Tamper-Resistant Storage Techniques for Multimedia Systems*

Elizabeth Haubert†‡, Joseph Tucek†‡, Larry Brumbaugh‡, William Yurcik‡

† Department of Computer Science
‡ National Center for Supercomputing Applications (NCSA)
University of Illinois at Urbana-Champaign

ABSTRACT

Tamper-resistant storage techniques provide varying degrees of authenticity and integrity for data. This paper surveys five implemented tamper-resistant storage systems that use encryption, cryptographic hashes, digital signatures and error-correction primitives to provide varying levels of data protection. Five key evaluation points for such systems are: (1) authenticity guarantees, (2) integrity guarantees, (3) confidentiality guarantees, (4) performance overhead attributed to security, and (5) scalability concerns. Immutable storage techniques can enhance tamper-resistant techniques. Digital watermarking is not appropriate for tamper-resistance implemented in the storage system rather than at the application level.

Keywords : Tamper-Resistant, Tamper-Proof , Storage, Multimedia, Authenticity, Integrity, Security, Reliability

1. INTRODUCTION

Authenticity and integrity are two critical issues with stored data. Various techniques are used to ensure the authenticity and integrity of data including carefully labeling data at creation, protecting the storage infrastructure from physical attacks, and backing up data to geographically diverse locations. For many classes of data, including multimedia data, these measures are insufficient.

Consider, for example, a retail store surveillance scenario. There are three sequences made by a camera: sequence one shows the empty store; sequence two shows a customer browsing the shelves; sequence three shows the same customer tucking many small, expensive products into his pockets. The store presses charges, but the customer's lawyer describes in careful detail how sequence three could have been digitally created from sequence two. The jury believes this constitutes reasonable doubt, and acquits.

For the purposes of this paper, tamper-resistant storage is defined as storage systems which use practical solutions to demonstrate with high reliability that data has not changed improperly. This paper uses the term tamper-resistant rather than tamper-proof, as few systems can provide complete protection against unauthorized copying or modification. In security terms, this entails two separate issues: authenticity, or establishing the origin of the data in question, and integrity, determining that the data has not been modified improperly. For the security surveillance example, authenticity means establishing that the origin of sequence three was the appropriate security camera; integrity requires proof that the sequence has not been altered since the security camera recorded it. This paper will survey and evaluate the various techniques currently available for establishing proper authentication and integrity of multimedia data.

The simplest and most obvious approach is to adapt the error correcting methods used to detect and correct simple bit errors in ordinary hardware for the correlated changes. The same codes which provide a simple way of detecting and correcting bit errors in stored and distributed media can also provide a measure against deliberate bit changes.

* This work is supported by a grant from the Office of Naval Research (ONR) under the auspices of the Technology, Research, Education, and Commercialization Center (TRECC) "Storage Security" Project 2004-2005.
Author emails are *{epartrid, tucek, ljbrumb, byurcik}@ncsa.uiuc.edu*

Encrypting data prevents unauthorized content modification. Self-verifying or content-addressable data incorporates data signatures into addresses and indices.

The remainder of this paper is organized as follows. Section 2 surveys the current state of the art in tamper-resistant storage. Section 3 classifies tamper-resistant storage techniques according to design choices. Section 4 clarifies the differences between tamper-resistant and immutable approaches to data protection. Section 5 discusses the use of watermarking for application-level tamper resistance and its relationship to tamper-resistant storage. Section 6 addresses particular challenges of tamper-resistant storage. Section 7 relates those challenges to design parameters for multimedia storage systems. We conclude with a summary and conclusions in section 8.

2. SURVEY OF TAMPER-RESISTANT STORAGE SYSTEMS

Tamper-resistant media is applicable in a number of different scenarios. The following five storage systems present approaches to tamper-resistant storage colored by the unique problems each addresses. In [1], the PASIS system was developed to protect against correlated node failures in distributed environments. The Transparent Cryptographic File System (TCFS)[2] , introduced in [2], and NCryptFS[3] , introduced in [3], use tamper-resistant techniques to guard against malicious users on a single system. In [4], the OceanStore[4] system provides globally distributed but untrusted storage, and uses tamper-resistant techniques to prevent adversaries from presenting a corrupted block as valid. The Secure File System[5] uses tamper-resistant techniques to improve the scalability of secure content distribution; [5] describes the seminal work on this topic. The rest of this section describes each system in more detail.

