TIMESEC

Digital Time-Stamping and the Evaluation of Security Primitives

This page presents the **Belgian Project TIMESEC** funded by the **OSTC** (Federal Office for Scientific, Technical and Cultural Affairs).

Contents

Project duration

TIMESEC is a project that started on August 1996, for a duration of 2 years.

Main objectives

Today, **computer and telecommunication securisation** is a major issue. Exchanging information requires several things such as: integrity, identification, non-repudiation, confidentiality,... which are necessary to each secure protocol to exchange messages. **Time-stamping** is of high importance for electronic contracts. It's important for EDI (Electronic Data Interchange which is essential for electronic trade), IPR (Intellectual Property Rights), interactive multimedia services (pay-TV, homeshopping, video online,...), and in general, Internet securisation.
The project is of strategic importance because up to now people who want to certify their techniques, their products, or their securisation machine in order to market them must export their products, their knowledge and their know-how to other countries (competitor). Our proposition will provide an answer to this paradoxical economic problem.
The method which will be develop during this project will be use to improve future evaluation; it will play a **major role in the standardization** and the **normalization of future systems** for information securisation. It seems to be essential to improve Belgian appraisal in this field, and to pave the way to a **Belgian evaluation center** designed according to these criteria.

Major expected outputs

The concrete objective of the project is to develop

- a complete system of digital time-stamping and electronic notarization
- as well as security primitives evaluation methods.

Deliverables

- **Time and cryptography (Technical Report 1), March 1997**
- **Evaluation Methodology for Security Primitives (Technical Report 2), December 1997**
- **Design of a Timestamping System (Technical Report 3), 1998**
- **Specification and Implementation of a Timestamping System (Technical Report 4), 1999**

Output documents

Presentation of progress (meeting Feb. 18, 1997)

- Evaluation methodology for security primitives (Bart Van Rompay, K.U.L.)
- Timestamping(Henri Massias, U.C.L.)

US-patent concerning Timestamping

Number	Title
5,136,646	Digital document time-stamping with catenate certificate
5,136,647	Method for secure timestamping of digital documents
5,373,561	Method of extending the validity of a cryptographic certificate
RE. 34,954	Method for secure timestamping of digital documents
5,781,629	Digital document authentication system

List of partners

- **extern members**
  1. Utimaco Belgium n.v.
  2. Belgacom
  3. Belnet
- **active partners**
  1. U.C.L.
  2. K.U.L.(Coordinator)

Contacts

1. **Ronny Bjones**, Director Technical Departement
   Utimaco Belgium n.v.
   Ambachtelijke Zone De Vunt 9
   3220 Holsbeek
   Belgium
   Phone: +32 16 44 01 35 - Fax: +32 16 44 01 40
2. **Marc Roger**
   Belnet
   Rue de la science 8
   1000 Bruxelles
   Belgium
   Phone: +32 2 238 34 70 - Fax: +32 2 231 15 31
3. **Olivier Delos**
   Belgacom
   Boulevard E.Jacqmain 177
   B-1210 Bruxelles
   Belgium

Phone: +32 2 202 92 76 - Fax: +32 2 202 84 52

4. **People working for TIMESEC at UCL:**

Prof. Jean Jacques Quisquater
UCL Crypto Group
Faculté des sciences appliquées
Université de Louvain
Place du Levant, 3
B-1348 Louvain-la-Neuve
Belgium
Phone: +32 10 47 25 41 - Fax: +32 10 47 25 98

Henri MASSIAS (research assistant)
UCL Crypto Group
Faculté des sciences appliquées
Université de Louvain
Place du Levant, 3
B-1348 Louvain-la-Neuve
Belgium
Phone: +32 10 47 81 39 - Fax: +32 10 47 25 98

5. **People working for TIMESEC at KUL:**

Bart Preneel (Coordinator)
Katholieke Universiteit Leuven
Departement Elektrotechniek-ESAT
Onderzoeksgroep COSIC
K.Mercierlaan 94
B-3001 Heverlee
Belgium
Phone: +32 16 32 11 48 - Fax: +32 16 32 19 86

Bart Van Rompay (research assistant)
Katholieke Universiteit Leuven
Departement Elektrotechniek-ESAT
Onderzoeksgroep COSIC
K.Mercierlaan 94
B-3001 Heverlee
Belgium
Phone: +32 16 32 11 34 - Fax: +32 16 32 19 86

Other related web pages

- Time Service Department
- Digital Notary
- Estonian Timestamping Project Cuculus
- PGP digital Timestamping Service
- PKITS(Public Key Infrastructure with Time Stamping)
- Network Time Protocol
- Time stamping links (author: Helger Lipmaa)

The access statistics of this page may be found here.

Goto : UCL | FSA | ELEC | DICE | CRYPTO GROUP

Last modified : August 17, 1999
Send any comment to : massias@dice.ucl.ac.be (H.Massias)

DESIGN OF A SECURE TIMESTAMPING SERVICE WITH MINIMAL TRUST REQUIREMENT

H. Massias, X. Serret Avila, J.-J. Quisquater
UCL Crypto group
Place du Levant, 3 , B-1348 Louvain-la-Neuve, Belgium
massias, serret, jjq@dice.ucl.ac.be

This paper presents our design of a timestamping system for the Belgian project TIMESEC. We first introduce the timestamping method used and we justify our choice for it. Then we present the design of our implementation as well as some of the important issues we found and the solutions we gave to them.

INTRODUCTION

The creation date of digital documents and the times expressed in them are becoming increasingly important as digital documents are being introduced into the legal domain.

We define "digital timestamp" as a digital certificate intended to assure the existence of a generic digital document at a certain time.

In order to produce fully trusted timestamps, very specific designs have been introduced. We give an overview of the most relevant methods and we introduce the one we used for the implementation of the Belgian project TIMESEC (see [PRQ+98]), justifying our choice for it. Then we present the design of the timestamping system we made for this project. We separate the different processes that are: document timestamping, timestamp verification, auditing, system start-up and system shutdown.

INTRODUCTION OF THE TIMESTAMPING TECHNIQUES

There are two families of timestamping techniques: those that work with a trusted third party and those that are based on the concept of distributed trust. Techniques based on a trusted third party rely on the impartiality of the entity that is in charge of issuing the timestamps. Techniques based on the distributed trust consist on making documents dated and signed by a large set of people in order to convince the verifiers that we could not have corrupted all of them. The trusted third party techniques can also be classified into two different kinds: those where the third party is completely trusted and those where it is partially

trusted. A detailed study of timestamping techniques can be found in [MQ97]. We believe that techniques based on distributed trust are not really workable in a professional environment, that is why we concentrate on the trusted third party approach. Nevertheless, we imposed to ourselves the requirement to lower the necessary trust on the third party to the maximum extend.

The "easy" solution, which consists on concatenating the document with the current time and sign the result, has been discarded because it has two main drawbacks:

1. We must completely trust the third party, called Secure Timestamp Authority (STA), which can issue undetectable back-dated timestamps.
2. The limited lifetime of cryptographic signatures, which can be shorter than the document time-to-life.

The timestamping method that we have chosen uses a binary tree structure and has been described in [HS91] and [HS97]. This method works by rounds. For each round a binary tree is constructed with the requests filled during it. The rounds have a fixed duration, which is the result of a trade-off between the timestamps accuracy and the number of requests submitted. In Figure 1 we can see a graphical representation of a round constructed using this method.

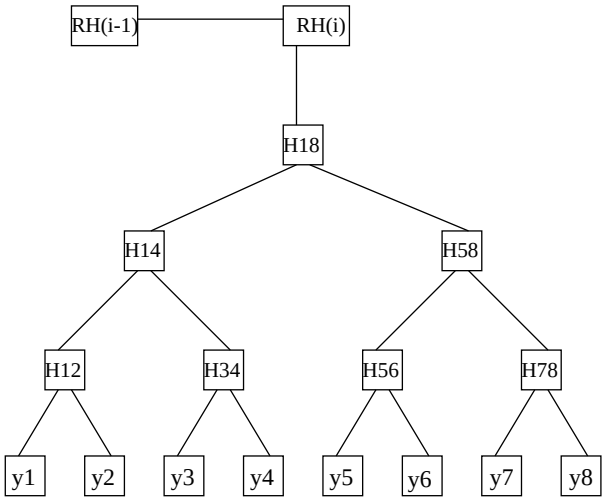


Figure 1: The binary tree structure

Each of the timestamp requests consists on a hash value of a given document. The leafs of the tree are each of those hash values. The leaf values are then

concatenated by two and hashed again to obtain the parent value (Ex: $H_{34} = H(y_3 \mid y_4)$). The process is repeated for each level until a single value is obtained. Finally, the top value of the round tree (H_{18}), called the "Round Root Value", is then concatenated with the value obtained for the preceding round (RH_{i-1}) and then hashed again to obtain the actual "Round Value" (RH_i).

The timestamp of the document contains all the values necessary to rebuilt the corresponding branch of the tree. For example, the timestamp for y_4 contains $\{(y_3, L), (H_{12}, L), (H_{58}, R), (RH_{i-1}, L)\}$. The verification process consists of re-building the tree's branch and the linking chain of "Round Values" until a trusted (from the verifier point of view) "Round Value" is recomputed. This verification method is explained in detail in [HS91] and [MQ97].

Periodically, one of the "Round Values" is published on an unmodifiable, widely witnessed media (Ex: newspaper...). These special "Round Values", which we will call "Big Round Values", are the base of the trust for all the timestamps issued. All verifiers must trust these "Big Round Values" as well as the time associated with them. This is a reasonable requirement because those values are widely witnessed. The absolute time trusted by all the potential verifiers is the time indicated by the unmodifiable media. We suppose that this time is the same than the time indicated by the STA for the "Big Round". Forcing the clients to check the timestamps as soon as they get them is another requirement. In that way the process is continuously audited and the STA will not have any margin to maneuver in an untrusted way.

A very useful method for extending the lifetime of timestamps is described in [BHS92]. It basically consists on re-timestamping the hash of the document as well as the original timestamp before the hash function is broken.

We build two trees in parallel for each round using two different hash functions (SHA-1 and RIPEMD-160). In that way, the system remains secure in the case of an unexpected break of one of the hash functions used.

DESCRIPTION AND ANALYSIS OF THE TIMESEC TIMESTAMPING IMPLEMENTATION

We will now introduce the basic design of the system we have developed, which is based on the technique introduced above.

Initially, the user designates a document to be timestamped. Two hashes of it are created using the SHA-1 and RIPEMD-160 algorithms. The request

containing the two hashes is then sent by the client to the STA . Upon request receipt, the STA creates the corresponding timestamp using the following process.

Main description of the timestamping process

The system design follows a highly decoupled multi-threaded approach. Each step is assigned to a specific component, which has its own different thread. In the Figure 2 we present a schematic outline of the process. The multi-thread approach is justified by the requirement to obtain a highly responsive and load independent implementation. By isolating the process charges into independent steps we try to decouple the load between them. Each step has also a working queue. Those queues are in charge of softening the speed differences between the different process steps.

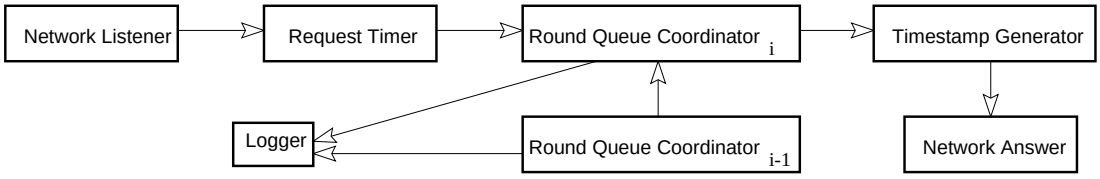


Figure 2: Interactions between the components

The "Network Listener" is in charge of continuously listen to the clients' timestamp requests. The "Request Timer" receives the constructed requests from the "Network Listener". Then, it times and forwards them to the actual "Round Queue Coordinator". Each round has its own "Round Queue Coordinator", which is in charge of compiling and processing into a tree all the requests belonging to the round. When the round tree has been computed it is forwarded to the "Timestamp Generator", which generates the corresponding timestamps. Once a timestamp is generated, the "Timestamp Generator" forwards it to the "Network Answer", which in turn forwards it to the client.

The Network Listener

The "Network Listener" responsibility is to listen the network continuously for timestamping requests. When it receives a data stream, the "Network Listener" checks it in order to determine if it is a valid request. In the case it is, it sends an affirmative contact response to the client, it creates a "Timestamp Request" object and adds it to the "Request Timer" queue. Then it goes back to listen

to the network. In the case the request message is not correct, it sends an error message to the client.

We tried to give as few tasks as possible to the "Network Listener" to let it listen the network, which is its primary task. In order to improve the overall performance, and to avoid the fact that a slow client connection could affect the other ones, several copies of the "Network Listener" can be active at the same time.

The Request Timer

There is only an instance of "The Request Timer" in the system. The "Request Timer" is in charge of ordering the requests received from the several "Network Listeners" and timing them accordingly. All delays introduced by the system before that point (namely, those introduced by the "Network Listener") are indistinguishable from network delays, and thus not taken into account. Once a request has been timed, the "Request Timer" tries to add it to the current round queue. As the rounds are closed asynchronously by the corresponding "Round Queue Coordinator" this operation is not always successful, in that case, the "Request Timer" re-times the request and retries to queue it until it finds an open round. In that process the request sequence is preserved in order to provide a consistent behavior.

Round Queue Coordinator creation: "Round Queue Coordinator" instances are created by the "Request Timer" upon processing a request corresponding to a non-existing round. The creation of the rounds that have no requests is delayed until a request is received. Once created, those empty rounds are immediately processed, introducing no significant delay into the process.

Round number determination: Round numbers form a non-interrupted increasing integer sequence. Rounds are always in synchronization with the round duration intervals. In other words, if the round duration is one minute, all rounds will start in an absolute minute boundary, independently from when the system has been started. "Big Rounds" are determined by the "Request Timer" using a similar approach to the one followed to determine the round boundaries. We do not restrict the duration of the round to a fixed value for the lifetime of the STA. To achieve this, the information about round and "Big Round" duration is introduced into the system at the start-up phase. If we wish to modify it, we must

first shutdown the system, change the values and then restart the system, which is the only safe procedure we had foreseen.

The Round Queue Coordinator

The first thing a "Round Queue Coordinator" does is to determine the offset between the actual time and the round due time. Requests will be accepted only if the round is still valid (round is open). When requested by the "Request Timer", the "Round Queue Coordinator" adds the request to the queue and logs it. This logged request will be latter used for process auditing purposes.

When the round time is over, it obtains the "Round Values" from the preceding round and it computes the round binary trees (one for each hash algorithm) to obtain the corresponding "Round Values". Then it gives the computed trees to the "Timestamp Generator" and finally adds to the log the "Round Values" and the "Round Root Values". Those logged values will be latter used for timestamp verification and process auditing purposes. If the actual round is a "Big Round" those values are forwarded to a fixed media as well.

As you may have noticed in the section "Introduction of the timestamping techniques", the binary tree is defined for a number of leafs (requests) that is a power of 2. In general, this is not the case. We could create fake requests to finish the tree, but this will add a lot of requests (if we have $2^n + 1$ requests, then we will need to add $2^n - 1$ fake requests). A smarter solution is to add a random value only when we need it. Then, we add at most n values (one for each level of the tree). We call these nodes "Special Node", which will be logged as well. Instead of random values we could choose to use 0 or another fixed value, this would be as secure as our choice if the hash functions were "perfect". As hash functions are only "presumably perfect", we though that we could made our design more secure with really few additional computations.

In our implementation, the STA queues the requests and computes the tree at the end of the round. At first sight, it could seem a more natural solution to build the tree as soon as the requests arrive. At the end of the round, the computation of the tree would then be ended by getting the last "Round Value" and computing the actual "Round Value". In fact, this solution is harder to implement, and has no effect on the security achieved as no one can check that the STA does not perform any reordering of the requests before it publishes the "Round Value".

The Timestamp Generator

The "Timestamp Generator" processes the round trees by pairs (one for each hash algorithm) in order to generate the timestamps for each of the requests contained in the trees. In order to maximize the system responsiveness, once a timestamp has been generated it is immediately forwarded to the "Network Answer". Finally, when all the timestamps contained in a round tree have been processed the tree is destroyed.

The Network Answer

The "Network Answer" is in charge of forwarding the processed timestamps to the clients. It has been specified in such a way that it can run several threads, in that way the rest of the timestamping process can be isolated from possible network delay problematic.

The timestamp verification process

First, the verifier designates a document and its corresponding timestamp for verification. Then, the verifier's system (his personal computer or a remote computer independent from the STA) generates the two document hashes and checks if they match with those contained in the timestamp. Afterwards, the "Round Value" is reconstructed using the data provided in the timestamp. If the computed "Round Value" is consistent with the one contained in the timestamp then the next step in the verification process is to compare this "Round Value" to the "Round Value" obtained from the STA repository. Finally, the verifier provides his system with the two "Big Round Values" that he founds in the "unmodifiable media"; the verifier's system gets all the necessary "Round Values" and "Root Round Values" from the STA and it checks the coherency of the two linking chains (one for each hash function).

The audit process

The auditor designates two "Big Rounds", which he fetches from a fixed media. The system behavior will be checked between these two "Big Round Values". For each round, the auditor's system gets all the hash values (leafs of the tree and "Special Nodes") and the "Round Value" from the STA. Then, it constructs the two trees and checks that the "Round Value" is consistent. These two steps are

repeated until all the considered rounds are checked or until an error has been found. In that way, all theoretically verifiable system behavior can be verified a posteriori.

The system start-up process

Here the most sensible issue is to be able to correctly start-up the system when an unexpected shutdown has occurred. If that is the case, the log will show an unfinished round; then the system marks all entries after the last complete round as invalid and publishes that round as a "Big Round". If the log was consistent, it accesses the last valid "Round Value" in the log and publishes it as a "Big Round". This process insures a fully verifiable behavior; we are able to detect non fully-processed requests.

The system shutdown process

The administrator signals the system to shutdown. No more timestamping requests are accepted. The system waits until the current round is finished and this "Round Value" is published as "Big Round".

REFERENCES

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

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

[HS97] S. Haber and W.S. Stornetta. Secure names for bit-strings. In *Proceedings of the 4th ACM Conference on Computer and Communication Security*, pages 28–35. ACM Press, April 1997.

[MQ97] H. Massias and J.-J. Quisquater. Time and cryptography. Technical report, TIMESEC Project (Federal Governement Project, Belgium), 1997. Available at `http://www.dice.ucl.ac.be/crypto/TIMESEC.html`.

[PRQ+98] B. Preneel, B. Van Rompay, J.-J. Quiquater, H. Massias, and X. Serret Avila. Design of a timestamping system. Technical report, TIMESEC Project (Federal Governement Project, Belgium), 1998. To be available at `http://www.dice.ucl.ac.be/crypto/TIMESEC.html`.

TIMESEC

Digital Timestamping and the Evaluation of Security Primitives

Design of a timestamping system

B. Preneel, B. Van Rompay — Katholieke Universiteit Leuven
J.-J. Quisquater, H. Massias, J. Serret Avila — Université Catholique de Louvain

WP3, Technical Report

Contents

Introduction

We expose here the complete design of a digital timestamping system according to the previous work ([MQ97, PRQM97]). As it was mentioned in the description of the project an authority will provide timestamps. We have chosen technics which lower the trust needed in this authority that we will call STA[1]. The amount of work necessary to verify timestamps will be proportional to the trust we have in the STA.

The underlying assumptions for time-stamping are the followings:

1. to establish that a document was created after a given moment in time, it is necessary to report events which could not have been predicted before they happened;

2. to establish that a document was created before a given moment in time, it is necessary to cause an event based on the document, which can be witnessed by others.

First we will clarify which service the STA will exactly provide and what is the meaning of the timestamps it will issue. We define which time reference we have chosen and how we keep an "accurate" time. Then we will expose the general behavior of the different parties: the client, the STA and the verifier. This part is common to the two technics that we will present. We have chosen to implement two different technics for two reasons: firstly because one of the technique is patented and secondly to be able to compare the implementation of these two methods. The first method which is the one patented, is the tree method. This method is already in use and commercialised by its inventors S. Haber and W.-S. Stornetta. The company selling this service is Surety Technics. The second method uses an object called accumulators. It has been introduce by J. Benaloh and M. De Mare. The drawback is that it is slower because it uses modular exponentiations instead of hash functions. The advantage is that the time certificates are smaller when the amount of requests is big enough. These two methods are separate and, as well, the implementation will also be separate. Only one method at a time is necessary to provide a reliable timestamping service. We will also briefly present a recent timestamping technique which could be interesting for a future version of our STA. We will use an ISO standards for the STA's signatures.

[1] Secure Time Authority

1 The time reference

To be useful, the STA must have a time which will be recognized by everybody who will use it. All the clients and verifiers must have a common reference. For them the absolute reference will be the STA. The verifier will be able to compare two documents that have been timestamped at different rounds. If we trust the STA, it will be possible to compare two documents which have been timestamped with the tree technic (3) during the same round; but it will not be possible to achieve that with the accumulator technic. For the tree technique the relative position of two timestamps issued during the same round can be easely determined by there position in the tree, if the STA is building it respecting the original request receiving order. It will be nearly impossible to compare two timestamps issued by two different STA even if they are using the same technic. To achieve that, it is necessary to define comparison rules between the two STA's, but we do not think that it is possible to define general rules that matches strong security requirements. Nervertheless, to make easier the task of defining rules to compare timestamps issued by different STA, we will use the NTP protocol which permits the synchronization of the computers internal clocks.

