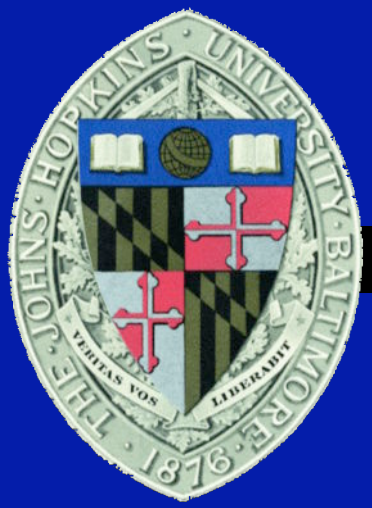
Managing the Lifetime of Versions in Digital Archives

Randal C. Burns

for the Digital Government (dg.o)

DIGARCH PIs Meeting

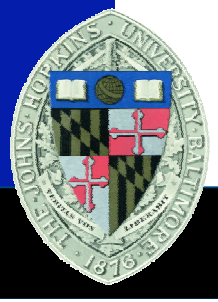
17 May 2005



Department of Computer Science, *Johns Hopkins University*

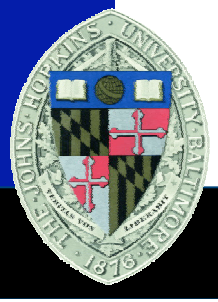
Project Goals

- Secure deletion for versioning archives
  - systems compliant with the security and auditability mandates of federal legislation
  - support privacy rights of individuals
  - limit liability of data owners and managers
- Development of technology
  - storage system and cryptographic tools
- Release an open-source file system
  - inexpensive compliance and privacy for everyone



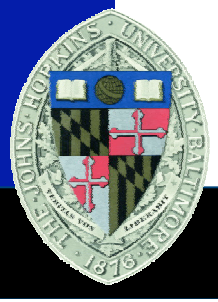
Regulating the Paperless World

- Congress and the courts are addressing the importance of managing electronic records
- Over 4,000 laws and regulations
  - corporate records and auditing (Sarbanes-Oxley, 2002)
  - Federal Information Security Management Act (2002)
  - Federal Records Act
- Some with explicit deletion requirements
  - Health Insurance Portability and Accountability Act (1996)



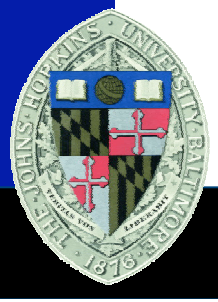
Fine-Grained, Secure Deletion

- Secure deletion = deleted data are irrecoverable
  - to the owner of the data or system administrators
  - when an adversary has physical access to a disk
  - when an adversary has encryption keys
- Fine-grained = a single version of a file may be deleted
- Present systems aren't good enough
  - free data blocks for use in future allocations
  - even after reallocation, overwritten data may be recovered



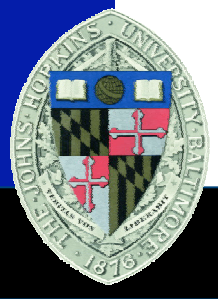
The Need for Secure Deletion

- For privacy protection
  - re-classifying information involves deletion
  - when a disk is retired or stolen
  - patients have the right to redact portions of their records
- To limit liability
  - records that go out of audit scope should do so forever
- Even in permanent archives
  - as part access control, changing policy
  - for storage management, any time data are moved



Obstacles to Secure Deletion

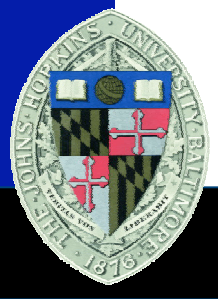
- Existing solutions do not translate to versioning archives
- Secure overwriting is untenably slow
  - data blocks are overwritten many times with alternating patterns of 1s and 0s
  - magnetic media is degaussed
- Cryptographic techniques are not fine-grained



