**Noname manuscript No.**
(will be inserted by the editor)

# Authenticity, Integrity and Proof-of-Existence for Long-Term Archiving: a Survey

**Martín A. G. Vigil · Daniel Cabarcas · Alexander Wiesmaier · Johannes Buchmann**

the date of receipt and acceptance should be inserted later

**Abstract** Digital archives that store electronic data for long periods are increasingly necessary for applications such as libraries, land registers, and medical records. Archived data is useful if protected from tampering to ensure authenticity, integrity and a date on which the existence of archived data was witnessed. We survey solutions that protected archived data using cryptography. We describe them and verify whether they successfully provide indefinite protection despite of the limitation of current cryptography schemes. Solutions are compared in regard to their goals, trust assumptions and efficiency. Based on our analysis, we elucidate open problems and promising research directions.

**Keywords** Long term · Archiving · Authenticity · Integrity · Existence

**Mathematics Subject Classification (2000)** 68P25 · 68P30 · 94A60

## 1 Introduction

A major task in old and modern societies is to preserve information for long time, often unbounded. Information is conveyed by data, whose storage and protection is usually delegated to archives. Protection is necessary to ensure that archived data remains trustworthy, i.e. data is faithful to its original purport and is free from tampering.

Archives are increasingly digitized, therefore they have to provide long-term protection to digital data. In this paper we focus on the protection goals *integrity* ("data have not been altered since their creation"), *authenticity* ("the origin of

Martín A. G. Vigil, Daniel Cabarcas and Johannes Buchmann
Technische Universität Darmstadt
Hochschulstraße 10, 64289, Darmstadt, Germany
E-mail: {vigil, cabarcas, buchmann}@cdc.informatik.tu-darmstadt.de

· Alexander Wiesmaier
AGT Group (R&D) GmbH
Hilpertstraße 20a, 64295 Darmstadt, Germany
E-mail: awiesmaier@agtgermany.com

the data can be identified"), and *proof-of-existence* or *witnessed-existence*("a time reference when the data was witnessed can be identified") [1]. We present and compare solutions for protecting digital data that is indefinitely stored in archives. Yet, there are archives that provide long-term confidentiality for digital data. For an overview of this subject, we refer the reader to [17].

Protection of digital data in the long term is challenging. A problem is to efficiently ensure long-term protection relying on cryptographic techniques, which do not provide long-term security. Therefore, regular updates on cryptographic techniques are necessary to achieve long-term protection. This is an end-less and complex process that archivists are reluctant to take [29]. Another problem is that cryptography techniques are not suitable for evolving data objects. For example, a digital signature can be verified only if the signed data object's bits have not changed. However, archivists need to update data objects' content in order to keep them compatible with new hardware and software [56].

Solutions for long-term protection make different trust assumptions. There is often a trade-off between the strength of the assumptions and the efficiency of the solution. In this paper we are particularly interested in exposing this trade-off for different long-term protection solutions.

### 1.1 Examples of Digital Archives

Archives are increasingly digital thanks to the low cost of storage and easy access. We present real-life examples of digital archives to show the relevance of this survey.

- Land registers keep track of transactions regarding land properties. Recently, some land register have migrated their data to digital media in order to reduce physical space and allow public access. Examples of such a service are the French land register of Alsace-Moselle [12] and the Estonian Land Register [26].
- Electronic invoices can be used for tax purposes in the European Union according to Directive 2001/115/EC [4]. Therefore, the storage of invoices is necessary. For instance, Slovenia provides a web tool [11] for tax declaration that requires companies and individuals to store invoices regarding immovable property for at least 20 years [9].
- In several countries, the heath sector has to store patients' medical records for years. Digital media is the preferred storage means. The retention time for medical records varies. For instance, in Germany they are retained for 10 years after the end of a treatment [21], or 30 years if x-ray was used [20]. In the United Kingdom (UK), retention is obligatory until 10 years after the patient's death [7].
- Patents are commonly preserved for several years. For instance, in the UK preserved periods can last up to 27 years [10].
- Digital libraries are a major example of archives. The US Library of Congress is in charge of storing and providing free access to digital content concerning the

[1] We prefer to call this goal *witnessed-existence* instead of *proof-of-existence* because we can only aim at producing evidence that the existence of an object was witnessed. A proof is out of reach with current technology.

American experience [1]. Other examples of digital libraries are found around the world, such as The National Library of Sweden [5], Library and Archives Canada [6] and The British Library [8].

For further examples, we refer the reader the case studies in InterPARES 2 [2].

### 1.2 Terminology

The literature covered by this survey uses inconsistent terminology. In order to be able to compare the different approaches this papers uses the following terminology taken mostly from RFC 4810 [59], and augmented with a few additional terms.

- **long-term**: an unbounded period.
- **data object**: a unit of digital data. For example, a document or a photograph recorded as a digital file.
- **archive**: an infrastructure of equipments and policies to preserve trustworthy data objects.
- **archivist**: a trusted entity that is committed to preserve trustworthy data objects in an archive [24].
- **submitter**: an entity that submits data objects to an archive.
- **retriever**: an entity that retrieves data objects from an archive.
- **validation evidence**: the evidence necessary to demonstrate that a data object is trustworthy.
- **trust assumption**: reliance on something or someone that is taken for granted.

### 1.3 Organization

This paper is organized as follows. Section 2 presents the security components commonly used to protect digital data. Section 3 presents solutions for long-term protection. They are compared in regard to trust assumptions, protection goals and format migration in Section 4, and efficiency in Section 5. In order to give a complete view of the field, we also present a number of archival solutions that fail to provide long-term protection in Section 6. Finally, Section 7 provides our final considerations and future work.

## 2 Security Components

In this section we introduce basic security components that are used as building blocks to protect data object in digital archives.

### 2.1 Cryptographic Hash Function

A *cryptographic hash function* or *hash function* is a mathematical function $h$ that maps bit strings of arbitrary length to strings of fixed length [46] and satisfies the following properties: a) given an output $y = h(x)$, finding $x$ is computationally

infeasible ("pre-image resistance"); b) given an input $x$ and the output $y = h(x)$, finding an input $x'$, such that $x' \neq x$ and $h(x') = y$ is computationally infeasible ("second pre-image"); and c) finding any different inputs $x$ and $x'$, such that $h(x) = h(x')$, is computationally infeasible ("collision resistant"). An important application of hash functions is verifying the integrity of a data object.

The security of a hash function relies on the hardness of defeating one of the hash function's properties. Therefore, security fades out as computer power and cryptanalysis evolve. For example, the hash function MD5, which was considered secure, is no longer secure [55]. In addition, a brute force attack is always possible, thus, in the long term, no single hash function can be secure.

## 2.2 Merkle Trees

A *Merkle tree* [47] is an ordered binary tree that is constructed from a sequence of data objects $(d_1, \ldots, d_n)$ using a hash function $h$. The leaves of the tree are the hash values $h(d_i)$ for $1 \leq i \leq n$ (in that order). A node that is not a leaf is $h(l\|r)$, where the left ($l$) and the right ($r$) child are concatenated ($\|$). Figure 1 shows an example of a Merkle tree.

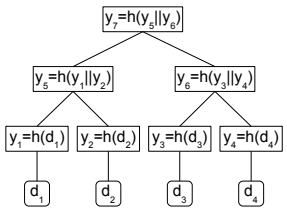


**Fig. 1** A Merkle tree whose leaves $y_1 \ldots y_4$ are digests of data objects $d_1 \ldots d_4$ and nodes $y_5 \ldots y_7$ are digests of the concatenation of their respective sibling nodes

Merkle trees can be used to verify whether a data object is an uncorrupted member of a sequence in the right position. The root of the tree serves as a digest for the entire tree. A data object is verified with respect to the root. Verification is done by reconstructing the path from the leaf to the root using the so-called *authentication path*, which consists of the siblings of the nodes in the path. Verification grows logarithmically in the number of elements in the sequence [47].

## 2.3 Write-once Media

A write-once medium, or write-once-read-many (WORM) memory, is a medium on which a data object can be written only a single time. Therefore, afterwards it is not possible to change the data object's copy on the medium. Examples are optical disks and software-hardware solutions that emulate the write-once property (e.g. [3]). By writing a data object on write-once media, we can produce evidence of integrity for the data object.