Belnet provides the NTP service.

1.1 The Network Time Protocol (NTP)

NTP enables computers connected to the Internet to know with a mesurable accuracy what time it is (see [Ser]). The time service is offered on the Internet by workstations and routers covering the whole world. Thanks to the special NTP software and a link established with a radio receiver and/or an atomic clock, these machines can keep track of time within an accuracy of a few picoseconds. Radio receivers pick up the time reference via GPS ("Global Positioning System") satellites or approved transmission stations. The NTP software then reads the time value of the atomic clock or the receiver and distributes it on the Internet by means of the NTP protocol. These machines are called "stratum-1 servers".

BELNET offers affiliated institutions an NTP service with an accuracy of about 100 milliseconds. Within institutions, therefore, the local time service can use this service to indicate the exact time to all connected machines.

1.2 The type of time used

The times of astronomical and weather phenomena and events that are observed internationally are often given in "Universal Time" (abbreviated UT) or "Greenwich Mean Time" (abbreviated GMT). The two terms are often used synonymously to refer to time kept on the Greenwich meridian (longitude 0).

However, in normal civil usage, UT or GMT refers to a time scale called "Universal Time Coordinated" (abbreviated UTC), which is the basis for the worldwide system of civil time. The UTC time is kept by a large number of highly precise atomic clocks at facilities around the world, and there is international coordination in maintaining UTC to better than a nanosecond (billionth of second) per day. The length of a UTC second is defined in terms of a count of radiation cycles of a

certain atomic transition of the element cesium, and is not directly related to any astronomical phenomena.

1.3 The BELNET's NTP service

BELNET's NTP service works as follows. The BELNET routers-concentrators located in Brussels and Louvain regularly request the exact time from several stratum-1 servers. At present, this synchronization operates with three NTP servers in Western Europe.

These two routers then communicate this time reference to the other BELNET routers-concentrators in Antwerp, Mons, Ghent, Louvain, Liège and Louvain-la-Neuve.

In this way, a hierarchy of NTP servers is created in which the stratum number increases as one moves down a level. The Louvain and Brussels routers-concentrators thus correspond to "stratum" 2, while the other routers-concentrators downstream are at stratum 3.

Operating through their NTP server, the institutions connected to BELNET can obtain, if they wish, a time reference from the BELNET stratum-3 servers.

There exist three Internet Time Protocols of the same kind

1. The Time Protocol (RFC 868) provides a simple way for a system to poll for the current time from a time server.

2. The Network Time Protocol (NTP version 3, RFC 1305), is a much more sophisticated protocol which allows a system to continuously synchronize its clock against multiple time servers (to protect against transiently misbehaving servers), with adjustment for network latency and clock drift. It is targeted at synchronizing machines with each other and to a common reference, the Universal Time, Coordinated through the Internet.

3. The Simple Network Time Protocol or SNTP (RFC 1361), is a simplified RFC 1305 implementation for clients. This allows a client to take advantage of some of the features of an RFC1305 service without a full RFC1305 implementation.

BELNET enables synchronization with its 7 CISCO 7000 routers and its workstation time.belnet.be:

The first three are peering at Stratum-2 (brussels.belnet.be, leuven.belnet.be and time.belnet.be) and five others at Stratum-3 (lln.belnet.be, liege.belnet.be, antwerpen.belnet.be, mons.belnet.be, gent.belnet.be). Stratum-2 servers are peering with each other and get their time from Stratum-1 servers which in turn are connected to a radio or atomic clock.

A typical NTP configuration for the xntpd server in a BELNET institution involves three or more local NTP servers, peering with each other and getting their time from the five BELNET Stratum-3 servers. Other machines inside the institution align their clocks on these 3 servers with NTP clients.

Hostname	layer	Protocol
brussels.belnet.be	Stratum-2	RFC1305
leuven.belnet.be	Stratum-2	RFC1305
time.belnet.be	Stratum-2	RFC1305
lln.belnet.be	Stratum-3	RFC1305
antwerpen.belnet.be	Stratum-3	RFC1305
mons.belnet.be	Stratum-3	RFC1305
gent.belnet.be	Stratum-3	RFC1305
liege.belnet.be	Stratum-3	RFC1305

Table 1: NTP servers

1.4 The security

We assume that the Stratum servers have the accurate time.

Mainly five types of attacks on a time service are possible. An attacker could cause a non-time server to impersonate a time server (*masquerade*), an attacker could modify some or all time messages sent by a time server (*modification*), an attacker could resend a time server's time messages (*replay*), an attacker could intercept a time server's time messages and delete them (*denial of service*), and an attacker could delay the time messages by, for example, deliberately flooding the network, thereby introducing large transmission delays (*delay*) (see [Bis90]). In NTP version 3, countermeasures against these attacks must have been taken.

The version of NTP actually in use is 3. NTP Version 3 authentication scheme uses one-way hash functions and private keys, which must be configured in advance. The NTP Version 4 authentication scheme uses public-key crypto-system and certified public values to avoid need for key predistribution or pairwise agreement.

2 The common part of the protocols

We consider the Client-Server situation. The general form of the scheme will be the same for each solution we will propose. The differences will be in the definition of the messages between the client and the server as well as the method used by the server to timestamp the request. The methods developed must permit to everybody to check that the server is honest.

2.1 The client side

The user has a software at his side for helping him doing a few tasks. The client chooses the document he wants to timestamp. This document can be every thing he wants: a concatenation of text files, a sound file, an image, a video, etc... The software creates then a temporary data stream which consists on the hash value of the document with SHA-1 followed by the hash value of the same document with RIPEMD-160 (see [PRQM97, section 6]). By making this mandatory, we are sure that the requests are "short" and we avoid the question of examining the contains (privacy). The output (request) is send to the STA which proceeds it and sends a time certificate as a response (see 2.2). The client *must* then check the validity of the certificate he received as well as if he agrees on the time put by the STA. If he does not agree, he must contact the STA and they should find a solution. If the client does not contest the timestamp at this moment he would not be able to do it in the future. The certificate will be signed by the STA for authentification purposes. So the client will have to give the identity of the STA with the certificate to any potential verifier. The main purpose of this signature is the authentication of the STA in front of the client (see [MQ97]). If the client has enough trust on the STA, he can use the signature and avoid the verification of the certificate consistence.

2.2 The server side

First of all, the server will not examine (check) the identity of the requester. We will also not examine the problem of checking if the client has the right for using this service (i.e. if he has paid for). The server will proceed by rounds. All the rounds will have the same duration (any reasonable time slot up to 1 second) that will be fixed in advance. He will start to proceed the requests as soon as he receives them. He will issue a value for each round which is also linked to the value for the previous round. By conception, it is necessary for the STA to wait the end of the round to be able to issue any of the timestamps for the documents belonging to this round. The maximum response time of the STA and the duration time of the rounds will be defined at the implementation. We will allow the STA to have some delay in issuing the timestamps, but he absolutely must timestamp the document in the round corresponding to the time slot covering the time arrival of the request from the client.

The value for each round will be stored by the STA. It can also be stored by a third party and available on-line for the verification step. In order to improve the security and low the trust needed in the STA, we publish some round values in some unmodifiable media (for example: CDROM, newspaper, ...) to be able to

completely rely on it. The STA will put all the values for the same day together and sign the result to limit the number of signature checks for the verifier. This signatures will be published online. For the current day, the signed round values will also be published on an online server. These publications on an online server will be done using a standard protocol (ftp, http, NFS, ...) to allow verification of certificate issued at the current day.

The identity of the STA must be given with the certificate, because without it, the certificate has no value anymore. The verifier must know at which online server he will obtain the value necessary to check the integrity of the STA.

2.3 The verifier side

We suppose that the verifier is able to securely check the signature of the STA (i.e. he knows his public key). When the verifier will have to check the validity of a timestamp issued by the STA, he will check the certificate. Then if it is valid, he will contact the online service to obtain the value for the round corresponding to the time the certificate has been issued and compare it to the one included in the certificate. If he doubt that the STA is really honest, he can check the value for the round at a third party or ask for all the round values in the corresponding interval between two values published and check that the linking is valid.

2.4 Computer failures

The STA will store the "vital" values at the repository as we will explain in 3. For a full implementation, it is necessary to protect the server from failures. We propose to use clusters and backups to achieve that. The purpose is to eliminate any single point of failure. This method is well known and widely use in much sensitive applications, so we will not detailed it and not implement it in the demonstrator because it is a bit outside the goals of TIMESEC and requires a strong computer infrastructure to be efficient.

3 Tree method

This method of time-stamping has been developed by S. Haber and W. S. Stornetta ([BHS93]). A more detailed explanation and a new application is given in [HS97]. An implementation of this method is in use at Surety Technologies (see: `http://www.surety.com`). It has been analyzed in [MQ97].

3.1 Method

The principal tool used in this digital time-stamping scheme is that of a one-way hash function. We will now explain what the server will do when he processes the requests during one round.

The STA builds simultaneously two binary trees as he receives the requests. One containing the hash value of the document using SHA-1 and the other with the hash value made using RIPEMD-160. For these trees, we will use the part of the request containing the hash made using the corresponding hash function. Because it has no sense to build the two trees using the concatenate of the two hash values of the client document because if one of these hash function is broken, the corresponding value and the corresponding tree are obsolete. So we group together the material using the same hash function.

Let now see in details the building of the binary tree using one of the one-way hash functions which we wil call H. The procedure is the same for the other hash function. We call y_{i_j} the hash values that are send during the round i, j represents the request ordinal during the round i. The leafs of the tree are the hash values included in the request. The value are concatenated by two and then hashed to give the parent value. If one stage has an odd number of values, then the last value is kept and we will proceed it at the next stage (in Figure 1, $H_{i,1_4} = y_{i_7}$).

Let see an example with 7 requests (figure 1).

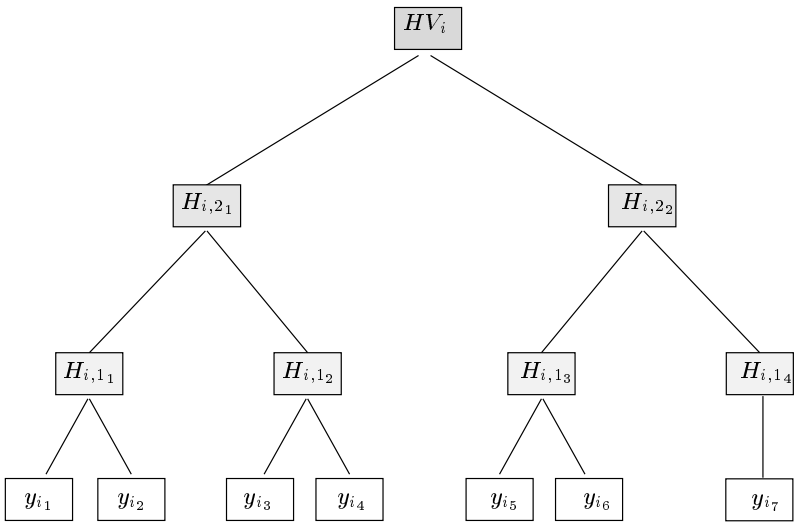


Figure 1: The binary tree structure

Then the root value (HV_i) will be concatenated with the previous round value (SHV_{i-1}) to obtain the current round value (see Figure 2).

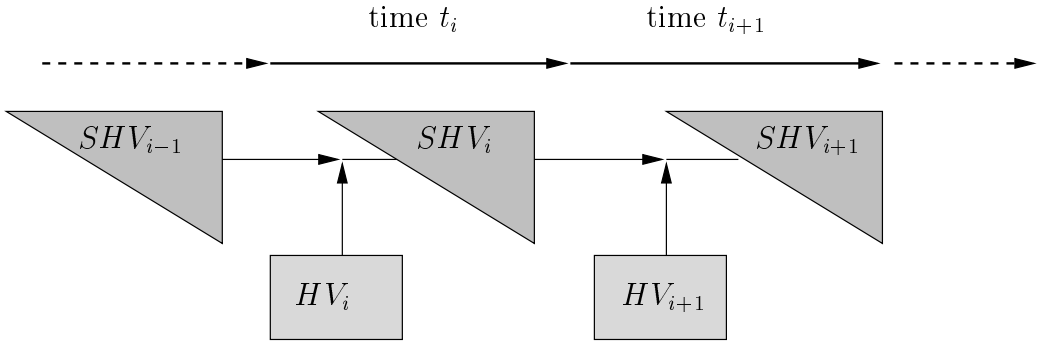


Figure 2: Calculation of round values

The values SHV_i are stored by the STA. The STA will sign them and put on the online server. We assume that only the STA can write or modify values on this server. The values are signed to prevent attacks during the communication from the client or verifier to the online server. For the demonstrator, the round values will simply be signed and available online. But for a really trusted version of the timestamping service, at least one round value must be widely published as explained before (2.2) for each day. The purpose of this is to enable the verifier to have a value which the STA can not have modified after generation. This value must be widely witnessed to make it infeasible to change.

3.2 The process

We will explain the process which is graphically illustrated in figure 3.

1. h is SHA-1 and h' is RIPEMD-160. The client sends the request $(y_{ij}^h, y_{ij}^{h'})$ to the STA, where $y_{i_j}^h = h(y_{i_j})$.

2. The STA computes the tree and the values for the round as explained in 3.1.

3. The STA signs and stores the values HV_i, SHV_i (corresponding to h) and HV_i', SHV_i' (corresponding to h') for the round binded with the number i and the time t_i of the round.

4. The STA publishes the signed values HV_i, SHV_i and HV_i', SHV_i' together with its binded information for the round on the online site.

5. The STA sends the certificate C to the client. In our example, if the request of the client is number 4 in the round, the certificate is

$$C = S(i, t_i, y_{i_4}^h, SHV_i, ((y_{i_3}^h, l), (h_{i,1_1}, l), (h_{i,2_2}, r)), SHV_{i-1},$$
$$y_{i_4}^{h'}, SHV_i', ((y_{i_3}^{h'}, l), (h_{i,1_1}', l), (h_{i,2_2}', r)), SHV_{i-1}')$$

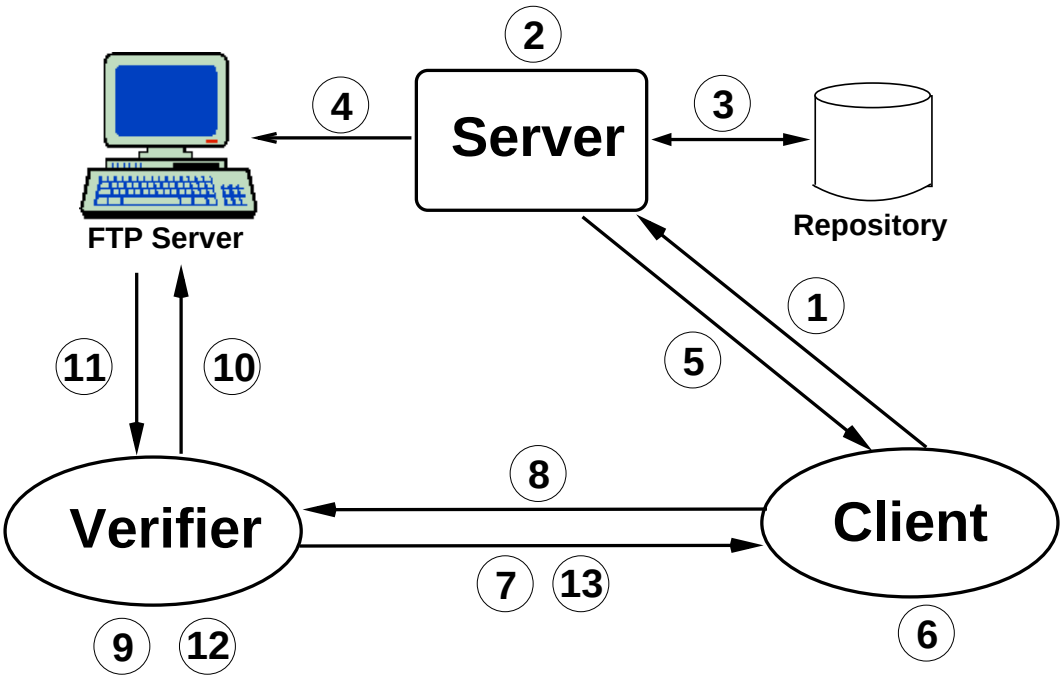


Figure 3: The process

where S is the signature provided by the STA, t_i is the time of the beginning of round i. This means that the document has been submitted between t_i and t_{i+1}. The values $y_{i_4}^h$ and $y_{i_4}^{h'}$ are not necessary in the certificate because they can be obtained from the document which is certified but it give redundancy that will be useful for the implementation.

6. The client checks his certificate. First he checks the signature, then rebuilds the two trees, compute the round values SHV_i and SHV_i' and compare it to the values contained in the certificate. Then he can ask the online server to verify the round values included in the certificate.

7. The client claims to the verifier that he has made this document at the time corresponding to the round i. The verifier asks the client for the certificate of the document.

8. The client sends the requested certificate to the the verifier as well as the identity of the STA.

9. The verifier makes the verification in the same way as the client in 6.

10. Then the verifier asks the online server for the values of the round i: t_i, SHV_i, SHV_i' and all the values necessary to rebuilt the chain between two values published which contain the round i.

11. The online server sends the round values SHV, SHV' and the values HV, HV' necessary to rebuild a chain that will at least begin and end with witnessed

values.

12. The verifier checks that the values for the round i matches with the one in the certificate, and he rebuilds the chain to verify the integrity of the STA.

13. At last, the verifier sends the result of the verification to the client .

The verifier should have the choice of checking only the certificate for one hash function. We propose that to improve speed and because it is only necessary to check the part of the certificate using one hash function when the other gets compromised.

3.3 The security

Most of the security aspects has been studied in [MQ97].

We are using in parallel two hash functions because the security relies on the robustness of the hash functions. If a hash function is broken, anybody is able to build a tree with values that he has chosen and with a root value equal to an existing root value. So, in the hypothesis that a hash function has been broken, anybody is able to generate a valid certificate for a value that has not been issued at the time of the timestamp. Nevertheless the attacker needs the collaboration of the STA to have a signature of this certificate.

Another variant of this attack, which is simpler to implement is that the attacker is able to find another document with the same hash value. So, the certificate for the first document is also valid for the second one. If we had not made mandatory the fact for the client to use SHA-1 and RIPEMD-160, a dishonest user could have use a weak hash function and then find in the future collisions as he wants. To make the job of the verifier easier (he do not have to check the robustness of the hash function) and to limit the level of his competence we impose the use of the hash functions that we know to be secure.

If one hash function is broken, all the evidence using this hash function are obsolete, so as in the Surety technique, it is more secure to use two hash functions in parallel in order to have the time to replace a broken one in case we could not have predict that before.

Another aspect of the security is the confidentiality of the secret key of the STA. If the signature key of the STA is compromised, then the verifier can't trust anymore the signature of the certificate, but he can always check the secure consistency of the certificate as we have explained earlier.

In case of the breaking of a hash function, the use of another hash function must be defined. To maintain the validity of the certificates, they must be retimestamp with the new hash functions before the two initial hash functions used are broken.

In case the signature key of the STA is compromised or has expired, a new key must be generated, but nothing else must be done from the timestamping process point of view.

3.4 The renewing process

When one hash function is compromised, all the time certificates must be renewed before the other hash function is broken too. To achieve that, it is necessary to

retimestamp not only the document submitted at the original time but also the old certificate. In this way, we will be able to prove that the document has first been timestamp at the date indicated in the old certificate and that at this time the hash functions used were not broken. We must also put a new hash of this document because otherways an attacker could use the certificate for a document which has the same hash values.

To summarize, for renewing a timestamp, the client will submit to the software a document containing the initial document and the old certificate. The request received by the STA will be a hash of this concatenation which will be processed in the habitual way but using the new hash functions.

4 Accumulators

An alternative method of timestamping was proposed by J. Benaloh and M. de Mare [BdM91]. A primitive, called one-way accumulator, processes the requests (hash values) which are sent during one round and computes a root value (as in the tree technique). The use of a suitable accumulator allows to minimize the storage requirement (size of the certificates). In their more recent article [BdM94] they defined a one-way accumulator which has the quasi-commutative property. It is a kind of hash function that can be used to hash several messages whose processing order is irrelevant. The verification procedure for the timestamping service is based on a proof of membership of a particular message during a round. The relative position of requests of the same round cannot be determined.

4.1 The problem

The input is a list of hash values (computed from the requests by the conventional hash functions SHA-1 and RIPEMD-160, as in the tree method) sent to the STA during one round: $y_i, 1 \leq i \leq m$. A one-way accumulator h is used to compute an output value z_m whose size is comparable to the size of one input value.

The protocol to treat the input values y_i is the following:

$$z_i = h(z_{i-1}, y_i), 1 \leq i \leq m \text{ with } z_0 = IV \text{(initial value)}.$$

The final result is $z_m = h(h(\cdots h(z_0, y_1), \cdots), y_m)$.

To prove that a request belongs to a certain round you must be able to verify that the corresponding hash value $y \in \{y_i, 1 \leq i \leq m\}$.

4.2 One-wayness

The one-way property of an accumulator implies that:

- given a pair (z_{i-1}, y_i), the computation of $h(z_{i-1}, y_i)$ must be 'easy'.
- given a pair (z_{i-1}, y_i) and $y' \neq y_i$, it must be 'difficult' to find $z' \neq z_{i-1}$ such that $h(z_{i-1}, y_i) = h(z', y')$.

