Institut Eurécom
Department of Corporate Communications
2229, route des Crètes
B.P. 193
06904 Sophia-Antipolis
FRANCE

Research Report RR-05-125

A Distributed Time Stamping Scheme

January 6 2005

Bonnecaze Alexis, Liardet Pierre, Gabillon Alban, Blibech Kaouther

A. Bonnecaze is with CNRS-EURECOM FRE2660, Department of Corporate Communications and LIF, Université de Provence, France,
P. Liardet is with LATP, UMR CNRS 6632, Université de Provence, France,
A. Gabillon and K. Blibech are with CSySEC/LIUPPA, Université de Pau, IUT de Mont de Marsan, rue du Ruisseau BP 201 40004 Mont de Marsan Cedex, France.

Tel : (+33) 4 93 00 26 26
Fax : (+33) 4 93 00 26 27
Email : alexis.bonnecaze@eurecom.fr

[1]Institut Eurécom's research is partially supported by its industrial members: Bouygues Télécom, Fondation d'entreprise Groupe Cegetel, Fondation Hasler, France Télécom, Hitachi, ST Microelectronics, Swisscom, Texas Instruments, Thales

Abstract

In this paper, we define a trusted reliable distributed time stamping scheme. This scheme is based on a network of servers managed by administratively independent entities.

keywords: Time Stamping System, Distributed System, One Way Accumulator.

1 Introduction

The advent of electronic commerce have made the security of communication a major concern. Many governments have chosen to communicate and conduct transactions with citizens, businesses, and other agencies in a secure online environment. The security requirements for electronic document exchange are to ensure the integrity of official communications, to protect constituent privacy, to authenticate people and processes and possibly, to control sensitive information.
Digital signatures help to provide ongoing assurance of authenticity, data integrity, confidentiality and non-repudiation. Time-stamping techniques allow us to certify that an electronic document was created at a certain date. This certification is mandatory for a lot of applications in various domains like patent submissions, intellectual property or electronic commerce.
The aim of this paper is to study a new time-stamping scheme. Our scheme involves n independent servers and relies on two main protocols $\mathcal{S}$ and $\mathcal{V}$. The first one, called time-stamping protocol, constructs the time-stamp certificate. The second one must be executed for verification.
The first time-stamping protocol was presented during Crypto '90 by Haber and Stornetta. One year later, Benaloh and de Mare proposed a formal definition for a time-stamping system based on a set of participants and three protocols [2]. Since then, a lot of new schemes were proposed and their security analysed [10],[4],[5],[9],[11]. Most of them use the concept of trusted Time-Stamping Authority (TSA) which is supposed to be able to securely time-stamp an electronic document. According to the Internet X.509 PKI Time-Stamp Protocol (rfc3161), the TSA is required:

1. to use a trustworthy source of time,
2. to include a trustworthy time value for each time-stamp token,
3. to include a unique integer for each newly generated time-stamp token.

The concept of trusted third party offers a solution to user's immediate needs. However, it may be difficult to build a third party server that can be trusted. Indeed a server may be corrupted or victim of denial of service attacks. In fact, we claim that time-stamping schemes relying on a unique third party server, cannot be trusted. Therefore our objective in this paper is to propose a time-stamping scheme using a multiserver architecture. Our protocol can be shortly described as follows:
The protocol uses n third party servers. For each time-stamping request, k servers among the n servers are randomly chosen to process the request. These k servers are said to be the active servers.
The security of our protocol depends on the number n of servers and on the number k of active servers.
The paper is organized as follows. Section 2 briefly analyses the weaknesses of existing protocols. In Section 3, we give the required properties of our model. In

Section 4, we analyse the "k among n" scheme: among the n servers of the system, only k are chosen at random to handle a given time-stamping request. Our time-stamping scheme is presented in section 5 and finally we propose a new random generator to obtain k servers among n.

