A Comprehensive Reference Architecture for Trustworthy Long-Term Archiving of Sensitive Data

Detlef Hühnlein*, Ulrike Korte†, Lucie Langer‡ and Alex Wiesmaier‡
*secunet Security Networks AG, Germany
Email: detlef.huehnlein@secunet.com
†Federal Office for Information Security, Germany
Email: ulrike.korte@bsi.bund.de
‡Technische Universität Darmstadt – CASED, Germany
Email: {langer, wiesmaie}@cdc.informatik.tu-darmstadt.de

***Abstract*—It is well known that the suitability of many cryptographic algorithms decreases with time. Hence, it is a challenging task to maintain the integrity, authenticity, confidentiality and availability of sensitive data over very long periods of time. A scalable approach to preserve the integrity and authenticity of archived data has been standardized in [12]. The confidentiality and availability of data can be supported using secret sharing techniques according to [26]. This paper introduces a reference architecture for trustworthy long-term archiving of sensitive data based on the combination of these approaches.**

***Index Terms*—long-term archiving; secret sharing; integrity; authenticity; confidentiality; availability**

I. Introduction and motivation

Paper based workflow and document handling is increasingly replaced by electronic business processes. This strongly affects the trustworthy long-term retention of relevant documents: As electronic documents are virtual, they require special measures to counter security threats.

The German Federal Office for Information Security (BSI) has been developing a Technical Directive [7] which regulates trustworthy long-term archiving in German government agencies. The directive refines the standardized Open Archival Information System (OAIS) model [15], which provides integrity and authenticity. However, as confidentiality and availability are often important requirements for application scenarios in practice, the proposed solution is not suitable for these without modification. This paper therefore broadens the approach in [7], resulting in a comprehensive reference architecture which also provides confidentiality and availability.

After giving an overview over previous work in Section II, general requirements for long-term retention are described in Section III. Section IV explains the fundamental modules, interfaces, and processes of the comprehensive reference architecture. Section V focuses on the archive timestamp management. Section VI summarizes and concludes the paper.

II. Related work

The large amount of previous work, which motivated and coined our proposed reference architecture, can be grouped into the following categories. First (cf. II-A) there has been basic work which deals with the general aspects of long-term archiving of digital data. Second (cf. II-B) there has been work which mostly concentrates on how to maintain integrity and authenticity of data over a long period of time and how to build related systems. Third (cf. II-C) there is work which mostly concentrates on questions related to confidentiality and availability. It seems that our proposal combines the different requirements for the first time in order to build a comprehensive reference architecture for long term archiving of sensitive data.

A. General aspects of long-term archiving

The general aspects of long-term archiving of digital data have already been treated in textbooks like [4] and [11]. Furthermore, there is the standardized OAIS reference model [15]. The general requirements for long-term archiving services documented in [30] should as well be considered in the design of any long-term archiving system.

B. Integrity and authenticity

It is known since more than a decade that timestamps [14] can be used to maintain the integrity and authenticity of digital data – and in particular digital signatures [18], [29] – over long periods of time. It is clear that standards such as [5], [6] do not provide scalable and cost efficient solutions for long-term archiving, because the timestamp renewal would require a new qualified timestamp for *each* archive object – if the archive objects should remain independent, which is often an important requirement in practice.

In order to minimize the number of qualified timestamps required, one may use Merkle's hash trees [20] as standardized in [12]. Whereas [12] defines an ASN.1 based Evidence Record Syntax (ERS), there is an XML based ERS version on its way [2]. There have been first proposals to build more comprehensive long-term archiving services and systems around the ERS standard [3], [31].

C. Confidentiality and availability

In a naive perspective it would seem to be the canonical choice to use encryption in order to realize confidentiality when archiving data. There have been plenty of proposals in this direction [1], [16], [17], [21], [24]. However, the use of encryption algorithms usually provides only conditional

security and – just as for the signature and hash algorithms above – it would be necessary to provide additional conservation mechanisms in case the used keys or algorithms tend to become weak over time.

In order to avoid such cumbersome procedures it seems to be preferable to use secret sharing mechanisms instead of encryption, as it is well known that there are unconditionally secure mechanisms [26] for this purpose.[1]

Although there have already been proposals for data storage systems based on secret sharing [13], [27], [28] they do not seriously[2] consider the problems of long-term integrity. Hence, our reference architecture seems to be the first proposal which provides integrity and confidentiality at the same time.