4.3 Quasi-commutativity

A one-way accumulator h is said to be *quasi-commutative* if for all z_{i-1} and for all y_i, y_{i+1},

$$h(h(z_{i-1}, y_i), y_{i+1}) = h(h(z_{i-1}, y_{i+1}), y_i).$$

This property ensures that if one starts with an initial value IV, and a set of values $y_1, y_2, ..., y_m$, then the accumulated hash would be unchanged if the order of the y_i were permuted.

4.4 Influence of the properties on the proof of membership

The one-way property ensures that no false membership proofs can be made.

To prove that y_j belongs to the set $\{y_i, 1 \leq i \leq m\}$, we must give $c = \{z_{j-1}, y_i, j \leq i \leq m\}$ (we call that the certificate for y_j) and recompute z_m.

So we only need to save z_{j-1} and y_i for $j \leq i \leq m$ (rather than saving all y_i). However, for most y_j, this will result in only a small constant factor improvement in the storage requirements. The amount of storage is linear with respect to the number of requests m.

But for a quasi-commutative accumulator the processing order of the y_i is irrelevant and the certificate is only $c = \{z, y_j\}$ with z the accumulated hash using $y_1, ..., y_{j-1}, y_{j+1}, ..., y_m$. Hence the amount of storage is constant with respect to m.

4.5 The proposed accumulator

The function proposed by Benaloh and de Mare is the modular exponentation: $e_n(z, y) = z^y \mathrm{mod} n$. The output value associated with the input $y_i, 1 \leq i \leq m$ is:

$$z_m = e_n(e_n(\cdots e_n(e_n(e_n(z_0, y_1), y_2), y_3), \cdots), y_m) = z_0^{\prod_{i=1}^{m} y_i} \mod n.$$

The one-way property of this function is based on the difficulty of finding modular roots. The function is clearly quasi-commutative: $(z_0^{y_1})^{y_2} \mathrm{mod} n = (z_0^{y_2})^{y_1} \mathrm{mod} n$.

The certificate necessary to prove the membership of a given y_j to the set $\{y_i, 1 \leq i \leq m\}$ is

$$c = z_0^{\prod_{\substack{i=1 \\ i \neq j}}^{m} y_i} \mod n.$$

The verification phase consists of computing $c^{y_j} \mod n$ which must be equal to z_m. We can remark that we don't verify that the certificate c is a power of z_0 because we have no means for doing that. This can be inconvenient in applications such as timestamping ([BdM94], [Jus97]).

4.6 Security

If an attacker wants to find a false certificate for $y' \notin \{y_i, i = 1, \cdots m\}$, he has to

- search z' such that $z'^{y'} \equiv z_m \pmod n$,
- or search Y' such that $\left(z_0^{y'}\right)^{Y'} \equiv z_m \pmod n$.

Solving the first situation is equivalent to breaking the RSA assumption. The second situation is equivalent to the discrete logarithm problem.

Benaloh and de Mare propose to use a modulus $n = pq$ with p and q prime numbers of the form $2r + 1$ where r is also a prime number. This is called a *rigid integer*. For a closer study of the security of accumulators see [MQ98].

We must notice that if someone knows the factorisation of n, he is able to generate a certificate for any documents he wants and for the time he chooses. The STA himself should not know this factorisation or he will be able to issue back-dated timestamps.

4.7 Nyberg's accumulators

An alternative one-way accumulator, which has no trapdoor and doesn't have to be provided by a trusted third party, was proposed by K. Nyberg [Nyb96]. The drawback is that, to prevent forgery, an output length of about 2.8 Megabits is required. Nyberg proposes to hash with a conventional hash function and use the result as a seed for a pseudo-random number generator which can give an output of the desired length.

5 Binary Linking Schemes

Recently a new method for timestamping has been introduced ([BLLV]). This technique is also based on rounds and hash functions. The main guideline is also to be able to check the work of the STA to allow having limited trust on it. The main advantage of this method is that we need less values to verify a timestamp. In the techniqueswe have described, we need all the round values between two published to verify a timestamp. The work on round values is as costly for the generation as for the verification, it is proportional to the number of values between two published. With this new technique it is only proportional to the logarithm of the number of values. A drawback is that we need a same fixed number of requests during each round (we can easely override this drawback by self generating requests for example). For us it may not be a problem if we just use this technique for the root values obtained using one of the preceding methods. We can consider two levels of rounds: one like we have described (root values) and the other that we call meta-round which proceed with this root values and is limited by two consecutive published values.

We will now give an overview of this method with an example.

The number of requests during a round is 15 in our example. So value 0 and value 15 will be published: we suppose that nobody is able to change it and everybody can access it securely and accurately. H is a hash function and y_i is a request. Let $l_i = H(y_i, l_{i-1}, l_{f(i)})$ where f is a function which enable to have smaller timestamp.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$f(i)$	0	1	0	3	4	3	0	7	8	7	10	11	10	7	0

Table 2: The function f

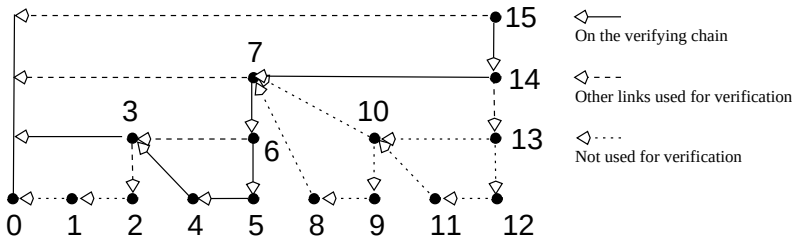


Figure 4: The timestamp of y_4

The certificate for the request y_4 (see figure 4) is :

$$\begin{aligned} \text{tail}(4) &:= (y_{15}, l_0, y_{14}, l_{13}, y_7, l_0, y_6, l_3, y_5, l_4) \\ \text{head}(4) &:= (y_4, l_3, y_3, l_2) \end{aligned}$$

It is easy to check that this certificate enables to reconstruct the smallest chain between the two round values: tail(4) from 15 to 4 and head(4) from 4 to 0.

We choose not to implement this technique in our present system because it is to new and it has not been enough studied. But it can be an open issue for the improvement of our system in the future.

6 The method of signature

According to the work done on signature scheme ([PRQM97]), we will use the international standard ISO/IEC 9796 part2. We could not use part 1 because the certificate that will be signed are too big.

7 Overview

We have presented a design of a timestamping system which matches the requirements of the project TIMESEC. The originality of this work is all what has been done on top of the timestamping algorithms (tree technique and accumulator technic). Our main goal was to be able to put the minimum possible trust on the third party providing the timestamping service. To achieve that we have chosen timestamping technics which allows anyone to check the work done by the STA. Another requirement was that the service should be as easy to use as possible without loss of security. For that, we have detailed the required behavior of the user by splitting the category of user in two parts: the client who ask for a time certificate and the verifier who will check the correctness of the certificate to be convinced by it. We also tried to reuse as much as possible of the existing Internet technology and infrastructure.

The system works by rounds, so the reliable accuracy of the timestamp is the length of a round. The most suitable use of the timestamping service is to be able to compare two documents in terms of issuing time (i.e. submission to the STA). With the tree technique it is also possible to compare two timestamps issued during the same round, but only under the condition that we trust the STA to process the requests in the same order of their arrival. Each round value depends on the preceding round values as well as on the requests that have been submitted during this round. At a fixed time laps that will be chosen for the implementation, a round value will be published on an unmodifiable media (i.e. every body can consult this value but nobody, even the STA, can change it). The security of our system ultimately rely on hash functions, so we use two hash functions in parallel to prevent the failure of one. The time certificates have a predetermined finite lifetime (the time after which the two hash functions will be broken), but it is possible to extend this lifetime by retimestamping the existing timestamp. All the requests will only contain hash values, so the documents that will be timestamped will never be known by the STA and their "secrecy" will not be exposed to possible transmissions issues.

We have defined the significance of the timestamps provided by our STA and how to generate them. We have been enough general to cover several timestamping needs: for notarial acts, medicine use, stock exchange ... We have also given indication for future possible improvements. Finally our job can also be useful for lawyers to define the status of electronic timestamp in the law, because for the moment, as far as we know electronic timestamps have no legal signification.

References

[BdM91] J. Benaloh and M. de Mare. Efficient broadcast time-stamping. Technical Report TR 91-1, Clarkson University Department of Mathematics and Computer Science, August 1991.

[BdM94] J. Benaloh and M de Mare. One-way accumulators: A decentralised alternative to digital signatures. In *Advances in Cryptology - Proceedings of Eurocrypt'93*, number 765, pages 274–285. Springer-Verlag, 1994.

[BHS93] D. Bayer, S. Haber, and W.S. Stornetta. Improving the efficiency and reliability of digital time-stamping. In R.M. Capocelli, A. De Santis, and U. Vaccaro, editors, *Sequences II: Methods in Communication, Security and Computer Science*, pages 329–334. Springer Verlag, New York, 1993.

[Bis90] M. Bishop. A security analysis of the ntp protocol. available at `http://www.belnet.be/services/ntp/ntp1.html`, 1990.

[BLLV] A. Buldas, P. Laud, H. Lipmaa, and J. Villemson. Time-stamping with Binary Linking Schemes. In *Advances in Cryptology - Proceedings of Crypto'98*. accepted at Crypto'98.

[HS97] S. Haber and W.S. Stornetta. Secure names for bit-strings. In *Conference Proceedings CS 97, Zurich, Switzerland*, pages 28–35. ACM, 1997.

[Jus97] M. Just. Some timestamping protocol failures. Technical Report TR-97-16, Carleton University, Canada, School of Computer Science, August 1997.

[MQ97] H. Massias and J.-J. Quisquater. Time and cryptography. Technical report, TIMESEC Project (Federal Governement Project, Belgium), 1997. available at `http://www.dice.ucl.ac.be/crypto/TIMESEC.html`.

[MQ98] H. Massias and J.-J. Quisquater. A survey of accumulators. Technical report, UCL Crypto group, University of Louvain-la-neuve, 1998. To appear.

[Nyb96] K. Nyberg. Fast accumulated hashing. In *Third Fast Software Encryption Workshop*, volume LNCS 1039, pages 83–88. Springer-Verlag, 1996.

[PRQM97] B. Preneel, B. Van Rompay, J.-J. Quiquater, and H. Massias. Evaluation methodology for security primitives. Technical report, TIMESEC Project (Federal Governement Project, Belgium), 1997. available at `http://www.dice.ucl.ac.be/crypto/TIMESEC.html`.

[Ser] BELNET Services. The network time protocol (ntp). available at `http://www.belnet.be/services/ntp/ntp.html`.

TIMESEC

Digital Timestamping and the Evaluation of Security Primitives
http://www.dice.ucl.ac.be/crypto/TIMESEC/TIMESEC.html

Specification and implementation of a timestamping system

J.-J. Quisquater, H. Massias, J. Serret Avila — Université Catholique de Louvain
B. Preneel, B. Van Rompay — Katholieke Universiteit Leuven

December 1999

Technical Report

Contents

List of Figures

1 Introduction

The goal of the TIMESEC project was the development of a complete system for digital timestamping, as well as a methodology for the evaluation of security primitives (in particular those which are used for timestamping). The previous technical reports of the project were:

- Time and cryptography [12] (a study of timestamping schemes);
- Evaluation methodology for security primitives [14] (an analysis of digital signature schemes and cryptographic hash functions);
- Design of a timestamping system [15] (a general design and overview of different techniques).

The first part of this final technical report is a reminder of this previous work and the conclusions that were drawn.

After analysing the different timestamping schemes (see [12] and [15]), it was decided to develop two seperate systems, based on different techniques: the tree technique and the accumulator technique. In this technical report we present the work done for the implementation of both these timestamping systems.

2 Reminder of the previous work

This is an abstract of the results and the conclusions that have been obtained earlier in the project.

Digital timestamping is a cryptographic technique used to provide temporal information about electronic documents. It allows other parties to validate the time at which a document has been created, and to check that the document hasn't been altered since.

There are two families of timestamping techniques:

- techniques that work with a trusted third party. They rely on the unpartiality of the entity that is in charge of issuing the timestamps. We can also classify these techniques into two different types: those where the third party is completely trusted and those where it is partially trusted.

- techniques that are based on the concept of distributed trust. They consist of making documents dated and signed by a large set of people in order to convince the verifiers that we could not have corrupted all of them. The main drawback is that it requires a lot of cooperation.

A detailed study of timestamping techniques can be found in the first technical report [12]. Nowadays, all the commercial implementations of timestamping services are based on the first approach. It is for this reason that we concentrate on this **trusted third party** approach.

An analysis of security primitives important for timestamping -digital signature schemes and hash functions- was presented in the second technical report [14]. Several candidates were proposed, which are expected to offer the required security for some years to come.

The design for the timestamping system was described in the third technical report [15]. The main goal was to *minimize the trust required in the third party providing the timestamping service*.

The "easy" solution, which consists of concatenating the document with the current time and signing the result has been discarded because it has two main drawbacks:

1. The first one is that we must completely trust the Secure Timestamp Authority (STA), which can issue undetectable back-dated timestamps.

2. The second one is related to the limited lifetime of cryptographic signatures, which can be shorter than the document time-to-life.

The timestamping method that we have chosen works by rounds, as described in [2] and [3]. All the timestamping requests (which contain hash values of particular documents) received by the STA during a round are accumulated and combined to produce a single round value. Different methods of accumulation are possible and we have chosen to implement two of them which seem the most practical (see sections 3 and 4).

The timestamp for an individual request processed during a particular round, consists of information which allows one to check that this request was part of the accumulation process which produced the corresponding round value. The first phase of the verification process consists of performing this check.

The round values which are calculated in succeeding rounds, are linked to each other by means of a hash function, thus creating an unforgeable temporal chain. The verification of a timestamp concludes by rebuilding this chain until a value trusted by the verifier is obtained.

Periodically, one of the round values is published on an unalterable and widely witnessed media (e.g., a newspaper...). These special round values, which we call "big round values", are the base of trust for all the issued timestamps. All verifiers must trust these big round values as well as

the time associated with them. This is a reasonable requirement because those values are widely witnessed.

The absolute time trusted by all the potential verifiers is the time indicated by the unmodifiable media. For the following discussions we suppose that this time is the same as the time indicated by the STA for the big round. Another requirement is to force the clients to check the timestamps as soon as they get them. In that way the process is continuously audited and the STA will not have any margin to manoeuver in an untrusted way.

A very useful method for extending the lifetime of timestamps is described in [2]. It basically consists of re-timestamping the document as well as the original timestamp before the hash function is broken.

Next we give a detailed description of the two different techniques for the round accumulation process.

3 Accumulation using a binary tree.

The first method of accumulation uses a binary tree structure and has been described in [7] and [8]. For each round a binary tree is constructed with the requests received during the round. In Figure 1 we can see a graphical representation of a round constructed using this method.

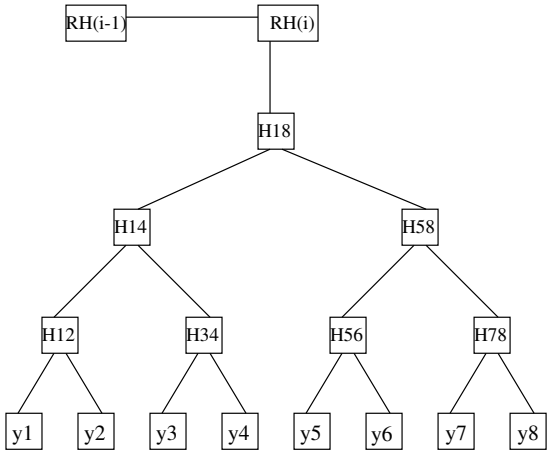


Figure 1: The binary tree structure

Each of the requests consists of a hash value of a given document. The leafs of the tree are each of those hash values. The leaf values are then concatenated by two and hashed again to obtain the parent value (e.g., $H_{34} = H(y_3 \mid y_4)$). This process is repeated for each level until a single value is obtained.
The top value of the round tree (H_{18}), which is called the "round tree value", is then concatenated with the value obtained for the preceeding round (RH_{i-1}), and finally hashed again to obtain the actual round value (RH_i).

The timestamp of a document requested during the current round contains all the values necessary to rebuild the corresponding branch of the tree. For example, the timestamp for y_4 contains $\{(y_3, L), (H_{12}, L), (H_{58}, R), RH_{i-1}\}$. The size of these timestamps increases with the number of requests processed in a round, on a logarithmic basis.

The verification process consists of rebuilding the tree's branch and the linking chain of round values until a trusted (for the verifier) round value is recomputed. This verification method is explained in detail in [7], [12] and [10].

The security relies on the hash function used: if this function is collision-free (this means that it is unfeasible to find two inputs which are processed to the same hash value), it is impossible to forge the binary round tree or the linking chain.

As an additional feature we can build two trees in parallel for each round, using two different hash functions (SHA-1 and RIPEMD-160, chosen according to the work in [14]). In that way, the system remains secure in case of an unexpected break of one of these hash functions.

4 Accumulation using a one-way accumulator

An alternative method of accumulation was proposed in [4]. During a round we collect the requests y_i ($1 \leq i \leq m$), which are hash values of individual documents. The accumulation is performed by means of the modular exponentiation operation. Starting from an initial value Z_0, the "round accumulation value" Z is computed as follows:

$$Z = Z_0^{\prod_{i=1}^{m} y_i} \mod N.$$

This value is then concatenated with the value obtained for the preceeding round (RH_{i-1}), and finally hashed to obtain the actual round value (RH_i).

To verify the timestamp for a particular request y_j we check the following equation:

$$Z = Z_j^{y_j} \mod N, \text{ with } Z_j = Z_0^{\prod_{\substack{i=1 \\ i \neq j}}^{m} y_i} \mod N.$$

Hence an individual timestamp for request y_j consists of the partial value Z_j and the round accumulation value Z (this last value is the same for all requests processed during the same round). The size of the timestamps is independent of the number of requests processed in the round. The drawback of this method is that the modular exponentiation operation is less efficient than a hash function.

The verification process consists of checking the equation above and rebuilding the linking chain of round values until a round value trusted by the verifier is recomputed.

The security of this method relies (besides on the hash function used) on the well known RSA problem: to produce a fake timestamp for a value y', one should find a value Z' with

$$Z'^{y'} = Z \mod N.$$

If the factorization of the modulus ($N = p \times q$) is unknown this is unfeasible. As an additional feature we should construct the primes of the modulus as follows: $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are also primes. The reason for this is that otherwise the repeated application of the modular exponentiation operation might reduce the size of the image so much that finding collisions would become feasible. The factorization of the modulus has to be kept secret, it must be provided by a trusted third party, which can however be off-line. An analysis of one-way accumulators is presented in [9].

5 A timestamping system based on the tree technique

We present here the specification and implementation of the timestamping system we made for the project TIMESEC. The methodology used for the specification is object oriented and is inspired from the ones presented in [13] and [6]. We used the Unified Modeling Language (UML) to represent this specification (see [5], [6], [13] and [1]). The comments on the implementation are made when presenting the specification.

First we present the different Use Cases of this system as well as the actors. The different processes are then described. The attributes and operations of the different classes are given. Interesting algorithms for the round tree generation and timestamp generation are presented. The last part is a resume of the most important issues that we solved for the implementation of this timestamping system.

A brief description of our analysis can be found in [10]. Some issues when building a timestamping system as well as when using timestamps are given in [11].

6 Overview of the global system

A first description of the global system has been done in [15]. We give here more details following those guidelines.

6.1 The actors

The main actors of our system are the client, the verifier and the administrator. The complementary actors are the publisher, the accurate time and the auditor. An illustration of the interaction between this different actors is given in Figure 2.

The role of each of those actors is the following:

Client: the client (User) sends a timestamp request to the STA, waits for the answer and verify the validity of it as soon as he receives it;

Verifier: the verifier is given a timestamp issued by the STA as well as the document originally timestamped and wants to verify if this timestamp is valid;

Administrator: the administrator is in charge of maintaining a working timestamping system;

Auditor: the auditor is in charge of verifying the correct behavior of the STA;

Publisher: the publisher is in charge of publishing values associated with a time in one or several unmodifiable media when asked by the STA;

Accurate time: the accurate time provides an accurate time to the STA.

6.2 The use cases

The use cases are the following:

1. document timestamp,
2. timestamp check,
3. system audit,
4. system start-up, and
5. system shutdown.

The description of each of those use case follows.

6.2.1 Use Case 1: “Document timestamp”

The activity diagram of this use case is represented on Figure 3.

Goal: Timestamp a document in a non-repudiable fully traceable way.

Preconditions: The system is working and “on-time”.

Primary actor: The user.

