all $A \in \mathcal{A}$.

→ **General Access Structure** $\mathcal{A} \subseteq 2^{\mathcal{P}}$

Monotone Erasure Codes

Let $\mathcal{F}$ be a file, $\mathcal{G} = G_1 \times \ldots \times G_n$ where $\mathcal{G}_i$ are (possibly different) sets.



Completeness: The nodes in each $A \in \mathcal{A}$ are able to reconstruct $\mathcal{F}$

Thank you for your attention

Vivien Bammert

Mariarosaria Barbaraci
Annalisa Cimatti
Christian Cachin

University of Bern
Institute of Computer Science
Cryptology and Data Security Group

July 10, 2026

References I

[Bal+14] Shobana Balakrishnan et al. “Pelican: A Building Block for Exascale Cold Data Storage”. In: *11th USENIX Symposium on Operating Systems Design and Implementation, OSDI ’14, Broomfield, CO, USA, October 6-8, 2014*. Ed. by Jason Flinn and Hank Levy. USENIX Association, 2014, pp. 351–365. URL: https://www.usenix.org/conference/osdi14/technical-sessions/presentation/balakrishnan.

[BCALC26] Vivien Bammert et al. “Monotone Erasure Codes”. In: *arXiv preprint arXiv:2605.22426* (2026).

[CT05] Christian Cachin and Stefano Tessaro. “Asynchronous Verifiable Information Dispersal”. In: *Distributed Computing, 19th International Conference, DISC 2005, Cracow, Poland, September 26-29, 2005, Proceedings*. Ed. by Pierre Fraigniaud. Vol. 3724. Lecture Notes in Computer Science. Springer, 2005, pp. 503–504. DOI: 10.1007/11561927_42.

References II

[DGWWR10] Alexandros G. Dimakis et al. “Network coding for distributed storage systems”. In: *IEEE Trans. Inf. Theory* 56.9 (2010), pp. 4539–4551. DOI: 10.1109/TIT.2010.2054295.

[Fik10] Andrew Fikes. “*Storage architecture and challenges*”. 2010. URL: https://cloud.google.com/files/storage_architecture_and_challenges.pdf.

[GGJR00] Juan A. Garay et al. “Secure distributed storage and retrieval”. In: *Theor. Comput. Sci.* 243.1-2 (2000), pp. 363–389. DOI: 10.1016/S0304-3975(98)00263-1.

[HSW24] Mathias Hall-Andersen et al. “Foundations of Data Availability Sampling”. In: *IACR Commun. Cryptol.* 1.4 (2024), p. 34. DOI: 10.62056/A09QUDHDJ.

References III

[Hua+12] Cheng Huang et al. “Erasure Coding in Windows Azure Storage”. In: *Proceedings of the 2012 USENIX Annual Technical Conference, USENIX ATC 2012, Boston, MA, USA, June 13-15, 2012*. Ed. by Gernot Heiser and Wilson C. Hsieh. USENIX Association, 2012, pp. 15–26. URL: https://www.usenix.org/conference/atc12/technical-sessions/presentation/huang.

[JB25] Mohammad Mussadiq Jalalzai and Kushal Babel. “MonadBFT: Fast, Responsive, Fork-Resistant Streamlined Consensus”. In: *CoRR* abs/2502.20692 (2025). arXiv: 2502.20692. DOI: 10.48550/ARXIV.2502.20692.

[KSW25] Quentin Kniep et al. *Solana Alpenglow Consensus. Alpenglow White Paper*. 2025. URL: https://www.anza.xyz/blog/alpenglow-a-new-consensus-for-solana.

References IV

[LS25] Thomas Locher and Victor Shoup. “MiniCast: Minimizing the Communication Complexity of Reliable Broadcast”. In: *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part V*. Ed. by Serge Fehr and Pierre-Alain Fouque. Vol. 15605. Lecture Notes in Computer Science. Springer, 2025, pp. 96–115. DOI: 10.1007/978-3-031-91092-0_4.

[Lub02] Michael Luby. “LT Codes”. In: *43rd Symposium on Foundations of Computer Science, FOCS 2002, Vancouver, BC, Canada, November 16-19, 2002, Proceedings*. IEEE Computer Society, 2002, p. 271. DOI: 10.1109/SFCS.2002.1181950.

[Mac05] David JC MacKay. “Fountain codes”. In: *IEE Proceedings-Communications* 152.6 (2005), pp. 1062–1068.

References V

[Nic+25] Nicolas Nicolaou et al. “OptimumP2P: Fast and Reliable Gossiping in P2P Networks”. In: *21st International Conference on Network and Service Management, CNSM 2025, Bologna, Italy, October 27-31, 2025*. Ed. by Walter Cerroni et al. IEEE, 2025, pp. 1–9. DOI: 10.23919/CNSM67658.2025.11297457.

[PD14] Dimitris S. Papailiopoulos and Alexandros G. Dimakis. “Locally Repairable Codes”. In: *IEEE Trans. Inf. Theory* 60.10 (2014), pp. 5843–5855. DOI: 10.1109/TIT.2014.2325570.

[PGK88] David A. Patterson et al. “A Case for Redundant Arrays of Inexpensive Disks (RAID)”. In: *Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 1-3, 1988*. Ed. by Haran Boral and Per-Åke Larson. ACM Press, 1988, pp. 109–116. DOI: 10.1145/50202.50214.

References VI

[RH17] Amogh Rajanna and Martin Haenggi. “Enhanced Cellular Coverage and Throughput Using Rateless Codes”. In: *IEEE Trans. Commun.* 65.5 (2017), pp. 1899–1912. DOI: 10.1109/TCOMM.2017.2676096.

[RS60] I. S. Reed and G. Solomon. “Polynomial Codes Over Certain Finite Fields”. In: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304. DOI: 10.1137/0108018.

[Sho06] Amin Shokrollahi. “Raptor codes”. In: *IEEE Trans. Inf. Theory* 52.6 (2006), pp. 2551–2567. DOI: 10.1109/TIT.2006.874390.

[Sho24] Victor Shoup. “Sing a Song of Simplex”. In: *38th International Symposium on Distributed Computing, DISC 2024, Madrid, Spain, October 28 - November 1, 2024*. Ed. by Dan Alistarh. Vol. 319. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 37:1–37:22. DOI: 10.4230/LIPICS.DISC.2024.37.

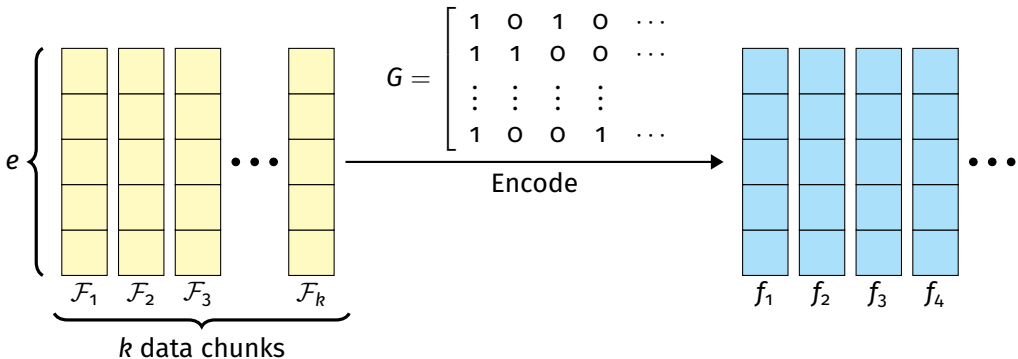
References VII

[Sub+14] Muralidhar Subramanian et al. “f4: Facebook’s Warm BLOB Storage System”. In: *11th USENIX Symposium on Operating Systems Design and Implementation, OSDI ’14, Broomfield, CO, USA, October 6-8, 2014*. Ed. by Jason Flinn and Hank Levy. USENIX Association, 2014, pp. 383–398. URL: https://www.usenix.org/conference/osdi14/technical-sessions/presentation/muralidhar.

[YPAKT22] Lei Yang et al. “DispersedLedger: High-Throughput Byzantine Consensus on Variable Bandwidth Networks”. In: *19th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2022, Renton, WA, USA, April 4-6, 2022*. Ed. by Amar Phanishayee and Vyas Sekar. USENIX Association, 2022, pp. 493–512. URL: https://www.usenix.org/conference/nsdi22/presentation/yang.

Random Linear Fountain Codes

Let $\mathcal{F}_i \in \mathbb{F}_2^e$



Random Linear Fountain Code

What is the chance that the receiver will be able to recover the original data?

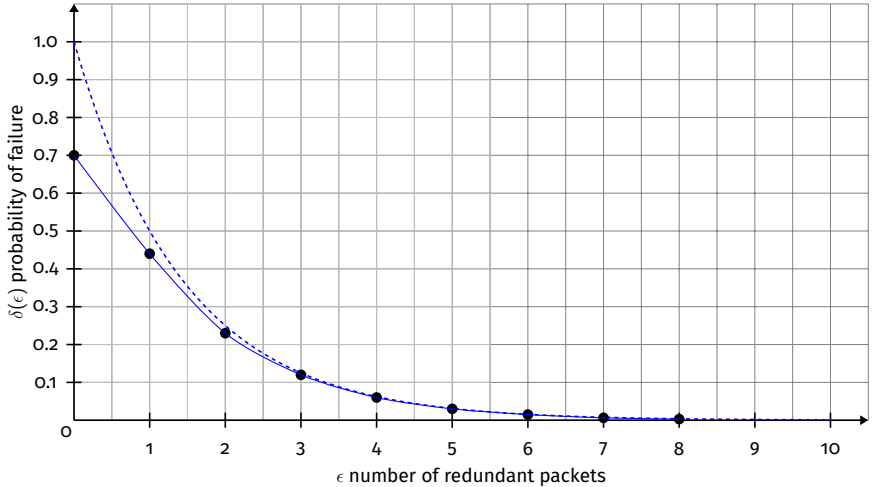
- If $N < k \longrightarrow$ no reconstruction possible
- If $N = k \longrightarrow$ probability that a random $k \times k$ binary matrix is invertible:

$$\prod_{i=0}^{k-1} \left(1 - 2^{-(k-i)}\right) \approx 0.289 \text{ for } k \geq 10$$

- If $N = k + \epsilon \longrightarrow$ Failure probability $\delta(\epsilon)$:

$$\delta(\epsilon) \leq 2^{-\epsilon}$$

Random Linear Fountain Code



Basic Protocols using Erasure Codes

Tutorial on Erasure Coding in Distributed Protocols

Annalisa Cimatti

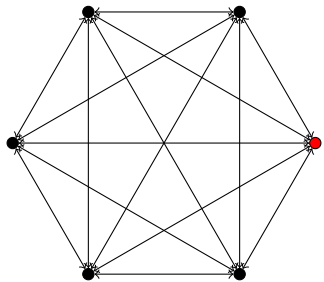
Vivien Bammert
Mariarosaria Barbaraci
Christian Cachin

University of Bern
Institute of Computer Science
Cryptology and Data Security Group

July 10, 2026

Setting

- Network of n nodes $P_1, \ldots, P_n$ communicating via p2p authenticated channels
- Network is asynchronous
- $t < \frac{n}{3}$ nodes may be Byzantine



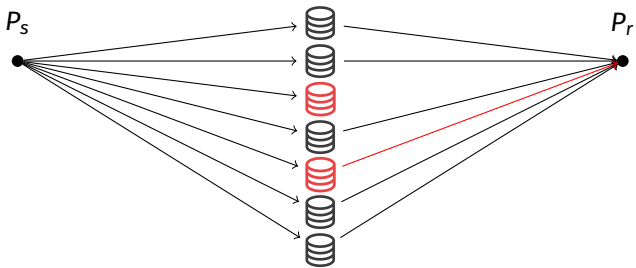
Overview

1. Asynchronous verifiable information dispersal (AVID)
2. AVID Improvements
3. Reliable Broadcast
4. RB Improvements

Asynchronous verifiable information dispersal (AVID)

AVID

Goal: allow a client P_s to distribute some data among n servers, of which up to t might be Byzantine, in such a way that a reader P_r can always recover the stored data correctly.



AVID

Goal: allow a client P_s to distribute some data among n servers, of which up to t might be Byzantine, in such a way that a reader P_r can always recover the stored data correctly.

An **asynchronous verifiable information dispersal** (AVID) protocol [CT05] is made of two functions:

- DISPERSE: a client P_s distributes a file F using an $[n, k]$-erasure code
- RETRIEVE: any client can reconstruct F from k fragments

AVID Properties

Termination. If P_s is honest, then all honest servers eventually complete the dispersal.

AVID Properties

Termination. If P_s is honest, then all honest servers eventually complete the dispersal.

Agreement. If some honest server completes the dispersal, then all honest servers eventually complete the dispersal.

AVID Properties

Termination. If P_s is honest, then all honest servers eventually complete the dispersal.

Agreement. If some honest server completes the dispersal, then all honest servers eventually complete the dispersal.

Availability. If k honest servers complete the dispersal, and an honest client P_r starts the retrieval protocol, then it eventually reconstructs some file F'.

AVID Properties

Termination. If P_s is honest, then all honest servers eventually complete the dispersal.

Agreement. If some honest server completes the dispersal, then all honest servers eventually complete the dispersal.

Availability. If k honest servers complete the dispersal, and an honest client P_r starts the retrieval protocol, then it eventually reconstructs some file F'.

Correctness. If k honest servers complete the dispersal, then there exists a fixed value G such that:

1. If P_s is honest and has dispersed a file F, then $G = F$.
2. If an honest client P_r reconstructs a file F', then $G = F'$.

AVID Protocol

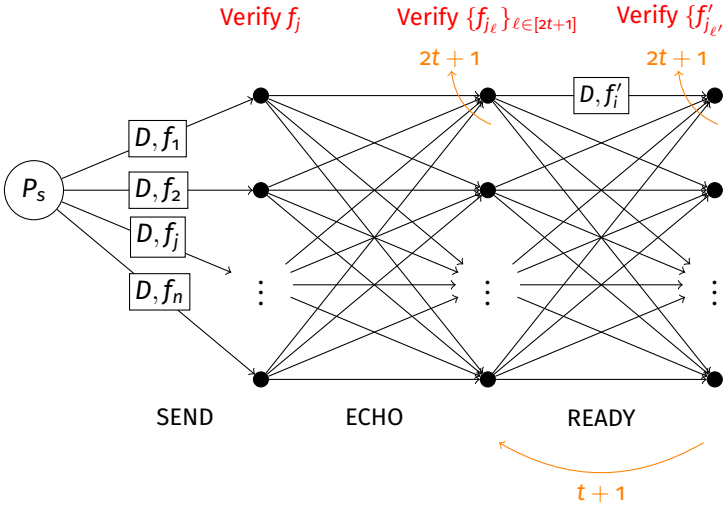
Let:

- $n = 3t + 1$ and $k = t + 1$
- $[f_1, \ldots, f_n] = \text{Encode}(F)$ be the encoding of the file F (computed by P_s)
- $D = [H(f_1), \ldots, H(f_n)]$ be the vector of hashes of the fragments

$$F \rightarrow \boxed{\textsc{Disperse}} \rightarrow (D, f_j)$$

$$\text{"retrieve"} \rightarrow \boxed{\textsc{Retrieve}} \rightarrow F'$$

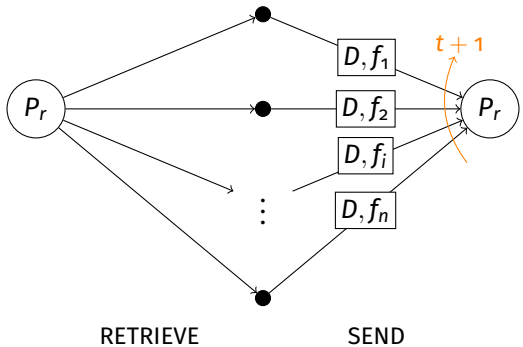
Disperse



$$F' = \text{Decode}([f_{j_1}, \ldots, f_{j_{2t+1}}])$$
$$[f'_1, \ldots, f'_n] = \text{Encode}(F')$$
$$H(f'_i) \stackrel{?}{=} D[i]$$

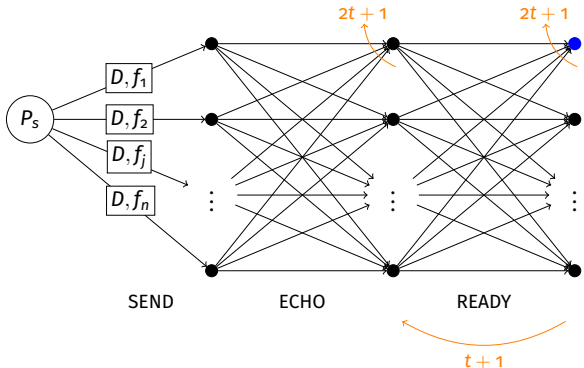
Retrieve

P_r collects $k = t + 1$ valid fragments and decodes them to retrieve F'



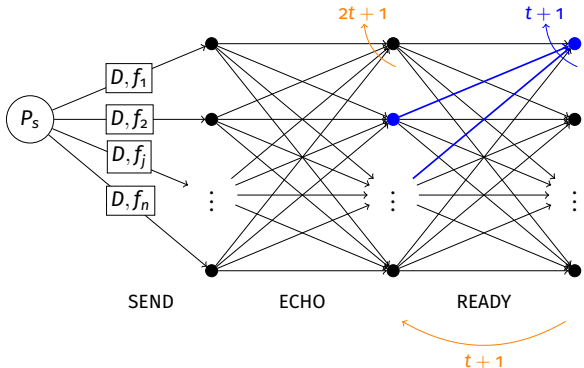
Proof sketch

Agreement. If some honest server completes the dispersal, then all honest servers eventually complete the dispersal.



