Network Security and Storage Security: Symmetries and Symmetry-Breaking

Donald Beaver*

11 October 2002

Abstract

It has been hypothesized that storage security and network security are essentially the same, at least insofar as mapping solutions from one domain in a straightforward manner to the other. We discuss similarities and differences that shed some doubt on the propriety of equating the two. While there are many ways to apply methods from one domain to another, there are fundamental differences between data at rest and data in motion. Storage is often an endpoint as well as a link, and it requires different treatment under such circumstances.

keywords: storage security, network security

1 Introduction

It seems at first glance that protecting data when it is shipped from one place to another should be similar to protecting data when it is stored. There is a body of cryptographic algorithms and security procedures that seems applicable to both cases, through straightforward data wrapping. Data that is secured against tampering should remain secure whether the parties at each end are separated in a spacelike (network) or a timelike (storage) fashion.

According to this view, the network or storage device is simply a path from one counterparty to another. The network or storage device is not itself a party to the communication. While simple and elegant, this point of view presupposes several relevant conditions, such as knowledge of the counterparties at each end.

These and other assumptions can oversimplify the storage security setting, often invalidating the reliance on direct application of network-oriented solutions. For example, the future counterparties might not exist at the time the data is stored. Moreover, the sender might not exist at the time the data is accessed. Yet certain cryptographic algorithms for network protection require interaction – key exchange being among the obvious examples.

Thus, viewing storage devices simply as a link between ultimate endpoints is conceptually useful but incomplete. There are many reasons to question whether storage is itself an endpoint.

*Seagate Research; 1251 Waterfront Place; Pittsburgh, PA 15222; donald.beaver@seagate.com

Proceedings of the First International IEEE Security in Storage Workshop (SISW'02)
0-7695-1888-5/03 $17.00 © 2003 **IEEE**

To arrive at a reasonable state of affairs for storage security, it will be necessary to consider the degree to which the extensive library of protection mechanisms developed for networking environments can be leveraged to the storage world. The observations made here are intended to motivate and facilitate the necessary exploration and debate.

2 Symmetries

There is a natural symmetry between the spaceshifting of data, or network communication, and the timeshifting of data, or storage. In each case, a source endpoint and a destination endpoint are separated by a path, and one can consider two principal events on a coarse scale: the generation of a message at the source, and its receipt at the destination. Drawing the analogy to relativity, the characteristic feature is whether those two events differ primarily in their spatial coordinates or primarily in their temporal coordinates.

Naturally, this is a very coarse view, but it suggests that communication and storage are somehow inherently symmetric. Threats and damage to data should somehow be proportional to the "space-time" distance between the two events: hazards accumulate against data at rest for a long time as well as against data moving for a short time over a long distance.

The field of data reliability indeed exhibits these symmetries. A Reed-Solomon code [RS60] applies equally well to transmitting data over wires as to encoding it on tape tracks for long-term storage, for example. The tracks are located near each other but there is an assumption that errors will be localized within tracks. One can carry the analogy further in both directions, such as Rabin's Information Dispersal Algorithm (IDA) [R89], which takes each component of a codeword and separates it spatially to a much larger extent, yet maintaining the intent to store over long time periods.

The important measure is the number of components that are ultimately corrupted, not where (or when) they have been. Intuitively, the number of errors is related to the degree of exposure to noise, regardless of whether that noise accumulated during a distant trip or a lengthy rest.

Confidentiality. While robustness and availability are cornerstones of security, confidentiality is equally important. Interestingly, the symmetry between time and space arises in similar techniques for protecting confidentiality. Blakley and Shamir's secret sharing techniques split a confidential value into several pieces, essentially using a random error correcting code, and disperse those values to different parties [B79, S79]. The randomization introduces an extra property: no sufficiently small subset of the parties can obtain any information whatsoever about the stored value.

In fact, one can extend the same procedures to protect communications across large distances by using separate paths [DD+93]. As long as fewer than some threshold number of paths are compromised, the message arrives not only intact (*i.e.* recoverable) but perfectly confidential as well.