2.1. PASIS: Byzantine-tolerant erasure-coded storage

PASIS is a prototype storage system implementing a novel decentralized consistency protocol to protect against Byzantine (correlated) failures of client or storage nodes. Storage nodes store data fragments and versioning information, while clients read and write data items.

To perform a write, a client determines the logical time and writes the time-stamped fragment to a quorum of storage nodes. Nodes keep all fragments, until freed by garbage collection. To perform a read, a client fetches the latest fragment from a threshold quorum, and determines if they represent a completed write. If not, additional and historical fragments are retrieve until a completed write is observed.

The improved security of this protocol comes from the use of erasure codes and checksums to support fragment verification. *N* data-fragments are generated during a write, one for each storage node, and any *m* of those fragments can be used to decode the original item or systematically generate the other *N-m* fragments. This capability, in combination with implemented checksums, can be used to reliably detect corrupted fragments. The write protocol ensures that only Byzantine storage nodes can return fragments that do not verify against the checksum.

Careful use of time-stamps, striping, and checksums means that this file system may make more data verification guarantees than most production systems Performance is improved relative to a naïve voting scheme, but still quite poor as qualitatively compared to a production system. Overheads inherent to supporting correlated errors including replicated writes and synchronization waits cause significant performance degradation to the system as a whole. Consequently, PASIS is most appropriate for seldom-changing, critical data.

2.2. TCFS: A transparent cryptographic file system

The Transparent Cryptographic File System (TCFS) is designed to support secure user access to remote sensitive files. While the system as a whole focuses primarily on data privacy, unauthorized data tampering is also addressed as a secondary concern.

TCFS runs in a local kernel space as a transparent layer between the virtual file system and storage file systems. User applications will see no changes as compared to direct interactions with the storage file systems, while protection/encryption utilities can interact with the relevant TCFS features using *mount* and *ioctl* system calls. All data

is encrypted or decrypted as it passes through the local TCFS, and therefore never accessed by the file system server or passed over the network in cleartext.

The key question with this approach is if encrypted data on disk is tampered with then brought back in to memory, what measures exist to detect whether data decrypts successfully. TCFS assumes that application-specific data-verification measures can be built into stored data. Because data on disk is securely encrypted, there is a high probability that a small change to raw data on disk (such as a bit-flip error) will result in a large change to the cleartext data. Therefore, there is also a high probability that these changes may be detected by an application-specific measure.

2.3. NCryptFS: A cryptographic file system

NCryptFS is a cryptographic file system designed to allow users to tailor the level of security to fit individual needs. To this end, NCryptFS supports a variety of ciphers, authentication methods, user name spaces, ad-hoc groups, challenge-response authentication, and transparent process suspension/resumption based on key validity. Tamper-detection is based on encryption techniques as with TCFS, which requires an assumption that detection will be application-specific.

The unique focus with NCrpytFS is the emphasis on user authentication and access controls as a means of preventing and controlling unauthorized changes. This is achieved primarily through use of an *attach*, a type of lightweight user-mode mount. The use of *attaches* allows owners to have personal encrypted directories. More importantly, each encryption key can easily correspond to a separate namespace with unique authorization criteria, further reducing the risk of leaked data. In concept, this is similar to allowing each user an entirely separate instance of a stackable FS, but with lower performance overheads. Because the attach does not hide data or introduce new data, it also poses a lower security risk for adding and removing users.

2.4. OceanStore: Self-verifying data for a global file system

More than any other system in this survey, OceanStore places a high premium on identifying the validity of distributed data. In particular, it utilizes erasure codes to fragment and reconstruct data. However, rather than relying on redundancy in the data fragments to detect and correct a corrupted fragment, which may take factorial time in the event of an error, OceanStore introduces a system for incorporating verification information of the complete data, fragment and fragment name into each block.