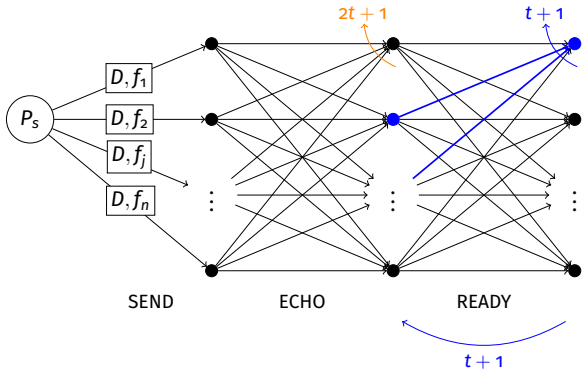
Proof sketch

Agreement. If some honest server completes the dispersal, then all honest servers eventually complete the dispersal.



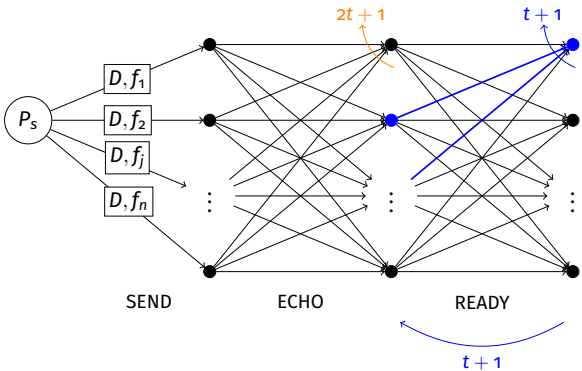
Proof sketch

Agreement. If some honest server completes the dispersal, then all honest servers eventually complete the dispersal.



Proof sketch

Agreement. If some honest server completes the dispersal, then all honest servers eventually complete the dispersal.

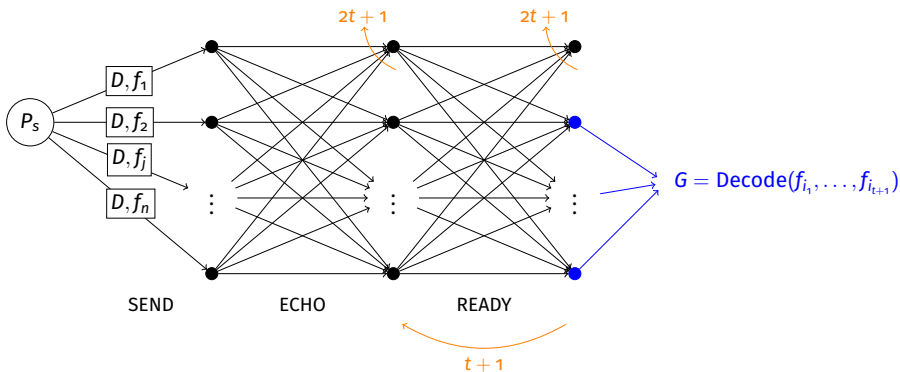


$\Longrightarrow$ all honest servers $(2t+1)$ send READY

Proof sketch

Correctness. If k honest servers complete the dispersal, then there exists a fixed value G such that:

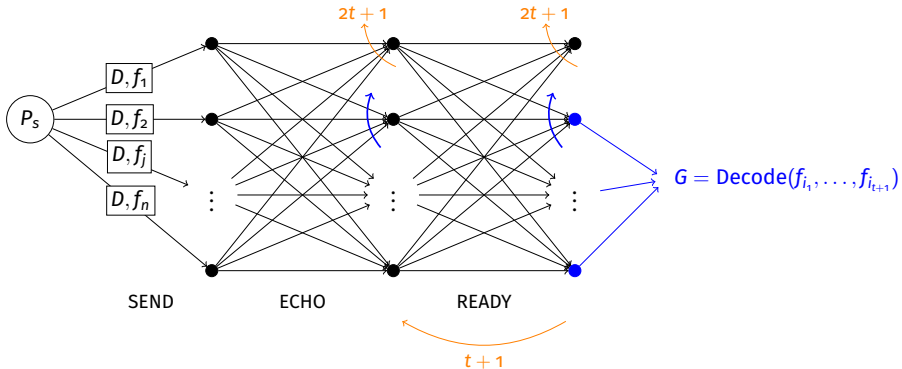
1. If P_s is honest and has dispersed a file F, then $G = F$.



Proof sketch

Correctness. If k honest servers complete the dispersal, then there exists a fixed value G such that:

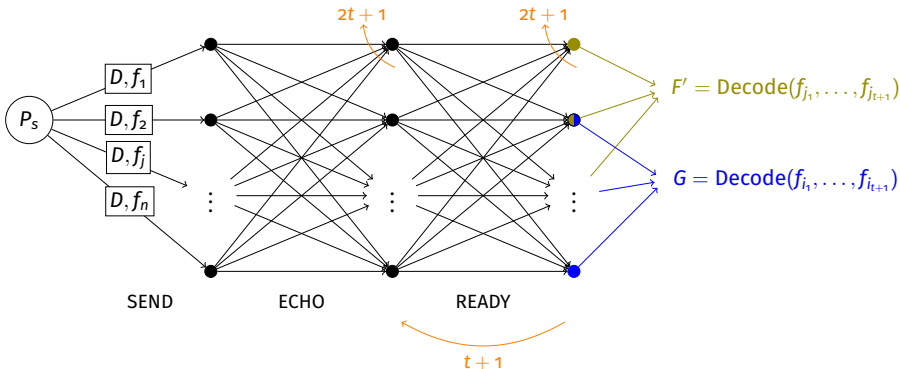
1. If P_s is honest and has dispersed a file F, then $G = F$.



Proof sketch

Correctness. If k honest servers complete the dispersal, then there exists a fixed value G such that:

1. If P_s is honest and has dispersed a file F, then $G = F$.
2. If an honest client P_r reconstructs a file F', then $G = F'$.



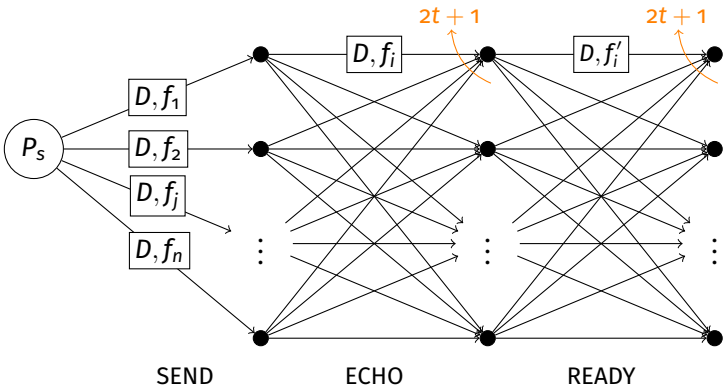
Efficiency measures

Communication complexity: bit length of messages sent by all parties

Storage complexity: overall bit length of the information stored by honest servers after they have completed the dispersal protocol

Efficiency of AVID

Communication complexity: $O(n^2 \frac{|F|}{k} + n^3\kappa) = O(n|F| + n^3\kappa)$ (DISPERSE)

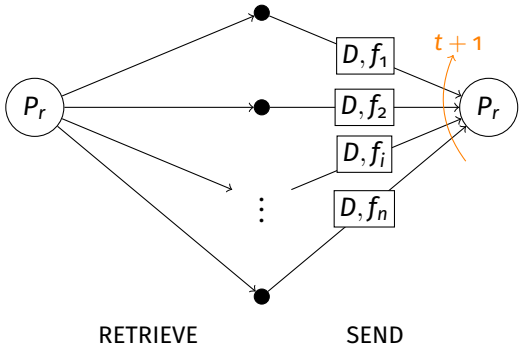


Each party sends (D, f_i) to the other n, where $|f_i| = \frac{|F|}{k}$ and $|D| = n\kappa$

$\rightarrow O(n\frac{|F|}{k} + n^2\kappa)$

Efficiency of AVID

Communication complexity: $O(n|F| + n^3\kappa)$ (DISPERSE) + $O(|F| + n^2\kappa)$ (RETRIEVE)



$O(n)$ nodes send (D, f_i) to P_r

$\rightarrow O(n\frac{|F|}{k} + n^2\kappa)$

Efficiency of AVID

Communication complexity: $O(n|F| + n^3\kappa)$ (Disperse) + $O(|F| + n^2\kappa)$ (Retrieve)

Storage complexity: $O(n\frac{|F|}{k} + n^2\kappa) = O(|F| + n^2\kappa)$

Efficiency of AVID

Communication complexity: $O(n|F| + n^3\kappa)$ (Disperse) + $O(|F| + n^2\kappa)$ (Retrieve)

Storage complexity: $O(n\frac{|F|}{k} + n^2\kappa) = O(|F| + n^2\kappa)$

Can we improve efficiency?

AVID Improvements

University of Bern

Commitment scheme

A **commitment scheme** allows a committer to publish a value C (commitment) satisfying the following two properties:

- **binding:** C binds the committer to a message M;
- **hiding:** C reveals no information about M.

Commitment scheme

A **commitment scheme** allows a committer to publish a value C (commitment) satisfying the following two properties:

- **binding:** C binds the committer to a message M;
- **hiding:** C reveals no information about M.

Example: In AVID, $D = [H(f_1), \ldots, H(f_n)]$ is a commitment to $[f_1, \ldots, f_n]$.

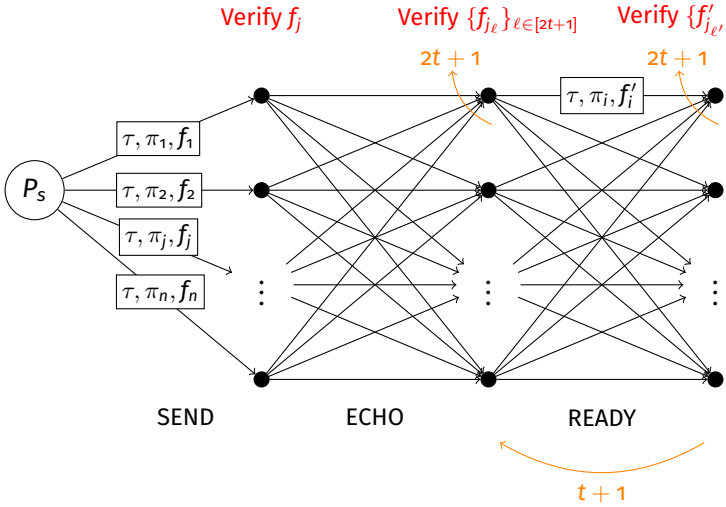
Commitments

- **Hash vector:** $[H(f_1), \ldots, H(f_n)] \rightarrow O(\kappa n)$

Commitments

- **Hash vector:** $[H(f_1), \ldots, H(f_n)] \rightarrow O(\kappa n)$
- **Merkle tree:** (τ, π_j) where τ is the root of the Merkle tree of $[f_1, \ldots, f_n]$ and π_j is the authentication path of $f_j \rightarrow O(\kappa log(n))$

AVID-H



$\rightarrow O(n|F| + \kappa n^2 log(n))$

Commitments

- **Hash vector:** $[H(f_1), \ldots, H(f_n)] \rightarrow O(\kappa n)$
- **Merkle tree:** (τ, π_j) where τ is the root of the Merkle tree of $[f_1, \ldots, f_n]$ and π_j is the authentication path of $f_j \rightarrow O(\kappa log(n))$
- **Homomorphic fingerprints [HGR07]:** universal hash functions preserving the algebraic structure of the fragments $\rightarrow O(\kappa n)$

Commitments

- **Hash vector:** $[H(f_1), \ldots, H(f_n)] \rightarrow O(\kappa n)$
- **Merkle tree:** (τ, π_j) where τ is the root of the Merkle tree of $[f_1, \ldots, f_n]$ and π_j is the authentication path of $f_j \rightarrow O(\kappa log(n))$
- **Homomorphic fingerprints [HGR07]:** universal hash functions preserving the algebraic structure of the fragments $\rightarrow O(\kappa n)$
- **Polynomial commitments**

$$[f_1, \ldots, f_n] \longleftrightarrow \Phi(x) = \sum_{i=0}^{n-1} \Phi_i x^i \text{ s.t. } \Phi(i) = f_i$$

KZG Commitments

Setup

- $\mathbb{G}, \mathbb{G}_T$ groups of order p and g generator of $\mathbb{G}$
- bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$
- $\alpha \leftarrow \mathbb{Z}_p^*$
- srs $= \left(g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^d}\right) \in \mathbb{G}^{d+1}$

Commit(Φ)

Given $\Phi(X) = \sum_{i=0}^{d} \Phi_i X^i$ over $\mathbb{F}_p$, the commitment to $\Phi(X)$ is

$$C = g^{\Phi(\alpha)} = \prod_{i=0}^{d} g^{\alpha^i \Phi_i}$$

Witness(Φ, z)

Compute $y = \Phi(z)$ and

$$w_z = g^{\frac{\Phi(\alpha) - y}{\alpha - z}}$$

KZG Commitments

Verify(C, z, y, w_z)

Given the commitment C, two field elements z, y, and the witness $w_z = g^{\frac{\Phi(\alpha)-y}{\alpha-z}}$, check

$$e(C, g) \stackrel{?}{=} e(w_z, g^{\alpha}/g^{z})\ e(g, g)^{y}$$

which holds iff $\Phi(z) = y$.

KZG Commitments

Verify(C, z, y, w_z)

Given the commitment C, two field elements z, y, and the witness $w_z = g^{\frac{\Phi(\alpha)-y}{\alpha-z}}$, check

$$e(C, g) \stackrel{?}{=} e(w_z, g^{\alpha}/g^{z})\ e(g, g)^{y}$$

which holds iff $\Phi(z) = y$.

Homomorphic property

If $\tilde{\Phi}(X) = \Phi_1(X) + \Phi_2(X)$, then:

- $\tilde{C} = g^{\Phi_1(\alpha)+\Phi_2(\alpha)} = C_1 \cdot C_2$
- $\tilde{\Phi}(z) = \Phi_1(z) + \Phi_2(z) = y_1 + y_2$ and $\tilde{w}_z = w_z^1 \cdot w_z^2$

Commitments

- **Hash vector:** $[H(f_1), \ldots, H(f_n)] \rightarrow O(\kappa n)$
- **Merkle tree:** (τ, π_j) where τ is the root of the Merkle tree of $[f_1, \ldots, f_n]$ and π_j is the authentication path of $f_j \rightarrow O(\kappa log(n))$
- **Homomorphic fingerprints [HGR07]:** universal hash functions preserving the algebraic structure of the fragments $\rightarrow O(\kappa n)$
- **Polynomial commitments**
  - **KZG [KZG10]:** polynomial commitment (single group element) and constant-sized witnesses to verify evaluations $\rightarrow O(\kappa)$ Drawback: TRUSTED SETUP

Commitments

- **Hash vector:** $[H(f_1), \ldots, H(f_n)] \rightarrow O(\kappa n)$
- **Merkle tree:** (τ, π_j) where τ is the root of the Merkle tree of $[f_1, \ldots, f_n]$ and π_j is the authentication path of $f_j \rightarrow O(\kappa log(n))$
- **Homomorphic fingerprints [HGR07]:** universal hash functions preserving the algebraic structure of the fragments $\rightarrow O(\kappa n)$
- **Polynomial commitments**
  - **KZG [KZG10]:** polynomial commitment (single group element) and constant-sized witnesses to verify evaluations $\rightarrow O(\kappa)$ Drawback: TRUSTED SETUP
  - **Succinct cryptographic proofs (Bulletproofs [Bün+18]):** polynomial commitment (single group element) and logarithmic-sized witnesses to verify evaluations $\rightarrow O(\kappa log(n))$ Gain: NO TRUSTED SETUP

Commitments

- **Hash vector:** $[H(f_1), \ldots, H(f_n)] \rightarrow O(\kappa n)$
- **Merkle tree:** (τ, π_j) where τ is the root of the Merkle tree of $[f_1, \ldots, f_n]$ and π_j is the authentication path of $f_j \rightarrow O(\kappa log(n))$
- **Homomorphic fingerprints [HGR07]:** universal hash functions preserving the algebraic structure of the fragments $\rightarrow O(\kappa n)$
- **Polynomial commitments**
  - **KZG [KZG10]:** polynomial commitment (single group element) and constant-sized witnesses to verify evaluations $\rightarrow O(\kappa)$ Drawback: TRUSTED SETUP
  - **Succinct cryptographic proofs (Bulletproofs [Bün+18]):** polynomial commitment (single group element) and logarithmic-sized witnesses to verify evaluations $\rightarrow O(\kappa log(n))$ Gain: NO TRUSTED SETUP
- **Catalano-Fiore vector commitments [CF13]**: RSA-based, constant-size commitments and constant-size authentication information $\rightarrow O(\kappa)$ Drawback: TRUSTED SETUP + computationally expensive

AVID Improvements

Scheme	Dispersal Cost	Retrieval Cost	Storage Cost (total)	Setup	Commitment
AVID [CT05]	$O(n\|F\| + \kappa n^3)$	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n^2)$	None	Hash vector
AVID-H [CT05]	$O(n\|F\| + \kappa n^2 \log n)$	$O(\|F\| + \kappa n \log n)$	$O(\|F\| + \kappa n \log n)$	None	Merkle tree
AVID-FP [HGR07]	$O(\|F\| + \kappa n^3)$	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n^2)$	None	Homomorphic FP
AVID-2 [ADVZ21]	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n \log n)$	$O(\|F\| + \kappa n \log n)$	CRS	Bulletproofs
AVID-1 [ADVZ21]	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n)$	$O(\|F\| + \kappa n)$	Trusted	KZG
AVID-M [YPAKT22]	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n \log n)$	$O(\|F\| + \kappa n \log n)$	None	Merkle tree
Near-optimal AVID [Alh+22b]	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n)$	$O(\|F\| + \kappa n)$	None	Hash vector
Lower bound	$\Omega(\|F\| + n^2)$	$\Omega(\|F\| + n)$	$\Omega(\|F\|)$	—	—

Table: Comparison of dispersal schemes by communication and storage costs.

