Long-Term Threats to Secure Archives

Mark W. Storer Kevin Greenan Ethan L. Miller
Storage Systems Research Center
University of California, Santa Cruz
mstorer@cs.ucsc.edu kmgreen@cs.ucsc.edu elm@cs.ucsc.edu

ABSTRACT

Archival storage systems are designed for a write-once, read-maybe usage model which places an emphasis on the long-term preservation of their data contents. In contrast to traditional storage systems in which data lifetimes are measured in months or possibly years, data lifetimes in an archival system are measured in decades. Secure archival storage has the added goal of providing controlled access to its long-term contents. In contrast, public archival systems aim to ensure that their contents are available to anyone.

Since secure archival storage systems must store data over much longer periods of time, new threats emerge that affect the security landscape in many novel, subtle ways. These security threats endanger the secrecy, availability and integrity of the archival storage contents. Adequate understanding of these threats is essential to effectively devise new policies and mechanisms to guard against them. We discuss many of these threats in this new context to fill this gap, and show how existing systems meet (or fail to meet) these threats.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Access controls; H.3 [**Information Systems**]: Information Storage and Retrieval

General Terms

Design, Security

Keywords

secure storage, survivable storage, archival storage, secret splitting, encryption, cryptography, threat modeling

1. INTRODUCTION

The drive to archive information in digital form brings new challenges. Recent legislation, such as Sarbanes-Oxley, have placed strict demands on the preservation and retrieval properties of long-term storage systems. Archival storage systems must meet specific demands inherent to the usage model of write-once, read-maybe and long data lifetimes. In addition, there are novel security threats that are unique to long-term storage.

Traditionally, the adversaries to storage security have been roughly classified into two categories, passive and active. The active adversary is a malicious entity that actively attempts to break in and compromise an archive. In contrast, a passive adversary's actions are restricted to eavesdropping on communication channels. In the realm of long-term storage, the long data lifetimes of the storage contents suggest that time itself becomes an active adversary. It is constantly working against the system and threatening to compromise data. Throughout this discussion we will see that threats to traditional data storage systems take on a different meaning when viewed with the long view of the future that archival storage demands.

Archival storage systems attempt to address the issues of longevity that traditional storage systems have long ignored. As a result, some have come to refer to our current era as the digital dark ages. Danny Hillis noted [3], "For example, when we finally shut down the old PDP-10 at the MIT Artificial Intelligence Lab, there was no place to put files except onto mag tapes that are by now unreadable. So we lost the world's first text editor, the first vision and language programs, and the early correspondence of the founders of artificial intelligence." As our collective knowledge and artistic heritage becomes increasingly digital, the need to preserve data indefinitely becomes critical. For archival storage to succeed it is vital that we have an understanding of the security threats it faces.

2. SECURITY THREATS

When dealing with archival storage, the time frames for data lifetimes extend from the rather short-term scale of months or years to the longer-term scale of decades or even indefinite. The presence of long data lifetimes introduces some unique security threats. Some of these threats are variations on common concerns that take on new meaning in the area of archival storage while others are new threats that traditional storage system are largely unaffected by.

2.1 Long-Term Secrecy

Secrecy in long-term storage presents a complex challenge. Often, systems that provide file secrecy do so through the use of encryption [8, 12, 14]. The use of encryption within a storage system can be described in terms of the encrypted data's relevant lifetime. In long-lived encryption, such as for an encrypted file in an archival storage system, the data is persistent and the key must be preserved for an indefinite period of time. In contrast, an example of short-lived encryption would be a communication channel that is secured through the use of encryption and a key that is relevant only for the duration of the session. Long-lived encryption introduces a number of security threats.

StorageSS'06, October 30, 2006, Alexandria, Virginia, USA.

In an archival storage system, data can be very difficult to reproduce. The software, hardware and even users that produced the data may no longer be available. Encryption keys are a single point of failure and key loss is effectively equivalent to data deletion.

