Improving Secure Long-Term Archival of Digitally Signed Documents

Carmela Troncoso Danny De Cock Bart Preneel
K.U.Leuven, ESAT/COSIC-IBBT,
Kasteelpark Arenberg 10,
B-3001 Leuven-Heverlee, Belgium.
firstname.lastname@esat.kuleuven.be

ABSTRACT

Long-term archival of signed documents presents specific challenges that do not need to be considered in short-term storage systems. In this paper we present a Secure Long-Term Archival System (SLTAS) that protects, in a verifiable way, the validity of today's digital signatures in a distant future. Moreover, our protocol is the first proposal that provides a proof of when a signature was created, without the possibility of backdating. We include a description of our scheme and an evaluation of its performance in terms of computing time and storage space. Finally, we discuss how to extend our system to achieve additional security properties. This paper does not focus on the long-term availability of archived information. nor on format migration problems.

Categories and Subject Descriptors

H.3.0 [**Information Storage and Retrieval**]: General

General Terms

Design, Security, Reliability

Keywords

Secure long-term archiving, Digitally signed documents, Retimestamping

1. INTRODUCTION

Nowadays many documents are created as (or transformed into) digital records and plenty of electronic services (e-government, e-commerce, stock exchange, etc.) depend on them. This requires digital repositories where this information can be stored and accessed in a reliable and secure way [34]. Traditionally, archival systems aim at ensuring integrity and availability, but even if an archive initially provides these properties, in the course of time the integrity will certainly degrade.

This paper is a step forward on the storage of digitally signed documents and on how to preserve the eternal validity of their digital signatures (and thus provide eternal protection of the integrity of the documents). Secure long-term archival of signed documents suffers from the fragile nature of the private signing keys and the validity status of the corresponding public verification key certificates. The algorithms we use today to compute digital signatures will show vulnerabilities in the long run due to progress in cryptology and the increasing computing power [36]. In order to preserve the validity of signatures, a Secure Long-Term Archival System (SLTAS) must implement a mechanism able to provide proof that the following requirements were checked at a particular time, namely before the document was archived:

- the document was signed between two indisputable moments in the past,
- the content of the signed document has not changed during the storage period, and
- the signature was generated with a signing key that was valid when it was created.

The first requirement guarantees that the signing time cannot be shifted before or after the two indisputable moments in the past, a property which is indispensable in some applications. Consider the scenarios of electronic stock market transactions, or betting systems. In these cases, allowing backdating of an order's signature may open the door to fraud which can have an important monetary impact.

Meeting the last requirement ensures the eternal validity of the signed document, as the revocation status of the signer's certificate may influence the validity of the signed document at a later stage. Only signed documents of which the signer's certificate was not revoked or suspended at the time of verification remain valid forever and are added to the archive. Let us consider a scenario where Alice and Bob use digital signatures to sign a contract. Although both have valid keys at the moment of signing, a year later Alice loses her key and has to revoke her certificate. In this case, the contract remains only valid if, and only if, there is a proof that both signers' certificates were valid when the contract was signed, thus before Alice revoked her certificate.

Protocols to fulfill the second requirement have been presented in [6, 7, 8, 14, 22], yet, to the best of our knowledge, there is no scheme that satisfies all three. In this paper we present the first protocol that simultaneously meets all three properties and at the same time requires a minimum level of trust in the SLTAS.

Our protocol consists of three phases. In the first one, the client creates a package including: the signed document

StorageSS'08, October 31, 2008, Fairfax, Virginia, USA.

to be archived, a proof of when the signature was generated and a proof that the signing key was valid at that moment. Then, the package is sent to the SLTAS, which verifies its correctness before it is inserted into the archive. The SLTAS returns an identification token that will be necessary to retrieve the document later on. Finally, the client can retrieve the package and confirm the consistency of the archived package to conclude that its content is genuine and that it was correctly validated at the time of archival.

The rest of the paper is organized as follows: Section 2 presents an overview of the research on Long-Term Archival systems. We describe our proposal in Section 3 and we discuss its pros and cons in Section 4. We briefly evaluate its performance in Section 5 and we outline possible future lines of research in Section 6. Finally we conclude in Section 7.

2. RELATED WORK

Many papers propose schemes that tackle individual long-term archival problems, mainly focusing on the long-term availability of documents. Some of the proposals that provide long-term availability, lack of a solution for the format migration problem stemming from the obsolescence of hardware and software. Other proposals, although they provide migration schemes, do not guarantee long-term integrity. Moreover, nearly all of these solutions rely on cryptographic techniques (mostly digital signatures), without taking into account that the security of currently available algorithms will most probably be limited or nonexistent in the long-term.

Several publications present schemes that provide "eternal" availability. In 1996, Anderson proposed a system that replicates data across the Internet in such a way that the owner only knows some of its locations [2]. Hence, a censor (not even the owner) cannot delete all existing copies of a file. Ganger et al. present PASIS [21], a survivable storage based on a decentralized architecture. It uses data distribution and redundancy schemes to ensure fault tolerance and to protect integrity and confidentiality of the documents by forcing the attacker to compromise several nodes in order to become a real threat. Another approach based on distributed storage, SafeStore [28], achieves data durability by combining replication across different publicly available Storage Service Providers. An efficient audit protocol is provided to take care of checking that the integrity of the stored documents is preserved over time. POTSHARDS [41] is a different distributed scheme where secrets are not replicated but split into shares [10, 40] and disseminated through different machines. A file can be recovered by recovering a portion of the shares that allows the original file's recovery. A user can only recover the file if she knows the correct combination of shares. In addition, the system uses so-called "approximate pointers" to allow data recovery even when the key is no longer available. An acclaimed proposal, LOCKSS [33], has become an international initiative to support libraries in the preservation of web-published documents in an easy and efficient way. LOCKSS is a peer-to-peer system where documents are replicated over peers. These peers cooperate to detect and repair damage to their content by voting. In this approach the main goal is availability of content in the future, even if little modifications have happened to the documents. Our protocol, on the contrary, aims at preserving the integrity of these documents.