2 Existing protocols and their weaknesses

Most of the existing systems rely on a centralized server model that has to be trusted. For making the server trustworthy and preventing it from forging fake time-stamping tokens, the method generally used is to link the tokens in a chronological chain. Periodically, a token is published on an unalterable and widely witnessed media like a newspaper. This scheme offers the following advantages:

- The publication provides us with an absolute time.
- After a token has been published at time t, the server cannot forge a fake time-stamping token former to time t.
- Since tokens are linked in a chronological chain, we can chronologically order the requests submitted between two publications.

However, this scheme has the following drawbacks:

- The publication step is costly and not convenient.
- Before the next publication, the server can tamper the tokens which have been issued since the last publication.
- The entire chronological chain must be stored for verification. In order to reduce the amount of information to be stored, most of the protocols use a binary tree structure also called Merkle Tree (recall that a Merkle Tree is a construction introduced by Ralph Merkle in 1979 [7] to build secure authentication and signature schemes from hash functions). This method allows us to reduce by a logarithmic factor the amount of information to be stored. However, protocols using linking informations are not always accurate and efficient. This is trivially the case when the number of time-stamped documents is very small while the frequency of publication is very low (typically a week). In that case, the accuracy of the time-stamp may not satisfy the client. Notice also that a scheme using a binary tree is not efficient when the number of documents is not close to a power of 2.
- Finally, centralized systems are very vulnerable to Denial of Service attacks.

3 Design requirements

Our aim is to design a multi-server time-stamping system which has to respect the following requirements:

1. being independent from any administrative entity (like a country, a multinational company,...);
2. being resistant against a Distributed Denial of Service (DDoS);
3. being resistant against material failures;
4. being robust against an attack involving less than $n/3$ servers. It is known that any protocol can be made provably secure (without any cryptographic assumptions) if and only if less than one third of the involved parties are corrupted;
5. being able to work without ever trusting a particular componant of the system;
6. being able to deliver an absolute time with an *a priori* fixed error of Δt;
7. being able to prove the datation, from the knowledge of the only time-stamp;
8. having a robust, simple and efficient verification protocol.

4 k among n scheme

In this section, we discuss the security of a general scheme lying on a distributed network of n servers where only k servers are involved in the calculation of a particular time-stamp. In the next section, we present our distributed scheme which does not have the security flaws of the general scheme presented in this section.
The k servers are the **active** servers. They are randomly chosen. The two values n and k depend on the required security level.
The **complete time-stamp**, used to verify and to prove the datation, is built from the k time-stamping **fragments** delivered by the active servers. The number of active servers is defined in order to maintain the required security level as well as to minimize the traffic inside the network.
The model has n servers. Among them, f are supposed to be failed (fs). Among these f servers, we assume that f_m servers are in Malicious Collusion (MC). Their aim is to create time-stamps with the same incorrect time. Each of the other $f_e = f - f_m$ servers independently delivers a fake time-stamp without colluding.
For a given document, active servers are randomly chosen in order to reduce the probability of DDoS: the attacker must attack the whole n servers ($n >> k$) to be sure to succeed. Following Requirement 4 , we assume that the number f of failed servers is bounded by $n/3$.
Let us now focus on a configuration where k servers time-stamp a given document. Each of these servers issues a time-stamping fragment and a vote allows them to obtain a certified complete time-stamp: a complete time-stamp is certified when

more than $k/2$ servers propose the same date t. Of course, two servers may correctly time-stamp the document with two different but very close values. There exist many solutions to solve the problem of determining t from a cloud of values. We may assume that two values represent the same date if they belong to a fixed range Δt. Another solution is to ask the client to time-stamp its document locally and submit this time-stamp for acceptance to the distributed system. These methods do not affect the security analysis of the scheme.
Let us now analyse the probability that an attack succeed, assuming that the servers are chosen in a uniform way. The probability P_0 that the k servers use the f failed servers is

$$P_0(f) := \frac{\binom{n-f}{k-f}}{\binom{n}{k}} \ (k \geq f).$$

However, all the f servers do not need to be used to forge a time-stamp. In the case where $f_m \geq k/2$, f_m MC servers, when active, may contribute to forge a certified stamp. The probability that $k/2$ MC servers be active (k pair) is

$$Q(f_m) := \begin{cases} 0 & \text{if } k > 2f_m; \\ \frac{\binom{f_m}{k/2}\binom{n-f_m}{k/2}}{\binom{n}{k}} & \text{otherwise.} \end{cases}$$