With the long-data lifetimes of archival storage, the use of encryption usually introduces the related problem of re-encryption at a future date. A number of scenarios could introduce the need to re-encrypt the contents of long-term file storage, including key rotations, compromised keys, compromised encryption algorithms and the need for access revocation. In some cases, re-encryption may only be needed for a few files, but in other cases, many petabytes of data might need re-encryption.

The re-encryption of large amounts of data must be done in a timely manner, especially if required for a key compromise or algorithm exploit. However, the time to apply the new encryption technique to a large amount of data will probably not be the limiting factor; rather, issues with maintaining keys and actually reading all of the old data quickly will be critical. Further, optimizations to speed the migration likely come with an associated management cost. For example, if a system chooses to save time by encrypting over the old algorithm, it must have a way of dealing with key histories and key distribution. In contrast, if the system chooses to decrypt the data before applying the new algorithm then it must have access to the users' encryption keys. Further, this access must must prevent a malicious user with system-level access from accessing data she is not authorized to view.

Even the traditional security threats and precautions associated with keyed encryption are made more difficult by the long data lifetimes found in archival storage. For example, cryptography is only computationally bound and it is very difficult to predict the future of cryptography and cryptanalysis. Technology such as quantum computing could usher in drastic changes to cryptography, and a long-term archival system which provides secrecy through encryption must be capable of handling those changes.

2.2 Locating Data

While the archival model of storage is write-once, read-maybe, users must still be able to find the data they stored should they desire to read it. Perhaps due to the fact that archival storage is not intended to be a user's primary data storage solution, it suffers from the potential problem of users forgetting where their data is stored. If users are unable to locate their data within an archival system, the availability aspect of security has been violated. For example, the Sarbanes-Oxley act specifies not only that data must be retained for a given period of time but also that it be retrievable in a timely manner.

One aspect of this problem lies in the location of data indices. A centralized index, maintained alongside the storage contents, has the benefit of relieving the user of remembering the details associated with storing their data. The centralized index is similar to a client that knows which bank their safety deposit box is in but not which box is theirs. This client might rely on a bank-maintained index which maps clients to their safety deposit box. One possible danger with this strategy is that an attacker that can compromise the centralized index is able to perform a much more targeted attack.

Another option, in contrast to the central index, places the onus of maintaining an index upon the client. In the safety deposit box analogy, the client would know which bank the safety deposit box is in as well as the location or ID of the safety deposit box itself. In this manner, even if a malicious person gained access to the vault, he would still have a brute-force problem of locating a specific box. With the personal index strategy, an attacker that compromises one user's index learns very little about the other users' data. This strategy suffers in that the client then has more long-term responsibility.

2.3 Authentication and User Accounts

In long-term secure storage, authentication must cope with a few scenarios not usually encountered in other storage arenas. If the storage system plans on providing file secrecy as part of security then it follows that the users must be able to authenticate themselves to the system as a first step in authorization. In other words, the users must show who they are before the system can determine what they are allowed to do.

A challenge unique to long-term secure storage is the problem that the user primarily attached to the data may no longer be available—for example, the user may be dead. Suppose a user wants to securely store their will in an archival storage system. The subsequent read may take place decades later by the owner's next of kin after the owner had passed away. A secure, archival storage system must thus be able to authenticate new users and establish their relationship to resources attached to existing users.

Additionally, if an archival storage system is able to authenticate a user to the system and determine her access permissions, it must be able to provide access to the entitled data in a meaningful fashion. For example, suppose a system utilizes encryption in order to provide file secrecy. If a user with rights to the data can view the file but has no access to the encryption key, she still has effectively no access. To continue the previous example, if a deceased user's next of kin proves his legal right to the data, the secrecy mechanism must function in the complete absence of the user that wrote the file.

2.4 Integrity Guarantees

Due to the rather short lifetime and limited reliability of traditional storage components, data begins to degrade as soon as it is placed on media. In short-term systems, this threat is relatively low because a great deal of frequently used data is accessed and updated on a regular basis. Archival storage assumes a write-once, read-maybe access pattern, thus the integrity of the data in the system must be actively checked at regular intervals.