III. Requirements for trustworthy long-term archiving

In the following we review the requirements for long-term retention in general [30].

Integrity: The integrity of any document retained must be ensured in a verifiable and conclusive way. The document must not be maliciously or accidentally altered, deleted or overwritten. Thus, there must be a means to detect whether a document has been altered or deleted, and the archiving system must provide methods to prevent undue modifications and to retrieve documents in their original state.

Authenticity: The authenticity of the documents must be preserved in order to keep the originator of the document identifiable and verifiable. As for integrity, it may be required to prove authenticity conclusively.

Readability: The data retained has to be readable, i.e. it must be possible to visualize the information contained. Thus, hardware to access the data must be available as well as software to interpret and present it.

Completeness: Any compilation of interrelated documents must be preserved in its entirety, i.e. the connection of these documents must be verifiable.

Negotiability: Documents under retention should be portable to other systems while maintaining the ability to verify their characteristics (e.g. their integrity). This is particularly important if the documents are to be used in a lawsuit.

Confidentiality: Any confidential information stored must be protected against unauthorized access.

Legal Compliance: The data must be stored in a way such that the applicable legal requirements are satisfied. If qualified electronic signatures according to § 2 Nr. 3 of the German Signature Act are archived, this implies for example that an integrity conservation mechanism with qualified timestamps must be applied.

Availability: The stored data must always be accessible by authorized entities. As system failures cannot be excluded this implies that a certain amount of redundancy is required for data storage.

[1]Note that the well-known one-time pad, which provides unconditionally secure encryption, may also be viewed as a 2 out of 2 secret sharing scheme.

[2]In order to provide integrity (against passive attackers) it is proposed in [27] to use algebraic signatures [25], but those signatures can be easily forged by an active adversary.

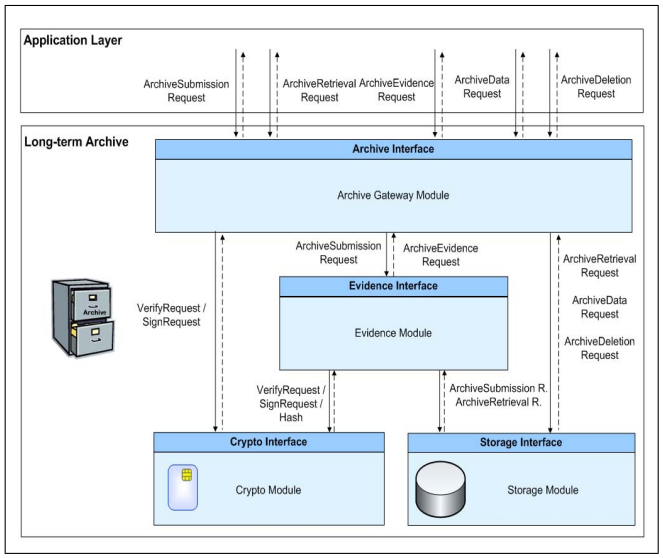

Fig. 1. Archive IT reference architecture

Migratability: As computing and storage technologies advance, the stored data must be easily migrated from one storage technology to its successor. Hence, a technology independent storage format is beneficial in the long term.

IV. Reference architecture

Based on the ERS standard [12] and the results of the ArchiSig[3] and ArchiSafe[4] [31] projects, the BSI has been developing a Technical Directive [7] which supports the integrity and authenticity of archived data. We extend and refine the architecture proposed in [7] in order to support confidentiality and availability as well. After a general overview we consider in particular the various modules (cf. IV-B) as well as the interfaces and processes (cf. IV-C).

A. Overview

The Long-Term Archive (LTA) of the proposed reference architecture (Fig. 1) consists of the following logical modules:

1) Archive Gateway Module (ArchiSafe)
2) Evidence Module (ArchiSig)
3) Crypto Module
4) Storage Module (eSafe)

The applications within the Application Layer (Fig. 1) may invoke the following functions:

1) ArchiveSubmissionRequest /-Response
2) ArchiveRetrievalRequest /-Response
3) ArchiveDataRequest /-Response
4) ArchiveEvidenceRequest /-Response
5) ArchiveDeletionRequest /-Response

[3]See http://www.archisig.de/

[4]See http://www.archisafe.de/

B. Modules

1) Archive Gateway Module – ArchiSafe: The ArchiSafe Module receives requests from applications such as Enterprise Resource Planning, Document Management Systems, or E-Mail. The objects to be archived are contained in a package which is referred to as XML Archival Information Package (XAIP) according to [15], which eases migration from one storage technology to another. If an application does not produce XAIPs, an additional component, called XML Adapter, is necessary to convert the archive package into an XAIP.