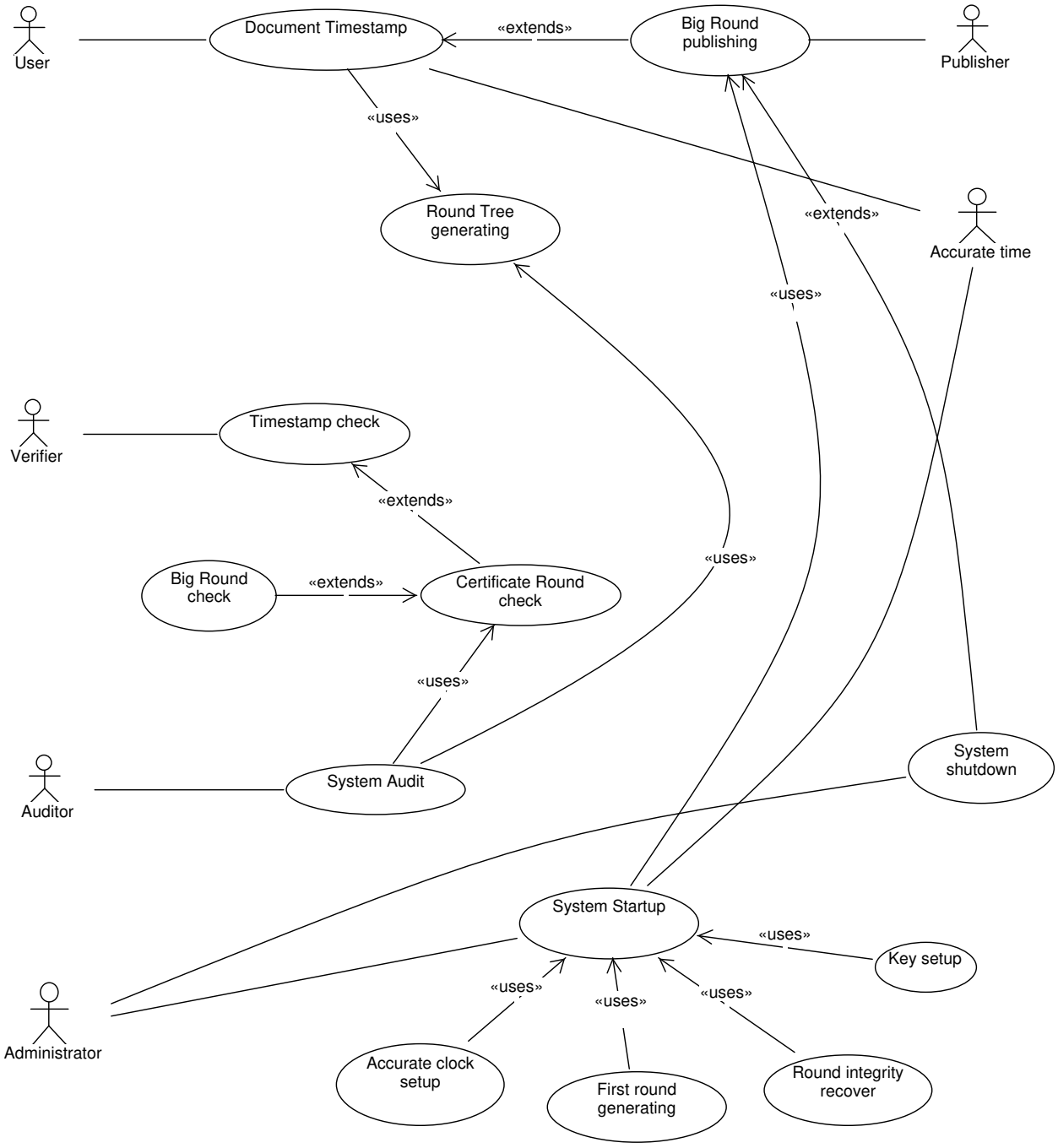


Figure 2: Use case diagram

Main scenario

1. User designates a document
2. Two hashes of the document are created.
3. For every one of the hashes: put document hash in round queue.
4. Put the two hashes and arrival time and request ordinal in log.
5. Once round is due (round due is determined by accurate time), generate round value.
6. Put round value in log.
7. Generate document timestamp certificate.
8. Give user, document timestamp certificate.

Extensions

6-A. If round="Big round" (BIG ROUND PUBLISHING)

6-A-i. Publish round value at "Unmodifiable media".

Success end condition: User have document timestamp, all the timestamping process is traceable.

6.2.2 Use Case 2: "Timestamp check"

The activity diagram of this use case is represented on Figure 4.

Goal: Verify the timestamp of a document.

Preconditions: The system is working and it exists a valid timestamp over a document. It exists an "unmodifiable media" which publishes big rounds.

Primary actor: The verifier.

Main scenario

1. The verifier designates a document and a "timestamp certificate" (which we will call simply certificate from now).
2. The system generates the document hash (*2) and checks if it matches with the one in the certificate.
3. The certificate (*2) structure is checked (including its construction and signature).
4. The result is returned to the verifier.

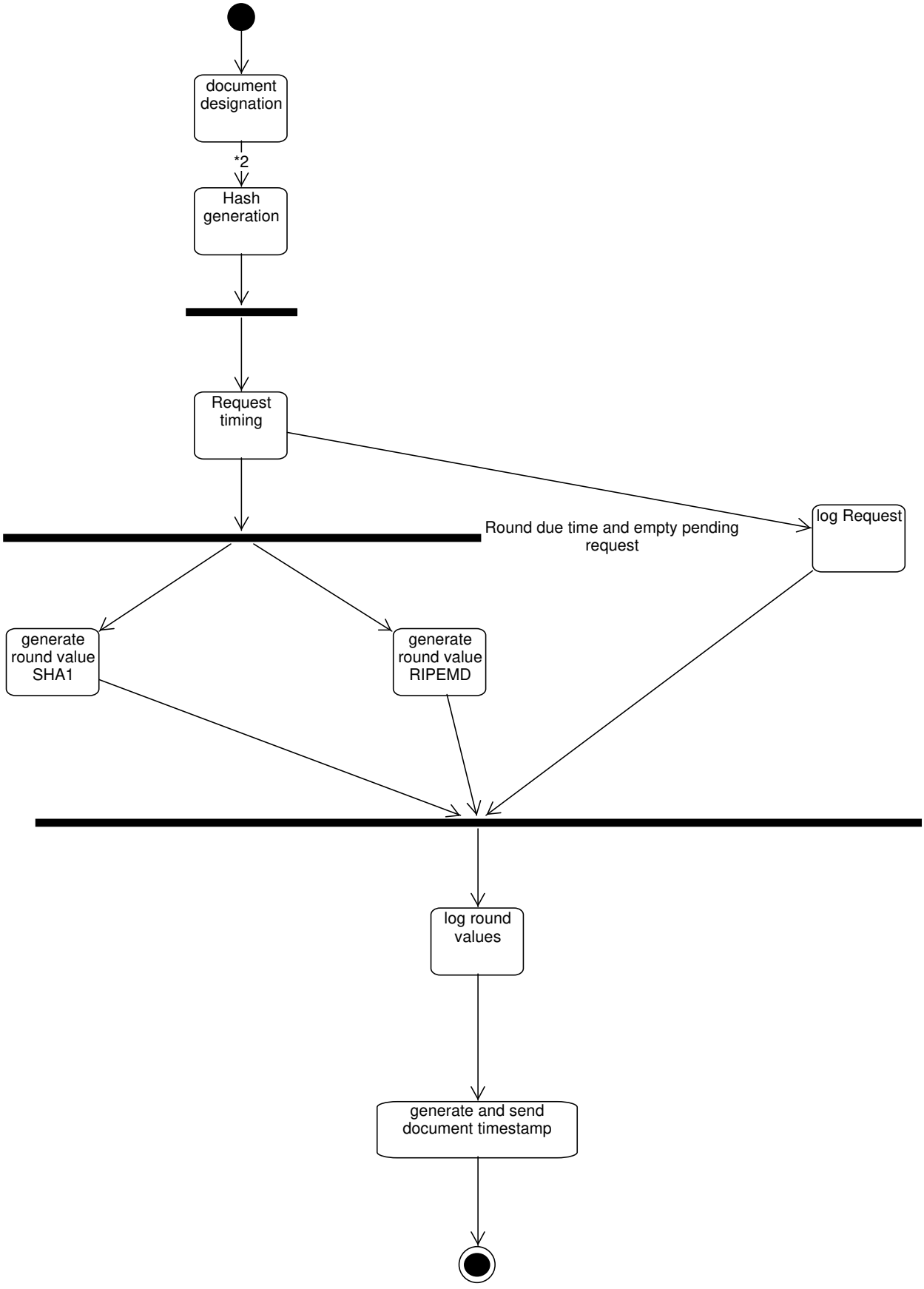


Figure 3: Document timestamp activity diagram

Extensions

3-A. The verifier wants to check the certificate against its round value. (CERTIFICATE ROUND CHECK)

3-A-i. The system access the log to obtain the round values of the certificate.

3-A-ii. The system checks if the round values match.

3-B. The verifier wants to check the round construction (BIG ROUND CHECK).

3-B-i. The verifier provides the system with the two big round values that can be found in the "unmodifiable media" (provided by the publisher actor).

3-B-ii. The system accesses the log and obtains all the round values between the two big round values.

3-B-iii. The system checks the coherency of all the rounds values obtained between the big round values.

Success end condition: All checks are verified to be true.

Failure condition: A check is false.

6.2.3 Use Case 3: "System audit"

The activity diagram of this use case is represented on Figure 5.

Goal: Verify the complete system behavior for a given amounts of rounds.

Preconditions: The system is working and it exists an "unmodifiable media" with published big round values.

Primary actor: The auditor.

Main scenario

1. The user provides the system with a set of consecutive big round values. System behavior will be checked between the first and the last big rounds.
2. The system obtains from the log all the hash values and the round values for a given round.
3. The system constructs the tree and checks that the round values are consistent.
4. Steps 2-3 are repeated until all round between the first and the last big rounds have been checked.
5. The result is returned to the user.

Success end condition: All checks are verified to be true.

Failure condition: A check is false.

The use of the last step illustrated in Figure 5 is to guarantee that all the round values between consecutive Big rounds have been checked and that the algorithm do not stop before the next big round. This step is not useful in our implementation because it is achieved by the main audit process.

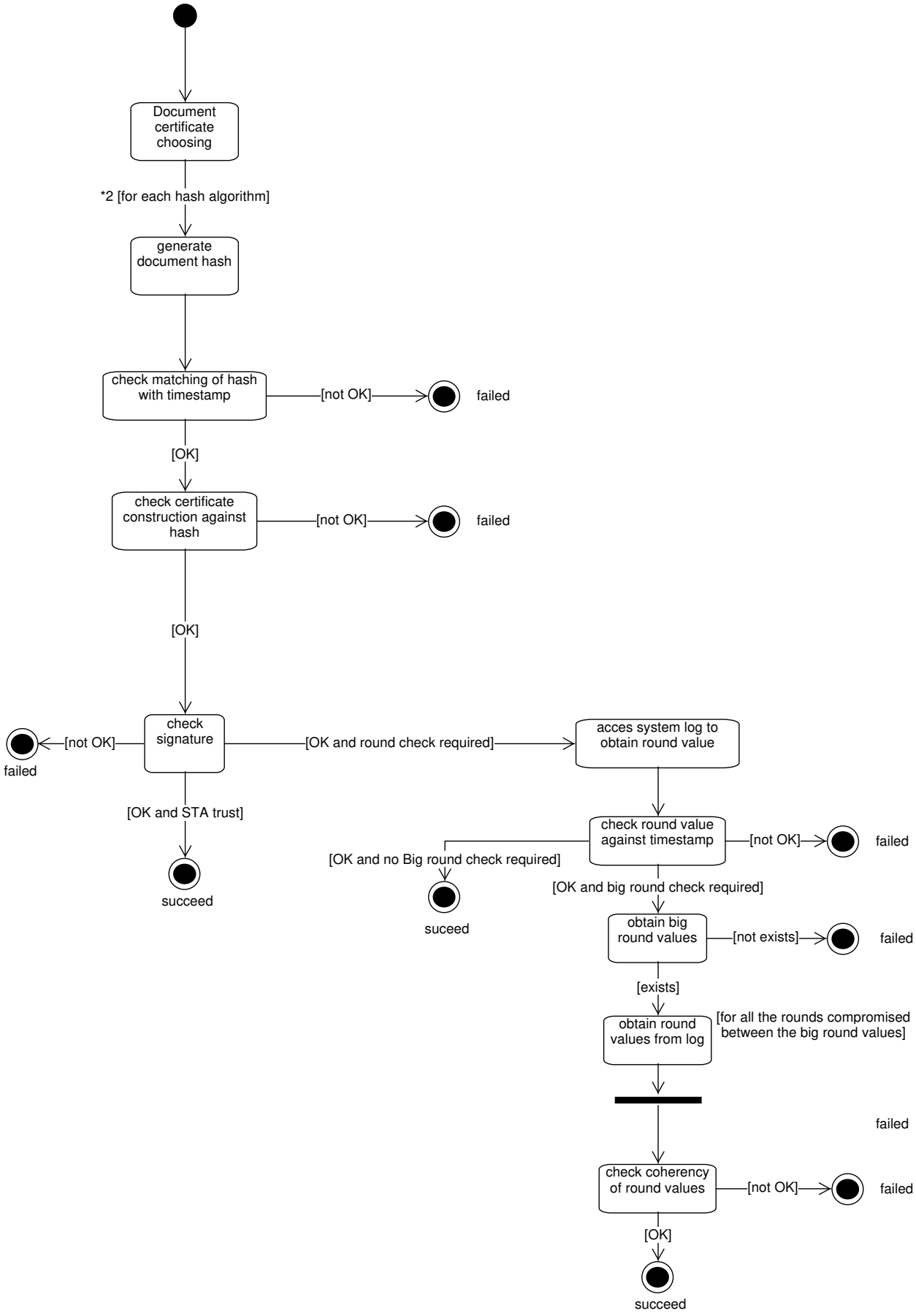


Figure 4: Timestamp check activity diagram

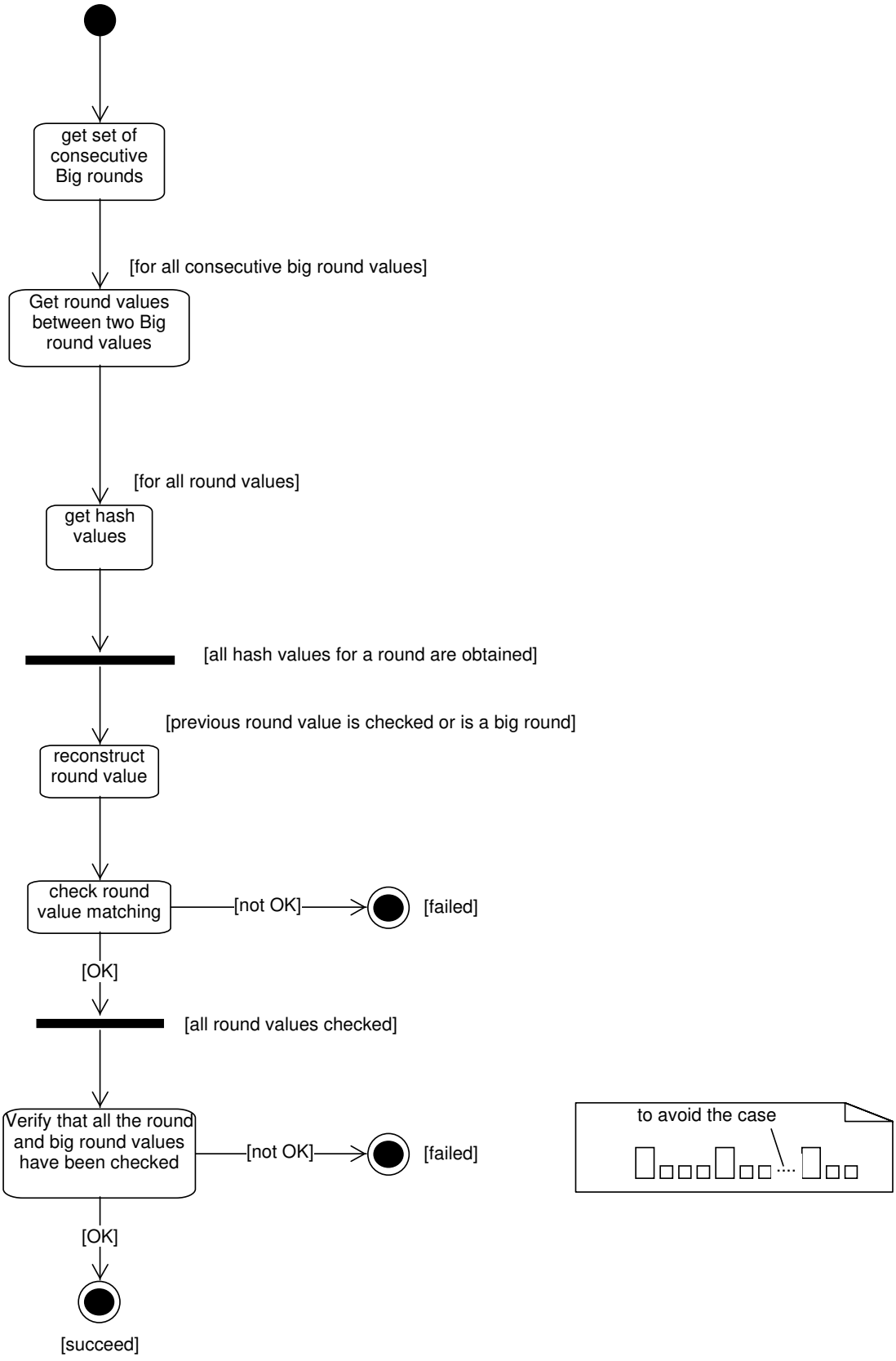


Figure 5: System audit activity diagram

6.2.4 Use Case 4: "System start-up"

The activity diagram of this use case is represented on Figure 6.

Goal: Start-up the system in a consistent and traceable way.

Preconditions: The system is stopped. A protected storage for the private key exists. An accurate time server exists and is accessible.

Primary actor: The administrator.

Main scenario

1. The system access the private-public keys.
2. The system synchronizes its clock with the accurate time server.
3. The system access the log and obtains the last round value.
4. The system publishes this value as BIG ROUND.
5. The other use cases one enabled.

Extensions

1-A. At first startup (KEY SETUP)

1-A-i. The administrator provides the system with public/private keys.

2-A. At first startup (ACCURATE CLOCK SETUP)

2-A-i. The system asks the administrator for the source of the accurate clock.

3-A. At first startup (FIRST ROUND GENERATING)

3-A-i. The system generates a hash = zero hash value = 0.

3-B. Incomplete round (ROUND INTEGRITY RECOVER)

3-B-i. The system accesses in the log the last valid round value.

3-B-ii. All entries after this value are marked invalid.

Success end condition: System is working in a consistent way.

6.2.5 Use Case 5: "System shut-down"

The activity diagram of this use case is represented on Figure 7.

Goal: Shut-down the system in a consistent and traceable way.

Preconditions: The system is working in a consistent way.

Primary actor: The administrator.

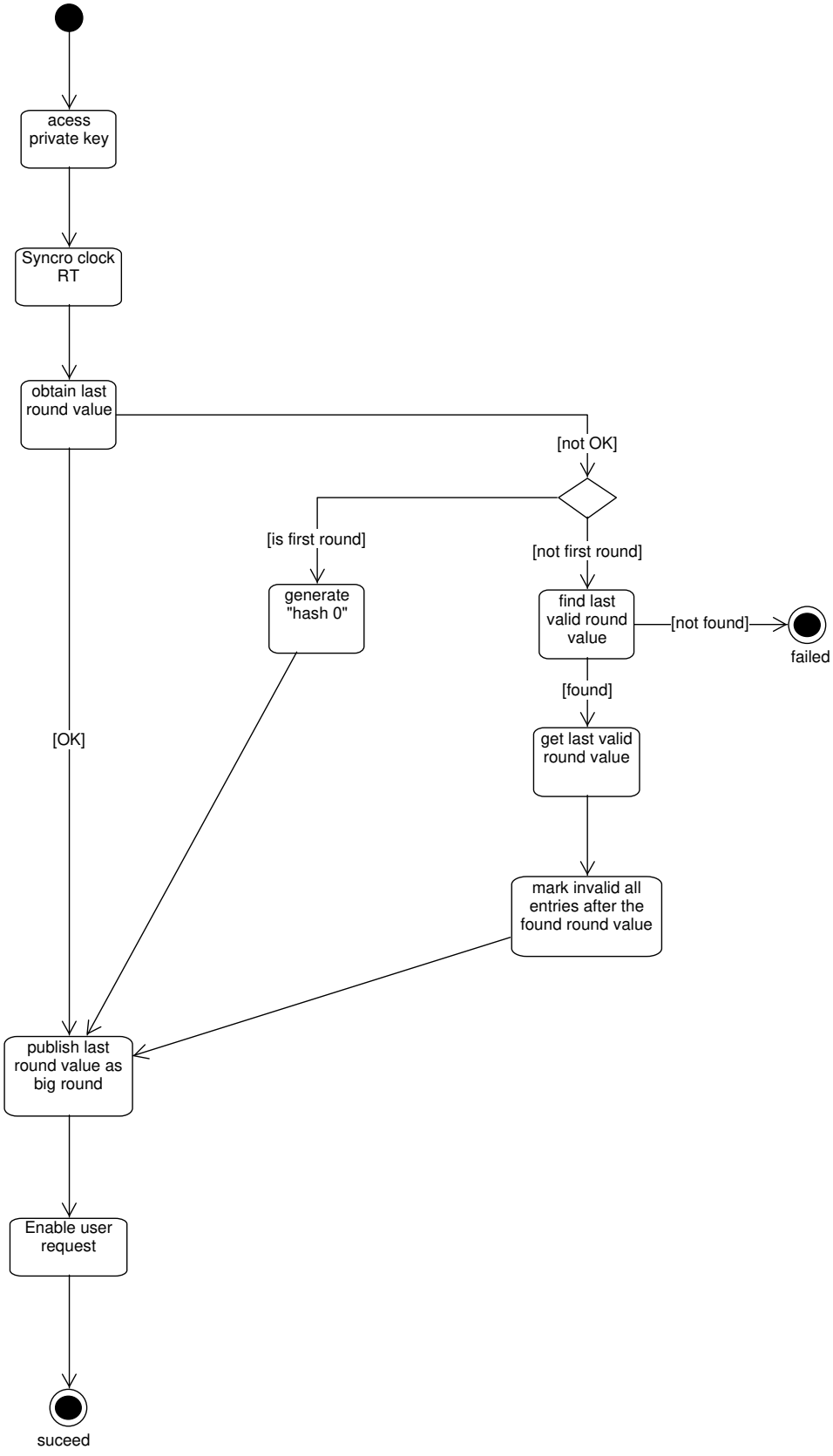


Figure 6: System start-up activity diagram

Main scenario

1. The administrator signals the system to shutdown.
2. No more user requests are accepted.
3. The system waits until the current round is finished.
4. The current round (Final round now) value is published as BIG ROUND.
5. The log is marked as consistent.