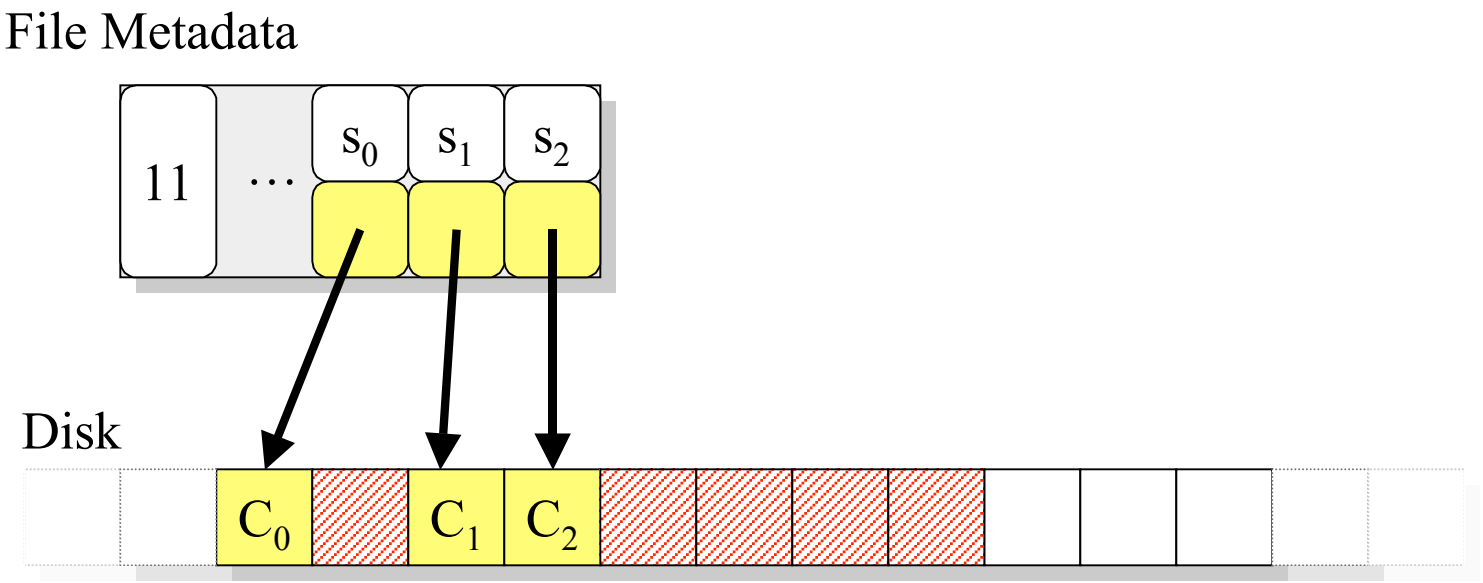
The Central Idea

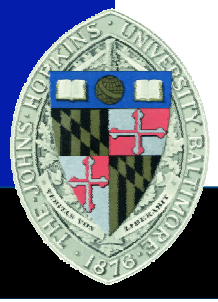
$$f_k(B_i, N) \rightarrow C_i \,||\, s_i$$

- A keyed transform
  - converts a data block and a nonce
  - into an encrypted block and a ***stub***
- When the key is private, data are secure and authenticated
- Securely deleting stub, securely deletes block, even after the key has been exposed



