Improving Time Stamping Schemes: A Distributed Point of View

A. Bonnecaze et P. Liardet et A. Gabillon et K. Blibech [†]

A. Bonnecaze is with CNRS-EURECOM FRE 2660, Department of Corporate Communications, 2229 route des Crètes, B.P. 193, 06904 Sophia-Antipolis, France and LIF, Université de Provence, France, alexis.bonnecaze@eurecom.fr.
P. Liardet is with LATP, UMR CNRS 6632, Université de Provence, 39 rue F. Joliot-Curie, 13453 Marseille cedex 13, France, liardet@gyptis.univ-mrs.fr.
A. Gabillon and K. Blibech are with CSySEC/LIUPPA, Université de Pau, IUT de Mont de Marsan, Département GTR 371, rue du Ruisseau BP 201 40004 Mont de Marsan Cedex, France, {alban.gabillon, kaouther.blibech}@univ-pau.fr.

Time stamping is a technique used to prove the existence of a digital document prior to a specific point in time. Today, implemented schemes rely on a centralized server model that has to be trusted. We point out the drawbacks of these schemes, showing that the unique serveur represent a weak point for the system. We propose an alternative scheme which is based on a network of servers managed by administratively independent entities. This scheme appears to be a trusted and reliable distributed time stamping scheme.

Mots-clés: Time Stamping System, Distributed System, One Way Accumulator, Hash Function, Denial of Service

I 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.

Numerous time stamping schemes have already been proposed and implemented. 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 [BenMar91]. Since then, a lot of new schemes were proposed and their security analysed [MasQui97], [BulLauLipVill98], [BulLip], [MasSerQui99], [Jus]. 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. The Network Working Group specifies, in the Internet X.509 PKI Time-Stamp Protocol (rfc3161), an Internet standards track protocol for time-stamping. This RFC describes the format of a request sent to a TSA and of the response that is returned. It also establishes several security-relevant requirements for TSA operation, with regards to processing requests to generate responses. According to this RFC, the TSA is required:

1. to use a trustworthy source of time,
2. to include a trustworthy time value for each time-stamp token,

[†]This work was supported by ACI Sécurité Informatique CHRONOS (http://acisi.loria.fr/).

3. to include a unique integer for each newly generated time-stamp token,

4. to produce a time-stamp token upon receiving a valid request from the requester, when it is possible,

5. to only time-stamp a hash representation of the datum.

As the first message of this mechanism, the requesting entity (called here the client) requests a time-stamp token by sending a request to the Time Stamping Authority (TSA). As the second message, the Time Stamping Authority answers by sending a response to the requesting entity. The TSA must sign each time-stamp message with a key reserved specifically for that purpose. A TSA may have distinct private keys, e.g., to accommodate different policies, different algorithms, different private key sizes or to increase the performance.

Most of the protocols use the concept of trusted third party even though 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 (DoS). Moreover, the problem may not have a malicious origin but a hardware or software origin. As we will see in the next section, one of the aim of existing protocols is to prevent the server from failing.

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 describes existing protocols and their security. The main drawbacks of these protocols are pointed out and we formalize the notion of accumulated functions which are used in some linking schemes. 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 out of n.

II Existing protocols and their security

The International Organization for Standardization edited a draft which mainly describes a general model on which time-stamping services are based. This work is divided into three parts (ISO/IEC 18014 Parts 1 to 3): the framework, mechanisms producing independant tokens and mechanisms producing linked tokens. Entities involved in a time-stamping scheme are a time-stamping authority (TSA) which is a trusted third party, a requester and a verifier. A TSA provides evidence that a document existed at a certain date. The requester is the entity applying for a time-stamp token and the verifier is the entity confirming the validity of the token.

In this paper, we classify time stamping schemes into three classes: simple, linking and distributed schemes.

II.1 Simple schemes

In the simple scheme, the time-stamping protocol can be described as follows:

- An entity (the time-stamping requester) who wishes a time-stamp token regarding document D transmits to the TSA a request message including a hash value $h(D)$ of D.

- The TSA generates a time-stamp token TS which includes at least the values $h(D)$, T, S and ID, where T represents the time, ID the identifier of the authority, and S the autority's signature on data that includes $h(D)$, T, ID.

- The authority sends TS to the requester.

During the verification process, a verifier has to compute the hash value of D and compare it with the value $h(D)$ included in TS. The verifier also verifies the signature S.