Integrity checks often come in the form of *disk scrubbing* (also called *auditing*) procedures. A scrubbing procedure periodically scans sections of the data and uses strong hashes to detect corruption, recovering data that is damaged. A variety of problems can arise related to these scrubbing procedures. First, an overactive scrubbing procedure can contribute to media failures. In addition, an under-active scrubbing procedure may provide no utility over the absence of disk scrubbing. Xin, *et al.* analyzed these problems [23], showing that opportunistic scrubbing provides a good balance between overactive and under-active techniques. Unfortunately, opportunistic disk scrubbing policies require scrubbing requests to piggy-back on regular access requests. In a write-once, read-maybe system such policies may not be sufficient.

In the long term, cryptographic hashes may not be sufficient for integrity checking procedures. Given a reasonable amount of time, it is possible for an adversary to find collisions in the hash used for disk scrubbing. The adversary can update the original data with incorrect data, thus fooling the integrity checking procedure.

Compared to the internal data integrity problem of data within an archive becoming corrupted, distributed storage systems have an additional external integrity challenge. These systems must have a method of ensuring that the other archives in the distributed system are behaving properly. In such systems, one popular method of monitoring external integrity is through the use of a challenge-

response protocol. Of course, a secure system cannot rely on protocols that compare data in-the-clear.

Some have suggested the use of algebraic signatures as a way of performing integrity checks without exposing too much information [16]. Algebraic signatures have the property that the signatures of the parity equals the parity of the data signatures. For example, suppose a RAID stripe uses single parity as an erasure encoding, where the data symbols $d_1, d_2, ..., d_m$ compute the parity p. The XOR-sum of the algebraic signatures of the data, $sig(d_1) \oplus sig(d_2) \oplus ... \oplus sig(d_m)$, equals the signature of the parity element p. This scheme can also be extended to multiple erasure correcting codes, such as XOR-based Reed-Solomon.

As long as data is striped across the archives into reliability groups, a restricted distributed disk scrubbing algorithm can be used without disclosing too much information. If the disk scrubbing algorithm runs unrestricted, an adversary has the potential to extract information without proper authentication. For example, suppose that any party that can authenticate with an archive can submit an unbounded number of signature requests. If an adversary constructs enough sets of overlapping signature requests to an archive, then the sets form a system of linear equations that can be solved to reveal data.

2.5 Slow Attacks

When dealing with archival storage, the time frames for data lifetimes extend from the rather short-term scale of months and years to the longer-term scale of decades. This long lifetime gives attackers a much larger window within which they can attempt to compromise a security system. With archival storage an assailant might have several decades of time to conduct an attack.

One difficulty with slow attacks is intrusion detection. If the attack is methodical enough to make only the slightest of changes at any one time and each step was spaced far enough apart, it would be difficult to detect by traditional signature matching algorithms. This technique compares audit data and network activity to a database of known attacks. Thus, if the audit data contains a slight enough anomaly, it may not be enough to trigger a match with a known attack signature.

Another difficulty with detecting slow attacks is the problem of maintaining attack history. Security logging is especially important for systems that utilize secret-sharing algorithms such as PASIS [22], POTSHARDS [17] and others [18, 24]. Secret-sharing algorithms, while provably secure, rely on keeping a sufficient number of secret shares secure. This method of insuring file secrecy thus necessitates the maintenance of a history of compromises.

An example of the threat that slow attack presents is as follows. Suppose a file is protected using a 3/5 scheme in which the file is split into five pieces, three of which are required to rebuild the file. Now suppose that an attacker has compromised the system and obtained one of the shares. A decade later that same attacker obtains a second share. Due to the provably secure nature of the secret-sharing algorithms it can be shown that the attacker can gain no information about the data. However the system must deal with the fact that the attacker is making progress. With one more share the file would be revealed. Thus, the system must either immediately deal with any compromise or maintain a history of compromises in order to intelligently schedule corrective action.

2.6 Migration and Recovery

Due to the long data lifetimes of archival data, long-term storage systems will witness events that require data to be moved between archives. Reasons for this change include the inevitable failure or obsolesce of hardware, system updates and possibly even data movement as a security factor.