A wide-visible medium is a write-once medium that ensures not only integrity, but also authenticity and witnessed-existence. It is a *trusted* and public record

whose long-term preservation at many locations makes tampering verify difficult [15]. Therefore, we can produce evidence of authenticity for a data object by writing an authentic copy of it on wide-visible media. Examples of such media are broadcast protocols to spread data [16] and newspapers. Wide-visible media are extensively witnessed, thereby ensuring witnessed-existence for data objects. This is because witnesses associate a data object on wide-visible media to the date *when* they came into contact with the data object. In addition, a wide-visible medium associates the witnessed-existence of a data object with other objects written on the same medium.

In the long term, it is not clear whether write-once and wide-visible media will be available in the future or how they will be accessed. For example, these media can disappear due to physical deterioration and witnesses can die.

### 2.4 Digital Signature

A *digital signature scheme* consists of three algorithms. The *key generation algorithm* generates a *secret signing key* and a *public verification key*. Given a data object and a secret signing key, the *signing algorithm* computes a *digital signature* on the data object. Correspondingly, given a data object, a signature on the data object, and a public verification key, the *verification algorithm* decides if the signature is valid or not valid. Examples of digital signature schemes currently used in practice are *RSA* [53] and variants of *ElGamal* [30, 40].

The security of a digital signature scheme relies on the hardness of generating valid signatures on data objects that were not approved by the legitimate signer (see [32] for a different variant of the notion of security). As computer power and cryptanalysis evolve, security of a digital signature scheme fades out. Moreover, for many such schemes, security assumes adversaries with limited computational power. This assumption becomes implausible in the long-term.

### 2.5 Public Key Infrastructure

The use of digital signatures relies on the availability of trustworthy public keys. *Public key infrastructures* provide such keys by issuing and disseminating *certificates* [41]. Certificates are data objects that are digitally signed by some *Certification Authority* (CA), which asserts that a public key belongs to a certain subject. A CA can sign certificates for itself and other entities, such as CAs and final users. The path of certificates between the CA and a descendant certificate is called *certificate chain*. The most used standard for certificates is X.509 [22].

Certificates have an explicit expiration date. However, various circumstances may cause a certificate to become invalid before its expiration date. For instance, if the corresponding private key was compromised. In such case, the issuer CA revokes the certificate. A CA is responsible for disseminating the revocation status of unexpired certificates that the CA issued. Revocation status is often published using Certificate Revocation List [22] and Online Certificate Status Protocol [50].

The trustworthiness of a certificate fades in the long term. A reason is the reliance on the certificate's signature, whose security weakens over time. Another reason is the lack of updated revocation status. After the certificate expires, its

revocation status is no longer updated by the issuer CA. Consequently, the certificate's trustworthiness cannot be guaranteed after expiration.

### 2.6 Timestamp

A trusted entity named *Timestamp Authority* (TSA) asserts the witnessed-existence of a given data object by issuing a *timestamp*. The most common standard for timestamps is the *PKIX timestamp* [13]. It consists of a digital signature on the data object's digest together with the current date and time. Another type of timestamp consists of the digest of a data object published on a wide-visible media (cf. Section 2.3).

A TSA can anchor a timestamp to a previous time-stamp, thereby establishing an order of creation between them. To do so, the TSA creates a timestamp for the hash of the concatenation between a data object and an existing timestamp. This technique is called *hash linking* and is detailed in [37]. Also, a TSA can efficiently timestamp several data objects using a Merkle tree. The TSA calculates a Merkle tree, whose leaves are the digests of each data object, and timestamps the tree's root. This procedure is described in [15].

A single timestamp does not provide long-term security because it relies on a cryptographic hash function (cf. Section 2.1), a digital signature scheme (cf. Section 2.4) or a wide-visible media (cf. Section 2.3).

### 2.7 The Lifetime of Cryptographic Algorithms

The security of cryptographic algorithms such as hash functions and digital signature algorithms is not everlasting (see Sections 2.1 and 2.4). Researchers and government agencies estimate for how long such algorithms will remain secure based on the best known cryptoanalitic techniques e.g. [43,19,51]. Estimates are subject to be revised if more efficient cryptanalysis is discovered.

## 3 Solutions for Long-Term Protection

In this section we describe a number of solutions that provide long-term authenticity, integrity and/or witnessed-existence for data objects in digital archives. To the best of our knowledge, only these solutions successfully address these protection goals in the long term.

### 3.1 ETSI Standards

In [27,28] the European Telecommunications Standards Institute (ETSI) proposes a solution that provides evidence of long-term integrity and witnessed-existence for signed data objects. This evidence extends the trustworthiness of a data object's signature beyond the lifetimes of certificates, timestamps and cryptographic algorithms, thereby ensuring long-term authenticity. The solution is based on digital signatures (cf. Section 2.4) and PKIX timestamps (cf. Section 2.6).

There are three players in this solution: an archivist, who preserves a set of data objects in the archive; a retriever, who can verify integrity, authenticity and witnessed-existence of data objects from the archive; and Timestamp Authorities (TSAs) that issue timestamps on the archivist's request.

A data object is encapsulated in a data structure that we refer to as *archival package*. The standard requires that the data object is signed by its originator and the signature is included in the *archival package*.

For each data object, the archivist maintains a sequence of timestamps that is used as evidence for the data object's integrity and witnessed-existence (this technique was originally proposed by Bayer et. al. in [14]). Before requesting the first timestamp, the archivist collects the data object signer's certificate chain and revocation status and includes this data in the *archival package*. A first timestamp is issued on the archival package and is included in the archival package. The next timestamps are constructed as follows. Before the most recent timestamp's security is over (see Sect. 2.6), the archivist collects the TSA's certificate chain and revocation status, includes this data in the *archival package*, requests a new timestamp on the *archival package*, and includes the new timestamp in the *archival package*.

In order to verify integrity, witnessed-existence and authenticity of a data object, a retriever verifies the signature on the data object, each timestamp, each certificate chain and each validation status included in the archival package. The verification of a timestamp consists of verifying the timestamp's digital signature using the TSA's public key. Note that the certificate chain and revocation status of the TSA that issued the most recent timestamp is not available in the archival package. The retriever collects this data by himself.

Indeed, this procedure proves integrity, witnessed-existence and authenticity of the data object at any verification time. Suppose the first timestamp is issued at time $t_0$ on an *archival package* $a_0$ which consists of: the data object, its signature, and evidence that shows that the key used to verify the signature is trusted at time $t_0$. Now suppose that the retriever verifies that the signature is valid at a time $t > t_0$. This would suffice to prove authenticity if the key used to verify the signature is still trusted at time $t$. However, as we saw in Section 2.4, this may not be the case. As we will show, a sequence of valid timestamps proves that $a_0$ has not been modified since $t_0$, a time when the key was still trusted, and thus proves the authenticity of the data object

A valid first timestamp proves that $a_0$ was not modified from $t_0$ until the time $t_1$, when a second timestamp is issued. Lets refer to the *archival package* at time $t_1$ as $a_1$. The second timestamp is issued on $a_1$ which consists of: $a_0$, the first timestamp, and evidence that shows that the TSA's key used to verify the first timestamp is trusted at time $t_1$. These three pieces of information constitute a proof that $a_0$ existed at time $t_0$ and that it has not changed since then. As long as the second timestamp is trusted, it proves that at time $t_1$ there was a proof that $a_0$ existed at time $t_0$. Therefore the second timestamp extends the proof that $a_0$ existed beyond $t_1$, even if the first timestamp is no longer trusted after $t_1$. In general, if for every $i = 0, \ldots, n$, the $i^{th}$ timestamp is issued at time $t_i$ and it is trusted between $t_i$ and $t_{i+1}$, then, as long as the $n^{th}$ timestamp is trusted, it proves that at time $t_n$ there was a proof that $a_0$ existed at time $t_0$. Since $a_0$ contains the data object, integrity and witnessed-existence follow. Moreover, since

$a_0$ constitutes a proof that the data object was authentic at time $t_0$, authenticity follows.

