www.ijiarec.com

ISSN:2348-2079

Volume-6 Issue-2

International Journal of Intellectual Advancements and Research in Engineering Computations

Securing Digital Documents using X.509 Certificates on Cloud Infrastructure

[1]Mr. S.Sambasivam,MCA.,MPhil., Associate professor,
[2]Ms. M. Ponni, III MCA,
Department of MCA, Nandha Engineering College(Autonomous),Erode-52.
Email ID: sammy2173@gmail.com, lavanyateen123@gmail.com

Abstract*—This paper is intended to provide a cloud-based digital signature platform with enhanced security solution in the field of cryptography. We proposed a new schema of digital signature where signature generation and verifications are done on the cloud environment. 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 provides a proof, 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.

Index Terms-Secure long-term archiving, Digitally signed documents, Re-time stamping.

I. 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. Traditionally, archival systems aim at ensuring integrity and availability, but even if an archive initially provides these properties, during 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 users from the fragile nature of the private signing keys and the validity status of the corresponding public verification key certificates. 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 time, namely before the document was archived:

Thedocument 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 can-not be shifted before or after the two indisputable moments. 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. 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 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. 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.

II. RELATED WORKS

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 hard-ware and software. Other proposals, although they provide migration schemes, do not guarantee long-term integrity. Moreover, nearly all these solutions rely on cryptographic techniques (mostly digital signatures), without considering 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 le. Ganger et al. present PASIS, 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, 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 is a different distributed scheme where secrets are not replicated but split into shares [10] and disseminated through different machines. A le can be recovered by recovering a portion of the shares that allows the original le's recovery. A user can only recover this 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, 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. Our protocol, on the contrary, aims at preserving the integrity of these documents.

Although all these proposals guarantee the existence andaccessibility 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.

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) in-formation from a given media. This architecture is based on three open standards: The Open Archival Information System Reference Model (OAIS), Extensible Access Method (XAM) and Object-based Storage Device (OSD). It preserves records by encapsulating them with the metadata needed to ensure its future availability.

Various papers [6, 7, 8] 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, retime stamps 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 comes close to our work. However, it does not in-corporate time stamping and retime stamping 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, pro-poses a mechanism to create a secure timeline to protect historic integrity.

Finally, standards [9] are being developed describing the requirements for an SLTAS.

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). Retime stamping the archived information with the current state-of-the-art algorithms prevents against the aging of cryptographic algorithms used for the archived packages.

III. SYSTEM METHODOLOGY

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] 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 time-stamp 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.

A.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) or Online Certificate Status Protocol (OCSP)responses. 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

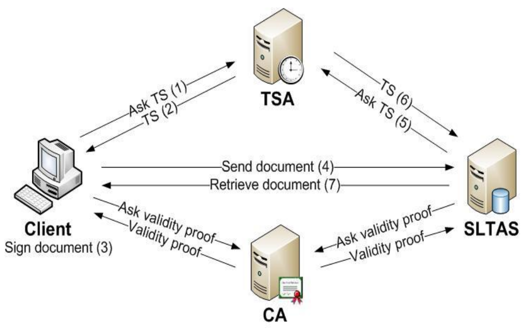

Figure 1: System architecture

certificate is not valid). One must 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.

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], use of a TSA. The client requests two timestamps (one be-fore 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 The server uses the TSA to provide evidence of the archival time of the pack-age received from the client, and to refresh the validity of the signatures later in time by retime stamping the archived information .

B. 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 must 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 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 is-sued 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 be-fore 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.

To complete the submission of a signed document to the SLTAS, the client must 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.

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.

