Space-Efficient Block Storage Integrity

Alina Oprea, Michael Reiter, and Ke Yang

NDSS '05

Presented by
Lucas Ballard and Josh Mason

Outline

- Description of the problem
- Related Work
- Background Material
- Proposed Schemes / Performance

The Problem

- Untrusted Network Area Storage/ Storage Area Network
- Want to secure your data
  - Confidentiality
  - Integrity
- Efficiency

Goal

- To efficiently provide confidentiality and integrity within the constraints of a SAN.
- This requires length-preserving operations

Security Model

- Confidentiality
- Integrity
  - The server returns a block that was never written to a specific location
  - The server returns an older version of a block

Efficiency

- Minimize Storage Overhead
  - block accesses
  - Client v. Server
- No Computationally-expensive algorithms

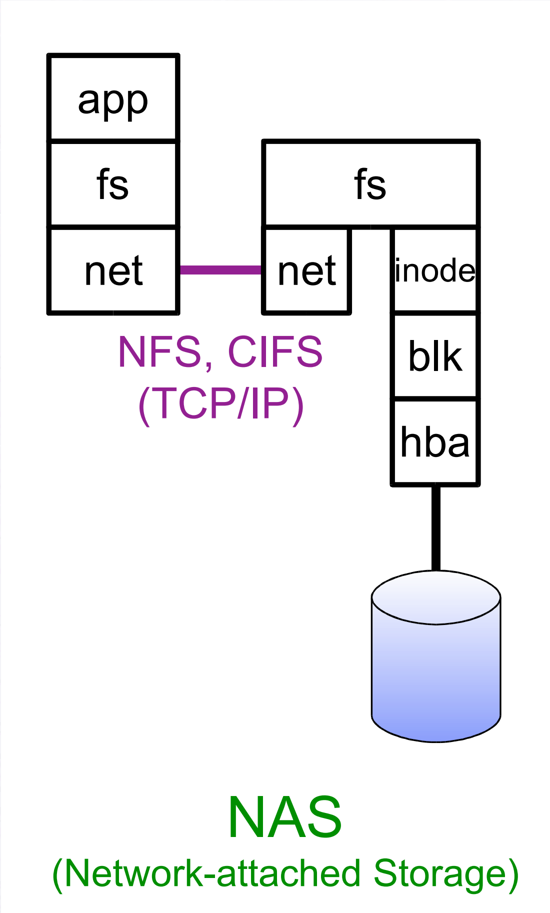
Related Work

Related Work

- NAS/SAN
- TCFS
- Sirius

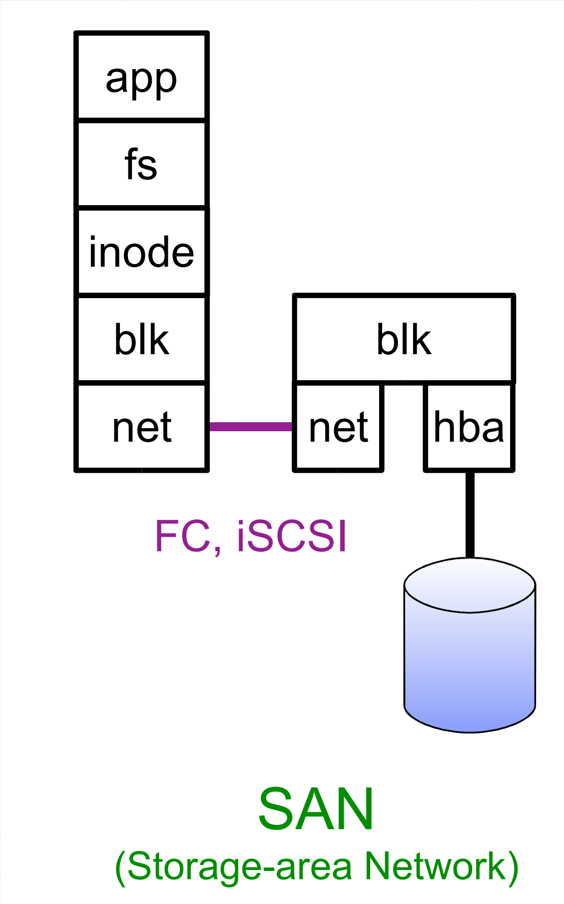
NAS

- Network Attached Storage
- Employs file I/O (fetch entire files, referenced by file names)
- Easy to implement/manage



SANs

- Storage Area Networks
- Employ block I/O (fetch a block at a time)
- Much faster, can be more bandwidth efficient
    - Efficiency determined by number of block accesses



