# Trade-offs in Protecting Storage: A Meta-Data Comparison of Cryptographic, Backup/Versioning, Immutable/Tamper-Proof, and Redundant Storage Solutions

Joseph Tucek, Paul Stanton, Elizabeth Haubert, Ragib Hasan, Larry Brumbaugh, William Yurcik
*National Center for Supercomputing Applications (NCSA)*
*University of Illinois at Urbana-Champaign*
{*tucek,pstanton,epartrid,rhasan,ljbrumb,byurcik*}*@ncsa.uiuc.edu*

## Abstract

*Modern storage systems are responsible for increasing amounts of data and the value of the data itself is growing in importance. Several primary storage system solutions have emerged for the protection of data: (1) Secure Storage through Cryptography, (2) Backup and Versioning Systems, (3) Immutable and Tamper-Proof Storage, and (4) Redundant Storage. Using results from published studies, we compare these four solutions against different requirements highlighting trade-offs in performance, space, attack resistance, and cost. We also present a case study of applying these solutions based on design work at NCSA. Lastly, we conclude that while different storage protection solutions may be appropriate for different requirements, some general conclusions can be made about current state-of-the-art storage protection solutions as well as directions for future research.*

## 1. Introduction

The potential for loss, corruption, and leakage of data in a storage system is of great concern. Data is the critical element of an organizational processes. Without data to work with, the computational resources powering modern activity are useless [31]. Worse yet would be computations performed with corrupted data. Also unacceptable is the disclosure of sensitive information due to accidents or malicious attacks with new legal liabilities and the ultimate damage to organizational reputation.

The earliest work in securing data focused on the physical security of the data. Guards, locked doors, and cameras did much to deter attackers in the early days of computing. Later, remote access and inter-networking required security to be moved to the system level. Operating systems, passwords, firewalls, and intrusion detection systems attempt to effect security in the networks and the computers themselves. However, while both good physical security and good system security are necessary to protect data, alone they are inadequate. An inside attacker with access privileges will find few obstacles to data destruction, while a determined malicious outsider will be obstructed by little more. Given this evolution, the focus of protection must shift toward data-centric solutions, with the storage system itself at the center.

The primary goal of storage security is to insure the CIA properties: confidentiality, integrity, and availability. To do so, a variety of techniques are available. Cryptographic techniques can do much towards ensuring confidentiality and integrity. Backup and versioning enhance the availability and integrity of data. Tamper-proofing and immutability effect strong guarantees on integrity. Redundancy provides better availability. Varying in strengths and weaknesses, these techniques, applied correctly, can provide a secure, well performing storage system. Applied incorrectly they can easily lead to a costly, slow system with a false sense of security.

In this paper, we highlight the trade-offs in protecting storage systems using different solutions in order to determine which solutions are appropriate under different circumstances. The remainder of this paper is organized as follows: Section 2 provides brief overviews of the different storage protection solutions. Section 3 compares the different solutions against each other using data available in the literature. Section 4 is case study based on storage protection design work at NCSA. We end in Section 5 with general conclusions as well as directions for future research.

## 2. Summary of storage protection solutions

### 2.1. Cryptography

Storage solutions that employ cryptography are primarily designed to insure data confidentiality through the use of strong encryption. Thus, given proper control of the

Proceedings of the 22nd IEEE/13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST 2005)
0-7695-2318-8/05 $20.00 © 2005 **IEEE**

decryption keys, only authorized users can gain access to specific data, and a breach of the storage system reveals nothing to an attacker. Such security is clearly beneficial and often times necessary, but it comes with a two-fold cost manifested in performance penalties associated with data access and the additional overhead of key management. Cryptographic storage systems differ in the way that they handle these added costs. Since cryptographic operations are computationally expensive and time intensive, some systems offload the cost to client machines rather than concentrating the load on the storage system itself. Additionally, some designs rely on a centralized, trusted server to manage cryptographic keys while others require users to distribute and revoke their own keys.

It should be noted that cryptographic storage systems are not, by themselves, designed for availability. Without incorporating them into a larger storage solution, they are often susceptible to denial of service and deletion attacks. Any destruction of data or a successful compromise of the storage server can prevent an authorized user from accessing data, but the attacker will not be able to recover the data itself as it is encrypted on disk. PASIS [34], described in more detail later, provides a method to ensure availability, making it an exception to this rule.

Performance within a cryptographic storage system is largely dependent on the type of cryptography employed. Cryptographic operations can easily dominate access latency, and as a result the choice of cryptographic scheme is one of the most major determining factors in performance. Older ciphers, especially DES and 3-DES, can be prohibitively expensive. Newer ciphers, such as AES, are easier to implement efficiently, and can allow much better performance, especially if dedicated hardware is used to offload the cryptographic operations. In the absence of cryptographic operations, most systems perform comparably to the underlying file system which they employ. To demonstrate the significance of cryptographic operations, Table 1 offers a performance comparison of select systems.

The Transparent Cryptographic File System [3] and NCryptFS [33] are two examples of file system level systems that perform all cryptographic operations within the operating system prior to writing data to disk. Both are designed to be layered on top of another file system, such as NFS. As an application requests data it is first transferred from the storage server to the client machines. The client machine is then responsible for decryption before the data is transferred to the application. Similarly, data to be written is first encrypted by the client machine and then transferred to the storage server. The result is that the server never handles unencrypted data. The only time that data is in plaintext form is while it is in use by the application. However, these systems have cryptographic keys that are localized to individual machines, so any data sharing requires the file owner to manually distribute the appropriate keys. This can become unwieldy to manage in larger storage environments. Additionally, key revocation in TCFS requires that files be re-encrypted and new keys distributed, while NCryptFS relies on a cumbersome timeout threshold, forcing users to periodically re-authenticate themselves during operations.

Similar to the file system solutions, PLUTUS [15] and SiRiUS [11] implement a cryptographic storage system over a standard remote file system. The work of encryption, decryption, and signing is done on the hosts, allowing a scalability that is not possible with a centralized server. A system daemon intercepts all file system access calls and routes them through the appropriate encryption/decryption operations. These systems differ from those like TCFS in that they are not directly embedded within the operating system, offering a wider range of key management. File owners are still required to manually distribute keys to additional users, however, the keys that actually encrypt the data are embedded within a key management system located on the storage server. Figure 1 shows the arrangement of keys in PLUTUS. Separate key-system keys and file-signing keys are used to control access to the lower level data keys. This allows revoking a user to occur by only changing keys at the management level, preventing large amounts of re-encryption. Additionally, PLUTUS employs a key rotation scheme that automates the key management process and limits the amount of individual user management.

