10

A Two-level Time-Stamping System

Alban Gabillon and Jungsoo Byun
SIS/Equipe Informatique. Université de Toulon et du Var. {gabillon, byun}@univ-tln.fr

Key words: Time-stamping protocols, Time-Stamping Authority, One Way Accumulator, Verification protocol, Digital Notary Services

Abstract: Time-Stamping is a cryptographic technique which allows us to prove that an electronic document existed at a certain point in time and that it has not been modified since then. Different time-stamping schemes have already been proposed. Most of them use the concept of trusted Time-Stamping Authority (TSA). A TSA is in charge of time-stamping documents and delivering a time-stamping certificate for each time-stamped document. The purpose of this paper is to propose a new time-stamping scheme using a *Local Time-stamping System* (LTS). The main idea can be summarised as follows: digests of the documents to be time-stamped are sent to a Local Time-stamping System (LTS). The LTS accumulates the digests into a *round value* using a round-based protocol. The round value is then time-stamped by a trusted and official TSA. We show how this time-stamping scheme could be useful for an organisation such as a digital library or a company.

1. INTRODUCTION

Like traditional paper documents, electronic documents need to be dated. However, electronic documents are easy to alter and forge. A date appended to a document can easily be replaced or modified by anybody having access to the document via a word processor.

Time-Stamping is a cryptographic technique which allows us to certify that an electronic document existed at a certain point in time and that it has not been modified since then. The first time-stamping protocol was proposed in [HS91]. A survey of the existing time-stamping protocols and of their security can be found in [MQ97][Pal98][Qal99][BLLV98][BLS00]. Most of the existing time-stamping schemes use the concept of trusted Time-Stamping Authority (TSA). A TSA offers *digital notary services* that is, a TSA is able to securely time-stamp an electronic document. For example, If Alice wants to time-stamp a document *d* then she must proceed as follows:

– Using a well known one-way[1] collision-free[2] hashing function *h,* Alice first computes a *fingerprint* (also called a *digest*) of the original document *d.*
– Alice sends the digest to the TSA. Notice that for confidentiality reasons document *d* is not sent to the TSA.
– The TSA sends back Alice a *time-stamping certificate* T_d which has been constructed with a particular *time-stamping protocol.*
– Alice can later present *d* and T_d to a verifier who needs to know when *d* was time-stamped, and who needs to make sure that *d* has not been modified since then. For this, the verifier must use the *verification protocol* associated with the time-stamping protocol which was used.

In most countries digital time-stamps and digital signatures have not received a legal value yet. Consequently, companies selling digital notary services and pretending to have the *authority* to time-stamp documents, have actually not been granted any official and legal authority. However, there is no doubt that in the near future, digitally signing and digitally time-stamping an electronic document will have the same legal force (and maybe more) than dating and signing a traditional paper document. At the same time, companies which will be officially and legally entitled to sell digital notary services will be clearly identified and organised. Most probably, they will integrate the internet Public Key Infrastructure (see [AT99] for an introduction to PKI and see [Aal00] for integrating TSA's to PKI). We, therefore, predict that the need for digitally signing and time-stamping electronic documents will then increase rapidly.

Now, let us consider an entity such as a company. The managers of the company may decide to time-stamp most of the electronic documents which are issued or received by the company. For this, they might ask each employee producing or receiving a given electronic document to contact a trusted and official TSA in order to have the document time-stamped. However, this scheme has some disadvantages:

– It is difficult to know which documents are time-stamped and which documents are not.
– It is expensive. The TSA will charge the company for each time-stamped document.

[1] one-way means that no portion of the original document can be reconstructed from the digest

[2] collision-free means that it is infeasible to find x and x' satisfying $h(x) = h(x')$

- Documents sent to the TSA by the employees might be personal documents which do not belong to the company.
- Documents cannot be time-stamped at a high rate.