In archival systems as in more traditional storage, hardware failure is inevitable [23]. A long-term system must thus be immune to the failure of any given component. Additionally, the act of recovery and migration of effected data to new hardware should not compromise the availability aspect of security.

Even if hardware failure were not a factor, storage technology changes at a rapid pace. Storage systems today are quite different from those of decades past in terms of performance, capacity, media and interfaces. It is not a stretch of the imagination to assume that decades from now, today's technology will seem just as anachronistic as a punch card or drum storage device looks today. A storage system that is intended to keep data secure for several decades must thus be able to adapt to these changes by incorporating new technology and migrating data from outdated components. However, it must do so without allowing any party to actually recover data.

Another possibility for the role of data migration in a secure long-term storage system is to use migration to effect greater security. Moving data would create a moving target that could help to limit an adversary's ability to launch a targeted attack. This strategy might help to mitigate the effectiveness of the types of slow attacks discussed in section 2.5.

3. STORAGE SYSTEMS

In this section we examine a number of storage systems and how they deal with the security threats that have been outlined in the previous section. The sampling of systems presented below should not be considered an exhaustive list but rather a representation of the diversity within the storage field. Most of these systems are not specifically archival systems but they provide a useful perspective on the shortcomings of traditional storage systems in long-term roles. Examining the appropriateness of each system for the task of long-term secure storage illustrates the need to design a storage systems expressively for the purpose of archival storage. The capabilities of each system with respect to the discussion presented in Section 2 are summarized in Table 1.

3.1 FreeNet

FreeNet [4] is a peer to peer distribution system which in many ways is a stark contrast to archival storage. While archival storage is concerned with the long-term persistence of data, FreeNet is expressively created for distributing content and makes little effort to maintain its contents' persistence.

One of the primary goals of FreeNet is to allow anonymous distribution of data. It utilizes encryption over all stored files but this is primarily to provide a host with plausible deniability over their contents and not for secrecy.

FreeNet relies on local data-stores and dynamic routing tables to locate data. File migration is handled in a similar fashion. As requests travel back through the system the requested data is copied to each host along the request path. Thus hot data is quickly replicated while the space occupied by cold data is eventually reclaimed. This is clearly not conducive to long-term storage. Additionally, integrity in FreeNet is only assured through hashing of short strings which accompany the data. These are susceptible to dictionary attacks due to their short length.

3.2 OceanStore

Oceanstore [9] is a global, persistent distributed storage system, which stores data across sets of untrusted nodes. The system's objective is to provide an all-in-one, secure, highly- available, global

	Secrecy	Authorization	Integrity	Slow Attacks	Migration
FreeNet	encryption	none	hashing		access based
OceanStore	encryption	signatures	versioning		access based
FarSite	encryption	certificates	merkle trees		continuous relocation
PAST	encryption	smart-cards	immutable files		
Publius	encryption	password (delete)	retrieval based		
SNAD / Plutus	encryption	encryption	hashing		
GridSharing	secret sharing		replication		
PASIS	secret sharing		repair agents, auditing		
CleverSafe	secret sharing		high replication degree		
POTSHARDS	secret sharing	pluggable	algebraic signatures		
LOCKSS	none		vote based checking		site crawling
Glacier		node auth.	signatures		
Venti			retrieval		

Table 1: Capability overview of a variety of storage systems. Each of the systems discussed is listed along with a brief description of the mechanism used to provide the stated aspect of long-term security.

storage architecture. Oceanstore is designed around the assumption that all of the nodes are untrusted and failure is inevitable. Data protection is achieved through encryption and replication.

3.3 FarSite

Farsite [1] was designed to serve the same function as a centralized file server. It utilizes a distributed architecture that attempts to utilize the unused resources across a network of loosely coupled, insecure and unreliable machines.

Secrecy in Farsite is accomplished through the explicit use of encryption. At file creation a symmetric key is generated and encrypted with the public-keys of users authorized to access the file.