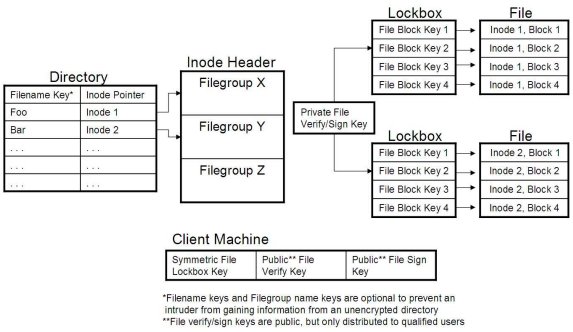


**Figure 1. PLUTUS key system**

As an alternative solution, PASIS [34] uses a threshold secret sharing mechanism to separate data among several servers based on the assumption of independent failures. If fewer servers than configurable thresholds fail, no data loss or leakage can result. Clients are able to corrupt data which they are authorized to write to, however, PASIS records which client initiates a particular operation, allowing the source of any corruption to be traced. Some large disadvantages to PASIS, however, include low read and write

**Table 1. Comparison of performance of different systems with different cryptographic methods. Note that the performance results were derived from unrelated studies of individual systems [3, 15] and cannot be used for a direct comparison.**

| Storage System | Sequential Read | Sequential Write |
|---|---|---|
| OpenAFS | 1.28 s | 1.57 s |
| PLUTUS w/o crypto | 1.39 s | 1.59 s |
| PLUTUS DES | 4.58 s | 4.27 s |
| PLUTUS 3-DES | 7.84 s | 7.92 s |
| NFS | 2.26 s | 1.39 s |
| TCFS w/o crypto | 2.46 s | 2.78 s |
| TCFS w/trivial crypto | 4.04 s | 9.05 s |
| TCFS 3-DES | 4.27 s | 15.92 s |

performance due to excess network traffic, and susceptibility to coordinated attacks and dependent server failures.

## 2.2. Tamper-proofing and immutability

Tamper-proofing and immutability are two techniques to improve the integrity of data. Tamper-proofing provides only integrity guarantees [14]. The purpose of tamper-proofing is to detect if modifications have occurred. Generally cryptographic hashing or signing is used to effect tamper-proofing. Immutability, on the other hand, provides both integrity guarantees and improvements in availability, by ensuring that once written, data cannot be overwritten. Enforcement of this property can be done in software, by the hardware, or be due to the physical nature of the media itself.

**2.2.1. Tamper-proofing.** It would seem that immutability subsumes tamper-proofing. Consider, however, a tamper-proof digital security camera. The camera adds a cryptographic signature to its images. If the images are later used in court, there can be no doubting their authenticity; the camera hardware signs only what it captures, and modifications become readily apparent. Suppose instead that the images were recorded to a CD-R. To modify the images one needs merely to save their work to another CD-R, and destroy the original. Without tamper-proofing, this would go undetected. At the same time, a tamper-proof security log does little good if deleted. Immutability and tamper-proofing fill different roles.

Figure 2 illustrates the relationship between select tamper-resistant, immutable, and cryptographic systems. Tamper-resistant systems use cryptography to provide integrity [14], while some cryptographic systems provide only confidentiality [33]. Some tamper-resistant techniques provide software immutability, but others, such as TCFS, do not. Hardware and media-enforced immutability often do not provide the metadata protection necessary for tamper-proofing.

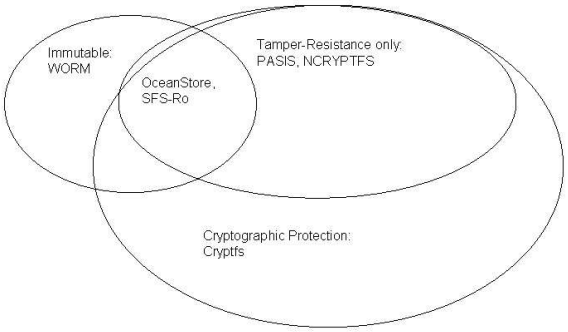


**Figure 2. Relationship between tamper-resistant and immutable techniques**

Cryptographic systems may provide tamper-proofing in addition to confidentiality [34, 3, 15, 11]. Content-addressable storage systems [9, 10] inherently include tamper-proofing. Two general techniques exist: cryptographic hashes and cryptographic signatures. In a hashing system, a summary of the data is made with a hash function. If a copy of this hash is retained, one can verify that the data remains unchanged. Cryptographic signatures generate additional information certifying that the signer "approves" of the data. This signature serves the same purpose as a hash. The signature can be checked by anyone, but cannot be forged. Therefore modified data can be detected.

Another axis along which to classify tamper-resistant systems is the data protected by the systems: metadata, such as a full file pathname, or the end-data itself. Protecting the metadata yields a somewhat different form of security: assurance that the path meets certain characteristics, and therefore that the data being retrieved is correct. This is particularly useful for systems using both tamper-resistant and immutable techniques. The verification overhead is reduced, as less data must be checked; particularly for large files. Furthermore, such systems can

offer strong guarantees that if the {path, address} pair is correct, then the data has not changed since retrieval. In contrast, in highly replicated or decentralized systems, including sufficient information to validate the complete path may become considerably more expensive than protecting the data alone. Thus in general, centralized systems and content-distribution networks incorporate metadata protection, such as in SFS, while decentralized systems such as PASIS rely on fragmentation and replication.

**2.2.2. Immutability.** Immutability refers to the property, which, once applied or given to an object, prohibits any subsequent changes to that object. In file systems, immutability refers to preventing any changes or modifications to the contents of the file [13].

