A System for Long-Term Document Preservation

Larry Masinter, Michael Welch; Adobe Systems Incorporated; San Jose, CA

Abstract

This paper analyzes the requirements and describes a system designed for retaining records and ensuring their legibility, interpretability, availability, and provable authenticity over long periods of time. In general, information preservation is accomplished not by any one single technique, but by avoiding all of the many possible events that might cause loss. The focus of the system is on preservation in the 10 to 100 year time span—a long enough period such that many difficult problems are known and can be addressed, but not unimaginable in terms of the longevity of computer systems and technology.

The general approach focuses on eliminating single points of failure - single elements whose failure would cause information loss - combined with active detection and repair in the event of failure. Techniques employed include secret sharing, aggressive "preemptive" format conversion, metadata acquisition, active monitoring, and using standard Internet storage services in a novel way.

1. Introduction

Recorded history is a history of documents and records; individuals and groups keep information in recorded form for later retrieval. The documents and records are descriptions of events, financial transactions, government activities, stories, opinions and so forth. Often a single document is used both for managing a transaction as well as recording the fact of the transaction for later review. In the last 50 years, computer systems and information automation have moved work processes and records online. Electronic records, however, do not have the same longevity properties as physical documents, and many unsolved problems remain.

It is important, then, to be able to retain such records, and to ensure their legibility, interpretability, availability, and provable authenticity over very long periods of time. In general, information preservation is accomplished not by any one single technique, but by avoiding all of the many possible unfortunate events that might interfere with preservation or cause loss. A system for long-term storage and archiving must guard against all of the problems which might cause inadvertent or unplanned loss.

The desired lifetime of personal and business records ranges from weeks to months to years and, in some cases, to decades. Thus, it is important to protect records against events that are foreseeable in those time frames. However, the technical means for information preservation for centuries or millennia are hard to imagine. We therefore define "long-term" as a 10 to 100 year span.

We begin our description of a system for long-term preservation with a survey of the variety of events which may affect documents over time, and we describe the approach we take to protect documents against such events. The second part of the paper gives an overview of a system we have designed and prototyped which combines these approaches to provide for guaranteed long-term archives of documents and information. Finally, we conclude with an analysis of the system and a comparison to related work.

2. Causes of Loss and How to Avoid Them

The general approach taken for reducing the risk of loss is to acknowledge that loss is inevitable, but that overall reliability can be improved using redundancy and active correction. That is, we attempt to design a system in which there are no "single points of failure"—single elements whose failure would cause information loss—and combine this with active detection and repair of failure. Avoiding single points of failure is a fundamental principle applied to the design of reliable systems, and is essential in a system designed to provide long-term access to archived documents. Some single points of failure are easily recognized, such as using a single physical device or single location. Other risks, such as file formats, protocols, software implementations, or encryption algorithms, are less obvious, but no less important.

This section describes different kinds of failures, and the ways in which we try to guard against them.

2.1. Media Failure

Any practical digital media for recording data—disk, tape, CD-ROM, and the like—will eventually fail. Failure of recording media comes sooner than many imagine. Currently the technique used for avoiding media failure is periodic media refresh—read in the digital data, check for errors using error correction techniques, and rewrite on new media. There are many variations on this theme, including RAID [16] disks and backup strategies. We do not rely on any single media copy for data safety.

2.2. Site Failure

A more serious cause of data loss can come when the physical location for holding the media is somehow threatened. For example, through earthquake, flood, electrical storm, or other kinds of physical disasters, entire sites holding data may be in jeopardy. Frequently the method employed for protecting against site failure is to backup the data off-site. However, this approach has limitations. Over time, backup media may fail undetectably. The offsite backup storage location itself may be at risk. Our approach is to use distributed multi-site redundant storage, spreading information among multiple physical locations in different geographic areas, and to include periodic checks of storage integrity.

2.3. Organizational Failure

Even with multi-site redundant storage, if multiple sites are controlled by a single institution, storage is at risk if the organization fails—loses funding, goes bankrupt, or decides to discontinue its business for data storage. In some cases – when an organization is managing its own multi-site storage system –

organization failure isn't a risk that needs to be guarded against, because the need for long-term archives might also go away at the same time as the storage. Some systems, such as LOCKSS [18], employ multiple storage sites, each run by independent organizations with separate financial backing. Our approach is to use multiple independent storage providers.

2.4. Software Failure

Another kind of difficulty occurs when there are bugs, errors or security flaws in software. If each storage site is running the same software, there is the possibility that a single flaw in the software will put the integrity of multiple sites at risk. For this reason, we propose using standards-based protocols for access to data storage, where different storage sites are running different implementations of the storage software, so that the integrity and reliability of stored data does not depend on the integrity and reliability of any single implementation.