OceanStore uses a cryptographic hash of each fragment to create a unique name. The hashes of all the fragments in a block are repeatedly concatenated and re-hashed, until a single hash is created. The result is a globally-unique identifier (GUID) of the block. Verification proceeds by reconstructing this tree. Should the infrastructure return a complete data block, it would be possible to re-create this verification tree and GUID. Significantly, it is possible to verify an individual fragment by verifying a path from root to leaf. As a result, data supplemented with such hashes may be considered self-verifying[6].

The computational overhead to verify a block is considerable, and could introduce significant delay into retrieval systems implementing this naming scheme. In contrast to the SFS-RO database, the verification tree would deliberately not be cached, further increasing the performance degradation. In the worst case, this scheme cannot be guaranteed to correctly identify which block has been corrupted, only the presence of a corrupted block, therefore still requiring a potentially factorial number of fragments in order to verify and restore data. The security guarantees, however, are significantly stronger than a system relying purely on encryption and may be desirable for some applications.

2.5. SFS-RO: A secure, read-only file system

Fu et al. have created a database containing all blocks and inodes of a file system, indexed by a cryptographic hash of the contents of each entry rather than by the memory address of the contents. A client program that wants to query this database requests FSINFO, a data structure including the root handle of the file system and the root handle of the

database, and must verify the signature on the result. Clients may then traverse the directories of the database to locate the <*filename, handle*> pair, retrieve the data stored at handle, and verify that the data is correct by calculating the cryptographic hash of the data.

The strength of SFS-RO is that it distributes an authentic, time-stamped key, and allows clients to verify that the data they receive was stored using that key. This is an improvement in both time and storage space over digitally signing every block of data. Signing the FSINFO offline both protects the private key from online listeners, and allows the distribution server to be untrusted.

3. CLASSIFICATION OF TAMPER-RESISTANT STORAGE TECHNIQUES

Each of the previous solutions emphasizes different threats to data, and so uses tamper-resistant techniques differently. This section will propose three axes to classify the solutions used: the role of error correction methods, the role of encryption, and the role of self-verifying data. The use of error correction methods gauges the importance of critical and long-term data in a particular design. The use of encryption evaluates the relative weight given in the threat model to random and malicious errors. The use of self-verifying data outlines which entities in the storage environment each design must trust. This section describes each of these classification axes in more detail and applies each to available solutions.

3.1. Role of error correction methods

Error correction methods may be used for cases when it is safe to assume that an attacker will not be able to obtain cleartext data or will not be able to interpret the storage format. In this case, any changes to the data will not be fundamentally different from data stored or transmitted over a particularly faulty medium, such as wireless transmissions.

The error-correction put on actual disks is rather good. On average, modern magnetic disks experience one uncorrectable bit error per 10^{14} bits transferred, which is approximately 1 uncorrectable error per 40 terabytes of stored data. However, many media libraries can be quite large, and in light of new regulations, are expected to grow significantly larger. Tape and optical storage are worse. The authors[7] show that a failure in a RAID-5 array is stressful on the array, and that the probability of bit errors during reconstruction is rather high.

Embedded Block Coding with Optimized Truncation (EBCOT) is an arithmetic, wavelet-based encoding technique. By dividing each subband into independently coded blocks, the technique has some robustness against correlated errors 8. Reversible Variable Length Codes (RVLC) permit two-way decoding, and so are less susceptible to multi-bit errors, but are generally less efficient to implement. [8]

For most systems where non-repudiation is an issue, however, simple bit correction techniques will be insufficient. The techniques discussed above, where data is partitioned and each portion encoded independently allows a different degree of confidence. Erasure codes, for example, require the client to obtain only m of N fragments to reconstruct an image. By reconstructing the image multiple times from different fragment subsets, the client may establish some degree of reliability in the final result. This technique is often used for key management systems, as it can be made Byzantine tolerant. However as shown in the PASIS implementation, implementing full Byzantine error-tolerance in this scheme results in significantly degraded performance [1]. The OceanStore [6] paradigm allows implementers to determine the encoding level of a given file, forcing security and efficiency trade-offs on an application-by-application basis.