Success end condition: System is correctly shutdown.

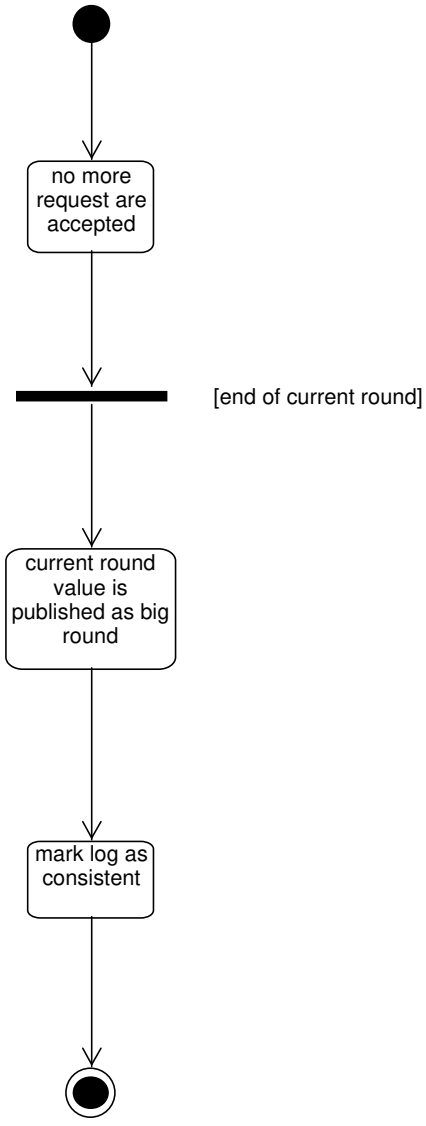


Figure 7: System start-up activity diagram

7 Description of the different processes

The critical process in this system is the timestamping process. The verification process as well as the audit process mainly reuse primitives of the timestamping process. When designing the components we took care of being able to reuse it for the other processes. The system design follows a highly decoupled multi-threaded approach. Each step is assigned to a specific component, which has its own different thread. The multi-thread approach is justified by the requirement to obtain a highly responsive and load independent implementation.

7.1 The timestamping process

When the client asks for a timestamp, a connection is established with the STA and remains open until the timestamp is send back to the client. In the case where this connection is broken, the timestamp is lost. This choice has been made for simplicity, but other types of connections (http, email, ...) can easily be implemented. Our design is independent from the type of connections, it is just an implementation issue.

By isolating the process charges into independent steps we try to decouple the load between them. Each step has also a working queue. Those queues are in charge of softening the speed differences between the different process steps.

The different components are the following:

- the Network Listener,
- the Request Timer,
- the Round Queue Coordinator,
- the Logger,
- the Timestamp Generator and
- the Network Answer.

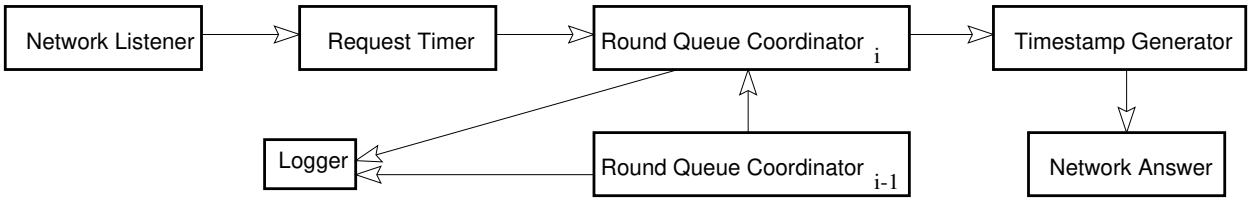


Figure 8: Interactions between the components

A schematic outline of the process is given in Figure 8.

Network Listener: The “Network Listener” is in charge of continuously listen to the clients’ timestamp requests.

Request Timer: The “Request Timer” receives the constructed requests from the “Network Listener”. Then, it times and forwards them to the actual “Round Queue Coordinator”.

Round Queue Coordinator: Each round has its own "Round Queue Coordinator", which is in charge of compiling and processing into a tree all the requests belonging to the round.

Timestamp Generator: When the round tree has been computed it is forwarded to the "Timestamp Generator", which generates the corresponding timestamps.

Network Answer: Once a timestamp is generated, the "Timestamp Generator" forwards it to the "Network Answer", which in turn forwards it to the client.

A more in depth description of the timestamping process is represented on a sequence diagram (see Annexe A).

7.1.1 The "Network Listener"

The state diagram of this component is given in Figures 9 and 10.

The "Network Listener" responsibility is to listen the network continuously for timestamping requests. When it receives a data stream, the "Network Listener" checks it in order to determine if it is a valid request. In the case it is, it sends an affirmative contact response to the client, it creates a "Timestamp Request" object and adds it to the "Request Timer" queue. Then it goes back to listen to the network. In the case the request message is not correct, it sends an error message to the client.

We tried to give as few tasks as possible to the "Network Listener" to let it listen to the network, which is its primary task. In order to improve the overall performance, and to avoid the fact that a slow client connection could affect the other ones, several copies of the "Network Listener" can be active at the same time.

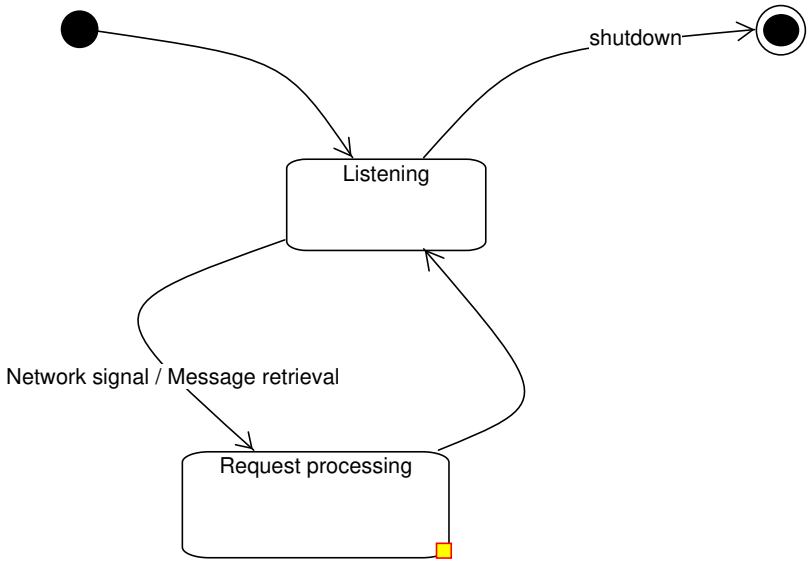


Figure 9: Network Listener state diagram

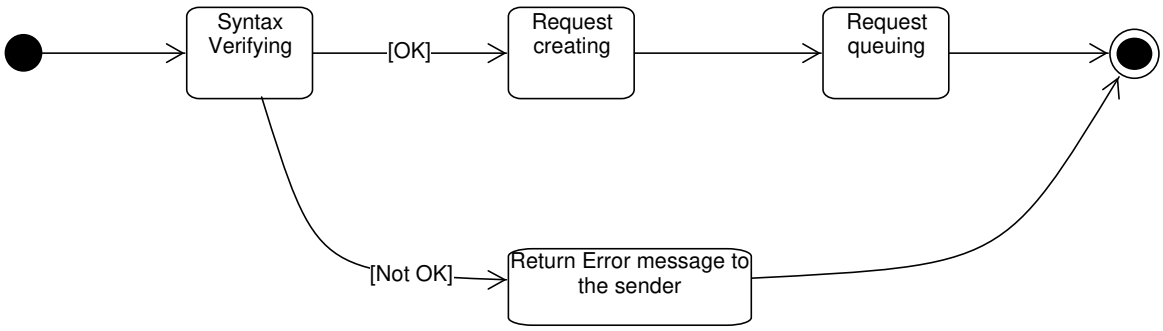


Figure 10: Request processing state diagram

7.1.2 The "Request Timer"

The state diagram of this component is given in Figures 11, 12 and 13.

There is only an instance of "The Request Timer" in the system. The "Request Timer" is in charge of ordering the requests received from the several "Network Listeners" and timing them accordingly. All delays introduced by the system before that point (namely, those introduced by the "Network Listener") are indistinguishable from network delays, and thus not taken into account. Once a request has been timed, the "Request Timer" tries to add it to the current round queue. As the rounds are closed asynchronously by the corresponding "Round Queue Coordinator" this operation is not always successful, in that case, the "Request Timer" re-times the request and retries to queue it until it finds an open round. In that process the request sequence is preserved in order to provide a consistent behavior.

Round Queue Coordinator creation: "Round Queue Coordinator" instances are created by the "Request Timer" upon processing a request corresponding to a non-existing round. The creation of the rounds that have no requests is delayed until a request is received. Once created, those empty rounds are immediately processed, introducing no significant delay into the process. The round tree value used for those empty rounds is a random value.

Round number determination: Round numbers form a non-interrupted increasing integer sequence. Rounds are always in synchronization with the round duration intervals. In other words, if the round duration is one minute, all rounds will start in an absolute minute boundary, independently from when the system has been started. "Big Rounds" are determined by the "Request Timer" using a similar approach to the one followed to determine the round boundaries. We do not restrict the duration of the round to a fixed value for the lifetime of the STA. To achieve this, the information about round and "Big Round" duration is introduced into the system at the start-up phase. If we wish to modify it, we must first shutdown the system, change the values and then restart the system, which is the only safe procedure we have foreseen.

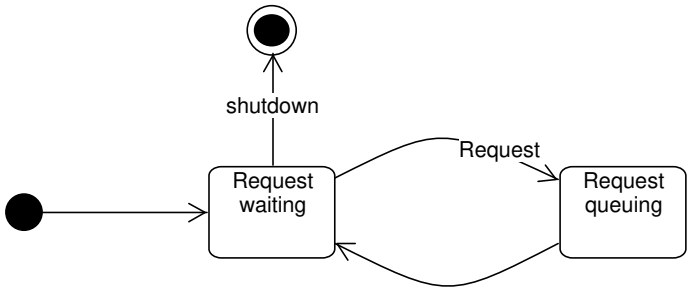
The time associated with a round is the time at which the round is closed.

When starting the system and creating the Request Timer, it test if the Initial Time for the system is greater than the actual time. In this case, it waits for this Initial Time. The actual round number is the round number corresponding to the last round of the previous execution of the system plus one. To have a monotonic increasing value for the round number (even if the system has been stopped for a long time, the round number of the round just following round number n is number $n+1$), we have introduce the variable "Offset Round". The "Round Due Time" is calculated using the following formula:

$$Round\,Due\,Time = Initial\,Time + (Round\,Number + Offset\,Round + 1) * Round\,Duration.$$

The "Offset Round" is calculated at the creation of the Request Timer with the following formula:

$$Offset\,Round = (time() - Initial\,Time)/Round\,Duration - Round\,Number.$$



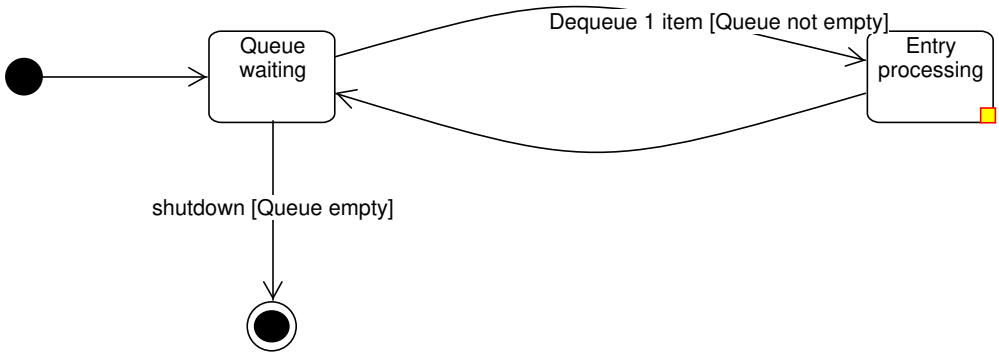


Figure 11: Request Timer state diagram

7.1.3 The "Round Queue Coordinator"

The state diagram of this component is given in Figure 14.

The first thing a "Round Queue Coordinator" does is to determine the offset between the actual time and the round due time. Requests will be accepted only if the round is still valid (round is open). When requested by the "Request Timer", the "Round Queue Coordinator" adds the request to the queue and logs it. This logged request will be latter used for process auditing purposes.

When the round time is over, it obtains the "Round Values" from the preceding round and it computes the round binary trees (one for each hash algorithm) to obtain the corresponding "Round Values". Then it gives the computed trees to the "Timestamp Generator" and finally adds to the log the "round tree values" and the "round values". Those logged values will be latter used for timestamp verification and process auditing purposes. If the actual round is a "Big Round" those values are forwarded to a fixed media as well.

Remark: A slight modification of these specifications has been done in the implementation. The tree is computed before getting the last round value. After obtaining the last round value, the

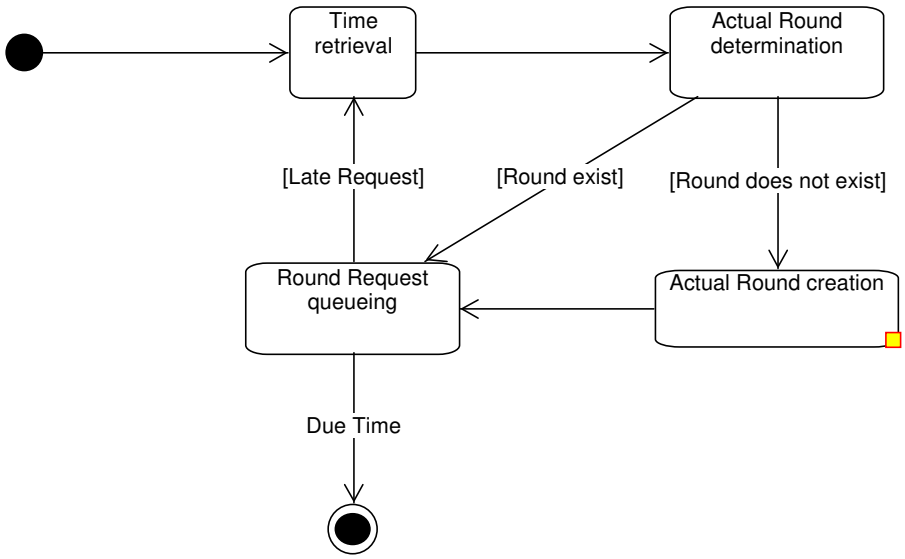


Figure 12: Entry processing state diagram

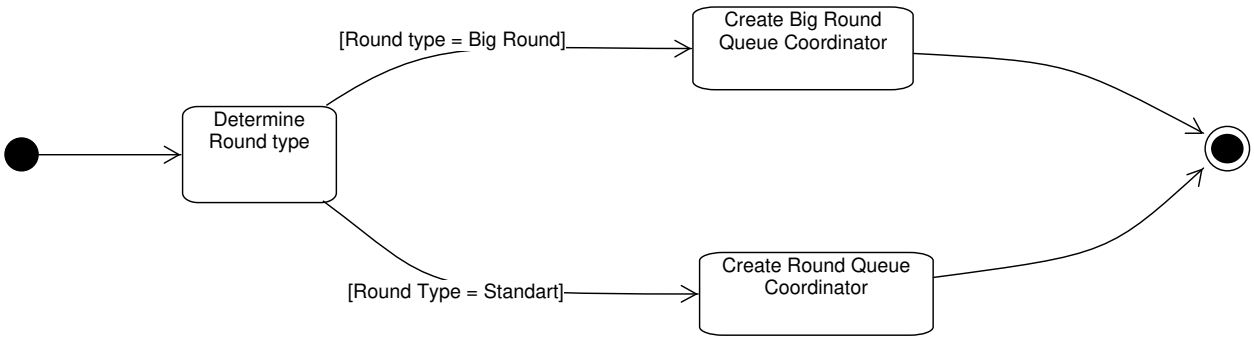


Figure 13: Actual round creation state diagram

actual round value is immediately computed using the round tree value. This change improve a bit the system in the un-probable event where the computation of the round $i-1$ is not finished when starting the tree calculation for round i.

As you may have noticed in the section 2, the binary tree is defined for a number of leafs (requests) that is a power of 2. In general, this is not the case. We could create fake requests to finish the tree, but this will add a lot of requests (if we have $2^n + 1$ requests, then we will need to add $2^n - 1$ fake requests). A smarter solution is to add a random value only when we need it. Then, we add at most n values (one for each level of the tree). We call these nodes "Special Node", which will be logged as well. Instead of random values we could choose to use 0 or another fixed value, this would be as secure as our choice if the hash functions were "perfect". As hash functions are only "presumably perfect", we though that we could made our design more secure with really few additional computations.

In our implementation, the STA queues the requests and computes the tree at the end of the round. At first sight, it could seem a more natural solution to build the tree as soon as the requests arrive. At the end of the round, the computation of the tree would then be ended by getting the last "Round Value" and computing the actual "Round Value". In fact, this solution is harder to implement, and has no effect on the security achieved as no one can check that the STA does not perform any reordering of the requests before it publishes the "Round Value".

7.1.4 The Timestamp Generator

The state diagram of this component is given in Figures 15, 16 and 17.

The "Timestamp Generator" processes the round trees by pairs (one for each hash algorithm) in order to generate the timestamps for each of the requests contained in the trees. In order to maximize the system responsiveness, once a timestamp has been generated, it is immediately forwarded to the "Network Answer". Finally, when all the timestamps contained in a round tree have been processed the tree is destroyed.

7.1.5 The Network Answer

The state diagram of this component is given in Figure 18.

The "Network Answer" is in charge of forwarding the processed timestamps to the clients. It has been specified in such a way that it can run several threads, in that way the rest of the timestamping process can be isolated from possible network delay problematic.

7.1.6 The Logger

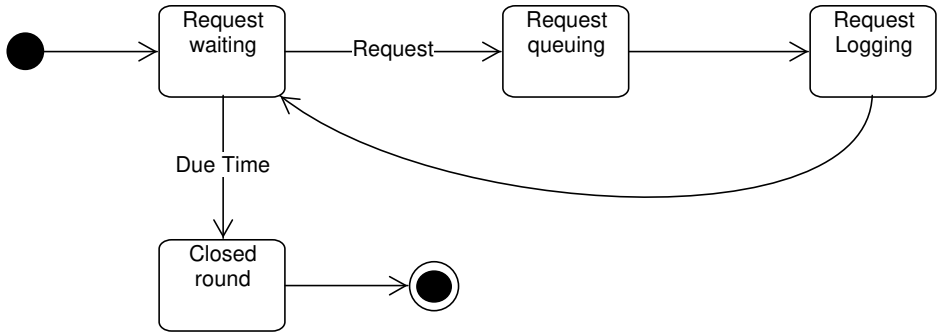
The state diagram of this component is given in Figures 19 and 20.

The logger logs the values when asked. It keeps also an index file that contains the position of the round values in the log file.

7.2 The timestamp verification process

The activity diagram of this process is given in Figure 4.