2.5. Malicious Modification or Destruction

For any of a number of reasons, long-term archives may be subject to attack. It is insufficient to rely on multiple copies without mechanisms for ensuring that a concerted attack on storage sites will not cause information to be destroyed or changed. We focus on increasing the number of sites which must be simultaneously attacked, and obscuring the identity of those sites for a particular document [8], as a way of reducing the risk of malicious modification or destruction [3].

2.6. Loss of Interpretability

Because the goal is to keep retrievable records of human activities, and not just sequences of bits, it is important to ensure that the data stored can be interpreted in the future. Thus, a long-term archive needs to be concerned with the data formats used and not just bit sequences. The relatively brief history of computer-based document processing is full of document formats for which interpreters are difficult to find or are unreliable. While others have imagined using "virtual machines" to store interpreters for individual file formats [15] [10], these themselves may be considered "single points of failure." Rather, we propose storing multiple representations of the same document at the time of creation of the archive. In addition, we propose that as many of those renditions as possible be "archival quality." What is an "archival quality" file format? It is one for which there are multiple interpreters, written in different languages, running on different operating systems, and for which there is a specification precise enough that it is possible to build a credible interpreter based solely on reading the specification.

Planning to migrate documents to new formats in the future, when the stored file formats are "almost obsolete," is not a practical solution. Delayed migration risks the possibility that, if information about a format is lost, it will be too late to migrate! Thus we emphasize early conversion, at time of archive, to one or more formats of archival quality.

2.7. Loss of Context

In a large archive of data, each document must be specifically identified and its context supplied; the context includes information such as the source of the document, its intended distribution, whether it is a draft or a final report, and other kinds of contextual information that, in conjunction with the document content, turn a document into a record.

Often, in day-to-day use of documents and content, this kind of contextual information is implicit—not explicitly represented and therefore at risk of being later forgotten. Thus, we include explicit acquisition of contextual (descriptive) metadata as an integral part of an archival record. For example, descriptive metadata for a text document might include the author, topic, creation date, distribution, and version of the work. Descriptive metadata for a photograph might include information about the subject, the date the photograph was taken, and the photographer.

Metadata also may suffer from "loss of interpretability"; thus it is important that metadata be stored in a format of archival quality.

2.8. Loss of Guarantee of Authenticity

Records of transactions may be subject to manipulation. For physical documents, while the technology for forgery may improve, the technology for detecting forgery also improves, to the point where it is frequently possible to be assured of the authenticity of a physical document.

However, the situation is more complicated for electronic records; simple maintenance processes or media refresh may cause loss of clues about document origins or dates and interfere with processes to manage physical custody of records which might otherwise be used to determine authenticity. Our approach to long-term guarantees of authenticity is to request that the storage services that are holding individual pieces of data also act as "notary" services, maintaining timestamps of receipt, possibly validating the authenticity of the document at the time of receipt, and individually guaranteeing the storage service's records of such facts. We propose that each original record have multiple notary services which individually attest to data authenticity, avoiding a single point of failure.

2.9. Authorization Failure

There are situations where multiple data storage locations might still be subject to centralized regulation, legal intervention, or other events affecting archived data even though the effects (such as data change or destruction) might be contrary to the original wishes of the principal creating the archive. For this reason, it is useful to consider the possibility of ensuring that the storage repositories are held under multiple legal jurisdictions, such that a legal intervention in a small subset of locations will still not cause the archived information to be lost, modified, destroyed, or revealed in ways inconsistent with the archiver's intent.

2.10. Loss of Privacy

While many individuals and organizations may have records or information that are not particularly sensitive or private, almost every organization has some records that are private. There are many different approaches to trying to guarantee privacy in storage systems, through both technical and operational means. Most of these systems, however, have a "single point of failure" because they rely on operational integrity of the storage system. In some systems, the data is stored encrypted, using a symmetric-key based encryption scheme, such as DES or AES, which uses random (or pseudorandom) encryption keys. The security of such key-based encryption relies on the inability of a polynomial time adversary to successfully recover the key and on the assumption that a particular underlying mathematical problem, such as factoring large prime numbers or determining quadratic residues, is "hard" and will remain so. Recent history has shown that algorithm security is relatively fragile [20].

Key-based encryption in a long-term archive suffers from two major drawbacks. First, the privacy of the document is based on the inability of an adversary to break the encryption within the time period during which the document must remain private. Clearly, if a document that must remain private indefinitely is archived, there is a problem, because an adversary would have an indefinite amount of time to break the key. Second, key-based encryption requires remembering an additional piece of information: the key. If the key is lost, so is the ability to decrypt the document. If the keys are stored online, they must be kept securely for the same duration as the document. Merely encrypting the key with another key does not reduce the complexity of the problem. Using key-based encryption may result in the use of non-random keys, such as keys based on a typed password. The minimal amount of entropy possible in passwords that can also be easily remembered leads to easily successful dictionary attacks.