When one turns to more straightforward cryptographic methods for protecting information, there is usually little distinction made between whether a message is scrambled for communication or storage. The same body of algorithms is loosely applied. Microsoft's NTFS file encryption mechanisms use the Data Encryption Standard (DES, or 3DES) in as facile a manner as any banking network's transmissions [M02].

Proceedings of the First International IEEE Security in Storage Workshop (SISW'02)
0-7695-1888-5/03 $17.00 © 2003 **IEEE**

Some Notation. A sender S and receiver R have natural analogies in the storage setting, whom we label the "storer" (for alliteration) and retriever. Simple communication consists of two events: ($\mathcal{S}$) sender S sends a message m, and ($\mathcal{R}$) receiver R receives a message (hopefully m). Likewise, simple storage consists of two events: ($\mathcal{S}$) storer S stores a document m, and ($\mathcal{R}$) receiver R retrieves a document (hopefully m). Let $\texttt{time}(\mathcal{E})$ denote the time at which event $\mathcal{E}$ occurred, and let $\texttt{loc}(\mathcal{E})$ denote where. (These are intended to be loose, informal notations; a send/store/receive/retrieve "event" could span a presumably short period of time.)

3 Symmetry Breaking

The loose correspondence between communication and storage starts to break down in the details. Let us consider a few primary categories:

- **Link or Endpoint.** Is storage merely a channel or is it a third party of its own right?
- **Access Control.** How are the desires of the "owner" of the data reflected, even when the owner may no longer be present?
- **Key Management.** Do the symmetries in data treatment extend to the treatment of the keys used for protection?
- **Data Behavior.** Are there empirically different characteristics of data that is transmitted as opposed to stored, and does this change how protection is implemented?
- **Architectural Implications.** Is there an advantage to separating storage and communication "layers," and what implications are there of applying optimizations from one domain to another?

In each of these categories there is a collection of reasons to consider storage as essentially different than network communication. Shifting data over long times starts to look different than sending it quickly over long distances. In timeshifting, there is less homogeneity in the environment from source to destination: involved parties change, algorithms change, keys change. Most significantly, however, timeshifting permits only one-way communication from sender to receiver, which eliminates a large portion of the techniques that can be applied in networking environments. Finally, even when physics might suggest deep equivalences between time and space or electricity and magnetism, there are significant engineering reasons to treat these concepts differently. The same applies to storage and networking.

4 Storage as Endpoint

A now-standard maxim in the field of data security is to secure the data from end-to-end rather than protecting each link in a path. Trusting intermediate links or nodes is generally a disadvantage, because it introduces weakest-link or weakest-node vulnerability. The natural inclination is to treat storage in the same way, relegating storage to the status of a link "in time," *i.e.* between the storage event and the retrieval event. If storage and networking

Proceedings of the First International IEEE Security in Storage Workshop (SISW'02)
0-7695-1888-5/03 $17.00 © 2003 **IEEE**

security are symmetric, then this practice is appealing; but we must consider whether the symmetries suffice.

Non-Interactive Communication

The most immediate breakdown in the apparent symmetry between storage and networking is in the one-way nature of the storage-based communication path. In a pair of communication events $(\mathcal{S}, \mathcal{R})$, there may well be interaction between S and R. But in a pair of storage events $(\mathcal{S}, \mathcal{R})$, the storer S cannot generally be assumed to be available.

There is an aspect of this distinction that, at a philosophical level, is a red herring. Indeed, a storer S might no longer exist when the retrieval event $\mathcal{R}$ occurs. Likewise, the retriever R might not exist when the storage event $\mathcal{S}$ occurs. But are these observations significant?

In the communication setting, there are characteristics that do not appear in the storage setting. Whereas a retriever may reasonably be the same as the storer, it is generally meaningless to have a receiver who is the same as a sender. In addition to triviality, it seems to suggest being in two places at once. Thus there are some lightweight, obvious differences between timeshifting and spaceshifting.