It is important to note that the time-stamp does not include any information regarding an other time-stamp. Thus the security of the method depends on authority's reliability. Nobody can detect a possible alteration of the token made by a malicious TSA. Different services are using a simple time-stamping scheme. Among them, the french company "La Poste" (its time-stamping service is developped by *Keynectis*), or some services available from Internet (like OpenTSA, e-timestamp or authentidate).

II.2 Linking schemes

Linking schemes have been developped in order to be able to detect fake time-stamps produced by a malicious TSA. In its simpler configuration, the method consists in linking the tokens chronologically, forming a chain [HabSto91]. It means that data related to the preceding tokens are included in the current token. The TSA may be implemented so that it maintains a database with two sequences of values, the A values and the S values. For a given time-stamp request at time $T(i)$, the TSA computes a *time-stamp info object* which includes time value, hash of the document, serial number and TSA name. Then it computes the following two values:

- $A(i)$, computed by hashing the timestamp info object, and
- $S(i)$, computed by hashing over inputs $A(i)$, $S(i-1)$ and possibly multiple other previous values $S(j), \;\; j < i-1$.

The TSA adds $A(i)$ and $S(i)$ to its database of A and S values. It then generates the timestamp token, encapsulating in a digested data object the timestamp info object and the values used for computing $S(i)$. The timestamp token is returned to the requester over a channel providing data integrity and origin authentication (for example, by using keys that do not need to last longer than the transaction lapse).

Periodically (every day or week), a token is published on an unalterable and widely witnessed media like a newspaper. The process of publication must be done in an authenticated manner.

The verification protocol consists in the following steps. First, recomputes a series of hashed values starting from the preceding published value. check whether the value $A(i)$ contained in the time-stamp token matches the time-stamp info object. Then try to reconstruct $S(i)$ from $A(i)$ and $S(i-1)$ (and possibly more S values). The value $S(i)$ must be the one of the TSA's database. After having compared the claimed linking information with the recomputed value, the recomputation goes on to the next published value. The verification succeds when the next published value is correct. Thus the verification is sandwiched by two publications.

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, one 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.
- Finally, centralized systems are very vulnerable to Denial of Service attacks.

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 [Mer79] to build secure authentication and signature schemes from hash functions). These methods allow us to reduce to a logarithmic factor the amount of information to be stored. They use the notion of round: a round can be a given number of requests, a period of time or a combination of both. During a round r, the TSA receives a certain number of requests (say x') and constructs a tree using hashes functions (like SHA-1 or MD5). Hashes of documents $H_1 \dots H_{x'}$ represent the leaves of the tree. However, in order to make the tree balanced, the number of leaves must be a power of 2. Let l be the smallest integer such that $x' \leq 2^l$ and let define $x := 2^l$. Then, $H_1 \dots H_x$ represent the leaves of the tree where $H_{x'} \dots H_x$ are some "padding" values used to obtain the required number of leaves. At the first step, TSA makes $x/2$ pairs from these leaves and hashes them to obtain $H_{(2i+1)(2i+2)} := h(H_{2i+1}||H_{2i+2})$, for $i = 0 \dots x/2-1$. where the function h represents a hash function and $||$ represents the concatenation. More generally, at step j, the TSA makes $x/(2^j)$ pairs by the same process, obtaining

$$H_{((2^j)i+1)((2^j)i+2)((2^j)i+3)\dots((2^j)i+2^j)} := h(H_{((2^j)i+1)\dots((2^j)i+2^{j-1})}||H_{((2^j)i+2^{j-1}+1)\dots((2^j)i+2^j)}),$$

where $i = 0 \dots x/2-1$. At step $Log_2(x)$, TSA computes by the same way the *Root Hash Value* of the round r: RHV_r and at step $Log_2(x)+1$, TSA obtains the token of the round r: $TS_r := h(TS_{r-1}||RHV_r)$, where TS_{r-1} denotes the time-stamp token of round $r-1$. Figure 1 illustrates the process for the case $x = 8$. A