First, the verifier designates a document and its corresponding timestamp for verification. Then, the verifier's system (his personal computer or a remote computer independent from the STA) generates the two document hashes and checks if they match with those contained in the



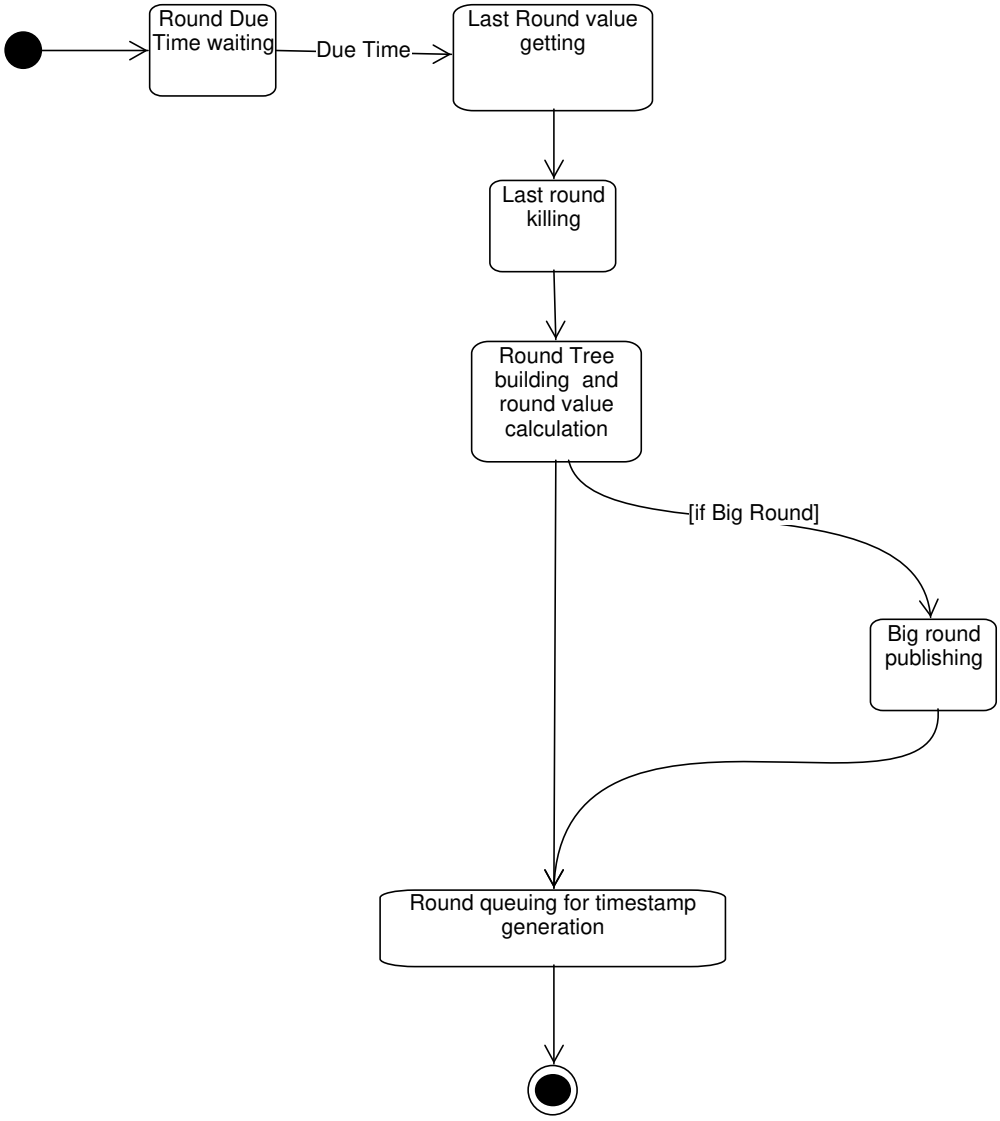


Figure 14: Round Queue Coordinator state diagram

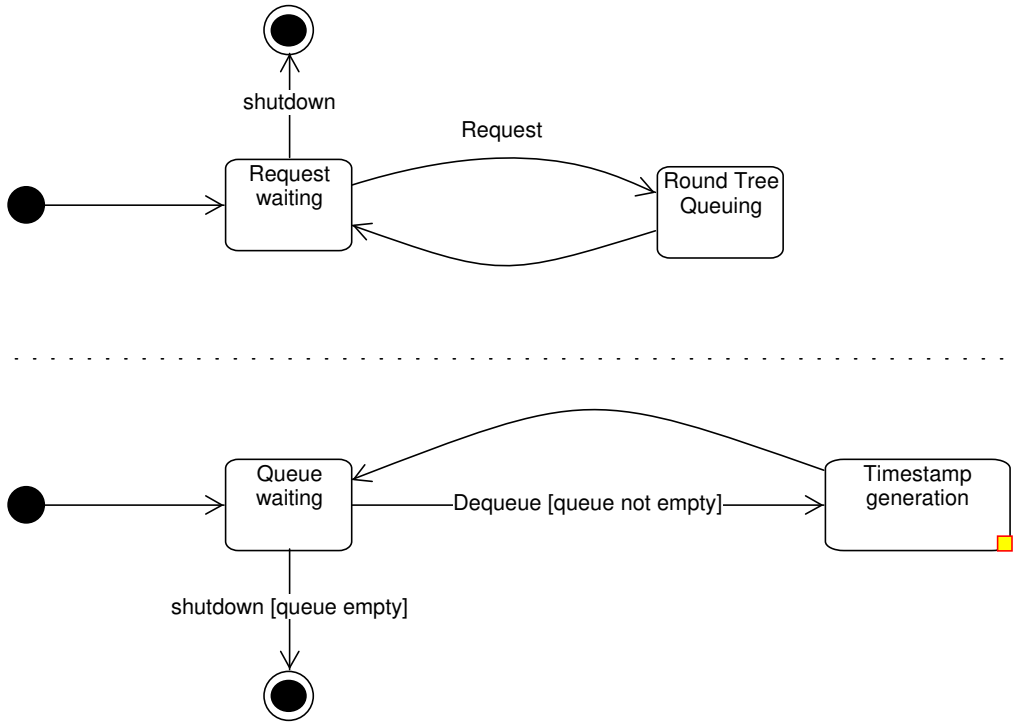


Figure 15: Timestamp Generator state diagram

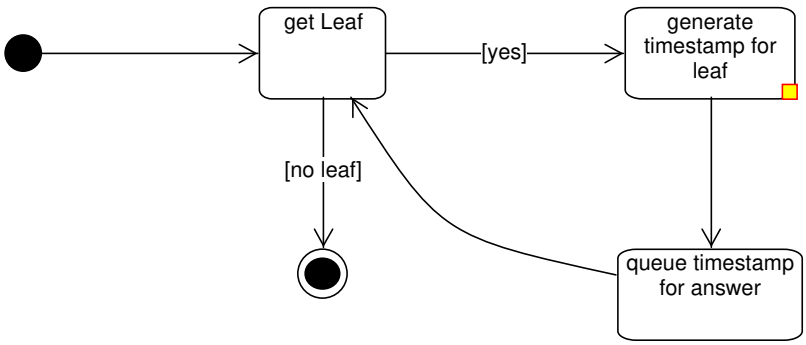


Figure 16: Timestamp generation state diagram

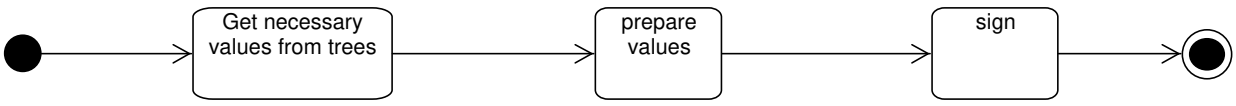


Figure 17: Generate timestamp for leaf state diagram

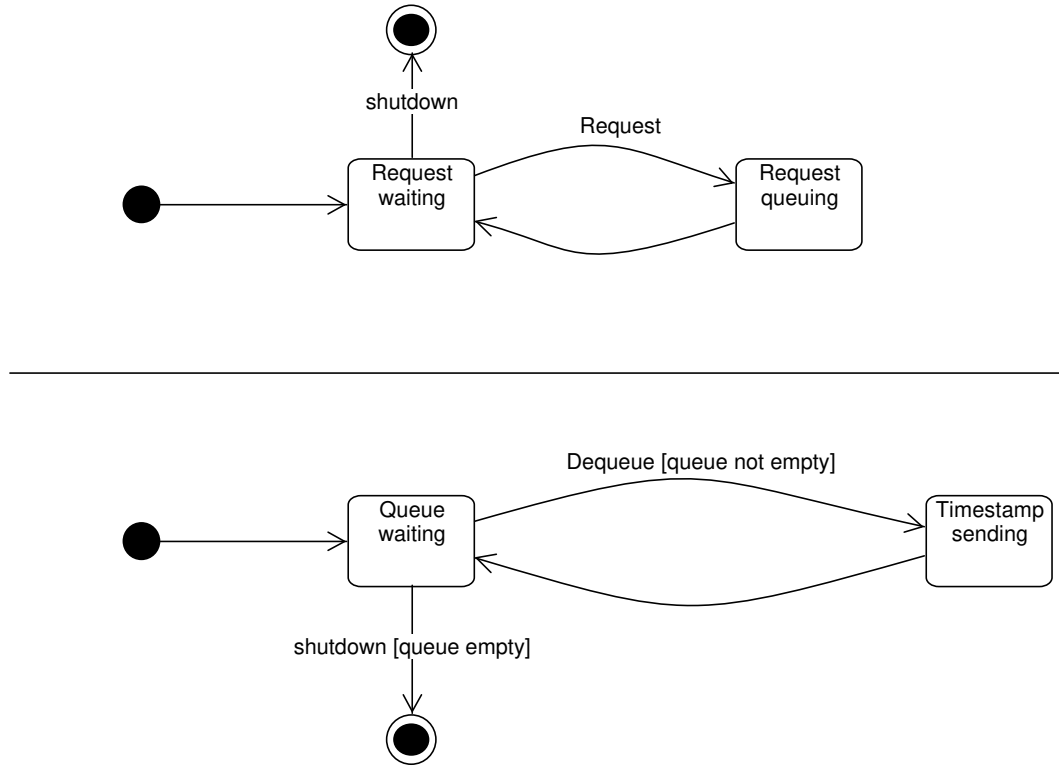


Figure 18: Network Answer state diagram

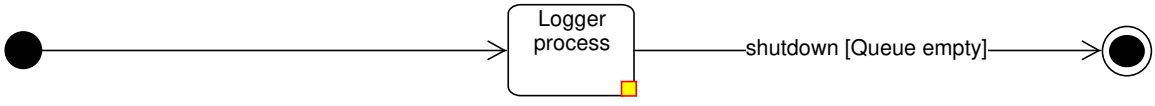


Figure 19: Logger state diagram

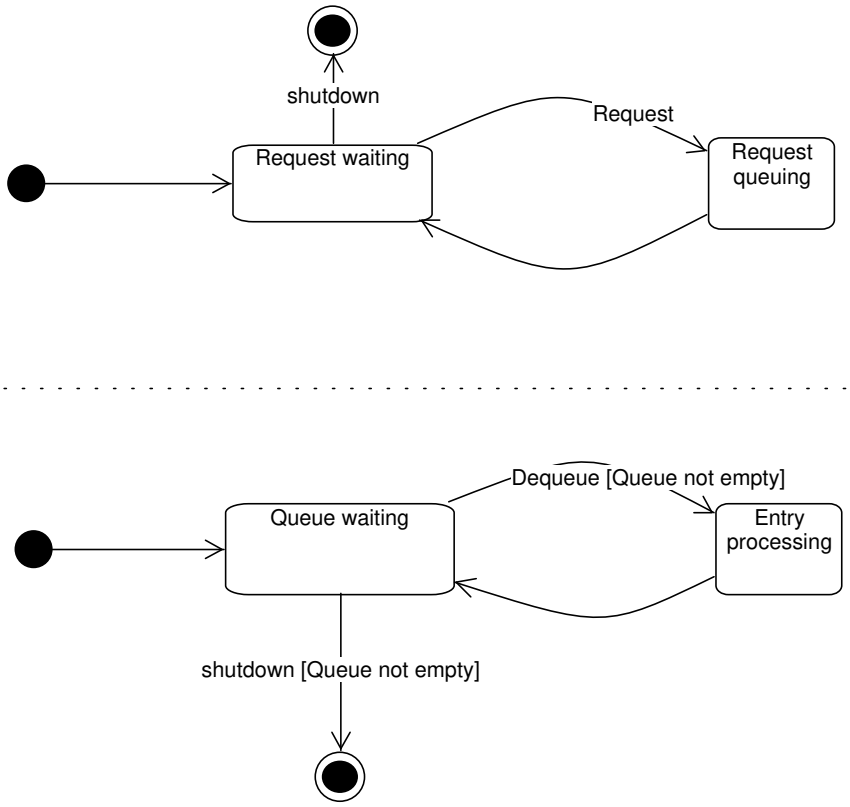


Figure 20: Logger process state diagram

timestamp. Afterwards, the "Round Value" is reconstructed using the data provided in the timestamp. If the computed "Round Value" is consistent with the one contained in the timestamp then the next step in the verification process is to compare this "Round Value" to the "Round Value" obtained from the STA repository. Finally, the verifier provides his system with the two "Big Round Values" that he founds in the "unmodifiable media"; the verifier's system gets all the necessary "Round Values" and "Root Round Values" from the STA and it checks the coherency of the two linking chains (one for each hash function).

This process is done at the client side and is not critical in terms of performances. A process is continuously working at the STA's side. It waits for verifications requests and answer with the round value and round tree value of the asked round. It also indicates if it is or not a Big round. Three connections are established with the STA. One to get the considered round value. One to get the preceding round values until the preceding Big round. The last one to get the next round values until the next Big round. The way to obtain the necessary round values can be implemented differently. Our concern here was the simplicity of implementation. The verification of the matching of the Big round values given by the STA compared to the one in the unmodifiable media has not been done. The forwarding of the Big round values to the unmodifiable media has also not been implemented.

7.3 The audit process

The auditor designates two "Big Rounds" that he fetches from a fixed media. The system behavior will be checked between these two "Big Round Values". For each round, the auditor's system gets all the hash values (leafs of the tree and "Special Nodes") and the "Round Value" from the STA. Then, it constructs the two trees and checks that the "Round Value" is consistent. These two steps

are repeated until all the considered rounds are checked or until an error has been found. In that way, all theoretically verifiable system behavior can be verified a posteriori.

7.4 The system start-up process

Here the most sensible issue is to be able to correctly start-up the system when an unexpected shutdown has occurred. If that is the case, the log will show an unfinished round; then the system marks all entries after the last complete round as invalid and publishes that round as a “Big Round”. If the log was consistent, it accesses the last valid “Round Value” in the log and publishes it as a “Big Round”. This process insures a fully verifiable behavior; we are able to detect non fully-processed requests. The recovery of the last round value when the log is not consistent (the last value in the log is not a valid round value) is not implemented in the actual version of our system.

7.5 The system shutdown process

The administrator signals the system to shutdown. No more timestamping requests are accepted. The system waits until the current round is finished and this “Round Value” is published as “Big Round”.

8 Description of the different classes

The class diagram of our system is represented in Figure 21. The active classes are the one that have a bold border.

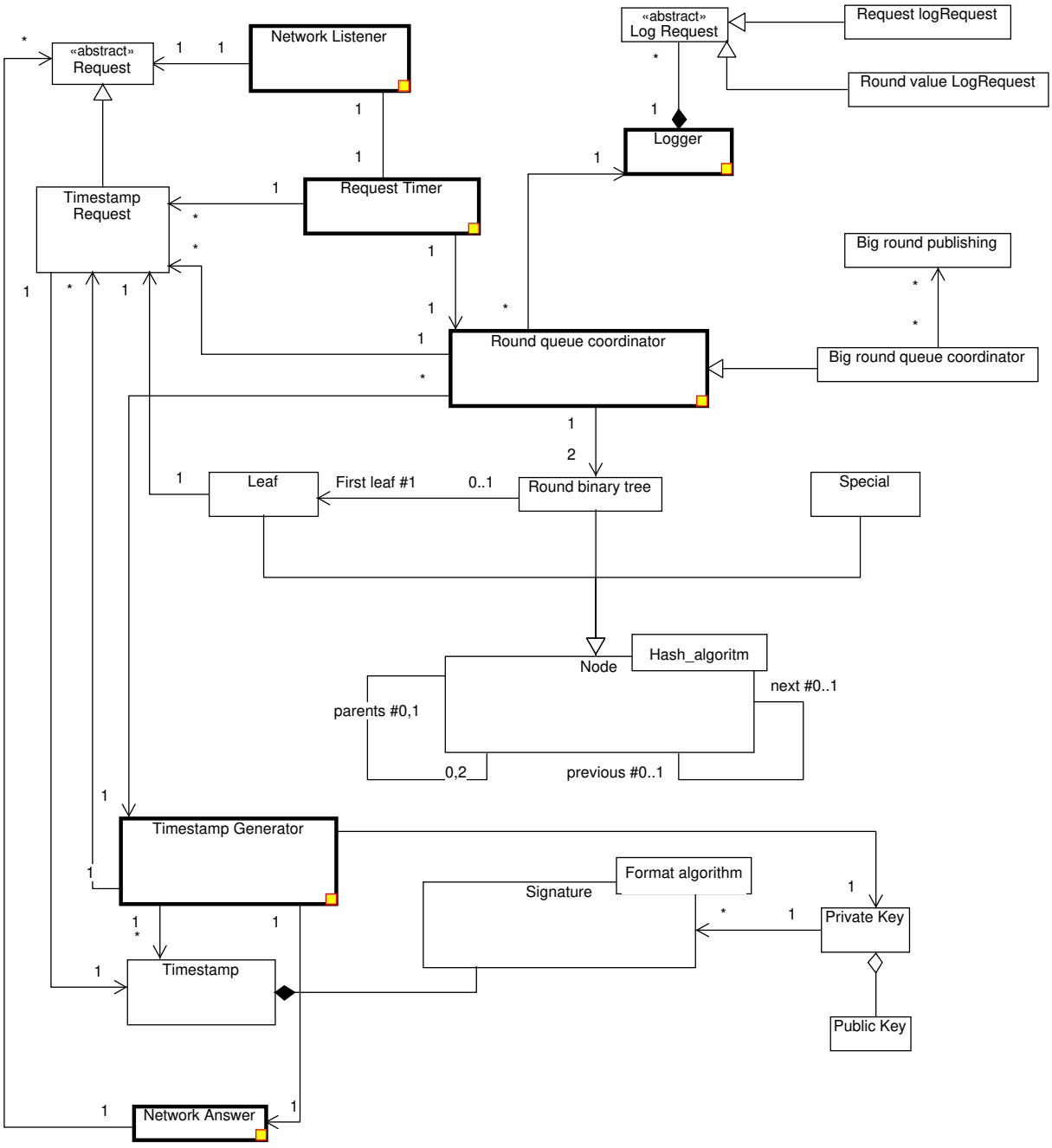


Figure 21: Class Diagram

The CRC cards of the active class are the following. It states the responsibility of those classes as well as the class that implement it.

Network listener	
• listen the network	(Socket)
• convert data stream into request	Request
• add request to Request-timer queue	Request-timer

Request timer	
• add Request to queue	
• time requests and round determination	Round queue coordinator
• add request to round	Round queue coordinator
• round creation	Round queue coordinator

Round queue coordinator	
• kill preceding round and log	
• add requests to queue and log	Hash tree queue and Logger
• compute round value and log	Round B. tree and Logger
• give computed trees to Timestamp generator	Timestamp generator

Logger	
• add requests to queue	Log request
• write queue request to disk	

Timestamp generator	
• get Tree	
• explore tree to generate timestamp for each request	Round B. tree
• (Optional) store timestamps	Timestamp storer
• send computed request to requester	Network answer

Network answer	
• add computed request to queue	
• answer queued computed request	Request, (Socket)

The attributes and operations of those classes are detailed in Annexe B.

In the class “Timestamp Request”, the “Process Time” has been mentionned to indicate the time of arrival but is not used in the actual version of the implementation.

9 Algorithms

9.1 Construction of the round tree

The two trees are constructed at the same time. We give the algorithm for just one hash function.

Algorithm 1 Tree construction

```
Begin
  construct the leafs with the queued requests
  while (number of nodes in the tree level > 1)
    process the node by two and construct the higher level
    if it remains one node then
      compute and hash a random value
      compute the node
    endif
  endwhile
//The tree round value is then computed
get the preceding round value to obtain the current round value
End
```

9.2 Generation of the timestamps

Algorithm 2 Timestamp generation

```
Begin
  while all the requests have not been answered
    get next leaf
    while we are not at the tree round value
      if this node position is R then
        get the left node value
      else
        get the right node value
      endif
      this node=parent node
    endwhile
    get the preceding round value
    put the round time, round number, request sequence in the timestamp
    sign the timestamp
    send the timestamp to the Network Answer
  endwhile
End
```

10 A timestamping system based on the accumulator technique

Next we describe an alternative system for timestamping which has also been developped for the TIMESEC project. This system is similar to the previous one in the dividing of the timestamping procedure in successive rounds. The main difference is the method used for processing (accumulating) all the timestamp requests of a round and generating the round value (see sections 3 and 4 for a description of both accumulation methods).

11 Overview of the global system

The main actors involved in the system are the STA (Secure Timestamp Authority), the clients and the verifiers.

The main use cases are the generation (by a client) and the verification (by a verifier) of a timestamp. In the first case the client wants a timestamp for a document he possesses. He submits a request to the STA and receives in return a timestamp certificate for his document. In the second case the verifier wants to check the correctness of a given timestamp certificate for a certain document.

The role of the STA is in processing all timestamp requests, making the necessary computations and sending back the timestamp certificates. It also generates an archive, which contains information on all succeeding timestamping rounds, and the linking between them. Finally it generates a log with all information necessary for reconstructing the round processes. This permits an audit of the system.

12 The timestamp generation process

This process allows a client of the system to obtain a timestamp certificate for a document in his possession. He will send a request to the STA and receive a timestamp in return. The certificate obtained proves the temporal information on his document. The scenario is as follows:

1. The client designates the document for which he wants a timestamp.
2. He generates a hash of this document using a secure, collision-free hash function.
3. He sends this hash value to the STA. This is the request.
4. At the end of the round the STA generates the round value, which depends on all requests of the round and on the previous round value.
5. For each request he generates the corresponding timestamp certificate.
6. He appends the round value to the archive.
7. He sends the timestamp certificates to the clients who requested them.
8. The STA also logs all important round information in the audit log.

Next we examine this process in more detail.

12.1 Use of a hash function

Instead of just sending his document, the client first produces a hash of it. In this way the document remains secret, and sending, processing and storing requests is simplified (because the hash is much shorter). The correspondence between the hash value and the document must be unique, hence he must use a secure, collision-free, hash function. The first step followed by the verifier is checking this correpondence, by applying the same hash function to the document and checking it with the hash contained in the timestamp certificate.

In the current implementation using the hash function RIPEMD-160 is suggested. The choice can also be left to the users: if the verifier agrees that a secure hash function was used by the client, he can trust the hash value contained in the certificate (by making the check with the hash function). RIPEMD-160 and SHA-1 both are good candidates, because they are well established algorithms (recently standardized by ISO/IEC). Both of them generate a hash value of 160 bits.

