

Verifying Distributed Erasure-Coded Data

James Hendricks, Gregory R. Ganger
Carnegie Mellon University

Michael K. Reiter
University of North Carolina at Chapel Hill

Motivation

- Storage systems must be reliable
 - Growing in size and importance
- Must tolerate more than just crashes
- Ideally would tolerate *Byzantine* faults
 - Both Byzantine faulty servers and clients

Recent Byzantine fault-tolerant storage systems

This is an important problem with a lot recent progress and interest:

LOFT	Hendricks et. al [SOSP 2007]
BFT-BC	Liskov & Rodrigues [ICDCS 2006]
AVID	Cachin & Tessaro [SRDS 2005, DSN 2006]
Ursa Minor	Abd-El-Malek et. al [FAST 2005]
PASIS	Goodson et. al [DSN 2004, SRDS 2005]
Oceanstore	Rhea et. al [FAST 2003]
Farsite	Adya et. al [OSDI 2002]
SBQ-L	Martin et. al [DISC 2002]

... and more ...

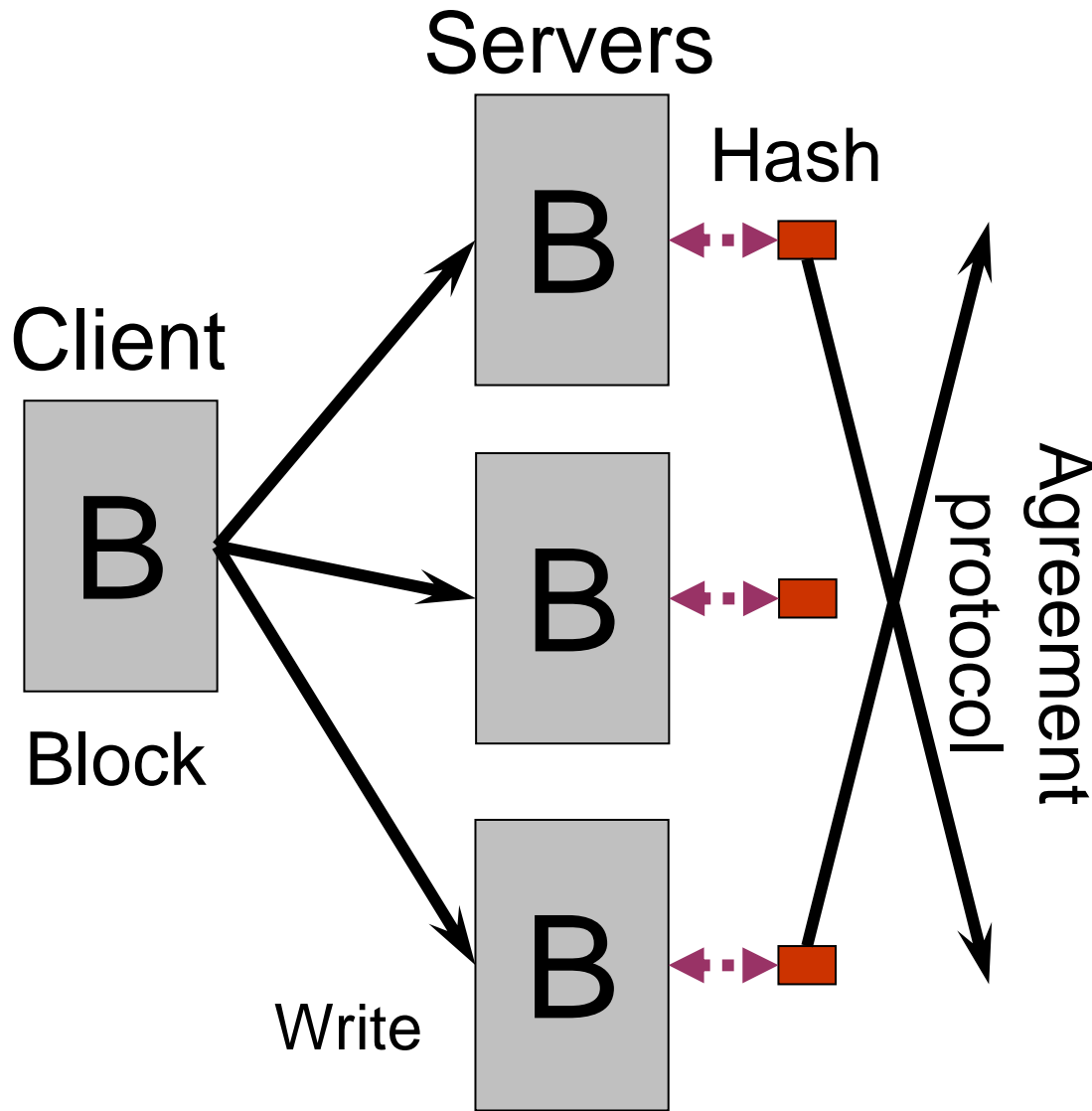
Outline

➔ Byzantine fault-tolerant storage:
Replication versus erasure-coding

Homomorphic fingerprinting

An example usage

Typical replication-based write protocol



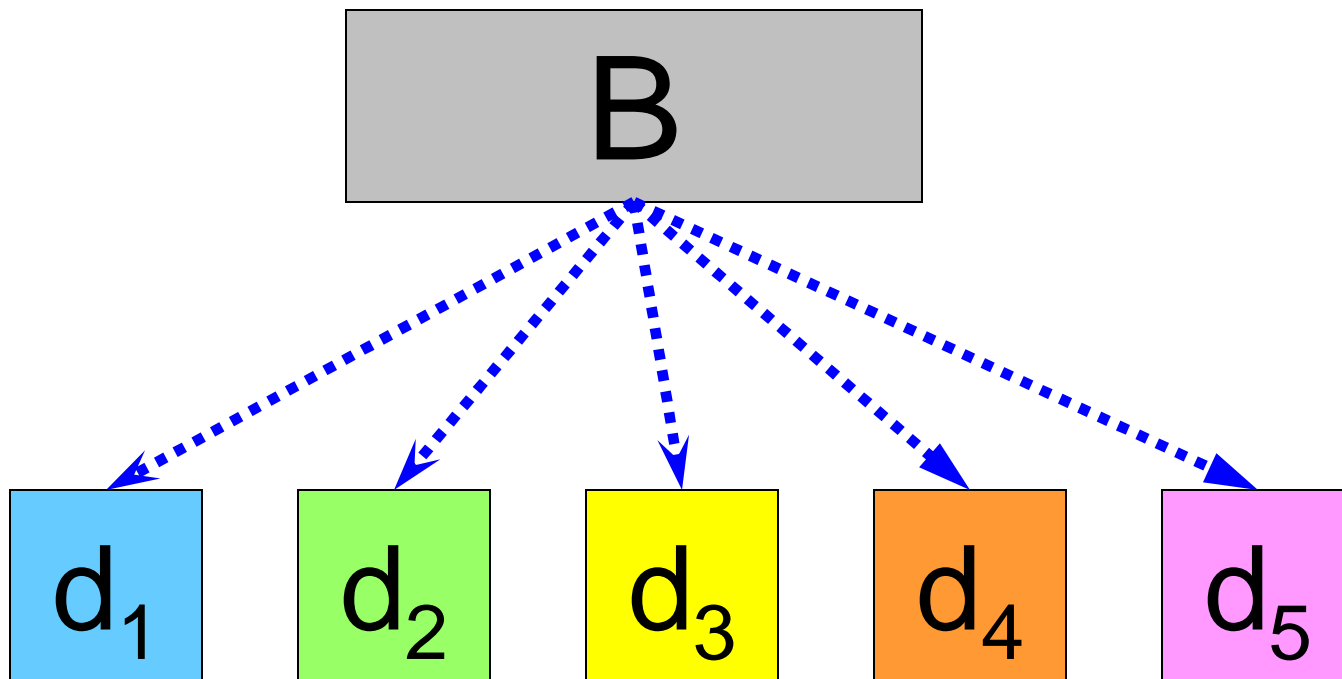
- (1) Client sends each server entire block
 - (2) Server hashes block
 - (3) Server runs agreement protocol on hash
- ▶ Bandwidth: $O(n \cdot |B|)$