There are two assumptions for the ETSI solution to guarantee long-term integrity, witnessed-existence and authenticity of a data object: that the underlying PKIs and TSAs are reliable, and that the cryptographic algorithms are not unexpectedly broken.

### 3.2 Evidence Record Syntax

Evidence Record Syntax (ERS) is a solution for long-term protection of digital data proposed by Gondrom et al. in RFC 4998 [34]. The solution provides long-term evidence of integrity and witnessed-existence for data objects. If data objects have been previously signed, long-term authenticity is guaranteed. The solution is based on Merkle trees (cf. Section 2.2) and PKIX timestamps (cf. Section 2.6).

There are three players in this solution: an archivist, who preserves a set of data objects in the archive; a retriever, who can verify integrity, authenticity and witnessed-existence for data objects from the archive; and Timestamp Authorities (TSAs) that issue timestamps on the archivist's request.

In a nutshell, ERS works as follows. For each data object, the archivist maintains a structure called *Evidence Record*, which is used as evidence for the integrity and witnessed-existence of the data object. An *Evidence Record* contains a set of timestamps[2]. A single timestamp can cover a group of data objects by using a Merkle tree, however, a copy of the timestamp is included in each object's *Evidence Record*. The archivist performs three different tasks to maintain *Evidence Records*:

- in an *initialization phase*, the archivist creates a Merkle tree from a group of data objects and includes a first timestamp to each of them,
- regularly, the archivist executes a *timestamp renewal*, in order to address the fading of the security timestamps,
- and also on a regular basis, the archivist executes a *Merkle tree renewal*, in order to address the fading of the security of hash functions.

We now describe each task in detail. In the initialization phase, the archivist selects a group of data objects and a hash function, constructs a Merkle tree whose leaves are the hash values of the data objects, and requests a timestamp on the tree's root. The archivist includes in each data objects' *Evidence Record* the following:

- the identifier of the hash function used to construct the Merkle tree,
- the authentication path for the corresponding leaf of the tree,
- and the timestamp issued on the Merkle tree's root.

Timestamp renewal of a previously issued timestamp works as follows. The archivist requests a new timestamp on the hash of the old timestamp. The archivist includes the new timestamp in each *Evidence Record* that contains the old timestamp.

[2] Actually, each timestamp is encapsulated in a structure called *Archive Timestamp*. We omit this detail to avoid confusion with PKIX timestamps.

ERS does not specify how the archivist manages validation evidence for timestamps' signatures, namely certificates and revocation status. It is a common practice to store validation evidence for the previous timestamp's signature before requesting the new timestamp. Not doing so adds an extra trust assumption on the system. Namely, that the archivist is trusted to check this evidence before requesting a new timestamp. We assume that the archivist collects and stores validation evidence for the previous timestamp's signature before requesting the new timestamp.

Merkle tree renewal of a previously constructed Merkle tree works as follows. The archivist identifies the data objects $d_0, \ldots, d_{n-1}$ that originated this Merkle tree. The archivist constructs a new Merkle tree whose leaves $y_0, \ldots, y_{n-1}$ are given by

$$y_i = h'(h'(d_i)||h'(R_i)) \ , \tag{1}$$

where $h'$ is a new hash function and $R_i$ is the *Evidence Record*[3] of the $i^{th}$ data object in its current state. As in the initialization phase, the archivist requests a timestamp on the tree's root, and includes in each data object's *Evidence Record*: the new hash function identifier, the authentication path for the corresponding leaf of the new Merkle tree, and the new root's timestamp.

It is the responsibility of the archivist to perform the corresponding renewals in a timely manner. That is, the archivist should execute a timestamp renewal before each timestamp becomes insecure, and a Merkle tree renewal before the hash function used to construct each Merkle tree becomes insecure.

In order to verify integrity and witnessed-existence of a data object, a retriever verifies each of the timestamps in the object's *Evidence Record*. Timestamps created in the initialization phase or by Merkle tree renewal refer to the Merkle tree's root. Thus, in order to verify such a timestamp, the retriever reconstructs the root using the authentication path, which is stored in the *Evidence Record*. Timestamps created by timestamp renewal, refer to the previous timestamp.

This procedure indeed proves integrity and witnessed-existence of the data object at any verification time, The first timestamp proves that the root of the Merkle tree existed since a time $t_0$ and identifies which hash function was used. The witnessed-existence of the root at time $t_0$, proves that the whole tree was constructed at that time. The authentication path proves that the data object was part of the tree. Suppose that the second timestamp is issued by timestamp renewal. In this case, the second timestamp is issued on the hash of the first timestamp. We are assuming that the timestamp contains evidence that shows that the TSA's key used to verify the first timestamp is trusted at time $t_1$. This two pieces of information constitute a proof that the root of the Merkle existed at time $t_0$. Therefore the second timestamp extends the proof of witnessed-existence of the data object beyond $t_1$, even if the first timestamp is no longer trusted after $t_1$. In general, the $n^{th}$ timestamp, issued by timestamp renewal, extends the proof of witnessed-existence of the data object beyond $t_{n-1}$, under the assumption that for every $i = 0, \ldots, n-1$, the $i^{th}$ timestamp was issued at time $t_i$ and it was trusted between $t_i$ and $t_{i+1}$.

Now suppose that at time $t_n$ the archivist runs a Merkle tree renewal. In this case, the $n^{th}$ timestamp proves that the root of the new Merkle tree existed since

[3] This is not exactly true. Only the sequence of all timestamps need to be hashed. The *Evidence Record* contains some extra information.

time $t_n$ and identifies which hash function was used. The witnessed-existence of the root at time $t_n$, proves that the whole tree was constructed at that time. The authentication path proves that the leaf was part of the tree. Recall that the leaf is given by (1). Hence it proves the witnessed-existence of the data object and its *Evidence Record* at time $t_n$. This constitutes a proof of the witnessed-existence of the data object at time $t_0$ provided that the hash function used to construct the *Evidence Record* was trusted between $t_0$ and $t_n$ and that for every $i = 0, \ldots, n-1$, the $i^{th}$ timestamp was issued at time $t_i$ and it was trusted between $t_i$ and $t_{i+1}$.

The procedure also guarantees long-term authenticity, provided that the data object was signed. In this case, before the archivist requests the first timestamp, he collects validation evidence (namely certificates and revocation status) for the data object's signature and includes this evidence in the signed data object.

There are two assumptions for ERS to guarantee long-term integrity, witnessed-existence and authenticity of a data object: a) that the underlying PKIs and TSAs are reliable; and b) that the cryptographic algorithms are not unexpectedly broken.

### 3.3 Auditing Control Environment

Auditing Control Environment (ACE) is a solution for long-term protection of digital data proposed by Song and JaJa in [54]. It provides long-term evidence of integrity and witnessed-existence for data objects. The solution produces two tiers of integrity evidence, allowing different levels of audits. The solution is based on Merkle trees (cf. Section 2.2) and wide-visible media (cf. Section 2.3).

We identify three players in this solution: an archivist, who preserves a set of data objects in the archive; an auditor who can verify integrity and witnessed-existence for data objects from the archive; and the so-called Integrity Management System (IMS), which is a third-party service provider that generates integrity information on the archivist's request.

The IMS and the archivist agree on a hash function. The archivist registers a data object in ACE by submitting the data object's digest to the IMS. The IMS collects a number of registration requests and creates an *integrity token* (IT) for each data object. The number of requests can be determined dynamically and it constitutes an *aggregation round*. The IMS creates a Merkle tree with the objects' digests of a round as its leaves. Each IT contains the object's digest, the time of registration and the corresponding authentication path in the Merkle tree. The IT is returned to the archivist, who stores it. The IMS stores the root of the Merkle tree. An IT and the root of the Merkle tree constitute the *first tier* of integrity information.

After a fixed time period, the IMS generates the *second tier* of integrity information. The IMS creates a Merkle tree whose leaves are digests of the first tier tree roots created during the time period. Figure 2 illustrates the two tier construction. Then, the IMS stores the root of this tree in a wide-visible medium. This root is the witness value of the state of the archive.

Before the hash function becomes insecure, the IMS and the archivist agree on a new hash function. The archivist reregisters in ACE each object that was registered with the previous hash function. To do so, the hash of the data object with the corresponding IT is calculated and submitted to the IMS.