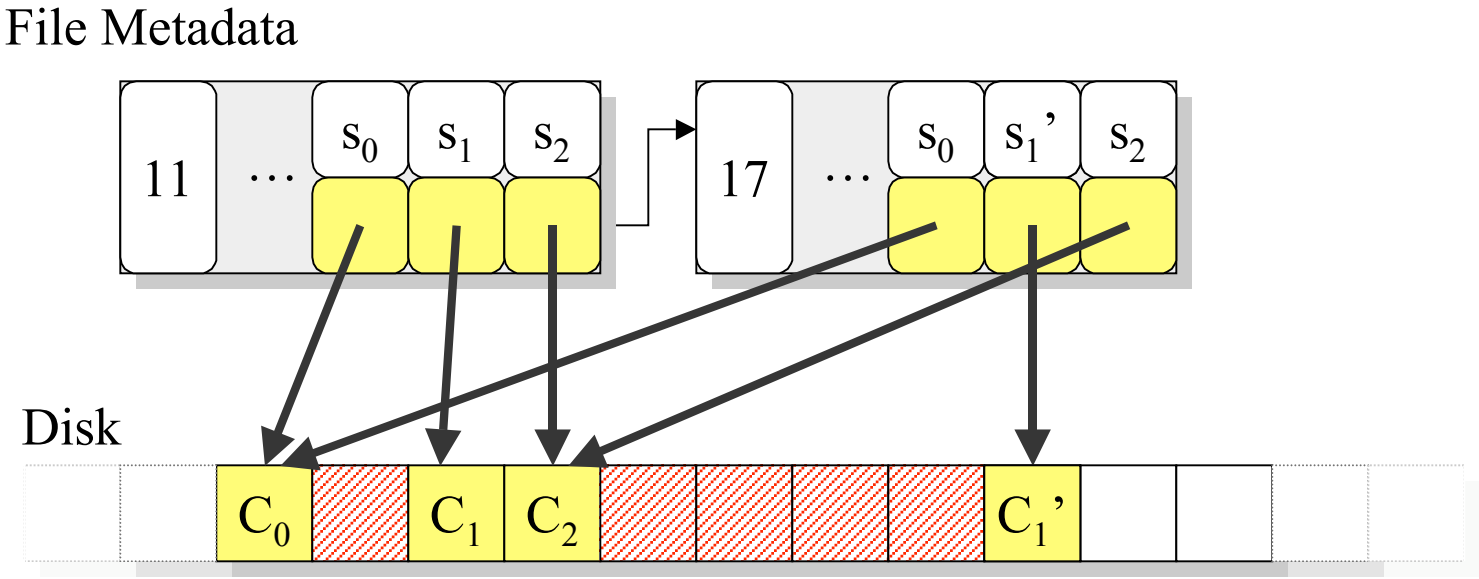
Secure Deletion Example

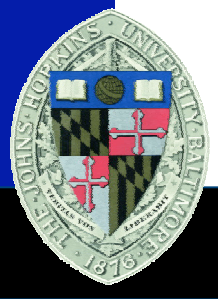




Secure Deletion Example

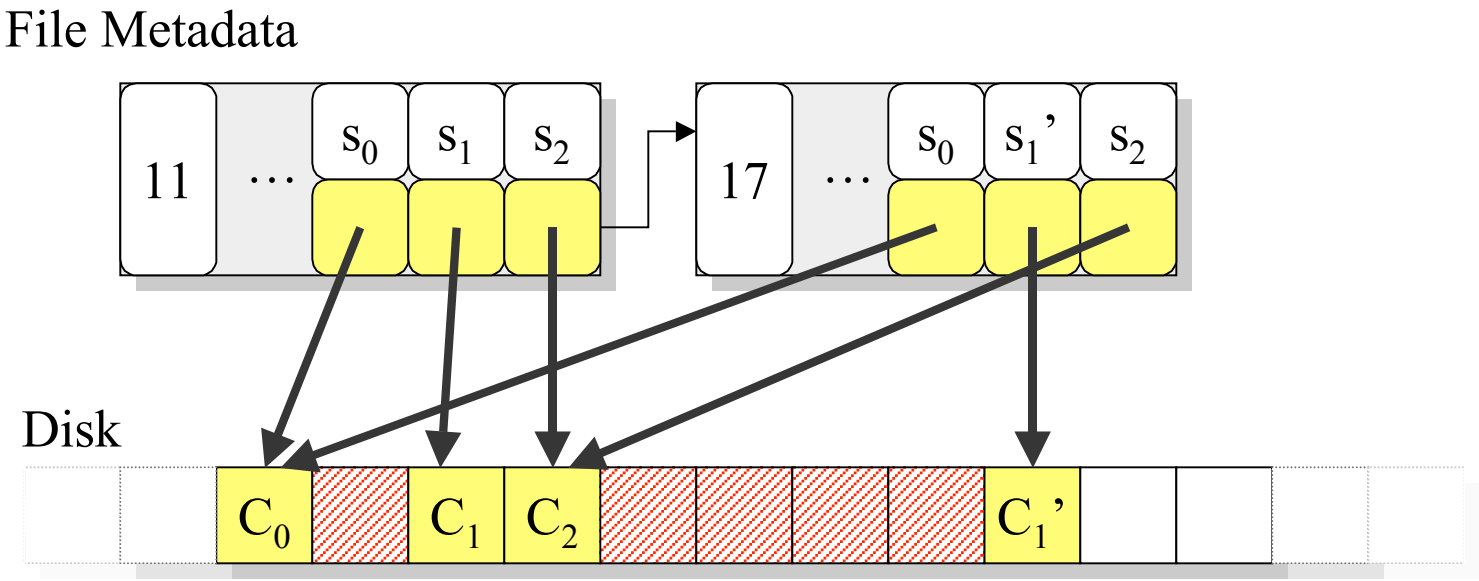
Receive a write to block #2
at time 17

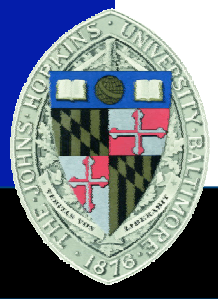




Secure Deletion Example

Delete file at time 11

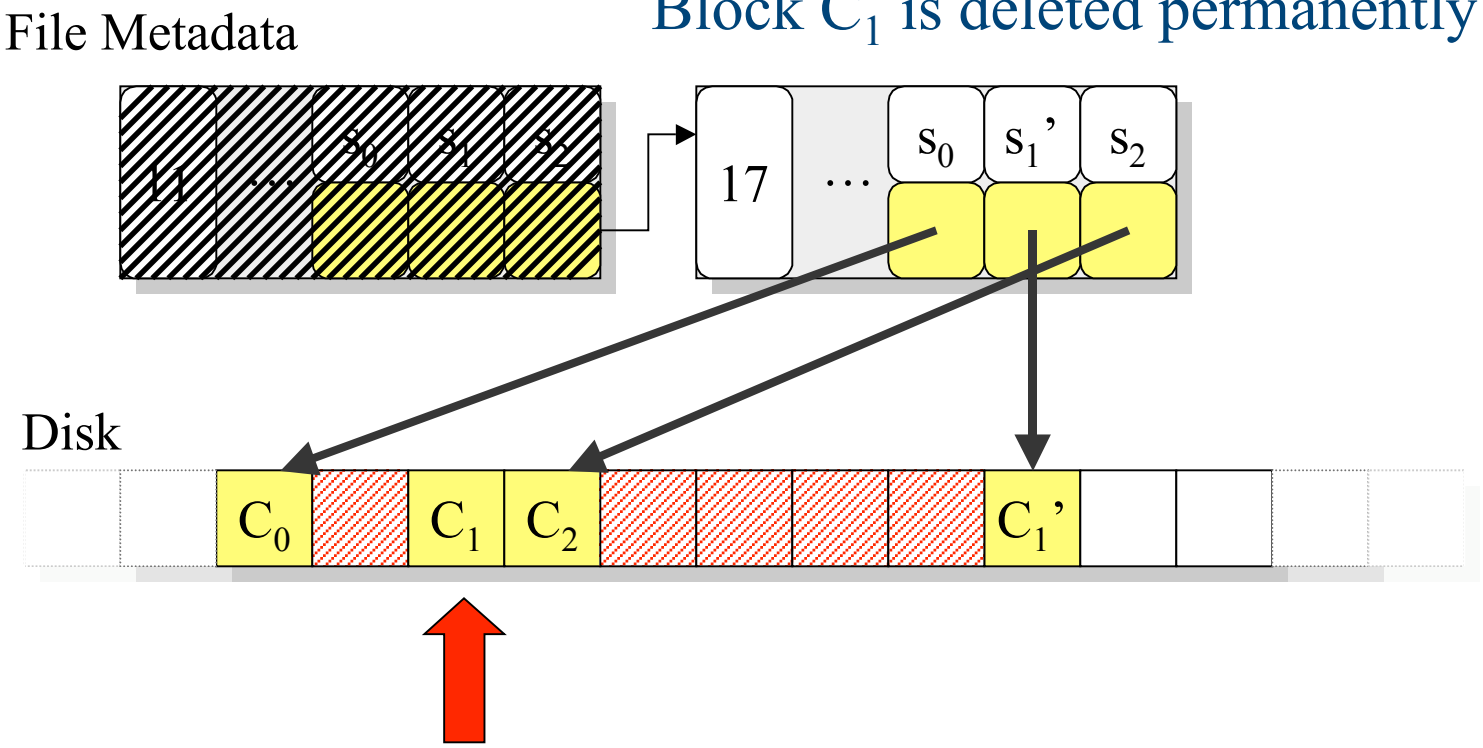


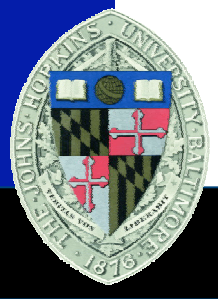


Secure Deletion Example

Delete file at time 11

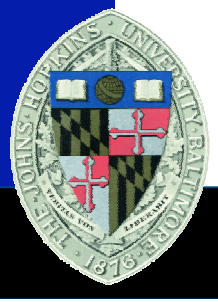
Block C_1 is deleted permanently





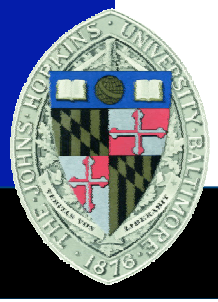
Features of our System Design

- Stubs are not secret
  - stored on disk as part of metadata
- Stubs make for efficient, secure deletion
  - stubs are stored contiguously
  - delete a large amount of data (1 MB) by overwriting a small, contiguous region of stubs (4 KB)
- Increases deletion performance by a factor of a thousand or more
  - when compared with secure overwriting
  - depending upon file size and system block size



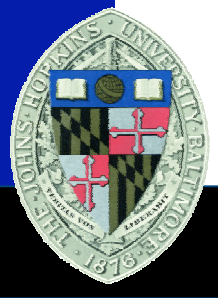
Applicability of Secure Deletion

- For systems that
  - use disk encryption