Simple calculation shows that this value is greater than P_0 if $k > 2f_m$. The graphs of Figure 1 compare probabilities $P_0(f)$ and $Q(f)$. They are calculated by Maple and are drawn continously because of the use of the Γ function instead of the factorial function ($n! = \Gamma(n+1)$). With the following numerical values : $n = 36$, $k = 18$, $f_m = 12 = f$, probability Q is about $3.1\ 10^{-2}$ ($P_0 = 1.4\ 10^{-5}$). Note that $P_0(k/2) = Q(k/2)$.

Obviously, $P_0(f)$ is stricly decreasing for $0 \leq f \leq k$ and $Q(f)$ is nondecresing in the range $[k/2, \ldots, n - k/2]$. We have indeed

$$\frac{Q(f)}{Q(f+1)} = 1 - \frac{(n-1-2f)k/2}{-f^2 + f((n-k/2)-1) + (n-k/2)}$$

with $-f^2 + f((n-k/2)-1) + (n-k/2) > 0$ for $-1 \leq f \leq (n-1)/2$.

For $k < n/2$, let us take the same values but with $k = 12$ instead of 18 (and $f = 12$). Calculation of $Q(12)$ gives a value close to $1/10$.

This scheme does not provide a satisfactory security since an attacker colluding with the MC servers could successfully backdate a document after a relatively small number of requests (about 10), without being discovered. This is due to the diffusion property of hash functions: a modification of one bit in the document leads to very different hashs. Therefore, the attacker may submit several times (almost) the same document until his request is processed by the MC servers.

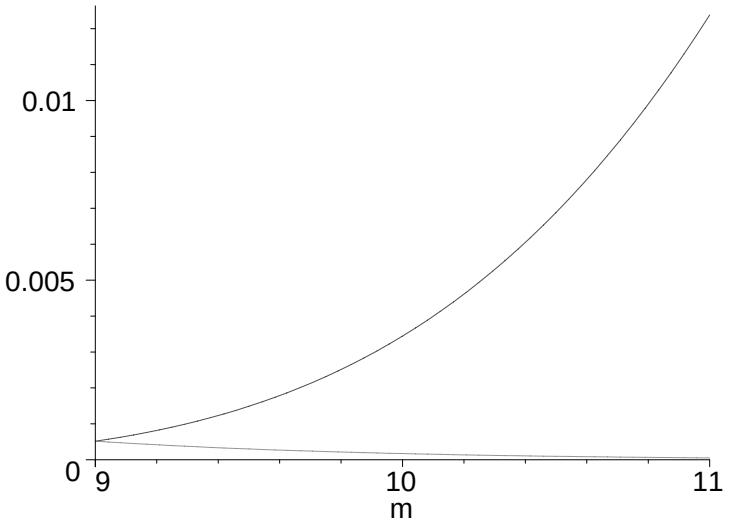

Figure 1: Graph of $\frac{\binom{m}{9}\binom{36-m}{9}}{\binom{36}{18}}$ and $\frac{\binom{36-m}{18-m}}{\binom{36}{18}}$, $m = 9...11$

5 A time-stamping scheme

In this section, we propose a time-stamping scheme which is not vulnerable to the attack presented previously. It is composed of internal time-stamping boxes and n servers managed by many independent entities. The idea of using distributed system to archive and sign documents has already been studied in [8]. Distributed time-stamping systems have also been studied, for example in [2] and [6]. However, none of these (rare) studies were able to design a secure and efficient system. Here, we aim to present a secure and reliable distributed scheme to time-stamp documents.