Locating data involves communicating with a directory group member. These nodes maintain the directory structure for a virtual hierarchy. In Farsite, files are presented in a hierarchical view but there can be multiple views of the data. Each view has its own root maintained by a set of directory root members. For long-term storage this makes locating data vulnerable to the failure of the directory group members. If there is an unlimited allowable number of roots however, it may be possible for each client to maintain its own view of the system.

Reliability and integrity come from replication and the use of Merkle trees respectively. File migration is also provided by the replication mechanism.

3.4 PAST

The PAST [6] system utilizes a overlay network to connect peers across the Internet with the aim of providing persistent storage with strong security. PAST achieves a high level of secrecy through the use of encryption on the client and authorization is achieved through judicious use of certificates. To facilitate this, PAST utilizes smart-cards to assist with the certificate operations. In addition to the usual concerns with long-lived encryption, the reliance on specialized hardware raises other concerns for long-term preservation. The authors do state that the role of smart-cards could be performed by trusted services.

Read requests are based on fileIds and routed using the Pastry [15] routing and location scheme. The smart-card is used in many aspects of this process. From naming integrity and quota management the smart-card is an essential token for system interaction. While convenient, for the long-term viability of this solution there must be a procedure for dealing with a lost smart-card.

Integrity and persistence in PAST is achieved through a two level approach. The first level attempts to prevent unwanted changes by making all data in the system immutable. The second level is through the randomized replication of data.

3.5 Publius

Publius [20] is another publishing system that gives the user a familiar URL based interface to system contents. While it is used to publish content it still utilizes encryption over the contents of the file as well secret splitting to manage the keys. Passwords are used to control the deletion and updating of contents. As discussed earlier, this reliance on encryption and passwords is a source of concern for long-term storage. The concern is somewhat mitigated as retrieval is limited to parsing a URL and the password is only used for deleting and changing contents. If the password is lost, the worst case scenario is that the data becomes effectively immutable.

In the Publius system, integrity is checked during retrieval. With a long-term archival system there are no guarantees for how frequently data will be accessed so this would be another area of concern. One possible solution would be to have an automated system which requests data for the sole purpose of insuring data integrity.

3.6 SNAD and Plutus

Secure Network-Attached Disk (SNAD) [12] was designed for secure storage. As such, it does not include any facilities for long-term archival usage beyond those present in many workstation-type file systems. Data location is straightforward; SNAD encrypts files individually, so any system that can store directory information in files, such as FFS [11] and ext3 [19] has location facilities.

Secrecy in SNAD is ensured using strong symmetric cryptography, with authorization accomplished using a public key infrastructure. Integrity in SNAD is ensured when the data is read through the use of hashes. Only a user that can decrypt the data can verify the hash, however, making it difficult to use SNAD for archival storage.

Since SNAD is intended for workstation file system use, it has few mechanisms aimed to ensure long-term data survival. For example, there is no mechanism to recover a lost private key. Similarly, there is no defense against “slow attacks;” however, it is unlikely that a slow attack would succeed because SNAD is only vulnerable to discovery of users’ private keys.

Other workstation-type secure file systems, such as Plutus [8], have similar properties with respect to long-term data survival. They ensure that data will never be revealed without the appropriate key, but are not appropriate for long-term data storage because they do

not address issues of key loss, algorithm compromise, and data longevity.

3.7 GridSharing and CleverSafe

The work of Subbiah and Bough [18] utilizes secret sharing to build a secure and fault tolerant data storage. While their system will function with a variety of secret sharing algorithms, the GridSharing system is designed for low-latency access and thus their testing shows that XOR secret sharing is the only viable algorithm. While the use of secret sharing provides secrecy without the need for encryption, the GridSharing system shows that its use may be limited in low-latency storage. The XOR algorithm in its basic form is an *m* of *m* scheme. This may be a problem in long-term storage since the data could not survive the loss of any of the secret shares.

Integrity in GridSharing is achieved through the use of replication. Using XOR based secret splitting would require very high levels of replication as each secret share is required for accurate reconstruction and thus the loss or corruption of any single share could compromise the entire file.

CleverSafe [5] has similar goals for data storage, protecting data using a custom-designed information dispersal algorithm to generate shares. As with GridSharing, CleverSafe provides little functionality to rebuild lost shares. Thus, CleverSafe cannot store data for long periods of time without extensive user intervention to ensure that sufficient data shares survive for long periods of time.