The ArchiSafe Module controls the processes and formats on the basis of standardized XML schemes and reviews the access rights of the callers. The security requirements of this module (especially concerning access control and information flow) are defined in a Common Criteria Protection Profile (PP) [8]. It is recommended to certify ArchiSafe products against this PP in order to assure that only reliable requests and archive objects can be sent to the Storage Module.

If the archive package is already signed on the application layer, the ArchiSafe Module calls the Crypto Module in order to verify the signatures. Furthermore, the ArchiSafe Module activates services of the ArchiSig Module and finally stores the archive package in the Storage Module.

2) Evidence Module – ArchiSig: In the ArchiSig project a concept has been developed which allows to renew the signatures of several documents by issuing just one timestamp. All documents are hashed and the resulting hash values are merged into a hash tree [12], [20]. Then, a timestamp for the root hash value of the tree is generated. This time-stamped hash value consolidates the validity of all involved documents at the time the timestamp was generated. By iteratively renewing this timestamp before the cryptographic algorithms or parameters used become weak or compromised, the validity of the documents is preserved for a potentially arbitrary period of time. This way legal compliance can be achieved. Upon request, the ArchiSig Module uses the stored hash trees and timestamps to generate Evidence Records (ERs) according to [12].

If cryptographic functions are needed, the ArchiSig Module requests them from the Crypto Module.

3) Crypto Module: The Crypto Module supports at least the following services: Generation of hash values, timestamps and (optionally) signatures as well as verification of signatures or timestamps along with associated certificate chains. If required, the Crypto Module can download the signer certificate from one of the authorized certificate service providers (CSP) and examine the certificates. Furthermore, the Crypto Module has the ability to ask for (qualified) timestamps from CSPs.

The Crypto Interface (Fig. 1) should comply with standardized interfaces such as [9], [23] in order to simplify the integration of cryptographic operations and to allow for interoperability between different Crypto Modules.

4) Storage Module – eSafe: This module supports a storage service, a search service and a deletion service. In [7] there are no functional requirements specified for this module besides the ability for a bit-exact reproduction of the stored data. With respect to our additional requirements of confidentiality and availability, however, we suggest to use a distributed architecture which ensures (k, n) threshold confidentiality. The idea is to split the archive object into shares, which are subsequently stored on different servers. This approach was introduced in [22] and uses secret sharing according to Shamir [26]. The advantage is that confidentiality does not hinge on the security of encryption algorithms that may be broken in the future. Thus, this approach is particularly suitable for long-term storage. Moreover, this concept enhances availability as any k out of n servers may be queried to provide a share in order to reconstruct the document.

In the following we briefly explain the method used to store a confidential document. In order to share a secret, choose n pairwise different public values $x_1, \ldots, x_n$ and a secret polynomial f of order $k-1$. The secret corresponds to $f(0)$. Apply the secret sharing algorithm [26], i.e. compute the function values $y_i = f(x_i)$ for $i = 1, \ldots, n$. In order to share a document, divide it into blocks of equal length and treat each block as a secret, i.e. apply the above-mentioned algorithm. For each block the values x_i are fixed while the secret polynomial varies. When all data blocks have run through the algorithm, the function values are sorted as follows: For each x_i, subsume all function values $y_i = f(x_i)$ of the iterations under a package called *share*. Thus, for each share-ID x_i there exists one share. Finally, the shares are distributed among the n storage servers. In order to reconstruct an archive object, all of its data blocks must be reconstructed by collecting k out of n shares and applying an interpolation formula.

C. Interfaces and processes

This section describes the operations which an application-level client can initiate. All interfaces are realized as web services using the basic request and response types defined in [23].

1) ArchiveSubmissionRequest /-Response: This service belongs to the main operations of the archive service. The client creates an XAIP, integrates it in the ArchiveSubmissionRequest and transfers the request to the archiving service.

The ArchiveSubmissionResponse contains a unique identifier called Archive Token. It enables the client to refer to the XAIP in order to export it (cf. IV-C2), to delete it (cf. IV-C5) or to request related ERs (cf. IV-C4). This service type is supported by the ArchiSafe, ArchiSig, and eSafe Modules.

If the XAIP contains digital signatures, these are verified using the VerifyRequest-function offered by the Crypto Module according to [9] and [23].