Several different strategies have been used to create immutable storage. Immutable storage can be classified [30] into three major catagories:

- Physical WORM technology or P-WORM
- Embedded WORM technology or E-WORM
- Software WORM technology or S-WORM

P-WORM technologies include media that is by nature immutable and unchangeable. Both optical and magneto-optical media are used to implement P-WORM. Among these, optical disks are more popular. Examples include CD-R and DVD-R. Also, optical jukeboxes are used which combine several optical drives with multiple WORM disks. Capacities of optical WORM devices range from several hundreds of megabytes to several gigabytes in recent Ultra Density Optical (UDO) disks [21]. Magneto-optical disk storage operates on the principal of changes to the magnetic properties of certain media at particular temperatures. It has the advantage over optical WORM because it tolerates mechanical damages to the media far better than optical disks. Reading occurs at speeds of magnetic disks. Currently available magneto-optical disks have capacities up to 9 GB. The greatest advantage of P-WORM technology is its widespread adoption by different vendors. Similarly, magneto-optical disks are available from different vendors. The mass marketing, research and development into P- WORM have also lowered its price, compared to other WORM technologies. The major problem with optical P-WORM devices is that write operations are extremely slow when compared to their magnetic counterparts. Also, the capacity of each physical media is relatively small. Finally, management of a large number of media disks proves to be a big problem.

In embedded WORM or E-WORM, the device driver and firmware work together to implement an immutable storage system. Many large storage vendors have come up with proprietary storage solutions incorporating E-WORM. Both magnetic tapes and disks are used in these solutions. Magnetic tape WORM uses a combination of tape storage and firmware based protection. Common tape is not usually write- protected, so this requires specialized tape drives with an embedded write protection mechanism. Magnetic disk WORM uses the disk firmware to add protection mechanism to magnetic disks. Tape libraries are somewhat similar in nature to the optical juke boxes, combining multiple WORM tapes into one unified storage space. The EMC Centera [9] product and the Network Appliance NetStore/Snaplock [19] products are examples of embedded WORM technology based on hybrid firmware/software approach. E-WORM devices have larger storage space available per tape or magnetic disk. Also, faster write operations are performed than with optical media. The cost is comparatively small. However, magnetic media is prone to problems related to physical damage. WORM tapes are not recognized or usable with non-WORM devices. Also, magnetic tape WORMs allow only sequential read/writes but not any random access to the contents. Finally, intruders with physical access to magnetic WORM tape can damage the tape and destroy the contents

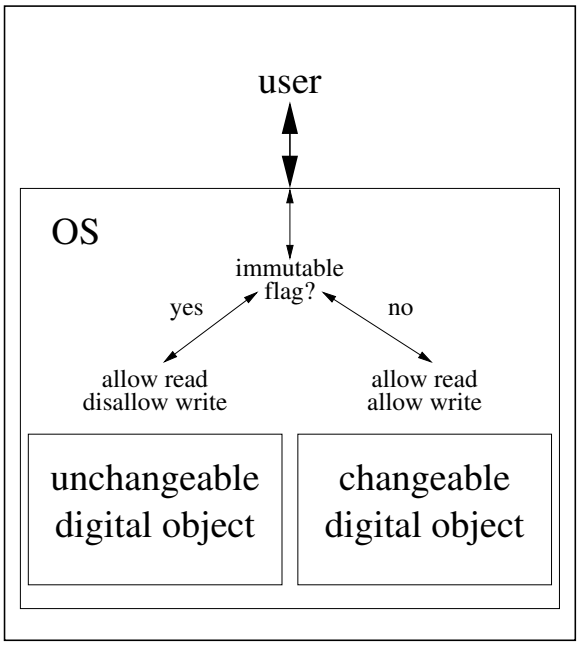


**Figure 3. S-WORM architecture**

The final category in immutable storage is software WORM or S-WORM (see Figure 3). Here, the operating system provides protection using mechanisms like capability-based schemes or modified file systems. For example, Unix provides a mechanism for indicating system files and directories immutable. System utilities like chattr and lsattr can be used to set the immutability flag and ren-

der a file impossible to modify even with root access. However, any malicious user gaining root level access can circumvent this protection. Linux Intrusion Detection System (LIDS) [16] provides immutability by enabling many of the all-powerful root privileges to be revoked. Therefore, an intruder who has gained root privileges cannot perform many activities used for exploiting the machine. LIDS also can prevent root kits that use Linux Kernel Modules (LKM), by allowing only loading of LKMs until the kernel is sealed by LIDS. In addition to these, LIDS can enforce access control list or ACLs on file system objects. A problem with using LIDS is that, all the above features are implemented using the Linux VFS. An intruder can still modify or delete a file by using the raw disk interface, circumventing the ACLs enforced by LIDS. To solve this, LIDS can be used to disable the capability `CAP_SYS_RAWIO`, preventing any raw access to storage devices. A major advantage of software-based WORM technology is that it is simple to implement and integrate with existing operating systems. On the other hand, all these solutions depend on the file system code to incorporate security. Anyone with root privileges could bypass the file system and access the physical media. Also, intruders who gains root access to a system have almost complete control over memory and can bypass any software based immutability mechanism.

Table 2 shows a comparison of the different WORM techniques available. The main issues and challenges in achieving immutable storage are capacity, data throughput, management overhead, cost and security. Magnetic tape is the cheapest, and has high capacities. However, it has slower data throughput, and management overhead. Optical jukeboxes have medium capacities, but are very costly. Magnetic disk based WORMs are fast, and have high capacities. But security is a problem with such devices. OS based techniques provide the best performance and capacities, but are prone to security problems. In view of this, a hybrid approach based on S-WORM and firmware based techniques seems to be the best solution for immutable storage.

### 2.3. Backup and versioning (time dimension)

Backup and versioning technology improve the availability and integrity of data in the time dimension. Backup provides a snapshot of a past system state. Modern backup solutions operate with constrained space and low system overhead, but require management configuration and leave large time spans of unprotected data. Versioning provides a more continuous data view but involves higher overheads and greater space requirements.