In and of themselves, these properties simply echo the definition of a communication-like data shift versus a storage-like data shift. The question, then, is not whether they are different (inherently, by definition, they are different) but whether the differences are pertinent. In other words, we wish to avoid basing our distinctions between storage and networking security circularly on the definitional aspects, but we may wish instead to explore whether there are significant consequences of those differences.

That said, there are at least two significant consequences of the inaccessibility of the storer S. First, in some way, the desires of the original storer S should somehow carry forward in time (and perhaps, adaptively) to the moment at which the retriever R requests access to the data. We will explore further implications of this in a moment.

Second, the particular implementations of data protection may be impossible to carry out if the storer S is not present – even though they work in the communications setting. For example, a Diffie-Hellman key exchange can be carried out between contemporaneous S and R but not between time-separated S and R.

Access Control

Even though the storer may no longer be present, his desires should carry forward to determine whether a given retriever can obtain the data. Those desires may be adaptive, in the sense that they may be dependent on events that occur in the meantime, or that they may be applied to a retriever who was unknown to S at the time of storage.

One can identify two main paradigms for applying access control. In *programmatic* access control, applications are equipped with the means to make their own access decisions using whatever criteria, policies, and specialized mechanisms that are deemed appropriate. A wide variety of application-specific policies can be supported, and applications may be expected to remain present to guide access to protected data.

Proceedings of the First International IEEE Security in Storage Workshop (SISW'02)
0-7695-1888-5/03 $17.00 © 2003 **IEEE**

In a *declarative* approach, common access methods are configured and the principals and data are described in a common representation. Generally, an access protection system such as an operating system stands between the principal who requests access and the data or operation requested.

To implement programmatic access control in the storage setting, an agent of some sort needs to remain present. To some degree this begs the question: it essentially requires S to stay around and gatekeep. Naturally, only those access-decision portions of S might need to be captured.

To implement declarative control, the medium must graduate from a dumb link to a more intelligent access control mechanism. File servers and network-attached storage are good examples. A common, simple language for access control and a recognized set of principals are basic components.

4.3 Proxy for the Sender

Assembling these observations, we come to the conclusion that a storage security solution generally requires some sort of proxy for the sender. There are needs that arise from the adaptive nature and desires of the storer as well as from the unavoidable requirements of specific implementations.

Access control seems to demand either a boiled-down adaptive version of the sender (for flexible programmatic control) or a common protection mechanism (for manageable declarative control).

Cryptographic mechanisms may demand interactive protocols between S and R which are by nature impossible to apply in the storage setting.

Having a proxy for the sender, of course, is essentially like promoting the storage "channel" to a third party. In order to maintain the sender's desires over time even when the sender is absent, and in order to maintain the ability to engage in further cryptographic interactions, the storage channel must arguably be an endpoint – security maxims notwithstanding.

5 Key Management and Trust Management

We have mentioned that certain cryptographic protocols may require interaction which is impossible across a temporal gap (*i.e.* between an arbitrary storer and retriever). There are other implementation-related security aspects which differentiate a temporal gap (where $\texttt{time}(\mathcal{R}) - \texttt{time}(\mathcal{S})$ is large) from a spatial gap (where $\texttt{time}(\mathcal{R}) - \texttt{time}(\mathcal{S})$ is small, while $\texttt{loc}(\mathcal{R}) - \texttt{loc}(\mathcal{S})$ may be large). These differences strengthen the separation between storage security and network security.

While it is important to employ tested cryptographic algorithms, managing keys and trust is an essential aspect of security. Efforts are needed to ensure that key information is not unnecessarily exposed. To ensure proper access control, it is also necessary to maintain reliable mappings from identities to keys. As time goes by, secret information can leak and reliable data can become stale. As discussed below, however, time has different impact on privacy versus reliability: slowly dissipating information works against privacy but can work

Proceedings of the First International IEEE Security in Storage Workshop (SISW'02)
0-7695-1888-5/03 $17.00 © 2003 **IEEE**

for consensus on reliable information. This means that certain hierarchical methods for key and trust management that work in networking may not work as naturally for storage.