Reliable Broadcast

University of Bern

Reliable broadcast

A protocol for **reliable broadcast** allows a sender to broadcast a message F to all servers such that either all or no honest server delivers F, even if the sender is faulty.

Reliable broadcast properties

Validity. If the sender is honest and broadcasts F, then every honest node eventually delivers F.

Agreement. If two honest nodes deliver messages F and F', then $F = F'$.

Integrity. Every honest node delivers at most one message F.

Totality. If an honest node delivers F, then all honest nodes eventually deliver F.

RB $\iff$ AVID

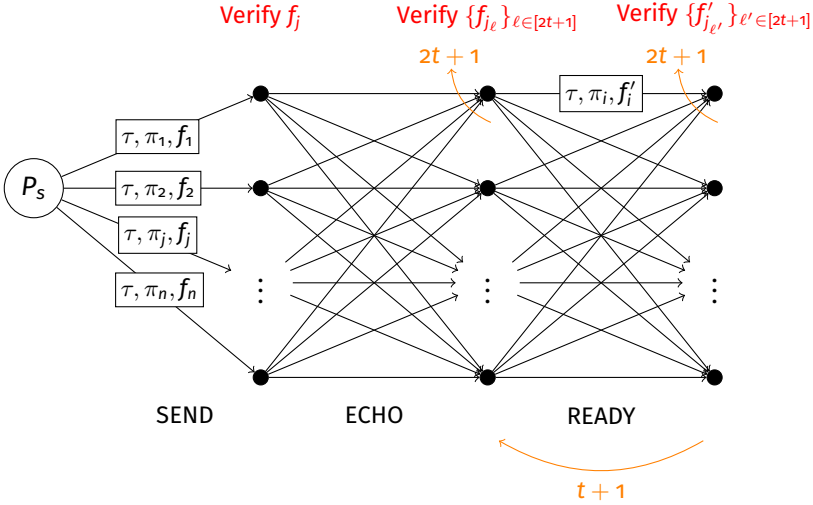
$$\text{RB} \implies \text{AVID}$$

A reliable broadcast protocol can be used to agree on the commitment relative to the file being dispersed.

$$\text{AVID} \implies \text{RB}$$

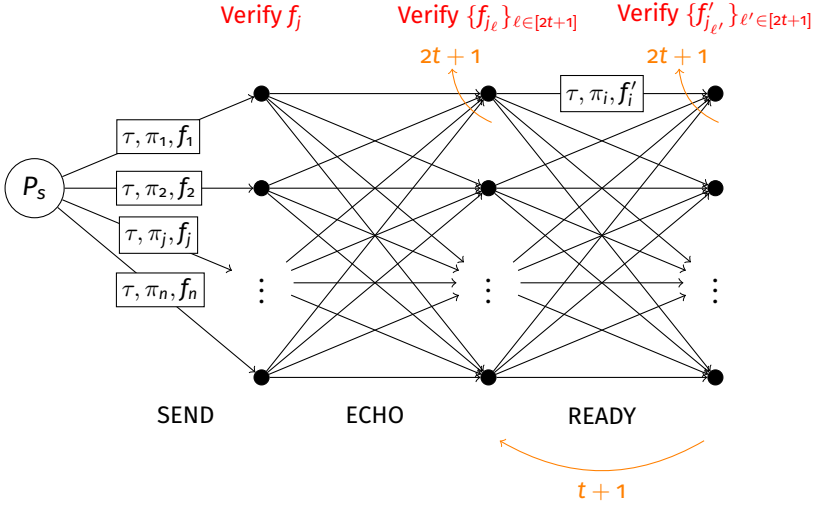
Combine Disperse and Retrive, so that at the end of Disperse, each server retrieves the whole file and stores it (instead of storing a single fragment).

AVID-H



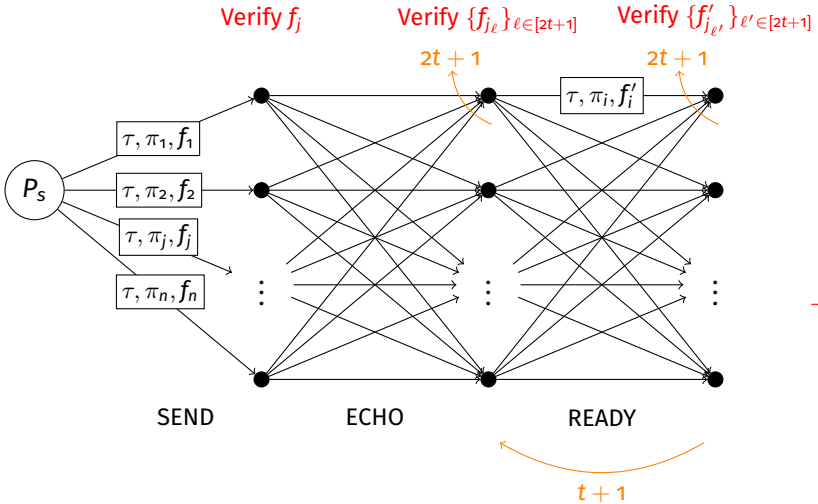
$\rightarrow$ store (τ, π_j, f_j)

Reliable broadcast from AVID-H



$\rightarrow$ store $\bar{F}$

Reliable broadcast from AVID-H



$\rightarrow$ store $\bar{F}$

$\rightarrow O(|F|n + \kappa n^2 log(n))$

RB Improvements

University of Bern

RB Improvements

RB from AVID-H: $O(|F|n + \kappa n^2 log(n))$

RB Improvements

RB from AVID-H: $O(|F|n + \kappa n^2 log(n))$

Dissemination

$\approx 3|F|n$

Lower bound: $|F|n$

Control

$O(\kappa n^2 log(n))$

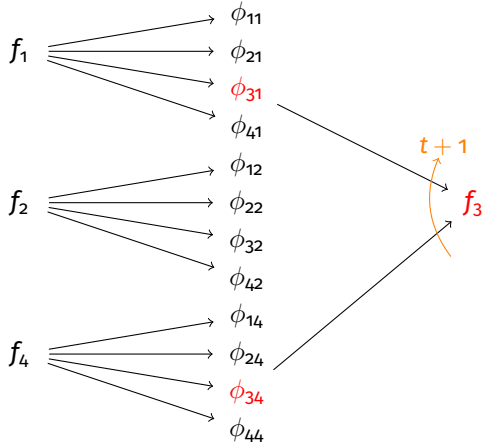
Lower bound: $\Omega(n^2)$

RB Improvements (Dissemination)

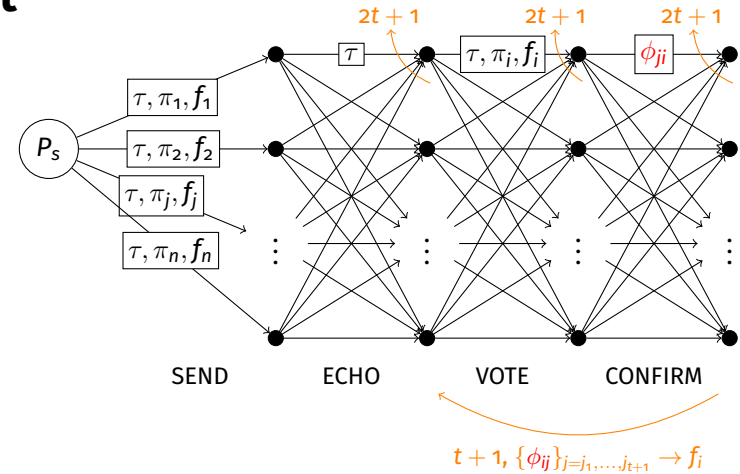
RB protocol	Communication complexity
Bracha	$O(\|F\|n^2)$
RB from AVID-H [CT05]	$\approx 3\|F\|n + O(\kappa n^2 log(n))$
[Loc24]	$\approx 2\|F\|n + O(\kappa n^2 log(n))$
MiniCast [LS25]	$\approx 1.5\|F\|n + O(\kappa n^2 log(n))$

MiniCast

Idea: encode F into fragments $[f_1, \ldots, f_n]$ using an $[n, 2t+1]$-erasure code, and then encode each fragment f_i into *mini-fragments* ϕ_{ji} using an $[n, t+1]$-erasure code.



MiniCast



Communication complexity: $\approx 1.5|F|n + O(\kappa n^2 log(n))$

RB Improvements

RB from AVID-H: $O(|F|n + \kappa n^2 log(n))$

Dissemination

$\approx 1.5|F|n$

Lower bound: $|F|n$

Control

$O(\kappa n^2 log(n))$

Lower bound: $\Omega(n^2)$

RB Improvements (Control)

Scheme	Communication Cost			Assumptions
	Broadcaster	Other node	Total	
Bracha	$O(n\|F\|)$	$O(n\|F\|)$	$O(n^2\|F\|)$	None (error-free)
RB from AVID-H	$O(\|F\| + \kappa n \log n)$	$O(\|F\| + \kappa n \log n)$	$O(n\|F\| + \kappa n^2 \log n)$	Hash
Das et al. [DXR21]	$O(n\|F\| + \kappa n)$	$O(\|F\| + \kappa n)$	$O(n\|F\| + \kappa n^2)$	Hash
CCBRB [Alh+22a]	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n)$	$O(n\|F\| + \kappa n^2)$	Hash
BalCCBRB [Alh+22a]	$O(\|F\| + \kappa n)$	$O(\|F\| + \kappa n)$	$O(n\|F\| + \kappa n^2)$	Hash
Nayak et al. [NRSVX20]	$O(n\|F\| + \kappa n)$	$O(\|F\| + \kappa n)$	$O(n\|F\| + \kappa n^2)$	q-SDH+DBDH+TSetup
SigBRB [Alh+22a]	$O(n\|F\| + \kappa n + n \log n)$	$O(\|F\| + \kappa + n \log n)$	$O(n\|F\| + \kappa n + n^2 \log n)$	ThrSig + TSetup
BalSigBRB [Alh+22a]	$O(\|F\| + \kappa n + n \log n)$	$O(\|F\| + \kappa + n \log n)$	$O(n\|F\| + \kappa n + n^2 \log n)$	ThrSig + TSetup
Lower bound	$\Omega(\|F\| + n)$	$\Omega(\|F\| + n)$	$\Omega(n\|F\| + n^2)$	–

BalCCBRB [Alh+22a]

Parametrized by an external predicate $P()$

BalCCBRB [Alh+22a]

Parametrized by an external predicate $P()$

- The sender only sends fragments of a message F (no commitment), produced using an error-correcting code

BalCCBRB [Alh+22a]

Parametrized by an external predicate $P()$

- The sender only sends fragments of a message F (no commitment), produced using an error-correcting code
- Once nodes receive enough fragments, they decode them to get a message F'

BalCCBRB [Alh+22a]

Parametrized by an external predicate $P()$

- The sender only sends fragments of a message F (no commitment), produced using an error-correcting code
- Once nodes receive enough fragments, they decode them to get a message F'
- Each node verifies that $P(F') = \mathit{true}$

BalCCBRB [Alh+22a]

Parametrized by an external predicate $P()$

- The sender only sends fragments of a message F (no commitment), produced using an error-correcting code
- Once nodes receive enough fragments, they decode them to get a message F'
- Each node verifies that $P(F') = \mathit{true}$
- If $P(F') = \mathit{true}$, node i sends $(\text{READY}, f_j, H(F'))$ to each other node j

BalCCBRB [Alh+22a]

Parametrized by an external predicate $P()$

- The sender only sends fragments of a message F (no commitment), produced using an error-correcting code
- Once nodes receive enough fragments, they decode them to get a message F'
- Each node verifies that $P(F') = \mathit{true}$
- If $P(F') = \mathit{true}$, node i sends $(\text{READY}, f_j, H(F'))$ to each other node j
- Upon receiving enough READY messages for the same $H(F')$, each node decodes the received fragments and outputs the result

BalCCBRB [Alh+22a]

Parametrized by an external predicate $P()$

- The sender only sends fragments of a message F (no commitment), produced using an error-correcting code
- Once nodes receive enough fragments, they decode them to get a message F'
- Each node verifies that $P(F') = \mathit{true}$
- If $P(F') = \mathit{true}$, node i sends $(\text{READY}, f_j, H(F'))$ to each other node j
- Upon receiving enough READY messages for the same $H(F')$, each node decodes the received fragments and outputs the result

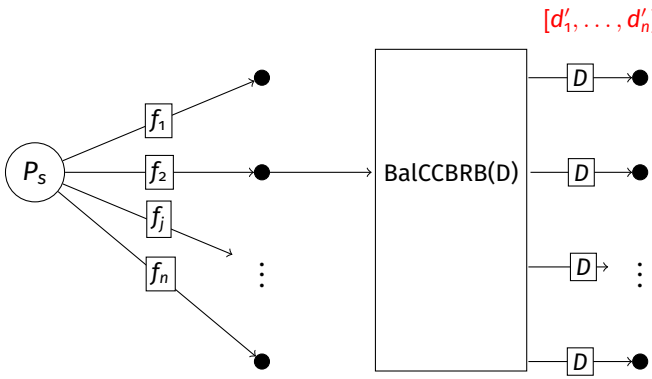
Communication complexity: $O(n|F| + \kappa n^2)$

Near-optimal AVID [Alh+22b]

Idea: use BalCCBRB to agree on the hash vector $D = [H(f_1), \ldots, H(f_n)]$ with external predicate $P()$ s.t. $P(D) = true \iff H(f_i) = D[i]$

Near-optimal AVID [Alh+22b]

Idea: use BalCCBRB to agree on the hash vector $D = [H(f_1), \ldots, H(f_n)]$ with external predicate $P()$ s.t. $P(D) = true \iff H(f_i) = D[i]$

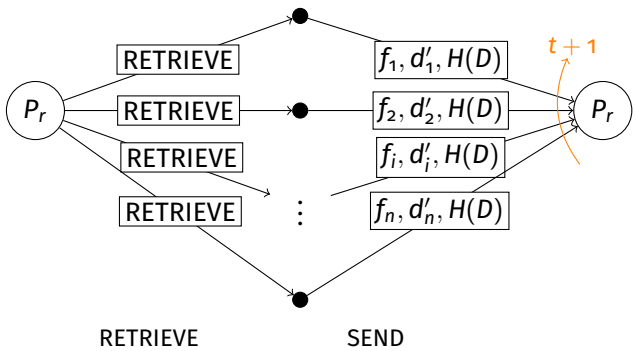


$\rightarrow$ store $[f_i, d'_i, H(D)]$

Near-optimal AVID [Alh+22b]

$F' = \text{Decode}([f_{i_1}, \ldots, f_{i_{t+1}}])$

$D' = \text{Decode}([d'_{i_1}, \ldots, d'_{i_{t+1}}])$



$\rightarrow$ if $H(F'[i]) = D'[i]$ for all $i \in [n]$, output F'

AVID Improvements

Scheme	Dispersal Cost	Retrieval Cost	Storage Cost (total)	Setup	Commitment
AVID [CT05]	$O(n\|F\| + \kappa n^3)$	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n^2)$	None	Hash vector
AVID-H [CT05]	$O(n\|F\| + \kappa n^2 \log n)$	$O(\|F\| + \kappa n \log n)$	$O(\|F\| + \kappa n \log n)$	None	Merkle tree
AVID-FP [HGR07]	$O(\|F\| + \kappa n^3)$	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n^2)$	None	Homomorphic FP
AVID-2 [ADVZ21]	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n \log n)$	$O(\|F\| + \kappa n \log n)$	CRS	Bulletproofs
AVID-1 [ADVZ21]	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n)$	$O(\|F\| + \kappa n)$	Trusted	KZG
AVID-M [YPAKT22]	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n \log n)$	$O(\|F\| + \kappa n \log n)$	None	Merkle tree
Near-optimal AVID [Alh+22b]	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n)$	$O(\|F\| + \kappa n)$	None	Hash vector
Lower bound	$\Omega(\|F\| + n^2)$	$\Omega(\|F\| + n)$	$\Omega(\|F\|)$	—	—

Table: Comparison of dispersal schemes by communication and storage costs.

Conclusion

AVID (and improvements)

Conclusion

AVID (and improvements)

Reliable Broadcast (and improvements)

Conclusion

AVID (and improvements)

Reliable Broadcast (and improvements)

Thank you for your attention!