The purpose of this paper is to propose a two-level time-stamping scheme using a *Local Time-stamping System* (LTS) and a trusted TSA. The main idea can be summarised as follows: if a particular employee has a document to time-stamp then this employee sends a digest of the document to the LTS. All the digests received by the LTS within a single round r are combined in order to produce the *round value* of the round r. This round value is then sent to a trusted TSA in order to be time-stamped.

In section 2 of this paper, we introduce the concept of LTS and we describe our 2-level time-stamping protocol as well as the corresponding verification protocol. In section 3, we present some possible solutions to manage time-stamped documents. In section 4, we discuss some security aspects. Section 5 concludes this paper.

2. LOCAL TIME-STAMPING SYSTEM

2.1 Principle

Behind the concept of LTS (Local Time-stamping System) there is a group of users. These users may be users working together in a particular organisation, for instance a company. As we have seen in the previous section, it is not realistic and efficient to ask an employee to contact a trusted TSA each time this employee has to time-stamp a document, especially if the company needs to have a high number of documents time-stamped everyday. The solution for this company is to install an LTS. In this section we describe the basic characteristics of such an LTS.

Our LTS uses a round-based protocol. A round has a certain duration which must be chosen by the administrator of the time-stamping system. It can be one second, one minute, one hour, one day. The smallest the round duration is, the better is the accuracy of the time-stamps. Documents which are time-stamped during the same round are all time-stamped with the same date and time by the TSA.

Let us assume that m documents $d_1, d_2, \dots d_m$ are to be time-stamped during round r. Their corresponding digests $y_1, y_2 \dots y_m$ are submitted to the LTS. These digests are combined in order to produce a single round value z which is called the *round value* of the round r (see next section 2.2). Once z has been calculated, it is sent to a distant trusted TSA. This TSA securely time-stamps z and returns a *time-stamping certificate for z, $Cert_{TSA}(z)$*. This certificate consists of z, an announced date and time, and a secure time-stamp. The LTS can verify (with the verification protocol associated with the time-stamping protocol used by the TSA) that the announced date and time match the date and time included in the time-stamp. The LTS then produces a *time-stamping certificate for each document* d_i (with $1 \leq i \leq m$). This time-stamping certificate $Cert_{LTS}$ (d_i) consists of y_i, z, $Cert_{TSA}(z)$ and everything needed to prove that y_i participated in the construction of z.

Later, if a verifier wants to check the time-stamp of a given time-stamped document then he must be provided with the document and the corresponding time-stamping certificate. The verifier must hash the document and verify that the digest he obtains is equal to the digest included in the certificate. If they are equal then it proves that the document has not been modified since it was time-stamped. Then, the verifier checks whether the digest participated in the construction of the round value which is included in the certificate. Finally, the verifier checks the time-stamping certificate for the round value.

Some of the advantages of having a LTS are the followings:

- Documents can be produced at a high rate. There is no (theoretical) limitation on the number of documents which can be time-stamped within a particular round.
- The company is charged for a single certificate per round, without regard to the number of documents to be time-stamped during one round.
- The company has the possibility to control the time-stamping operations precisely. It can fix who has the right to time-stamp what. Auditing the time-stamping operations becomes easy.

Notice that we did not mention whether the documents which are time-stamped are digitally signed or not (see [RSA78] for a presentation of digital signatures). This is because there is absolutely no difference between time-stamping a signed document and time-stamping an unsigned document. If a document which is signed has to be time-stamped then the digest which is sent to the LTS is computed from the document with its signature appended to it.

Regarding this topic, let us mention that in [BHS92] it is shown that time-stamping a digitally signed document extends the life-time of the digital signature.

2.2 Protocol

Firstly we must say that the protocol used by the LTS is *not* a time-stamping protocol. It is a protocol which aims at securely constructing a single global time-stamping request from several local time-stamping requests. The global request is then submitted to the trusted TSA. The protocol we have chosen for the LTS was first proposed in [BM94]. It uses a one-way accumulator.