Although all these proposals guarantee the existence and accessibility of documents in the future, none of them presents a solution for weakening of cryptographic primitives, nor for the obsolescence of software to access or convert these documents. Thus, even if documents are available in the far future, it is most likely that recovering their content or proving that their integrity has been maintained is impossible.

Other systems concentrate on dealing with the future obsolescence of software and hardware; i.e., how to store and migrate data such that it is readable well into the future. The Public Record Office Victoria, in Australia, proposes VERS (Victorian Electronic Records Strategy) [44]. The system uses XML encapsulation to include metadata together with the stored document. This extra data ensures the existence of enough information to read the document in the future, even if the software or hardware which was used to create the document is no longer available. VERS uses digital signatures to preserve the integrity of a document, but does not provide a method to confirm the validity of these signatures over time. If the signing algorithms are broken, tampering with the documents cannot be detected anymore, thus its usability is questionable. Further, the revocation of any of the archived document's signing certificates invalidates the archived documents' signature(s). A similar approach to preserve the ability to read (and interpret) information from a given media in the far future has been presented in [16, 17]. This architecture is based on three open standards: the Open Archival Information System Reference Model (OAIS) [20], Extensible Access Method (XAM) [12] and Object-based Storage Device (OSD) [19]. It preserves records by encapsulating them with the metadata needed to ensure its future availability. Amongst this information, the field *Fixity* accounts for the integrity of the document.

Various papers [6, 7, 8, 14, 22] address the obsolescence of cryptography when dealing with digital signatures validation far in time (including the fact that the public key certificates used may be invalid or no longer available at the time of validation). In all these approaches, a digitally signed document is stored in an SLTAS which uses timestamps [1] in order to protect the validity of the initial signature over time. The archive verifies the signatures at regular intervals, and, if they are valid, retimestamps them. This process serves to account for any weakness that may have appeared in the signing algorithms (under the reasonable assumption that a Time Stamping Authority will always use a non-broken state-of-the-art algorithm to issue timestamps). Again, the proposed systems do not consider migration of formats, limiting their functionality with respect to the long-term availability of the documents.

The system presented by Maniatis and Baker in FAST 2002 [31] comes close to our work. However, it does not incorporate timestamping and retimestamping of the archive, and focuses on minimizing the trust in the key-archiving service, a property which is not required when protecting the integrity of archived information. A later work [32], proposes a mechanism to create a secure timeline to protect historic integrity.

Finally, standards [9, 42] are being developed describing the requirements for an SLTAS. They provide guidelines for its implementation including all aspects of security: availability (submit, manage, retrieve or delete archived objects), integrity (the archive should support demonstration of integrity of a document), confidentiality (against third parties and the archive managers), etc. In the same line,

XAdES [24] is a W3C note that proposes an XML format for advanced signatures including long term properties (XAdES-X, XAdES-XL and XAdES-A).

In the next section we present our protocol, a scheme that protects the integrity and validity over time of the signed documents in the archive. The protocol uses timestamps as in previous publications, but goes one step further allowing to prove that the signature existed, and was valid, at a certain point in time (even before being stored in the SLTAS). Retimestamping the archived information with the current state-of-the-art algorithms prevents against the aging of cryptographic algorithms used for the archived packages.

Note that our scheme does not focus on the long-term availability, nor does it on the format migration problem, or key-archival, but on the reliable integrity protection and trustworthy archival of signed documents.

3. OUR SCHEME

In this section we present our scheme to achieve long-term integrity of signed documents providing proof of the indisputable validity in a distant future of the document's signature(s). Our scheme uses timestamps as in [6, 7, 8, 14, 22] to place objective limits on the period between which the signature must have been produced, and to bind the times when the correctness of the signature and the validity of the corresponding signer(s) certificate(s) were formally verified.

The main difference between our approach and previous ones is twofold. First, all previous work considers as first proof of existence of a signature the timestamp collected by the SLTAS upon reception of a signed document. Our protocol, on the contrary, moves this proof to the client side. For this purpose, the client collects a timestamp before the document is first signed to prove that the signature was created after a certain moment, and (optionally) a second timestamp after the signature was created. This is crucial for the archival of contracts, patents, etc. as these signed documents are typically retrieved from an archive much later than the actual time of signing, and this period could be of great importance. Secondly, our design differs from preceding approaches in that we collect and archive indisputable evidence of the validity of the signer's certificate. This is important because the validity of the signer's certificate may have changed since the document had been added to the archive.

3.1 Architecture

Our scheme considers a client-server architecture, with two additional trusted parties (TPs), as shown in Figure 1. For simplicity, the figure represents both client and server using the same TPs. We note that this is not required for the correct functioning of the protocol.

The first trusted party is a Certification Authority (CA) that is needed to issue public-key certificates for both the client and the server, and to provide proofs of their validity, e.g., in the form of Certificate Revocation Lists (CRL) [23] or Online Certificate Status Protocol (OCSP) [37] responses (we recommend the use of OCSP responses over CRLs, as the former provide an explicit and size-efficient confirmation of a certificate's revocation status). These proofs are used, together with the public key certificate of the creator of the signature, as reference information when verifying digital signatures (a signature is invalid if the corresponding certificate is not valid). One has to be aware that, in the long run, the certificates may be revoked or even simply expire. For that matter, whenever a signature is placed during the protocol, the verification reference information is collected and stored beside the signature itself to allow further validations.