University of Bern
Institute of Computer Science
Cryptology and Data Security Group

July 10, 2026

References I

[ACLY00] Rudolf Ahlswede et al. “Network information flow”. In: *IEEE Trans. Inf. Theory* 46.4 (2000), pp. 1204–1216. DOI: 10.1109/18.850663.

[ACTZ24] Orestis Alpos et al. “Asymmetric distributed trust”. In: *Distributed Comput.* 37.3 (2024), pp. 247–277. DOI: 10.1007/S00446-024-00469-1.

[ACVZ25] Ignacio Amores-Sesar et al. “DAG-based Consensus with Asymmetric Trust”. In: *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2025, Hotel Las Brisas Huatulco, Huatulco, Mexico, June 16-20, 2025*. Ed. by Alkida Balliu and Fabian Kuhn. ACM, 2025, pp. 151–161. DOI: 10.1145/3732772.3733527.

[ADVZ21] Nicolas Alhaddad et al. “Succinct Erasure Coding Proof Systems”. In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 1500. URL: https://eprint.iacr.org/2021/1500.

References II

[Alh+22a] Nicolas Alhaddad et al. “Balanced Byzantine Reliable Broadcast with Near-Optimal Communication and Improved Computation”. In: *PODC*. ACM, 2022, pp. 399–417.

[Alh+22b] Nicolas Alhaddad et al. “Brief Announcement: Asynchronous Verifiable Information Dispersal with Near-Optimal Communication”. In: *PODC ’22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*. Ed. by Alessia Milani and Philipp Woelfel. ACM, 2022, pp. 418–420. DOI: 10.1145/3519270.3538476.

[BC24] Foteini Baldimtsi and Christian Cachin, eds. *Financial Cryptography and Data Security - 27th International Conference, FC 2023, Bol, Brač, Croatia, May 1-5, 2023, Revised Selected Papers, Part I*. Vol. 13950. Lecture Notes in Computer Science. Springer, 2024. ISBN: 978-3-031-47753-9. DOI: 10.1007/978-3-031-47754-6.

References III

[BNNP25] Dan Boneh et al. “Data Availability Sampling with Repair”. In: *IACR Cryptol. ePrint Arch.* (2025), p. 1414. URL: https://eprint.iacr.org/2025/1414.

[Bün+18] Benedikt Bünz et al. “Bulletproofs: Short Proofs for Confidential Transactions and More”. In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2018, pp. 315–334.

[Cel26] Celestia Labs. *Celestia – Fibre Optic Performance for Millisecond Markets*. https://celestia.org. 2026.

[CF13] Dario Catalano and Dario Fiore. “Vector Commitments and Their Applications”. In: *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. Lecture Notes in Computer Science. Springer, 2013, pp. 55–72. DOI: 10.1007/978-3-642-36362-7_5.

References IV

[CGR11] Christian Cachin et al. *Introduction to Reliable and Secure Distributed Programming (2. ed.)* Springer, 2011. ISBN: 978-3-642-15259-7. DOI: 10.1007/978-3-642-15260-3.

[Civ+24] Pierre Civit et al. "Efficient Signature-Free Validated Agreement". In: *38th International Symposium on Distributed Computing, DISC 2024, Madrid, Spain, October 28 - November 1, 2024*. Ed. by Dan Alistarh. Vol. 319. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 14:1–14:23. DOI: 10.4230/LIPICS.DISC.2024.14.

[CO18] Christian Cachin and Olga Ohrimenko. "Verifying the consistency of remote untrusted services with conflict-free operations". In: *Inf. Comput.* 260 (2018), pp. 72–88. DOI: 10.1016/J.IC.2018.03.004.

References V

[CP23] Benjamin Y. Chan and Rafael Pass. “Simplex Consensus: A Simple and Fast Consensus Protocol”. In: *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part IV*. Ed. by Guy N. Rothblum and Hoeteck Wee. Vol. 14372. Lecture Notes in Computer Science. Springer, 2023, pp. 452–479. DOI: 10.1007/978-3-031-48624-1_17.

[CT05] Christian Cachin and Stefano Tessaro. “Asynchronous Verifiable Information Dispersal”. In: *24th IEEE Symposium on Reliable Distributed Systems (SRDS 2005),26-28 October 2005, Orlando, FL, USA*. IEEE Computer Society, 2005, pp. 191–202. DOI: 10.1109/RELDIS.2005.9.

[Dan+25] George Danezis et al. “Walrus: An Efficient Decentralized Storage Network”. In: *CoRR* abs/2505.05370 (2025). arXiv: 2505.05370. DOI: 10.48550/ARXIV.2505.05370.

References VI

[DGWWR10] Alexandros G. Dimakis et al. “Network coding for distributed storage systems”. In: *IEEE Trans. Inf. Theory* 56.9 (2010), pp. 4539–4551. DOI: 10.1109/TIT.2010.2054295.

[DRZ18] Sisi Duan et al. “BEAT: Asynchronous BFT Made Practical”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. Ed. by David Lie et al. ACM, 2018, pp. 2028–2041. DOI: 10.1145/3243734.3243812.

[DXR21] Sourav Das et al. “Asynchronous Data Dissemination and its Applications”. In: *CCS*. ACM, 2021, pp. 2705–2721.

[Eig26] Eigen Labs. *EigenDA – State of the Art Data Availability*. https://www.eigenda.xyz. 2026.

[FBW06] Christina Fragouli et al. “Network coding: an instant primer”. In: *Comput. Commun. Rev.* 36.1 (2006), pp. 63–68. DOI: 10.1145/1111322.1111337.

References VII

[FLMT25] Hanwen Feng et al. “Faster Hash-based Multi-valued Validated Asynchronous Byzantine Agreement”. In: *55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2025, Naples, Italy, June 23-26, 2025*. IEEE, 2025, pp. 303–316. DOI: 10.1109/DSN64029.2025.00040.

[Gao+22] Yingzi Gao et al. “Dumbo-NG: Fast Asynchronous BFT Consensus with Throughput-Oblivious Latency”. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*. Ed. by Heng Yin et al. ACM, 2022, pp. 1187–1201. DOI: 10.1145/3548606.3559379.

[GHSY12] Parikshit Gopalan et al. “On the Locality of Codeword Symbols”. In: *IEEE Trans. Inf. Theory* 58.11 (2012), pp. 6925–6934. DOI: 10.1109/TIT.2012.2208937.

[Gor+25] Guy Goren et al. “Shelby: Decentralized Storage Designed to Serve”. In: *CoRR* abs/2506.19233 (2025). arXiv: 2506.19233. DOI: 10.48550/ARXIV.2506.19233.

References VIII

[GRKM25] Moritz Grundei et al. “From Indexing to Coding: A New Paradigm for Data Availability Sampling”. In: *CoRR* abs/2509.21586 (2025). arXiv: 2509.21586. DOI: 10.48550/ARXIV.2509.21586.

[HGR07] James Hendricks et al. “Verifying distributed erasure-coded data”. In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing, PODC 2007, Portland, Oregon, USA, August 12-15, 2007*. Ed. by Indranil Gupta and Roger Wattenhofer. ACM, 2007, pp. 139–146. DOI: 10.1145/1281100.1281122.

[Ho+06] Tracey Ho et al. “A Random Linear Network Coding Approach to Multicast”. In: *IEEE Trans. Inf. Theory* 52.10 (2006), pp. 4413–4430. DOI: 10.1109/TIT.2006.881746.

[HSW24a] Mathias Hall-Andersen et al. “Foundations of Data Availability Sampling”. In: *IACR Commun. Cryptol.* 1.4 (2024), p. 34. DOI: 10.62056/A09QUDHDJ.

References IX

[HSW24b] Mathias Hall-Andersen et al. “FRIDA: Data Availability Sampling from FRI”. In: *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VI*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14925. Lecture Notes in Computer Science. Springer, 2024, pp. 289–324. DOI: 10.1007/978-3-031-68391-6_9.

[JB25] Mohammad Mussadiq Jalalzai and Kushal Babel. “MonadBFT: Fast, Responsive, Fork-Resistant Streamlined Consensus”. In: *CoRR* abs/2502.20692 (2025). arXiv: 2502.20692. DOI: 10.48550/ARXIV.2502.20692.

References X

[KZG10] Aniket Kate et al. “Constant-Size Commitments to Polynomials and Their Applications”. In: *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*. Ed. by Masayuki Abe. Vol. 6477. Lecture Notes in Computer Science. Springer, 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11.

[Lab18] I. Storj Labs. *Storj: A decentralized cloud storage network framework*. [Online]. 2018.

[LLTW20] Yuan Lu et al. “Dumbo-MVBA: Optimal Multi-Valued Validated Asynchronous Byzantine Agreement, Revisited”. In: *PODC ’20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*. Ed. by Yuval Emek and Christian Cachin. ACM, 2020, pp. 129–138. DOI: 10.1145/3382734.3405707.

References XI

[Loc24] Thomas Locher. “Byzantine Reliable Broadcast with Low Communication and Time Complexity”. In: *OPODIS*. Vol. 324. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 16:1–16:17.

[LS25] Thomas Locher and Victor Shoup. “MiniCast: Minimizing the Communication Complexity of Reliable Broadcast”. In: *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part V*. Ed. by Serge Fehr and Pierre-Alain Fouque. Lecture Notes in Computer Science. Springer, 2025, pp. 96–115. DOI: 10.1007/978-3-031-91092-0_4.

[Lub02] Michael Luby. “LT Codes”. In: *43rd Symposium on Foundations of Computer Science, FOCS 2002, Vancouver, BC, Canada, November 16-19, 2002, Proceedings*. IEEE Computer Society, 2002, p. 271. DOI: 10.1109/SFCS.2002.1181950.

References XII

[Mac05] David JC MacKay. “Fountain codes”. In: *IEE Proceedings-Communications* 152.6 (2005), pp. 1062–1068. URL: https://docs.switzernet.com/people/emin-gabrielyan/060123-capillary-aron-article/ref/MacKay05.pdf.

[NNT22] Kamilla Nazirkhanova et al. “Information Dispersal with Provable Retrievability for Rollups”. In: *Proceedings of the 4th ACM Conference on Advances in Financial Technologies, AFT 2022, Cambridge, MA, USA, September 19-21, 2022*. Ed. by Maurice Herlihy and Neha Narula. ACM, 2022, pp. 180–197. DOI: 10.1145/3558535.3559778.

[NRSVX20] Kartik Nayak et al. “Improved Extension Protocols for Byzantine Broadcast and Agreement”. In: *DISC*. Vol. 179. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 28:1–28:17.

References XIII

[PD12] Dimitris S. Papailiopoulos and Alexandros G. Dimakis. “Locally Repairable Codes”. In: *CoRR* abs/1206.3804 (2012). URL: http://arxiv.org/abs/1206.3804. arXiv: 1206.3804.

[PD20] Yiannis Psaras and David Dias. “The InterPlanetary File System and the Filecoin Network”. In: *50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020 - Supplemental Volume*. IEEE, 2020, p. 80. DOI: 10.1109/DSN-S50200.2020.00043.

[PSX24] Zengyu Pang et al. “Asynchronous verifiable information dispersal protocol of achieving optimal communication”. In: *Comput. Networks* 247 (2024), p. 110470. DOI: 10.1016/J.COMNET.2024.110470.

[Q K25] R. Wattenhofer Q. Kniep J. Sliwinski. *Solana Alpenglow Consensus*. Alpenglow White Paper. 2025.

References XIV

[Rab89] Michael O. Rabin. “Efficient dispersal of information for security, load balancing, and fault tolerance”. In: *J. ACM* 36.2 (1989), pp. 335–348. DOI: 10.1145/62044.62050.

[RS60] I. S. Reed and G. Solomon. “Polynomial Codes Over Certain Finite Fields”. In: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304. eprint: https://doi.org/10.1137/0108018. DOI: 10.1137/0108018.

[RSK10] K. V. Rashmi et al. “Optimal Exact-Regenerating Codes for Distributed Storage at the MSR and MBR Points via a Product-Matrix Construction”. In: *CoRR* abs/1005.4178 (2010). URL: http://arxiv.org/abs/1005.4178. arXiv: 1005.4178.

References XV

[She+25] Zhirong Shen et al. “A Survey of the Past, Present, and Future of Erasure Coding for Storage Systems”. In: *ACM Trans. Storage* 21.1 (2025), 4:1–4:39. DOI: 10.1145/3708994.

[Sho06] Amin Shokrollahi. “Raptor codes”. In: *IEEE Trans. Inf. Theory* 52.6 (2006), pp. 2551–2567. DOI: 10.1109/TIT.2006.874390.

[Sho24] Victor Shoup. “Sing a Song of Simplex”. In: *38th International Symposium on Distributed Computing, DISC 2024, Madrid, Spain, October 28 - November 1, 2024*. Ed. by Dan Alistarh. Vol. 319. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 37:1–37:22. DOI: 10.4230/LIPICS.DISC.2024.37.

[TB14] Itzhak Tamo and Alexander Barg. “A Family of Optimal Locally Recoverable Codes”. In: *IEEE Trans. Inf. Theory* 60.8 (2014), pp. 4661–4676. DOI: 10.1109/TIT.2014.2321280.

References XVI

[Vaj+18] Myna Vajha et al. “Clay Codes: Moulding MDS Codes to Yield an MSR Code”. In: *16th USENIX Conference on File and Storage Technologies, FAST 2018, Oakland, CA, USA, February 12-15, 2018*. Ed. by Nitin Agrawal and Raju Rangaswami. USENIX Association, 2018, pp. 139–154. URL: https://www.usenix.org/conference/fast18/presentation/vajha.

[WDBU19] S. Williams et al. *Arweave: A protocol for economically sustainable information permanence*. Arweave Yellow Paper. 2019.

[Yan+24] Changlin Yang et al. “Scaling Blockchains with Error Correction Codes: A Survey on Coded Blockchains”. In: *ACM Comput. Surv.* 56.6 (2024), 139:1–139:33. DOI: 10.1145/3637224.

References XVII

[YPAKT22] Lei Yang et al. “DispersedLedger: High-Throughput Byzantine Consensus on Variable Bandwidth Networks”. In: *19th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2022, Renton, WA, USA, April 4-6, 2022*. Ed. by Amar Phanishayee and Vyas Sekar. USENIX Association, 2022, pp. 493–512. URL: https://www.usenix.org/conference/nsdi22/presentation/yang.

gib

Applications in Consensus & Blockchains

Tutorial on Erasure Coding in Distributed Protocols

Mariarosaria Barbaraci

Vivien Bammert
Annalisa Cimatti
Christian Cachin

University of Bern
Institute of Computer Science
Cryptology and Data Security Group

July 10, 2026

What did we learn?

Under Byzantine assumptions and an asynchronous communication model we can define a primitive, *Asynchronous Verifiable Information Dispersal* (AVID), which leveraging erasure codes achieves the following properties:

What did we learn?

Under Byzantine assumptions and an asynchronous communication model we can define a primitive, *Asynchronous Verifiable Information Dispersal* (AVID), which leveraging erasure codes achieves the following properties:

- Termination
- Agreement
- Availability
- Correctness

What did we learn?

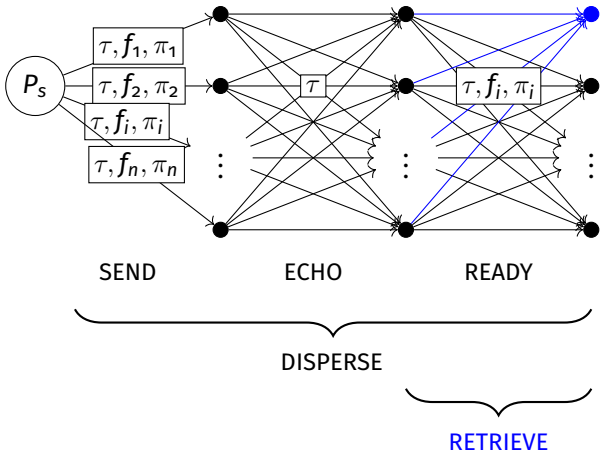
The main goal of AVID is to minimize storage complexity.

Optimizations concentrated also on reducing communication complexity.

	Disperse	Retrieve	Storage
AVID [CT05]	$O(n\|F\| + \kappa n^3)$	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n^2)$
Near-optimal AVID [Alh+22b]	$O(\|F\| + \kappa n^2)$	$O(\|F\| + \kappa n)$	$O(\|F\| + \kappa n)$
Lower bound	$\Omega(\|F\| + n^2)$	$\Omega(\|F\| + n^2)$	$\Omega(\|F\| + n^2)$

What did we learn?

Using the two protocols of AVID, *Disperse* and *Retrieve*, we can always construct a *Byzantine Reliable Broadcast*.



Outline

- Discussion on properties
- *Weaker* primitives exploiting erasure codes with fragment verifiability
- Integration in consensus protocols
- Applications to Data Availability Sampling (DAS)

Discussion on properties

1. Retrieval from ANY sufficiently large set

Discussion on properties

1. Retrieval from ANY sufficiently large set (Implementation-dependant)
   - Possible when fragments are echoed;
   - Every server holds its fragment after dispersal.

Discussion on properties

1. Retrieval from ANY sufficiently large set (Implementation-dependant)
   - Possible when fragments are echoed;
   - Every server holds its fragment after dispersal.
2. Termination guarantees

Discussion on properties

1. Retrieval from ANY sufficiently large set (Implementation-dependant)
   - Possible when fragments are echoed;
   - Every server holds its fragment after dispersal.
2. Termination guarantees (Agreement property)
   - Every server completes *dispersal*;
   - A server is informed a dispersal instance occurred in the network.