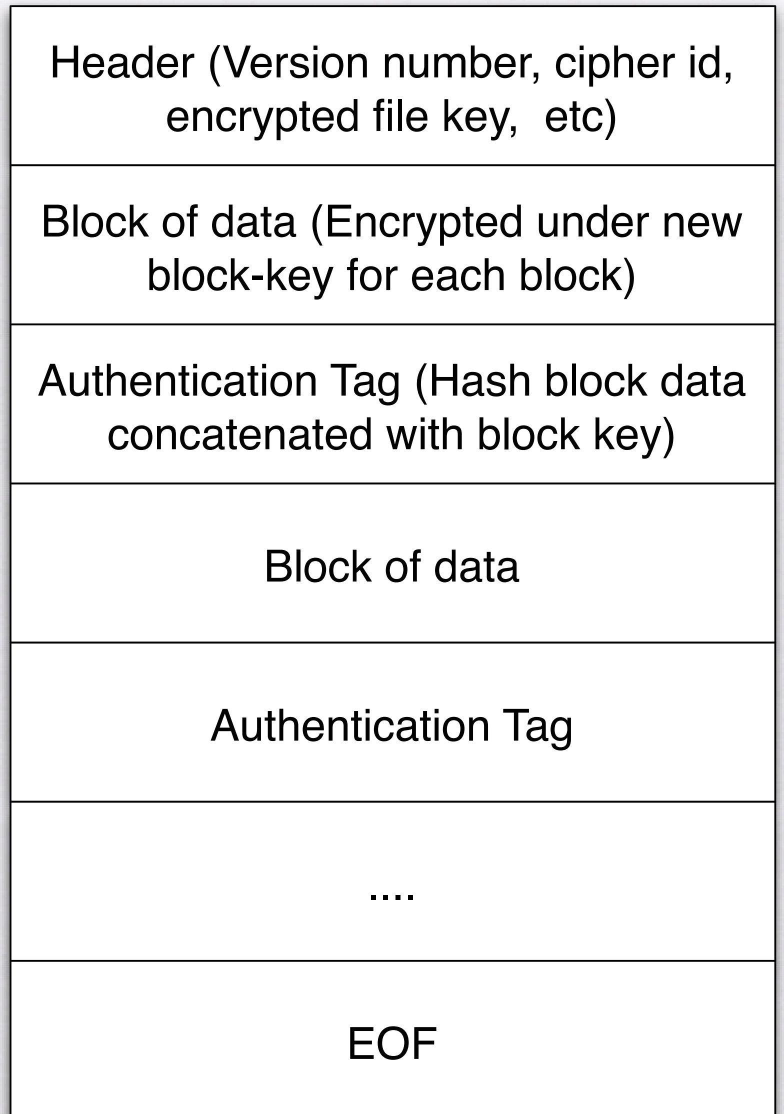
TCFS Model

- By Cattaneo, et. al. Usenix 2001.
- Distributed filesystem
- Server deals only with encrypted data
- User trusts his client machine, not the server housing data

TCFS Keys

- Each user has a master key
- For each file, a file key is randomly chosen
- For each block, a block key is formed.
  - Hash of file-key and block number

TCFS (cont)


Header (Version number, cipher id, encrypted file key, etc)
Block of data (Encrypted under new block-key for each block)
Authentication Tag (Hash block data concatenated with block key)
Block of data
Authentication Tag
....
EOF

TCFS - Achieved Security Goals

- Files cannot be read without file-key or user master key
- Cannot tell two cipher texts decrypt to the same plain text
- Cannot tell if two cipher blocks are the same plain text block
- Cannot reorder blocks
- Cannot modify blocks

Is TCFS Applicable?

- Requires accessing the block itself as well as the authentication tag
- Also requires accessing the header

Sirius Model

- Goh, et al. NDSS 2003.
- Data on an untrusted network file server
- Multi-user
- Provides access control

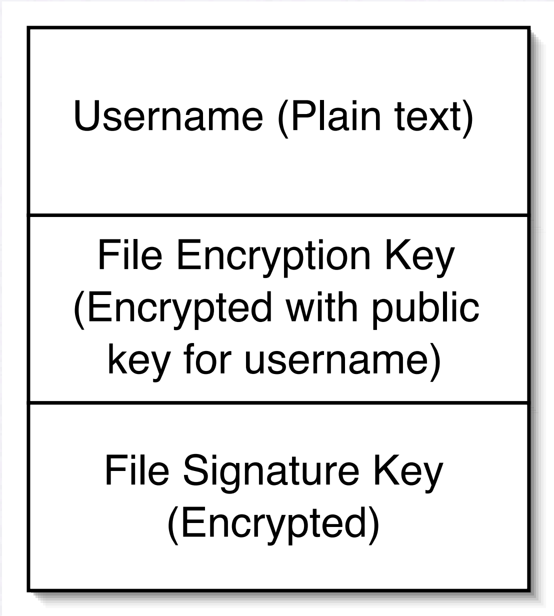
Sirius Keys

- FEK - File encryption key
- FSK - File signature key
- MEK - master encryption key
- MSK - master signature key
- User public/private keys

MD-File

Encrypted Key Block (Owner)	Encrypted Key Block (User 1)	File Signature Public Key (FSK)	Timestamp	Filename	Owner's Signature

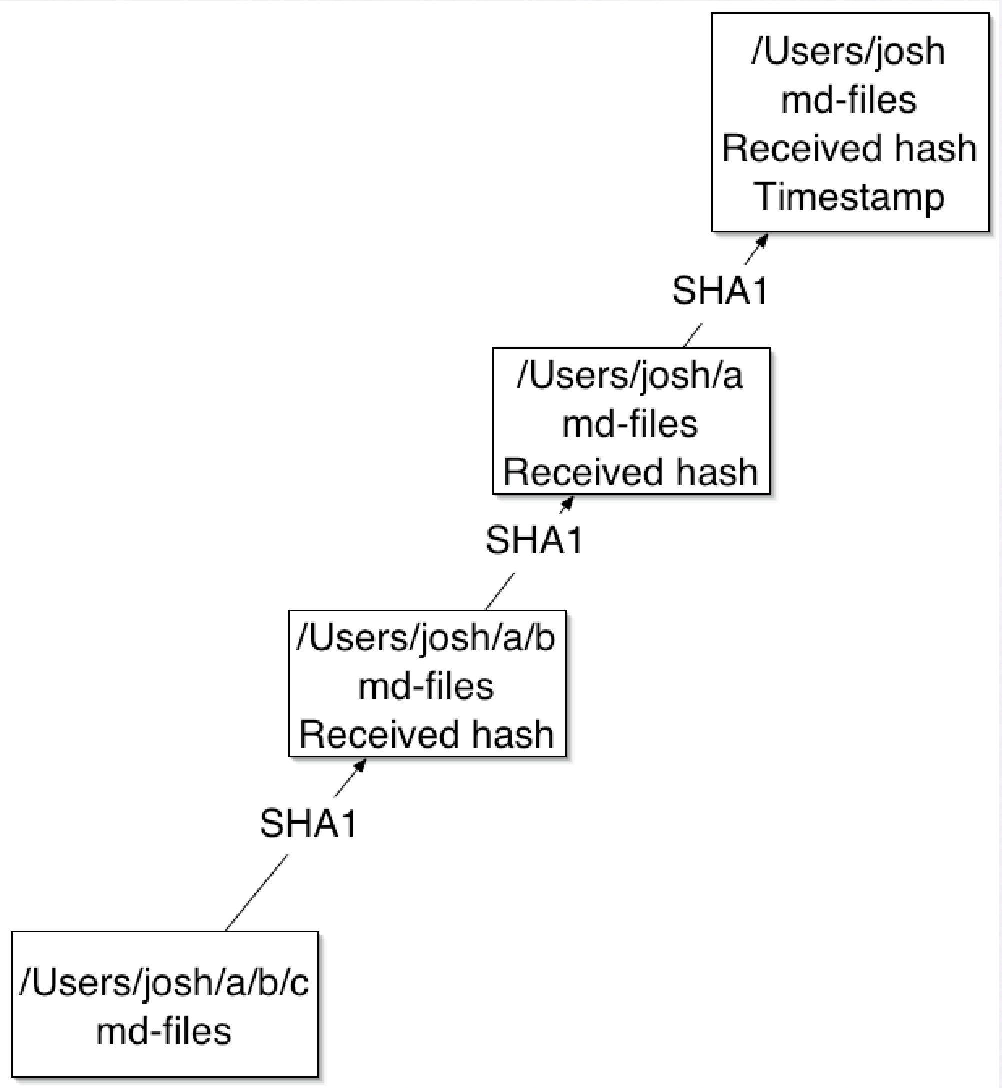
Encrypted Block Explained



Encrypted File

Encrypted File Data	Signature (Hash) signed with FSK

mdf-file


/Users/josh
md-files
Received hash
Timestamp
SHA1
/Users/josh/a
md-files
Received hash
SHA1
/Users/josh/a/b
md-files
Received hash
SHA1
/Users/josh/a/b/c
md-files