A one-way accumulator is a one-way function *owa* which is *quasi-commutative.* This means that if one starts with an initial value x_0 and a set of values $y_1, y_2 ... y_n$ then the accumulated hash

$$z = owa(owa(...owa(owa(x_0,y_1),y_2)...y_{n-1}),y_n)$$

would be unchanged if the order of the y_i were permuted. Consequently,

if $z_i = owa(owa(...owa(owa(x_0, y_1), y_2) ...y_{i-1}),y_{i+1})...y_{n-1}), y_n)$ then $z = owa(z_i, y_i)$.

Proving that y_i participated in the construction of z means verifying that $z = owa(z_i,y_i)$.

One can refer to [BM94] in order to know more about accumulators.

Knowing this, we can now present the protocol of our LTS. The protocol uses the modular exponentiation as a one-way accumulator:

Let $y_1, y_2 ... y_m$ be the digests of the documents $d_1, d_2, \ldots d_m$ to be time-stamped during round r.

$y_i = h(d_i)$ with h being a well known one-way collision-free hashing function. See [MOS97] for a presentation of such functions.

Let x_0 be an initial value. Let $n = pq$ with p and q being two safe primes (see [BM94] for the definition of a safe prime).

Let z be the round value of the round. z is computed as follows:

$$z = x_0^{\prod_{i=1}^{m} y_i} \bmod n = ((\Lambda\ ((x_0^{y_1} \bmod n)^{y_2} \bmod n)\Lambda\ )^{y_m})\bmod n.$$

The round value z is sent to the TSA for being time-stamped. The time-stamping certificate $Cert_{TSA}(z)$ is returned by the TSA.

For each digest y_j the *partial* round value z_j is computed by the LTS.

$$z_j = x_0^{\prod_{i=1, i \neq j}^{m} y_i} \bmod n.$$

Finally, the time-stamping certificate $Cert_{LST}(d_j)$ for the document d_j is produced by the LTS.

$$Cert_{LTS}(d_j) = (y_j, z_j, z, Cert_{TSA}(z)).$$

Note that the time-stamp is produced by the TSA. Presenting the scheme used by the TSA is irrelevant in this paper. However, we can mention that the scheme used by the TSA may be the scheme defined in [BLLV98] and later refined in [BLS00]. Indeed this scheme seems to be the most reliable of all the existing time-stamping schemes.

Note also that in [BM94], one-way accumulators are presented as a solution for time-stamping documents. Therefore, the TSA could also use a one-way accumulator. However, it has been shown that there is no efficient construction of accumulators without trapdoor (see [San99]).

In our paper, since we do not use accumulators for time-stamping, this trapdoor problem is not of our concern. We only use an accumulator for securely computing the round value which is time-stamped by the TSA. Using a one-way accumulator has the advantage of providing us with a very simple verification protocol.

2.3 Verification protocol

A verifier who needs to check the time-stamp of document d_j must be provided with the pair $(d_j, Cert_{LTS}(d_j))$. The verifier must check whether

1. $h(d_j) \overset{?}{=} y_j$ (has the document been modified since it was time-stamped ?)

2. $z_j^{y_j} \bmod n \overset{?}{=} z$ (did the document participate to the construction of z ?)

3. Finally the verifier must check $Cert_{TSA}(z)$ with the verification protocol corresponding to the time-stamping protocol used by the TSA.

3. MANAGING TIME-STAMPED DOCUMENTS

In the previous section, we did not mention anything about the management of the time-stamping certificates and the time-stamped documents. In this section we suggest two basic solutions for managing time-stamped documents efficiently.

3.1 Centralised management

Centralised management can be used by entities like digital libraries, e-print or e-publishers. Here, documents to be time-stamped are also documents to be *published.*

Instead of sending the digests, the users send the documents to the LTS. The LTS computes the digests itself[3] , and securely stores the time-stamping certificates and the time-stamped documents. Notice that documents sent by the users may be encrypted for confidentiality reasons using Secure Socket Layer.