Both GridSharing and CleverSafe suffer from the rebuilding problem that, in order to recover lost shares, the system must first rebuild the data that contains the missing shares. While this makes system implementation easier, it reduces security and longevity by exposing the system to long-term decay.

3.8 PASIS

The PASIS [22] architecture is centered around providing a highly-available, fault-tolerant, secure storage system. Secrecy and redundancy is provided using a general threshold sharing scheme such as Shamir's secret sharing scheme. Since the shares are placed across storage nodes, an intruder would have to compromise several storage nodes in order to compromise data secrecy. PASIS relies on a directory service to translate file objects into the object shares and their respective location.

Integrity checking and correction is provided in PASIS through the use of a repair agent on each storage node. The status of each archive is actively monitored as part of the system's aggressive self-maintenance features. Additionally, as each request is considered suspect, PASIS also employs a system of versioning and request auditing. Audit logs allow the system to roll back any changes committed by a malicious user within a given window. This still raises the issue of how long to maintain the audit logs. In long-term archival storage an intruder may be able to space out the changes in a such a way that by the time the change is detected, the audit logs required to undo the damage are no longer available.

Unlike CleverSafe and GridSharing, PASIS can rebuild lost shares in a secure way [21]. However, this approach is computationally intensive and still may leak some information if sufficiently many servers are compromised; the number of servers that must be compromised is lower than the number of servers required to rebuild the data.

3.9 POTSHARDS

The POTSHARDS [17] system shifts data secrecy from encryption to authentication by using secret splitting to store secret shares across multiple authentication domains. Additionally, with proper authorization it is possible for the archives to collude and reconstruct any user's data or even all data stored in the system.

Before distributing shares across a set of archives, files are transformed by splitting the file into fragments using secret sharing algorithms. User- level redundancy is provided by a second level of secret-splitting optimized for redundancy. The result is a set of shards which are stored across a disjoint set of erasure encoded failure domains providing long-term, system-level redundancy. The shards in a lost archive can be rebuilt without revealing data because there is no way for any archive to discover which shards make up which objects.

Each client is responsible for storing and preserving an index over its own files. The user index is stored in the system so that it can be reconstructed given proper authorization should it become lost or corrupted. A system-level index holds a shard to archive mapping along with the information needed to reconstruct failed archives.

Each archive in the POTSHARDS system is responsible for maintaining the integrity of its own data. Archives check the integrity of their data by storing a cryptographic hash over sets of shards. In order to ensure each archive is actively performing integrity checks, an outside party can perform distributed integrity checks using algebraic signatures.

3.10 LOCKSS

LOCKSS [10] is a content distribution system designed for libraries that mimics the behavior of a web cache. As the purpose of LOCKSS is to ensure public access to data, it specifically does not include a secrecy aspect for securing the contents of files.

The data location aspects of LOCKSS are in keeping with its web-cache like behavior. Since the data appears to originate from its originally published source, the system does not present its own index to the user. Instead, it relies upon existing indexing systems and web content directories.

Integrity guarantees are provided through a voting mechanism. The system's contents are organized in archival units and a policy of voting uses cooperating systems to check the integrity of its AUs and repair any damage that may occur.

As with integrity checking, file replication is an active process within LOCKSS. In a three step process, nodes collect newly published data from journal websites, distribute the data by acting as a proxy cache for local requests and preserve their contents though the voting procedure with other LOCKSS systems.

3.11 Glacier

Glacier [7] is a decentralized storage system that relies on a large number of replicas to insure availability. It was designed based on for environment described by Bolosky [2] which found an expected node lifetime of 290 days.

There is no specific encryption mechanism for file secrecy in Glacier, so it avoids the problems discussed earlier. However is does utilize signed manifests for insuring integrity and thus still uses cryptographic primitives. In effect, while there is no specific secrecy policy, it still suffers from the problems introduced by long-term encryption.