Security Properties

Short-lived, session keys provide a way to protect against long-term exposure. Sender S and receiver R establish a temporary key to perform the direct encryption of messages. Typically, long-lived keys are used to establish the session keys. This process reduces the amount of clear and cipher text associated with any given key, and it allows optimizations based on empirical efficiency differences between symmetric and public-key cryptosystems.

Because the session keys are short-lived, they can be disposed after the session is complete, removing one vulnerability that an attacker might capitalize on. Where public-key cryptosystems are used, receivers must maintain the secrecy of their decryption keys, but this is a much smaller amount of information to manage than a collection of symmetric keys for per-association or per-message encryptions.

Because confidentiality is bootstrapped from personally-held asymmetric private keys, the authenticity and integrity of the public keys are more prominent issues. The public keys are, of course, public – which vastly simplifies their management, since one need not worry about their slow or eventual leakage. Without concerns over privacy, the task of ensuring that all participants share common information reliably (*e.g.* a mapping from IDs to public keys) is achievable in a more robust fashion, for example by spreading it to many repositories. It is inherently difficult to prevent repositories from exposing information; it is inherently easier to prevent repositories from forging false signatures. (Staleness, especially in the form of revoked directory information, is a more thorny issue; but it, too, is addressable through *greater* information sharing rather than *restricting* information sharing.)

In short, authenticity and integrity are an important currency in public-key architectures, where the natural pressure toward information "leakage" is desirable (to propagate authentic, fresh information) rather than a liability (compromise of multiple, confidential keys).

As in the case of interactive cryptosystems discussed earlier, however, there is an obstacle in the storage setting: the receiving/retrieving party may be unknown at the time of storage. This is not merely an aspect of a particular, interactive cryptographic operation but a fundamental obstacle. There is no simple way to establish a "session" at the moment of storage with an unknown or non-existent future counterparty (without, of course, having a long-lived proxy agent as discussed in §4.3).

Re-encryption

Let us set aside the management issue of how to allow and enable an initially unspecified future retriever to learn the decryption key. A central problem is that if there is a single key k for the association (S, R, m), the key remains vulnerable over a long period.

Re-encryption of stored data is, in part, the conceptual equivalent of a session between networked counterparties. There are several motivations.

If storage is indeed an intermediate endpoint, then the responsibility for protecting confidentiality is shifted to the storage system. By re-encrypting the data, there is less vulner-

Proceedings of the First International IEEE Security in Storage Workshop (SISW'02)
0-7695-1888-5/03 $17.00 © 2003 **IEEE**

ability to exposure through compromise by the storer S; the storer S can dispose of secret key information and avoid further sensitive management.

A second motivation includes the slow radiation of secret information. An intricate system is akin to atomic decay: a variety of vulnerabilities can conspire to expose sensitive information at unpredictable moments. (What is important about this imprecise characterization is that it reflects unpredictable but gradual leakage through imperfect implementation, as opposed to direct cryptanalytic attack, for instance.) Re-encryption of the same data items over time helps to protect against such gradual loss [HJ+95, DK+02].

Finally, cryptographic algorithms and modes evolve over time. Specific examples such as DES may eventually come to be regarded as weak, as cryptanalytic knowledge or computing power expands. Network communications usually take less than 30 years to complete (AOL notwithstanding), but confidential legal documents may need to be stored for that long. Storage systems may therefore need to accommodate re-encryption due to algorithm changes, not just secret-key vulnerability.

5.3 Heterogeneity of Timeshifting, Homogeneity of Spaceshifting

Taken together, there is a variety of reasons why shifting data through time is a more heterogeneous process than communicating it across space. In the gap between the storage event $\mathcal{S}$ and the retrieval event $\mathcal{R}$, a large number of contingencies can evolve.