  - share-content between files or versions
- This includes versioning file systems and content-indexing archives



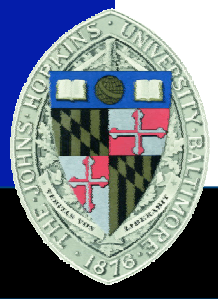
Project Tasks

- Development of secure deletion algorithms
  - provable security
  - minimize space overhead
- System development
  - compliance features for our ext3cow open-source, versioning file system for Linux
  - build into content sharing archives
- Key management for versions



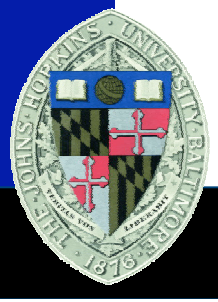
Research Directions

- Secure deletion across multiple replicas
  - delete a file system image and its backup(s)
  - ability to delete and fault-tolerance compete
- Strong auditability
  - provably secure version histories



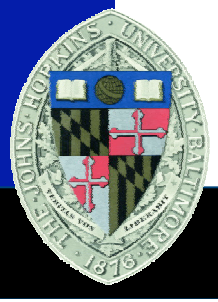
A Paperless World

- Information is becoming entirely electronic
  - financial records, medical records, federal data
  - 300 million computers storing 150,000 terabytes
- Tradeoffs in electronic record keeping
  - eases use, sharing, and indexing/searching
  - creates a new set of vulnerabilities
    - exposure of data that are deleted or discarded
    - the undetected modification of archived data