Data location in Glacier is based on a circular naming structure and data location algorithm. This has the advantage of relieving the client from the onus of maintaining an index. The system was designed with large-scale correlated failure in mind and thus this technique may be more effective than the use of indices.

3.12 Venti

Venti [13] is a system specifically designed for archival storage. Key to the system is content-addressable storage. This provides built-in consistency checking information and enforces an immutable data policy that fits into the archival model. One concern with content-addressable storage is that users must have a way of remembering exactly what they are looking for. Put another way, content-addressable storage is great for filing but bad for finding. Some, such as Howard Besser, may argue that searching on metadata is a better fit for archival storage.

Integrity in Venti is performed primarily at retrieval time. Both the client and server are able to perform an integrity check as they are able to compute the fingerprint of the data and compare it to the request fingerprint.

Venti's properties make a it good candidate to act as the storage layer in a archival system, but as a stand-alone system it does not address the full needs of long-term storage. Functionality that deals with archive loss, and data migration might be implemented in higher layers.

4. CONCLUSION

Long-term archival storage systems introduce integrity, authentication and privacy threats that do not generally exist in non-archival storage systems. We have presented a general model for secure long-term storage systems and a set of threats that may lead to system compromise. The focus of this paper was not to solve the many problems that may arise in long-term system, but rather enumerate potential threats. We have presented new threats specific to the long-term storage problem and existing threats from the perspective of long-term storage.

In an effort to motivate the need for storage systems specifically tailored for the archival storage model, we have examined how a variety of existing systems deal with the threats to long-term data survivability. While many of these systems are still being developed and modified none of them specifically addressed all of the security concerns. Of greater concern is that there are threats which none of the system addressed. In particular the risk from slow compromises is an area that must be addressed in future archival storage systems. To deal with hardware changes, migration will also be an area that a long-term system must be able to accommodate. This demonstrates the need for storage systems that are specifically designed for the write-once, read-maybe usage model and long-data lifetimes found in archival storage.

An important aspect of documenting and discussing these threats to long term secure storage is that is has assisted us in the design of our own archival storage system. Our hope is that by listing the threats that such a system must contend with, future efforts to build secure archival storage systems will be more focused and complete.

Acknowledgments

We thank Kaladhar Voruganti and the other members of the Storage Systems Research Center (SSRC) for spirited discussions that helped focus the content of this paper. We also thank the sponsors of the SSRC, including Los Alamos National Lab, Livermore National Lab, Sandia National Lab, Hewlett-Packard Laboratories, IBM Research, Intel, Microsoft Research, Network Appliance, Rocksoft, Symantec, and Yahoo.

5. REFERENCES

[1] ADYA, A., BOLOSKY, W. J., CASTRO, M., CHAIKEN, R., CERMAK, G., DOUCEUR, J. R., HOWELL, J., LORCH, J. R., THEIMER, M., AND WATTENHOFER, R. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)* (Boston, MA, Dec. 2002), USENIX.

[2] BOLOSKY, W. J., DOUCEUR, J. R., ELY, D., AND THEIMER, M. Feasibility of a serverless distributed file system deployed on an existing set of desktop pcs. In *sigmetrics00* (2002), pp. 33 –43.

[3] BRAND, S. *The Clock of the Long Now*, new york, ny ed. Basic Books, 1999.

[4] CLARKE, I., SANDBERG, O., WILEY, B., AND HONG, T. W. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science 2009* (2001), 46+.

[5] CLEVERSAFE. Highly secure, highly reliable, open source storage solution. Available from http://www.cleversafe.org/, June 2006.

[6] DRUSCHEL, P., AND ROWSTRON, A. PAST: A large-scale, persistent peer-to-peer storage utility. In *Proceedings of the 8th IEEE Workshop on Hot Topics in Operating Systems (HotOS-VIII)* (2001), pp. 75–80.

[7] HAEBERLEN, A., MISLOVE, A., AND DRUSCHEL, P. Glacier: Highly durable, decentralized storage despite massive correlated failures. In *Proceedings of the 2nd Symposium on Networked Systems Design and Implementation (NSDI)* (Boston, MA, May 2005), USENIX.