Optionally we could enforce to use both of these hash functions in parallel (as in the previous system). Then the timestamps would remain secure if one of them is unexpectedly broken. In this case the system would process both hash values, by performing a round accumulation for each hash function in parallel. The certificates will contain values for both hash functions, and in the the verification process seperate checks will be made for each hash function.

12.2 Submitting a request to the STA

To ask a timestamp for a document the client sends the hash value he generated in the previous step to the STA, along with his identity (this is the request). Our implementation uses a web interface for this step: the client enters a hash (up to 40 hexadecimal characters, which is equivalent to 160 bits) and his email-address, and submits it. The website is secured with SSL (providing encryption and server authentication) and also contains code for the RIPEMD-160 hash function used in the previous step.

12.3 Collecting the requests for a round

The STA collects all requests during a round. The hash values and corresponding identities which are received, are stored in two separate files. When the round finishes the system moves these files to a new directory, where the processing (round accumulation) starts. At the same time, new requests for the next round start to be collected in the old directory.

12.4 Round accumulation

The STA processes the collected hash values with the round accumulation technique described in section 4. Modular exponentiation is used as a one-way accumulator, in order to compute the round accumulation value Z and the partial values Z_i (for $1 \leq i \leq m$) for the hash values y_i (which were contained in the requests of the clients, and are up to 160 bits long).

In a straightforward implementation $m \cdot (m-1)$ exponentiations would be required to compute the partial values Z_i for m requests ($m-1$ exponentiations for each partial value). In a more careful implementation the number of operations can be reduced by reusing intermediate values: $(m/2+1) \cdot (m-1)$ exponentiations are sufficient. One more exponentiation allows the computation of the round accumulation value Z from one of the partial values Z_i.

12.5 Sending back the timestamps

The STA is now ready for sending back all timestamps to the clients who requested them. The timestamp for a particular request contains the hash value y_i (which was sent by the client to the STA), the partial value Z_i and the round accumulation value Z (this last value is the same for all clients of the same round). The certificate further includes the number and the time of the round.

The identities (email addresses) which were stored, are used for sending the timestamps to the clients who requested them.

12.6 Updating the archive with the round value

The STA now links the current round to the previous one. The round accumulation value Z which was computed is concatenated to the round value of the last round, and the result is hashed with the hash function RIPEMD-160 to obtain the new round value.

The round accumulation value Z and the new round value are then appended to the archive, together with the number and the time of the round (i.e., the time at which all computations for the round were completed).

Note that the linking procedure can be extended by applying a second hash function (SHA-1) in parallel, in order to protect against an unexpected break of one of the hash functions.

12.7 Updating the audit log

The STA also keeps a log containing all important values used in a round. For each round the hash values and identities contained in the requests are stored, as well as the computed partial values, the round accumulation value, and the actual round value (after linking with the previous round).

This allows any other independent party to check all the work done by the STA.

12.8 Publishing round values

At certain time intervals the STA should have a round value published in some authentic, unmodifiable and widely witnessed media. We call this a big round, and the time connected to such a round should be trusted by all participants to the system.

12.9 Receiving the timestamps

Each client receives the timestamp(s) that he requested in the past round. In our implementation he receives an email with a certificate that contains all the information which proves that the hash of his document was part of the round accumulation process. The round itself is identified by its number and time. The email further includes the time at which the request was sent and the machine (IP-address) it was sent from. A client should immediately verify a timestamp that he receives (see the next section for this procedure).

13 The timestamp verification process

This process allows a verifier given a document and corresponding timestamp certificate, to check the correctness of the certificate and the temporal information about the document contained in it. The scenario is as follows:

1. The verifier designates a document together with the corresponding timestamp certificate.
2. He generates a hash of the document (using the same hash function as the client did in the generation process).
3. He checks that this hash is identical to the hash value contained in the certificate.
4. He checks the structure of the certificate.
5. He uses the archive to check the linking with preceeding/succeeding rounds until trusted (published) values are encountered.

Next we examine this process in more detail.

13.1 Checking the hash value contained in the certificate

The first step for the verifier to check a document's timestamp, is to compute the hash of the document and check that the result is equal to the hash value contained in the timestamp certificate. For this he uses the same hash function as the client did in the first step of the timestamp generation process.

For a secure collision-free hash function (such as RIPEMD-160 or SHA-1) this proves that the timestamp belongs to the document given.

13.2 Checking the structure of the certificate

Next the verifier checks the structure of the certificate, by using the values contained in it in the equation that reconstructs the round accumulation value from the partial value and the document's hash value (see section 4). This involves one modular exponentiation. Code for performing this check is provided on the secure website (this code also needs the modulus of the system).

Due to the intractibility of the RSA problem this proves that the document's hash was part of the accumulation process that generated the round accumulation value.

13.3 Checking the round's time

The verifier already knows that the document's timestamp belongs to the round characterized by the round accumulation value contained in the certificate. He will now check the temporal information provided by this round.

The certificate includes the number and time of the round, and this information can be used to look up the round in the system's archive (which is accessible via the secure website). The archive provides the information necessary for checking the linking between all successive rounds (the secure website provides code for performing this check). In this way the verifier can follow the temporal chain in both directions until he encounters trusted (published) values. In that way the round is proven to be situated between two such trusted (big) rounds.

14 Security and efficiency of the accumulator technique

The main advantage of using the accumulator technique is that the size of the timestamps is constant, independent of the number of requests in a round. The drawback is that the modular exponentiation operation which is used as a one-way accumulator is less efficient than hashing. We have shown that the number of exponentiations necessary for the timestamp and round value generation is quadratic in the number of requests but with a careful implementation we have been able to reduce this number by about half.

Another drawback is that, contrary to the tree technique, we cannot preserve the relative position of requests of the same round. However this information could not be proven in the system using the tree.

With regards to security, as in the first system we need a secure, collision-free, hash function. The client needs to hash his document before sending the request to the server. The STA needs a hash function to link all successive rounds. In this system the security also depends on the intractibility of the RSA problem. For this we need a modulus that cannot be factored, for the time being a 1024-bit modulus should be sufficient for this, and this can be extended at a later time. The modulus must be provided by a *trusted* party, but this party can remain off-line, and needs not be involved further in the system.

15 User interface

As noted before, our implementation of the timestamping system using the accumulator technique uses a web interface, allowing clients to easily submit their timestamp requests. The website is also secured via SSL, which provides encryption for the communication and authentication of the server. In order to submit, the client enters the hash value of his document that he wants a timestamp for, and his email address to which the timestamp certificate will be sent. The webpage viewed by the users is shown in Annex C.

The archive providing the temporal information of the rounds and the linking information between them, is accessible on a seperate page (an example of which is shown in Annex D). Note that the values shown in the archive are in the format used by the multi-precision library that we used for our C-programming. This format first shows the length of a value (the number of 16-bit words in hexadecimal), and then the value itself in hexadecimal.

Explanation about the system and code for hashing documents and verifying timestamps is also provided for.

16 Issues to think about when designing a concrete system

The remarks that we make in this section come from our experience in designing and implementing a complete timestamping system (see also [10]). Our principal design requirement was to minimize the trust required in the third party issuing the timestamps.

Some of the remarks that we make here were already made previously in this report. We state them here again to summarize the important problems that must be solved when implementing a timestamping service and the solutions we have given to them.

We implemented the two techniques summarized in section 2. For the tree technique we added the feature of building two trees in parallel for each round, using two different hash functions (SHA-1 and RIPEMD-160). This allows to remain secure in case of an unexpected break of one of the hash functions used.

The client submits two hashes of the document he wants to timestamp using the same hash functions as the ones used by the STA. The STA has no knowledge of the kind of document that is being timestamped. It also can not check that the two hashes received have been computed from the same document or with the supposed hash functions. If this is not the case, the timestamped document will not be validated by the verification process, but this can not be detected in advance.

As is demonstrated in [11], signatures should not be used for the timestamping process. In our designs we only use it for STA authentication purposes when sending back the timestamp to the client.

A good timestamping technique should be auditable. In both of our designs, all the computations of the STA can be checked at any time. All the values timestamped, as well as the round values, are logged. With this information any outside entity can recompute and check all the round values issued by the STA.

System based on the tree technique. The binary tree is defined for a number of leafs (request) that is a power of 2. In general,this will not be the case . We could create fake requests to finish the tree, but it will add a lot of requests. Imagine that we have $2^n + 1$ requests, we need then to add $2^n - 1$ fake requests. A smarter solution is to add a random value only when needed. We add at most n values (one for each level of the tree). We call this node "special node". Another solution could be to use the 0 value or a fixed value instead. It is as secure as the solution we use if the hash functions are "perfect". As hash functions are only "presumably perfect", we though that we could make our design more secure with really few additional computations.

In our implementation the STA queues the requests, and computes the tree at the end of the round. At first sight, it could seem a better and more secure solution to build the tree as soon as the requests arrive and at the end of the round finish the computation of the tree by getting the last round value. In fact, this solution is harder to implement, and has no effect on the security because no one can check that the STA does not perform any reordering of the requests before it publishes the round value.

Our design uses a round queue for each round. There is a different thread assigned to each round queue, which is mainly waiting for the end of the round. Once the round is closed, the thread wakes up and constructs the round tree finally obtaining the round value. The time indicated in the timestamp is only determinated when the request is put in the round queue. The round queue does not accept any request after the end of the round. In that way, the computation of the tree can begin immediately. Another solution could be to determine the time as soon as the request is received. However we detected during the implementation design that it will then be difficult to know which are the requests belonging to a round waiting to be processed once the time for the round has expired. On the other hand, if there is an abnormal delay between the reception of a

request and the queuing, then the computation of the tree and then the issuing of the timestamps will be delayed.

The values timestamped as well as the round values are logged on a file. This feature permits to define an audit process.

When checking the validity of a timestamp, the verifier asks the STA for the necessary values. These values are all the round values between the last and next big rounds. With this material, the verifier will be able to check the validity of the timestamp without having to trust the STA at all.

System based on the accumulator technique. The second system uses modular exponentiation instead of hashing for accumulating the requests. As in the previous case all requests received during a round are collected, and when the round finishes they will be processed by the one-way accumulator (modular exponentiation operation).

For each request y_j (see section 4 for the notations used in this discussion) we need to generate the corresponding partial value Z_j. We also need the round accumulation value Z, which is the same for all requests processed in the same round. The number of modular exponentiations required for computing these values for all requests of the round is quadratic in m (the number of requests), but by careful implementation we can reduce this number to about half the number required in a straightforward implementation.

To use the system, clients have to submit a 160-bit hash (using RIPEMD-160) of their document. The STA will make the necessary computations when the current round has finished, and then send back all timestamps (containing the necessary values) to the corresponding clients. Succeeding rounds are also linked to each other by using the RIPEMD-160 hash function. As in the other implementation, we can improve this further by using a second hash function (SHA-1) in parallel.

The security of the system also relies on the modulus used. This modulus should be provided by some trusted third party (which can be different from the STA, and remain off-line), and its prime factors must remain secret from anyone else. A length of 1024 bits seems appropriate for the time being, and can be extended at a later time.

All requests and round values are logged in a file, which permits auditing of the system.

A particular timestamp is verified by checking the correspondence between the document and its hash value (input of the request), checking the equation $Z = Z_j^{y_j} \mod N$, and finally checking the linking of succeeding rounds until trusted (published) round values are obtained.

17 Annexes

Annexe A

Annexe B

Annexe C

Annexe D

References

[1] S.-S. Alhir. *UML in a Nutshell*. O'Reilly, 1998. ISBN 1-56592-448-7.

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

[3] J. Benaloh and M. de Mare. Efficient broadcast time-stamping. Technical Report TR 91-1, Clarkson University Department of Mathematics and Computer Science, August 1991.

[4] J. Benaloh and M. de Mare. One-way accumulators: 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.

[5] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999. ISBN 0-201-57168-4.

[6] M. Fowler and K. Scott. *UML Distilled*. Addison Wesley, 1998. ISBN 0-201-32563-2.

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

[8] S. Haber and W.S. Stornetta. Secure names for bit-strings. In *Proceedings of the 4th ACM Conference on Computer and Communication Security*, pages 28–35. ACM Press, April 1997.

[9] H. Massias. A survey of accumulators. Technical report, UCL Crypto Group Technical Report, February 1998.

[10] H. Massias, X. Serret Avila, and J.-J. Quisquater. Design of a secure timestamping service with minimal trust requirements. In A. Barbé, E.C. van der Meulen, and P. Vanroose, editors, *Twentieth Symposium on Information Theory in the Benelux*, pages 79–86, May 1999.

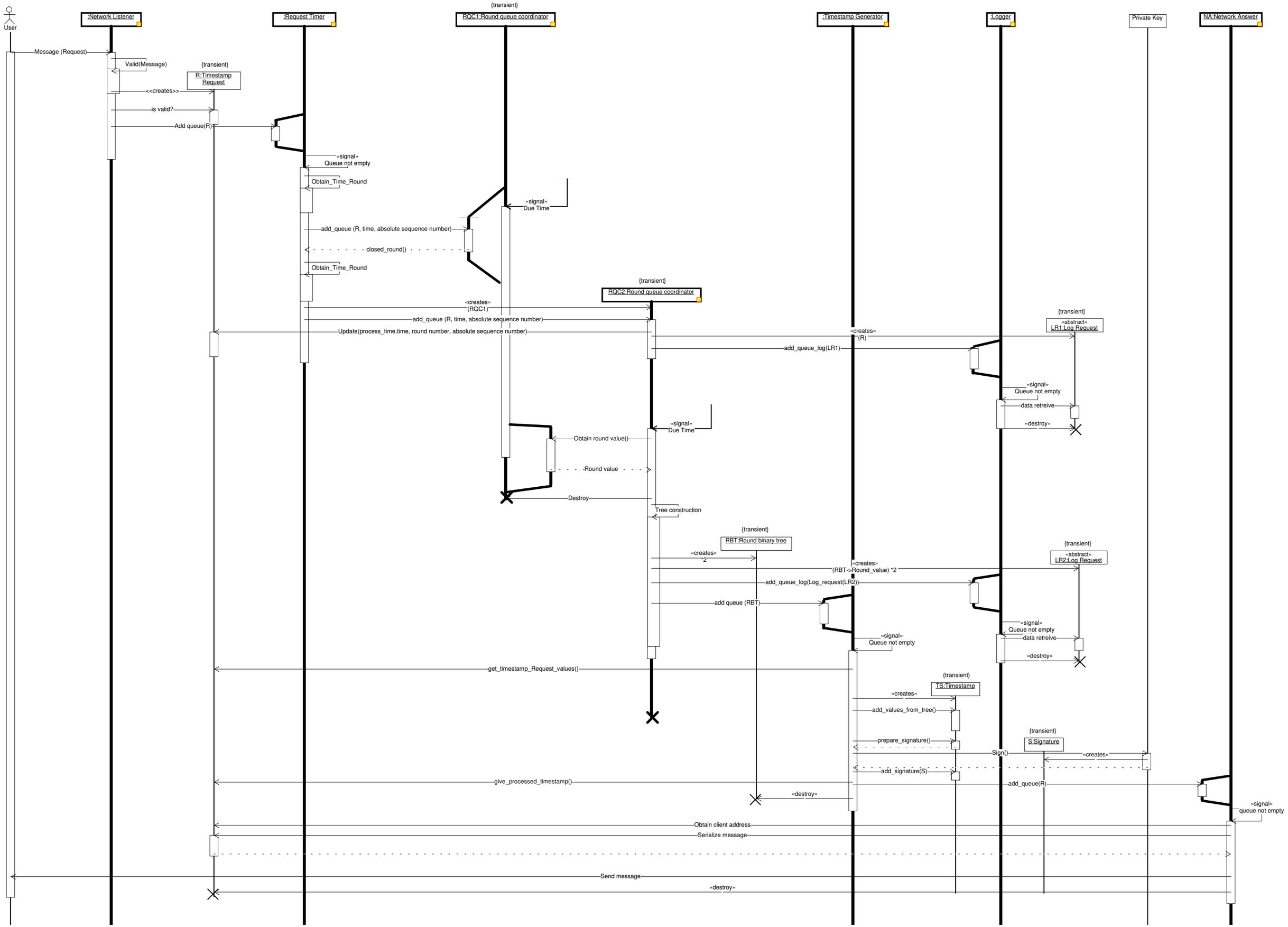
[11] H. Massias, X. Serret Avila, and J.-J. Quisquater. Timestamps: Main issues on their use and implementation. Accepted at Wet Ice '99, Stanford CA, June, 1999.

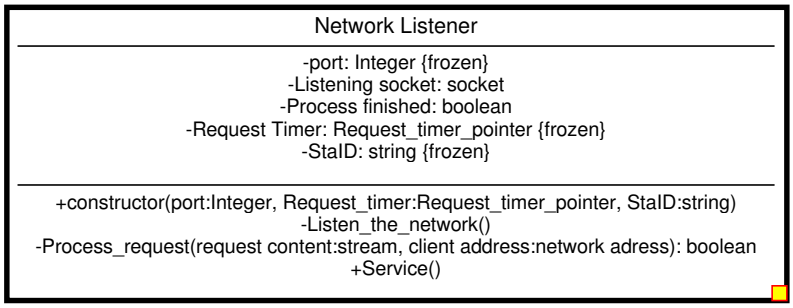
[12] H. Massias and J.-J. Quisquater. Time and cryptography. Technical report, TIMESEC Project (Federal Governement Project, Belgium), 1997. Available at `http://www.dice.ucl.ac.be/crypto/TIMESEC.html`.

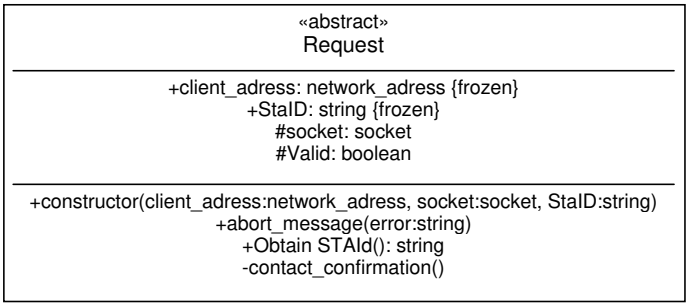
[13] R. Pooley and P. Stevens. *Using UML: Software Engineering with Objects and Components*. Addison-Wesley, 1998. ISBN 0-201-36067-5.

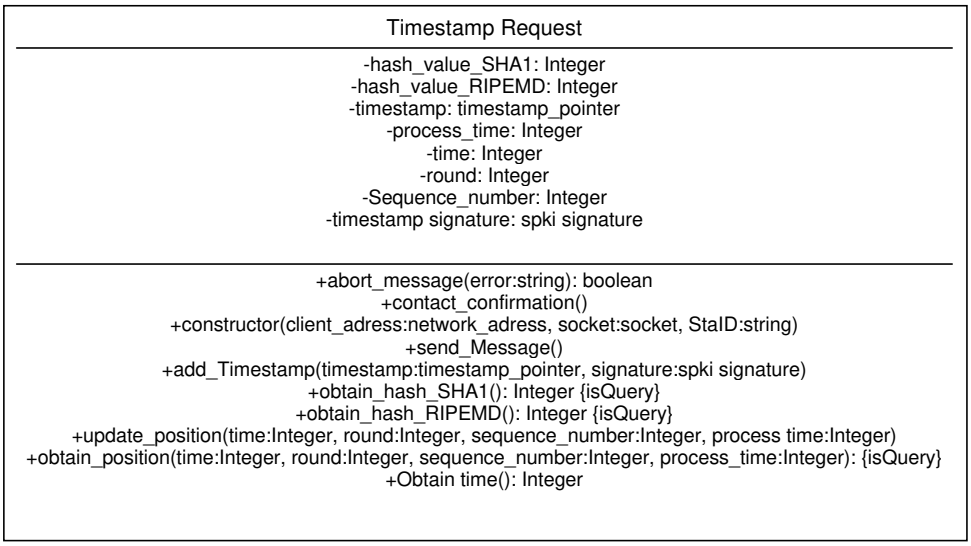
[14] B. Preneel, B. Van Rompay, J.-J. Quiquater, and H. Massias. Evaluation methodology for security primitives. Technical report, TIMESEC Project (Federal Governement Project, Belgium), 1997. Available at `http://www.dice.ucl.ac.be/crypto/TIMESEC.html`.