In addition to the greater need to maintain secrecy (because the retrieving endpoint is unknown) as opposed to merely authenticity/integrity, and in addition to the evolution of algorithms as discussed above, there are also trust issues. The access control decision may be based on when a retriever R requests access, *e.g.* when the retriever is subscribing to a data service. The link between a retriever's identity and the cryptographic information (such as signing or encryption keys) employed to grant access can change over time, such as when a digital certificate is revoked or a certification authority is compromised.

The heterogeneity of the path across time seems to require more direct and persistent attention than a short-lived connection across space. Issues of key management, algorithm evolution, and trust management over time suggests a role for storage systems as an endpoint rather than a link that is "locked out" from the real action.

6 Data Characteristics

In a network setting, messages are treated as individual units, encrypted and authenticated as a whole. Even when a message is repeated, it is typically encrypted again along with a nonce to ensure freshness and avoid replay.

Files, on the other hand, are manipulated in a more local and piecewise fashion. Reads and writes operate on portions of the file, and not necessarily in sequential order. Unlike the issuing of a new message, each file modification does not result in a complete rewriting of the file. It would be highly impractical to consider each successive version of a file as somehow equivalent to a network message.

A natural intermediate ground is to treat storage blocks as the analogous unit to a network message. In persistent storage systems (as opposed to volatile memory), blocks

Proceedings of the First International IEEE Security in Storage Workshop (SISW'02)
0-7695-1888-5/03 $17.00 © 2003 **IEEE**

are read or written as an atomic unit. File storage protocols such as ATA or SCSI do not issue byte level changes. In this regard, encrypting and ensuring integrity might be able to leverage network techniques.

Nevertheless, the integrity of a file as a whole must be ensured, requiring some way to tie together the various blocks that comprise it. Data structures and cryptographic modes need to be optimized for arbitrary sequences of changes, unlike the stream-oriented optimizations suitable for network message passing.

Basing the confidentiality of a file, however, on the confidentiality of its constituent blocks, however, is a weak approach. Observing the existence of changes in the first block of a file, say, is tantamount to a side channel, and it violates cryptographic standards for confidentiality (such as IND-CPA security [BD+98]).

The data access patterns that characterize storage as opposed to message passing will require new mechanisms with new optimizations, or a strict weakening of the security levels afforded to network settings. It remains to be seen whether the random-access, incremental nature of storage data patterns can be accommodated with similar ease to that of message streaming. Note that this is not merely a cryptographic issue but also one of client complexity and possibly key management.

Of course, one cannot say *a priori* that achieving equal performance is impossible, but it does suggest difficulty in equating storage security and network security.

7 Architectural Implications

In the OSI networking model, there are seven distinct layers with distinct levels of abstraction. Yet it might be said that they share one simple task: shipping bits from one place to another.

In the same vein, storage security and network security share one simple task: protecting bits in transit from one place or time to another. The central architectural question is whether the methods for protecting bits are essentially the same when they are shipped through space as when they are shipped through time.

Arguably, the underlying toolset is similar. Cryptographic methods are central. Thus, unlike the sharp distinction between wire voltage encodings (OSI layer 1) and session establishment (OSI layer 5), the degree or depth of abstraction is not much different. Perhaps network security and storage security are close enough cousins to be assigned to one floor of the architecture.

If so, however, we argue that there should be at least two distinct rooms on that floor. While there are superficial similarities and a common gene pool, the features and needs are sufficiently divergent to warrant separation.

Reflections on Architecture. An architectural model such as OSI is intended to achieve several goals. One is that of comprehension: an engineer should be able to grasp the general workings of a system and to operate at a given level of abstraction without overwhelming complexity. Another is to ensure that work on one component can proceed independently of work at another, with coordination through well-defined interfaces.

An aspect of this organization is that knowledge and techniques developed within a partition can be mapped to similar instances within that partition. Thus, for example, there

Proceedings of the First International IEEE Security in Storage Workshop (SISW'02)
0-7695-1888-5/03 $17.00 © 2003 **IEEE**

are many techniques for session establishment, yet they share sufficient commonality that experience and optimizations for one style of session can be mapped (or lend understanding) to another.