Figure 1 represents the design of a digital library. Users of the library can submit their documents for time-stamping and publishing. The time-stamping certificates are stored in a Time-stamps Directory whereas the documents themselves are published in a Documents Directory. Anybody downloading a published document may also ask for the corresponding time-stamping certificate in order to verify the authenticity of the document. Notice that, the LTS may return a *copy* of $Cert_{LTS}(d_j)$ to the user who submitted d_j. This copy serves as an acknowledgement and has only an informational purpose.

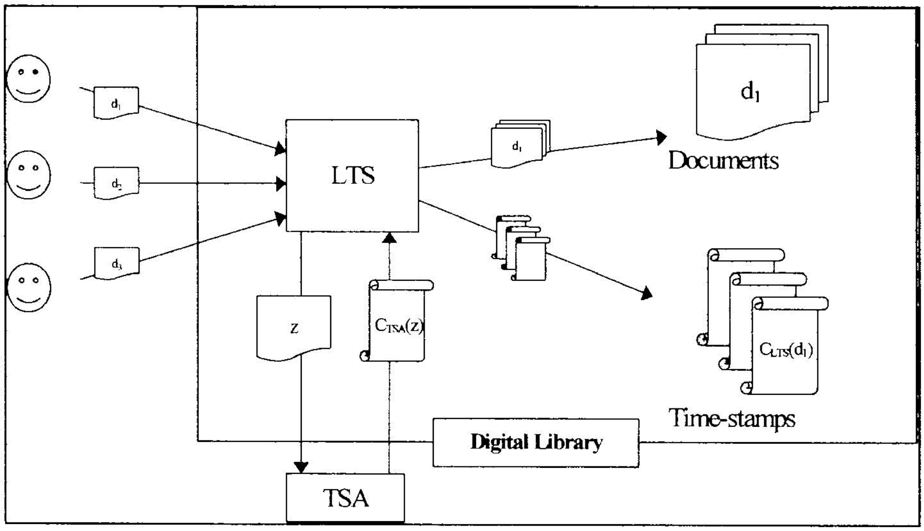

Figure 1. Centralised management of time-stamped documents

[3] The LTS could directly use the document to compute the round value. However, since a digital document is big, the computation would be too slow. Therefore, it is better to hash each document before computing the round value.

3.2 Distributed management

Distributed management can be used by entities like companies. Here documents are time-stamped for internal management reasons. Users send the digests of the documents to the LTS. The LTS, in return, provides them with the time-stamping certificates. The documents and their corresponding time-stamping certificates are locally managed by each user.

Figure 2 represents the design of a company where each user actually acts on behalf of a particular department of the company. Time-stamped documents and their corresponding certificates are locally retained and managed at the departmental level.