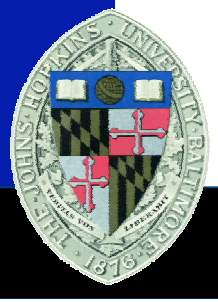
Distilling Regulatory Requirements

- Audit Trail
  - Files should be versioned over time
  - Versions need to accessible in real-time
- Secure Storage
  - Privacy and confidentiality
- Authentication and Non-repudiation
  - Binding a person to the changes they make
  - Able to make a strong statement about the validity of data



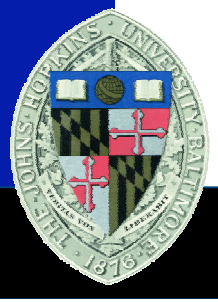
Existing Solutions

- Secure Overwrite [Gutmann 1996]
  - data blocks are overwritten many times with alternating patterns of 1s and 0s
  - magnetic media is degaussed
- Key Disposal [Boneh & Lipton 1996]
  - data encrypted with a key
  - key is securely deleted, eliminating meaningful data access



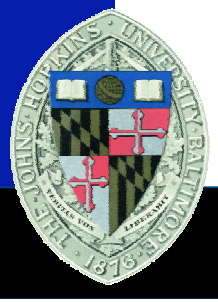
The Ext3cow File System

- Open-source file system that implements file system snapshot and versioning
  - Captures immutable, point-in-time views of the entire file system
- Novel and intuitive *time-shifting* interface for accessing the past
- Encapsulated entirely in the file system
- Low storage overhead and negligible performance degradation