Replication is wasteful

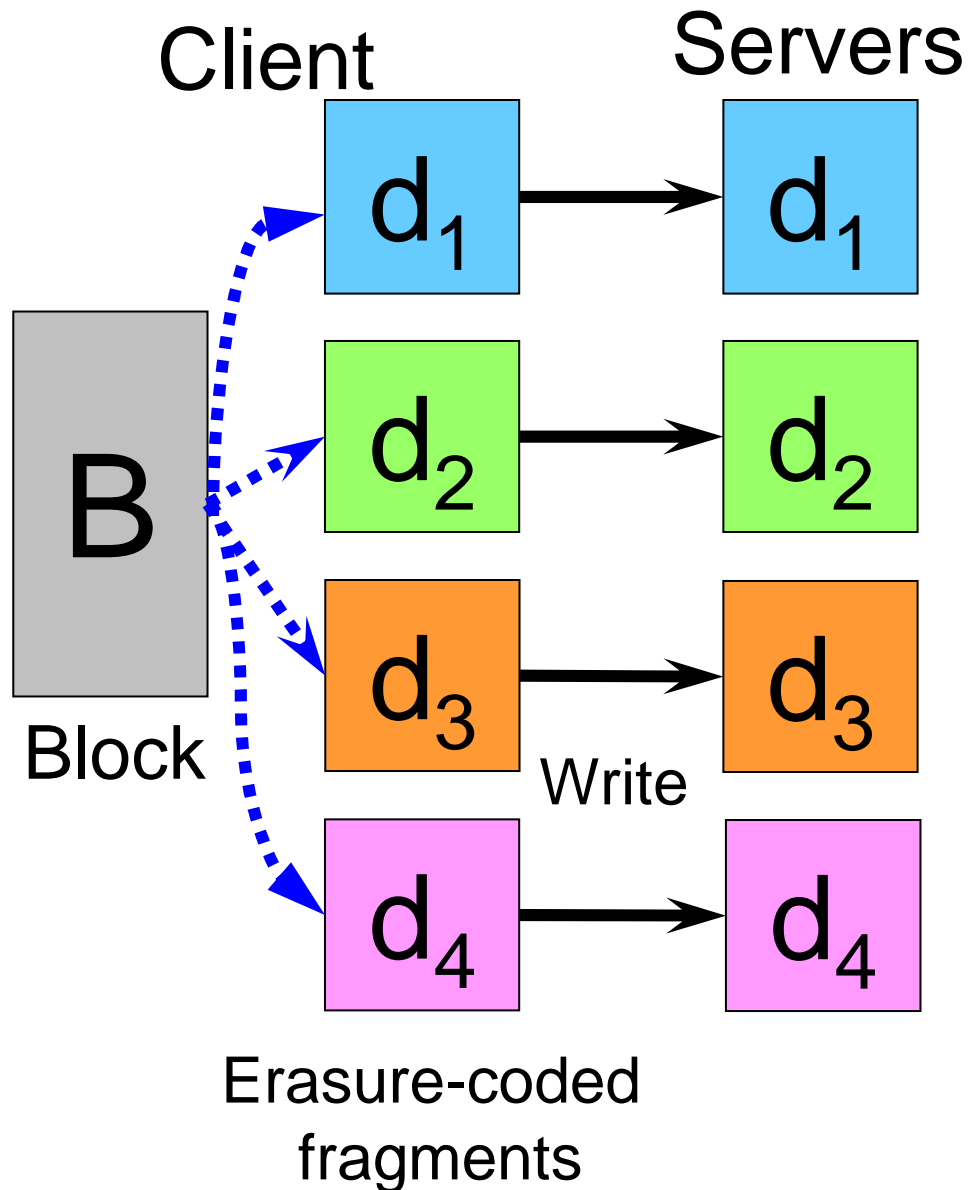
- Problem: Replication has high overhead
 - Writing block B requires $O(n \cdot |B|)$ network bandwidth, disk I/O bandwidth, and disk capacity
- Solution: *Erasure code* block
- **Definition:** An *m-of-n erasure code* divides block B into n fragments, each size $|B|/m$, such that any m fragments can be used to reconstruct block B
 - Examples: Reed-Solomon, Rabin's IDA, parity

Example erasure coding

Example: A 3-of-5 erasure code divides block B into 5 fragments, each size $|B|/3$, such that any 3 fragments can be used to reconstruct block B