5.1 The time-stamping scheme

Each client c has a calculation box B to which she submits the document D to time-stamp. Two levels of security are provided. The time-stamping can either be performed locally by the box thanks to classical protocol (level 0), or by the distributed system of n servers (level 1). The box

1. calculates the hashed value of D, denoted $h(D)$;
2. level 0 : locally time-stamps the document; end of the protocol.
   level 1 : determines the k active servers;
3. signs the hash on behalf of the client c to form the request $r := (h(D))_c$;

4. sends the request r to the k servers and waits for an acknowledgment from each of them.

When level 1 is chosen, the document D is time-stamped in accordance with a scheme which can be described in the following way. Time is discretized in rounds of length Δt. Servers and clients are synchronized regularly and we do not take into account possible network malfunctions. Each round is identified by an absolute date. For example, a round can be identified by: January 1^{st} 2005 at 9.05am. During a round t, each server receives a number of requests which is approximately the same if this number is large enough, since active servers are chosen randomly. Suppose that the server S_i receives p_i requests. Let T_{S_i} denote the array formed by the p_i requests during the round identified by t. At the beginning of the next round (identified by $t+1$), the server S_i signs the concatenation of t with T_{S_i} to obtain $CS_{i,t}$. This is the stamp of the server S_i for the round t. Define $CS_{i,t} := (T_{S_i}||t)_{S_i}$. This stamp is then broadcasted to every node in the network. Hence, each server knows the list of all the distinct x requests $r_1, \ldots, r_x$, which have been submited during round t.

At the beginning of the next round (identified by $t+2$), the server S_i calculates the **global time-stamp** of the round t, CG_t and records it in its database. CG_t serves as a published value and is defined using one way accumulator functions that we have to define first.
Let Λ and E be two sets and define a family of maps

$$\begin{aligned} T_y : \quad & E \to E \\ & x \mapsto T_y(x) \end{aligned}$$

and dual maps

$$\begin{aligned} {}_xT : \quad & \Lambda \to E \\ & y \mapsto {}_xT(y). \end{aligned}$$

We assume that

i) ${}_xT$ is a one way function;

ii) for all a and b in Λ, $T_a \circ T_b = T_b \circ T_a$.

F defined as $F(x,y) := T_y(x)$ is an accumulator function in the sense of [3]. We denote by T_{ab} the composition $T_a \circ T_b$.
The global time-stamp CG_t of the round t is defined by

$$CG_t := h(T_\rho(t)),$$

where $\rho := r_1 \ldots r_x$. For each of its clients in the round t, the server calculates and signs the **client time-stamp**, which is composed by the following values:

$$r,\ t,\ CG_t,\ CG_{t,r} := T_{\rho_r}(t)$$

where r is the request of the client to whom the stamp is to be sent, t is the round identifier, CG_t is the global stamp and $\rho_r := r_1 \dots r_{r-1} r_{r+1} \dots r_x$. Hence, $CG_{t,r}$ is the accumulation of all the requests but r.

For a given document, each box receives k stamps. If all the requests are received by the server during round t, then all the k stamps are the same. However, it may be possible that some servers receive the request during round t while some others receive it during the round $t+1$. This situation does not constitute a problem. Indeed, the proof that the document has been time-stamped during round t can be done as long as at least one client time-stamp has been created.

5.2 Verification scheme

Server databases are to be consulted by verifiers. Each database record consists of the values $CG_t,\ r$, and t.
The verification of the time-stamp is as follows:

1. The request r has been involved in the construction of the global time-stamp if
$$CG_t = h(T_{\rho_r} \circ T_r(t)).$$
2. If required, the verifier can check that CG_t is the published value corresponding to the round t.

The accuracy of the datation depends on Δt, on the time t' of propagation in the network. The range of error of the time-stamping is then less than $\Delta t + t'$.

5.3 Robustness of the scheme

Attacks can be arranged into two categories:

- Attacks performed on existing time-stamps.
- Attacks performed during the construction of time-stamps.