2) ArchiveRetrievalRequest /-Response: The operation ArchiveRetrievalRequest retrieves the archive object identified by the Archive Token. The archive service returns the XAIP. This service is supported by the ArchiSafe and eSafe Modules.

3) ArchiveDataRequest /-Response: The operation ArchiveDataRequest allows to request only a certain part of the archive object, for example its owner name or other metadata. This service is supported by the ArchiSafe and eSafe Modules.

4) ArchiveEvidenceRequest /-Response: This service enables a client to request an ER in order to verify the integrity and authenticity of a stored XAIP. The ArchiveEvidenceResponse returns an ER proving that the data existed at a certain point in time and has not been altered since then. If the archived data has been signed, the authenticity of the data is implied as well as the conclusiveness of the signature. This service is supported by the ArchiSafe and ArchiSig Modules.

5) ArchiveDeletionRequest /-Response: This service allows a client to delete an XAIP identified by the Archive Token. This service is supported by the ArchiSafe and eSafe Modules.

V. Archive timestamp management

The process of iteratively renewing the timestamp is an internal process of the Evidence Module. According to [12], the archive timestamp management splits into two different scenarios, called Timestamp Renewal (cf. V-A) and Hash Tree Renewal (cf. V-B). By applying the distributed secret sharing storage system a third scenario is implied, which we call Share Renewal (cf. V-C).

A. Timestamp Renewal

If the hash algorithm used to build the hash tree is considered secure, Timestamp Renewal is sufficient to preserve the validity of the hash tree. Before the algorithms or parameters used to generate the current timestamp of the root node are compromised or involved certificates expire, the current timestamp is enclosed in a new timestamp using valid certificates and secure algorithms and parameters [12].

B. Hash Tree Renewal

If the hash algorithm used to build the hash tree is considered insecure, Hash Tree Renewal is applied to preserve the validity of the hash tree. Before the current hash tree is compromised, the tree is rebuilt from scratch using a hash algorithm considered secure [12].

C. Share Renewal

As explained in Section IV-B4, each share belonging to the same archive object is stored on a different server. Whenever a server is added or removed from the system, for each archive object a share is added or, respectively, removed from the system. New shares can be added to the system without changing the existing shares. When removing a share from the system, the remaining shares of the same archive object have to be recomputed in order to invalidate the removed share. This is called *Share Renewal*.

Whether renewing the shares has influence on the archive timestamp management depends on the configuration of the application or, respectively, of the LTA. We present two basic modes, *Inside Shared Mode* and *Outside Shared Mode*, and sketch further possible modes.

Assuming a k out of n secret sharing, in both modes storing an archive object leads to storing n shares within the eSafe Module. Retrieving an archive object leads to accessing k shares in the eSafe Module. However, the modes differ in the number of shares that have to be accessed for building or renewing the hash tree. In addition, the modes provide different security properties.

In *Inside Shared Mode*, the secret sharing mechanisms are applied by the eSafe Module and are invisible outside the module. The application sends the entire objects to be archived to the LTA. Thus, the hash tree is built from hash values of the archive objects. The size of the tree depends on the number of archive objects. Building or renewing the tree leads to accessing k shares (by and within the eSafe Module) per archive object. Renewing archive object shares does not influence the archive timestamp management, as it does not influence the hash values in the hash tree. This is the more efficient mode, but the objects' confidentiality is not protected outside the eSafe Module by the secret sharing mechanisms.

In *Outside Shared Mode*, the secret sharing mechanisms are applied by the application and are visible outside the eSafe Module. The application sends the shared archive objects to the LTA. From the point of view of the ArchiSafe and ArchiSig modules, these shares are regular archive objects. Thus, the hash tree is built from the hash values of the shares. The size of the hash tree depends on the number of shares. Building or renewing the hash tree leads to accessing n shares per entire object. Renewing object shares influences the archive timestamp management, as the respective hash values in the hash tree have to be replaced. This is the less efficient mode, but the archive objects' confidentiality is protected outside the eSafe Module by the secret sharing mechanisms. In order to prevent attacks involving combining less than k shares, some countermeasures such as storing the value k together with the shares have to be taken.

Combinations or variants of the above-mentioned modes produce a good tradeoff between efficiency and security. A reasonable mode seems to be having the shares generated by the application, but nevertheless building the hash tree using the hash values of the entire objects. Another interesting variant is distributing the application-generated shares among several different LTAs.