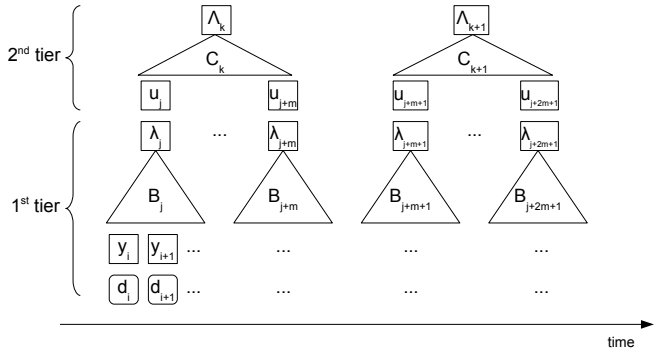


**Fig. 2** Integrity Evidence in ACE, where $d_i$ is a data object with digest $y_i$, $B_j$ is a first tier Merkle tree with root $\lambda_j$, $u_j$ is the digest of $\lambda_j$ and $C_k$ is a second tier Merkle tree with root $\Lambda_k$

.

ACE considers two kinds of audits, a first tier audit and a second tier audit. As a maintenance policy, the archivist carries out first tier audits on a regular basis. An audit can also be requested by a retriever of the archive to verify the integrity of a data object. Second tier audits can be triggered by a disagreement in a first tier audit or also as a maintenance policy.

A first tier audit works as follows. The auditor requests the data object and corresponding IT from the archivist. The auditor then computes a digest of the object and compares it with the digest stored in the IT. If the values are equal, the auditor requests from the IMS the first tier tree root corresponding to the time specified in the IT. Then, the auditor computes the root of the Merkle tree using the digest and authentication path stored in the IT and compares it to the value provided by the IMS. If the values are equal, the auditor accepts, otherwise, it rejects the object. If an object has been registered multiple times, the procedure is repeated for each IT.

Under the assumption that the IMS is reliable, it is easy to see that this procedure indeed proves integrity and witnessed-existence of the data object. Moreover, if the object is appropriately reregistered before the hash function becomes insecure, the first tier audit proves integrity and witnessed-existence of the data object at any verification time. The first IT proves the witnessed-existence and integrity of the data object from the time it is issued $t_0$ until the time $t_1$ when a second IT is issued. Because the first IT is attached to the object before reregistering it, the second IT extends the proof of witnessed-existence beyond $t_1$, even if the first IT is no longer trusted after $t_1$.

As we will see next, the second tier audit provides a proof of integrity that is independent of the IMS reliability. Moreover, it exposes any wrong doing of the IMS, thus providing an incentive for the IMS to act correctly.

In the second tier audit, we assume that the auditor has carried out the first tier audit. Then, the auditor request from the IMS the second tier authentication path corresponding to the first tier root. Using this path and the first tier root, the auditor computes the second tier Merkle tree root and compares it to the corresponding value stored in a wide-visible medium. If the values are equal, the auditor accepts, otherwise, it rejects the object.

Under the assumption that a wide-visible medium is trustworthy, it is easy to see that this procedure indeed proves integrity and witnessed-existence of the data object. Moreover, if the IMS fails to provide a second tier authentication path that yields the value stored in a wide-visible medium, then the auditor can conclude that the IMS provided incorrect values.

There are two assumptions for ACE to guarantee long-term integrity and witnessed-existence of a data object: a) that it is not possible to tamper with the wide-visible medium, and b) that the cryptographic algorithms are not unexpectedly broken.

### 3.4 Content Integrity Service

Content Integrity Service (CIS) is a solution for long-term protection of digital data proposed by Haber et al. in [35,36]. It provides long-term evidence of integrity and witnessed-existence for data objects. The solution allows objects to be transformed and produces an auditable record of every transformation. The solution is based on timestamps (cf. Section 2.6)

There are four main players in this solution: an archivist, who preserves a set of data objects in the archive; a retriever, who can verify integrity and witnessed-existence for data objects from the archive; Timestamp Authorities (TSAs) that issue timestamps on the archivist's request; and a submitter, who submits data objects to be archived and can request changes on them.

Much as in the ETSI standards described in Section 3.1, in CIS, the archivist maintains a sequence of timestamps for each data object that is used as evidence for its integrity and witnessed-existence. The first timestamp is issued on the data object and when the most recent timestamp is about to expire, the archivist requests a new timestamp on the data object together with all its previous timestamps.

The most important difference with ETSI standards is that CIS allows data objects to be transformed. Typical transformations that may occur are format conversions, annotations, additions of metadata, and later modifications of the document. When a submitter requests to change a data object, the archivist stores the modified data object, together with a description of the transformation. Then, the archivist requests a new timestamp on the concatenation of the old object, the new object, the transformation description and previous timestamps. The goal is to memorialize – and enable later verification of – the transformation, while preserving the assurance of integrity all the way back to that of the original form of the document.

The authors of CIS argue that PKIX timestamps are not suitable for long-term archiving. The reason is that the trustworthiness of these timestamps relies on digital signatures, whose signing keys are hard to protect over time. Alternatively, they propose to timestamp data objects using hash linking and wide-visible media. They defend that this technique is more reliable because the trustworthiness of timestamps depends solely on the collision resistance of hash functions.

The verification algorithm is analogous to that of ETSI. Following the analysis described in Section 3.1, it is easy to see that the procedure proves integrity and witnessed-existence of a data object at any verification time.

The authors of CIS implemented a prototype of CIS within HP's Digital Media Platform (DMP). This is an architecture for content storage and processing oper-

ations [35]. The solution presents an XML-based interface for service interaction, and a graph data model for documents. The authors of CIS claim that the DMP Repository Abstraction is convenient for handling the integrity metadata used by the CIS.

There are three assumptions for CIS to guarantee long-term integrity and witnessed-existence of a data object: a) that TSAs are reliable; b) that the wide-visible media are trustworthy; and c) that the cryptographic algorithms are not unexpectedly broken.

### 3.5 Lot of Copies Keep Stuff Safe

Lot of Copies Keep Stuff Safe (LOCKSS) is a solution for protection of digital data described in [45]. It guarantees integrity for archived data objects. The solution consists of establishing a peer-to-peer network of archives with replicated content. On a regular basis, archives communicate with each other to compare copies of the same content. They use a voting protocol to identify corrupted copies.

LOCKSS was developed for libraries to preserve access to journals and other archival information published on the Web. In this context, it is assumed that retrievers of a particular library trust their library. LOCKSS does not provide any evidence of integrity for data objects to retrievers. Instead, it is used by legitimate libraries to maintain reliable copies of their digital content.

The overall goal of LOCKSS design is that with high probability legitimate archivists have a correct version of their content despite failures and attacks, and with low probability, even a powerful adversary can damage a significant proportion of the legitimate archives without detection.

LOCKSS works as follows. Archivists establish a peer-to-peer network with other archives. On a regular basis, peers vote on large archival units (AUs), normally a year's run of a journal. A LOCKSS peer invites into its poll a number of peers, hoping they will offer votes on their version of the AU. An invited peer computes a fresh digest of its own version of the AU, which it returns in a vote. If the caller of the poll receives votes that overwhelmingly agree with its own version of the AU (a landslide win), it is satisfied. If it receives votes that overwhelmingly disagree with its own version of the AU (a landslide loss), it repairs its AU by fetching the copy of a voter who disagreed, and invites peers to vote on the new AU. This process is repeated until landslide wins. If the result of the poll justifies neither a landslide win nor a landslide loss (an inconclusive poll), then the caller raises an alarm to attract human attention to the situation.

We present here a simplified version of LOCKSS opinion poll protocol for illustration. Archivist $A$ wants to verify the integrity of its copy of an object $d_i$, where $i$ is the unique identification for $d$ in LOCKSS. Archivist $A$ chooses a hash function $h$, generates a nounce $N_A$ and sends a *poll* request $(i, h, N_A)$ to a number of archives that have copies of the object. When an archive $B$ receives the request, it generates a nounce $N_B$, computes $y_B = h(N_A \| N_B \| d'_i)$, where $d'_i$ is its copy of the object, and sends the response $r_B = (N_B, y_B)$ to $A$. Then $A$ evaluates $y'_B = h(N_A \| N_B \| d_i)$ and compares $y_B$ with $y'_B$. Archivist $A$ does the same for each response and if the majority of comparisons succeed, then $A$ considers its local copy $d_i$ uncorrupted. Otherwise, $A$ replaces his local copy $d_i$ by a copy $d'_i$ from a peer that answered the poll and starts the protocol again.