Discussion on properties

1. Retrieval from ANY sufficiently large set (Implementation-dependant)
   - Possible when fragments are echoed;
   - Every server holds its fragment after dispersal.
2. Termination guarantees (Agreement property)
   - Every server completes *dispersal*;
   - A server is informed a dispersal instance occurred in the network.
3. Retrievability guarantees

Discussion on properties

1. Retrieval from ANY sufficiently large set (Implementation-dependant)
   - Possible when fragments are echoed;
   - Every server holds its fragment after dispersal.
2. Termination guarantees (Agreement property)
   - Every server completes *dispersal*;
   - A server is informed a dispersal instance occurred in the network.
3. Retrievability guarantees (Commitment or implementation-dependant)
   - Fragments can be verified independently,
   - Verification is with respect to the content itself *before* reconstruction.
   - Provides resilience against malicious clients.

Discussion on properties

1. Retrieval from ANY sufficiently large set (Implementation-dependant)
   - Possible when fragments are echoed;
   - Every server holds its fragment after dispersal.
2. Termination guarantees (Agreement property)
   - Every server completes *dispersal*;
   - A server is informed a dispersal instance occurred in the network.
3. Retrievability guarantees (Commitment or implementation-dependant)
   - Fragments can be verified independently,
   - Verification is with respect to the content itself *before* reconstruction.
   - Provides resilience against malicious clients.
4. Certificate of retrievability
   - A third party can verify data is *available*.

Applications to Consensus

DispersedLedger

- DisperseLedger [YPAKT22] improves on a HoneyBadgerBFT.
- Asynchronous BFT protocol implementing *state machine replication (SMR)*.
- It decouples the *agreement* on transaction ordering from *downloading* the full content.
- It uses a modified version of AVID to increase throughput.

AVID-M

- Based on the construction of AVID-H [CT05] with Reed-Solomon codes and Merkle trees.

AVID-M

- Based on the construction of AVID-H [CT05] with Reed-Solomon codes and Merkle trees.
- At the end of Disperse(B), each server outputs and agrees on the *root* of the Merkle tree.

AVID-M

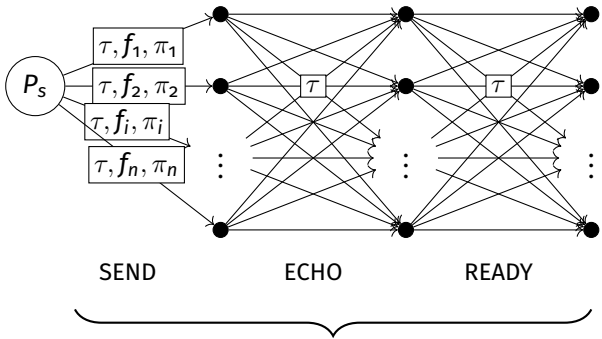
- Based on the construction of AVID-H [CT05] with Reed-Solomon codes and Merkle trees.
- At the end of Disperse(B), each server outputs and agrees on the *root* of the Merkle tree.
- Fragments are sent just in the first communication step, from the sender to the server.

AVID-M

- Based on the construction of AVID-H [CT05] with Reed-Solomon codes and Merkle trees.
- At the end of Disperse(B), each server outputs and agrees on the *root* of the Merkle tree.
- Fragments are sent just in the first communication step, from the sender to the server.
- Only later, each server sends its fragments in the retrieve protocol.

AVID-M

- Based on the construction of AVID-H [CT05] with Reed-Solomon codes and Merkle trees.
- At the end of DISPERSE(B), each server outputs and agrees on the *root* of the Merkle tree.
- Fragments are sent just in the first communication step, from the sender to the server.
- Only later, each server sends its fragments in the RETRIEVE protocol.



AVID-M

- Based on the construction of AVID-H [CT05] with Reed-Solomon codes and Merkle trees.
- At the end of Disperse(B), each server outputs and agrees on the *root* of the Merkle tree.
- Fragments are sent just in the first communication step, from the sender to the server.
- Only later, each server sends its fragments in the retrieve protocol.

AVID-M

- Based on the construction of AVID-H [CT05] with Reed-Solomon codes and Merkle trees.
- At the end of Disperse(B), each server outputs and agrees on the *root* of the Merkle tree.
- Fragments are sent just in the first communication step, from the sender to the server.
- Only later, each server sends its fragments in the retrieve protocol.

Does this construction ensure all AVID properties?

AVID-M

- Based on the construction of AVID-H [CT05] with Reed-Solomon codes and Merkle trees.
- At the end of DISPERSE(B), each server outputs and agrees on the *root* of the Merkle tree.
- Fragments are sent just in the first communication step, from the sender to the server.
- Only later, each server sends its fragments in the RETRIEVE protocol.

Does this construction ensure all AVID properties? Yes

Let's think about Correctness ...

AVID-M

- Based on the construction of AVID-H [CT05] with Reed-Solomon codes and Merkle trees.
- At the end of DISPERSE(B), each server outputs and agrees on the *root* of the Merkle tree.
- Fragments are sent just in the first communication step, from the sender to the server.
- Only later, each server sends its fragments in the RETRIEVE protocol.

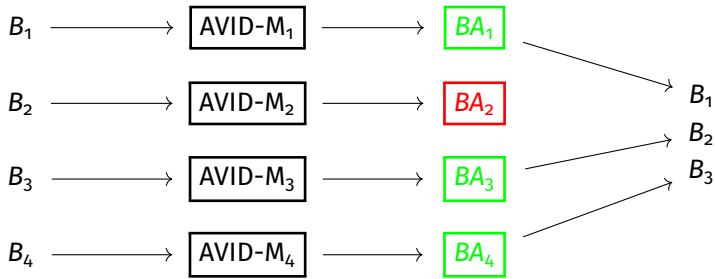
Does this construction ensure all AVID properties? Yes

A server re-encodes a decoded file F', computes the associated Merkle tree. If the new root $r' \neq r$, the output is $\perp$, otherwise, the output is $F' = F$.

This *consistently* occurs in every replica that invokes RETRIEVE.

DispersedLedger – Protocol

Idea: All successfully dispersed blocks are committed to in the next epoch, and from then on any node can retrieve the block. The fact that nodes can retrieve blocks asynchronously at any time makes the protocol faster (compared to, e.g., HoneyBadger).



Dumbo-MVBA

Dumbo-MVBA [LLTW20] optimizes the *communication complexity* of *Multi-value Validated Byzantine Agreement* (MVBA), which originally had a communication of

$$O(n^2|v| + \kappa n^2 + n^3)$$

Dumbo-MVBA

Dumbo-MVBA [LLTW20] optimizes the *communication complexity* of *Multi-value Validated Byzantine Agreement* (MVBA), which originally had a communication of

$$O(n^2|v| + \kappa n^2 + n^3)$$

to the near-optimal

$$O(n|v| + \kappa n^2)$$

Dumbo-MVBA

Dumbo-MVBA [LLTW20] optimizes the *communication complexity* of *Multi-value Validated Byzantine Agreement* (MVBA), which originally had a communication of

$$O(n^2|v| + \kappa n^2 + n^3)$$

to the near-optimal

$$O(n|v| + \kappa n^2)$$

Remainder: The *lower bound* for communication complexity in asynchronous Byzantine agreement is $\Omega(n|v| + n^2)$ for $n > 3f$ and $f \in \Theta(n)$.

Dumbo-MVBA

Asynchronous Provable Dispersal Broadcast (APDB)

- Introduces a *weaker* version of AVID.

Dumbo-MVBA

Asynchronous Provable Dispersal Broadcast (APDB)

- Introduces a *weaker* version of AVID.
- It provides a *provable* dispersal (PD) protocol, and

Dumbo-MVBA

Asynchronous Provable Dispersal Broadcast (APDB)

- Introduces a *weaker* version of AVID.
- It provides a *provable* dispersal (PD) protocol, and
- a *recast* protocol (RC).

Dumbo-MVBA

Asynchronous Provable Dispersal Broadcast (APDB)

- Introduces a *weaker* version of AVID.
- It provides a *provable* dispersal (PD) protocol, and
- a *recast* protocol (RC).
- APDB ensures the following properties:

Dumbo-MVBA

Asynchronous Provable Dispersal Broadcast (APDB)

- Introduces a *weaker* version of AVID.
- It provides a *provable* dispersal (PD) protocol, and
- a *recast* protocol (RC).
- APDB ensures the following properties:
  - **Termination**: If a sender P_s is honest and all the honest parties activate PD for ID (without calling *abandon*), then each honest server would output *store* and a valid *lock* for ID; moreover, the sender outputs a valid *done*.
  - **Recast-ability**
  - **Provability**
  - **Abandon-ability**

Dumbo-MVBA

Asynchronous Provable Dispersal Broadcast (APDB)

- Introduces a *weaker* version of AVID.
- It provides a *provable* dispersal (PD) protocol, and
- a *recast* protocol (RC).
- APDB ensures the following properties:
  - **Termination**
  - **Recast-ability**: If all honest server invoke RC for ID and at least one honest inputs a valid lock, then: (i) all honest servers recover a common value; (ii) if the sender dispersed a value *v* and it has not been corrupted until an honest server had output a valid lock, then all honest servers recover *v*. (Recall AVID Correctness)
  - **Provability**
  - **Abandon-ability**

Dumbo-MVBA

Asynchronous Provable Dispersal Broadcast (APDB)

- Introduces a *weaker* version of AVID.
- It provides a *provable* dispersal (PD) protocol, and
- a *recast* protocol (RC).
- APDB ensures the following properties:
  - **Termination**
  - **Recast-ability**
  - **Provability**: If the sender of a PD with ID outputs a valid done, then at least $f + 1$ honest servers output a valid lock.
  - **Abandon-ability**

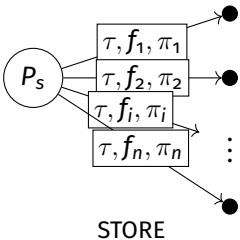
Dumbo-MVBA

Asynchronous Provable Dispersal Broadcast (APDB)

- Introduces a *weaker* version of AVID.
- It provides a *provable* dispersal (PD) protocol, and
- a *recast* protocol (RC).
- APDB ensures the following properties:
  - **Termination**
  - **Recast-ability**
  - **Provability**
  - **Abandon-ability**: If every server cannot produce a valid block for ID and at least $f + 1$ honest servers invoke abandon, no server would deliver a valid lock for ID.

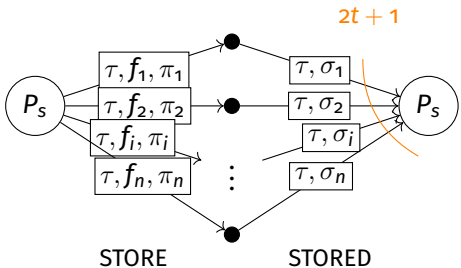
Dumbo – APDB

Provable Dispersal (PD)



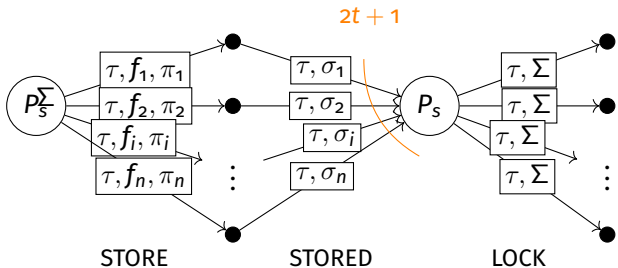
Dumbo – APDB

Provable Dispersal (PD)



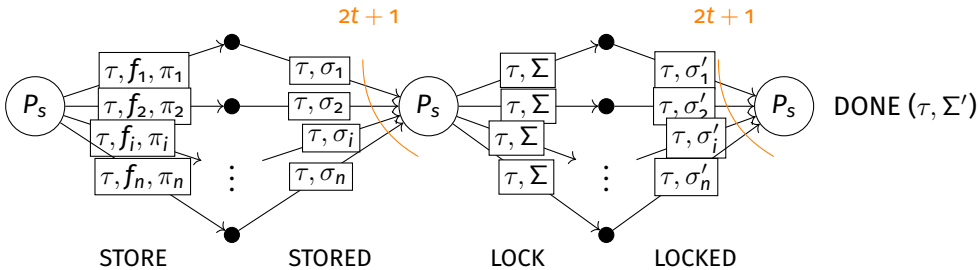
Dumbo – APDB

Provable Dispersal (PD)



Dumbo – APDB

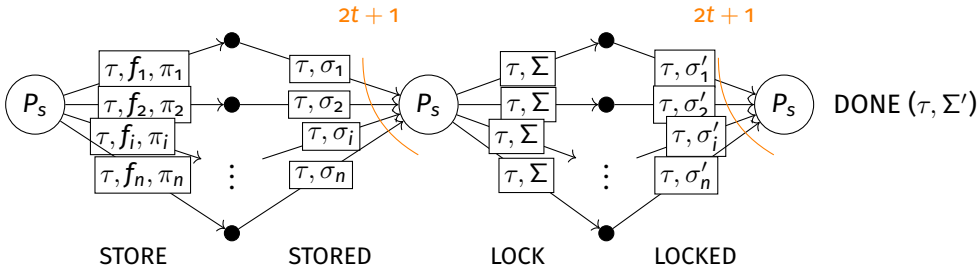
Provable Dispersal (PD)



A *lock proof* consists of a *quorum of signatures* (Σ) from the parties attesting they have received and verified a fragment of the dispersed value.

Dumbo – APDB

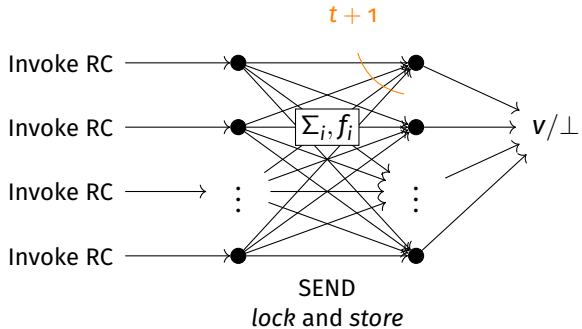
Provable Dispersal (PD)



Communication complexity: $O(|F| + \kappa n)$

DUMBO–APDB

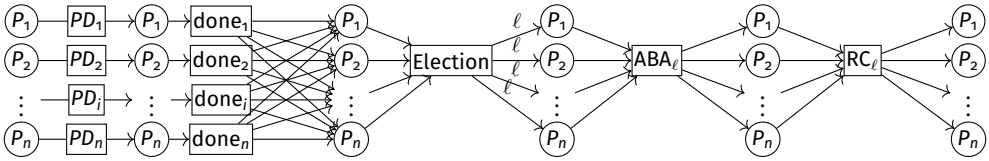
Recast (RC)



Communication complexity: $O(n|F| + \kappa n^2)$

Dumbo-MVBA

Main protocol



DISPERSE and LOCK $2f + 1$ values

AGREE and RECAST value from P_ℓ

Communication complexity: $O(n|F| + \kappa n^2)$

DispersedSimplex

Simplex [CP23] is a BFT atomic broadcast protocol that

- It ensures **safety**,
- in periods of network synchrony, **liveness**.

Simplex allows a block to be finalized in two rounds of voting or otherwise skipped.

DispersedSimplex

Simplex [CP23] is a BFT atomic broadcast protocol that

- It ensures **safety**,
- in periods of network synchrony, **liveness**.

Simplex allows a block to be finalized in two rounds of voting or otherwise skipped.

DispersedSimplex [Sho24]:

- it applies *information dispersal* techniques to leader dissemination; and
- it introduces *stable leaders*.

DispersedSimplex – Erasure code

- The leader disseminates a payload $|F|$ with a $(n - 2t, n)$-erasure code.
- Fragment verifiability uses Merkle trees.
- (f_i, π_i) is a *certified fragment at position i*.
- A proposal is identified by a *tag* $\tau := (|F|, r)$.
- Every fragment has size $\frac{|F|}{n-2t}$.

DispersedSimplex – Data structures

Block (B) is composed of the triple (v, v', τ), where v is the current slot, v' the parent slot and τ an identification tag.

DispersedSimplex – Data structures

Block (*B*) is composed of the triple (v, v', τ), where v is the current slot, v' the parent slot and τ an identification tag.

Support share is issued by a party p_i on a block B upon receiving a valid *certified fragment* (f_i, π_i). The share includes f_i, π_i and a valid *signature share* σ_i on an object Support(B).

DispersedSimplex – Data structures

Block (*B*) is composed of the triple (v, v', τ), where v is the current slot, v' the parent slot and τ an identification tag.

Support share is issued by a party p_i on a block *B* upon receiving a valid *certified fragment* (f_i, π_i). The share includes f_i, π_i and a valid *signature share* σ_i on an object Support(B).

Commit share is issued by a party p_i once the block *B* has been added to the local *block tree*, tracking all the blocks supported so far. The share includes a valid *signature share* σ_i on Commit(B).

DispersedSimplex – Data structures

Block (B) is composed of the triple (v, v', τ), where v is the current slot, v' the parent slot and τ an identification tag.

Support share is issued by a party p_i on a block B upon receiving a valid *certified fragment* (f_i, π_i). The share includes f_i, π_i and a valid *signature share* σ_i on an object Support(B).