The first type of attack consists in modifying the stamp while keeping its provability property. The success of such attacks depends on the robustness of the cryptographic functions that the system uses. The choice of the cryptographic functions is essential since time-stamps are to be valid for a long time (a few years).
Let us now study the robustness of our scheme against attacks of the second category. The number of failed servers being less than $n/3$, an audit is always able to detect either an error or an attack.
Backdating is impossible since the time-stamp would not be provable.
Postdating (which can be seen as a form of denial of service) is possible. It requires that the k active servers, in malicious collusion, wait during λ rounds before handling the request. The probability of such an attack is not negligible. It is equal

Attack	Possibility	Detection	Probability	range error
Antidating	NO	–	–	–
Postdating	YES	YES	$< \binom{n/3}{k}/\binom{n}{k}$	$+\Delta t$
DDoS	YES	YES	$< \binom{n/3}{k}/\binom{n}{k}$	$+\Delta t$

Table 1: Robustness of the scheme

to $P_{n,k} := \binom{n/3}{k}/\binom{n}{k}$. With the following parameters: $n = 24$, $k = 5$, we obtain a probability of $P_{24,5} = 1.3\ 10^{-3}$. In order to reduce the consequence of this attack, the client is requested to send again the time-stamping request to k new active servers when no acknowledgement has been received after a time of Δt. In this case, time-stamping is performed after a delay of Δt and the precision of the time-stamp is reduced. This solution also holds for a pure DoS attack. This study is resumed in Table 1, where the last column represents, in case of successful attack, the difference between the correct date and the date of the time-stamp.

Our protocol makes use of three types of cryptographic functions. Hash functions, signatures, and accumulators. Accumulator functions may be a simple modular exponentiation. In this case, we have $T_r(x) := x^r \mod s$, where the parameter s is an RSA modulus, and can be defined according to the recommandations of [3]. Notice that ${}_xT : r \mapsto x^r \mod s$ is a one way function since finding r is finding the discrete logarithm of ${}_xT(r) = x^r \mod s$ which is known as a hard problem. Moreover, the property $ii)$ also holds since $x^{ab} = x^{ba} \mod s$. With the previous notation, $CG_t := h(t^{\prod_{j=1}^{x} r_j} \mod s)$ and $CG_{t,r} := t^{\prod_{j=1}^{x} r_j / r} \mod s$. Hence the equation used for verification is now $CG_t = h((CG_{t,r})^r \mod s)$.
However, public key algorithms of RSA type being not efficient, accumulators based on this technique may not represent a viable solution. We recommand to use new algorithms like XTR (Efficient Compact Subgroup Trace Representation) or algorithms based on elliptic curves. In the case of elliptic curves, we use an additive group instead of a multiplicative group. We have $T_r(t) := r.t$, where t is a point of the curve and r an integer. In this case, accumulators and signatures may use common parameters (same curve, same field, ...) in order to simplify the scheme. This study, furthermore very interesting, is outside the scope of this paper.

5.4 A mixed architecture

Our scheme is based on two main protocols. A local time-stamping protocol and a distributed one. This "mixed architecture" represents an interesting aspect of the scheme since it has some valuable advantages for practical applications:

1. in order to reduce costs, a company may want to locally time-stamp some documents and use the distributed protocol as a publication. In this case, the local protocol is a classical linking scheme and regularly, a round stamp is time-stamped and published;

2. a mobile device may use its local protocol while it is off-line and use the

distributed protocol as soon as the connection to the system is available;

3. a client which the network has failed is able to time-stamp locally its document, waiting the connection to the distributed system;

4. a high secure system which is never connected to external systems may locally time-stamp its documents and regularly submit (in a secure way) a round stamp to the distributed system.

This list of applications is not exhaustive.

6 Random generators

Our scheme needs a generator to select k distinct elements from a set E of n elements at random. Hence, we seek for a uniform generator, cryptographically secure and satisfying some requirements particularly on the time of execution but also on the memory space.
There exist severals generating algorithms. The simplest way to build such a uniform random generator is to select an element $a_1 \in E$ and then select an other element $a_2 \in E \setminus \{a_1\}$ and repeat the process until we get k elements. This method is not always time efficient. Some algorithms are based on Markov chains issuing from random walks on a finite group G. We refer the interested reader to the basic surveys of N. Sloane [14] and D. Aldous [1].
One of the most famous generating algorithm is probably RANKSB of S. Wilf (see [13] for details). This algorithm is suitable for us as it is fast and not greedy.