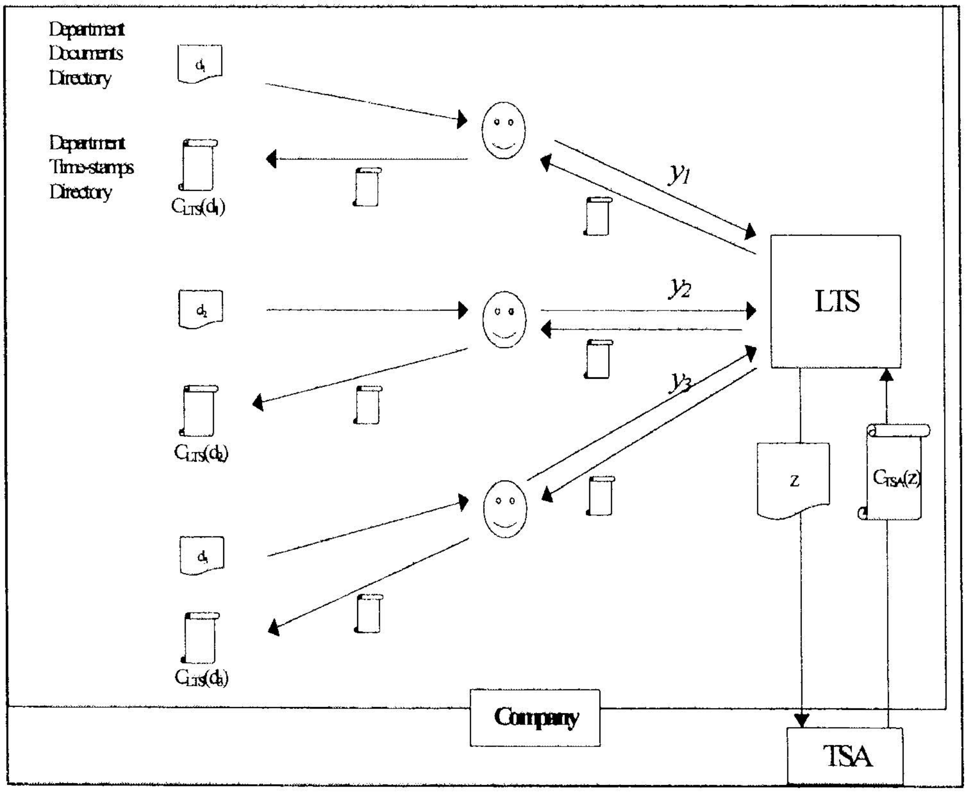

Figure 2. Distributed management of time-stamped documents

4. SECURITY

4.1 Security of the LTS

In the protocol used by the LTS, a round value is only used to securely accumulate several document digests in order to,

- centralise the time-stamping requests,
- pay for only one certificate,
- enable time-stamping at a high rate.

Since the LTS does not time-stamp the documents, it cannot backdate a document. Therefore, the only concern of the participants to the LTS should be to verify that the LTS cannot discard their requests or delay the execution of their requests. Notice that this potential problem, which is called a denial of service, is not specific to our LTS. All existing time-stamping systems cannot be trusted regarding this aspect.

In [Qal99], Quisquater et al. have designed a time-stamping system using an accumulator. However, their aim was to build a trusted system for a TSA. In order to make their system trusted, Quisquater et al. have proposed to link and accumulate the successive round values into a "big round value" which is regularly published into a widely witnessed media (like a daily newspaper). The verification protocol of [Qal99] includes steps 1 and 2 of our verification protocol but it also includes one more step: the verifier of a time-stamp must be provided with the round values belonging to the corresponding big round. By linking and accumulating these round values, the verifier can then reconstruct the corresponding trusted big round value. This technique ensures that, after a big round value has been published at a certain date, forging a time-stamp indicating an earlier date is impossible. With our LTS, we do not need to publish any round value. Indeed each round value is time-stamped by a TSA that we assume completely trusted and recognised as having a legal force.

The only real problem that we see might be the following: suppose the LTS sends two round values z_1 and z_2 successively to the TSA. Due to some network congestion round value z_2 reaches the TSA before round value z_1. This may be possible especially if the round duration is short. Both round values are certified but z_2 receives a time-stamp indicating a date earlier than the date included in the time-stamp of z_1! This problem is actually a problem common to all time-stamping systems. It comes from the fact that the time which is taken into account for processing a request is not the time when the request is sent, but the time when the request is received.

4.2 User authentication

Basically, we can make the distinction between the following two types of users:

- Users who have the right to use the LTS.
- Users who have the right to consult the time-stamped documents and verify the time-stamps.

Users who have the right to consult the time-stamped documents and verify the time-stamps can be everybody or a very restricted number of people. This depends on whether the documents are public or whether they are private. In the case of a digital library (see section 3.1), we can assume that everybody has the right (for example through an anonymous access) to consult the documents and their corresponding time-stamps. Now, in the case of a company, it is clear that access to some of the documents will be restricted.