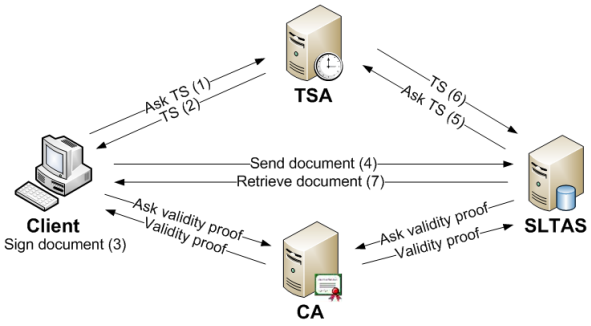

Figure 1: System architecture

In order to provide the timestamps needed to achieve the desired functionality, our architecture includes a second trusted party: the Time Stamping Authority (TSA). A timestamp consists of a cryptographic hash given by the client (a hash of the signed document and its signature in our case), and the current time (provided by the TSA) that are digitally signed by the TSA. This timestamp proves the existence of the signed data before that moment without the possibility of post- or backdating. Client and server make in our scheme, unlike previous proposals [6, 7, 8, 14, 22], use of a TSA. The client requests two timestamps (one before and one after signing a document) to place a limit on the period during which the signature generation must have taken place, and the proof that the corresponding signer's certificate was valid (see Section 3.2). The server uses the TSA to provide evidence of the archival time of the package received from the client, and to refresh the validity of the signatures later in time by retimestamping the archived information (see Section 3.3).

3.2 Client Side

When a client has a document which she wants to archive in the SLTAS, she first signs it in order to provide an initial proof of authenticity. Additionally, the client has to provide a proof of the time of the signature generation, and reference information that proves that the signing key was valid at that point. Both requirements are indispensable later on as evidence in case of disputes over the document (e.g., existence of a contract, registration time of a patent, etc). For instance, this proof can consist of the most recently issued CRL, or a fresh OCSP response that proves that the document's signing key was valid at that time. In previous schemes, the time of the signature generation is given by a timestamp (TS_1) over the signature itself. However, this proves only that the signature was created at any point before the timestamp, but there is no proof of how long before this time the creation took place. To solve this issue, our protocol demands from the client two timestamps (TS_0 and TS_1), one before and one after the signature of the document to bound the time of signing. Figure 2 illustrates the problem and our solution.

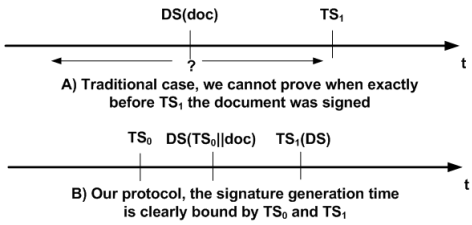

Figure 2: Bounding the signature creation time. A) Traditional case, B) Our protocol

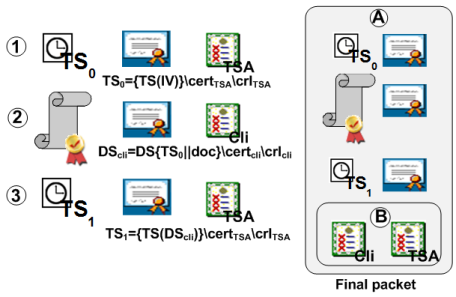

Figure 3: Creating the packet at the client side.

In order to complete the submission of a signed document to the SLTAS, the client has to create a package with: the signed document (*doc*), the two timestamps mentioned above, and a proof of the validity of the signing key used for the signature (CRL, OCSP, etc.). Besides, the client includes the certificates necessary to verify these timestamps and signature later in time. The client proceeds as follows (Figure 3 illustrates the steps and the final result):

1. Requests a timestamp (TS_0) from the TSA over an initialization vector (IV) and collects the TSA certificate chain. The IV is a value known both to the client and the SLTAS (this value needs to be known in order to allow the complete validation described in Section 3.3.3).

2. Signs the document, together with the first timestamp (this guarantees that the signature was generated after the time indicated by TS_0):

$$DS_{cli} = DS(TS_0||doc),$$

   and collects the appropriate certificate. Optionally, the client can also collect the reference information necessary to prove the validity of this certificate at the time of signing.

3. Requests a second timestamp (TS_1) over DS_{cli}, and collects the corresponding TSA certificate chain. This proves that the client signed the document between the times mentioned in TS_0 and TS_1, and (if the proof was collected) that the signer's certificate was valid at the time the signature was created. The signature thus cannot be post- or backdated, and the archival system will be able to confirm that the signing key was valid at the time the signed document was archived.

4. Create a packet with all the information (packet A in Figure 3).

Once this packet is ready, the client can submit it to the SLTAS.

We note that although we have considered only one user signing a document, it is easy to extend the protocol to multiple signers. In this case, the second step of the protocol would be split into as many steps as there are users signing the document. TS_0 and TS_1 would then bind the time of the global signing. The signers would create their signatures in sequential order, such that this order can be proved afterward, obtaining:

$$TS_0||DS_A(TS_0||doc)||DS_B(DS_A(TS_0||Doc))||\cdots$$
$$\cdots||DS_Z(DS_Y(\cdots(DS_B(DS_A(TS_0||Doc))\cdots))||\cdots||TS_1$$

3.3 Server Side

The Secure Long Term Archival System (SLTAS) performs four types of operations: archiving (receive a new document from a client and store it), retimestamping (refresh the integrity, validity and time of signing proofs), retrieval (answer client requests for documents) and removal (eliminate a document from the storage). The rest of this section explains how the server carries out these operations.