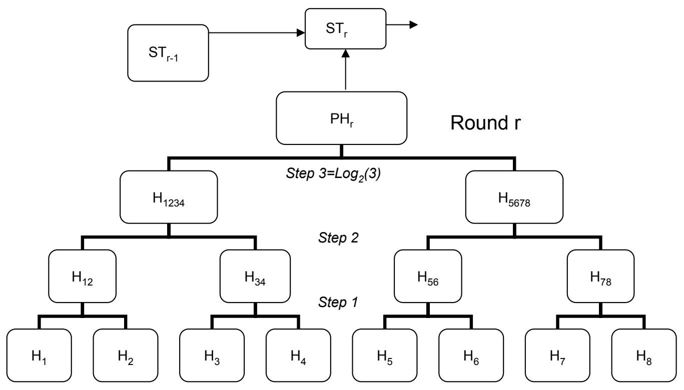

Fig. 1: Simple binary tree structure

time-stamp of H_i includes the date T_i, the identity of the token ID_i, the hashed document H_i to be time-stamped and L_i which contains the values used for computing RHV_r from H_i. For example L_1 must contain the values H_2, H_{34}, and H_{5678}. The TSA adds TS_r to its database and Periodically publishes the the tokens of the rounds. Note that this scheme does not allow to know the order of requests processed in a same round. Depending on the implementation, T_i may not be exactly equal to the time of the round.

The verification is as follows: first, check whether the hash of the document equals the value H_i included in the time-stamp. Second, check the consistency between the value TS_r stored in the TSA's database and the one generated from the time-stamp token.

There exist some other schemes based on trees which are slightly different like the one proposed by Buldas et al. [BulLipSch2000]. This scheme uses a special tree called *threaded authentication tree* which makes the time-stamping protocol more complex. The aim of this scheme is to provide a relative datation of documents. Time-stamp tokens do not include a time parameter but an evidence of their chronological order. They provide two informations: the identity of the round they belong and their order in this round. The verification procedure consists in deciding, given two time-stamp tokens, which has been generated first. As usual, hash values of documents and signatures must be checked and some values are to be reconstructed and compared to the ones included in the token and the ones published in a newspaper. Thus, a verifier does not have to communicate with the TSA in order to carry out the verification.

Notice that a scheme using a balanced binary tree is not efficient when the number of documents is not close to a power of 2. The worst case being reached when this number is 2^n+1. Even though linking schemes do not have to use a tree structure, it is hard to find in the literature an other structure. In fact, tree structure is not always accurate and efficient. This is 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 who wishes an absolute datation.

Recently, Blibech et al [BliGab05] proposed to use a new structure called skip-list (originately developped by Bill Pugh [Pug90] as an alternative to balanced trees). A skip-list is a probabilistic data structure that can be used in place of a balanced tree. Algorithms for insertion and deletion in skip lists are simpler and faster than equivalent algorithms for balanced trees.

Apart from the preceding schemes, there exists an other linking scheme which is based on the use of accumulator functions [BendeMar93]. Accumulator functions represent a very interesting (algebraic) alternative to these above data structures. Using these functions, the verification process can be done in just one operation. Moreover, the amount of information that has to be stored does not depend on the number of time-stamped documents. For Benaloh and de Mare, accumulator function has two properties. It is a one way function which is *quasi commutative*. A function $h : X \times Y \to X$ is quasi commutative if for all $x \in X$ and for all $y_1,\ y_2 \in Y$,

$$h(h(x,y_1),y_2) = h(h(x,y_2),y_1).$$

In this paper, we prefer to define a one way accumulator in the following way:
Let Λ and E be two sets and define a map $F : E \times \Lambda \to E$, called a kernel accumulator which defines the following two dual families of maps

$$\begin{array}{rrl} T_y : & & E \to E \\ & x \mapsto & T_y(x) := F(x,y) \end{array}$$

and

$$\begin{array}{rrl} {}_xT : & & \Lambda \to E \\ & y \mapsto & {}_xT(y) := F(x,y). \end{array}$$

Note that ${}_xT$ is defined to be a one way function. For any word $a = a_1 \ldots a_s$, $a_i \in E$, we set $T_a = T_{a_1} \circ \cdots \circ T_{a_s}$. Suppose that l requests $r_1, \ldots, r_l$ have been received during round t. The global time-stamp token CG_t of the round t is defined by

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

where $\rho := r_1 \cdots r_l$, and the time-stamp token corresponding to the request r_i is defined by:

$$CG_{t,r_i} := T_{\rho_{r_i}}(t),$$

where $\rho_{r_i} := r_1 \cdots r_{i-1} r_{i+1} \cdots r_l$. Hence, CG_{t,r_i} is the accumulation of all the requests but r_i. The verification is very simple: the verifier just has to check whether the following equality holds