VI. Conclusion and outlook

We introduced a comprehensive reference architecture for the trustworthy long-term archiving of sensitive data based on the primitives introduced in [12] and [26]. Integrity and authenticity is preserved by applying efficient timestamps, which are renewed in reaction to predictable security threats. Confidentiality and availability is achieved by applying secret sharing mechanisms. In contrast to the timestamps, the security of the shares does not degrade over time.

In order to improve the level of security and guard against a total breakdown of the system in case of an unexpected security threat, it seems appropriate to apply multiple algorithms concurrently for the cryptographic primitives used. The idea behind this concept is that the unexpected breakdown of a number of different algorithms at the same time is the more unlikely the more algorithms are applied. In order to take advantage of this effect it is important to combine the various

s	C_{all}	$T_{all}[yrs]$	X_{given}	X_{any}
10^3	$4 * 10^{10}$	1	$8 * 10^8$	$5 * 10^6$
10^6	$4 * 10^{22}$	10^{12}	$8 * 10^{20}$	$5 * 10^{15}$
10^9	$4 * 10^{34}$	10^{24}	$8 * 10^{32}$	$5 * 10^{24}$
10^{12}	$4 * 10^{46}$	10^{36}	$8 * 10^{44}$	$5 * 10^{33}$

TABLE I
SECURITY ESTIMATES FOR DIFFERENT VALUES OF s

algorithms correctly. For certificates and timestamps this can be done as proposed in [19]. For building the hash tree this can be done as shown in [10].

In order to enhance confidentiality and availability, the eSafe should not be part of a single LTA but rather be distributed among several LTAs as suggested in Section V-C. For the LTA, a share is then just an archive object like any other. The IDs of the object shares are managed by the client. In this scenario, the LTA is no single point of failure as any k out of n LTAs may be queried to provide a share in order to reconstruct an object. Furthermore, confidentiality is ensured on a high level as the archived object is not readable outside the client as long as k LTAs do not cooperate maliciously. As the LTAs are considered trustworthy, this risk is low. Thus, this approach provides a strong notion of confidentiality and availability.

Table I exemplarily sums up the calculations we made on the security of the eSafe. A more thorough and detailed presentation of the security analysis is subject to future work. For the presented values we assume a 4 out of 7 secret sharing and $1ms$ to recombine 4 shares: s is the total number of shares stored in the eSafe. C_{all} are the possible 4-combinations out of s shares. T_{all} is the time needed to build all 4-combinations i.e. to reconstruct all archive objects. X_{given} is the number of randomly picked 4-combinations needed to rebuild at least one given archive object with probability 0.5. X_{any} is the number of randomly picked k-combinations needed to rebuild at least one arbitrary archive object with probability 0.5. We see that for large but realistic values of s the effort to maliciously reconstruct archive objects is not practically affordable.

REFERENCES

[1] A. Adya, W. J. Bolosky, M. Castro, R. Chaiken, G. Cermak, J. Douceur, J. Howell, J. R. Lorch, M. Theimer, and R. Wattenhofer, "Farsite: Federated, available, and reliable storage for an incompletely trusted environment," in *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*. USENIX, 2002.

[2] A. J. Blazic, S. Saljic, and T. Gondrom, "Extensible markup language evidence record syntax," Internet Draft of August 3, 2009.

[3] A. J. Blazic and P. Sylvester, "Provision of long-term archiving service for digitally signed documents using an archive interaction protocol," in *EuroPKI*, ser. Lecture Notes in Computer Science, vol. 3545. Springer, 2005, pp. 240–254.

[4] U. M. Borghoff, P. Rödig, and J. Scheffczyk, *Long-Term Preservation of Digital Documents, Principles and Practices*. Springer, 2003.

[5] ETSI, "Technical Specification XML Advanced Electronic Signatures (XAdES)," TS 101 903, Version 1.3.2, 2006.

[6] ETSI, "Electronic Signatures and Infrastructures (ESI) – Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES)," TS 101 733, Version 1.7.4, 2008.

[7] Federal Office for Information Security, "Trustworthy Long Time Archiving," Technical Directive (BSI-TR-03125), 2009, in preparation.

[8] Federal Office for Information Security and Federal Physical Technical Institute, "Common Criteria Protection Profile for an ArchiSafe Compliant Middleware for Enabling the Legally compliant Long-Term Preservation of Electronic Documents," BSI-CC-PP-0049-2008, 2008.