3.3.1 *Archiving a document*

Upon reception of a document, the SLTAS checks that the signature and timestamps are valid and collects the information necessary to confirm the signature's validity in the future (if it was not already provided by the client).

Once the signatures have been successfully validated, the SLTAS will request a timestamp (TS_2), which proves that the packet validation and archival took place between TS_1 (the last timestamp requested by the client) and TS_2. Finally, the SLTAS will store all the collected information. The complete procedure is as follows:

1. The SLTAS receives a packet and checks that its signature and timestamps are valid. If it was not provided by the client, the SLTAS also collects the reference information necessary to validate the signatures in a far future (packet B in Figure 4).

2. Optionally, if the SLTAS has a policy with respect to the maximum time allowed between the generation of the two first timestamps, it checks the time span between TS_0 and TS_1.

3. Once the correctness of the packet is confirmed, the SLTAS requests a new timestamp (TS_2) that bounds the packet verification time between TS_1 and TS_2.

4. It stores the whole packet (packet C in Figure 4).

Lastly, the SLTAS calculates an identification token for the packet that the client can use as a proof of ownership of the archived information. This token is sent to the client, who stores it to retrieve the document later in time.

3.3.2 *Retimestamping*

The main purpose of an SLTAS is to extend the reliability of the claim that a signed document was validly signed in the past. In our scheme, as in [6, 7, 8, 14, 22], this property is achieved by retimestamping the stored documents and all the additional information that supports this claim at

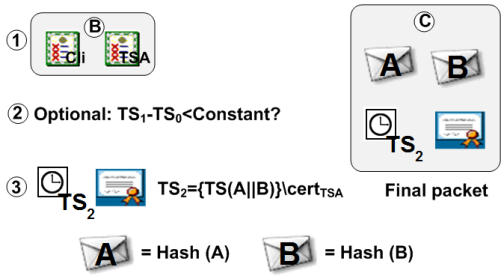

Figure 4: Processing the packet at the server side.

regular intervals (the duration of this interval depends on the application, content, importance, etc. of the archived information).

Each new timestamp is applied using the state-of-the-art cryptographic algorithms at the time of retimestamping, and is used to add an extra layer of security creating an onion. The inner onion layer consists of the package originally sent by the client, and subsequent layers that are added by the server are linked to it. For each of the archived documents, the SLTAS tests its integrity by validating the latest signature on the package and then requests a timestamp over the complete onion. We illustrate a retimestamping operation in Fig 5. At time given by timestamp TS_D, the onion contains packet C (see Fig 4), the timestamp itself and the reference information to validate it. When later on a new layer has to be added (at time given by TS_E), packet D is validated, reference information of this validation collected and a new timestamp is required.

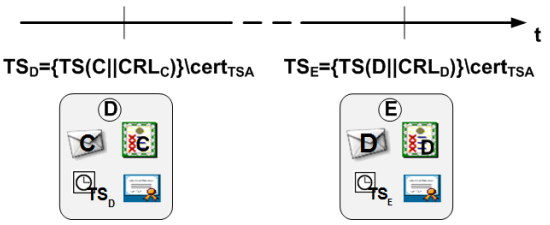

Figure 5: Retimestamping process.

The protocol works under the reasonable assumption that the Time Stamping Authority always uses the latest – state-of-the-art – signing and hashing algorithms, which ensures the protection of the integrity of the full onion until the next timestamping cycle.

3.3.3 *Retrieval of documents*

By storing all the timestamps (the initial and the refreshing ones), along with their certificates and proofs of validity (CRL, OCSP responses,...), we create a verifiable path to assert that the archived document has not been tampered with between the moment it was first signed and the time given by the timestamp in the outer onion.

When a client requests an archived signed document, she is first asked for a proof of ownership or for the identification token that was issued as a result of storing the information in the SLTAS (see Section 4.4). Assuming this proof is correct, the client receives the whole onion; i.e., the original document and signature and all the layers of timestamps. After having verified the information received from the SLTAS, the client (or anyone else such as a judge) will be convinced that the signed document's integrity has been preserved. The verifier proceeds with any of the following two validations:

- **Simple validation**: if the verifier completely trusts the correct operation of the SLTAS, she will also trust that the retimestamping process has been correctly performed. In this case, the verifier only confirms that the last timestamp is indeed valid to conclude that the signed document from the archive originates from its claimed signer(s) and has remained unmodified since the time of archival. This follows from the properties of the inner onion layers that have been verified each time a new layer was added by the SLTAS.
- **Complete validation**: the verifier can also perform a complete confirmation of the correctness of all the timestamping layers over the document to confirm that the archived document is genuine. In this case the verifier starts with the simple validation, after which she will unwrap it and check the next layer with the information available (i.e., certificate of the TSA and proof of its validity at the time the timestamp was requested). The verifier repeats this validation procedure until she reaches the original signature of the document and the document itself. At this point, she can also verify the time when the signature was created and validated. An example of this operation is depicted in Figure 6.

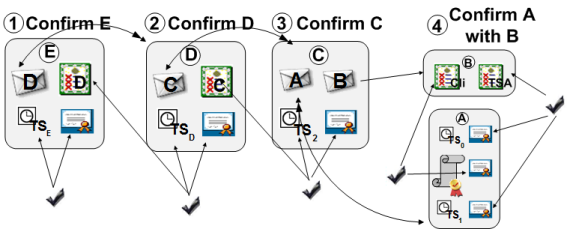

Figure 6: Validation process.