Modern backup systems are quite mature [5]. Since the work of the Amanda project [25], effort has focused on improving backup management. A modern backup system, such as Legato Networker or Tivoli, usually has a number of backup clients. These clients are individual servers or workstations. Software on the clients sends data to a central backup server. This server has a quantity of disk to accumulate backup data. Once sufficient data has been accumulated, it is written off to tape. Meta-data about the backups is kept on-disk for efficiency. A newer refinement of this process is to replace the tape with disk arrays. This disk-to-disk backup, made possible by the falling cost of disk drives, improves performance, especially for restores. However, tape is still the cheapest media for bulk backup, especially given it's low power costs. Massive Arrays of Idle Disks [6], or MAID, examines the potential of idling many of these disks to save power. Power usage can be reduced by 90% with only marginal reductions in performance compared to an active array. Recent work has focused on managing mobile hosts and prioritizing backup based on organization or economic value [20]. The prevalence of laptops in modern enterprise environments reduces the effectiveness (see Figure 4) of backup management software, and necessitates different strategies.

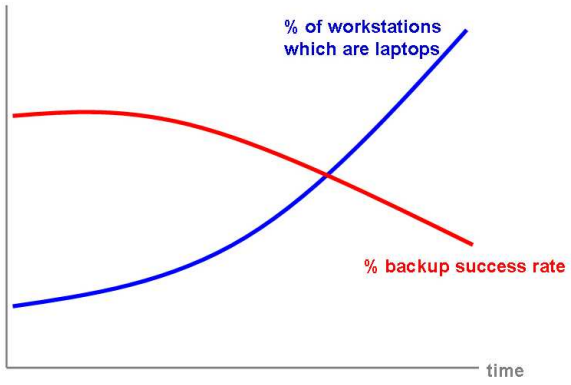


**Figure 4. Effectiveness of backup as a function of mobility**

Notable versioning file systems include S4 [28], the Repairable File Service (RFS) [35], and the Elephant file system [23]. S4 and RFS are both comprehensive versioning systems—all writes are logged up to a specified age. S4 [28] operates under the assumption that the host system is untrustworthy. Every write up to the age of the detection window is maintained. An administrative interface can be tunneled through the host system, using strong cryptography to defeat a potentially compromised host. Studies of daily write traffic show that total versioning is reasonable given usual disk capacities. RFS [35] extends the S4 concept to include logging to isolate damage caused by a malicious modification. Given the identification of a malicious process, RFS uses a dependency tree of operations

**Table 2. Comparison of immutable techniques**

| | CD/DVD-R | Optical Jukebox | Tape | Disk | Unix FS Tools | OS Techniques |
|---|---|---|---|---|---|---|
| Cost/MB | High | Very High | Very Low | Medium | Medium | Medium |
| Capacity | Low | Medium | Very High | High | High | High |
| Speed | Low | Low | Very Low | High | High | High |
| Security | High | High | Medium/Low | Medium | Very Low | Low |
| Management Overhead | High | Medium | High/Medium | Low | Low | Low |

to automatically identify a recovery state. The Elephant file system [23] keeps "landmark" versions, representative of the file system state at increasing intervals in time. Individual files may be assigned different versioning levels: no versioning, complete versioning, windowed versioning, or landmark versioning. While this is good for accidental modifications or deletions, landmark versioning is insufficient to protect against a malicious attacker.

### 2.4. Redundancy (space dimension)

Redundancy improves availability and integrity in the space dimension - data is replicated in different space so it can be recovered instantly. The best example of storage system redundancy is RAID [4]. Either through mirroring (RAID 1), parity (RAID 5), or error-correcting codes (RAID 3), RAID can recover from hardware failures.

Erasure codes [12] are another important storage system protection technique based on redundancy. Erasure codes allow a more flexible allotment between failure resistance and space usage than ordinary parity or mirroring. On the other hand, erasure codes are computationally expensive, imposing large penalties on write performance.

Double failures are not uncommon due to common environmental conditions or malicious attack. To survive double failures, a redundancy technique such as Row-Diagonal Parity [7] may be used. Data loss from double failures generally comes from the failure of individual blocks on other disks during the rebuild of a failed disk. These failed blocks which could ordinarily be scrubbed are exposed under the stressful conditions of an array rebuild. Row-diagonal parity keeps two units of parity information rather than one. This additional parity information is aligned such that any two disk failure can be recovered. Write performance is somewhat worse than RAID-5, but read performance is equivalent.

Some data is of course more important than other data. D-GRAID [26] recognizes this. D-GRAID uses the extra free space generally available in a disk array to store additional copies of more critical data, such as file-system metadata. Under failure, D-GRAID allows the majority of processes to complete even though data they require may be missing. Additionally, the extra copies of commonly read data can improve performance. This automatic balancing between replication levels is the key idea of the HP AutoRAID [29] system. AutoRAID puts newly written data in a RAID-1 style redundancy pattern. As space is filled, older, unaccessed data is moved to a more efficient RAID-5 style redundancy. This allows better storage efficiency without sacrificing performance. Additionally, the system is self-tuning, not requiring administrative overhead.

## 3. Meta-data comparison

Existing literature provides plenty of statistics as to the properties of these various techniques. However, it is scattered throughout the papers describing each technology. Additionally, there is little direct comparison across general techniques. Therefore we find it necessary to provide a meta-data comparison of all of the techniques.

### 3.1. Performance

Of primary concern is performance. Regardless of security, a system that does not meet necessary performance levels is of little use. Generally, performance is measured in terms of CPU overhead, latency, and network bandwidth.

For cryptography, the primary cost is CPU overhead. However, depending on how the cryptography is implemented, the magnitude of the costs can vary greatly. One important factor is the choice of cipher. 3DES can result in throughput as low as 10MB/sec, while Blowfish can approach 53 MB/sec [32]. Performance comparisons of different ciphers can be found in [8, 27, 18]. The new AES cipher, Rijndael, can be very high performing if properly implemented. On the other hand, public key cryptographic operations, critical in many digital signature algorithms, are among the slowest. Even so, modern processors can perform such operations in millisecond times. Additionally, most private key ciphers are well suited to implementation in hardware, with inexpensive FPGAs and ASICs supporting multi-megabyte per second performance [17]

Tamper-proofing imposes similar overheads as cryptography, but lesser in magnitude, as hashing and signature algorithms tend to be rather efficient. Since individual clients will tend to want to verify data on their own, the load can be

**Table 3. Comparison matrix**

| | Confidentiality | Integrity | Availability | Cost |
|---|---|---|---|---|
| Encryption | High | Medium | None | CPU |
| Secret Sharing | High | High | High | CPU, Latency |
| Tamper-Proofing | None | High | None | CPU |
| Immutability | None | High | High | Latency, Space |
| Backup | None | Medium | Medium | Bandwidth, Space |
| Versioning | None | Medium | High | Space |
| RAID | None | Low | Medium | Space |

easily spread out among many machines, without leaving a single bottleneck.