To avoid the difficulties inherent in key-based encryption, we turn to "secret sharing" [19] [4], which provides for (N, K) threshold storage. An adversary who obtains fewer than K "shares" is unable to learn any information about the document, no matter how much compute power is applied. Threshold secret sharing does not require an encryption key, resulting in one less piece of data that must be remembered over the lifetime of the document.

3. System for Long-Term Archives

We now discuss the design and implementation of a system which provides for long-term archive and data preservation, based on the principles outlined above. The system design is discussed by reference to the functional steps used to perform the major operations.

3.1. Select Content to be Archived

Any system for long-term archives needs a mechanism for selecting the content to be archived. There are a wide range of possibilities, because archiving might be an automatic part of any kind of record transaction system, or might be initiated by a manual process. In our initial implementation, selection is a manual process, but other kinds of automatic selection are easy to imagine.

3.2. Prepare for Archive

There are several steps involved in preparing material for storage in an archival repository, independent of the manner in which storage is guaranteed.

3.2.1. Gather Descriptive Metadata

For an archive to be useful over the long term, it must include descriptive metadata. The process for gathering descriptive metadata is somewhat application-dependent, and might be integrated as part of some other process, or might be manually gathered and verified.

In our prototype, we gather simple descriptive metadata (author, date created, date archived); the system guesses and the user confirms the information in a dialog window. Other more complex processes for automatically gathering metadata are the area of active research [17].

We represent metadata using an XML data structure, Adobe's Extensible Metadata Platform [2]. XMP is a published standard which is of "archival quality"—there are multiple implementations on multiple platforms and the specification is explicit. XMP is used to embed metadata directly into a file.

3.2.2. Convert Content to Multiple Archival Formats

To protect against format obsolescence, we propose converting data from its original form into one or more "archival formats", such as PDF/A [13], XHTML [25], or DNG [1]. By archiving the same document in multiple formats, each document becomes its own virtual "Rosetta stone," aiding future interpretability and mitigating the chances of loss due to format obsolescence.

3.2.3. Package Together the Original, Conversions, and Metadata

There is some risk that documents, conversions and metadata will become separated, if stored separately. While there are many ways of linking multiple components, we have taken the straightforward approach of creating a single file which contains the original document, the metadata, and the converted (archival format) documents. Once metadata is gathered and the document is converted into appropriate formats, we maintain the relationship between them by bundling them together and storing them as a single "archival package."

3.3. Distribute to Remote Sites

Having prepared the material in a form suitable for archiving, we must now store the resulting package in a way that will guarantee its availability and integrity over a long period of time. This process involves several components. The basic principle that links these steps is the use of "secret sharing" [19] [4], in which we use an (efficient) implementation for creating N "shares" of data (where each "share" is (nearly) the same size as the original bundle), for distribution to N sites, any K of which are necessary and sufficient for reconstructing the original package.

3.3.1. Select Sites for Distribution

Secret sharing involves distributing packages of data to N sites, where any K of the sites' data are necessary to reconstruct the original. The larger K is, the more impervious the system is to malicious attacks on data privacy. The larger N-K is, the more impervious the system is to inadvertent or intentional data loss. The larger N is, the more costly data storage is; the larger K is, the more costly data retrieval will be. If privacy is not a requirement, we can use K=1, meaning every "share" is really just a copy of the original data.

Many of the threats facing long term storage, such as site failure, organizational failure, authorization failure, and malicious loss, are addressed by distributing copies of the archival data to numerous distinct remote locations. We must select remote sites such that the probability of loss due to geographic, organizational, economic or political failure is mitigated. This process involves some amount of evaluation of the qualities of the sites: Are they in the same geographic location? Do they share the same risks for political instability? Are they controlled by the same financial institution?

In our current implementation, we manually configure the selection of remote sites, but selection might be automated. Selection should take into consideration the geographical location and control of the site, in addition to the perceived trustworthiness, reliability and availability of the providers. Reliability ratings may be measured based on past observations [5]. Sites with a long history of availability or that are operated by well established organizations may be considered more likely to survive in the long term. We also consider other characteristics of the organization that operates a site. To reduce the risks of malicious loss, selecting sites operated by industry competitors might be a good choice, as they might be less likely to collude.

Sites could also be chosen based on existing contractual relationships. Parties interested in archiving their documents may enter into a contract, in which each party agrees to store data from the other. Techniques such as auctions and bidding for storage space [6] are also possible.

Based on the selection of sites for distribution and the evaluation of the reliability and risks associated with those sites, we can choose the variables N (number of shares) and K (threshold of shares to reconstruct).

Determining the number of shares to generate and the threshold for recovery involves a tradeoff between preserving the document and protecting privacy [24]. Our system provides flexibility by allowing the user to configure their desire for privacy versus document preservation. Support for public documents is provided by using a threshold of one, in which case each "share" is simply a copy of the original archival package.