The detailed version of the protocol (see [45]), includes some additional features:

- The poller chooses two sets of poll participants, an inner circle of trusted peers and a outer circle of new peers. Only the inner circle is used for deciding on the AU. By polling the outer circle, the poller tests the reliability of newcomers for expanding its set of trusted peers.
- A confidential channel between archivists is established using a key exchange protocol before starting the protocol.
- LOCKSS requires that pollers as well as voters expend provable computational effort [25], that is, they are required to prove that they spent computational effort proportional to the underlying protocol operation (hashing of an AU).

The protocol indeed protects the integrity of the archive. The simplified protocol allows legitimate archivists to detect damage of a data object, and to repair it. This is true only if there are enough trusted peers with healthy copies of the data object. In order to defend the system from an attack, LOCKSS makes it costly for an adversary to sway an opinion poll in his favor or to waste peers' resources by requiring provable computational effort. Archivists also defend from malicious peers by only considering previously tested peers. Finally, by raising an alarm when a peer determines that a poll is inconclusive, LOCKSS makes it likely that an attack is detected before it progresses.

There are two assumptions for LOCKSS to guarantee long-term integrity of a data object, a) that the archivist that serves the retriever is reliable; b) that the majority of peers are run by reliable archivists; and c) that an archivist can find enough reliable peers with copies of its content.

LOCKSS is not vulnerable to an unexpected break of a cryptographic algorithm because no evidence of integrity is built. Moreover, the absence of integrity proofs allows LOCKSS to address format obsolescence of archived data objects.

### 3.6 Optimized Certificates

Optimized Certificates (OCs) is a solution proposed by Custódio et. al. in [23] that provides evidence of long-term integrity and witnessed-existence for signed data objects. This evidence extends the trustworthiness of a data object's signature beyond the lifetimes of certificates, timestamps and cryptographic algorithms, thereby ensuring long-term authenticity.

In this solution, the original validation evidence for a data object's signature (namely certificates, revocations status, and PKIX timestamps) is *summarized* by a single attestation. A reliable party produces such attestation, therewith asserting that: a) the signer's certificate is valid to verify the data object's signature; b) the signed data object is uncorrupted; and c) the existence of the data object's signature was witnessed. The attestation is called *Optimized Certificate* and is almost a copy of the data object signer's certificate. The OC has the structure of an X.509 certificate to keep compatibility with existing Public Key Infrastructures. The OC contains the signer's public key and name. Its validity period is the moment the reliable party issued the OC, therefore it is not revocable. The OC includes the data object's signature and digest, and a date on which these data was witnessed. These data is structured as an X.509 Extension. Thus, the OC works as a timestamp for

the signed data object. The reliable parties are assigned with a traditional X.509 certificate, which is issued by a Root CA. This certificate can be revoked.

There are three players in this solution: an archivist, who preserves a set of data objects in the archive; the notaries, which are the reliable parties that issue OCs on the archivist's request; and a retriever, who can verify integrity, witnessed-existence and authenticity for data objects from the archive using OCs.

The archivist initially replaces the validation evidence for a data object' signature by an OC. For each signed data object, the archivist requests an OC from a notary by submitting the signed data object's digest and signature, the signer's certificate chain and revocation status. If there are timestamps, they are submitted with the TSAs' certificate chains and revocation status. The notary returns an OC. The archivist replaces the validation evidence available in the signed data object by the OC and the notary's certificate.

On a regular basis, the archivist monitors OC's trustworthiness for each signed data object. An OC becomes untrustworthy if the cryptographic algorithms get insecure or the notary's certificate expires. Before an OC becomes untrustworthy, the archivist renews it. The archivist submits to a notary the existing OC, the corresponding issuer's certificate and the signed data object's digest, which is calculated with a secure cryptographic algorithm. The notary returns a new OC. The archivist replaces the old OC and the corresponding issuer's certificate by the new OC's and the notary's certificate.

A notary receives OC requests, which comprise the signed data object's digest and either a) certificate chains, revocation status, PKIX timestamps, and the data object's signature or b) an OC and its issuer's certificate. The notary verifies whether certificates are valid and the cryptographic algorithm are still secure. If the verification succeeds, the notary creates an OC. He sets the OC's public key to be the the data object signer's public key. He defines the OC's validity period to be the current time. He inserts into the OC a witnessed-existence evidence that comprises: a) the data object's signature, which was either directly submitted by the archivist or is taken from the submitted OC; b) the signed data object's digest; and c) a date, which is taken either from a submitted timestamp, from the submitted OC, or is the current time. The notary signs the OC and returns it.

Given a signed data object, the OC and the notary's certificate, the retriever checks authenticity by verifying if: a) the signature is valid under the public key in the OC; b) the OC's signature is valid under the public key in the notary's certificate; c) the OC's validity period is within the validity period of the notary's certificate; d) the signature on the notary's certificate is valid under the public key in Root CA's certificate; and e) the notary's certificate is neither expired nor revoked. The retriever checks integrity and witnessed-existence by verifying if the signed data object's digest and signature equals the digest and signature included in the OC.

Note that the size of validation evidence is constant in the long term. This is because no further data is added, rather validation evidence is replaced by new objects created with up to date cryptographic algorithms and certificates.

This solution indeed ensures authenticity, integrity and witnessed-existence for a signed data object if a retriever trusts all involved notaries. Trust is required because the archivist discards the original validation evidence, which therefore cannot be verified in the future. Thus, given an OC issued at $t_1$, a retriever infers that the signer's certificate was valid at $t_1$, since a trusted notary *properly* verified

it at $t_1$. Like the signer's certificate, the retriever infers the validity for the OCs. That is, for every $i = 2, \ldots, n$, given the $i^{th}$ OC issued at $t_i$, the retriever infers that the $i - 1^{th}$ OC was valid at $t_i$. Thus, if the retriever can verify the validity of the current OC, i.e. the $n^{th}$ at $t_n$, then he infers that the $1^{st}$ OC issued at $t_1$ was valid. Consequently, the retriever infers that the signer's certificate was valid at $t_1$.

There are three assumptions for the OC solution to guarantee long-term integrity, witnessed-existence and authenticity of a data object: a) that the underlying PKIs and TSAs are reliable; b) all notaries are reliable; and c) that the cryptographic algorithms are not unexpectedly broken.

## 4 Goals and Means

We have surveyed a number of solutions that provide long-term protection for data objects. We now compare the solutions on three aspects: a) the security goals they achieve, b) the assumptions that need to be made in order to achieve such goals, and c) their support for format migration. Tables 1 and 2 summarize our findings.

**Table 1** The classification of surveyed solutions in regard to achieved security goals and required trust assumptions

| | Security Goals | | | Trust Assumptions | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Solution | Integrity | Witnessed-existence | Authenticity | PKI | TSA | No unexpected break | Wide-visible media | Reliable actors | External copies |
| ETSI | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| ERS | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | | |
| ACE | ✓ | ✓ | | | | ✓ | ✓ | | |
| CIS | ✓ | ✓ | | | | ✓ | ✓ | | |
| LOCKSS | ✓ | | | | | | | ✓ | ✓ |
| OC | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | |

The security goals assessed in this survey are long-term integrity, authenticity and witnessed-existence. We identify integrity as the minimum goal for long-term protection in digital archives. Long-term evidence of witnessed-existence is a stronger goal in the sense that it implies integrity. It is achieved by all solutions except LOCKSS. Three of the solutions surveyed (ETSI, ERS and OC) provide long-term authenticity. In all three cases authenticity is achieved in the same way. Data objects are stored together with a digital signature so an evidence of witnessed-existence of the signed data objects proves authenticity. ACE and CIS provide evidence of witnessed-existence. They do not provide authenticity under the given trust assumptions. Evidence of authenticity could easily be provided, by signing data-objects and a PKI that certifies public keys.