Immutability, backup, and redundancy impose relatively small performance overheads. Optical and tape media can limit the storage bandwidth in immutable storage systems–this can be overcome by using disk-based systems. Backup is not particularly performance sensitive, although bottlenecks may exist in tape and network bandwidth. Redundancy generally provides performance improvements, if the additional data can be used to provide benefits to read performance. Penalties are generally paid in write performance, but these penalties again are well understood [4].

### 3.2. Space

Cryptographic and tamper-proof systems incur similar sorts of overhead in the form of additional metadata. In cryptographic systems, this is key management data. Tamper-proofing requires the storage of digital signatures and hashes, which imposes an additional storage cost per file. If integrity is manages on a per-file basis, this overhead is negligible except when dealing with many small files. If integrity information is done at a finer-grained level, then the space overhead may become burdensome–however, this is generally a configurable trade-off.

Immutability and comprehensive versioning impose similar space overheads, in that space requirements continually grow. Depending on the application, the rate of growth can vary a lot, although for many systems it is reasonable [28]. Workstation disks typically experience no more than 200MB a day of write traffic, and so can easily be comprehensively versioned, perhaps to an immutable system.

Redundancy achieves its gains primarily through additional space requirements. However, these additional space requirements are generally very configurable. Standard RAID-5 systems require $(n+1)$ disks for $n$ disks worth of usable space. Row-diagonal parity and other "RAID-6" schemes use $(n+2)$ disks. Erasure codes allow an arbitrarily flexible choice of space overhead, although with added complexity. Finally, both D-GRAD [26] and HP AutoRAID [29] impose variable amounts of overhead, adjusting to disk usage patterns to provide better protection when the space is available. When disk space is not available, they approach RAID-5 efficiency.

Finally, backup also imposes space overheads, in the form of multiple versions of the data. Complicated rotation schemes, like the Towers Of Hanoi, can minimize this overhead at the expense of higher management complexity. Additionally, refinements to backup such as differential and incremental backups can further reduce the overhead required. Proper use of backup management software [25, 20] can reduce the management overhead of more complicated backup techniques.

### 3.3. Resilience to attack

It is difficult to measure the strength of security techniques. Although frameworks attempting to do so exist [22], they are mostly qualitative, and attempting to assign quantitative meaning to them seems arbitrary. Therefore, we attempt a qualitative comparison of the security properties of the different technologies. Table 3 provides a summary of our comparison. We base our model comparisons on [22].

Confidentiality is only increased by cryptographic techniques, such as encryption and secret sharing. However, these techniques alone cannot ensure data protection. Each relies on the assumption that the keys necessary to decrypt the data are kept confidential. However, management of these keys can be centralized, and is an easier problem than securing every system in an enterprise.

The integrity of a system is most protected by tamper-proof and encryption techniques. RAID only provides protection against hardware failures, but is easily thwarted by any purposeful attack. Backup and versioning provide some protection, but if an attacker makes a change to data under the guise of an authorized user, both backup and versioning will blindly backup and version the corrupted data. They do however allow the recovery of uncorrupted data given non-trivial task of detecting when the corruption occurred, and having made additional copies in the relevant time period. Encryption and tamper-proofing, while they make corruption difficult and detectable, can do little to

prevent deletion. Tamper-proofing, while it does little to prevent corruption, will alert users to the presence of corruption, in essence allowing only deletion attacks.

Secret sharing and especially immutability provide strong guarantees for integrity. Secret sharing requires multiple failures of authorization to occur. Finally, immutability prevents modifications from occurring at all, and therefore completely protects the integrity of data.

Availability, like integrity, can be influences by multiple techniques. Encryption can actually reduce availability, since loss of the encryption keys will cause the associated data to be lost. Backup and RAID improve availability moderately. A proper backup system will allow data to be recovered, however there will be a period of unavailability while the restore is in progress, and if the required backup does fall into the backup window, no recovery will be possible at all. RAID protects against hardware failures quite well, but software failures can result in the RAID system storing incorrect data. The incorrect data will be protected against hardware failures, but it will still be incorrect. Secret sharing, immutability, and versioning all increase availability. Again, secret sharing requires multiple points of failure for data to become unavailable. This can protect against both software and hardware attacks. Theoretically, an immutable system will never lose the data that is placed into it. Finally, versioning can act as a very complete backup system. Since versioning is generally on-line, the recovery period will be very short. Since versioning is often quite space efficient, a large window of changes can be kept. An attacker may make many changes to a system to perform a denial of service against the versioning system–S4 [28] addresses those attacks by rate limiting changes if malicious behavior is detected. Together, these qualities allow versioning to greatly increase availability.

It is important to realize that all of these techniques introduce additional complexity into the storage system. This complexity provides vulnerabilities–opportunities for administrators to make configuration mistakes, and opportunities for attackers to find additional flaws. Successfully dealing with the extra complexity is necessary to garner any additional protection benefits at all.

### 3.4. Cost

Given that these techniques are adding functionality to the storage, there will be additional complexity in the resultant system. This complexity imposes overhead, in terms of additional equipment, human resources, floorspace, power, etc. For example, to store 1 PB in a tape library for a year, one may consume $9,400 worth of power. Under the same conditions, a disk array may consume $91,500 worth of power [6]. The specific additional resources imposed can vary greatly from technique to technique and are also dependent on the specific implementation details of a particular environment.

Although it may seem that the additional hardware and software required to meet performance and space requirements dominates cost, the highest overhead is often increased storage management costs. Organizations spend significantly more money managing their storage infrastructures than they do acquiring and implementing technology. The root causes are the organizational complexities that evolved as storage demands increase and as technology grows more powerful. As a result, storage management responsibilities are typically dispersed throughout the enterprise, creating cost inefficiencies. One solution is consolidating storage management – storage-area networks, network-attached storage, direct-attached primary storage, secondary storage and backup – under the umbrella of a storage administrator or administration team. Storage administrators must manage performance (maximize throughput and minimize response time), manage storage infrastructure resilient to failure, and manage scalability that grows without downtime or redesign.