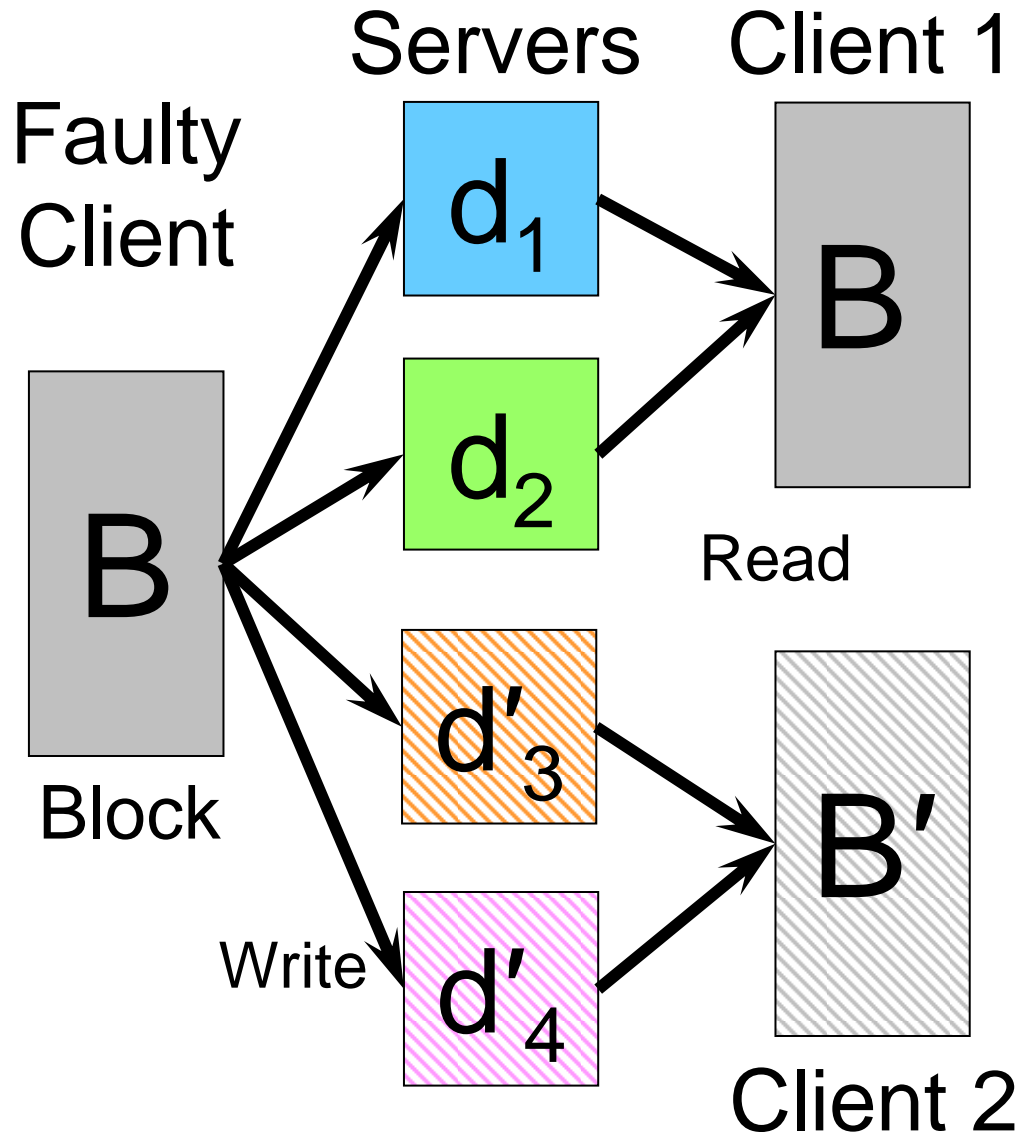


Writing erasure-coded data



- (1) Client erasure codes block
- (2) Client sends each fragment to a server
 - ▶ Good news—
Bandwidth: $O(|B|)$
 - ▶ Bad—Now what?
Can't hash data
because each server
has a different fragment

What could go wrong?



(1) Faulty client writes inconsistently encoded block

(2) Client 1 reads block B

(3) Client 2 reads block $B' \neq B$

E.g., bank auditors read "\$25", ATM reads "\$25 million"

Summary so far

Byzantine fault-tolerant erasure-coded storage

- Important for write bandwidth

But it introduces a problem: how to verify that data was encoded correctly?

Our contribution: Homomorphic fingerprinting

- Allows servers to verify *distributed* erasure-coded data
- Little extra bandwidth or computation

Outline

Byzantine fault-tolerant storage:

Replication versus erasure-coding

→ Homomorphic fingerprinting

An example usage

Definition: Fingerprinting

Definition: A fingerprinting function $fp(r,d)$:

- Adversary provides two fragments $d \neq d'$
- Choose random value r
- Probability that $fp(r,d) = fp(r,d')$ is bounded and small

As in universal hashing [Carter&Wegman77]
and Rabin's fingerprint [Rabin81]

Example: Evaluation fingerprint (1)

(1) Represent fragments as coefficients of a polynomial

$$d = \begin{array}{|c|c|c|c|c|} \hline a_4 & a_3 & a_2 & a_1 & a_0 \\ \hline \end{array}$$

$$d(x) = a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x^1 + a_0 \cdot x^0$$

(2) Fingerprint: Evaluate polynomial at random value r

$$\text{fp}(r,d) = d(r) = a_4 \cdot r^4 + a_3 \cdot r^3 + a_2 \cdot r^2 + a_1 \cdot r^1 + a_0 \cdot r^0$$

Example: Evaluation fingerprint (2)

- (1) Adversary provides two fragments $d \neq d'$
- (2) Represent fragments as coefficients of a polynomial