[9] Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik), "eCard-API-Framework – part 1 - 7," Technical Directive (BSI-TR-03112), Version 1.1, 2009.

[10] M. Fischlin and A. Lehmann, "Security-amplifying combiners for hash functions," in *Advances in Cryptology - Crypto 2007*, ser. Lecture Notes in Computer Science, vol. 4622. Springer-Verlag, 2007, pp. 224–243.

[11] H. M. Gladney, *Preserving Digital Information*. Springer, 2007.

[12] T. Gondrom, R. Brandner, and U. Pordesch, "Evidence Record Syntax (ERS)," Request For Comments – RFC 4998, August 2007.

[13] G. R. Goodson, J. J. Wylie, G. Ganger, and M. K. Reiter, "Efficient byzantine-tolerant erasure-coded storage," in *Proceedings of the 2004 Intl Conference on Dependable Systems and Networking (DSN 2004) (June 2004)*. IEEE Computer Society, 2004, pp. 135–144.

[14] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," *Journal of Cryptology*, vol. 3, no. 2, pp. 99–111, 1991.

[15] International Standardization Organization, "ISO 14721: space data and information transfer systems – open archival information system – reference model," International Standard, 2003.

[16] A. Iyengar, R. Cahn, J. A. Garay, and C. Jutla, "Design and implementation of a secure distributed data repository," in *Proceedings of the 14th IFIP International Information Security Conference (SEC 98)*, 1998, pp. 123–135.

[17] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: scalable secure file sharing on untrusted storage," in *Proceedings of the Second USENIX Conference on File and Storage Technologies (FAST) (San Francisco, CA, Mar. 2003)*. USENIX, 2003, pp. 29–42.

[18] P. Maniatis and M. Baker, "Enabling the archival storage of signed documents," in *Proceedings of the 2002 Conference on File and Storage Technologies (FAST)*. USENIX, 2002, pp. 1–14.

[19] S. Maseberg, "Fail-Safe-Konzepte für Public Key Infrastrukturen," Ph.D. dissertation, Department of Cryptology and Computeralgebra, Technische Universität Darmstadt, July 2002.

[20] R. C. Merkle, "Protocols for public key cryptosystems," in *Symposium on Security and Privacy, Oakland, CA, USA*, April 1980, pp. 122–134.

[21] E. L. Miller, D. D. E. Long, W. E. Freeman, and B. C. Reed, "Strong security for network-attached storage," in *Proceedings of the 2002 Conference on File and Storage Technologies (FAST) (Monterey, CA, Jan. 2002)*. USENIX, 2002, pp. 1–14.

[22] T. Miyamoto, S. Doi, H. Nogawa, and S. Kumagai, "Autonomous distributed secret sharing storage system," *Systems and Computers in Japan*, vol. 37, no. 6, pp. 55–63, 2008.

[23] OASIS, "Digital signature service core protocols, elements, and bindings, version 1.0," 2007.

[24] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Zhao, and J. Kubiatowicz, "Pond: the oceanstore prototype," in *Proceedings of the Second USENIX Conference on File and Storage Technologies (FAST)*. USENIX, 2003, pp. 1–14.

[25] S. J. Schwarz and E. L. Miller, "Store, forget, and check: Using algebraic signatures to check remotely administered storage," in *Proceedings of the 26th International Conference on Distributed Computing Systems (ICDCS 06) (Lisboa, Portugal, July 2006)*. IEEE, 2006.

[26] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, pp. 612–613, 1979.

[27] M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti, "Potshards: Secure long-term storage without encryption," in *USENIX Annual Technical Conference*. USENIX, 2007, pp. 143–156.

[28] A. Subbiah and D. M. Blough, "An approach for fault tolerant and secure data storage in collaborative work environements," in *Proceedings of the 2005 ACM Workshop on Storage Security and Survivability (Fairfax, VA, Nov. 2005)*. ACM-Press, 2005, pp. 84–93.

[29] C. Troncoso, D. D. Cock, and B. Preneel, "Improving secure long-term archival of digitally signed documents," in *Proceedings of the 4th ACM international workshop on Storage security and survivability*. ACM-Press, 2008, pp. 27–36.

[30] C. Wallace, U. Pordesch, and R. Brandner, "Long-term archive service requirements," Request For Comments – RFC 4810, March 2007.

[31] W. Zimmer, T. Langkabel, and C. Hentrich, "Archisafe: Legally compliant electronic storage," *IT Professional*, vol. 10, no. 4, pp. 26–33, 2008.