3.3.4 *Removal of a document*

Although the main purpose of an SLTAS is to provide evidence far in the future that a signed document was validly signed in the past, it may be the case that a client (or the archival system itself) would like to delete a document from the archive.

When considering an SLTAS, the removal of a document could have two interpretations. On the one hand, the traditional sense of physically eliminating all content related with the document (the archived information together with its associated onion). On the other hand, the removal operation can also be seen as stopping the archival protection operations of the SLTAS over the document, i.e., the onion corresponding with the archived information is no longer retimestamped. This implies that it will be no longer possible to demonstrate the validity and time of signing of the archived information once the cryptographic algorithms that

were used to calculate the outer onion get compromised. One method or the other would be more appropriate depending on the type of archived information, the storage space available, or the type of service the SLTAS offers.

4. DISCUSSION

4.1 Modification and migration of documents

Our system does not support any modifications to archived information, as its main goal is to preserve the long-term integrity of signed documents, and these are inherently unmodifiable.

Migration of format is a special type of modification, to ensure the availability of content against the obsolescence of hardware and software. However, in our scheme any modification to a stored document implies the meaninglessness of the validity proofs. Haber and Kamat propose in [22] a technique to modify the content of a document not by erasing or changing the original document itself, but by storing the modifications performed and the modified document along with the original. This method can be applied to the problem of format/software obsolescence, as updates to the format of the documents to maintain their readability could be addressed as any other modification. Storing the transformation algorithm would create an auditable path between original and reformatted document such that one can be sure that they present the same content. However, this approach is limited by the fact that applying the algorithm to the original document may no longer be possible due to the obsolescence of conversion software for the archived content.

We note that, although our protocol does not focus on the availability of a document's content, it is easy to adapt it to include a mechanism similar to Haber and Kamat's to deal with the migration problem. Moreover, even if the document is no longer available due to obsolescence, the validity check could still be performed thus guaranteeing that the original document has not been modified.

4.2 Storage overhead

The approach presented in this paper implies an ever growing storage space need. In order to preserve proofs of validity and existence at a point in time of a signature, we need to store the initial timestamps, the signed document itself, the initial reference information, and the onion of timestamps (a new signature and its corresponding reference information for each onion layer). As available disk space grows exponentially in time [43], the retimestamping overhead becomes negligible.

We note that the overhead of the signatures and their corresponding reference information becomes even more negligible with respect to the size of the files stored in the system when they grow, for example, if the SLTAS would store video or other kinds of multimedia information (see Section 5).

4.3 More security properties

Our protocol, as described, focuses on providing the eternal validity of digitally signed information, and does not guarantee "eternal" availability against large-scale disasters, human errors, component faults, etc. [4, 5]. Possible solutions to provide such properties have been introduced in existing systems such as SafeStore [28] or PASIS [21], and are out of the scope of our proposed scheme. However, the output of our system can be used as input of any of the proposals mentioned in Section 2 to improve the long-term availability of the documents and their content.

An additional desired property (as stated in [42]) is confidentiality of archived documents (including confidentiality towards the SLTAS). In order to provide this functionality, an SLTAS must accept encrypted data which has several drawbacks in the long run that no technical measures can prevent. Over time, the encryption algorithm will become weaker. If this is the case, the SLTAS (or an outside attacker) may be able to decrypt the original document and violate its confidentiality. Furthermore, when receiving an encrypted document, the SLTAS does not know what it is archiving, as the plaintext corresponding to the ciphertext is only known by its owner/owners. The SLTAS only protects the integrity of the encrypted bit string, and cannot provide any guarantees regarding the actual content of the file. Moreover, the owner of the encrypted information stored in the SLTAS is responsible for the key management of the archived information in the long-term, to ensure the document can be decrypted in the future. Management and escrow mechanisms to protect encryption keys are outside the scope of our scheme. Note that the scheme we propose does not depend on the availability of genuine signing keys of the signer(s), and on the genuine signing keys of the Time Stamping Authorities that are involved while timestamping and retimestamping. Signing keys do not need to be archived to prevent key compromises, and are therefore not subject to key archival services.

4.4 Other issues

An SLTAS can be used for numerous purposes that may require different levels of authentication. This can be extended to each of the operations a user performs on the SLTAS. In most applications, the SLTAS needs to be authenticated towards the user, but this does not necessarily hold in the other direction. A user may need to be authenticated to insert new documents into the archive, but not for reading them (e.g., public repositories). Furthermore, for some applications (e.g., patents repository) a user may like to have anonymity when accessing the SLTAS, or at least privacy when retrieving information [15, 38]. In our scheme, we did not consider any authentication [35] at the beginning of the protocol, as it depends on the concrete specification of the SLTAS.

The effectiveness of our protocol is based on the reasonable assumption that, in general, signature algorithms and cryptographic hash functions that are used in signature schemes (for instance, RSASSA-PSS [29]) are not suddenly broken. Normally these algorithms become weak in several steps, first theoretically and later in practice, giving time to the Time Stamping Authority to update its signature scheme and the corresponding algorithms, and to the SLTAS to retimestamp the documents it stores before there is any danger to compromise the archived information's integrity. If this assumption becomes invalid, the integrity of the last layer cannot be guaranteed any longer. An archived onion will then only be protected by its inner layers, and if all the signing and hashing algorithms that were used are broken, the integrity of the document and the proof of its validity can no longer be sustained.