The different solutions surveyed make a number of assumptions in order to guarantee long-term protection. We identified them as follows.

- **PKI:** there are Public Key Infrastructures in place that are reliable.
- **TSA:** there are reliable Timestamp Authorities.
- **No unexpected break:** cryptographic algorithms are not unexpectedly broken.
- **Wide-visible media:** there exist media available, where published data is widely witnessed.
- **Reliable actors:** there are reliable actors other than trusted CAs and TSAs.

– **External copies:** the archivist can find enough reliable peers with copies of data objects. The number of peers can be regarded as a parameter.

In order to analyze the solutions we group them according to their assumptions. ETSI and ERS form one group, ACE and CIS form a second group, while OC and LOCKSS are analyzed separately.

The ETSI and ERS solutions require the same assumptions. They rely heavily on digital signatures thus the need for PKIs. Both rely on TSAs to provide timestamps. ERS uses Merkle trees to improve efficiency. Although cryptographic information is renewed, it is assumed that cryptographic primitives are stable in the sense that they are not unexpectedly broken.

OC requires similar assumptions as ETSI and ERS. However, OC assumes notaries as additional reliable actors. The aim at improving efficiency if compared to solutions that accumulate evidence. For instance, ETSI, ERS, CIS and ACE.

ACE and CIS rely mainly on wide-visible media for long-term protection. They publish regularly a witness value on wide-visible media. The witness value can be regarded as a commitment by the archivist to a state of the archive. They use hash functions in order to make witness values small. Although they address the fading of the hash functions, it is assumed that the hash functions are not unexpectedly broken. The wide-visible media provide witnessed-existence in rather coarse time intervals (e.g. one week or one month). In order to distinguish finer grained time intervals, additional trust assumptions are required. ACE in fact suggests using TSAs.

LOCKSS relies heavily on reliable actors, the archivists themselves. Additionally it is assumed that the archivist can compare its data objects with enough reliable peers.

Format obsolescence is not the focus of this survey. However, it is important for archivists particularly in the long term. Format migration requires to alter content, thus it compromises integrity. From the solutions surveyed, ETSI, ERS and OC do not provide support for format migration as shown in Table 2.

**Table 2** The classification of surveyed solutions in regard to the support to format migration of data objects

| Solution | Format Migration Support |
|---|---|
| ETSI | |
| ERS | |
| ACE | ✓ |
| CIS | ✓ |
| LOCKSS | ✓ |
| OC | |

There is an important discussion about the choice of trust assumptions that can plausibly hold in the long term. The two main options are to use digital signatures or to publish witness values. Digital signatures have the advantage of a mathematical proof to support protection claims. However, digital signatures require maintenance of public keys and thus PKIs. Some of the authors surveyed oppose digital signatures arguing that such infrastructures are impractical in the long term. The alternative is to publish witness values in wide-visible media, such

as a newspaper. It is unclear to us how reliable this alternative is in the long term. For a wide-visible media to guarantee a long-term witness, it would need to be reliably archived, thus shifting the protection problem from data objects to copies of the wide-visible media. Sometimes such media is of a physical nature (e.g. a printed newspaper) raising concerns about material deterioration and interoperability with digital solutions.

Reliable actors are used to avoid other assumptions or to remove evidence. However, a reliable actor is a further failure point in the protection of data objects.

## 5 Efficiency

In this section we compare the different solutions with respect to storage overhead. Additionally, we briefly analyze the archivist's effort to maintain evidence in each solution. This comparison is not conclusive because the surveyed sources not always provide enough details about efficiency.

ETSI produces a significant storage overhead. For each data object, the system stores a chain of timestamps that grows over time. In addition, the system stores validation evidence for each timestamp's signature. Troncoso et al. argue in [58] that this overhead is affordable, based on Moore's [48] and Kryder's [60] laws. The archivist has to maintain a timestamp chain for each data object, adding new timestamps whenever necessary. According to [39], ETSI standards "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".

ERS requires roughly the same storage overhead as ETSI. The Merkle tree construction allows for a single timestamp to protect several data objects. However, the same timestamp is replicated in each data object. The archivist needs to request fewer timestamps if compared to ETSI. On the other hand, the archivist needs to maintain the Merkle trees. New data objects cannot be inserted as leaves of an existing and timestamped Merkle tree. Therefore, a new Merkle tree has to be created and timestamped. As a consequence, in a long-term scenario there would be multiple Merkle trees that will require regular and individual renewals of timestamps and Merkle trees.

CIS offers some advantages in storage overhead over ETSI and ERS. Despite similarities with ETSI, the use of timestamps based on wide-visible media has the advantage that there is no need to store validation evidence. For a single data object, CIS demands less timestamps than the ETSI and ERS solutions. The reason is that timestamps in CIS rely only on the security of hash functions and, therefore, last longer than PKIX timestamps. However, in terms of management, both the CIS and ETSI solutions require similar effort by the archivist. The archivist needs to maintain a chain of timestamps for each data object.

ACE requires less storage overhead than ETSI and ERS. The evidence of integrity stored in a single IT is presumably smaller than a PKIX timestamp because there is no need to store validation evidence for signatures (namely certificates and revocation status). Moreover, one would expect fewer ITs for a single object than PKIX timestamps in ETSI and ERS. This is because IT relies only on hash functions. The witness information that is stored in a wide-visible media is also relatively small. The authors of ACE estimate that, if IMS produces one witness value per day, the total size of all the witness values over a year is around 100 KB.

The archivist's effort to maintain the ITs for each data object is similar to ERS and CIS. There is, however, an extra burden for IMS, which stores Merkle tree roots for each round. The authors implemented a prototype and show the results in [54].

LOCKSS keeps no evidence of integrity thus carries no storage overhead. However, archives must invest considerable computational resources to run polls on a regular basis. The authors of LOCKSS claim that acceptable performance can be achieved at a low cost. They estimate in [45] that archives can maintain 210 journals, at a hardware cost of 5 US$ per journal/year, checking each journal every 3 months against 17 peers.

OC allows for constant storage overhead in the long term, being more efficient than other surveyed solutions. This is because the old objects in validation evidence are replaced by new objects using up to date cryptographic algorithm and certificates. In terms of management, OC is equal to ETSI, CIS and ACE. That is, the archivist needs to monitor each data object's OC and request a new one from a notary in a timely manner.

## 6 Solutions that fail to provide Long-term Protection

In this section we present a number of solutions that fail to provide long-term authenticity, integrity and witnessed-existence. We also briefly explain why they fail. This list is not intended to be exhaustive. For further examples we refer the reader to [56].

Authentic Timestamps [52] is a solution for archival storage. Besides integrity and witnessed-existence it also produces evidence of *witnessed-non-existence* for data objects. The authors introduce a data structure that implements an append-only, persistent, authenticated dictionary by combining ideas from Merkle and Patricia trees. The solution uses PKIX timestamps (Section 2.6) for witnessed-existence. The scheme does not address the weakening of hash algorithms or timestamp signatures hence it does not provide long-term protection.

Key Archive Service and Timestamp (KASTS) is an archiving system proposed by Maniatis and Baker in [44]. The system aims at extending the validity of digital signatures beyond the expiration date of its certificate. It does so by timestamping a signature at the time it is produced and archiving old signature verification keys. Verification keys are stored by a trusted third party named *Key Archive Service*. Validation evidence is structured in authenticated search trees [18] and only the roots of the trees are timestamped. KASTS keeps only the most recent timestamp, thus a retriever lacks evidence to evaluate the chronological order of authenticated search trees. KASTS does not address the weakening of the hash algorithm used to construct authenticated search trees. Therefore, KASTS does not provide long-term protection.

Certified Accountable Tamper-evident Storage (CATS) is a network storage system proposed by Yumerefendi and Chase in [62]. It provides evidence of witnessed-existence, witnessed-non-existence, integrity and authenticity for data objects. The evidence is based on authenticated search trees, digital signatures (Section 2.4) and wide-visible media (Section 2.3). The focus of CATS is on providing means of accountability for the players of an archival solution. A CATS server maintains a shared directory of data objects where users can read and write. Clients can verify

that the server is faithful and cannot deny their operations. CATS does not address the weakening of hash algorithm or signature keys, hence it does not provide long-term protection.