3.3.2. Use Secret Sharing to Create Distribution

Given N and K established above, we use a novel and efficient algorithm for secret sharing to produce shares of the archival package. Shamir-style secret sharing requires generating random numbers (to be used as coefficients of polynomials) and doing arithmetic (multiplication, addition and division) on integers modulo a prime. The key insight that leads to improved performance was the use of 257 as the prime number, on a 32-bit processor. Each byte of the original document is treated as an individual "secret." This allows for efficient generation of multiple random numbers at a time (as a string of K random bytes), the use of a pre-computed lookup table to eliminate modulo division operations and 32-bit arithmetic with CPU-supported modulo operators to compute the polynomial value for each of the N shares. The results pack into vectors of 8-bit numbers, with a separate "overflow" bit-vector for those cases where the polynomial value equals 256 exactly. Each "share" is thus 12.5% larger than the original data.

3.3.3. Send Shares to Remote Sites

Once shares of the data have been generated, they must be sent to remote sites for storage. One of the evaluation criteria for site selection can be whether the sites are based on different operating systems and different storage system implementations, to protect against simultaneous loss of information. In order to increase the likelihood of finding such different, independent services, we have chosen to focus on storage sites where storage and access can be obtained using standard Internet services. Our implementation uses email. We select storage providers among Internet service providers (ISPs) around the world. We use SMTP to transmit shares to the ISPs, and POP or IMAP to retrieve, enumerate, or test the shares provided. We index the shares at a storage provider using additional headers and/or message subject.

This system can easily be extended to use web hosting (with WebDAV for storage and HTTP for access), FTP-based services, or other standards-based storage services.

3.4. Manage Archives

At this stage, the documents originally archived have each been shared and stored in multiple repositories. Now we must ensure that the data will persist unaltered for the desired lifetime of the document. Guaranteeing that data remains available and consistent is a difficult problem.

The lifetime of traditional archiving media, such as paper, may likely be maximized using a hands-off approach. Because physical paper media may be damaged by excessive handling, the content may be inspected for authenticity, and then placed in a physically secured location, such as a bank vault, and guarded from access. The archiver is generally confident that media placed in the safe will remain intact for a long period of time.

Digital media is susceptible to a wide range of situations that may jeopardize its integrity. Physical media for digital data, such as magnetic or optical disks, are fragile and subject to loss, ranging from random bit errors to entire disk crashes. Unlike paper media, the lifetime of digital media is much less predictable. We therefore take the position that digital media requires a form of active management: a periodic process that verifies the integrity of the digital data and performs any necessary recovery steps as the result of a detected failure. The following steps are necessary for active management of the archived material.

3.4.1. Organize Records of Content

We envision the management of records about archived data, searching the archives, and organizing the index of archived data to be a separate function from the archival storage itself. To support this, we create "proxy" files for each archived file, where the proxy itself contains a record of the identity of the services where the individual shares are stored, as well as a copy of its metadata. We have chosen a simple XML representation for this information. Such records can be searched, shared, displayed, and used as the material for indexing records on the local disk.

Part of the reason for separating archiving from archive management is that the technology and requirements for search and retrieval, indexing, and so forth are likely to change more rapidly than the archives themselves.

3.4.2. Store Data for a Long Time

Given that we are using commercial Internet Service Providers with email storage banks, this part of the system is not specifically a part of our prototype implementation. However, commercial storage services tend to use standard industry techniques for ensuring long-term data storage and integrity. Most utilize offsite backup; many use RAID storage and other storage network techniques for improving the reliability of their service.

Routine hardware maintenance and upgrades add additional storage space into the system and provide media refresh to further reduce the chances of media failure.

3.4.3. Monitor Stored Material for Loss or Modification

Simply distributing shares of a document to various locations is not enough to guarantee that the data will remain available for a long period of time. It is inevitable that data stored on digital storage media will eventually suffer from loss or corruption. Shares may be intentionally deleted or modified by a malicious site or user. Corruption may not be realized until the data is read as part of an attempted retrieval operation, when it may already be too late to recover from the damage. We therefore take a proactive approach, and periodically monitor the availability and integrity of archived shares.

Accidental loss may result from media or hardware failure, site or organizational failure, or operational errors such as accidental deletion. We also must consider malicious sites, which may intentionally destroy or alter a share while claiming it is available or unmodified. Our monitoring process will detect any loss or modification, regardless of the cause.

To protect against the threat of loss, we periodically query storage sites for the availability of shares. The data may be retrieved to guarantee that it is, in fact, accessible. Protecting against corruption or integrity failure requires more than verifying availability: we must also ensure that the contents are unaltered.

Verifying the integrity of a share does not require remembering the share data. Instead, we must only know small facts about the data, which are significantly smaller in size. We use a question and answer style of verification, where the question is such that only an unmodified copy of the data could answer the question correctly. The questions must be unknown to the storage provider prior to being asked. For example, a question may be a secure hash algorithm and a random nonce, and the correct answer is the result of hashing the share data with the nonce. Without prior knowledge of the nonce and algorithm, a malicious site cannot predetermine the hash result or attempt to find collisions. Many such information-theoretically secure questions can be imagined, such as computing an arithmetic or logical function on some subset of bytes.