Commit share is issued by a party p_i once the block B has been added to the local *block tree*, tracking all the blocks supported so far. The share includes a valid *signature share* σ_i on Commit(B).

Complain share is issued by a party p_i if by the time of Δ_t it has not yet sent a commit share. The share includes a valid *signature share* σ_i on Complain(B).

DispersedSimplex – Data structures

Block (B) is composed of the triple (v, v', τ), where v is the current slot, v' the parent slot and τ an identification tag.

Support share is issued by a party p_i on a block B upon receiving a valid *certified fragment* (f_i, π_i). The share includes f_i, π_i and a valid *signature share* σ_i on an object Support(B).

Commit share is issued by a party p_i once the block B has been added to the local *block tree*, tracking all the blocks supported so far. The share includes a valid *signature share* σ_i on Commit(B).

Complain share is issued by a party p_i if by the time of Δ_t it has not yet sent a commit share. The share includes a valid *signature share* σ_i on Complain(B).

Certificates for each share type are objects of the form SuppCert(B, σ), CommitCert(B, σ), and ComplainCert(B, σ), respectively. They can be assembled upon receiving $n - t$ shares of the corresponding type. Upon first assembling/receiving the certificate, a replica broadcasts it.

DispersedSimplex – Block tree

Each replica adds a block B to the local data structure, *block tree*, if **each** of the following conditions holds:

1. the replica has received a *support certificate* for the block B;
2. the parent block B' is present in the block tree, unless B is the genesis;
3. the replica has received $n - 2f$ valid support shares, thereby enabling successful reconstruction of block B; and
4. additionally, $B \neq \perp$ is valid against a predicate decided by the application layer.

DispersedSimplex – Block tree

Each replica adds a block B to the local data structure, *block tree*, if **each** of the following conditions holds:

1. the replica has received a *support certificate* for the block B;
2. the parent block B' is present in the block tree, unless B is the genesis;
3. the replica has received $n - 2f$ valid support shares, thereby enabling successful reconstruction of block B; and
4. additionally, $B \neq \perp$ is valid against a predicate decided by the application layer.

How do we check for successful reconstruction?

DispersedSimplex – Block tree

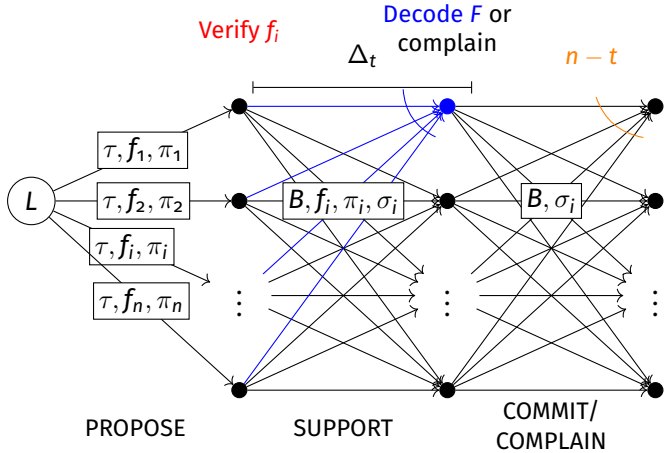
Each replica adds a block B to the local data structure, *block tree*, if **each** of the following conditions holds:

1. the replica has received a *support certificate* for the block B;
2. the parent block B' is present in the block tree, unless B is the genesis;
3. the replica has received $n - 2f$ valid support shares, thereby enabling successful reconstruction of block B; and
4. additionally, $B \neq \perp$ is valid against a predicate decided by the application layer.

How do we check for successful reconstruction?

A server re-encodes a decoded file F', computes the associated Merkle tree. If the new root $r' \neq r$, the output is $\perp$, otherwise, the output is $F' = F$.

DispersedSimplex – Protocol



DispersedSimplex – Observations

- The timeout Δ_t is used to send a *complaint* share in case a correct party hasn't received a *support certificate* and the fragments to reconstruct the block yet.
- No correct process issues both a complaint and a commit share for the same slot.

Kudzu

- Based on DispersedSimplex;
- Introduces the fast path, i.e., assumes a network with $n \geq 3t + 2p + 1$, where if up to t nodes are corrupt, and up to p can be unresponsive;
- Uses an $(t + p + 1)$-erasure code;
- Upon receiving $n - t - p$ first (or support) votes, a replica can create a *notarization certificate* and send the finalization vote. However, upon receiving the first $n - p$ votes, it can *fast* finalize a block.
- A notarization certificate assembled by $n - t - p$ partial signatures encompasses the contribution of $n - 2t - p$ honest replicas. Given the lower bound of n, $n - 2t - p \geq t + p + 1$;
- Only a block B, and not a timeout, can be fast finalized;

Solana's Alpenglow

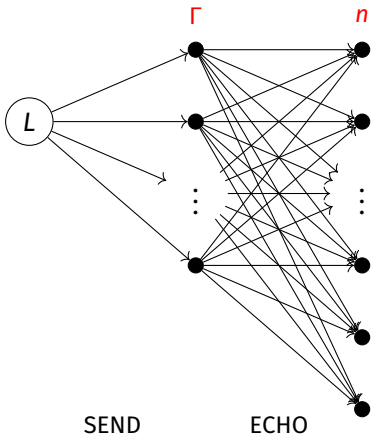
Goal: reduce the leader bandwidth bottleneck by leveraging the total available bandwidth at each node and achieving optimal **throughput** asymptotically.

Model: permissionless, i.e., the number of nodes in the network varies between **epochs**.

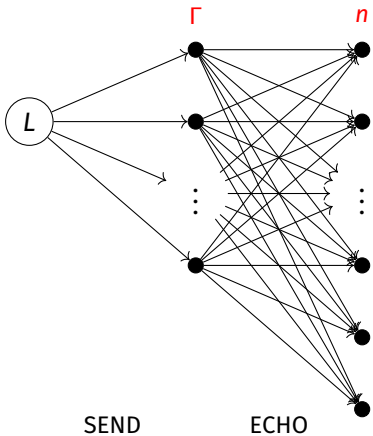
Fault-tolerance: stake-based, i.e., Byzantine node control less than 20% of the stake. Additionally, *other* nodes with up to 20% can crash.

The leader of an epoch L disseminates a block B to a subset of $\Gamma < n$ nodes. The reconstruction threshold of the erasure codes is set to a value γ that satisfies: $\frac{\Gamma}{\gamma} > \frac{5}{3}$.

Rotor

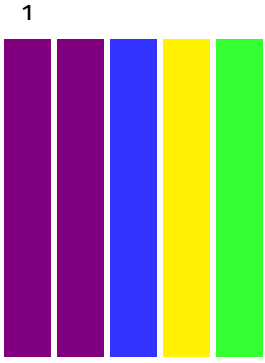


Rotor



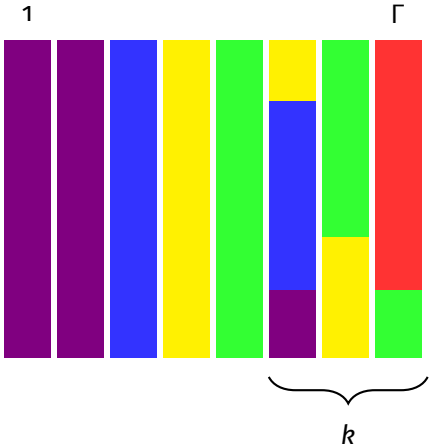
- The overhead of the code follows the fault-tolerant assumption, and, if the leader is honest, for $\gamma \to \infty$, with probability one, a slice is received correctly.
- As overhead approaches infinity, the latency approaches δ.

Smart sampling



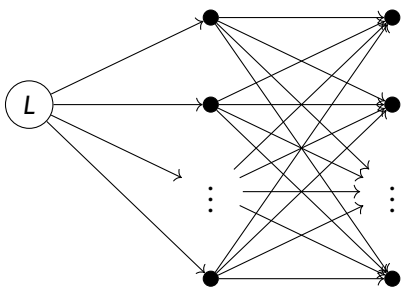
- For each server *i* with relative stake $\rho_i > \frac{1}{\Gamma}$, we fill $\lfloor \rho_i \Gamma \rfloor$ bins (each corresponding to a *relay* node), and calculate the remainder $\rho'_i = \rho_i - \frac{\lfloor \rho_i \Gamma \rfloor}{\Gamma}$. Otherwise $\rho'_i = \rho_i$.

Smart sampling



- For each server i with relative stake $\rho_i > \frac{1}{\Gamma}$, we fill $\lfloor \rho_i \Gamma \rfloor$ bins (each corresponding to a *relay* node), and calculate the remainder $\rho_i' = \rho_i - \frac{\lfloor \rho_i \Gamma \rfloor}{\Gamma}$. Otherwise $\rho_i' = \rho_i$.
- Calculate a partition of the relative stake $[\rho_1, \ldots, \rho_n]$ into the remaining $k = \Gamma - \Pi_{i=0}^{n} \lfloor \rho_i \Gamma \rfloor$ bins.

MonadBFT – RaptorCast



- It uses rateless codes.
- Potentially infinite.
- The leader produces $N_p = k \times r + n$ fragments.
- r is the redundancy factor (packet loss, a Byzantine assumption on the stake, necessary extra overhead).
- Verification mainly consists of authenticating the sender.

Insights

- With some differences, the consensus protocols considered so far provide a mechanism to skip a proposal if decoding fails, which captures the presence of a malicious leader.

Insights

- With some differences, the consensus protocols considered so far provide a mechanism to skip a proposal if decoding fails, which captures the presence of a malicious leader.

- Rateless codes applied to a Byzantine setting do not require less redundancy, but improve latency as decoding is asymptotically linear.

Insights

- With some differences, the consensus protocols considered so far provide a mechanism to skip a proposal if decoding fails, which captures the presence of a malicious leader.
- Rateless codes applied to a Byzantine setting do not require less redundancy, but improve latency as decoding is asymptotically linear.
- As a verification strategy all this protocols compose erasure codes with signatures (from the sender) or computationally-efficient VC such as Merkle tree.

Applications to Storing Blockchain Data

University of Bern

The Blockchain Data-Availability Problem

- Blockchain platforms (most prominently Ethereum) do not have enough capacity
  $\Longrightarrow$ Separation between Layer-1 and Layer-2 networks
  - Layer-2 networks (Arbitrum, Base, Starknet …) execute transactions fast
  - Post only condensed state updates to Layer 1
  - Where can the transaction data from the Layer 2 be found?

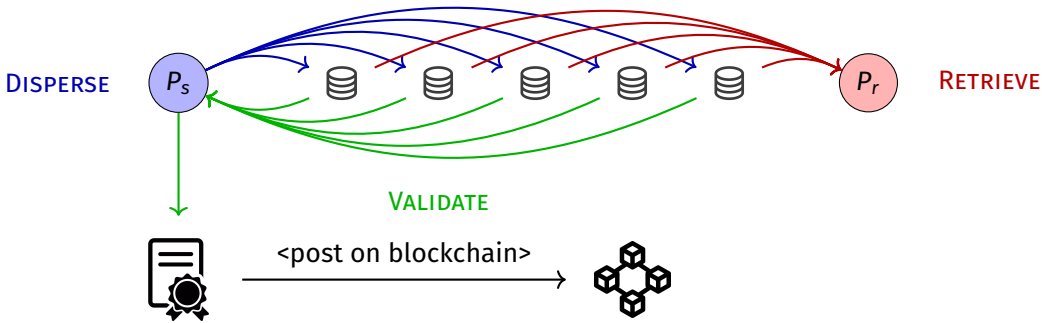
The Blockchain Data-Availability Problem

- Blockchain platforms (most prominently Ethereum) do not have enough capacity
  $\Longrightarrow$ Separation between Layer-1 and Layer-2 networks
  - Layer-2 networks (Arbitrum, Base, Starknet …) execute transactions fast
  - Post only condensed state updates to Layer 1
  - Where can the transaction data from the Layer 2 be found?
- Modern consensus algorithms decouple data *dissemination* from *ordering*
  $\Longrightarrow$ Block dissemination is removed from the critical path of consensus
  - Order is established over cryptographic references to blocks
  - How are the blocks with transaction stored?

The Blockchain Data-Availability Problem

- Blockchain platforms (most prominently Ethereum) do not have enough capacity
  $\Longrightarrow$ Separation between Layer-1 and Layer-2 networks
  - Layer-2 networks (Arbitrum, Base, Starknet …) execute transactions fast
  - Post only condensed state updates to Layer 1
  - Where can the transaction data from the Layer 2 be found?
- Modern consensus algorithms decouple data *dissemination* from *ordering*
  $\Longrightarrow$ Block dissemination is removed from the critical path of consensus
  - Order is established over cryptographic references to blocks
  - How are the blocks with transaction stored?
- Dedicated services for data availability resolve both issues

Vote-based Data Availability



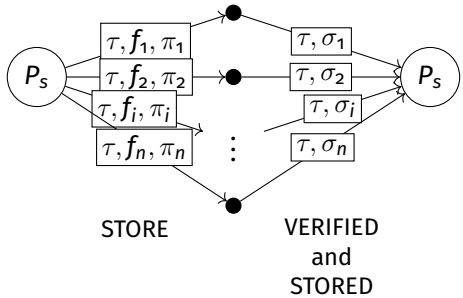
Certificate of Availability

Information Dispersal

with Provable Retrievability for Rollups

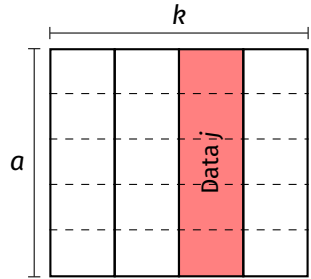
- Nazirkhanova et al. [NNT22] propose a weaker version of AVID thought for *data availability* use cases. They introduce *Semi-AVID-PR*, namely a weaker version of AVID with *provable retrievability*.
- Semi-AVID-PR does not have *session identifiers*, but leverages **binding commitments** to *link* the data between invocations of dispersal/retrieval.
- The **content-based** paradigm shift allows to unequivocally identify *what has been stored*.