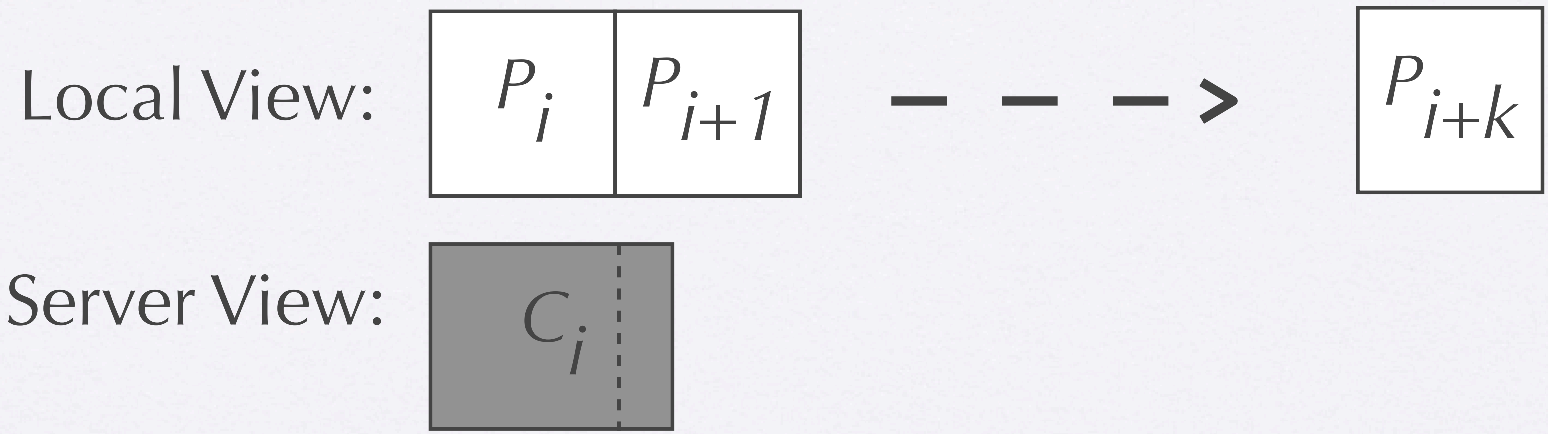
Is Sirius Applicable?

- This scheme requires accessing a file and verifying the signature
- Our model does not allow extra block accesses

Back to Current Model

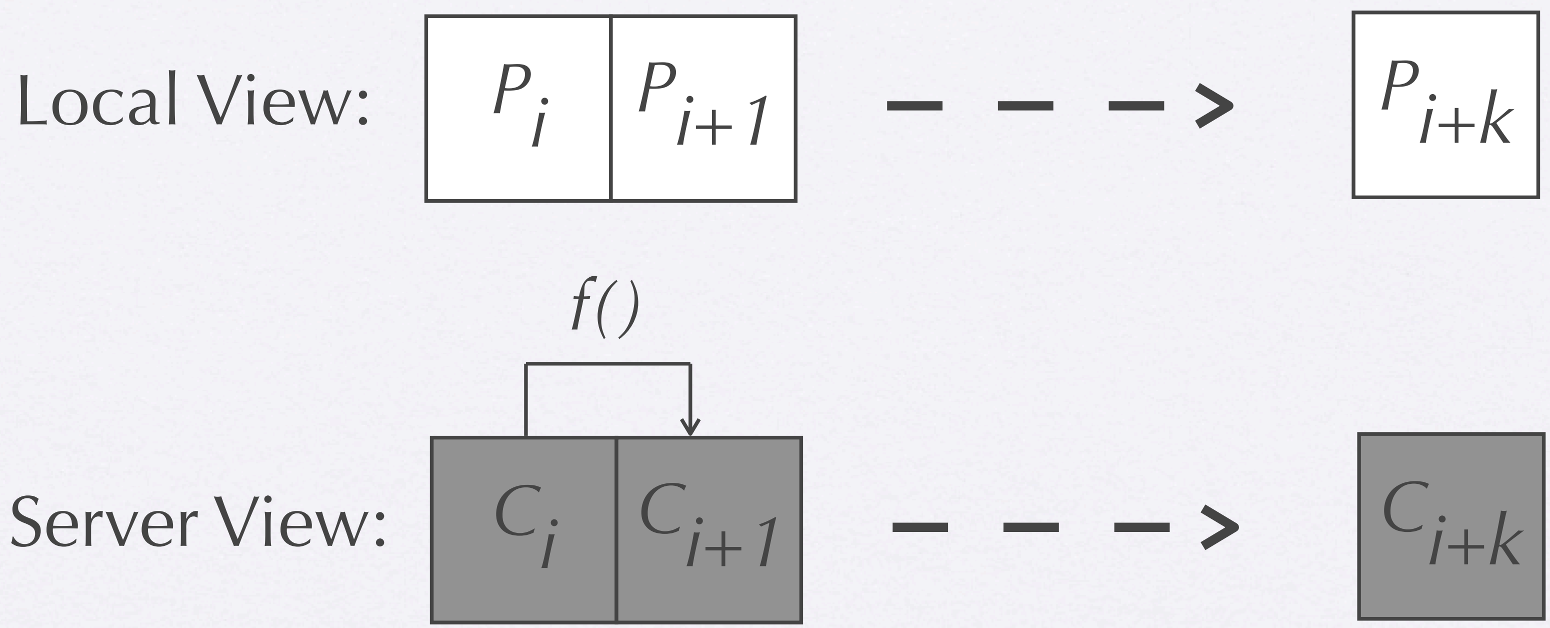
- Other Models achieve security, what about efficiency?
- Efficiency Mandates:
  - Space preserving encryption
  - Cannot Chain blocks (CBC)
  - Cannot store MACs remotely
  - No Signatures