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

$$DS_{cli} = DS(TS_0 jjdoc),$$

   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. Create a packet with all the information

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:

$$TS_0jjDS_A(TS_0jjdoc)jjDS_B(DS_A(TS_0jjDoc))jj$$
$$jjDS_Z(DS_Y(\quad(DS_B(DS_A(TS_0jjDoc))\quad))jj\ jjTS_1$$

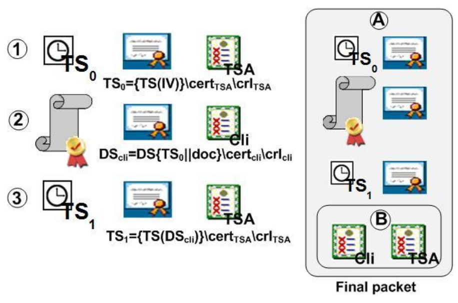
①
TS0
TS0={TS(IV)}\certTSA\crlTSA
TSA
②
DScli=DS{TS0||doc}\certcli\crlcli
Cli
③
TS1
TS1={TS(DScli)}\certTSA\crlTSA
TSA
A
TS0
TS1
B
Cli
TSA
Final packet

Figure 2: Creating the packet at the client side.

C. Server side

The Secure Long Term Archival System (SLTAS) per-forms four types of operations: archiving (receive a new document from a client and store it), retime stamping (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.

Archiving A Document

Upon reception of a document, the SLTAS checks that the signature and timestamps are valid and collects the in-formation 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 in-formation 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

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.

D.Retime Stamping

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], this property is achieved by retime stamping the stored documents and all the additional information that supports this claim atregular 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 retime stamping 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. At time given by timestamp TS_D, the onion contains packet C, the timestamp itself and the reference information to validate it. When later 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.

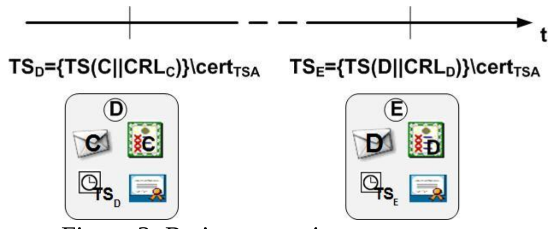

Figure 3: Retime stamping process.

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 time stamping cycle.

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 because of storing the in-formation in the SLTAS. Assuming this proof is correct, the client receives the whole onion; i.e., the original document and signature and all the layers of time-stamps. 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 retimes tamping 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 time stamping 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 in-formation 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.

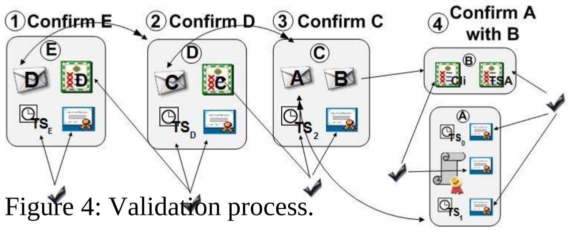

Figure 4: Validation process.

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 stopping the archival protection operations of the SLTAS over the document, i.e., the onion corresponding with the archived information is no longer re-time stamped. 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 thatwere used to calculate the outer onion get compromised. One method or the other would be more appropriate de-pending on the type of archived information, the storage space available, or the type of service the SLTAS fires.

IV. 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 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 sig-nature was created. Second, the careful retime stamping of the signed information and its validity proof, so that this evidence does not become obsolete over time because 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], by bounding the time of existence and validity of the sig-nature 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.

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 services. In J. Pribyl, editor, Proceedings of Pragocrypt'96, pages 242{252, Prague, 1996.Czech Technical University Publishing House.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In S. De CapitanidiVimercati 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 study 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, Roussopoulos, P. Maniatis, TJ Giuli,andBungale. A fresh look at the reliability of long-term digital storage. SIGOPS Opera. Syst. Rev., 40(4):221{234, 2006.

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

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

[8] A. JermanBlazic and P. Sylvester. Provision of long-term archiving service for digitally signed documents using an archive interaction protocol. In 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 Jonker and M. Petkovic, editors, Secure Data Management, volume 3178 of Lecture Notes in Computer Science, pages 18{27. Springer, 2004.

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