3.2. Role of encryption

Significant work has been done in the field of encryption to generate functions that may be used to establish authenticity or integrity of data. Because such algorithms are frequently interchangeable, this paper focuses on how such techniques may be used in a large-scale multimedia storage system.

Traditionally, the technique used to establish message origin and integrity is to enable some form of digital signatures. This usually requires the editor to generate a message digest of the data unit at publication time (the time at which data is determined "complete"). This signature is either incorporated into the data file, or distributed in such a way that a client may request and obtain the signature from other more trusted sources.

A primary problem with generated signatures is that users frequently misunderstand the importance of verifying signed content, forget, or misunderstand the relevant procedures [9]. Automatically generated data compounds this problem. In order to alleviate this problem and to simplify the performance overhead, several secure file systems have been created. Cryptographic file systems essentially enforce (among other things) restrictions on who may access and change data, allowing producers to make stronger claims about data authenticity and integrity. This approach has been used most visibly in the NCryptFS [3] and TCFS[2] file systems, but has been incorporated to a lesser degree in projects such as the Secure File System[5] and Self-Securing Storage[10].

3.3. Role of self-verifying data

While cryptographic file systems allow reliable creation of data, and ensure a level of integrity for data stored on disk, they do not generally address how signatures may be treated independently from data, and so made readily available. This is the problem addressed by self-verifying data. The most common variety of self-verifying data is content-addressable data, where a message digest is created from the data itself, and incorporated into naming conventions. Thus if the client is able to obtain the data, the client has already obtained the verification information. Furthermore, because data verification information is tied to the name, a client may verify that the data has not changed since the reference to that data has been created. This allows for protection against the case where the client obtains both the data and its authentication digest.

Content-addressable storage has been applied in the industry Centera system from EMC[11], and is currently being incorporated into several distributed indexing systems, including OceanStore [4], all projects based on the DHash system from MIT, including SFS, SFS-RO, and Chord among others[5].

3.4. Summary

From this, we may make two conclusions, First, error correction techniques that are specific to the codec work better; therefore, it is better to correct the bit stream directly. Second, encryption on top of error correction can result in small errors magnified by the encryption. In general, using a counter-mode or electronic codebook mode of encryption rather than a chaining mode encryption will be more reliable for long-term storage media. This may result in a small loss in potential security.

Similarly, tamper-resistance does not imply immutability. Consider the scenario discussed with the PASIS system. Message fragments are a fixed-size, encoded version of the original information. Unless servers are allowed to accumulate multiple fragments, there is no information leak, and each fragment does not necessarily need any additional protection against unauthorized writes. To overwrite any information, the publisher needs to only re-calculate and re-distribute a new set of fragments. Given a sufficient distribution of fragments and a paradigm to restrict the writer population, this can guarantee tamper-resistance with a high degree of probability. However, because data may change over time, there is no guarantee of immutability. Table 1 contains a summary description of each surveyed storage system, and the primary goal and unique characteristic of each.

Storage Project	University Affiliation	Primary Publication Year	Category	Identifying Characteristic(s)	Fundamental Goal
PASIS	Carnegie Mellon	2003	CryptFS	Uses erasure codes, so getting data only needs m of N data fragments, and cross-checksums to detect corrupted data fragments	Protect against correlated failures of clients and storage nodes
TCFS	University of Salerno	2001	CryptFS	Adds a layer between VFS and storage file systems to handle all encryption, so that clients need not trust either storage or the FS	Combat eavesdropping and data tampering
NCryptFS	SUNY Stony Brook	2003	CryptFS	Creates an infrastructure which allows each encryption key's data to have a unique namespace and authentication procedure	Combat data tampering by unauthorized users
OceanStore	UC Berkeley	2000	Self-verifying	GUID: create a "verification tree" hierarchically hashing the fragments of a block to verify 1. Each fragment and 2. Decoded block	Prevent adversaries from declaring that corrupted blocks are valid
SFS-RO	MIT and NYU	2002	Self-verifying	1.file names contain public keys 2.blocks/inodes named by SHA-1 hash of contents 3.groups of handles are hashed recursively	Clients should be able to verify integrity of content

Table 1: A sample summary of significant works in tamper-resistent storage.