Finally, at the client side measures have to be taken in order to securely store the identification token of a document. Tampering with this token must be prevented, as any

modification could mean losing the ability to recover the archived information. This is a very delicate issue, as the obsolescence of the signature algorithms may also affect the signed token. If a document stored in the archive must be kept confidential, i.e., if the document should only be made available to the entity that archived the information, the SLTAS must require authentication before granting access to the archived information (e.g., using access control lists) because the SLTAS does not perform any encryption or decryption services. This would complicate sharing archived documents but this is outside the scope of this paper.

5. IMPLEMENTATION AND EVALUATION

In order to evaluate our protocol, we have written a Java implementation of our SLTAS using RSASSA-PSS [29] signatures as an example. Our implementation confirms that the creation of the initial onion and retimestamping previous onions require negligible time. Furthermore, given Moore's law, this becomes even cheaper on a daily basis. Finally, we show that the storage overhead is affordable.

To carry out our experiments we assumed that the client generates a 1024-bit RSA [39] signature, and that the CAs, Root CAs, OCSP responders, and Time Stamping Authority involved generate 2048-bit RSA signatures to issue the certificates, OCSP responses and timestamps that are included in the first onion. In order to consider the increasing need of security, we chose to enlarge the RSA modulus length of the signing keys that are used to issue the certificates, OCSP responses and timestamps for each onion layer with 256 bits per iteration.

We assume that the SLTAS would retimestamp its archive every three years over a period of 75 years, i.e., starting today with a 2048-bit RSA modulus, ending in 2080 with an 8192-bit RSA modulus. Given [30], it would be conservative to expect that an 8192-bit RSA modulus offers adequate protection until 2100. Therefore the overhead figures presented overestimate by far the authentic numbers that a real system would need to deal with. It is also important to consider that the RSA-based signing scheme may be compromised, in which case the SLTAS would update its signing scheme to use stronger cryptographic techniques to protect the integrity of information (e.g., elliptic or hyper-elliptic curves [27]).

The timings we present do not include hashing the information in the onion nor network latencies when collecting OCSP responses and timestamps; only the generation and verification of the signatures and timestamps were taken into account.

5.1 Archiving a document

As explained in Section 3.2, in order to submit a signed document to the SLTAS the client has to bound the time the signature was generated by requesting two timestamps from the TSA, one before and one after signing. Additionally, the client has to include the certificate chains corresponding to the signatures that are necessary to verify the signing certificate and, optionally, a proof of their validity at the time of signature creation. Once all evidence is accumulated, the client can submit this information to the SLTAS. The overhead given by the timestamps and the reference information for the first onion layer is only 9.7 Kbytes when using realistic X.509 [25] certificate chains with a document signer's certificate of 1400 bytes, Certification Authority (CA) or Timestamping Authority (TSA) certificates of 975 bytes, and self-signed Root CA certificates of 920 bytes. Creating the first onion takes less than one second in our non-optimized implementation.

Upon reception of the packet with the document and the reference information, the SLTAS has to verify its correctness and add a new timestamp that bounds the time of archival. This operation takes less than 350 milliseconds nowadays, and most probably much less in the future [36], which makes it suitable for a large scale server.

5.2 SLTAS Internal operations

In order to preserve the validity of the signatures, the SLTAS periodically retimestamps all the documents it stores. The time needed to retimestamp an individual document using the model described above decreases significantly each time the retimestamping operation if we also take into account that the available computing power doubles every 18 months [36]. We conclude that the operation time is not a restriction for the good functioning of the server.

5.3 Retrieving a document

When a client or a judge needs to verify that the integrity of a signed document has been successfully maintained over time, she can perform any of the validation schemes described in Section 3.3.3. The complete validation operation implies the validation of each of the signatures in the onion layers, including the client's signature; therefore it gives an upper bound for the time of verification of both validation schemes. In Figure 7 we can see the time needed to perform a complete validation depending on the number of retimestamping layers on the document (we recall that we consider three years as retimestamping period, and that we apply the correction of Moore's law). Again, the time to validate 25 retimestamping layers (equivalent to 75 years in our model) is very reasonable.

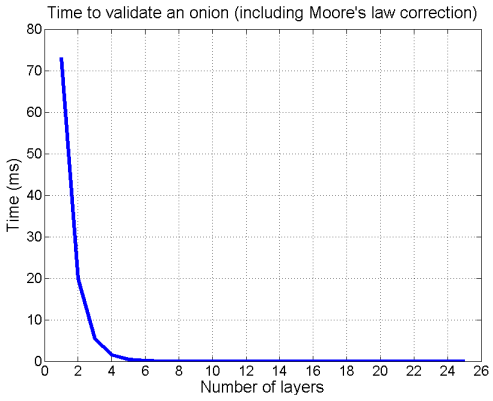

Figure 7: Time to validate an onion (including Moore's correction).

5.4 Storage overhead

As argued in the Section 4, the storage space needed by the SLTAS grows indefinitely. In order to preserve the integrity of the document over time, it is necessary to store the

initial and subsequent timestamps, together with their validity reference information. However, as shown in Figure 8, the overhead of the growing certificates, OCSP responses and timestamps in each onion layer's is affordable, and for certain types of documents (e.g. multimedia information) it is even negligible (see Figure 9).