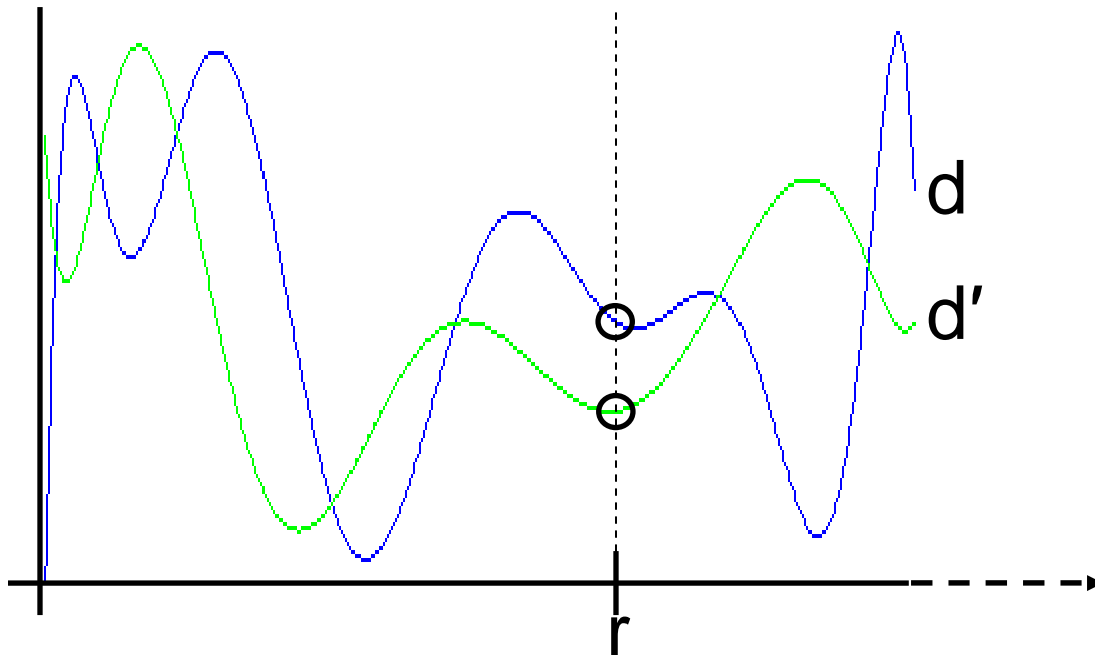
$$d = \begin{array}{|c|c|c|c|c|} \hline a_4 & a_3 & a_2 & a_1 & a_0 \\ \hline \end{array} \quad \begin{array}{|c|c|c|c|c|} \hline a'_4 & a'_3 & a'_2 & a'_1 & a'_0 \\ \hline \end{array} = d'$$

- (3) Choose random value r
- (4) Fingerprint: Evaluate polynomial at r

$$\text{fp}(r,d) = d(r) = a_4 \cdot r^4 + a_3 \cdot r^3 + a_2 \cdot r^2 + a_1 \cdot r^1 + a_0 \cdot r^0$$
$$\text{fp}(r,d') = d'(r) = a'_4 \cdot r^4 + a'_3 \cdot r^3 + a'_2 \cdot r^2 + a'_1 \cdot r^1 + a'_0 \cdot r^0$$

➔ Probability that $d(r) = d'(r)$ is bounded and small

Example: Evaluation fingerprint (3)



$$d = \boxed{a_{10}} \boxed{a_9} \dots \boxed{a_1} \boxed{a_0} \quad \boxed{a'_{10}} \boxed{a'_9} \dots \boxed{a'_1} \boxed{a'_0} = d'$$

$$d(r) = a_{10} \cdot r^{10} + a_9 \cdot r^9 + \dots + a_1 \cdot r^1 + a_0 \cdot r^0$$

$$d'(r) = a'_{10} \cdot r^{10} + a'_9 \cdot r^9 + \dots + a'_1 \cdot r^1 + a'_0 \cdot r^0$$

(3) Probability that $d(r) = d'(r)$ is bounded and small

Linear erasure codes

A *linear erasure* code has this structure:

$$d_j = \sum b_{ij} \cdot d_i = b_{j1} \cdot d_1 + b_{j2} \cdot d_2 + \dots + b_{jm} \cdot d_m$$

for constants b_{ij}

Many erasure codes are linear

e.g. Reed-Solomon, Rabin's IDA, parity

Definition: Homomorphic Fingerprinting

Goal:

Encoding of fingerprints = fingerprint of encoding

For example,

$$\text{If } d_j = \sum b_{ij} \cdot d_i$$

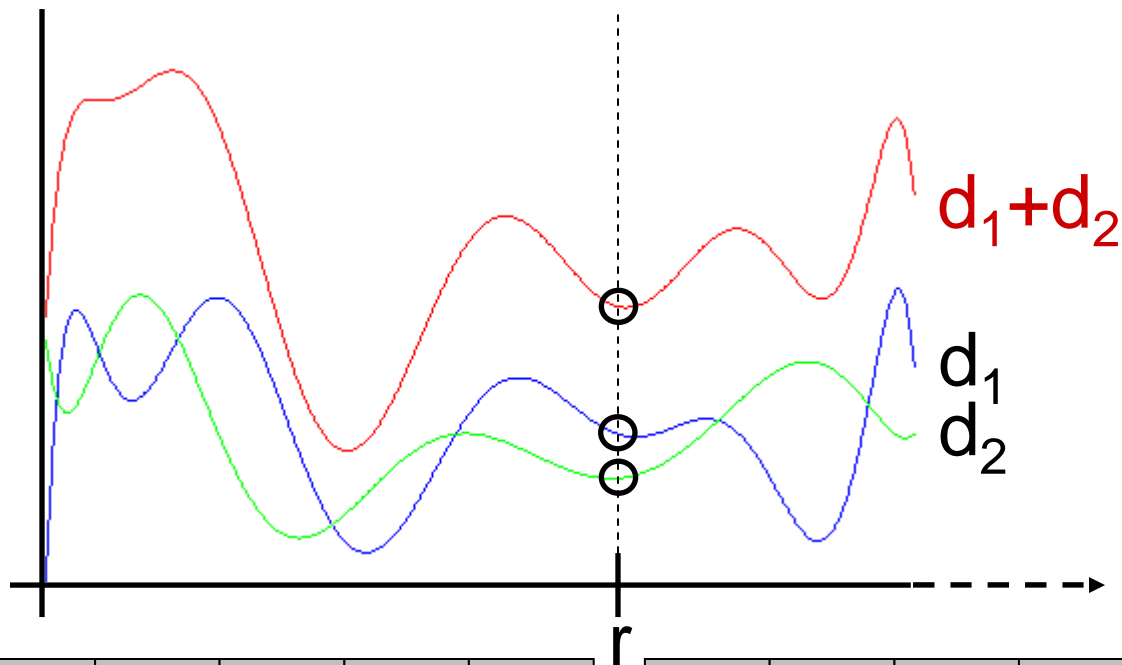
$$\text{Then } \text{fp}(r, d_j) = \text{fp}(r, \sum b_{ij} \cdot d_i) = \sum b_{ij} \cdot \text{fp}(r, d_i)$$