4. TAMPER-RESISTANT VS. IMMUTABLE STORAGE

Immutable storage means that once written, data is never deleted or modified. Changes to data require versioning information be associated with the original, or creation of a new copy of the data. Of the surveyed systems, both OceanStore and SFS-RO fundamentally assume that data will not change over time, and PASIS is most appropriate for seldom-changing data. Given the computational overheads associated with tamper-resistant storage, an implemented system might rather store files in an immutable storage system. However, not all tamper-resistant techniques require static data. In particular, TCFS and NCryptFS are designed to allow authorized users to change data whenever necessary, while protecting against unauthorized changes. Figure 1 summarizes the use of static data in each of the surveyed system. Thus the two techniques are independent, but immutable storage systems may be used to enhance tamper-resistance. This section examines the possible relationship between tamper-resistant and immutable storage.

Consider a digital camera producing tamper-resistant output. When the camera takes a picture, it adds critical relevant information to the image file such as the time the picture is taken and a camera identifier, and then digitally signs the resulting file. Such a camera might be particularly relevant for the video surveillance scenario presented in the introduction. With this signature information enforced in the hardware generating the image, the camera couldn't/wouldn't lie about when a given picture was taken – and presumably neither could anyone referencing the

picture at a later date. Therefore, given an image it would be possible to determine the validity of timestamp indicating time of creation, and that no one had since modified the picture. Our hypothetical jury could no longer question whether the shoplifter had been "framed". If the camera had simply burned the image to a CD, the images it captured would have been immutable: a CD is read-only, and therefore immutable storage. However, anyone could make another CD with a modified image, and simply destroy or not present the original CD. Immutability does not directly imply tamper-resistance.

To summarize, immutable storage is stored system where no changes can be made to a particular copy of data. Additionally, it is very difficult, or even impossible, to destroy that copy through the file system or other storage interface. Tamper-resistant storage means that there exists some extra data that can be made publicly available (in the case of a hash) or cannot be faked (in the case of a cryptographic signature from a trusted party/device) that verifies the integrity of the data. Furthermore, good tamper-resistance implies support for tamper-detection, including an ability to potentially identify the nature of the tampering. Figure 1 summarizes the relationship between immutable storage and the systems surveyed in this paper.

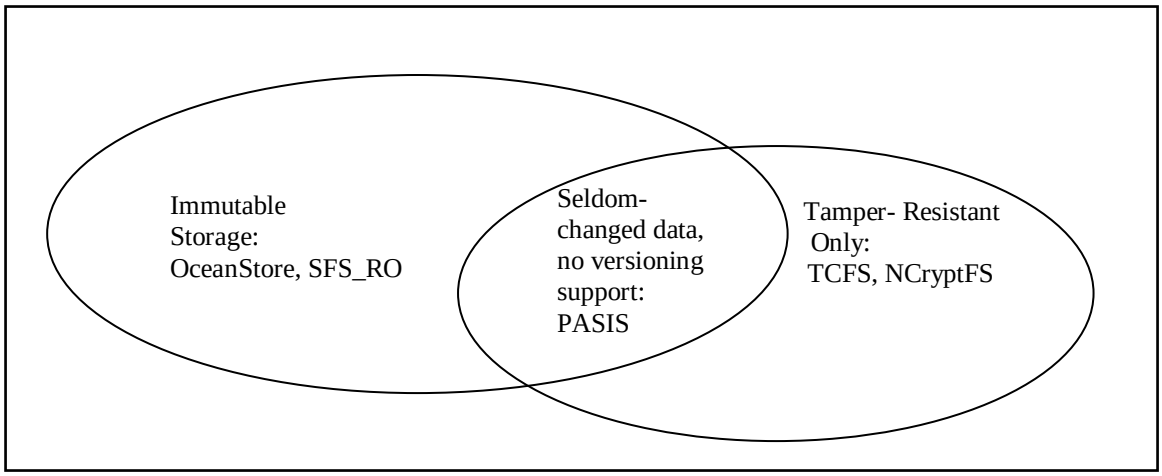

Figure 1: A Venn diagram showing the relationship between immutable and tamper-resistant storage.