Space Preserving *E()*



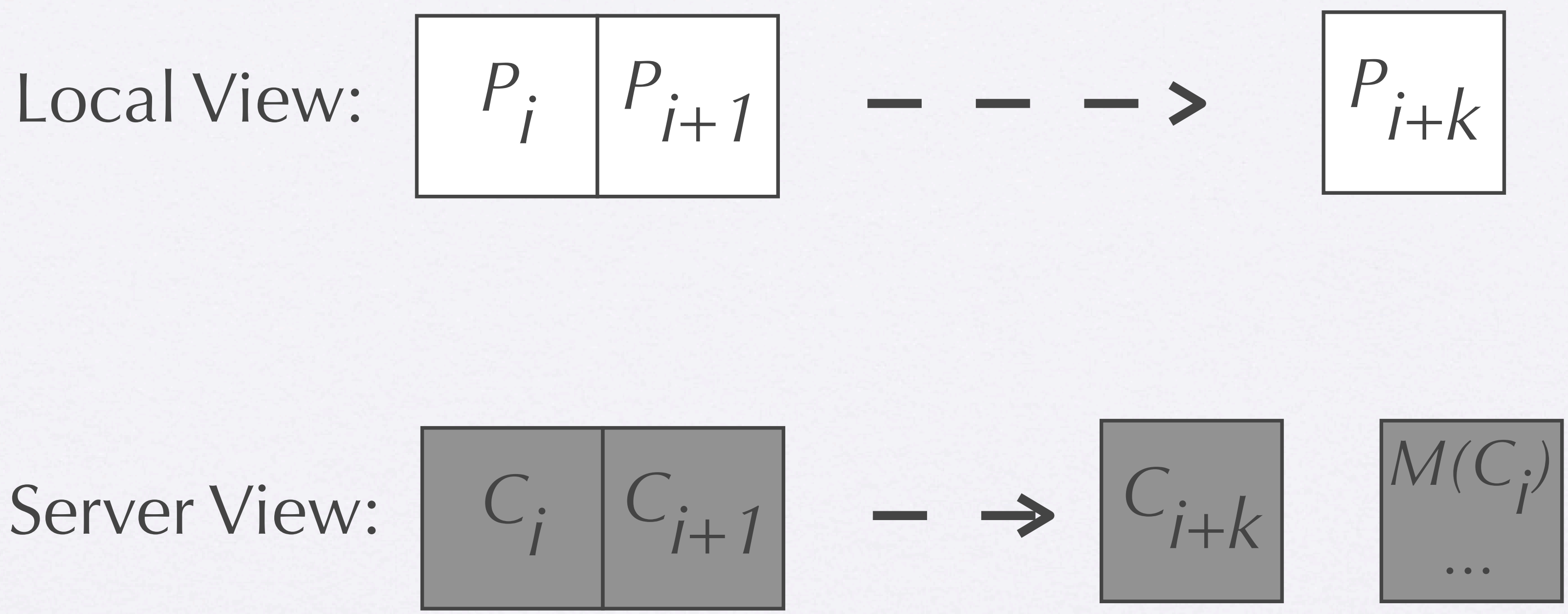
Two remote block access for each local block access!
Much slower

Chaining $E()$



Cannot chain to ensure diversity!

MACs



Cannot store MACs remotely

How to do things in place?

- Start with Encryption
- Return to integrity

In-place Encryption

- Block cipher with block length dividing disk block size
- Must be secure --- random
- Tweakable Block Ciphers
  - Liskov, Rivest, Wagner (Crypto ‘02)
  - Formalizes the concept

Tweakable Encryption

- Goal: provide another input to the ***BLOCK CIPHER*** to guarantee random encryption
  - ***NOT*** a Mode of Operation
  - Security of block cipher shouldn't depend on usage

Tweakable Encryption

- Formally:

$$\mathcal{E} : \mathcal{K} \times \mathcal{T} \times \mathcal{M} \rightarrow \mathcal{M} = E_K^T(M) = C$$

$$D_K^T(C) = M \leftrightarrow E_K^T(M) = C$$

$$\mathcal{K} = \{0,1\}^k$$

$$\mathcal{T} = \{0,1\}^t$$

$$\mathcal{M} = \{0,1\}^m$$

- Note: Not a mode of operation

- Security of scheme is ***not*** based on secrecy of the tweak

Not a new idea

- IVs are a form of tweak
- Hasty Pudding Cipher (R. Schroeppel)
- Mercy Cipher (L. Granboulan *et. al.*)
- OCB (Rogaway *et. al.*)

Bad Constructions

Similar to DESX:

$$E_K^{T_1,T_2}(M) = E_K(M \oplus T_1) \oplus T_2$$

T_1 and M are linked

M_a: 0**1**101100

T_a: 0**0**111101

M_b: 0**0**101100

T_b: 0**1**111101

Bad Constructions (2)

$$E_K^T(M) = E_{K \oplus T}(M)$$

Due to scheduling algorithms,
Some block ciphers don't use all key bits
(e.g., Loki and Lucifer --- Bihim, 1994)

Key: 0**1**010011

T1: 1**1**110010

T2: 1**0**110010

Provably-Secure Constructions

- Encrypting twice:

$$E_K^T(M) = E_K(T \oplus E_K(M))$$

Properties of Hashes

Second Preimage Resistance

Given x find x' s.t. $h(x) = h(x')$

Preimage Resistance

Given $h(x)$ find x

Collision Resistance

Find x, x' s.t. $h(x) = h(x')$

Provably-Secure Constructions (2)

- Involving special hash function

$$E_K^T(M) = E_K(M \oplus h(T)) \oplus h(T)$$

$$h : \mathcal{T} \to \mathcal{M}$$

Problematic in practice?
(SHA1 v. AES, MD5 v. AES-256)

Construction used in Paper

- "A Tweakable Enciphering Mode"
  - Halevi and Rogaway, Crypto '03
- Present CMC[E] (CBC-Mask-CBC)
  - Changes block cipher (e.g., AES) to a tweakable block cipher
  - CMC[E]'s block size > E's block size