Ext3cow Status

- Fully implemented file system available at: *www.ext3cow.com*
  - Thousands of visitors, hundreds of downloads
- Active development mailing list
- Ext3cow being used as the foundation of other research and industrial projects
  - JHU, UCB, UCSC, Columbia, USC
  - Infrant Technologies
- A paper on the implementation of ext3cow to appear in *ACM Transactions on Storage*, May, 2005



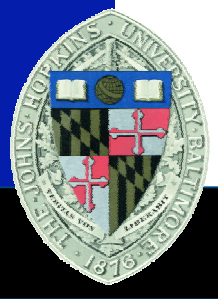
Our Algorithms

All-or-Nothing Deletion

- In AON, all ciphertext blocks must be present in order to decrypt a block
- The stub is an expansion of the encrypted data
- Without stub, data is irrecoverable
- Efficient, however, weak against known-plain text attacks

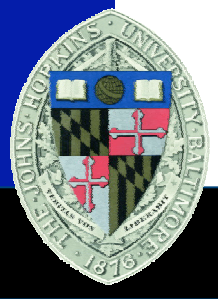
Random Key Deletion

- Create a random key for every block encrypted
- Encrypt data with random key
- Stub is the encryption of random key with the user's key
- May be slower, requires more space



Electronic Record Legislation

- **HIPAA (1996)**
  - Technical security mechanisms
  - Physical safeguards
- **E-SIGN (2000)**
  - Digital contracts are as legitimate as paper contracts
- **FISMA (2002)**
  - Framework for ensuring security controls for storage
  - Security of system must be commensurate with security of data
- **Sarbanes-Oxley (2002)**
  - CEO, CFO responsible for accurate financial reports
  - Management assessment of internal controls
  - Real time disclosure
  - Criminal penalties for altering documents
- **Gramm-Leach-Bliley (2002)**
  - Consumer records kept confidential
  - Protect against threats and unauthorized access
- **Federal Records Act**
- **DoD Directive 5015.2**



AON Encryption

Input: Data $d_1, \ldots d_n$, Block ID id, Counter x, Encryption key K, MAC key M

$$1: ctr_1 \leftarrow id \,||\, x \,||\, 1 \,||\, 0^{128-|x|-|id|-1}$$

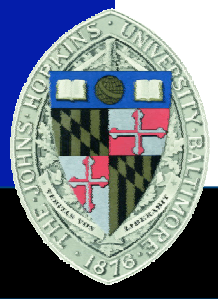
$$2: c_1, \ldots, c_m \leftarrow AES-CTR_K^{ctr_1}(d_1, \ldots, d_m)$$

$$3: t \leftarrow HMAC-SHA-1_M(c_1, \ldots, c_m)$$

$$4: ctr_2 \leftarrow id \,||\, x \,||\, 0^{128-|x|-|id|}$$

$$5: x_1, \ldots, x_m \leftarrow AES-CTR_t^{ctr_2}(c_1, \ldots, c_m)$$

$$6: x_0 \leftarrow x_1 \oplus \ldots \oplus x_m \oplus t$$



Output: Ciphertext $x_1, \ldots x_m$, Stub x_0

Random Key Encryption

Input: Data $d_1, \ldots d_n$, Block ID id, Counter x, Encryption key K, MAC key M

$$1: k \xleftarrow{R} \mathrm{K}_{AE}$$

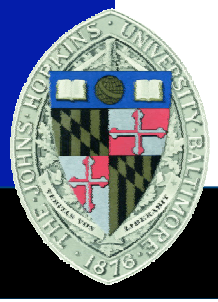
$$2: nonce \leftarrow id \,||\, x$$

$$3: c_1, \ldots, c_m \leftarrow AE_k^{nonce}(d_1, \ldots, d_n)$$

$$4: ctr \leftarrow id \,||\, x \,||\, 0^{128-|x|-|id|}$$

$$5: c_0 \leftarrow AES - CTR_K^{ctr}(k)$$

$$6: t \leftarrow HMAC - SHA - 1_M(ctr, c_o, r)$$



Output: Ciphertext $c_1, \ldots c_n$, Stub x_0, t, $c_1, \ldots c_m$