Limiting Liability in a Federally Compliant File System

Zachary N. J. Peterson Randal Burns Adam Stubblefield

Department of Computer Science
The Johns Hopkins University
{zachary,randal,astubble}@cs.jhu.edu

1 Policy and Problem Statement

Congress has begun to explicitly addressed the importance of maintaining and securing electronic information, be it personal health information, top-secret defense data, or the accounting information of a publicly traded company. There exist over 4,000 local, state and federal acts and regulations with regard to storage, all with a varying range of requirements for securely maintaining electronic records. Examples include the Health Insurance Portability and Accountability Act (HIPAA) of 1996, the Gramm-Leach-Bliley Act (GLBA) of 1999, and the more recent Federal Information Security Management Act (FISMA) and Sarbanes-Oxley Act (SOX) of 2002. Some acts and regulations require the use of strong encryption for privacy, confidentiality and non-repudiation, as well as a means for securely transmitting data. Some legislation requires an auditable trail of changes made to electronic records that is accessible in real-time. This involves versioning files over time and providing a means of quickly retrieving versions from a particular point in time.

The ability to securely delete electronic records is as important as the act of securely maintaining them. Users must be confident that records that are deleted will never be recovered under a subpoena or other more devious method. Some legislation specifies a scope of time for which a company or agency is explicitly liable for their electronic records. For records that fall out of scope, there may be a desire to reduce liability by removing them forever. Further, destroying private information, such as personal health or financial statements, with per-version granularity may also be desirable.

While methods for securely deleting data from magnetic storage exist, all fail to meet the combined requirements put forth by legislators. Secure overwriting [4] is a method by which data blocks are overwritten many times with alternating patterns of 1s and 0s in order to degauss the magnetic media, making the data safe from magnetic force microscopy. This process is often lengthy and can exhibit poor performance for systems that have noncontiguous block allocation (a common side effect of file versioning). It is possible to securely delete data that have been strongly encrypted by simply "throwing away" the key used for encryption [2]; without a key, data may never be decrypted and read again. This method is very successful in a system that employs one key per file, but becomes unusable in a block-versioning file system, as data blocks may be shared between file versions. Lastly, there exist user-space tools for secure deletion. However, these tools are inappropriate for the regulatory environment. User-space tools leak information because they are unable to delete metadata managed by a file system. Further, they can't be interposed between file operations, may leak actual data on truncates, and are difficult to use synchronously.

2 Technology and Solution

We have developed a cryptographic technique that minimizes the use of secure overwriting while providing both authenticated encryption and efficient secure deletion of individual versions of a file in a block-

versioning file system. A keyed transform;

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

takes a data block (B_i), a key (k) and a nonce (N) and creates an output that can be partitioned into a secure data block (C_i), where $|B_i| = |C_i|$, and a short *stub* (s_i), whose length is a function of the scheme's security parameter (in practice, s_i might be 128 bits). When the key (k) remains private, the transform acts as a secure authenticated encryption algorithm [1]. Moreover, to securely delete an entire block, only the stub or a small subset of the bits in the secure data block need to be securely overwritten, *even if the adversary is later given the key (k).* This models the situation where a key is exposed (*e.g.* by a subpoena) after a block has been deleted and is similar to the notion of an All-or-nothing Transform [3].

Our design allocates all the stubs of a file contiguously in the file's metadata so that by securely overwriting the metadata, all data blocks associated with that file are securely deleted. When data blocks are shared between versions, the respective stubs are shared as well. In this way, when a previous version of a file is deleted, blocks that are shared with future versions are preserved. When deleting all version of a file it will often be more efficient to delete a small subset of bits from each data block than to delete the stubs that are replicated in each version. Deleting data blocks is the preferred technique for removing all versions of a file since versions often share much data in common.

Secure deletion and versioning does not add any complexity to key management. Our design supports the key management techniques used in any disk-encrypting file system [5]. The same encryption key may be used for multiple versions and only needs to be changed when access to a file changes. Our security model does not allow for the revocation of access to data in the past; users may continue to access a file version to which they had permissions in the past.

We are currently implementing this scheme in ext3cow (www.ext3cow.com) [6], an open-source, block-versioning file system designed to meet many of the requirements set forth by electronic record management legislation. Ext3cow is implemented entirely in the file system and, therefore, does not suffer any of the performance or security degradation problems that exist in user-space or other intermediate designs. It also provides a *time-shifting* interface that permits real-time access to data in the past through a continuous view of time. Some of the benefits of this design include: easy access to on-line backups; a way to detect and recover from system tampering; read-only point-in-time snapshots for data mining; and, file-oriented deletion recovery.

References

[1] M. Bellare and C. Namprempre. Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm. In T. Okamoto, editor, *Advances in Cryptology - Asiacrypt 2000 Proceedings, Lecture Notes in Computer Science Vol. 1976*. Springer-Verlag, 2000.

[2] D. Boneh and R. Lipton. A revocable backup system. In *Proceedings of the 6th USENIX Security Symposium*, pages 91–96, July 1996.

[3] V. Boyko. On the security properties of OAEP as an all-or-nothing transform. In *Proceedings of CRYPTO '99*. Springer-Verlag, 1999.

[4] P. Gutmann. Secure deletion of data from magnetic and solid-state memory. In *Proceedings of the 6th USENIX Security Symposium*, pages 77–90, July 1996.

[5] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable secure file sharing on untrusted storage. In *Proceedings of the USENIX Conference on File and Storage Technologies (FAST)*, pages 29–42, March 2003.

[6] Z. N. J. Peterson and R. Burns. Ext3cow: The design, implementation, and analysis of metadata for a time-shifting file system. Technical report, Hopkins Storage Systems Lab, Department of Computer Science, The Johns Hopkins University, 2003.