Among the several strategies to improve storage security, the highest management costs are associated with backup, especially as machines have become more mobile and difficult to track [20]. Even in the case where the storage is centrally located, a mistake in backup procedure imposes immediate costs in the form of lost data. Somewhat lesser management costs exist in the initial deployment of the other techniques, however, they can be amortized across the entire lifetime of the deployment. Once set up, most of these techniques will generally continue to function without additional management costs until such time as a failure occurs–after which the costs imposed are well worth the initial investment.

## 4. Case study: storage protection design for NCSA

As compute power has increased at NCSA, so has the corresponding mass storage system[1, 2]. While storage protection against reliability failures using backup and redundancy solutions has been the primary focus, a new threat has emerged from Internet attacks against High Performance Computing (HPC) Centers during Spring 2004. NCSA reported intrusions on its supercomputing clusters but these intrusions did not result in the loss of data.

In a worse case scenario, the scientific community using high performance computing has petabyte datasets that are at risk. These scientific datasets fall into one or more of these categories: (1) one of a kind, developed over decades of research; (2) not independent, the datasets fit together uniquely such that the loss of one dataset has widespread effects; and (3) benchmarks, if one dataset were to be cor-

rupted and used unknowingly by a scientific community it would make all subsequent research results suspect.

### 4.1. Storage environment

NCSA supports large scale scientific computing. As a result, the storage environment must deal with large data sets and high performance requirements. It is infeasible to deploy a storage system meeting both the performance and the capacity requirements at once. Therefore, the storage environment can be partitioned into two environments: high-performance storage intended to support short and medium term persistence requirements of scientific computing jobs, and a mass storage system intended to supply large quantities of long term storage. Figure 5 shows the basic layout of a portion of NCSA's storage environment.

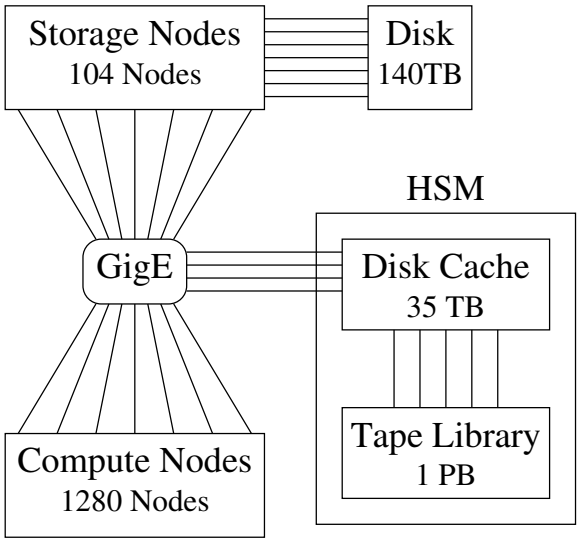


**Figure 5. Logical diagram of storage environment**

Each supercomputer at NCSA has its own performance storage system. Rather than describe the environment of every supercomputer, we take Tungsten, currently the largest cluster machine at NCSA, as an example system. Tungsten is a IA32 Linux cluster, with 1280 dual processor compute nodes. Supporting the compute nodes are 104 storage servers running the Lustre [24] cluster file system. In total, they provide 140TB of disk space, to be used as scratch space. Individually, each storage server is capable of providing 40 MB/sec of bandwidth to the disks, with a total available bandwidth of 11.1 GB/sec.

Typical workloads for the system is massive reads and writes. When a scientific computing job starts, every node associated with the job will attempt to read in its initial dataset simultaneously. After the job runs for a while, it will need to write out a partial result. At that time, every node associated with the job will simultaneously attempt to write out its state. Typically, jobs cannot continue until the IO operations are complete, so providing maximum bandwidth is the ultimate concern.

Being a cluster system, Tungsten is built from commodity components. Although the quality and reliability of each individual component is satisfactory, the large number of components ensures that some failures will occur. Some level of redundancy is built into the system, but as the system is intended for scratch use, fail stop and failsafe conditions aren't disastrous, although the associated downtime is still undesirable.

The mass storage system is a unified system which all of the machines at NCSA share. It provides archival storage for all of the NCSA's high-performance computing users, as well as providing storage for the datasets that are in active use. Given the large data requirements, it is necessary to use a hierarchical storage manager. The bulk of the storage is provided by about 1 petabyte of tape (LTO2), with 35 TB of disk acting as a cache. Given that this is permanent storage, each file is duplicated within the system, so that the failure of any one tape will not cause loss of data. Although tapes have been lost, the system has never lost data.

Access to the mass storage system is through an FTP interface. All transfers are whole file, and larger latencies (averaging in the seconds, peaking in the minutes) are allowed. High performance users are encouraged to prefetch files to avoid these latencies. The system is not intended to nor required to be high performance–it is not in general a bottleneck to site performance.

Somewhat unusual for a storage system is the mix of traffic. The read to write ratio is about .3; there are more writes than reads. This inverse scenario implies that much data is written to tape which is never read again. Additionally, lack of quotas encourages users to keep everything. As a result, storage use is growing at about 2 TB/week, and is accelerating.

### 4.2. Confidentiality

Cryptography is the most direct means of improving the confidentiality of a system. Given that the authentication and metadata control are effective, encryption of the data can provide protection against unauthorized reads. The confidentiality of data at NCSA is not generally critical, however for a corporate, military, or grid computing environment, confidentiality guarantees may be necessary.

To provide on-disk encryption for the high performance system would require providing extra processing power to perform the cryptographic operations. A single P4 2.1 GHz

can do AES in software at a rate of 50 MB/s [8]. Dedicated hardware can easily hit 200 MB/s or greater, even with inexpensive (50 dollar FPGA) parts. Given that the write bandwidth of individual storage nodes on Tungsten is 40 MB/s, it is feasible to encrypt all data on disk. Support for the additional metadata would have to be added to Lustre, but this too seems reasonable. On-wire encryption would impose an additional load on the compute nodes. However, since typically the IO causes the compute nodes to stall, it may be possible to support the additional computational load without reducing the performance of user jobs. Ideally, support for the cryptography would be implemented at the network interface to minimize use of the system bus. As Tungsten uses gigabit ethernet to access its storage nodes, either purchasing network interfaces with cryptographic acceleration or designing such cards as semi-custom hardware would be feasible. This would not be feasible if Tungsten used a proprietary interconnect, such as Myrinet, to communicate with the disk. In such a case, on-wire encryption would probably reduce performance somewhat. Either way, allowing an encrypted file system as an option, so that users can choose to encrypt files if they require the protection, or not if they would prefer the performance, seems the most reasonable choice.