5. WATERMARKING AND TAMPER-RESISTANCE

A notable exception from this paper as contrasted to other work on tamper-resistant multimedia is the discussion of watermarking techniques for tamper-resistance. Watermarking is a technique where some distinguishing piece of information is associated with the original data in a way that is difficult to extract or remove[12]. Techniques may be robust, where the primary goal of the tamper-resistant technique is to harden the watermark against removal. Techniques may also be described fragile, where the goal is to detect and localize the alteration[13]. There are three classes of attacks: subtractive, where an attacker attempts to remove the watermark, distortive, where an attacker deforms or obfuscates the watermark, and additive, where an attacker introduces a false or invalid watermark[14]. Watermarks are usually incorporated into data at publication time, but some techniques support a transfer of ownership from producer to consumer[15].

The key difference between watermarking techniques and tamper-resistant storage architectures is the level at which the algorithm is implemented which limits the implementation flexibility. An application has knowledge of the proper structure of data and can distinguish limits on the legal contents of that data. Tamper-resistant techniques implemented in the storage system have much more limited information. The file system can store a limited amount of metadata with a particular file, but this metadata is generally limited in scope: access control information, file type, age, and size are common examples. A block-level approach has even more limited information: only addresses and raw contents. While the implementation techniques and considerations for watermarked data are similar to the tamper-resistant techniques used in storage, the process for checking validity of a watermark will differ between

applications. Thus implementing a full watermarking scheme in the storage system would present significant complexity and processing overhead.

6. CHALLENGES

This section outlines a set of challenges that must be addressed in a storage security system. Authenticity, integrity, and confidentiality guarantees are difficult to quantify, but users selecting a security system must be analyze proposed solutions against these points. Other metrics such as scalability and performance overhead may be used to objectively compare implementations. Table 2 illustrates which of these metrics are specifically addressed by each surveyed systems.

6.1. Authenticity guarantees

For many applications, the most important challenge is establishing and maintaining the origin of the data. Tamper resistant techniques to support this include methods to associate signatures with data and establishing versioning for data fragments. Authentication support to identify data owners, such as with the NCryptFS system also makes some progress towards meeting this challenge.

6.2. Integrity guarantees

The primary focus with tamper-resistant storage techniques have been directed towards ensuring that data has not changed improperly, rather than establishing the origin of data. This effort makes sense – if data origin can be established, and it can be ensured that data has not changed since the time of creation, then both authenticity and integrity goals may be met. Without an integrity guarantee, however, statements about the initial origin of data are significantly weakened.

Ensuring that data does not change at all from the time of creation, in other words, that data is immutable, solves many integrity issues, but not all. If data is allowed to change even by creating new versions or copies of the data, verification techniques must first ensure that those changes are valid, and second, that clients accessing updated data are able to correctly read updated data. A correct read in this situation means both accessing a valid chunk of data and verifying the integrity of that data.

6.3. Confidentiality guarantees

A third threat in storage systems is the confidentiality of data. The storage system must not only make certain guarantees against unauthorized users corrupting data, it must also prevent unauthorized users from reading data. Many tamper-resistant techniques use encryption to protect against unauthorized writes; some also prevent unauthorized reads, or could with minor modifications.

6.4. Scalability

Modern storage systems are already well into the terabyte and petabyte ranges, and total volume increases yearly. Any storage security solution, including tamper-resistant storage, must be able to handle such a massive scale in addressing, number of users and number of processes. In short, all parts of the system must support automation and expansion, at least in a protected manner.

6.5. Performance overhead

One of the most significant challenges faced will be to meet the above challenges without imposing large performance overheads. The exact tradeoffs between performance overhead and security must be evaluated on an application-by-application basis, but the following generalization may be made. The more a system provides support for correlated failures, the more synchronization overhead such a system will incur. There is an additional interaction between performance and security. Security systems are implemented because the environment is assumed to be hostile; the system must continue to function and provide reasonable performance even after failures.