CMC[E]

$$E^{T}_{K,K_2}(P_1 \ldots P_m):$$

$$\mathbb{T} \leftarrow E_{K_2}(T)$$

$$\mathbb{P} \leftarrow CBC[E](K, \mathbb{T}, P_1 \ldots P_m)$$

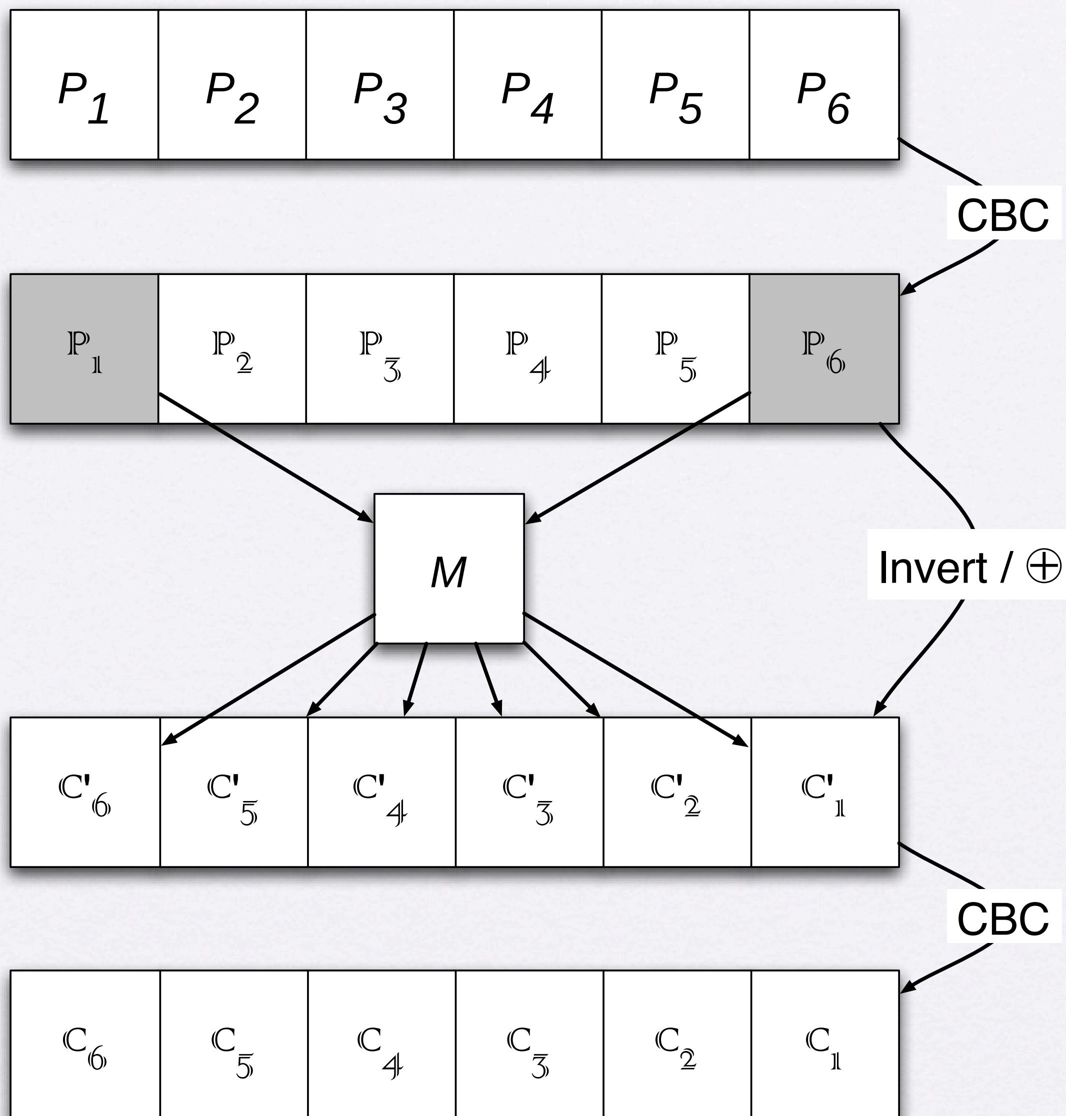
$$M \leftarrow 2(\mathbb{P}_1 \oplus \mathbb{P}_m)$$

$$\mathbb{C}' \leftarrow \mathsf{INV}\oplus(\mathbb{P}, M)$$

$$\mathbb{C} \leftarrow CBC[E](K, 0^{|\mathbb{T}|}, \mathbb{C}')$$

$$\mathbb{C}_1 \leftarrow \mathbb{C}_1 \oplus \mathbb{T}$$

$$\textbf{return } \mathbb{C}$$



P1 P2 P3 P4 P5 P6
CBC
P1 P2 P3 P4 P5 P6
M
Invert / ⊕
C'6 C'5 C'4 C'3 C'2 C'1
CBC
C6 C5 C4 C3 C2 C1

CMC[E] (2)

- Decryption: invert E, same algorithm
- Notes:
  - 2m+1 calls to E
  - Provably secure (reduces to security of E as a PRP)

How to do things in place? (2)

- MACs
- Offload to client (now hashes)
    - Reduces remote block-accesses
- How can we do this efficiently?

Generic Secure Storage System

Generic Storage Scheme

- INIT
  - generates keys
- E (K, bid, m)
  - outputs ciphertext
- D(K, bid, c)
  - outputs plaintext

Generic Storage Scheme (2)

- WRITE (*K*, *bid*, *M*)
  - $E_K^{bid}(M) = C$ send *C*, *bid* to server
- READ (*K*, *bid*, *C*)
  - $D_K^{bid}(C) = M$ receive *M* from server
- VER(*M*, *bid*)
  - Verifies that *M* is valid

Three schemes

- Naive (S1) -- Motivational Example
- Efficient (S2) -- Efficient, lacking in security
- Hybrid (S3) -- Less efficient, secure

S1

- WRITE
  - Send $E_K^{bid}(M) = C$ to server
  - store bid, SHA1(M)
- READ
  - Receive $D_K^{bid}(C) = M$ from server
- VER
  - check SHA1(M) with stored version

S1 (2)

- Security: server cannot insert data
  - Would break second-preimage resistance
- Efficiency: store 22-24 bytes per block!
  - 2% extra on 1024 byte block
- (SHA1 per verification)
- Can we do better?