Providing encryption for the mass storage system would be relatively cheap and reasonable. Although additional hardware would be required to provide for the cryptographic operations, it is less than what is needed for the high-performance systems. Additionally, the centralized nature of the storage system would simplify the management of cryptographic metadata.

### 4.3. Integrity

Integrity can be provided by tamper-proofing, immutability, and redundancy. Tamper-proofing imposes similar overheads as cryptography, and therefore has similar feasibility and implementation possibilities. For both the high performance and mass storage domains, it is feasible. For mass storage especially, tamper-proofing may be very desirable. Purposeful corruption of data sets, if undetected, will corrupt resulting computations, and can spread throughout many users' work. Most worrisome is that backup protection is useless unless when the corruption occurred can be determined, and even then it may be beyond the backup window.

Immutability can prevent this. For obvious reasons, it is infeasible to make the entire high performance storage immutable, but critical system files and files identified by users can be protected. The mass storage system, however, is essentially paying the cost already. No quota is imposed on users, so there is no incentive to users to delete anything, and there is no evidence to support that they do. Disallowing deletes would mostly be a formalism to the current behavior of the system. Additionally, the growth of new data in the system is exponential–old data accounts for only a small portion of storage used. Finally, given the low performance requirements and large economies of scale available, the cost of storage is relatively low. Include capacity in the tape library, the cost of the storage system is about one dollar per gigabyte.

### 4.4. Availability

Primarily, we can increase the availability of the system by improving the availability of the hardware. Some cluster systems at NCSA are already implemented with high availability hardware; while Tungsten is not heavily redundant as these systems, the ability of Lustre to support some failover can be leveraged to eliminate single points of failure. Except for the immediate moment-to-moment usability of the system, the high performance storage is too ephemeral and too volatile to bother with other techniques.

The mass storage already has high availability–it has never lost a file. The tape library supports sufficient redundancy to eliminate it as a worrisome point of failure. The disk cache in front of it is of more a concern, however, as it uses RAID-3, it is unlikely to fail.

The largest availability concerns relate to loss and corruption; that is, permanent loss of availability. Better integrity guarantees will, as a side effect, eliminate any such potential. Of secondary concern is occasional latency issues. While the average latency is acceptable, on the order of 30 seconds, peak latencies can exceed an hour. Although it is uncertain to us at this time the specific causes of these peak latencies, in the long term, a move away from tape and towards spinning disk may be feasible [6].

### 4.5. Remarks

The storage environment at NCSA is divided into two sub-environments, the high-performance systems and the mass storage system. Both are large and heavily used, but have unique characteristics. The high-performance systems are volatile, and cannot stand performance degradation. The mass storage system, on the other hand, is characterized simply by being large–over a petabyte.

Given these differences, they must be treated differently. The high-performance systems can be protected with cryptography if additional hardware is used. However, letting users disable the cryptography for their particular jobs is desirable–most users at NCSA are not concerned with confidentiality but solely with performance. Tamper-proofing does not provide many benefits to HPC, although it is technically feasible. Immutability is not possible, nor is secret sharing, as both techniques are ill-suited to write-intensive

applications. The same volatility which makes immutability impossible also makes backup and versioning difficult. Finally, the high-performance systems already incorporate some element of redundancy, providing sufficient availability for current purposes. Our recommendation is to leave the high-performance systems as they are, as they meet current needs. An exception would be any grid and utility computing–the increased confidentiality requirements of shared resources will require additional protection and necessitate changes.

The mass storage system can also be protected with cryptography–encrypt on-disk is possible without any special hardware, while encrypt on-wire would require hardware acceleration for the clients. Tamper-proofing would be both feasible and desirable, however it is subsumed by immutability. Currently the system is already paying the costs of immutability by the nature of its use–formalizing a policy of immutability would provide stronger guarantees of integrity for free. Finally, the availability of the mass storage system could be improved in terms of quality of performance–occasionally the latency of the system is high. To improve the availability of data in the mass storage system, we propose examining large disk arrays as an alternative to much or all of the tape.

## 5. Conclusions

In this paper we have summarized the current state of the art of the four primary storage protection solutions: (1) cryptography, (2) immutable and tamper-proof storage, (3) backup and versioning, and (4) redundancy. Depending on the environment under consideration, each of these solutions may be appropriate. For example, at NCSA, for HPC storage cryptographic techniques are the most applicable, although not required, and for mass storage immutability provides the most benefits.

We feel that interesting areas of future work in storage protection should focus on three areas:

1. *Usability.* These techniques add complexity to an already complex system, and management errors by administrators can nullify the potential benefits. Additionally, if additional burden is pressed on to users, they will attempt to circumvent the system.

2. *Unification with clusters.* Currently there are cluster file systems, and there are cryptographic and versioning file systems. It is necessary to actually implement some techniques from secure storage into a parallel file system, such as Lustre.

3. *Leveraging unique properties.* These techniques were designed with general purpose storage in mind. However, there are many properties of high performance and massive storage systems which differ from general systems. Taking advantage of the differences may allow for solutions which work much better for specific tasks.

We hope we have started a discussion of the trade-offs in making storage protection decisions, especially as it relates to large and high-performance installations. There is much work to be done.

## References


[1] M. L. Butler. Storage issues at NCSA: How to get file systems going wide and fast within and out of large scale linux cluster systems. In *NASA/IEEE Goddard Conference on Mass Storage Systems and Technologies*, 2002.

[2] M. L. Butler. 1 petabyte production storage environments and file systems (presentation). Supercomputing Conference (SC), 2004.

[3] G. Cattaneo, L. Catuogno, A. D. Sorbo, and P. Persiano. The design and implementation of a transparent cryptographic file system for unix. In *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, pages 199–212. USENIX Association, 2001.