We have implemented monitoring as a self-serve process, where users are responsible for monitoring their own archived data. Any party given the share location and question and answer sets, however, may perform the monitoring process. As monitoring does not require knowledge of the data, we envision supplementing self-monitoring with the use of independent third-party monitors to perform periodic checks and notify the owners upon detection of loss.

3.4.4. Repair When Data Loss Occurs

When a share is determined to be corrupt, we can take corrective action to repair the damage. Given at least the threshold number of the shares, we can recover the polynomials used to produce the shares, and an individual share may be reconstructed. This regenerated share is then resubmitted to the original site or distributed to a new site. If users themselves detect loss, they may retrieve and regenerate shares. Independent monitors may be authorized to initiate a repair process by using protocols for dynamically generating additional shares or changing the threshold of an encryption without requiring the reconstruction of the original data [23].

3.4.5. Recover Archived Content Records

Shares of archived documents are dispersed amongst a wide range of Internet accessible services. Location information is stored in a simple “proxy” record on the user's local system. Knowing the share locations is required to retrieve a document from the archive, so we must guarantee that the proxy records survive as long as the archived data.

We have implemented a simple process of recreating the proxy records of archived content from the archives themselves as a mechanism for disaster recovery. This is a fairly time consuming process (since it requires enumerating all of the archived content), but we have demonstrated the ability to reconstruct the index using only the archived content. It is possible to periodically archive the index of archived content itself as a way of speeding this recovery.

It is worth noting that as long as a sufficient number of shares of the data remain available in the network, their corresponding proxy record can be reconstructed, and so our monitoring and recovery of share data protects the proxy records as well.

3.5. Retrieve Data From Archive

Archived data must remain accessible regardless of the state of the document submission system. The retrieval process is independent, requiring only the ability to communicate with the remote sites holding shares of a document, using standard Internet protocols such as POP, IMAP, FTP, or HTTP. At archive time, authorized document retrievers are given proxy records containing all of the information necessary to locate and obtain shares to recover the original archival package, including the identity of each remote site, their last known location, and the necessary handles to retrieve a share. Access control to archived documents is controlled by knowledge of the share locations and handles. Given the vast potential space of share locations and handles, only users who have been given the proxy record can successfully locate enough shares of a given document.

To recover a document from the archive, a user issues a retrieval request with a particular proxy record. At least the threshold K number of the identified shares are retrieved and verified for authenticity. We then use Lagrangian polynomial interpolation to reconstruct the contents of the original package, which contains the original document and metadata. We have implemented retrieval as an automatic process given a proxy record. In the absence of such a process, these steps may be performed manually.

3.6 Delete Archived Data

To delete a document from the archive, it is necessary to convince a sufficient number of shareholders to delete their shares. Of course, the ability to delete a document introduces the risk that a document will be deleted inadvertently or maliciously. Thus, the access control for deletion must be controlled carefully.

4. Evaluation

In this section, we evaluate our prototype system, particularly in the context of design decisions made during implementation and deployment. We discuss the challenges and practical problems faced which were previously overlooked and only realized by actually attempting to build such a system.

4.1. Efficient Secret Sharing

Public key and threshold encryption are often considered too time-consuming for use on large pieces of data. To better quantify the efficiency of our modified Shamir secret sharing algorithm, described in section 3.3.2, we have taken timing measurements and compared our implementation to the open source secret sharing implementation from the Crypto++ library [7]. Our results show considerable improvement in both the time to produce shares and to recover a document from shares.

All measurements were taken on an Intel Pentium 4 1.8GHz processor machine with 1 GB of RAM running the Linux 2.6.7 kernel. The Crypto++ source code was modified to measure and report timing information and compiled with GCC 3.3.3. Our algorithm is implemented in Java and executed using the 1.5.0_05 compiler and virtual machine distributed by Sun Microsystems. For each experiment, we set the number of shares N = 10, and we vary the threshold K from 2 to 10.

Figure 1 shows the performance comparison of Crypto++ and our algorithm "SecretShare" for generating 10 shares of a 14KB source file as we vary the threshold K from 2 to 10. Figure 2 plots the time required to reconstruct shares with threshold K. The timing information for both of these figures is the average of 1000 runs. Figures 3 and 4 show the results of repeating the experiment using a 2MB source file, as the average of 100 runs.