7 Conclusion

Our scheme represents an alternative solution to classical monoserver schemes. The level of security and reliability can be achieved by carefully adjusting the k and n parameters. The distributed publication allows us to avoid a costly publication in a (not electronic) newspaper.
This scheme also offers the possibility to time-stamp documents in an off-line mode. In that case, the system may locally adopt a classical linking scheme and the publication would be done by the n servers when on-line. It may find applications where clients are not to be continuously connected to the system like, for example, in spontaneous networks.

8 Acknowledgement

This work was supported by ACI Sécurité Informatique CHRONOS (http://acisi.loria.fr/). The first author would like to thank Dr. J.-C. Pazzaglia (Institut Eurécom) for helful discussions.

References

[1] D. Aldous, *Random walks on finite groups and rapidly mixing Markov chains*, Séminaire de Probabilités XVII (1981/82), 243–297. Lecture Notes in Math. 1059, Springer, Berlin (1983), 245-255. .

[2] J. Benaloh and M. de Mare *Efficient Broadcast time-stamping* Technical Report 1, Clarkson University Department of Mathematics and Computer Science, August 1991. url = ”citeseer.ist.psu.edu/benaloh91efficient.html

[3] J. Benaloh and M. de Mare *One-Way Accumulators: A Decentralized Alternative to Digital Signatures* Advances in Cryptology–EUROCRYPT’93. LNCS, vol.765, pp.274–285, Springer–Verlag, 1994.

[4] A. Buldas, P. Laud, H. Lipmaa and J. Villemson *Time-stamping with Binary Linking Schemes*, Advances on Cryptology — CRYPTO ’98, Lecture Notes in Computer Science, Springer-Verlag, (1998), 486–501.

[5] A. Buldas and H. Lipmaa *Digital Signatures, Timestamping and the Corresponding Infrastructure*, url = citeseer.ist.psu.edu/article/buldas98digital.html

[6] Stuart Haber and W. Scott Stornetta *How to Time-Stamp a Digital Document*, Journal of Cryptology: the Journal of the International Association for Cryptologic Research 3(2), pages 99-112, 1991.

[7] R. Merkle, *Secrecy, authentication, and public key systems*, Ph.D. dissertation, Dept. of Electrical Engineering, Stanford Univ., 1979.

[8] P. Maniatis, T.J. Giuli and M. Baker *Enabling the Long-Term Archival of Signed Documents through Time Stamping* Computer Science Department, Stanford University, Technical Report, 2001, url = citeseer.ist.psu.edu/maniatis01enabling.html

[9] H. Massias and X. Serret and J. Quisquater *Timestamps: Main issues on their use and implementation* In Proceedings of IEEE 8th International Workshops on enabling Technologies: Infrastructure for Collaborative Enterprises - Fourth International Workshop on Enterprise Security, pages 178-183, June 1999. ISBN 0-7695-0365-9.

[10] H. Massias and J. Quisquater, *Time and cryptography* Technical report, Universit’e catholique de Louvain, March 1997. TIMESEC Technical Report WP1.

[11] M. Just *Some Timestamping Protocol Failures*, url = ”citeseer.ist.psu.edu/just98some.html”

[12] D.E. Knuth *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Addison-Wesley, reading Mass., second edition (1981).

[13] A. Nijenhuis and H.S. Wilf, *Combinatorial Algorithms for Computers and Calculators*, Acad. Press, Inc., second ed. 1978.

[14] N.J.A. Sloane, *Encrypting by Random Rotations*, Cryptography Proceedings of the Workshop on Cryptography, Burg Feuerstein, Germany, Edited by Thomas Beth, LNCS 149 (1983),71-128