Limiting Liability in a Federally Compliant File System

Zachary N. J. Peterson

The Johns Hopkins University

Hopkins Storage Systems Lab, Department of Computer Science

Regulatory Requirements

- Data Maintenance Acts & Regulations
  - HIPAA, GISRA, SOX, GLB
  - 4,000+ State and Federal Laws and Regulations with regards to storage
- Audit Trail – creating a “chain of trust”
  - Files are versioned over time
  - Authenticated block sharing (copy-on-write) between versions.
- Disk Encryption
  - Privacy and Confidentiality
  - Non-repudiation

Secure Deletion in a Regulatory Environment

- Desire to limit liability when audited
  - Records that go out of audit scope do so forever
  - When a disk is subpoenaed old or irrelevant data are inaccessible
- Existing Techniques
  - Secure overwrite [Gutmann]
  - File key disposal in disk encrypted systems [Boneh & Lipton]
- Existing solutions don't work well in block-versioning file systems

Technical Problems

- Secure overwriting of noncontiguous data blocks is slow and inefficient
  - When versions share blocks, data to be overwritten may be noncontiguous
- Cannot dispose file keys in a versioning file system
  - Blocks encrypted with a particular key need to be available in future versions
- User space tools are inadequate
  - Can't delete metadata
  - Can't be interposed between file operations
  - Truncate may leak data
  - Difficult to be synchronous

The Big Idea

$$E_{K_x}(B_i) \rightarrow \tilde{B}_i, l_i$$

$$|B_i| = |\tilde{B}_i|$$

$$D_{K_x}(\tilde{B}_i, l_i) \rightarrow B_i$$

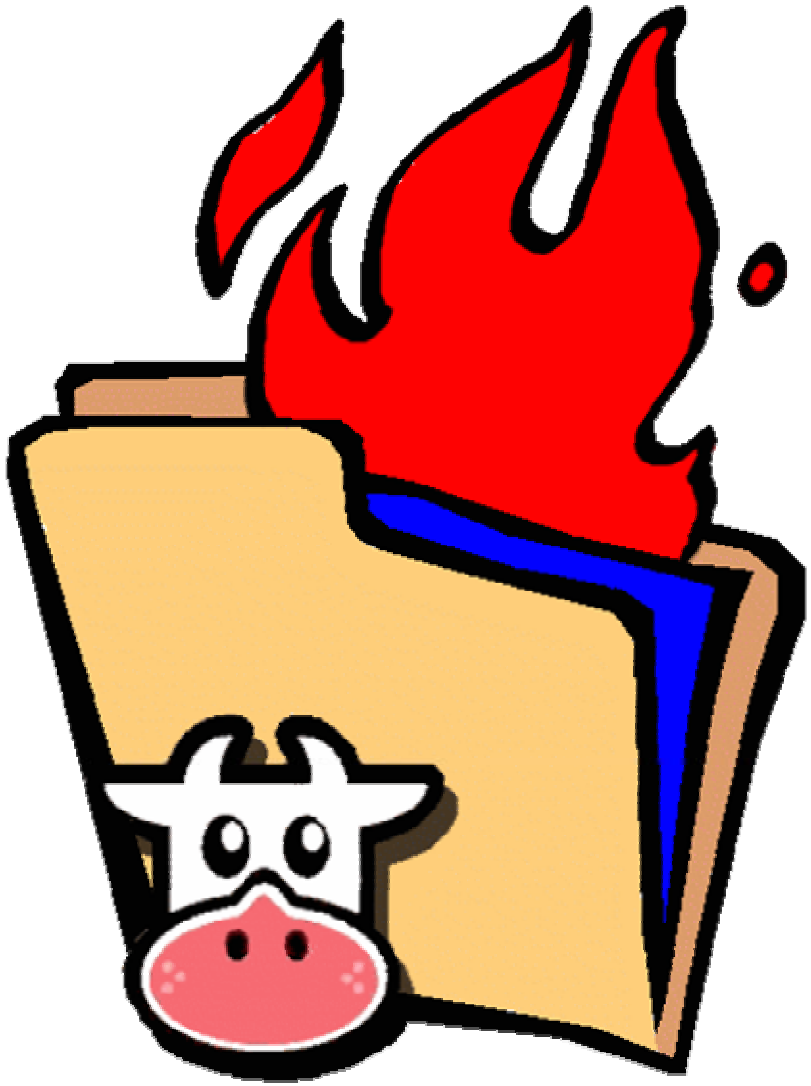
- Encryption creates an encrypted block and a small 128-bit stub (l_i)
- Securely deleting stub, securely deletes block

Features of our System

- Stubs are stored with the inode
- When a version is deleted, the inode and its stubs are securely overwritten
  - In effect securely removing all data associated with that version
- Secure deletion with minimal secure overwriting
  - More efficient than securely overwriting noncontiguous data blocks
  - Versions may use the same encryption key
  - Does not increase key overhead

Interested?

- We have a fully working (insecure) snapshot file system for the Linux 2.4 kernel (2.6 soon).
  - Time-shifting
  - Snapshot reclamation
- Web site: www.ext3cow.com
  - Download the patch.
  - Read the technical report.
  - Join the mailing list.
- Email: zachary@jhu.edu



The Law and Storage

- Health Insurance Portability and Accountability Act (HIPAA)
- Government Information Security Reform Act (GISRA)
- Federal Information Security Management Act (FISMA)
- Sarbanes-Oxley (SOX)
- Gramm-Leach-Bliley (GLB)
- PATRIOT Act
- Federal Records Act
- DoD Directive 5015.2
- 4,000+ State and Federal Laws and Regulations with regards to storage.

What are the requirements?

- Authorization
  - Access controls (role-based authorizations)
  - Encryption (confidentiality)
  - Digital signatures (non-repudiation)
- Authentication
- Audit Trail
  - Record of all changes
- Secure Storage and Transmission
  - More encryption?
- Integrity & Reliability
  - Unaltered records.
  - Trusted content.

Introducing Ext3cow

- A file system based on ext3 that supports file system *snapshot* with a *time-shifting* interface.
  - Creates immutable views of a file system as it appeared at a specific point in time.
  - Versions of a file are created with copy-on-write (cow) of blocks.
  - Snapshots are addressed with an epoch number that corresponds to a system time (`gettimeofday`).

Securing our COW file system

- Challenges
  - How to encrypt files that share blocks between versions.
  - How to change permissions such that a user who had access to a file in the past is not able to access current versions.
  - Securely deleting files such that they are no longer able to be subpoenaed.