Our algorithm outperforms Crypto++ in all four tests. As expected, the data shows that total computation time scales linearly with the threshold of encryption. In addition to higher performance, SecretShare scales better, particularly during reconstruction. Figure 4 shows that for the 2 MB file, Crypto++ scales linearly with the threshold, requiring an additional 1.1 seconds per increase in K, while SecretShare requires roughly 0.25 seconds per increase in K. Assuming a 10 Mbps Internet connection, reconstruction takes approximately one tenth of the time as retrieval. We currently retrieve shares and validate them before reconstructing, but this process could be pipelined to produce an overall improvement in speed.

We note that the performance of our implementation likely suffers additional delay as a result of running in the Java environment. A native code implementation of our algorithm may even further outperform the Crypto++ C++ implementation.

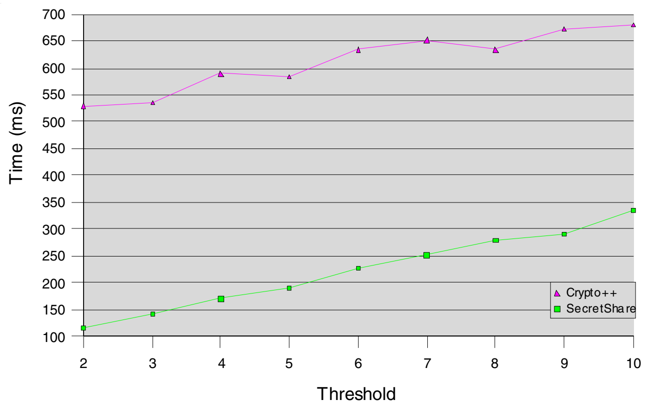

***Figure** 1. Generating 10 Shares from a 14KB source file*

4.2. The Benefits of Standard Protocols

Our initial implementation contained a "shareholder" service, which communicated with a simple protocol and remote method calls to submit, store, and retrieve shares. This design, while fully functional, suffered a serious flaw: survivability! It is important to remember that not only the data must survive, but also access to it. With a single implementation, any flaw in the code or underlying runtime system may cause permanent loss. If the specification for the "shareholder" service or protocol were lost, the shares would no longer be retrievable. We also considered the difficulty we would face trying to widely deploy such a service in a short period of time.

With these issues in mind, we redesigned our system to rely only on readily accessible Internet services based on standard protocols. Using protocols such as SMTP, POP, and IMAP offer numerous advantages: they are ubiquitous, deployed on servers throughout the globe; they have numerous implementations in different programming languages and on different operating platforms; and, their specifications are readily available as well published standards. These traits enable us to easily and quickly deploy an archiving system, and protect us against software and operating system flaws that might otherwise jeopardize archived data.

4.3. Recoverability

In our design we strive to eliminate single points of failure using techniques such as replication, distribution, threshold encryption, and format conversion. Despite such emphasis, we must also accept that eventual failures are inevitable. As a result, we must ensure that whatever is lost can be recovered.

Active monitoring allows us to detect errors at remote sites and recover from them by reconstructing the lost share. Equally as important, and initially overlooked in our design, is the ability to recover proxy records. Proxy records are likely not distributed and not replicated, resulting in a single point of failure. To facilitate their recovery, we must ensure that archived content contains sufficient information to reconstruct the proxy record. This leads us to an interesting trade-off. If share data contains sufficient information to reconstruct a proxy record, locating a single share may enable locating all of the other shares. The security of archived documents from malicious parties is enhanced by the anonymity of share locations.

We therefore chose to use a hash, which is stored with each share, to perform proxy record recovery, and we require the user to