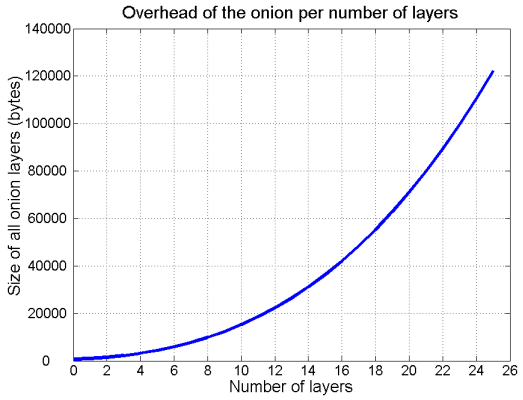

Figure 8: Overhead of the onion.

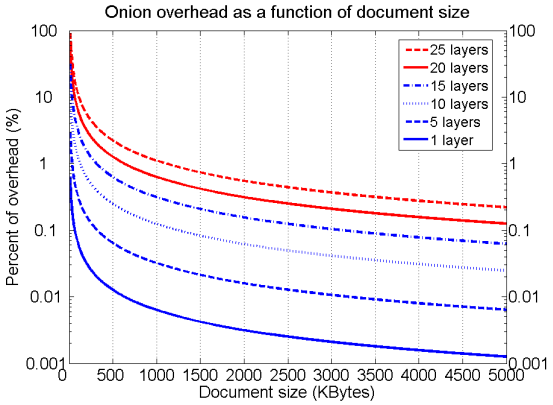

Figure 9: Onion overhead as a function of the document size.

6. FUTURE WORK

The presented design for an SLTAS uses PKI technology to protect the archive. This implies the use of public-key certificates of the document signer, that may contain sensitive information (e.g., official name, unique identifiers). During the period of archival, these certificates are not only validated by the SLTAS, but they are stored along with the document itself and therefore available to any client that retrieves it. To avoid privacy concerns, a line of future research would be to extend the protocol to use anonymous credentials [11, 13], which would protect sensitive data from the server and other clients.

The obsolescence of cryptography also affects the solution of other problems associated with long-term archiving. Lately, several papers [3, 18, 26] address the problem of proving data possession, where the user of an external archival facility wants to check that the server correctly maintains archived her document while avoiding the cost of retrieving it in terms of bandwidth, memory, etc. The authors provide "long-term" solutions based on RSA signatures, symmetric cryptography or MAC algorithms without accounting for the possible weakening of these primitives over time and its consequences. More work is needed to improve these protocols to ensure that the obsolescence of cryptography will not be an impediment for their correct functioning in the future.

7. CONCLUSION

The correct archival of signed documents is an essential building block for systems that depend on the indisputability of these documents. Applications for e-government, e-health, e-commerce or even more basic cases, such as simple web services (like forums or web browsing) have the need to securely archive their signed application-level information [34] to prevent that actors can deny the production of these signatures, e.g., by simply revoking the certificate corresponding with the signing key. Guaranteeing this property involves two aspects. First, collecting a proof that the signer's certificate was valid as soon as possible after the signature was created. Second, the careful retimestamping of the signed information and its validity proof, so that this evidence does not become obsolete over time as a consequence of the aging of the underlying cryptographic primitives.

The contribution of this paper consists in a protocol that deals with the weakening of cryptography over time and the need to re-validate signed documents and their certificates. Our scheme permits to prove far in the future the validity of a digital signature in the past, thus providing the *eternal* validity of this signature. Furthermore, it allows binding the time span of the signature generation.

Our protocol improves all previous schemes [6, 7, 8, 14, 22], by bounding the time of existence and validity of the signature between two moments in time, instead of just proving that it existed before the moment of archival. Our approach requires the minimum amount of trust in the participating servers and provides full verifiability of the actions taken by the SLTAS. We have also shown that the time and space overhead associated with our protocol does not present an impediment to its usability. Finally, as discussed in Section 4, our protocol can be integrated with other proposed solutions to account for migration problems, availability, etc.

8. REFERENCES

[1] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato. Internet X.509 public key infrastructure Time-Stamp protocol (TSP). RFC 3161, Internet Engineering Task Force, August 2001.

[2] R. Anderson. The eternity service. In J. Přibyl, editor, *Proceedings of Pragocrypt '96*, pages 242–252, Prague, 1996. Czech Technical University Publishing House.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In S. De Capitani di Vimercati and P. Syverson, editors, *CCS'07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 598–609, New York, NY, USA, October 2007. ACM.

[4] M. Baker, K. Leaton, and S. Martin. Why traditional storage systems don't help us save stuff forever. In *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks, 1st Workshop on Hot Topics in System Dependability (HotDep-05)*, Yokohama, Japan, June 2005. IEEE Computer Society Press.

[5] M. Baker, M. Shah, D. S. H. Rosenthal, M. Roussopoulos, P. Maniatis, TJ Giuli, and P. Bungale. A fresh look at the reliability of long-term digital storage. *SIGOPS Oper. Syst. Rev.*, 40(4):221–234, 2006.

[6] A. Jerman Blazic. Long term trusted archive services. In *First International Conference on the Digital Society ICDS*, page 29. IEEE Computer Society, Jan 2007.

[7] A. Jerman Blazic and B. Dzonova-Jerman-Blazic. Implementing trustworthy internet based long term electronic preservation service – the eKeeper project. In M. H. Hamza, editor, *IASTED International Conference on Communications, Internet, and Information Technology*, pages 291–296, 2004.

[8] A. Jerman Blazic and P. Sylvester. Provision of long-term archiving service for digitally signed documents using an archive interaction protocol. In D. W. Chadwick and G. Zhao, editors, *EuroPKI*, pages 240–254, 2005.

[9] R. Brandner, U. Pordesch, and T. Gondrom. RFC 4998: Evidence record syntax (ERS). RFC 4998, Internet Engineering Task Force, August 2007.