$$CG_t = T_{\rho_{r_i}} \circ T_{r_i}(t).$$

For time-stamping purposes, modular exponentiation is the only used accumulator function. This is so because no better function has been found since then. The scheme can be described as follows:
first, an integer s has to be selected and agreed by all parties (this value plays the role of an RSA modulus, and can be defined according to the recommandations of [BendeMar93]). Then a started value x_0 has to be chosen. This value may possibly be a representation of the current date. Let us now define $x = x_0^2 \mod s$. Using the same notations as above, we define $T_{r_i}(x) := x^{r_i} \mod s$. In particular, we must have $(r, \phi(s)) = 1$, where ϕ denotes the well known Euler function. Notice that ${}_xT\ :\ r_i \mapsto x^{r_i} \mod s$ plays the role of a one way function since finding r_i is finding the discrete logarithm of ${}_xT(r_i) = x^{r_i} \mod s$ which is known as a hard problem. Moreover, the equality $x^{ab} = x^{ba} \mod s$ also holds. With the previous notation, let us define the values $z := \prod_{j=1}^{x} r_j$ and $z_i := \prod_{j=1}^{x} r_j / r_i$. Then, the global and partial time-stamp tokens are $CG_t := t^z \mod s$ and $CG_{t,r_i} := t^{z_i} \mod s$. The equation used for verification is now

$$CG_t = (CG_{t,r_i})^{r_i} \mod s.$$

A party who would like to demonstrate the correctness of the datation needs only produce the values z and z_i. In their paper, Benaloh and de Mare discuss the security of such a method. They conclude, saying: *the only claim which we can make about forgery is that a user cannot produce a valid time stamp for a document that was not anticipated at the time indicated by the stamp.* They also notice that forging unanticipated documents is infeasible. An important drawback of the scheme is that the generation of the RSA modulus must be done by a trusted third party.

Protocols based on simple binary tree have been really used in existing projects or services. Academic or national projects abroad tended to use linking protocols as follows.

- TIMESEC (Federal Office for Scientific, Technical and Cultural Affairs in Belgium: 1996-1998) developped a linking protocol using tree structure with hash functions such as SHA-1 and RIPEMD-160.
- Cuculus (Estonia: 1997) proposed a linear linking protocol based on hash function and digital signature.
- PKITS (Public Key Infrastructure with Time Stamping Authority is a project of European Trusted Services, 1997-1998) deploys a linking protocol in which not a single but multiple TSAs operate. Each of the TSAs periodically sends its linking information to an other randomly selected TSA in order to obtain a time-stamp of the linking information. Digital signature is not used.

Generally, commercial services use simple protocols. Note however that the protocol of *Digital Notary* (Surety.com, USA: 1992-) is a linking protocol based on a tree structure.

All these methods, based on Merkle tree, take aim at preventing the TSA from forging a time-stamp. If a TSA is willing to maliciously alter a certain time-stamp, it has to alter all the tokens related to it. However, the TSA may be the victim of a denial of service attack or of a breakdown of hardware/software origin. It therefore represents a single point of failure in the scheme. Moreover, these schemes may not be scalable and may not provide a satisfactory time granularity. Reducing the size of a round may increase the cost of a token.

II.3 Distributed scheme

Unavailability and loss of data problems can be solved using a multiserver scheme. Hence, multiserver time-stamping schemes represent a very interesting alternative to mono-server schemes even though they have not been studied a lot. Distributed schemes are composed by a set of servers which may be managed independently. They are more robust against breakdowns or network attacks. Moreover, depending on the scheme, the capacity of the system in term of number of requests may be improved. The idea of using a distributed system is not new. A distributed solution to archive documents has been studied in [ManGiuBak01] and distributed time-stamping systems have also been studied, for example in [BenMar91], [HabSto91] or [Tak99]. However, these studies were neither able to design a secure and efficient system, nor able to give a viable solution to the problem of denial of service.

Designing a distributed scheme, one has to solve specific problems like:

1. how to select the active servers?
2. how many servers should be selected?
3. should the servers be synchronized?
4. how to minimize the number of interactions between the servers?
5. how to construct the time-stamp from the received token fragments?
6. how to minimize the size of the time-stamp? (this size should not depend on the number of servers)

In Section V, we propose a protocol which represents a solution to these problems. More generally, we think that modern cryptography has the potential to provide efficient solutions to some of the above questions. This will probably be an attractive topic of research in the near future.

III Design requirements

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

1. resistance against a Distributed Denial of Service (DDoS);
2. resistance against material failures;
3. robustness 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;
4. the security must not depend on a particular componant of the system;
5. the system has to deliver an absolute time with an *a priori* fixed error of Δt;
6. the system must be robust, simple and with an efficient verification protocol.