Semi-AVID-PR – Protocol



How does Semi-AVID-PR guarantees *retrievability?*

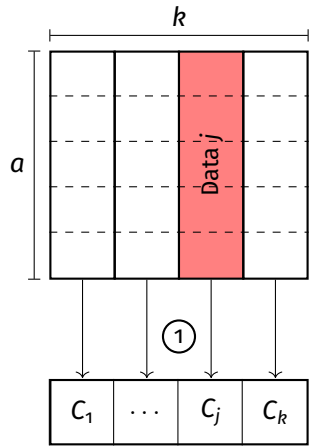
Semi-AVID-PR



Semi-AVID-PR – KZG Commitments

1 KZG.Commit($f_j(X)$) $\rightarrow C_j$

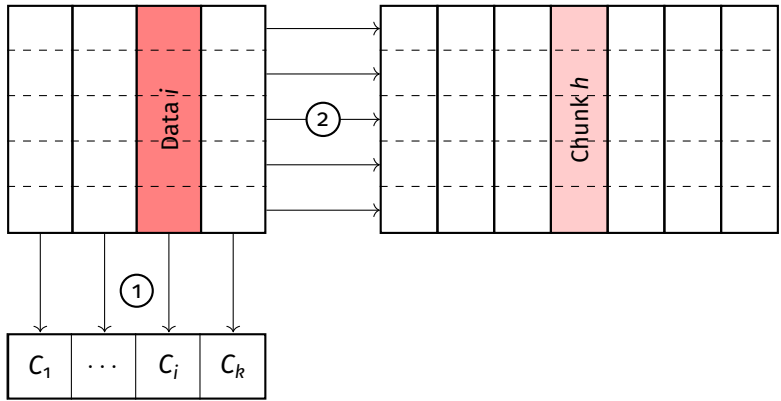
Semi-AVID-PR – KZG commitments



Semi-AVID-PR – Encoding

2 RS.Encode($f_{i_1,\ldots,f_k}$)

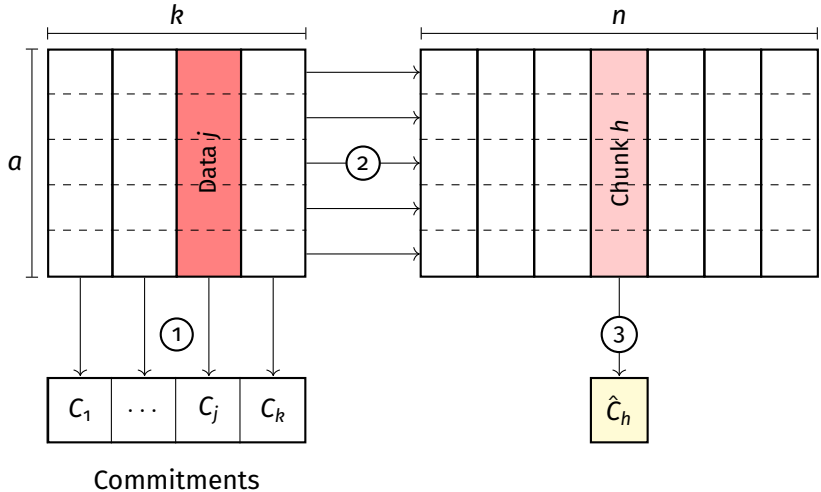
Semi-AVID-PR – Encoding



Semi-AVID-PR – Commitment on the chunk

(3) KZG.Commit($\hat{f}_h(X)$) $\rightarrow \hat{C}_h$

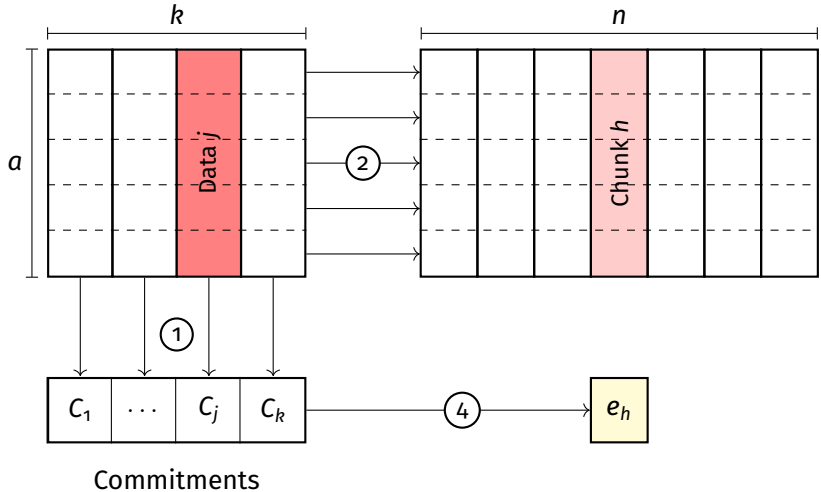
Semi-AVID-PR – Commitment on the chunk



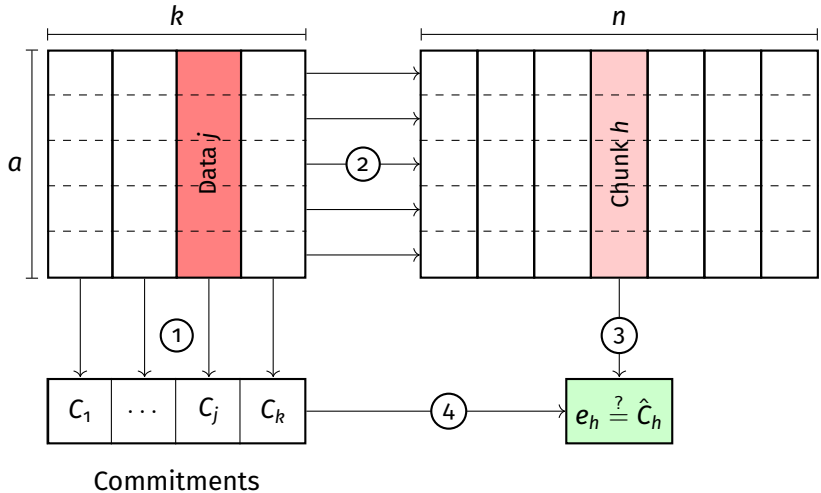
Semi-AVID-PR – Encode KZG vector

4 $[\text{RS.Encode}(C_1, \ldots, C_k)]_i \rightarrow e_h$

Semi-AVID-PR – Encode KZG vector



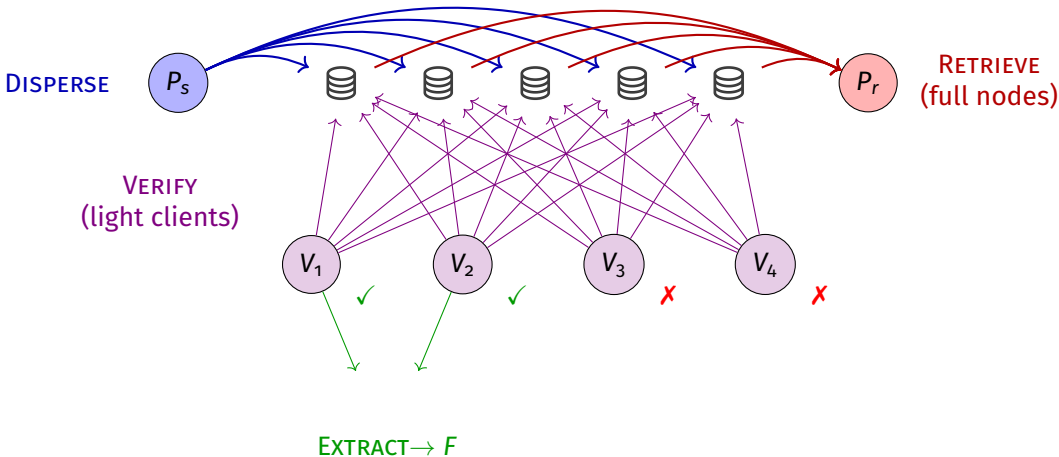
Semi-AVID-PR – Verification



Data Availability Sampling (DAS)

- It allows clients to verify availability of data provided by an *untrusted* source,
- only by sampling random positions of the encoded data.
- Over multiple verification, a client can recover the data.

Data Availability Sampling (DAS)



Data Availability Sampling – Definition

Components

- $\textsc{Disperse}(F) \to (D, [f_1, \ldots, f_n])$; produces a commitment to and an encoding of F
- $\textsc{Verify}(D) \to (b, tran)$; interactively verifies with storage nodes if fragment(s) from encoding match commitment, outputs success b and transcript $tran$ from interaction.
- $\textsc{Extract}(D, tran_1, \ldots, tran_\ell) \to F$ or $\perp$; reconstructs F from successfully verified transcripts.
- $\textsc{Retrieve}(D) \to F$ or $\perp$; interactively reads sufficiently many fragments from storage nodes to reconstruct F.

DAS – Properties

Completeness: If $\textsc{Encode}(F) \to (D, [f_1, \ldots, f_n])$, and $\textsc{Verify}(D)$ accesses $[f_1, \ldots, f_n]$ and outputs $\forall i \in [\ell]$, $b_i = 1$ and $tran_i$, then $\textsc{Extract}(D, [tran_1, \ldots, tran_\ell]) = F$.

DAS – Properties

Completeness: If $\textsc{Encode}(F) \to (D, [f_1, \ldots, f_n])$, and $\textsc{Verify}(D)$ accesses $[f_1, \ldots, f_n]$ and outputs $\forall i \in [\ell]$, $b_i = 1$ and $tran_i$, then $\textsc{Extract}(D, [tran_1, \ldots, tran_\ell]) = F$.

Soundness: Given transcript *tran* from successful verification, with enough storage nodes, $\textsc{Extract}(D, [tran_1, \ldots, tran_\ell])$ returns a file F consistent with commitment D.

DAS – Properties

Completeness: If $\textsc{Encode}(F) \to (D, [f_1, \ldots, f_n])$, and $\textsc{Verify}(D)$ accesses $[f_1, \ldots, f_n]$ and outputs $\forall i \in [\ell]$, $b_i = 1$ and $tran_i$, then $\textsc{Extract}(D, [tran_1, \ldots, tran_\ell]) = F$.

Soundness: Given transcript $tran$ from successful verification, with enough storage nodes, $\textsc{Extract}(D, [tran_1, \ldots, tran_\ell])$ returns a file F consistent with commitment D.

Consistency: Given any two sets of transcripts $(tran_1, i)_i^{\ell_1}$ and $(tran_2, i)_i^{\ell_2}$ from successful verification, with enough storage nodes, $\textsc{Extract}(D, [tran_{1,1} \ldots, tran_{1,\ell_1}])$ returns a file F_1 consistent with F_2 output of $\textsc{Extract}(D, [tran_{2,1} \ldots, tran_{2,\ell_2}])$.

DAS – Properties

Completeness: If $\textsc{Encode}(F) \to (D, [f_1, \ldots, f_n])$, and $\textsc{Verify}(D)$ accesses $[f_1, \ldots, f_n]$ and outputs $\forall i \in [\ell]$, $b_i = 1$ and $tran_i$, then $\textsc{Extract}(D, [tran_1, \ldots, tran_\ell]) = F$.

Soundness: Given transcript *tran* from successful verification, with enough storage nodes, $\textsc{Extract}(D, [tran_1, \ldots, tran_\ell])$ returns a file F consistent with commitment D.

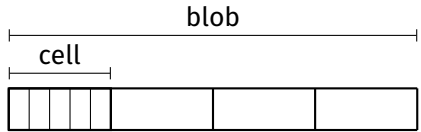
Consistency: Given any two sets of transcripts $(tran_1, i)_i^{\ell_1}$ and $(tran_2, i)_i^{\ell_2}$ from successful verification, with enough storage nodes, $\textsc{Extract}(D, [tran_{1,1} \ldots, tran_{1,\ell_1}])$ returns a file F_1 consistent with F_2 output of $\textsc{Extract}(D, [tran_{2,1} \ldots, tran_{2,\ell_2}])$.

Remarks

- Multiple verifiers act independently
- No correct majority among storage nodes needed (for sampling)

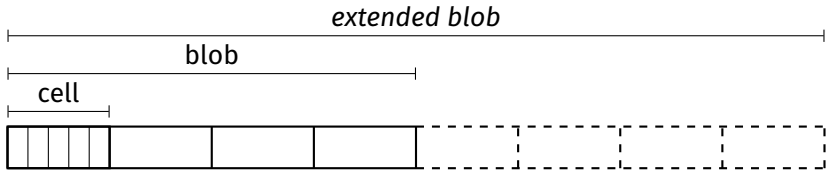
PeerDAS – Ethereum's DAS Protocol

- Each blob consists of 4096 field elements: 64 cells of 64 field elements each.



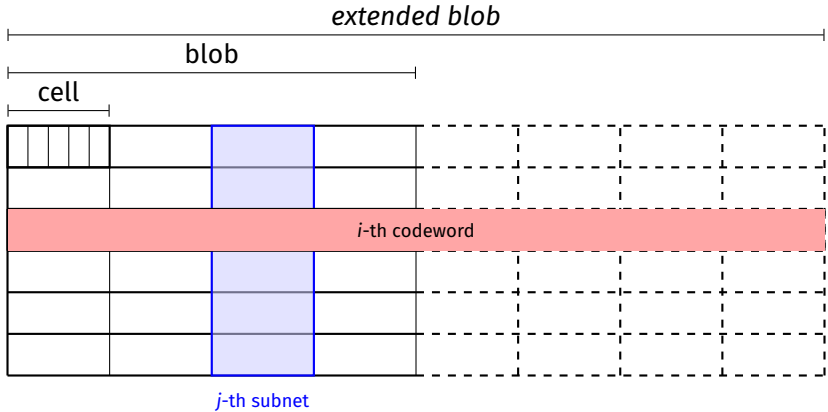
PeerDAS

- Each blob is *extended* to double its size using an MDS code, e.g., Reed-Solomon.
- A KZG commitment is used to commit to the polynomial resulting from the encoding procedure of degree $< k$.

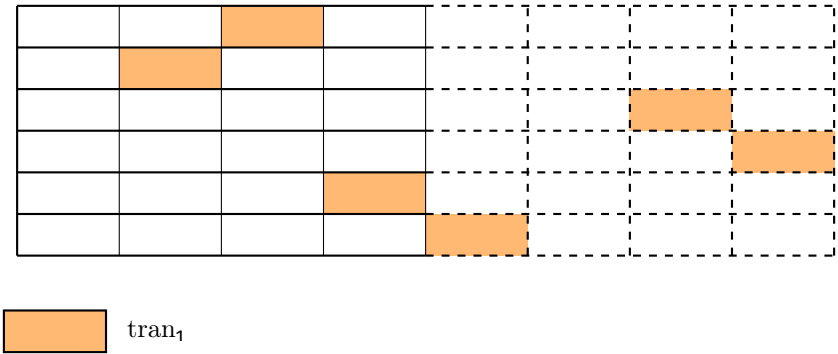


PeerDAS – Full block

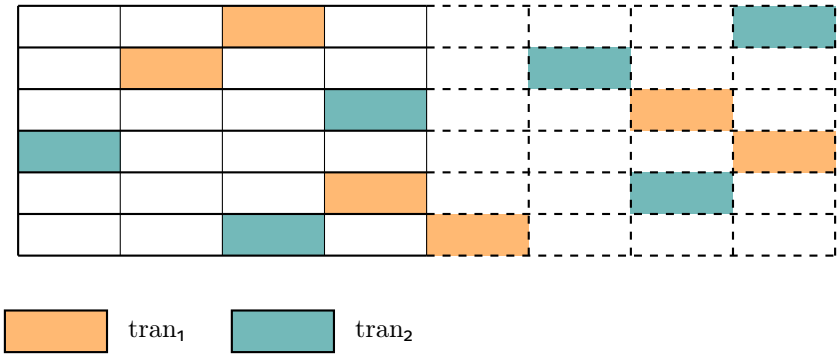
- Each subnet gets a column.



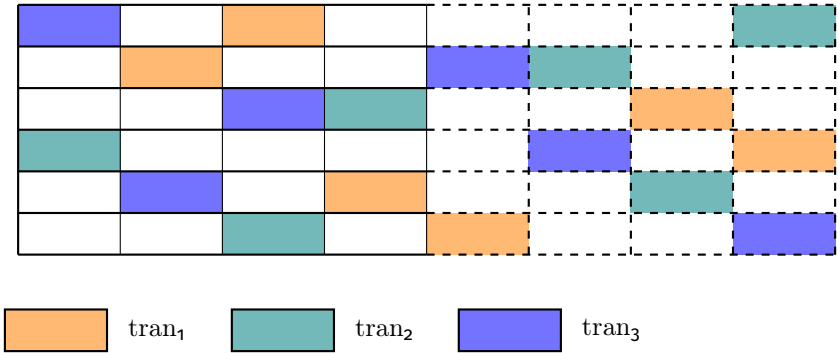
PeerDAS – Sampling example



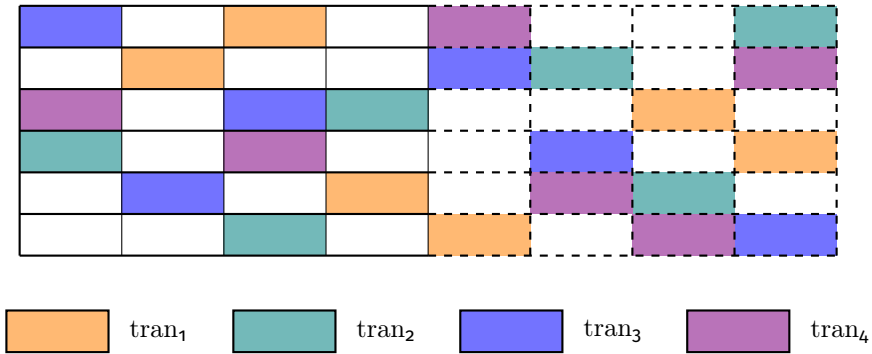
PeerDAS – Sampling example



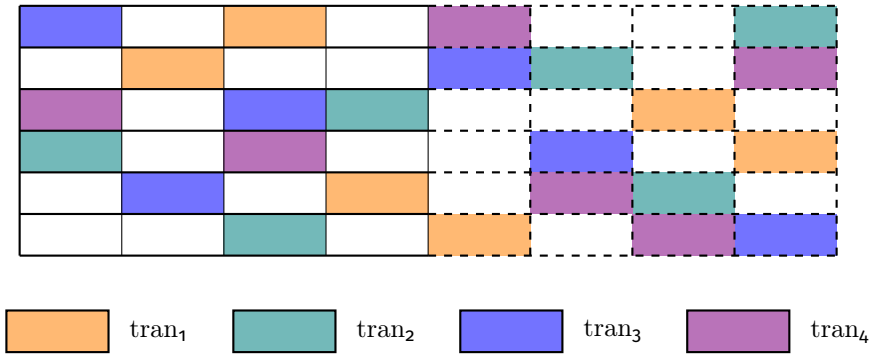
PeerDAS – Sampling example



PeerDAS – Sampling example



PeerDAS – Sampling example



Once a client gets *half* of the evaluation points on each row, it can reconstruct the corresponding data.

Thank you for your attention!

University of Bern
Institute of Computer Science
Cryptology and Data Security Group

July 10, 2026

References I

[ACLY00] Rudolf Ahlswede et al. “Network information flow”. In: *IEEE Trans. Inf. Theory* 46.4 (2000), pp. 1204–1216. DOI: 10.1109/18.850663.

[ACTZ24] Orestis Alpos et al. “Asymmetric distributed trust”. In: *Distributed Comput.* 37.3 (2024), pp. 247–277. DOI: 10.1007/S00446-024-00469-1.

[ACVZ25] Ignacio Amores-Sesar et al. “DAG-based Consensus with Asymmetric Trust”. In: *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2025, Hotel Las Brisas Huatulco, Huatulco, Mexico, June 16-20, 2025*. Ed. by Alkida Balliu and Fabian Kuhn. ACM, 2025, pp. 151–161. DOI: 10.1145/3732772.3733527.

[ADVZ21] Nicolas Alhaddad et al. “Succinct Erasure Coding Proof Systems”. In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 1500. URL: https://eprint.iacr.org/2021/1500.

References II

[Alh+22a] Nicolas Alhaddad et al. “Balanced Byzantine Reliable Broadcast with Near-Optimal Communication and Improved Computation”. In: *PODC*. ACM, 2022, pp. 399–417.

[Alh+22b] Nicolas Alhaddad et al. “Brief Announcement: Asynchronous Verifiable Information Dispersal with Near-Optimal Communication”. In: *PODC ’22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*. Ed. by Alessia Milani and Philipp Woelfel. ACM, 2022, pp. 418–420. DOI: 10.1145/3519270.3538476.