Confluence or Turbulence. The topic of of "security" is conceptually distinct from other architectural concerns. The question is whether "network security" is sufficiently distinguishable from "storage security." Let us touch on examples where blurring the two – carrying optimizations from one setting to the other – can lead to unforeseen vulnerabilities.

A natural way to secure stored data is to encrypt it on disk. An early example of this is Blaze's Cryptographic File System [B93], a local file-encryption system. Other variants with file-level or full-device encryption and network support can be found in further work, *e.g.* CryptFS [ZB+98], Secure Drive [S93], Secure FileSystem [G94], PGPdisk [N98], Secure File System [MK+99], Cepheus [F99], Microsoft Encrypting File System [M02], Secure Network Attached Disks [ML+02].

Alternative approaches store cleartext data and encrypt it for transmission across networks, *e.g.* Andrew File System [HK+88] Networked Attached Secure Disks [G99], iSCSI [SS+01]. The Microsoft EFS also decrypts the stored data and re-encrypts it for network transmission.

When storage and network security are commingled, it is tempting to optimize performance by transmitting the stored, encrypted file directly across the network. Indeed, many of the systems listed above do precisely that. It can be argued that this saves on the costs of apparently unnecessary re-encryption [RK+].

This introduces vulnerabilities unnecessarily, however. A network eavesdropper can tell whether the file has changed between different retrievals. Such side channel information is not available in a layered approach, where a disk-encrypted file is re-encrypted by a network security layer. By using a nonce or incremented IV, a network-encrypted message avoids leaking replay information (up to length).

A commingled approach jumbles vulnerability models. An eavesdropper on the Internet is not the same as an intruder with read-access to a disk behind a firewall. Yet the optimization of using disk-encryption alone will provide the weak Internet eavesdropper with otherwise unavailable powers.

As before (see §6), the replay of disk-encrypted data violates stricter cryptographic standards such as IND-CPA. An attacker can now infer information about the cleartext that would otherwise have been indistinguishable if the data had been re-encrypted or nested.

An architecturally pure implementation would separate the network needs and vulnerabilities from the storage needs and vulnerabilities. Returning to the OSI analogy, an application can perform error correction itself, but one does not therefore invade the data link layer to remove the error checking done there. In fact, there are likely to be optimizations done for the lower layer that are not easily seen to be subsumed by a particular implementation at a higher layer.

A vulnerability analysis for the network setting will often contain a different set of characteristics than the vulnerabilities for stored data. Combining network and storage security blurs these distinctions and may introduce unexpected vulnerabilities.

Proceedings of the First International IEEE Security in Storage Workshop (SISW'02)
0-7695-1888-5/03 $17.00 © 2003 **IEEE**

Conclusions

Although there is a great deal of symmetry between storage and network security, there is an equally strong set of reasons to treat them as separate topics, both conceptually and architecturally. At a high level, there is simply a need to protect bits along a path between a sending/storing event and a receiving/retrieving event. The symmetries break down on several levels, and there is significant reason to elevate a storage system to the status of an endpoint – an otherwise dangerous practice.

Because storage is essentially a one-way interaction between storer and retriever, the technical tools are more limited, and imposing policy and access control desires tends to require a proxy or management system in which some degree of trust must be placed. Re-encryption and algorithm evolution may require active, ongoing management of the data. Unlike the network setting, this tends to impose a separate endpoint between S and R.

Data manipulation patterns are different. In network settings, messages are streamed as units, whereas in storage settings, data is modified and accessed in arbitrary patterns. These requirements mean that algorithms and methods optimized for network security are unlikely to serve the storage setting well.

When networking and storage layers are blurred for optimization, adverse consequences can result. The vulnerabilities of networks are not necessarily correlated with the vulnerabilities of storage systems, and solutions for one may be insufficient for the other.

In sum, there is a variety of reasons to dismiss the idea that storage security is solved by leveraging network security, despite superficial similarities.