IV 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 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 even) 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 2 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 $[k/2, \dots, n - k/2]$. We have indeed

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

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

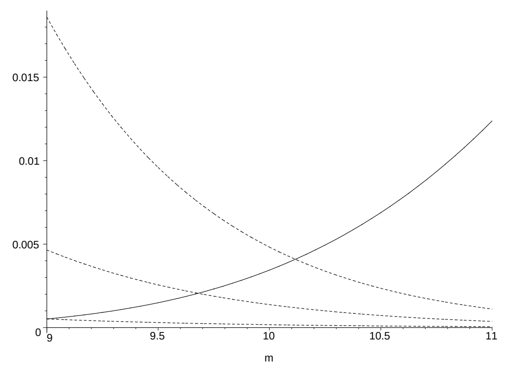

Fig. 2: Graph of $\frac{\binom{m}{9}\binom{36-m}{9}}{\binom{36}{18}}$ and $\frac{\binom{36-m}{18-m}}{\binom{36}{18}}$, $m = 9 \dots 11$

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 hashes. Therefore, the attacker may submit several times (almost) the same document until his request is processed by the MC servers.

Even though k among n scheme cannot be used alone to design secure time-stamping schemes, it can however be used in combination with linking methods. This is the subject of the next section.

V A time-stamping scheme

In this section, we propose a time-stamping scheme which is not vulnerable to the attack presented previously. The main parties involved in the scheme are the following:

- **The client** needs to time-stamp some documents.
- **The box** is given to each client. Depending on the required security level, it can either locally time-stamp a document or randomly determine k active servers for a distributed time-stamping.
- **The network of servers** builds a time-stamp for each document, using time-stamping protocol $\mathcal{S}$.
- **The replicated database** serves as a publication media. Each server of the network records the time-stamps of all the clients. Therefore each server holds an entire copy of the time-stamp database.
- **The verifier** runs verification protocol $\mathcal{V}$ in order to check the correctness of the time-stamp of a given document.

Next paragraph describes in details this scheme.

V.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 randomly 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 are synchronized regularly with reliable temporal sources. 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 $r_1, \dots, r_x$, 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 have been defined in subsection II.2.

The global time-stamp CG_t of the round t is defined by

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

where $\rho := r_1 \dots r_x$. 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:

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

where r_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_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.

V.2 Verification scheme

Server databases are to be consulted by verifiers. Each database record consists of the values CG_t, r_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
$$CG_t = h(T_{\rho_r} \circ T_{r_r}(t)).$$
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 $\Delta t + t'$.

V.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 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 and the RSA modulus may be computed in a distributed way. In this case, using the same notation as above, we have $T_{r_r}(x) := x^{r_r} \mod s$, and, $CG_t := h(t^{\prod_{j=1}^{x} r_j} \mod s)$ and $CG_{t,r} := t^{\prod_{j=1}^{x} r_j / r_r} \mod s$. Hence the equation used for verification is

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

Tab. 1: Robustness of the scheme

now $CG_t = h((CG_{t,r})^{r_r} \mod s)$.

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.

V.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.

VI Random generators

Our scheme requests a generator to randomly select k distinct elements from a set E of n elements. Hence, we seek for a uniform generator, cryptographically secure and which satisfies 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 new element, distinct from the preceding one, and repeat the process until we get k elements. We consider two classical cases:

- Each selected element is dropped from the set E. Consequently, we have to build k random generators of q symbols, with $n-k+1 \leq q \leq n$. This leads to a possible bias if we use a binary source generator.
- Selected elements are kept in the set. In this case, only one generator is used and the problem of possible bias concerns just this generator. However, it may take a lot of draws before we obtain k elements since in our model, k is not very small compare to n. Therefore, the probability to draw an already selected element is not negligible. Recall that the average number of draws is approximately $n \log n$ (see for example [Knu81]).

One of the most famous generating algorithm is probably RANKSB of H. Wilf (see [NiWi78] for details). Its execution time, in average, is $O(k)$, when $k << n$. But in our case, it will be around $O(n)$ and execution time is not constant.