Users who have the right to use the LTS have to be registered by the LTS. The LTS must authenticate each user sending a time-stamping request.

5. CONCLUSION

In this paper, we presented a practical local time-stamping system which can serve as an intermediary between a group of users and an official and trusted TSA. We described the advantages of implementing a LTS in an organisation like a digital library or a company. However, further work remains to be done:

- We have to study how to implement a LTS. In particular we must define a policy for managing efficiently both the documents and the time-stamps directories.
- We need to define a security policy for managing the users authorisations. We must also propose a solution for implementing this policy (through an LDAP server for example).

Finally let us note that the Achilles' heel of our LTS is the TSA. Indeed commercial companies like Surety [Sur] selling digital notary services have not been granted any official authority to deliver time-stamping certificates. Companies forming the embryo of the internet PKI also lack such an official authority (see [ES00] for a discussion on this topic) and since, most probably, official TSA's will integrate the internet PKI, we must wait for a while before we can really have a TSA having a legal force at our disposal.

REFERENCES

[Aal00] C. Adams, P. Cain, D. Pinkas, R. Zuccherato. *Internet X.509 Public Key Infrastructure Time Stamp Protocol.* Draft IETF. January 2001.

[AT99] A. Arsenault and S. Turner. *Internet Public Key Infrastructure PKIX roadmap.* Draft 04, IETF, October 1999.

[BHS92] D. Bayer S. Haber and W. Stornetta. *Improving the efficiency and reliability of digital time-stamping.* In Springer Verlag editor. Sequences' 91: Methods in Communication, Security and Computer Science, pages 329-334,1992.

[BLB00] Ahto Buldas, Helger Lipmaa, Berry Schoenmakers. *Optimally Efficient Accountable Time-Stamping.* Public Key Cryptography '2000, volume 1751 of Lecture Notes in Computer Science, pages 293-305, Melbourne, Australia, 18--20 January 2000. Springer Verlag.

[Bal98] Ahto Buldas, Peeter Laud, Helger Lipmaa, Jan Willemson. *Time-Stamping with Binary Linking Schemes.* Advances in Cryptology --- CRYPTO '98, volume 1462 of Lecture Notes in Computer Science, pages 486-501. Springer-Verlag, 1998.

[BM94] J. Benaloh and M. de Mare. *One-way accumulator: A decentralized alternative to digital signatures.* In Tor Helleseth, editor. Advances in Cryptology. Proceedings of Eurocrypt'93, number 765, pages 274-285. Springer-Verlag, 1994.

[ES00] C. Ellison and B. Schneier. *Ten Risks of PKI: What you are not being told about PKI.* Computer Security Journal. Vol XVI, Number 1,2000.

[HS91] S. Haber and W. Stornetta. *How to timestamp a digital document.* Journal of Cryptology,vol 3(2), pages 99-112,1991.

[MOS97] A. Menezes, P. Oorschot and S. Vanstone. *Handbook of Applied Cryptography,* CRC Press, 1997.

[MQ97] H. Massias, J.J. Quisquater. *Time and Cryptography.* Technical report, TIMESEC project, 1997.

[Pal98] B. Preneel, B. Van Rompay, J.J. Quisquater, H. Massias, J. Serret Avila. *Design of a timestamping system.* Technical report, TIMESEC project, 1998.

[Qal99] J.J. Quisquater, H. Massias, J. Serret Avila, B. Preneel, B. Van Rompay. *Specification and Implementation of a timestamping system.* Technical report, TIMESEC project, 1999.

[RSA78] R. Rivest, A. Shamir, L. Adleman. *A method for obtaining digital signatures and public-key cryptosystems.* Communications of the ACM, 21, pages 120-126. 1978

[San99] Tomas Sander. *Efficient Accumulators without Trapdoor.* In the second International Conference on Information and Communication Security, Sydney, Australia, 9-11 November 1999.

[Sur] Surety Technology. http://www.surety.com