For linear erasure codes, true if

$$\text{fp}(r, d_1 + d_2) = \text{fp}(r, d_1) + \text{fp}(r, d_2) \quad \text{and}$$

$$\text{fp}(r, b \cdot d) = b \cdot \text{fp}(r, d)$$

Eval fp is homomorphic for “+”



$$d_1 = \boxed{a_{10}} \boxed{a_9} \dots \boxed{a_1} \boxed{a_0} \quad \boxed{c_{10}} \boxed{c_9} \dots \boxed{c_1} \boxed{c_0} = d_2$$

$$d_1(r) = a_{10} \cdot r^{10} + a_9 \cdot r^9 + \dots + a_1 \cdot r^1 + a_0 \cdot r^0$$

$$+ d_2(r) = c_{10} \cdot r^{10} + c_9 \cdot r^9 + \dots + c_1 \cdot r^1 + c_0 \cdot r^0$$

$$\underline{(d_1 + d_2)(r) = (a_{10} + c_{10}) \cdot r^{10} + \dots + (a_0 + c_0) \cdot r^0}$$

Details

$$d = \begin{array}{|c|c|c|c|c|} \hline a_4 & a_3 & a_2 & a_1 & a_0 \\ \hline \end{array}$$

Coefficient “a” must be represented carefully

Use extension field of the encoding field

See paper for details

Performance: 410 MB/s on 3 GHz Pentium D

How to choose random value r ?

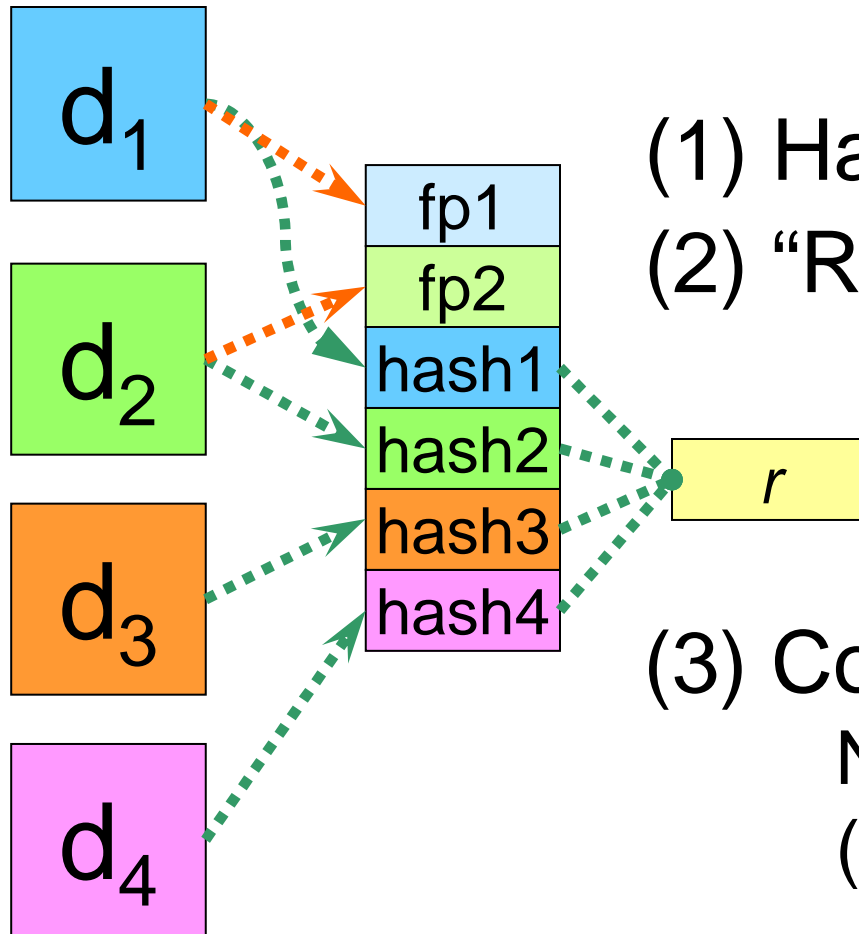
(1) Use a distributed pseudo-random function