S2

- Selectively store hashes of plaintext
- Which ones?
  - Relation between CMC[E] and PRPs
  - if C is modified, or decrypted with wrong tweak, $D_K^{bid}(C) = M$ will have random output (high entropy)

Sidenote on Entropy

- Informally:
  - Measure of uncertainty
  - bits of information in a string
  - theoretical lower bound on compression
- ciphertext has high entropy

Entropy (2)

- Formally if $X \sim p(x)$

$$H(X) = \sum_{x \in \mathcal{X}} -p(x) \log p(x)$$

Entropy (3)

- Examples (range is a 2 bit space)
- Example: 1,4,2,1,1,3,2,1 (realization of *X*)

$$H(X) = \frac{1}{2}\log 2 + \frac{1}{8}\log 8 + \frac{1}{4}\log 4 + \frac{1}{8}\log 8 = \frac{7}{4}$$

Entropy (4)

- Example: 1,4,2,3,1,3,2,4 (realization of *X*)

$$H(X) = \frac{1}{4}\log 4 + \frac{1}{4}\log 4 + \frac{1}{4}\log 4 + \frac{1}{4}\log 4 = 2$$

- Example: 1,1,1,1,1,1,1,1 (realization of *X*)

$$H(X) = 1\log 1 = 0$$

Back to S2

- When to store hash of data?
- Need to differentiate between tampered ciphertexts and legitimate random data
- Only store hashes for random data
- How to determine... IsRand(M)
    - Compares H(M) to a threshold (τ)

IsRand

- Two versions: based on range of *X*
  - 4 bit range and 8 bit range
  - Partition blocks into chunks, compute H()
  - Compare to **τ**

Computing threshold

- Determine τ:
  - Compute entropy of Random 1K blocks
  - 8 bit: 7.73-7.86 bits $\tau = 7.73$
  - 4 bit: 2.55-2.64 bits $\tau = 2.55$

S2 Modifications

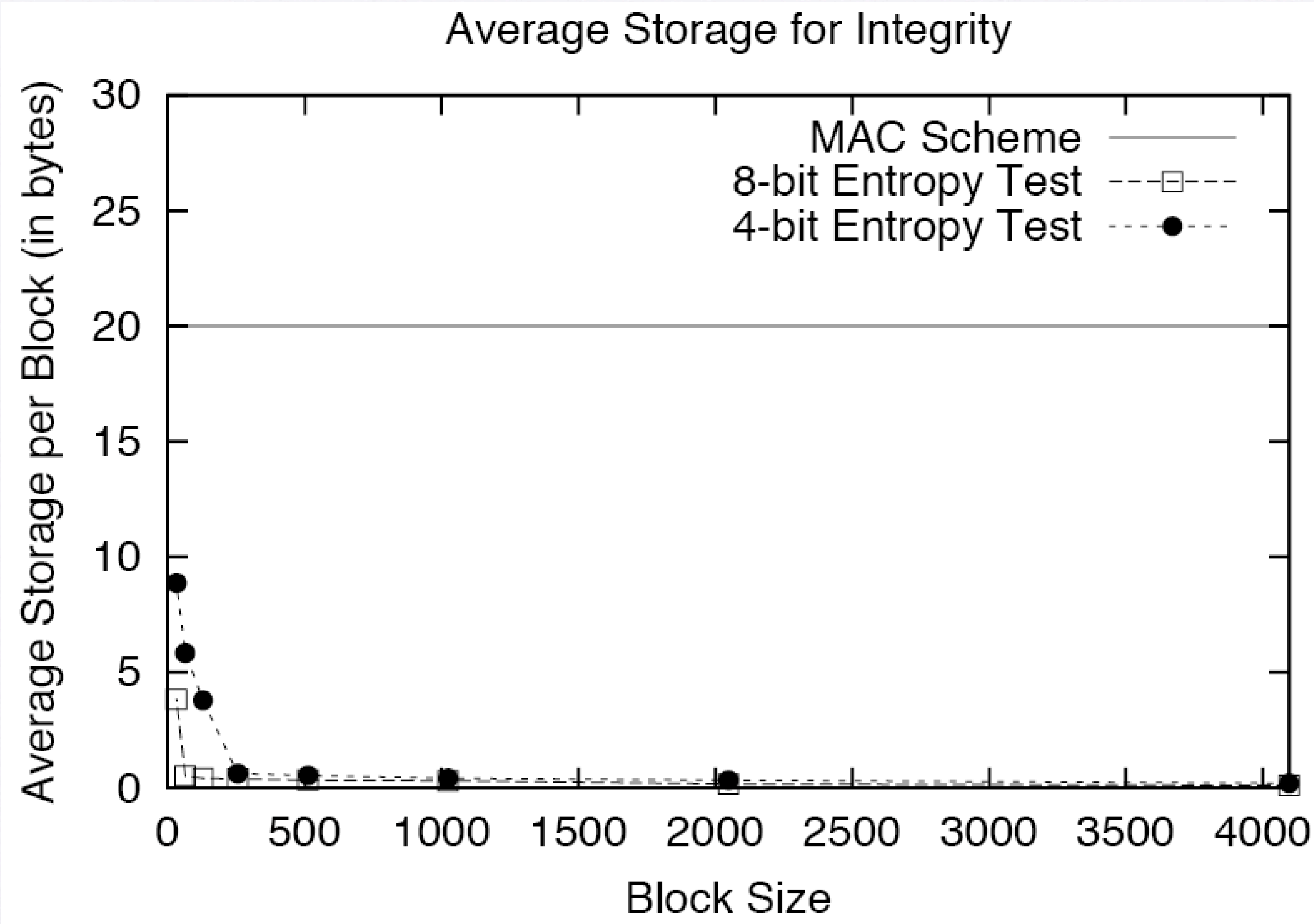
- Write:
  - compare IsRand(M) to **τ** (store hash)
  - proceed as before
- Ver:
  - compute IsRand(M) (check hash)