References

[BD+98] M. Bellare, A. Desai, D. Pointcheval, P. Rogaway. "Relations Among Notions of Security for Public-Key Encryption Schemes." *Proc. Crypto'98*, LNCS 1462, Springer-Verlag, 26–45 (1998).

[B79] G.R. Blakley. "Safeguarding Cryptographic Keys." *Proc. AFIPS Conf.* 48:313-317 (1979).

[B93] M. Blaze. "A Cryptographic File System for Unix." *Proc. ACM CCCS '93*, (1993).

[DK+02] Y. Dodis, J. Katz, S. Xu, M. Yung. "Key-Insulated Public Key Cryptosystems. *Proc. EuroCrypt'02*, LNCS 2332, Springer-Verlag, 65–82 (2002).

[DD+93] D. Dolev, C. Dwork, O. Waarts, M. Yung. "Perfectly Secure Message Transmission." *JACM* 40:1:17–47 (1993).

K. Fu. "Group Sharing and Random Access in Cryptographic Storage File Systems." MIT Master's Thesis (June 1999).

H. Gobioff. "Security for a High Performance Commodity Subsystem." Ph.D. Thesis, *CMU CS-99-160* (1999).

Proceedings of the First International IEEE Security in Storage Workshop (SISW'02)
0-7695-1888-5/03 $17.00 © 2003 **IEEE**

[G94] P. Gutmann. "Secure FileSystem." www.cs.auckland.ac.nz/~pgut001/sfs/index.html, (1994/2002).

[HJ+95] A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung. "Proactive Secret Sharing Or: How to Cope with Perpetual Leakage." *Proc. Crypto'95*, LNCS 963, Springer-Verlag, 339–352 (1995).

[HK+88] J. Howard, M. Kazar, S. Menees, D. Nichols, M. Satyanarayanan, R. Sidebotham, M. West. "Scale and Performance in a Distributed File System." *ACM TOCS* **6**:1 (1988).

[MK+99] D. Mazieres, M. Kaminsky, M. Kaashoek, E. Witchel. "Separating Key Management from File System Security." *Proc. SOSP'99*, 124–139 (1999).

Microsoft Corporation. "Encrypting File System for Windows 2000." www.microsoft.com/windows2000/techinfo/howitworks/security/encrypt.asp, (2002).

[ML+02] E. Miller, D. Long, W. Freeman, B. Reed. "Strong Security for Distributed File Systems." *Proc. FAST'02* (January 2002).

Network Associates, Inc. PGPdisk. www.pgpi.org/products/pgpdisk (1998/2002).

M.O. Rabin. "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance." *JACM* **36**:2:335–348 (1989).

[RS60] I.S. Reed, G. Solomon. "Polynomial Codes over Certain Finite Fields." *SIAM J. Appl. Math*, 300–304 (June 1960).

[RK+] E. Riedel, M. Kallahalla, R. Swaminathan. "A Framework for Evaluating Storage System Security." *Proc. FAST'02* (January 2002).

[SS+01] J. Satran, D. Smith, K. Meth, O. Biran, J. Hafner, C. Sapuntzakis, M. Bakke, M. Wakeley, L. Dalle Ore, P. Von Stamwitz, R. Haagens, M. Chadalapaka, E. Zeidner, Y. Klein. Internet Draft - iSCSI, www.ietf.org/internet-drafts/draft-ietf-ips/iscsi-08.txt (2001).

[S79] A. Shamir. "How to Share a Secret." *CACM* **22**:612–613 (1979).

[S93] E. Swank. "SecureDrive." www.stack.nl/~galactus/remailers/securedrive.html, (1993/2002).

[ZB+98] E. Zadok, I. Badulescu, A. Shender. "Cryptfs: A Stackable Vnode Level Encryption File System." *Columbia Tech. Rept. CUCS-021-98*, (1998).

Proceedings of the First International IEEE Security in Storage Workshop (SISW'02)
0-7695-1888-5/03 $17.00 © 2003 **IEEE**