[10] R. Brinkman, J. Doumen, and W. Jonker. Using secret sharing for searching in encrypted data. In W. Jonker and M. Petkovic, editors, *Secure Data Management*, volume 3178 of *Lecture Notes in Computer Science*, pages 18–27. Springer, 2004.

[11] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In B. Pfitzmann, editor, *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118, London, UK, 2001. Springer.

[12] C. Casten. The power of XAM. SNIA Data Management Forum.

[13] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[14] S. Chokhani and C. Wallace. Trusted archiving. In *Proceedings of the 3rd Annual PKI R&D Workshop*. NIST, April 2004.

[15] G. Danezis and C. Diaz. Space-efficient private search. In R. Dhamija and S. Dietrich, editors, *Proceedings of Financial Cryptography (FC2007)*, volume 4886 of *Lecture Notes in Computer Science*, pages 148–162, Tobago, 2007. Springer-Verlag.

[16] M. Factor, D. Naor, S. Rabinovici-Cohen, L. Ramati, P. Reshef, and J. Satran. The need for preservation aware storage: a position paper. *Operating Systems Review*, 41(1):19–23, 2007.

[17] M. Factor, D. Naor, S. Rabinovici-Cohen, L. Ramati, P. Reshef, J. Satran, and D. L. Giaretta. Preservation datastores: Architecture for preservation aware storage. In *MSST*, pages 3–15, 2007.

[18] D. G. Filho and P. Barreto. Demonstrating data possession and uncheatable data transfer. Technical report, 2006. `http://eprint.iacr.org/`.

[19] International Comitee for Information Technology. ANSI/INCITS 400-2004 SCSI Object-based Storage Device Commands (OSD), 2004.

[20] International Organization for Standarization. ISO standard 14721:2003 space data and information transfer systems–a reference model for an open archival information system (OAIS), 2003.

[21] G. R. Ganger, P. K. Khosla, M. Bakkaloglu, M. W. Bigrigg, G. R. Goodson, S. Oguz, V. Pandurangan, C. A. N. Soules, J. D. Strunk, and J. J. Wylie. Survivable storage systems. In *Proceedings of the DARPA Information Survivability Conference & Exposition (DISCEX)*, pages 184–195, Anaheim, CA, USA, June 2001. IEEE CS Press.

[22] S. Haber and P. Kamat. A content integrity service for long-term digital archives. In *IS&T Archiving Conference (Archiving 2006)*, volume 3, pages 159–164. The Society for Imaging Science and Technology, May 2006.

[23] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. RFC 3280, Internet Engineering Task Force, April 2002.

[24] European Telecommunication Standard Institute. ETSI TS 101 903 - XML advanced electronic signatures (XAdES), 2002.

[25] ITU. ITU-T rec. X.509|ISO/IEC 9594-8: The directory: Authentication framework, 2000.

[26] A. Juels and B. S. Kaliski. PORs: proofs of retrievability for large files. In S. De Capitani di Vimercati and P. Syverson, editors, *ACM CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 584–597, New York, NY, USA, 2007. ACM.

[27] N. Koblitz. Hyperelliptic cryptosystems. *Journal of Cryptology*, 1(3):139–150, 1989.

[28] R. Kotla, M. Dahlin, and L. Alvisi. SafeStore: A durable and practical storage system. In *Proceedings of the 2007 USENIX Annual Technical Conference*, pages 129–142. USENIX, June 2007.

[29] RSA Laboratories. PKCS♯1v2.1: RSA cryptography

standard, June 2002.

[30] A. Lenstra. Key lengths. *Handbook of Information Security, Volume II: Information Warfare; Social, Legal and International Issues; and Security Foundations*, Volume II: Information Warfare; Social, Legal and International Issues; and Security Foundations:617–635, 2006.

[31] P. Maniatis and M. Baker. Enabling the archival storage of signed documents. In *FAST*, pages 31–45, 2002.

[32] P. Maniatis and M. Baker. Secure History Preservation Through Timeline Entanglement. In *Proceedings of the 11th USENIX Security Symposium*, San Francisco, CA, USA, August 2002.

[33] P. Maniatis, M. Roussopoulos, T. J. Giuli, D. S. H. Rosenthal, and M. Baker. The LOCKSS peer-to-peer digital preservation system. *ACM Trans. Comput. Syst.*, 23(1):2–50, 2005.

[34] J. McAdams. 27 billion gigabytes to be archived by 2010. *Computerworld*, December 2007.

[35] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*, chapter 10 - Identification and Entity Authentication, pages 385–424. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[36] G. E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[37] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 internet public key infrastructure online certificate status protocol - OCSP. RFC 2560, Internet Engineering Task Force, June 1999.

[38] R. Ostrovsky and W. E. Skeith III. A survey of single-database private information retrieval: Techniques and applications. In T. Okamoto and X. Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 393–411. Springer-Verlag, 2007.

[39] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

[40] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[41] M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti. POTSHARDS: Secure long-term storage without encryption. In *Proceedings of the 2007 USENIX Annual Technical Conference*, pages 143Ű–156. USENIX, June 2007.

[42] C. Wallace, R. Brandner, and U. Pordesch. Long-term archive service requirements. RFC 4810, Internet Engineering Task Force, March 2007.

[43] C. Walter. Kryder's law. *Scientific American*, page 2, 2005.

[44] A. Waugh, R. Wilkinson, B. Hills, and J. Dell'oro. Preserving digital information forever. In *DL '00: Proceedings of the fifth ACM conference on Digital libraries*, pages 175–184, New York, NY, USA, 2000. ACM Press.