Experiments

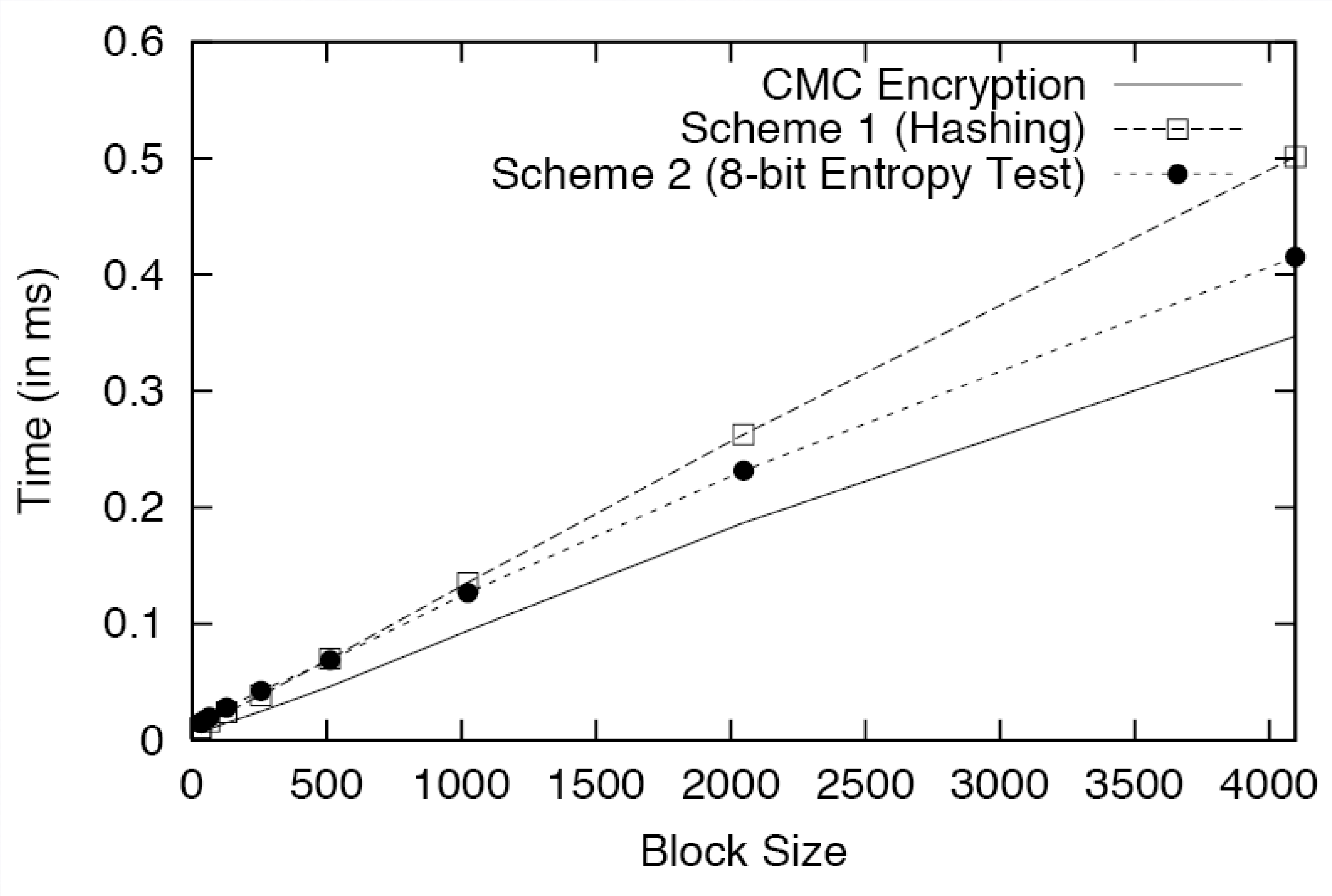
Experimental Setup

- Collected 1 month of disk traces
- One user, normal load
- 200 MB disk
- 1K blocks (some tests varied this)

S2 Performance



S2 Performance



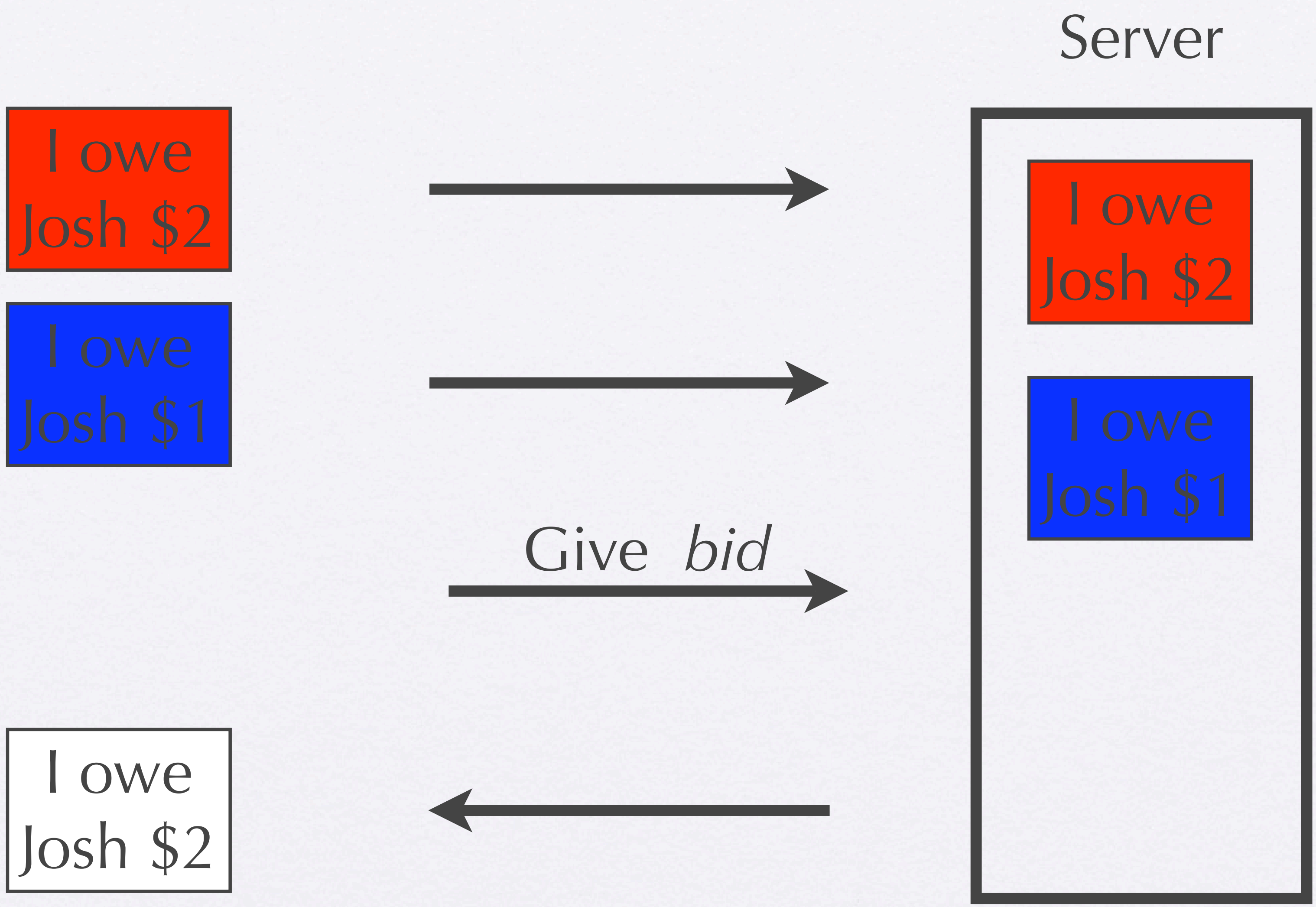
S2 Security

- Server cannot trick (with high probability) a client into accepting a block that has not been written.
- What about replays?