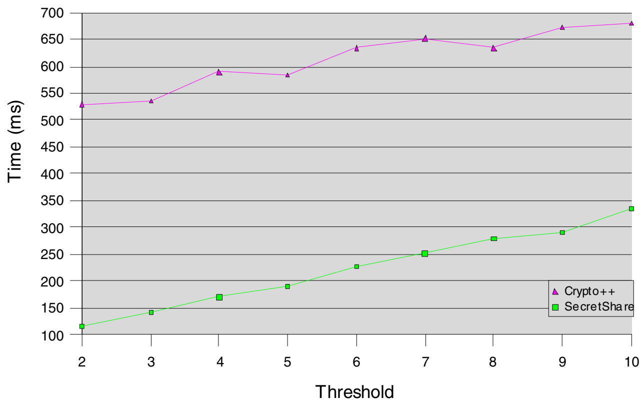

***Figure** 2. Reconstructing a 14KB source file*

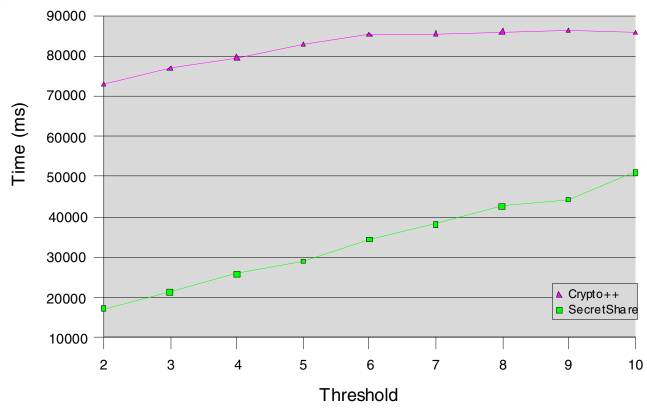

Figure 3. Generating 10 shares of a 2MB source file

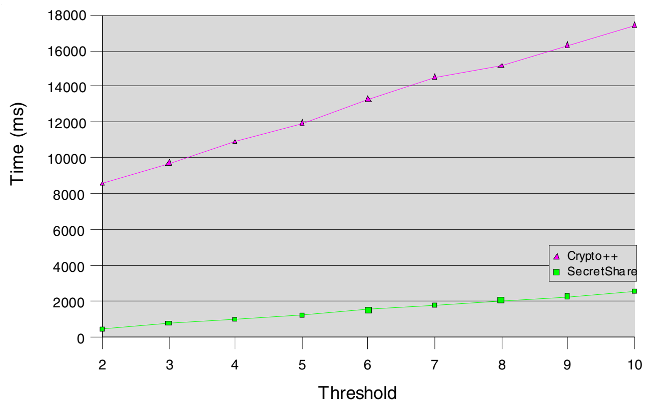

Figure 4. Reconstructing a 2MB source file

remember the possible locations of their shares (in this case, the email providers with whom they have accounts). By itself, this hash does not provide any information about other share locations. Only by enumerating the shares from several locations can we connect shares together and reconstruct a proxy record.

5. Related Work

The idea of achieving long-term preservation through redundancy and coding or information dispersal techniques has been investigated by others. OceanStore [14] is concerned with providing highly-available persistent storage through replication and caching. They offer "deep archival storage" as a side effect of versioning and replication. More recently, Silverback [22] uses a peer-to-peer overlay network with erasure coding and secure hashing to provide distributed "archival" versioning. Both systems use self-verifying GUIDs [21] as means for guaranteeing authenticity of data. Neither of these systems, however, addresses all of the potential threats necessary in a digital archiving system (privacy, context, interpretability).

Other digital library or archiving systems, such as LOCKSS [18], also use a form of periodic document integrity checking. Repositories in LOCKSS occasionally hold votes, where a repository poses a question about a document, and the others all publicly respond with their calculated answer. Repositories who vote in the majority are considered to have a valid document, and those who vote "incorrectly" refresh their copy from a valid source. Our system provides a similar function, but differs significantly in the requirements. We can perform share verification without any knowledge of the nature of the documents being checked. This feature makes possible the use of independent storage auditing. The system is limited to public documents.

Archival Intermemory [11] [12] uses distribution and erasure coding, modeling storage as persistent, write-once RAM. Intermemory provides long-term storage of bits but does not address archival issues such as security, authenticity, or interpretability.

DSpace [9] is a digital repository for preservation of scientific journals and other research materials, organized by communities and collections, including facilities for search. It is possible to archive data as a bundle which contains metadata and multiple representations of the document. DSpace does not effectively support document privacy, and as a single implementation, does not leverage standards based protocols to avoid failures due to software defects. The system may also be susceptible to malicious destruction or disruption by adversaries as the set of available DSpace servers is widely known (and even linked to on their website).

6. Future Work

The system outlined in section 3 attempts to address the issues outlined in section 2, by providing a total system which guards against many of the kinds of loss that might occur when storing records and documents for a long time. There are additional qualities and features which are desirable that we have not addressed fully.

First, there is the problem of identification and location of the actual services for storage of data over a long time -- what happens when storage services move? This problem is serious for long-term archives. Few data storage services have the same network address today that they did 20 years ago.

Secondly, there are some features for guaranteeing data integrity that we have considered but not implemented. It should be possible to combine a reliable and auditable data storage system with digital signatures, with the idea that a validated signature that is archived would carry along its validity for the lifetime of the archive. This would be a significant advantage over current timestamp systems in which the timestamp of the signature is only valid for the lifetime of the certificate of the timestamp authority, or else requires a process of periodic revalidation, which itself can introduce error and unreliability. In our current implementation, if the ISPs email storage service has no way for a user to modify the date of receipt, the email timestamp itself becomes a validator of the time of archive of the original data.

Some significant difficulties with access control over the long term also exist, both for the ability to read the documents in the future, but also for the control over the ability to delete documents. In our current implementation, access to archived data is controlled by access to multiple ISPs storage systems. These identities – typically controlled by username/password pairs – may not have a long enough lifetime to be useful for 10-100 year archives. For this reason, we believe long-term access control will require a combination of capability and role-based systems rather than identity-based access control. Such a system may also greatly simplify the discovery and reconstruction of lost proxy records by enabling a secure way for users to "remember" information about their share locations.