6.6. Summary

Because each of the surveyed systems had different design goals, each provides different guarantees for each metric. PASIS and OceanStore operate in a distributed, fault-prone environment; therefore, each system provides strong guarantees for authenticity, integrity and scalability. Because PASIS provides "perfect" survivability, it has very heavy performance overheads. Similarly OceanStore has a significant overhead to collect and reconstruct data. In contrast, TCFS and NCryptFS were intended to provide integrity and confidentiality without disturbing the user whenever possible. As a result, these systems have much lower performance overheads, but do not provide support for large-scale systems. Finally, SFS-RO restricts the system to seldom-changing data, but uses performance enhancements to boost scalability in a content-distribution network. Table 2 summarizes this analysis.

Evaluation Axis Addressed By System						
Tamper-Resistant System		Authenticity	Integrity	Confidentiality	Low Performance Overhead	Scalability
	PASIS	X	X			X
	TCFS		X	X	X	
	NCryptFS		X	X	X	
	OceanStore	X	X	X		X
	SFS-RO	X	X		X	X

Table 2: A summary of evaluation challenges addressed by each surveyed system.

7. SECURE STORAGE AND MULTIMEDIA SYSTEMS

Multimedia data represents some of the most sensitive and simultaneously, most protectable data. When selecting tamper-resistant techniques for a multimedia storage system, several axes must be considered. The most important of these are the frequency of data updates, and the file size of data under consideration.

The frequency of updates must be considered, because of the storage systems surveyed, those assuming immutable data are able to offer stronger techniques. The size of data in the storage system will affect the overheads associated with per-file and per-fragment security overheads, particularly for systems assuming self-verifying data. Most published multimedia data will fall under the categories medium and large file sizes. We anticipate never changed and seldom changed data to be more likely as well. Many multimedia files, although large, are rarely changed after creation or production time. Video conferencing source might be saved as a record of a particular conversation, but should not change after creation. Movie files might change occasionally as users adapt for a particular display, but in general will be viewed more often than modified. Table 3 gives examples of different multimedia files with different file size and access patterns.

Change Frequency / File Size	Never Changed	Rarely Changed	Frequently Changed
Small	Copyrighted sound clip Archived email	Image of course lecture slides	Graphical data summary (a daily stock plot)
Medium	Photographs	Telephone answering system sound clips	Satellite Imagery
Large	Surveillance Video	Home Movie	Unpublished Animation Video Sequence

Table 3: Examples of multimedia with varying file size and access patterns

For small files and medium files with static or rarely changed content, the OceanStore system offers the strongest reliability guarantees with reasonable performance. Small file size ensures that replicating files will be inexpensive, and the slow change rate mitigates the overhead from versioning. For small or medium frequently modified files, modifying the local file system to meet the security needs, as with NCryptFS or TCFS, provides more interactive access to data.

For large file systems, the considerations are quite different. Replication of very large files is quite expensive, and therefore not desirable. In addition, fragmentation of large files ensures that most fragments will be rarely changed, even for frequently changed files, because change to one fragment will affect only some fragments of data. This alters the range of security options and sources of performance penalty associated with many measures. Finally, a significant fraction of sensitive multimedia data, particularly from large files, is already associated with a considerable amount of metadata for indexing and retrieval. Incorporating self-verification measures into the processing of such data should not present a significant additional cost, particularly if security measures can be handled in parallel or with hardware support. Therefore, using a system such as the SFS-RO content distribution system is more desirable for storage systems with very large file sizes. Table 4 summarizes these recommendations.

Change Frequency / File Size	Never Changed	Rarely Changed	Frequently Changed
Small	OceanStore	OceanStore	NCryptFS
Medium	OceanStore	SFS-RO on mutable media	NCryptFS
Large	SFS-RO in immutable media	SFS-RO on mutable media	SFS-RO on mutable media

Table 4: Recommended tamper resistant technique with varying file size and rates of change.

8. CONCLUSIONS

This paper has described and classified the current state-of-the-art in tamper-resistant storage techniques, in particular as those techniques relate to multimedia systems. The key problems of designing a tamper-resistant storage system can be summarized as:

1. How to fragment the original data?
2. How to ensure that retrieved fragments reconstruct valid?
3. How to ensure that retrieved fragments construct data that was only modified by authorized users (where there may be potentially 0 authorized users)?