[4] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. RAID: High-performance, reliable secondary storage. *ACM Computing Surveys*, 26(2):145–185, 1994.

[5] A. Chervenak, V. Vellanki, and Z. Kurmas. Protecting file systems: A survey of backup techniques, 1998.

[6] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–11. IEEE Computer Society Press, 2002.

[7] P. Corbett, B. English, A. Goel, T. Grcanac, S. Kleiman, J. Leong, and S. Sankar. Row-diagonal parity for double disk failure correction. In *Proceedings of the USENIX FAST '04 Conference on File and Storage Technologies*, pages 1–14, San Francisco, CA, March 2004. Network Appliance, Inc., USENIX Association.

[8] W. Dai. Speed comparison of popular crypto algorithms. http://www.eskimo.com/ weidai/benchmarks.html, Jan. 2004.

[9] EMC Corperation. EMC: Products: Platforms: Centera. http://www.emc.com/products/systems/centera.jsp?openfolder=platform, Oct. 2004.

[10] K. Fu, M. F. Kaashoek, and D. Mazières. Fast and secure distributed read-only file system. In *Proceedings of the 4th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2000)*, pages 181–196, San Diego, California, October 2000.

[11] E.-J. Goh, H. Shacham, N. Modadugu, and D. Boneh. SiRiUS: Securing Remote Untrusted Storage. In *Proceedings of the Tenth Network and Distributed System Security (NDSS) Symposium*, pages 131–145. Internet Society (ISOC), February 2003.

[12] G. Goodson, J. Wylie, G. Ganger, and M. Reiter. Efficient byzantine-tolerant erasure-coded storage, 2003.

[13] R. Hasan, J. Tucek, P. Stanton, L. Brumbaugh, and W. Yurcik. The techniques and challenges of immutable storage with applications in multimedia. In *Proceedings of IS&T/SPIE International Symposium*, Jan. 2005.

[14] E. Haubert, J. Tucek, L. Brumbaugh, and W. Yurcik. A survey of tamper-resistant storage techniques for multimedia systems. In *Proceedings of IS&T/SPIE International Symposium*, Jan. 2005.

[15] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus – scalable secure file sharing on untrusted storage. In *In Proceedings of the Second USENIX Conference on File andStorage Technologies*, Mar. 2003.

[16] LIDS Project. LIDS Project—secure linux system. http://www.lids.org, Oct. 2004.

[17] T.-F. Lin, C.-P. Su, C.-T. Huang, and C.-W. Wu. A high throughput low cost AES cipher chip. In *Proceedings of the 3rd IEEE Asia-Pacific Conference on ASICs (AP-ASIC)*, Aug. 2002.

[18] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[19] Network Appliance, Inc. http://www.netapp.com/products/filer/snaplock.html, Jan. 2005.

[20] G. A. Pluta, L. Brumbaugh, W. Yurcik, and J. Tucek. Who moved my data? A backup tracking system for dynamic workstation environments. In *Proceedings of the 18th Large Installation Systems Administration Conference (LISA XVIII)*, Nov. 2004.

[21] G. Quickel. Infinistore virtual disk–optical worm killer? Global Distribution White Paper, June 2002.

[22] E. Riedel, M. Kallahalla, and R. Swaminathan. A framework for evaluating storage system security. In *Proceedings of the Conference on File and Storage Technologies*, pages 15–30. USENIX Association, 2002.

[23] D. S. Santry, M. J. Feeley, N. C. Hutchinson, A. C. Veitch, R. W. Carton, and J. Ofir. Deciding when to forget in the elephant file system. In *Symposium on Operating Systems Principles*, pages 110–123, 1999.

[24] P. Schwan. Lustre: Building a file system for 1000-node clusters. In *Proceedings of the 2003 Linux Symposium*, July 2003.

[25] J. D. Silva and O. Guðmundsson. The Amanda network backup manager. *Proceedings of the Seventh Systems Administration Conference (LISA VII) (USENIX Association: Berkeley, CA)*, page 171, 1993.

[26] M. Sivathanu, V. Prabhakaran, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Improving storage system availability with D-GRAID. In *Proceedings of the USENIX FAST '04 Conference on File and Storage Technologies*, pages 15–30, San Francisco, CA, March 2004. University of Wisconsin, Madison, USENIX Association.

[27] A. Slagell. A simple, portable and expandable cryptographic application program interface. Master's thesis, University of Illinios at Urbana-Champaign, 2003.

[28] J. D. Strunk, G. R. Goodson, M. L. Scheinholtz, C. A. N. Soules, and G. R. Ganger. Self-Securing storage: Protecting data in compromised systems. In *Symposium on Operating Systems Design and Implementation*, pages 165–180, October 2000.

[29] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan. The HP AutoRAID hierarchical storage system. In H. Jin, T. Cortes, and R. Buyya, editors, *High Performance Mass Storage and Parallel I/O: Technologies and Applications*, pages 90–106. IEEE Computer Society Press and Wiley, New York, NY, 2001.

[30] R. Williams. P-WORM, E-WORM, S-WORM: Is a sausage a wienie? Imaging World, July 1996.

[31] R. Williams. Tao of backup wailing wall homepage. http://www.taubackup.com/wailing.cgi, Oct. 2004.

[32] C. P. Wright, J. Dave, and E. Zadok. Cryptographic File Systems Performance: What You Don't Know Can Hurt You. In *Proceedings of the 2003 IEEE Security In Storage Workshop (SISW 2003)*, pages 47–61, Washington, DC, Oct. 2003.

[33] C. P. Wright, M. Martino, and E. Zadok. NCryptfs: A secure and convenient cryptographic file system. In *Proceedings of the Annual USENIX Technical Conference*, pages 197–210, San Antonio, TX, June 2003.

[34] J. J. Wylie, M. W. Bigrigg, J. D. Strunk, G. R. Ganger, H. Kilite, and P. K. Khosla. Survivable information storage systems. *Computer*, 33(8):61–68, 2000.

[35] N. Zhu and T. Chiueh. Design, implementation, and evaluation of repairable file service. In *Proceedings of Dependable Systems and Networks (DSN03)*, 2003.