7. Conclusion

One problem to be solved before individuals and businesses will be comfortable with using all-digital documents, photographs and other electronic media is to provide assurances that the content is as safe, or safer, than their physical counterparts. We believe our work makes significant steps towards addressing major problems of long-term digital records by using a multi-faceted approach to guard against the many problems which might otherwise arise.

References

[1] Adobe Systems Incorporated, The Digital Negative (DNG), http://www.adobe.com/products/dng/

[2] Adobe Systems Incorporated, Extensible Metadata Platform (XMP), http://www.adobe.com/products/xmp/

[3] R. J. Anderson, The Eternity Service, Proceedings of Pragocrypt. (1996).

[4] G.R. Blakley, Safeguarding Cryptographic Keys, Proceedings of the National Computer Conference. (1979).

[5] B. Cooper, H. Garcia-Molina, “Peer-to-peer data trading to preserve information” ACM Transactions on Information Systems 20, 2. (2002).

[6] B. Cooper, H. Garcia-Molina, Bidding for storage space in a peer-to-peer data preservation system, Proceedings of the 22nd International Conference on Distributed Computing Systems. (2002).

[7] Crypto++ 5.2.1., http://sourceforge.net/projects/cryptopp/

[8] R. Dingledine, M. Freedman, D. Molnar, The Free Haven Project: Distributed Anonymous Storage Service, Proceedings of the Workshop on Design Issues in Anonymity and Unobservability. (2000).

[9] DSpace Federation, http://www.dspace.org/

[10] H.M. Gladney, R.A. Lorie, “Trustworthy 100-Year Digital Objects” ACM Transactions on Information Systems 22, 3. (2004).

[11] A. Goldberg, P. Yianilos, Towards an Archival Intermemory, Proceedings of IEEE Advances in Digital Libraries. (1998).

[12] A. Goldberg, P. Yilanos, Y. Chen, J. Edler, A. Gottlieb, S. Sobti, A Prototype Implementation of Archival Intermemory, Proceedings of the Fourth ACM International Conference on Digital Libraries. (1999).

[13] ISO 19005-1, Document management – Electronic document file format for long-term preservation. (2005).

[14] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, B. Zhao, OceanStore: An Architecture for Global-Scale Persistent Storage, Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems. (2000).

[15] R. Lorie, Long Term Preservation of Digital Information, Proceedings of the First ACM/IEEE Joint Conference on Digital Libraries. (2001).

[16] D. Patterson, G. Gibson, R. Katz, A Case for Redundant Arrays of Inexpensive Disks (RAID), International Conference on Management of Data. (1988).

[17] G. W. Paynter, Developing Practical Automatic Metadata Assignment and Evaluation Tools for Internet Resources. Proceedings of the Fifth ACM/IEEE Joint Conference on Digital Libraries. (2005).

[18] D. Rosenthal, V. Reich, Permanent Web Publishing, Proceedings of FRENIX Track: 2000 USENIX Annual Technical Conference. (2000).

[19] A. Shamir, “How to Share a Secret” Communications of the ACM 22, 11. (1979).

[20] X. Wang, Y.L. Yin, H. Yu, Finding Collisions in the Full SHA-1, The 25th Annual International Cryptology Conference. (2005).

[21] H. Weatherspoon, C. Wells, J. Kubiatowicz, Naming and Integrity: Self-Verifying Data in Peer-to-Peer Systems, Proceedings of the International Workshop on Future Directions in Distributed Computing. (2002).

[22] H. Weatherspoon, C. Wells, P. Eaton, B. Zhao, J. Kubiatowicz, “Silverback: A Global-Scale Archival System” U.C. Berkeley Technical Report UCB//CSD-01-1139. (2001).

[23] T. Wong, C. Wang, J. Wing, Verifiable Secret Redistribution for Archive Systems, Proceedings of the First International IEEE Security in Storage Workshop. (2002).

[24] J. Wylie, G. Ganger, P. Khosla, M. Bakkaloglu, M. Bigrigg, G. Goodson, S. Oguz, V. Pandurangan, C. A. N. Soules, J. Strunk, “Survivable Information Storage Systems” DARPA Information Survivability Conference and Exposition II, 2. (2001).

[25] W3C, XHTML 1.0: The Extensible HyperText Markup Language (Second Edition), http://www.w3.org/TR/xhtml1/

Author Biography

Larry Masinter is a Principal Scientist at Adobe Systems, where his work has focused on document processing, product interoperability and architecture. He worked at Xerox PARC on topics including document management, digital libraries, Internet service, and leading Internet and Web standards efforts. He received his BA in Mathematics from Rice University (1970) and his PhD in Computer Science from Stanford University (1980). He received the ACM Software Systems Award and is an ACM Fellow.

Michael Welch is a PhD student in Computer Science at the University of California, Los Angeles. He received his BS in Computer Science and Engineering (2003) and MS in Computer Science (2005) from UCLA. His research efforts focus on secure, private, and customized data storage and search.