[Naor, Pinkas, Reingold 99]

(2) Use a Random Oracle

[Bellare and Rogaway 93]

Random Oracle approach: the checksum



(1) Hash each fragment

(2) “Random” $r = \text{hash}(\text{hashes})$

(3) Compute m fingerprints

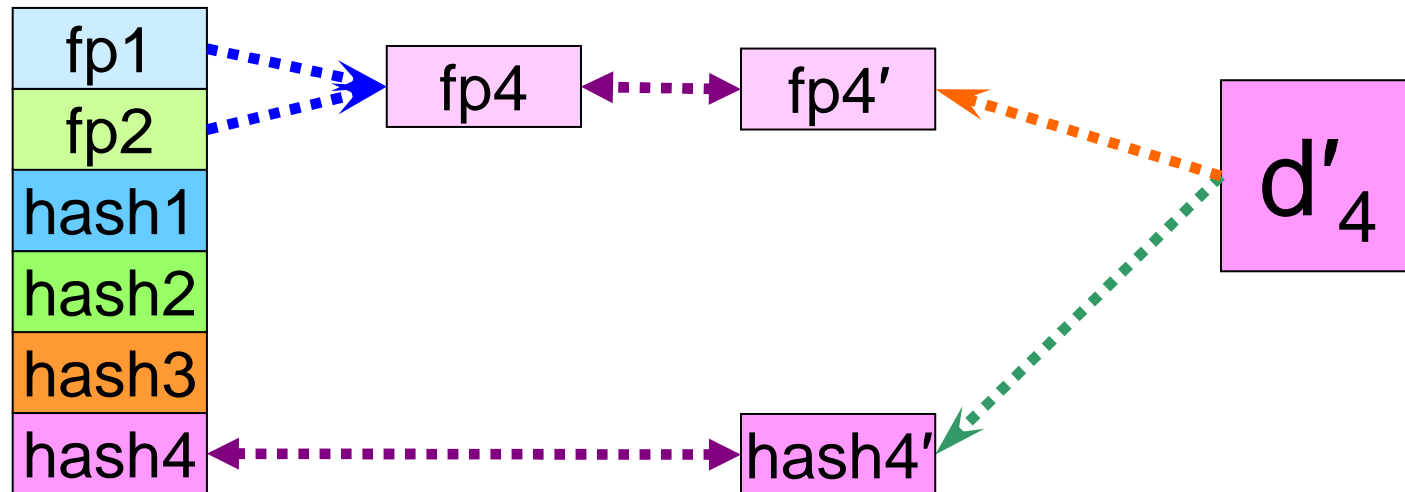
No need to compute all n

(Encoding of fingerprints

= fingerprint of encoding)

Fragment consistent w/ checksum

Fragment is “consistent” if hash and fingerprint match checksum



Key property: Block decoded from consistent fragments is unique

Outline

Byzantine fault-tolerant storage:
Replication versus erasure-coding

Homomorphic fingerprinting

→ An example usage

AVID: Asynchronous Verifiable Information Dispersal

Asynchronous Verifiable Information Dispersal

Cachin and Tessaro, SRDS 2005

Properties:

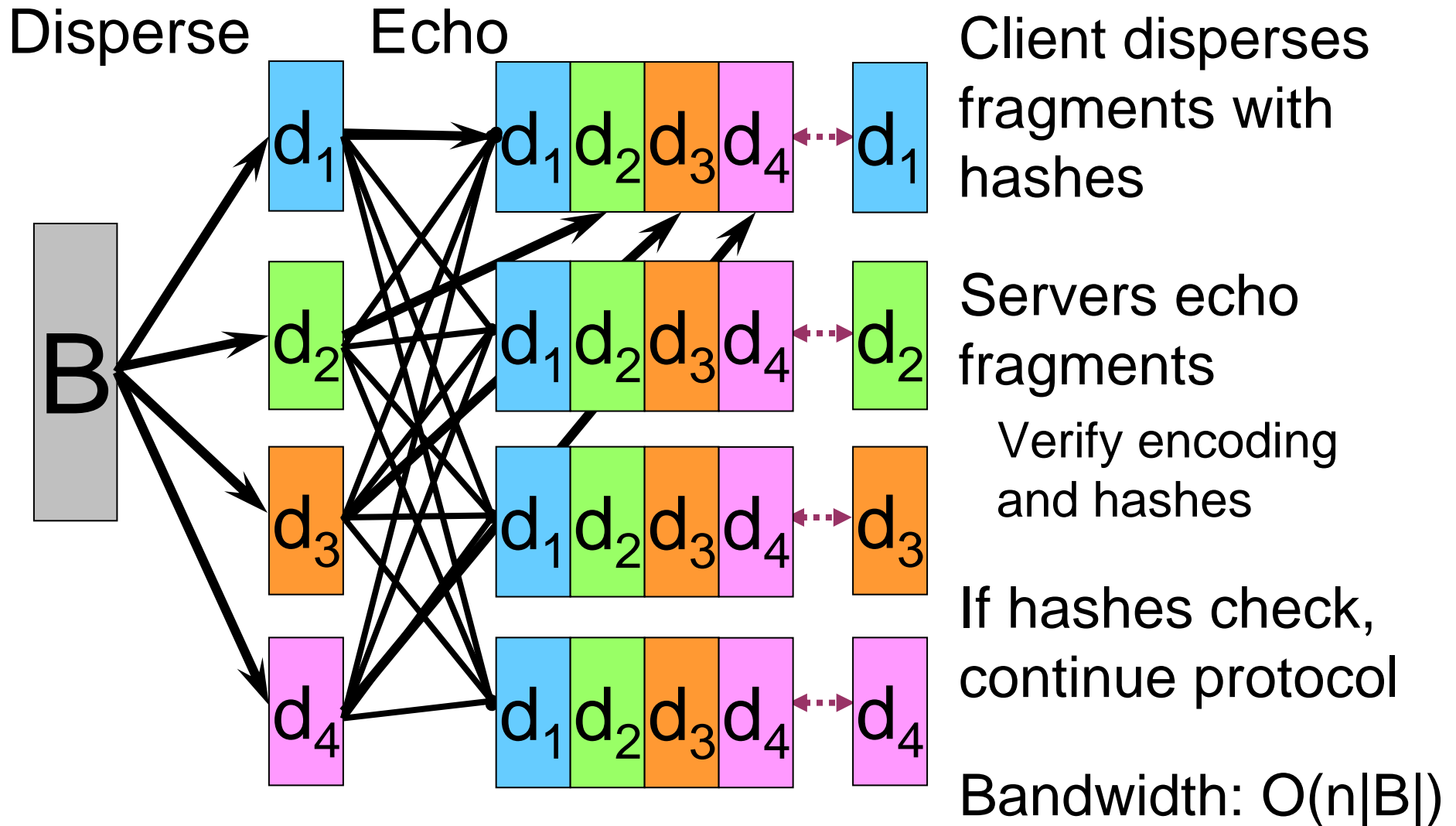
Correct clients always read the same block

If correctly written block, a correct reader reads it

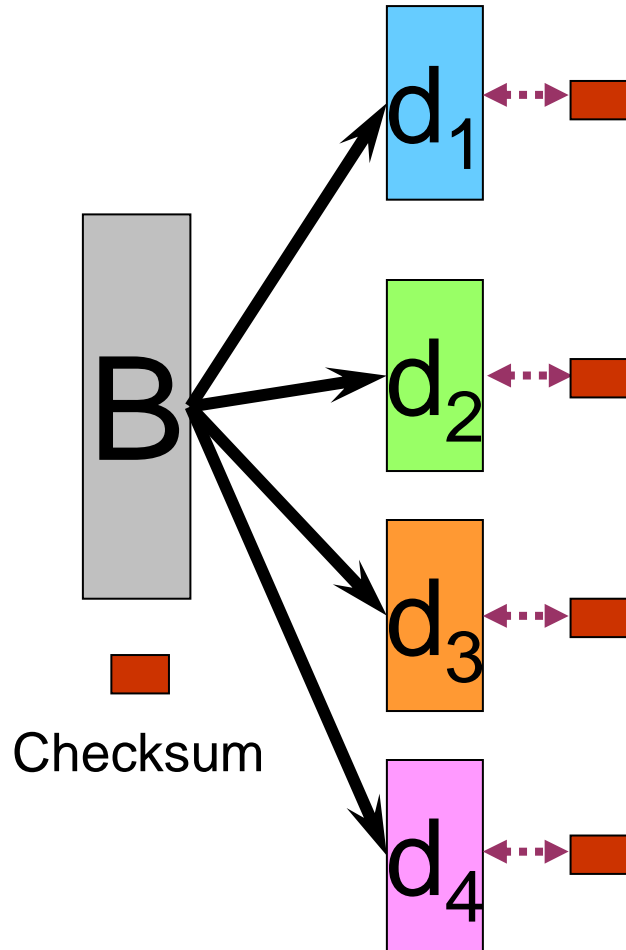
Can use to build a Byzantine storage system

Cachin and Tessaro, DSN 2006

Example: AVID



Example: AVID-FP



Add homomorphic fingerprinting to AVID

Send checksum rather than hashes

Each server verifies its fragment with checksum

If fragment consistent, continue protocol

Bandwidth: $O(|B|)$

Summary

Propose *homomorphic fingerprinting* to allow reasoning about distributed data

- Can use to verify that distributed erasure-coded data is encoded correctly

- Fingerprinting functions are fast and simple

Can lower overhead of Byzantine fault-tol. storage

Our SOSP 2007 paper builds on this technique

- “Low-overhead Byzantine fault-tolerant storage”