Maniatis et al. propose in [31] to build a network of Timestamp Authorities called Prokopius. They use a peer-to-peer archival storage network as their publication medium for grounding values. This allows clients to verify timestamps issued by formerly trusted TSAs, even if they are no longer in service. Such TSA can be used by an archivist to provide protection for data objects. The solution is based on timestamps, Merkle trees and Byzantine Fault Tolerant protocol [4]. Although Prokopius increases the protection of traditional timestamps, it does not address long-term protection as defined in this survey. In particular it does not deal with the weakening of hash algorithms, which will compromise the evidences of authenticity and witnessed-existence for timestamps.

Victorian Electronic Record Strategy (VERS) is a solution for long-term protection of digital data developed by a team commissioned by the Australian government [57, 61]. VERS is a prototype intended to test archiving techniques to maintain government records. The solution provides evidence of long-term integrity and authenticity for data objects. The solution is based on digital signatures (Section 2.4). An important design decision of VERS is not to use CA certificates (Section 2.5). The authors argue that CAs are likely to disappear during the archival period of data objects and, therefore, it is better not rely on certificates. Without certificates and revocation data, there is no proof that binds the signer's identification and public key. Alternatively, the authors suggest to confirm the binding between signer's identification and public key by: a) comparing the public key with other records signed by the same signer, and b) the archivist maintaining a signed certification record that binds the identification of an authorized user in VERS and corresponding public keys. In either case, long-term protection relies solely on trust on the archivist with no accountability. We do not consider this to be a solution for long-term protection.

## 7 Conclusions and Future Work

We surveyed solutions for long-term protection of digital data, describing and comparing them in regard to their protection goals and trust assumptions. A brief comparative analysis of the solutions' efficiency was also provided. Additionally, we identified a number of archival solutions that fail to provide long-term protection, and showed why they fail.

A number of open problems arise from our analysis. There are desirable properties missing from long-term protection. One example is evidence of witnessed-nonexistence. Trust assumptions can be reduced or adjusted to satisfy particular requirements of an archive. Rigorous analysis of long-term protection is possible using security models recently developed [49, **?**]. To the best of our knowledge there is no comprehensive analysis available of the efficiency of archival solutions. This is particularly important in the long term, in order to determine whether a solution is scalable for specific applications.

[4] The Byzantine failure assumption [42] models systems whose components may behave in unexpected ways due to hardware failures, network congestion or malicious attacks. Byzantine failure-tolerant algorithms must cope with such failures.

Publishing witness values on wide-visible media has been proposed as an alternative to digital signatures for timestamping. However, it is unclear if the wide-visible media assumption is realistic. Unlike PKI, there is no established infrastructure that provides assurance for wide-visible media. The preservation of validation evidence based on a physical wide-visible media, e.g. a newspaper, leads archivists back to old issues such as physical space and deterioration. An interesting question that arises from TSAs and wide-visible media is whether reliable dating indeed needs a trusted party.

Further research can aim at quantifying and comparing trust in the surveyed solutions. To do so, the methodology proposed by Huang and Nicol in [38] can be used. An interesting question that may be answered is whether the trust in solutions that use notaries and those that do not use is equal in the long term. If so, then it will be an incentive to use OC instead of other solutions, thereby reducing overhead of storage.

CIS and ACE stand as the best surveyed solutions in regard to security goals, trust assumptions, efficiency, and format migration together. Both solutions address all the surveyed protection goals and require less trust assumptions than other solutions. In terms of efficiency, the storage overhead in CIS and ACE is lower than in other solutions, except from OC. We exclude OC because it uses a notarial approach, hence having stronger trust assumptions. For the management of evidence, a detailed analysis is necessary to fairly compare CIS and ACE with ERS. Format migration is supported by both solutions.

It is generally perceived that archivists and cryptographers disagree on protection means for archives. Archivists play the role of a trusted party while cryptographers replace trusted parties with mathematical assumptions. Archivists demonstrate authenticity or integrity using the information conveyed by a data object, therefore they use procedural controls and documentation. Cryptographers, on the other hand focus on the digital codification of a data object, thus they use mathematical algorithms on data objects' bits. Thus, a promising research would consider harmonizing archivists' and cryptographer's approaches by balancing trust between cryptographic evidences and trusted parties. In this direction, notarial approaches appear as the right choice.

**Acknowledgements** The authors thank *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES/Brazil) for partially supporting this work.

## References

1. American Memory from the Library of Congress - Home Page. URL `http://memory.loc.gov/ammem/index.html`
2. case studies. URL `http://www.interpares.org/ip2/ip2_case_studies.cfm`
3. Centera Compliance Edition Plus. URL `http://www.emc.com/products/detail/hardware/centera-compliance-edition-plus.htm`
4. Council Directive 2001/115/EC of 20 December 2001 amending Directive 77/388/EEC with a view to simplifying, modernising and harmonising the conditions laid down for invoicing in respect of value added tax. Official Journal L 015 , 17/01/2002 P. 0024 - 0028
5. Home - Kungliga biblioteket
6. Home - Library and Archives Canada
7. Records Management: NHS Code of Practice Part 2. URL `http://www.dh.gov.uk/prod_consum_dh/groups/dh_digitalassets/documents/digitalasset/dh_093027.pdf`

8. THE BRITISH LIBRARY - The world's knowledge
9. VAT in the European Community - Slovenia. URL http://ec.europa.eu/taxation_customs/resources/documents/taxation/vat/traders/vat_community/dec2007/vat_ec_si_en.pdf
10. Retention and Disposal Policy for Patent Related Records (2011). URL http://www.ipo.gov.uk/p-retentiondisposal.pdf
11. eDavki elektronsko davčno poslovanje (2012). URL http://edavki.durs.si
12. EPELFI :: Livre Foncier - Accueil (2012). URL https://www.livrefoncier.fr
13. Adams, C., Cain, P., Pinkas, D., Zuccherato, R.: Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP). RFC 3161 (Proposed Standard) (2001). URL http://www.ietf.org/rfc/rfc3161.txt. Updated by RFC 5816
14. Bayer, D., Haber, S., Stornetta, S.: Improving the efficiency and reliability of digital time-stamping. In: Sequences II: Methods in Communication, Security, and Computer Science (Proceedings of the Sequences Workshop, 1991), pp. 329–334. Springer-Verlag, New York (1993)
15. Bayer, D., Haber, S., Stornetta, W.: Improving the efficiency and reliability of digital time-stamping. Sequences II: Methods in Communication, Security, and Computer Science pp. 329–334 (1993)
16. Benaloh, J., de Mare, M.: E cient broadcast time-stamping. Clarkson University Department of Mathematics and Computer Science TR pp. 91–1 (1991)
17. Braun, J., Buchmann, J., Mullan, C., Wiesmaier, A.: Long term confidentiality: a survey. Designs, Codes and Cryptography (2012). Accepted for publication, to appear. Preliminary version: Cryptology ePrint Archive, Report 2012/449
18. Buldas, A., Laud, P., Lipmaa, H.: Accountable certificate management using undeniable attestations. In: Proceedings of the 7th ACM conference on Computer and communications security, pp. 9–17. ACM (2000)
19. Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen: Bekanntmachung zur elektronischen Signatur nach dem Signaturgesetz und der Signaturverordnung (Übersicht über geeignete Algorithmen) (2010). URL http://www.bundesnetzagentur.de/cae/servlet/contentblob/148572/publicationFile/3994/2010AlgoKatpdf.pdf
20. Bundesregierung: Verordnung über den Schutz vor Schäden durch Röntgenstrahlen (Röntgenverordnung - RöV) (1987). URL http://www.gesetze-im-internet.de/bundesrecht/r_v_1987/gesamt.pdf
21. Bundesärztekammer: (Muster-)Berufsordnung für die in Deutschland tätigen Ärztinnen und Ärzte (Stand 2011) (2011). URL http://www.bundesaerztekammer.de/page.asp?his=1.100.1143
22. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard) (2008). URL http://www.ietf.org/rfc/rfc5280.txt
23. Custódio, R.F., Vigil, M.A.G., Romani, J., Pereira, F.C., da Silva Fraga, J.: Optimized Certificates - A New Proposal for Efficient Electronic Document Signature Validation. In: S.F. Mjø lsnes, S. Mauw, S.K. Katsikas (eds.) EuroPKI, *Lecture Notes in Computer Science*, vol. 5057, pp. 49–59. Springer (2008)
24. Duranti, L.: Continuity and transformation in the role of the archivist: The findings of the interpares project. Journal of the Japan Society for Archival Science **3**(6), 27–41 (2007)
25. Dwork, C., Naor, M.: Pricing via processing or combatting junk mail. In: E. Brickell (ed.) Advances in Cryptology - CRYPTO 92, *Lecture Notes in Computer Science*, vol. 740, pp. 139–147. Springer Berlin / Heidelberg (1993)
26. Estonian Centre of Registers and Information Systems: Registrite ja infosüsteemide keskus - What is Land Register? URL http://www.rik.ee/land_register
27. ETSI: XML Advanced Electronic Signatures (XAdES) (2006)
28. ETSI: CMS Advanced Eletronic Signatures (CAdES) (2007)
29. Foscarini, F.: Cultures of trust: Legal, technical, and archival perspectives on the use of digital signature technologies. In: GI Jahrestagung (1), pp. 37–47 (2008)
30. Gamal, T.E.: On computing logarithms over finite fields. In: H.C. Williams (ed.) CRYPTO, *Lecture Notes in Computer Science*, vol. 218, pp. 396–402. Springer (1985)
31. Giuli, P., Maniatis, P., Giuli, T., Baker, M.: Enabling the Long-Term Archival of Signed Documents through Time Stamping. In: Proceedings of the 1st USENIX Conference on File and Storage Technologies, FAST '02, pp. 31–46. USENIX Association (2002). URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.16.361