[BC24] Foteini Baldimtsi and Christian Cachin, eds. *Financial Cryptography and Data Security - 27th International Conference, FC 2023, Bol, Brač, Croatia, May 1-5, 2023, Revised Selected Papers, Part I*. Vol. 13950. Lecture Notes in Computer Science. Springer, 2024. ISBN: 978-3-031-47753-9. DOI: 10.1007/978-3-031-47754-6.

References III

[BNNP25] Dan Boneh et al. “Data Availability Sampling with Repair”. In: *IACR Cryptol. ePrint Arch.* (2025), p. 1414. URL: https://eprint.iacr.org/2025/1414.

[Bün+18] Benedikt Bünz et al. “Bulletproofs: Short Proofs for Confidential Transactions and More”. In: *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2018, pp. 315–334.

[Cel26] Celestia Labs. *Celestia – Fibre Optic Performance for Millisecond Markets*. https://celestia.org. 2026.

[CF13] Dario Catalano and Dario Fiore. “Vector Commitments and Their Applications”. In: *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*. Ed. by Kaoru Kurosawa and Goichiro Hanaoka. Vol. 7778. Lecture Notes in Computer Science. Springer, 2013, pp. 55–72. DOI: 10.1007/978-3-642-36362-7_5.

References IV

[CGR11] Christian Cachin et al. *Introduction to Reliable and Secure Distributed Programming (2. ed.)* Springer, 2011. ISBN: 978-3-642-15259-7. DOI: 10.1007/978-3-642-15260-3.

[Civ+24] Pierre Civit et al. "Efficient Signature-Free Validated Agreement". In: *38th International Symposium on Distributed Computing, DISC 2024, Madrid, Spain, October 28 - November 1, 2024*. Ed. by Dan Alistarh. Vol. 319. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 14:1–14:23. DOI: 10.4230/LIPICS.DISC.2024.14.

[CO18] Christian Cachin and Olga Ohrimenko. "Verifying the consistency of remote untrusted services with conflict-free operations". In: *Inf. Comput.* 260 (2018), pp. 72–88. DOI: 10.1016/J.IC.2018.03.004.

References V

[CP23] Benjamin Y. Chan and Rafael Pass. “Simplex Consensus: A Simple and Fast Consensus Protocol”. In: *Theory of Cryptography - 21st International Conference, TCC 2023, Taipei, Taiwan, November 29 - December 2, 2023, Proceedings, Part IV*. Ed. by Guy N. Rothblum and Hoeteck Wee. Vol. 14372. Lecture Notes in Computer Science. Springer, 2023, pp. 452–479. DOI: 10.1007/978-3-031-48624-1_17.

[CT05] Christian Cachin and Stefano Tessaro. “Asynchronous Verifiable Information Dispersal”. In: *24th IEEE Symposium on Reliable Distributed Systems (SRDS 2005),26-28 October 2005, Orlando, FL, USA*. IEEE Computer Society, 2005, pp. 191–202. DOI: 10.1109/RELDIS.2005.9.

[Dan+25] George Danezis et al. “Walrus: An Efficient Decentralized Storage Network”. In: *CoRR* abs/2505.05370 (2025). arXiv: 2505.05370. DOI: 10.48550/ARXIV.2505.05370.

References VI

[DGWWR10] Alexandros G. Dimakis et al. “Network coding for distributed storage systems”. In: *IEEE Trans. Inf. Theory* 56.9 (2010), pp. 4539–4551. DOI: 10.1109/TIT.2010.2054295.

[DRZ18] Sisi Duan et al. “BEAT: Asynchronous BFT Made Practical”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*. Ed. by David Lie et al. ACM, 2018, pp. 2028–2041. DOI: 10.1145/3243734.3243812.

[DXR21] Sourav Das et al. “Asynchronous Data Dissemination and its Applications”. In: *CCS*. ACM, 2021, pp. 2705–2721.

[Eig26] Eigen Labs. *EigenDA – State of the Art Data Availability*. https://www.eigenda.xyz. 2026.

[FBW06] Christina Fragouli et al. “Network coding: an instant primer”. In: *Comput. Commun. Rev.* 36.1 (2006), pp. 63–68. DOI: 10.1145/1111322.1111337.

References VII

[FLMT25] Hanwen Feng et al. “Faster Hash-based Multi-valued Validated Asynchronous Byzantine Agreement”. In: *55th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2025, Naples, Italy, June 23-26, 2025*. IEEE, 2025, pp. 303–316. DOI: 10.1109/DSN64029.2025.00040.

[Gao+22] Yingzi Gao et al. “Dumbo-NG: Fast Asynchronous BFT Consensus with Throughput-Oblivious Latency”. In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*. Ed. by Heng Yin et al. ACM, 2022, pp. 1187–1201. DOI: 10.1145/3548606.3559379.

[GHSY12] Parikshit Gopalan et al. “On the Locality of Codeword Symbols”. In: *IEEE Trans. Inf. Theory* 58.11 (2012), pp. 6925–6934. DOI: 10.1109/TIT.2012.2208937.

[Gor+25] Guy Goren et al. “Shelby: Decentralized Storage Designed to Serve”. In: *CoRR* abs/2506.19233 (2025). arXiv: 2506.19233. DOI: 10.48550/ARXIV.2506.19233.

References VIII

[GRKM25] Moritz Grundei et al. “From Indexing to Coding: A New Paradigm for Data Availability Sampling”. In: *CoRR* abs/2509.21586 (2025). arXiv: 2509.21586. DOI: 10.48550/ARXIV.2509.21586.

[HGR07] James Hendricks et al. “Verifying distributed erasure-coded data”. In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing, PODC 2007, Portland, Oregon, USA, August 12-15, 2007*. Ed. by Indranil Gupta and Roger Wattenhofer. ACM, 2007, pp. 139–146. DOI: 10.1145/1281100.1281122.

[Ho+06] Tracey Ho et al. “A Random Linear Network Coding Approach to Multicast”. In: *IEEE Trans. Inf. Theory* 52.10 (2006), pp. 4413–4430. DOI: 10.1109/TIT.2006.881746.

[HSW24a] Mathias Hall-Andersen et al. “Foundations of Data Availability Sampling”. In: *IACR Commun. Cryptol.* 1.4 (2024), p. 34. DOI: 10.62056/A09QUDHDJ.

References IX

[HSW24b] Mathias Hall-Andersen et al. “FRIDA: Data Availability Sampling from FRI”. In: *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part VI*. Ed. by Leonid Reyzin and Douglas Stebila. Vol. 14925. Lecture Notes in Computer Science. Springer, 2024, pp. 289–324. DOI: 10.1007/978-3-031-68391-6_9.

[JB25] Mohammad Mussadiq Jalalzai and Kushal Babel. “MonadBFT: Fast, Responsive, Fork-Resistant Streamlined Consensus”. In: *CoRR* abs/2502.20692 (2025). arXiv: 2502.20692. DOI: 10.48550/ARXIV.2502.20692.

References X

[KZG10] Aniket Kate et al. “Constant-Size Commitments to Polynomials and Their Applications”. In: *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*. Ed. by Masayuki Abe. Vol. 6477. Lecture Notes in Computer Science. Springer, 2010, pp. 177–194. DOI: 10.1007/978-3-642-17373-8_11.

[Lab18] I. Storj Labs. *Storj: A decentralized cloud storage network framework*. [Online]. 2018.

[LLTW20] Yuan Lu et al. “Dumbo-MVBA: Optimal Multi-Valued Validated Asynchronous Byzantine Agreement, Revisited”. In: *PODC ’20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*. Ed. by Yuval Emek and Christian Cachin. ACM, 2020, pp. 129–138. DOI: 10.1145/3382734.3405707.

References XI

[Loc24] Thomas Locher. “Byzantine Reliable Broadcast with Low Communication and Time Complexity”. In: *OPODIS*. Vol. 324. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 16:1–16:17.

[LS25] Thomas Locher and Victor Shoup. “MiniCast: Minimizing the Communication Complexity of Reliable Broadcast”. In: *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part V*. Ed. by Serge Fehr and Pierre-Alain Fouque. Lecture Notes in Computer Science. Springer, 2025, pp. 96–115. DOI: 10.1007/978-3-031-91092-0_4.

[Lub02] Michael Luby. “LT Codes”. In: *43rd Symposium on Foundations of Computer Science, FOCS 2002, Vancouver, BC, Canada, November 16-19, 2002, Proceedings*. IEEE Computer Society, 2002, p. 271. DOI: 10.1109/SFCS.2002.1181950.

References XII

[Mac05] David JC MacKay. “Fountain codes”. In: *IEE Proceedings-Communications* 152.6 (2005), pp. 1062–1068. URL: https://docs.switzernet.com/people/emin-gabrielyan/060123-capillary-aron-article/ref/MacKay05.pdf.

[NNT22] Kamilla Nazirkhanova et al. “Information Dispersal with Provable Retrievability for Rollups”. In: *Proceedings of the 4th ACM Conference on Advances in Financial Technologies, AFT 2022, Cambridge, MA, USA, September 19-21, 2022*. Ed. by Maurice Herlihy and Neha Narula. ACM, 2022, pp. 180–197. DOI: 10.1145/3558535.3559778.

[NRSVX20] Kartik Nayak et al. “Improved Extension Protocols for Byzantine Broadcast and Agreement”. In: *DISC*. Vol. 179. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 28:1–28:17.

References XIII

[PD12] Dimitris S. Papailiopoulos and Alexandros G. Dimakis. “Locally Repairable Codes”. In: *CoRR* abs/1206.3804 (2012). URL: http://arxiv.org/abs/1206.3804. arXiv: 1206.3804.

[PD20] Yiannis Psaras and David Dias. “The InterPlanetary File System and the Filecoin Network”. In: *50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020 - Supplemental Volume*. IEEE, 2020, p. 80. DOI: 10.1109/DSN-S50200.2020.00043.

[PSX24] Zengyu Pang et al. “Asynchronous verifiable information dispersal protocol of achieving optimal communication”. In: *Comput. Networks* 247 (2024), p. 110470. DOI: 10.1016/J.COMNET.2024.110470.

[Q K25] R. Wattenhofer Q. Kniep J. Sliwinski. *Solana Alpenglow Consensus*. Alpenglow White Paper. 2025.

References XIV

[Rab89] Michael O. Rabin. “Efficient dispersal of information for security, load balancing, and fault tolerance”. In: *J. ACM* 36.2 (1989), pp. 335–348. DOI: 10.1145/62044.62050.

[RS60] I. S. Reed and G. Solomon. “Polynomial Codes Over Certain Finite Fields”. In: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304. eprint: https://doi.org/10.1137/0108018. DOI: 10.1137/0108018.

[RSK10] K. V. Rashmi et al. “Optimal Exact-Regenerating Codes for Distributed Storage at the MSR and MBR Points via a Product-Matrix Construction”. In: *CoRR* abs/1005.4178 (2010). URL: http://arxiv.org/abs/1005.4178. arXiv: 1005.4178.

References XV

[She+25] Zhirong Shen et al. “A Survey of the Past, Present, and Future of Erasure Coding for Storage Systems”. In: *ACM Trans. Storage* 21.1 (2025), 4:1–4:39. DOI: 10.1145/3708994.

[Sho06] Amin Shokrollahi. “Raptor codes”. In: *IEEE Trans. Inf. Theory* 52.6 (2006), pp. 2551–2567. DOI: 10.1109/TIT.2006.874390.

[Sho24] Victor Shoup. “Sing a Song of Simplex”. In: *38th International Symposium on Distributed Computing, DISC 2024, Madrid, Spain, October 28 - November 1, 2024*. Ed. by Dan Alistarh. Vol. 319. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 37:1–37:22. DOI: 10.4230/LIPICS.DISC.2024.37.

[TB14] Itzhak Tamo and Alexander Barg. “A Family of Optimal Locally Recoverable Codes”. In: *IEEE Trans. Inf. Theory* 60.8 (2014), pp. 4661–4676. DOI: 10.1109/TIT.2014.2321280.

References XVI

[Vaj+18] Myna Vajha et al. “Clay Codes: Moulding MDS Codes to Yield an MSR Code”. In: *16th USENIX Conference on File and Storage Technologies, FAST 2018, Oakland, CA, USA, February 12-15, 2018*. Ed. by Nitin Agrawal and Raju Rangaswami. USENIX Association, 2018, pp. 139–154. URL: https://www.usenix.org/conference/fast18/presentation/vajha.

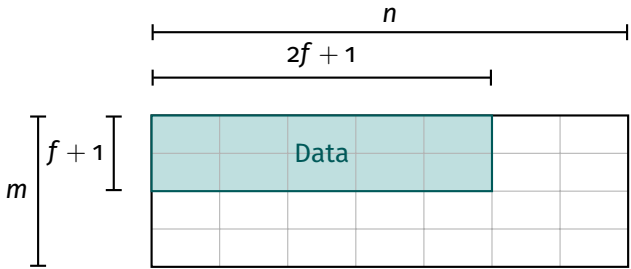
[WDBU19] S. Williams et al. *Arweave: A protocol for economically sustainable information permanence*. Arweave Yellow Paper. 2019.

[Yan+24] Changlin Yang et al. “Scaling Blockchains with Error Correction Codes: A Survey on Coded Blockchains”. In: *ACM Comput. Surv.* 56.6 (2024), 139:1–139:33. DOI: 10.1145/3637224.

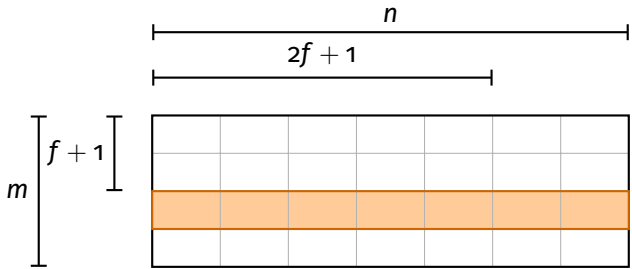
References XVII

[YPAKT22] Lei Yang et al. “DispersedLedger: High-Throughput Byzantine Consensus on Variable Bandwidth Networks”. In: *19th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2022, Renton, WA, USA, April 4-6, 2022*. Ed. by Amar Phanishayee and Vyas Sekar. USENIX Association, 2022, pp. 493–512. URL: https://www.usenix.org/conference/nsdi22/presentation/yang.

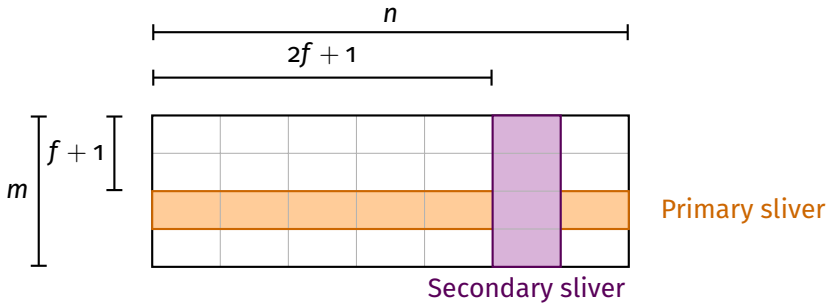
Decentralized storage



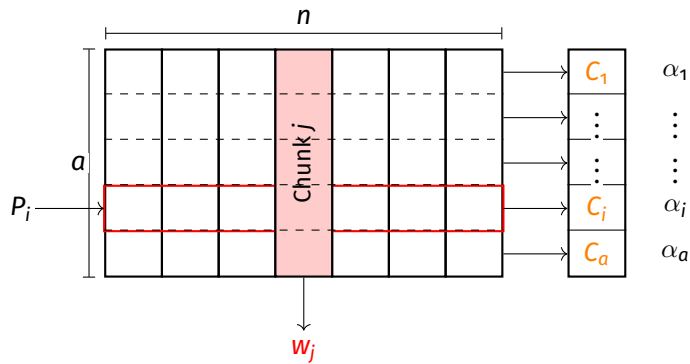
Decentralized storage



Decentralized storage

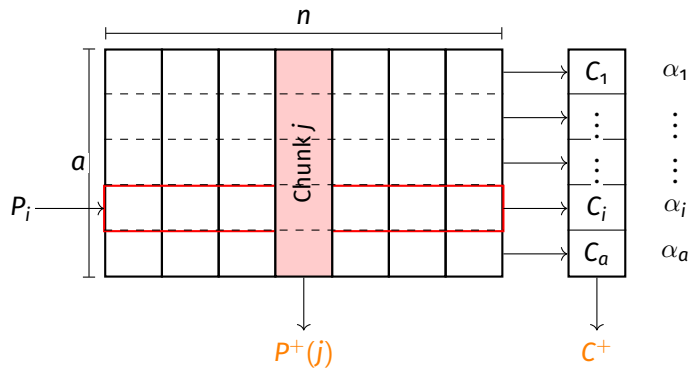


AVID-1 – Protocol



w_j opening of $(j, P^+(j))$ $\quad P^+(X) = \sum_{i=1}^{a} \alpha_i \cdot P_i(X)$ $\quad \alpha_i = \ell_i(c),$ $\quad c = H(C_1 || \ldots || C_a)$

AVID-1 – Protocol



$$P^+(j) = \sum_{i=1}^{a} \alpha_i \cdot P_i(j) \qquad C^+ = \Pi_{i=1}^{a} C_i^{\alpha_i} \qquad \alpha_i = \ell_i(c), \qquad c = H(C_1 || \ldots || C_a)$$