[8] KALLAHALLA, M., RIEDEL, E., SWAMINATHAN, R., WANG, Q., AND FU, K. Plutus: scalable secure file sharing on untrusted storage. In *Proceedings of the Second USENIX Conference on File and Storage Technologies (FAST)* (San Francisco, CA, Mar. 2003), USENIX, pp. 29–42.

[9] KUBIATOWICZ, J., BINDEL, D., CHEN, Y., EATON, P., GEELS, D., GUMMADI, R., RHEA, S., WEATHERSPOON, H., WEIMER, W., WELLS, C., AND ZHAO, B. OceanStore: An architecture for global-scale persistent storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)* (Cambridge, MA, Nov. 2000), ACM.

[10] MANIATIS, P., ROUSSOPOULOS, M., GIULI, T. J., ROSENTHAL, D. S. H., AND BAKER, M. The LOCKSS peer-to-peer digital preservation system. *ACM Transactions on Computer Systems 23*, 1 (2005), 2–50.

[11] MCKUSICK, M. K., JOY, W. N., LEFFLER, S. J., AND FABRY, R. S. A fast file system for UNIX. *ACM Transactions on Computer Systems 2*, 3 (Aug. 1984), 181–197.

[12] MILLER, E. L., LONG, D. D. E., FREEMAN, W. E., AND REED, B. C. Strong security for network-attached storage. In *Proceedings of the 2002 Conference on File and Storage Technologies (FAST)* (Monterey, CA, Jan. 2002), pp. 1–13.

[13] QUINLAN, S., AND DORWARD, S. Venti: A new approach to archival storage. In *Proceedings of the 2002 Conference on File and Storage Technologies (FAST)* (Monterey, California, USA, 2002), USENIX, pp. 89–101.

[14] RIEDEL, E., KALLAHALLA, M., AND SWAMINATHAN, R. A framework for evaluating storage system security. In *Proceedings of the 2002 Conference on File and Storage Technologies (FAST)* (Monterey, CA, Jan. 2002).

[15] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, distributed object location and routing for large-scale

peer-to-peer systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms* (Heidelberg, Germany, Nov 2001), pp. 329–350.

[16] SCHWARZ, S. J., T., AND MILLER, E. L. Store, forget, and check: Using algebraic signatures to check remotely administered storage. In *Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS '06)* (Lisboa, Portugal, July 2006), IEEE.

[17] STORER, M., GREENAN, K., MILLER, E. L., AND MALTZAHN, C. POTSHARDS: Storing data for the long-term without encryption. In *Proceedings of the 3rd International IEEE Security in Storage Workshop* (Dec. 2005).

[18] SUBBIAH, A., AND BLOUGH, D. M. An approach for fault tolerant and secure data storage in collaborative work environements. In *Proceedings of the 2005 ACM Workshop on Storage Security and Survivability* (Fairfax, VA, Nov. 2005), pp. 84–93.

[19] TWEEDIE, S. EXT3, journaling file system, July 2000.

[20] WALDMAN, M., RUBIN, A. D., AND CRANOR, L. F. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proceedings of the 9th USENIX Security Symposium* (Aug 2000), pp. 59–72.

[21] WONG, T. M., WANG, C., AND WING, J. M. Verifiable secret redistribution for threshold sharing schemes. Tech. Rep. CMU-CS-02-114-R, Carnegie Mellon University, Oct. 2002.

[22] WYLIE, J. J., BIGRIGG, M. W., STRUNK, J. D., GANGER, G. R., KILIÇÇÖTE, H., AND KHOSLA, P. K. Survivable storage systems. *IEEE Computer* (Aug. 2000), 61–68.

[23] XIN, Q., SCHWARZ, T. J. E., AND MILLER, E. L. Disk infant mortality in large storage systems. In *Proceedings of the 13th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '05)* (Atlanta, GA, Sept. 2005), IEEE.

[24] ZANIN, G., MEI, A., AND MANCINI, L. V. A secure and efficient large scale distributed system for data sharing. In *Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS '06)* (Lisboa, Portugal, July 2006), IEEE.