Existing systems use a combination of error-detection techniques, authentication techniques and encryption schemes to answer these questions. Improving the understanding of these solutions is crucial for evaluating the relative trade-offs between security and overhead for a particular class of applications. The relationship between tamper-resistant and immutable storage systems was also evaluated, with the conclusion that immutable storage may enhance tamper-resistant techniques. Finally, challenges for future study include metrics for authenticity, integrity and confidentiality guarantees, scalability, and overhead related to security.

ACKNOWLEDGEMENTS

We would like to first thank NCSA Program Manager E. J. Grabert for facilitating the funding of the StorageSS TRECC project. We would also like to thank the other members of the StorageSS group including Ragib Hasan, Greg Pluta, and Paul Stanton. Finally, we would like to give special thanks to UIUC Computer Science Professors Marianne Winslett and Yuanyaun Zhou.

REFERENCES

1. G. Goodson, J. Wylie, G. Ganger, and M. Reiter, “Efficient Byzantine-tolerant Erasure-coded Storage,” *Proceedings of the International Conference on Dependable Systems and Networks (DSN-2004)*. Florence, Italy, 2004. Supercedes Carnegie Mellon University Parallel Data Lab Technical Report CMU-PDL-03-104, December 2003

2. G.Cattaneo, L. Catuogno, A. Del Sorbo, and P. Persiano, "A Transparent Cryptographic File System for Unix," In *Proceedings of the USENIX Annual Technical Conference, Freenix Track.* Boston, MA, 2001.

3. C. Wright, M. Martino, and E. Zadok. "NcryptFS: A Secure and Convenient Cryptographic File System," In *Proceedings of the USENIX Conference General Track,* San Antonio, TX, June 2003.

4. J. Kubiatowitz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "Oceanstore: An Architecture for Global-Scale Persistent Storage," In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems*, Cambridge, MA, Nov 2000.

5. K. Fu, M. F. Kaashoek, and D. Mazieres, "Fast and Secure Distributed Read Only File System," In *Proceedings of the 4th USENIX Symposium on Operating Systems Design and Implementation*, San Diego, CA, Oct 2000.

6. H. Weatherspoon, C. Wells, and J. Kubiatowicz, "Naming and Integrity: Self-Verifying Data in Peer-to-Peer Systems," In *Proceedings of the International Workshop on Future Directions in Distributed Computing (FuDiCo 2002)*, June 2002.

7. P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leoung, and S.Sankar, "Row-Diagonal Parity for Double Disk Failure Correction," In *Proceedings of the 3rd USENIX Conference on File and Storage Technologies*, San Francisco, CA, 2004.

8. T.C.Yang, S. Kumar, Y. Bao, and C.C. Kuo, "Robust EBCOT Coding Technique for Wireless Image Transmission," *IEEE Wireless Communications and Networking Conference*, New Orleans, LA, September 1999

9. Whitten and J.D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," In *Proceedings of the Eight USENIX Security Symposium*, August 1999.

10. J. Strunk, G. Goodson, M. Sheinholtz, C. Soules, and G. Ganger, "Self-Securing Storage: Protecting Data in Compromised Systems," In *4th Symposium on Operating System Design and Implementation*, San Diego, CA Oct 2000

11. EMC Corporation. "Centera CAS System," *EMC Solution Brief.* 2004

12. S. Wolhusen, "On the Limitations of Digital Watermarks: A Cautionary Note," *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics*, July 1998, Vol. 2, pp. 489-495.

13. D. Kundur and D. Hatzinakos, "Towards a Telltale Watermarking Technique for Tamper-Proofing," *IEEE International Conf. on Image Processing*, Chicago, USA, Oct 1998.

14. F. Perez-Gonzales and J. Hernandez, "A Tutorial on Digital Watermarking," *Proc. of the 33rd IEEE Annual Carnahan Conference on Security Technology*, Oct 1999

15. L. Qiao and K. Nahrstedt, "Watermarking Schemes and Protocols for Protecting Rightful Ownership and Customer's Rights," Journal of Visual Communication and Image Representation, Vol. 9 No. 3, pp. 194-210, Academic Press, September, 1998.