False Negative Rate

- Pr. that a block that is modified decrypts to a sequence with H < τ (and is therefore accepted)
- for 1024 byte block
  - 4 bit test: false neg. of ~ 2^{-90} (hash?)
  - 8 bit test: false neg. of ~ >>2^{-90}

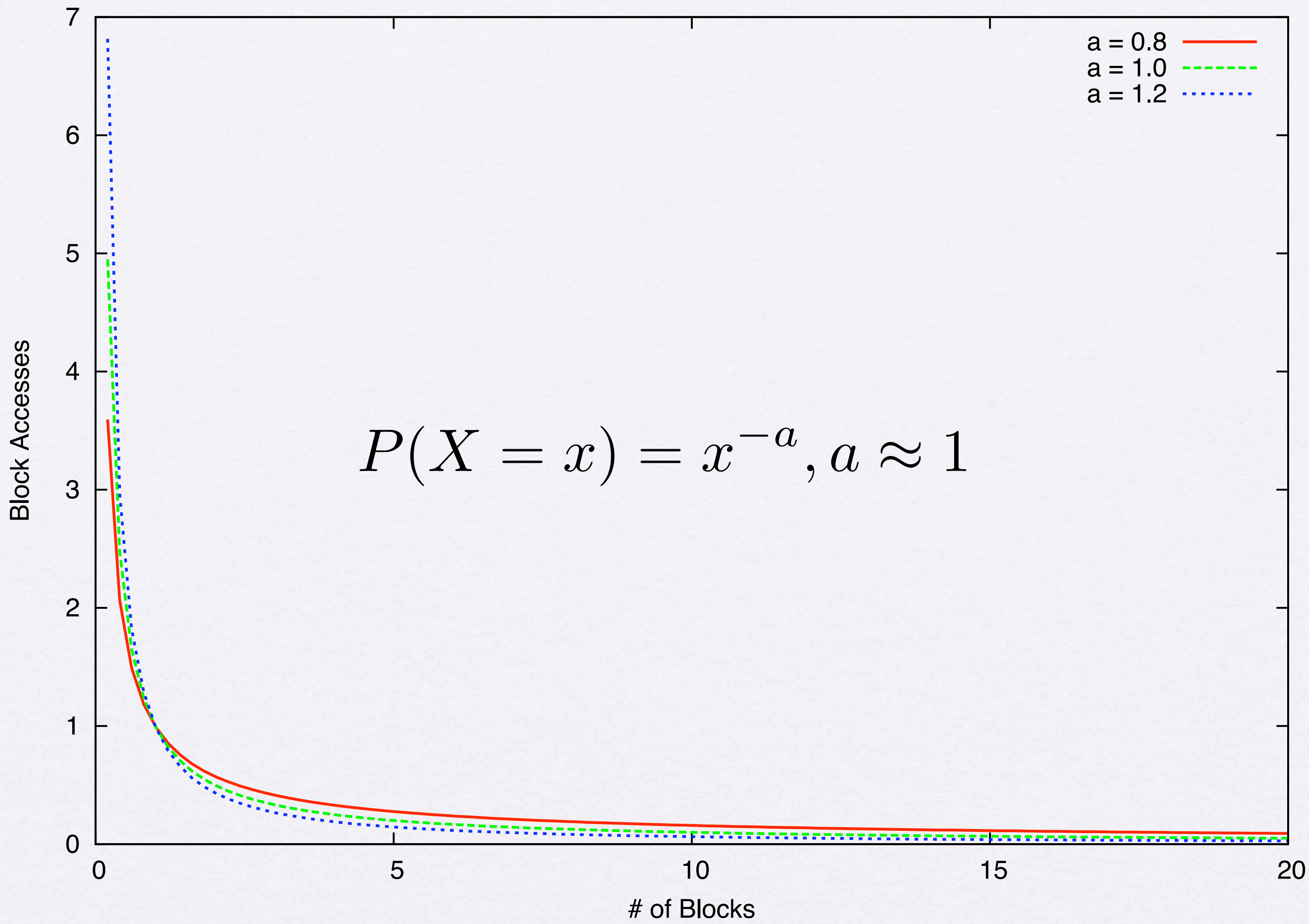
S2 Security


Server
I owe Josh $2
I owe Josh $1
Give *bid*
I owe Josh $2
I owe Josh $1
I owe Josh $2

How can we fix it?

- Only a problem if we write to a block twice
- Fortunately, block access follow Zipf dist.
  - i.e., few blocks accessed frequently
  - many blocks accessed once

Zipf distribution



Changes

- Associate tweaks with # of writes
- Store a flag for each block
- On write, mark the flag
- On second write, increment a counter (c)
- Change E(), D():

$$E_K^{bid||c}(M) \qquad D_K^{bid||c}(C)$$

- Recall Construction with Tweaks

Storage Comparison

S1	S2	S3
16.263 MB	0.022 MB	0.351 MB

813,124 distinct blocks, 113,785 written only once

Do these numbers seem to add up? (no)

Conclusion

- Model: untrusted SAN
- Provide confidentiality/integrity within limited model
- Does so efficiently
- Provides Theoretical ***AND*** Analytical results

Neat Tricks

- Exploit Entropy of bad decryptions
- Exploit File Access Patterns

References

- E. Biham, "New types of Cryptanalytic Attacks using Related Keys," Journal of Cryptology, Fall 1994.
- G. Cattaneo, L. Catuogno, A. Del Sorbo, P. Presiano, "The Design and Implementation of a Transparent Cryptographic File System for UNIX," USENIX 2001.
- E. Goh, H. Shacham, N. Modadugu, D. Boneh, "SiRiUS: Securing Remote Untrusted Storage," NDSS 2003.

References (2)

- S. Halevi, P. Rogaway, "A Tweakable Enciphering Mode," Crypto 2003.
- A. Oprea, M. K. Reiter, K. Yang, "Space-Efficient Block Storage Integrity," NDSS 2005.
- M. Liskov, R. L. Rivest, D. Wagner, "Tweakable Block Ciphers," Crypto 2002.