32. Goldreich, O.: Foundations of Cryptography: Volume 2, Basic Applications. Cambridge Univ Pr (2009)
33. Gondrom, T., Brandner, R., Pordesch, U.: Evidence Record Syntax (ERS) (2007). URL http://www.ietf.org/rfc/rfc4998.txt
34. Haber, S.: Content Integrity Service for Long-Term Digital Archives. In: Archiving 2006, pp. 159–164. IS&T, Ottawa, Canada (2006)
35. Haber, S., Kamat, P., Kamineni, K.: A content integrity service for digital repositories (2008)
36. Haber, S., Stornetta, W.S.: How to time-stamp a digital document. Advances in Cryptology Crypto 90 **3**(2), 437–455 (1991). DOI 10.1007/BF00196791. URL http://www.springerlink.com/content/x771v2341pw02711
37. Huang, J., Nicol, D.: A calculus of trust and its application to pki and identity management. In: Proceedings of the 8th Symposium on Identity and Trust on the Internet, pp. 23–37. ACM (2009)
38. Huehnlein, D., Korte, U., Langer, L., Wiesmaier, A.: A Comprehensive Reference Architecture for Trustworthy Long-Term Archiving of Sensitive Data. In: New Technologies, Mobility and Security (NTMS), 2009 3rd International Conference on, pp. 1–5 (2009). DOI 10.1109/NTMS.2009.5384830
39. Johnson, D., Menezes, A., Vanstone, S.A.: The elliptic curve digital signature algorithm (ecdsa). Int. J. Inf. Sec. **1**(1), 36–63 (2001)
40. Kohnfelder, L.M.: Towards a practical public-key cryptosystem. Tech. rep., Massachusetts Institute of Technology (1978)
41. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Transactions on Programming Languages and Systems (TOPLAS) **4**(3), 382–401 (1982)
42. Lenstra, A.K., Verheul, E.R.: Selecting cryptographic key sizes. J. Cryptology **14**(4), 255–293 (2001)
43. Maniatis, P., Baker, M.: Enabling the Archival Storage of Signed Documents. In: Proceedings of the 1st USENIX Conference on File and Storage Technologies, FAST '02. USENIX Association, Berkeley, CA, USA (2002). URL http://portal.acm.org/citation.cfm?id=1083323.1083327
44. Maniatis, P., Roussopoulos, M., Giuli, T.J., Rosenthal, D.S.H., Baker, M.: The LOCKSS peer-to-peer digital preservation system. ACM Transactions on Computer Systems **23**(1), 2–50 (2005). DOI 10.1145/1047915.1047917. URL http://portal.acm.org/citation.cfm?doid=1047915.1047917
45. Menezes, A.J., Vanstone, S.A., Oorschot, P.C.V.: Handbook of Applied Cryptography, 1st edn. CRC Press, Inc., Boca Raton, FL, USA (1996)
46. Merkle, R.C.: A Certified Digital Signature. In: G. Brassard (ed.) Advances in Cryptology - CRYPTO '89, *Lecture Notes in Computer Science*, vol. 435, pp. 218–238. Springer (1989). DOI http://dx.doi.org/10.1007/0-387-34805-0_21
47. Moore, G., et al.: Cramming more components onto integrated circuits. Proceedings of the IEEE **86**(1), 82–85 (1998)
48. Müller-Quade, J., Unruh, D.: Long-term security and universal composability. In: S. Vadhan (ed.) Theory of Cryptography, *Lecture Notes in Computer Science*, vol. 4392, pp. 41–60. Springer Berlin / Heidelberg (2007)
49. Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560 (Proposed Standard) (1999). URL http://www.ietf.org/rfc/rfc2560.txt. Updated by RFC 6277
50. National Institute of Standards and Technology: Recommendation for Key Management – Part 1: General (Revised) (2007). URL http://csrc.nist.gov/groups/ST/toolkit/documents/SP800-57Part1_3-8-07.pdf
51. Oprea, A., Bowers, K.D.: Authentic Time-Stamps for Archival Storage (2009)
52. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2), 120–126 (1978)
53. Song, S., JaJa, J.: Techniques to audit and certify the long-term integrity of digital archives. International Journal on Digital Libraries **10**(2-3), 123–131 (2009). DOI 10.1007/s00799-009-0056-2. URL http://www.springerlink.com/index/10.1007/s00799-009-0056-2
54. Stevens, M., Sotirov, A., Appelbaum, J., Lenstra, A., Molnar, D., Osvik, D., de Weger, B.: Short chosen-prefix collisions for MD5 and the creation of a rogue CA certificate. In: S. Halevi (ed.) Advances in Cryptology - CRYPTO 2009, *Lecture Notes in Computer Science*, vol. 5677, pp. 55–69. Springer Berlin / Heidelberg (2009)

55. Storer, M.W., Greenan, K., Miller, E.L.: Long-term threats to secure archives. In: Proceedings of the second ACM workshop on Storage security and survivability, pp. 9–16. ACM (2006)
56. Team, V.E.R.S.P., Office, V.P.R.: Victorian Electronic Records Strategy Final Report, vol. 54. Public Record Office Victoria (1998). URL http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Victorian+Electronic+Records+Strategy+-+Final+Report#0
57. Troncoso, C., De Cock, D., Preneel, B.: Improving secure long-term archival of digitally signed documents. In: Proceedings of the 4th ACM international workshop on Storage security and survivability, StorageSS '08, pp. 27–36. ACM, New York, NY, USA (2008). DOI 10.1145/1456469.1456476. URL http://doi.acm.org/10.1145/1456469.1456476
58. Wallace, C., Pordesch, U., Brandner, R.: Long-Term Archive Service Requirements. RFC 4810 (Informational) (2007). URL http://www.ietf.org/rfc/rfc4810.txt
59. Walter, C.: Kryder's law. Scientific American **293**(2), 32 (2005)
60. Waugh, A., Wilkinson, R., Hills, B., Dell'oro, J.: Preserving Digital Information Forever. In: Proceedings of the fifth ACM conference on Digital libraries, pp. 175–184. ACM, San Antonio, Texas, United States (2000)
61. Yumerefendi, A.R., Chase, J.S.: Strong accountability for network storage. Trans. Storage **3**(3) (2007). DOI http://doi.acm.org/10.1145/1288783.1288786. URL http://doi.acm.org/10.1145/1288783.1288786