Now, we propose an algorithm based on the notion of random walks on a finite group G. We refer the interested reader to the basic surveys of N. Sloane [Sl83] and D. Aldous [Al82]. Let $Q^{(m)}$ be the distribution on G determined by the issue of the walk after m steps starting from the identity element. The initial distribution is Q (the probability to go from g to h in the group is $Q(g^{-1}h)$). Let U be the uniform distribution on G. When $Q^{(m_0)} > cU$ for an integer m_0 and a constant $c > 0$ (mixing case), the total variation distance $d(m) := \max_{A \subset G} |Q^{(m)}(A) - U(A)|$ between $Q^{(m)}$ and U tends to 0 at an exponential rate. But this majoration is generally not numerically useful. Consider the case where G is the permutation group of E. We choose the walk represented by the mixing of a deck of n cards by the following method: insert equally likely the topmost card within the deck. From [AlDi86] (Theorem 1), $d(n \log n + \gamma n) \leq e^{-\gamma}$ for all $\gamma > 0$ and $n \geq 2$. Let us apply this result for $n = 24$. An explicit version of Stirling formula gives $d(1393 + 17s) \leq \frac{1}{2^s 24!}$. This method does not provide us with a uniform generator, but the bias, represented here by the quantity $d(m)$, can be made negligible by using enough computing power. Moreover, the fluctuation of performances of the binary source generator can be corrected by increasing the number of iterations.

VII Conclusion

In this paper, we described the different methods to digitally time-stamp a document. Then, we evaluated the k among n scheme and finally, we proposed a new scheme based on both notions of k among n and linking schemes. 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.

References

[Al82] Aldous (D.), *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), pp. 245-255.

[AlDi86] Aldous (D.) and Diaconis (P.), *Shuffling cards and stopping times*, Am. Math. Monthly **93** (1986), pp. 333-348.

[BenMar91] Benaloh (J.) and de Mare (M.) *Efficient Broadcast time-stamping* Technical Report 1, Clarkson University Department of Mathematics and Computer Science, August 1991. url = "citeseer.ist.psu.edu/benaloh91efficient.html

[BendeMar93] Benaloh (J.) and de Mare (M.) *One-Way Accumulators: A Decentralized Alternative to Digital Signatures* Advances in Cryptology–EUROCRYPT'93. LNCS, vol.765, pp. 274-285, Springer-Verlag, 1994.

[BliGab05] Blibech (K.), Gabillon (A.), *Authenticated dictionary based on Skip Lists for time stamping systems*, submitted 2005.

[BulLauLipVill98] Buldas (A.), Laud (P.), Lipmaa (H.) and Villemson (J.) *Time-stamping with Binary Linking Schemes*, Advances on Cryptology — CRYPTO '98, Lecture Notes in Computer Science, Springer-Verlag, (1998), pp. 486-501.

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

[BulLipSch2000] Buldas (A.), Lipmaa (H.) and Schoenmakers (B.), *Optimally Efficient Accountable Time-Stamping*, In Yuliang Zheng and Hideki Imai, editors, Public Key Cryptography '2000, volume 1751 of Lecture Notes in Computer Science, pp. 293-305, Melbourne, Australia, January 18–20, 2000. Springer-Verlag.

[HabSto91] Haber (S.) and Stornetta (W.S.) *How to Time-Stamp a Digital Document*, Journal of Cryptology: the Journal of the International Association for Cryptologic Research 3(2), pp. 99-112, 1991.

[Knu81] Knuth (D.E.) *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Addison-Wesley, reading Mass., second edition (1981).

[Mer79] Merkle (P.), *Secrecy, authentication, and public key systems*, Ph.D. dissertation, Dept. of Electrical Engineering, Stanford Univ., 1979.

[ManGiuBak01] Maniatis (P.), Giuli (T.J.) and Baker (M.) *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

[MasSerQui99] Massias (H.) and Serret (X.) and Quisquater (J.) *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, pp. 178-183, June 1999. ISBN 0-7695-0365-9.

[MasQui97] Massias (H.) and Quisquater (J.), *Time and cryptography* Technical report, Université catholique de Louvain, March 1997. TIMESEC Technical Report WP1.

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

[Knuth81] Knuth (D.E.) *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Addison-Wesley, reading Mass., second edition (1981).

[NiWi78] Nijenhuis (A.) and Wilf (H.S.), *Combinatorial Algorithms for Computers and Calculators*, Acad. Press, Inc., second ed. 1978.

[Pug90] Pugh (W.), *Skip Lists: Skip lists: A probabilistic alternative to balanced trees*, Communications of the ACM, 33(6) pp. 668-676, June 1990.

[Sl83] Sloane (N.J.A.), *Encrypting by Random Rotations*, Cryptography Proceedings of the Workshop on Cryptography, Burg Feuerstein, Germany, Edited by Thomas Beth, LNCS 149 (1983), pp. 71-128

[Tak99] Takura (A.), Ono (S.), Naito (S.) *A Secure and trusted Time Satmping Authority* Proceedings of IWS 99, 1999, pp. 123-128.