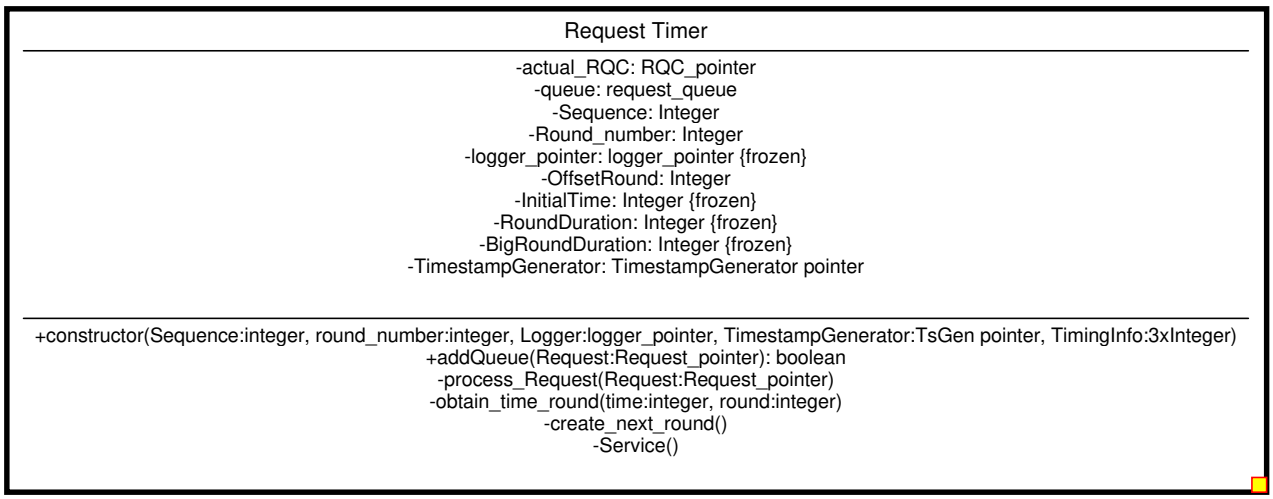
[15] B. Preneel, B. Van Rompay, J.-J. Quiquater, H. Massias, and X. Serret Avila. Design of a timestamping system. Technical report, TIMESEC Project (Federal Governement Project, Belgium), 1998. To be available at `http://www.dice.ucl.ac.be/crypto/TIMESEC.html`.

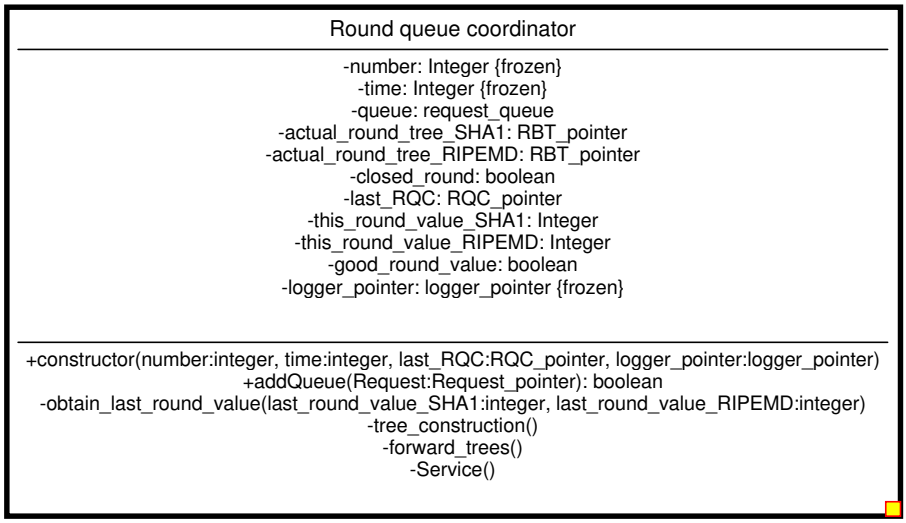

User
:Network Listener
:Request Timer
{transient}
RQC1:Round queue coordinator
:Timestamp Generator
:Logger
Private Key
NA:Network Answer
Message (Request)
Valid(Message)
{transient}
R:Timestamp Request
<<creates>>
is valid?
Add queue(R)
«signal»
Queue not empty
Obtain_Time_Round
«signal»
Due Time
add_queue (R, time, absolute sequence number)
closed_round()
Obtain_Time_Round
{transient}
RQC2:Round queue coordinator
«creates»
(RQC1)
add_queue (R, time, absolute sequence number)
Update(process_time,time, round number, absolute sequence number)
{transient}
«abstract»
LR1:Log Request
«creates»
(R)
add_queue_log(LR1)
«signal»
Queue not empty
data retreive
«destroy»
«signal»
Due Time
Obtain round value()
Round value
Destroy
Tree construction
{transient}
RBT:Round binary tree
«creates»
*2
{transient}
«abstract»
LR2:Log Request
«creates»
(RBT->Round_value) *2
add_queue_log(Log_request(LR2))
add queue (RBT)
«signal»
Queue not empty
«signal»
Queue not empty
data retreive
«destroy»
get_timestamp_Request_values()
{transient}
TS:Timestamp
«creates»
add_values_from_tree()
{transient}
S:Signature
prepare_signature()
Sign()
«creates»
add_signature(S)
give_processed_timestamp()
«destroy»
add_queue(R)
«signal»
queue not empty
Obtain client address
Serialize message
Send message
«destroy»

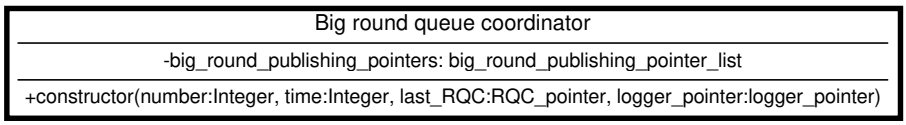

Network Listener
-port: Integer {frozen}
-Listening socket: socket
-Process finished: boolean
-Request Timer: Request_timer_pointer {frozen}
-StaID: string {frozen}
+constructor(port:Integer, Request_timer:Request_timer_pointer, StaID:string)
-Listen_the_network()
-Process_request(request content:stream, client address:network adress): boolean
+Service()

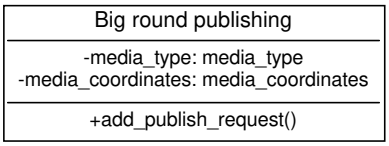

«abstract»
Request
+client_adress: network_adress {frozen}
+StaID: string {frozen}
#socket: socket
#Valid: boolean
+constructor(client_adress:network_adress, socket:socket, StaID:string)
+abort_message(error:string)
+Obtain STAId(): string
-contact_confirmation()

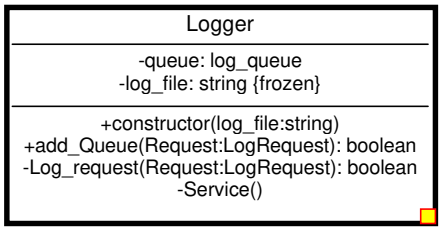

Timestamp Request
-hash_value_SHA1: Integer
-hash_value_RIPEMD: Integer
-timestamp: timestamp_pointer
-process_time: Integer
-time: Integer
-round: Integer
-Sequence_number: Integer
-timestamp signature: spki signature
+abort_message(error:string): boolean
+contact_confirmation()
+constructor(client_adress:network_adress, socket:socket, StaID:string)
+send_Message()
+add_Timestamp(timestamp:timestamp_pointer, signature:spki signature)
+obtain_hash_SHA1(): Integer {isQuery}
+obtain_hash_RIPEMD(): Integer {isQuery}
+update_position(time:Integer, round:Integer, sequence_number:Integer, process time:Integer)
+obtain_position(time:Integer, round:Integer, sequence_number:Integer, process_time:Integer): {isQuery}
+Obtain time(): Integer

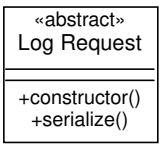

Request Timer
-actual_RQC: RQC_pointer
-queue: request_queue
-Sequence: Integer
-Round_number: Integer
-logger_pointer: logger_pointer {frozen}
-OffsetRound: Integer
-InitialTime: Integer {frozen}
-RoundDuration: Integer {frozen}
-BigRoundDuration: Integer {frozen}
-TimestampGenerator: TimestampGenerator pointer
+constructor(Sequence:integer, round_number:integer, Logger:logger_pointer, TimestampGenerator:TsGen pointer, TimingInfo:3xInteger)
+addQueue(Request:Request_pointer): boolean
-process_Request(Request:Request_pointer)
-obtain_time_round(time:integer, round:integer)
-create_next_round()
-Service()

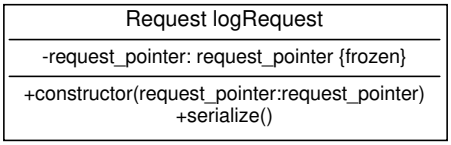

Round queue coordinator
-number: Integer {frozen}
-time: Integer {frozen}
-queue: request_queue
-actual_round_tree_SHA1: RBT_pointer
-actual_round_tree_RIPEMD: RBT_pointer
-closed_round: boolean
-last_RQC: RQC_pointer
-this_round_value_SHA1: Integer
-this_round_value_RIPEMD: Integer
-good_round_value: boolean
-logger_pointer: logger_pointer {frozen}
+constructor(number:integer, time:integer, last_RQC:RQC_pointer, logger_pointer:logger_pointer)
+addQueue(Request:Request_pointer): boolean
-obtain_last_round_value(last_round_value_SHA1:integer, last_round_value_RIPEMD:integer)
-tree_construction()
-forward_trees()
-Service()

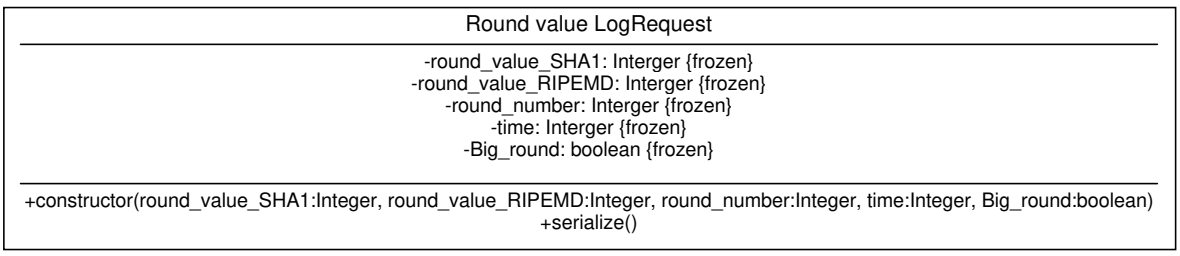

Big round queue coordinator
-big_round_publishing_pointers: big_round_publishing_pointer_list
+constructor(number:Integer, time:Integer, last_RQC:RQC_pointer, logger_pointer:logger_pointer)


Big round publishing
-media_type: media_type
-media_coordinates: media_coordinates
+add_publish_request()


Logger
-queue: log_queue
-log_file: string {frozen}
+constructor(log_file:string)
+add_Queue(Request:LogRequest): boolean
-Log_request(Request:LogRequest): boolean
-Service()


«abstract»
Log Request
+constructor()
+serialize()


Request logRequest
-request_pointer: request_pointer {frozen}
+constructor(request_pointer:request_pointer)
+serialize()


Round value LogRequest
-round_value_SHA1: Interger {frozen}
-round_value_RIPEMD: Interger {frozen}
-round_number: Interger {frozen}
-time: Interger {frozen}
-Big_round: boolean {frozen}
+constructor(round_value_SHA1:Integer, round_value_RIPEMD:Integer, round_number:Integer, time:Integer, Big_round:boolean)
+serialize()

Algoritm:
Hash Algoritm

Node

-parent: Node_pointer
-localisation: {L,R,T} {frozen} (T means that this node is the tree round value)
-next: Node_pointer
-previous: Node_pointer {frozen}
-value: Hash_value {frozen}

+constructor(children_L:Node_pointer, children_R:Node_pointer, previous:Node_pointer)
+next(): Node_pointer {isQuery}
+previous()(): Node_pointer {isQuery}
+localisation(): {L,R} {isQuery}
+parent(): Node_pointer {isQuery}
+value(): Hash_value {isQuery}

Round binary tree

-first_leaf: Node_pointer {frozen}
-number: Integer {frozen}
-time: Integer {frozen}

+constructor(time:integer, number:integer, first_Request:Node_pointer)

Leaf

-Request: Request_pointer {frozen}

+constructor(Request:Request_pointer, previous:Node_pointer)
+Request(): Request_pointer {isQuery}

Special

+constructor(rand_seed:integer, previous:Node_pointer)

Timestamp Generator

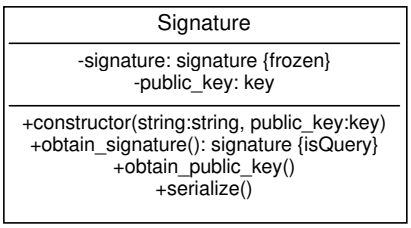
-queue: RBT_queue
-private_key: key {frozen}

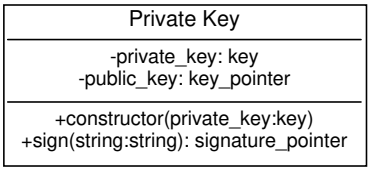
+add_Queue(RBT_SHA1:RBT_pointer, RBT_RIPEMD:RBT_pointer): boolean
-process_tree_request()
-process_client_request()
-Service()

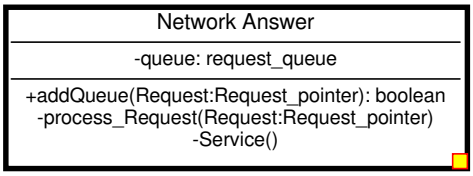
Timestamp

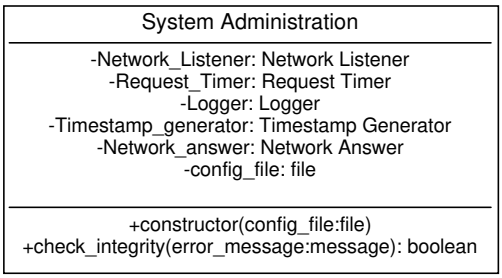
-time: Integer {frozen}
-round_number: Integer {frozen}
-sequence_number: Integer {frozen}
-signature: signature_pointer
-tree_branch_SHA1: string {frozen}
-tree_branch_RIPEMD: string {frozen}

+constructor(tree_branch_SHA1:string, tree_branch_RIPEMD:string)
+serialize(): string
+prepare_signature(): string
+add_signature(signature:signature_pointer)


Signature
-signature: signature {frozen}
-public_key: key
+constructor(string:string, public_key:key)
+obtain_signature(): signature {isQuery}
+obtain_public_key()
+serialize()


Private Key
-private_key: key
-public_key: key_pointer
+constructor(private_key:key)
+sign(string:string): signature_pointer


Network Answer
-queue: request_queue
+addQueue(Request:Request_pointer): boolean
-process_Request(Request:Request_pointer)
-Service()


System Administration
-Network_Listener: Network Listener
-Request_Timer: Request Timer
-Logger: Logger
-Timestamp_generator: Timestamp Generator
-Network_answer: Network Answer
-config_file: file
+constructor(config_file:file)
+check_integrity(error_message:message): boolean

COSIC Time Stamping Authority

Hash value:

Email:

Request Time Stamp

View the archive (roundstamps and linking values)

How to use the system

Technical overview

The RIPEMD-160 hash function

Linux executables: Check timestamps, Check linking, Hash a binary file using RIPEMD-160

This is an experimental system developped for the Belgian TIMESEC project, funded by the OSTC.

More info:
bart.vanrompay@esat.kuleuven.ac.be

Thanks to:
antoon.bosselaers@esat.kuleuven.ac.be (MP library)
joris.claessens@esat.kuleuven.ac.be (web interface)

Archive: round and linking values

% example, using a 512-bit modulus and a round time interval of 10 minutes
% ROUNDSTAMP = the round accumulation value,
% 512-bits long (generated by modular exponentiation)
% LINKING VALUE = the round value,
% (after linking to the previous round value), 160-bits long (generated with RIPEMD-160)

```
INITIAL LINKING VALUE
   a
4e35 b5af 9d3e b2b8 1166 de91 bc8b 383d 31b9 d4b0

ROUNDSTAMP 1: GMT+01 TIME = Mon Dec 13 14:00:05 1999
  20
b347 fd6c e46a 4419 bb8d 754c 167e 0eaf b9e8 e863 c1f1 e360 e22d 6790 ebaa 90cf
b7da 827d 9510 6b16 f742 349d bcb5 9754 6eed 0ba0 eb54 4134 9211 38b9 22cc 752b

LINKING VALUE
   a
7328 feca 0652 ce73 31b2 cd74 d2f0 ae90 bc73 d68f

ROUNDSTAMP 2: GMT+01 TIME = Mon Dec 13 14:10:05 1999
  20
1b4f 9816 791c ed5f 16ca 99a6 caed 79b8 728b b8a8 79b3 62f7 1fde fb0e b73d ae83
cc6b 5f47 52c8 9c5c f3b7 1734 0e7b 946d 86fb 8f53 321c 8f3d 2f75 907d 58e6 4030

LINKING VALUE
   a
c447 37f7 7b6f 026c 38bf 4ca0 b1b0 629a 19e4 aa02

ROUNDSTAMP 3: GMT+01 TIME = Mon Dec 13 14:20:05 1999
  20
9c5e 3f95 ea1c d135 02cd a4f8 e86b b684 c042 986a 2a1d 4b31 ac25 9c50 b028 7b04
f439 5ac8 8875 33de 63e6 e023 dffb c115 577f 4e2a a32e bc5b 37a0 5fc3 71f1 594a

LINKING VALUE
   a
b857 d0e6 7bcf ae5a 1bd1 3dda d116 b845 d1e6 5b47

ROUNDSTAMP 4: GMT+01 TIME = Mon Dec 13 14:30:05 1999
  20
24a8 9b18 abc8 f072 410c 65a8 67bc 2156 e72a ee0e 2626 6457 bc73 37a5 d072 137a
a60c e7d2 72bf 28e1 0252 279c c135 c5d9 8eb1 8a3b 1b0b ff44 f321 1f6c 095d 66e6

LINKING VALUE
   a
47fb 2db0 0a03 3b8a c33e 841a f136 4c3c e5dc 8aed

ROUNDSTAMP 5: GMT+01 TIME = Mon Dec 13 14:40:05 1999
  20
b551 6c3f a6da d3a8 c8d9 5c90 2dba f8b6 5132 7bb9 0e13 215d 70d1 2d02 ecf7 cc28
84b4 6bbd 409d 7722 84cc 1f8a a154 5e14 113d cecb 30eb 38f0 74d7 a985 4fe9 a369

LINKING VALUE
   a
30a2 9cd4 5c9e f2ce b978 6082 142f 640f 55fe 3d1c
```

US005136646A

United States Patent [19]

Haber et al.

[11] **Patent Number: 5,136,646**

[45] **Date of Patent: Aug. 4, 1992**

[54] **DIGITAL DOCUMENT TIME-STAMPING WITH CATENATE CERTIFICATE**

[75] Inventors: **Stuart A. Haber**, New York, N.Y.; **Wakefield S. Stornetta, Jr.**, Morristown, N.J.

[73] Assignee: **Bell Communications Research, Inc.**, Livingston, N.J.

[21] Appl. No.: **666,896**

[22] Filed: **Mar. 8, 1991**

[51] **Int. Cl.**[5] **H04L 9/00**; H04L 9/30

[52] **U.S. Cl.** **380/49**; 380/23; 380/25; 380/30

[58] **Field of Search** 380/3-5, 380/9, 10, 28, 30, 49, 50

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,145,568	3/1979	Ehrat	380/50 X
4,625,076	11/1986	Okamoto et al.	380/30 X
4,868,877	9/1989	Fischer	380/30 X
4,881,264	11/1989	Merkle	380/50 X
4,972,474	11/1990	Sabin	380/28
5,001,752	3/1991	Fischer	380/30 X

OTHER PUBLICATIONS

"The MD4 Message Digest Algorithm", R. L. Rivest, Crypto '90 Abstracts, Aug. 1990, pp. 281–291.

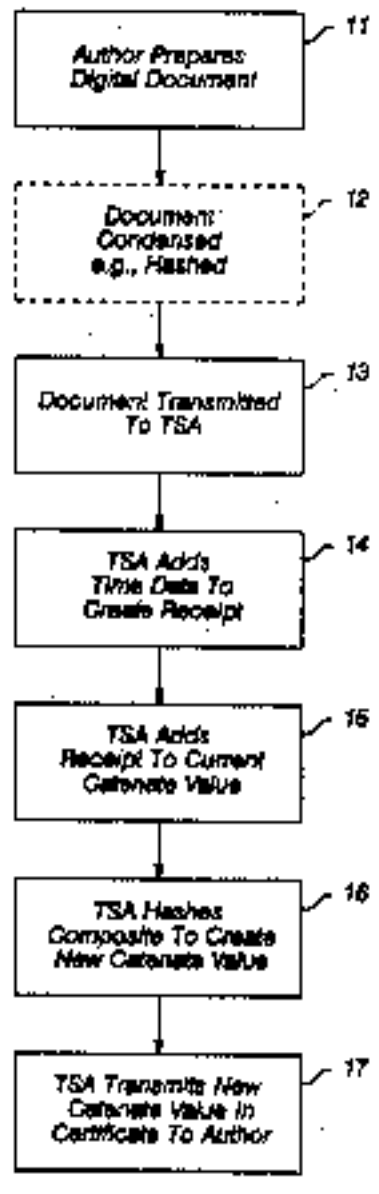
Primary Examiner—Bernarr E. Gregory

Attorney, Agent, or Firm—Leonard Charles Suchyta; Lionel N. White

[57] **ABSTRACT**

A system for time-stamping a digital document, for example any alphanumeric, video, audio, or pictorial data, protects the secrecy of the document text and provides a tamper-proof time seal establishing an author's claim to the temporal existence of the document. Initially, the document may be condensed to a single number by means of a one-way hash function, thereby fixing a unique representation of the document text. The document representation is transmitted to an outside agency where the current time is added to form a receipt. The agency then certifies the receipt by adding and hashing the receipt data with the current record catenate certificate which itself is a number obtained as a result of the sequential hashing of each prior receipt with the extant catenate certificate. The certified receipt bearing the time data and the catenate certificate number is then returned to the author as evidence of the document's existence. In later proof of such existence, the certificate is authenticated by repeating the certification steps with the representation of the alleged document, the alleged time data, and the catenate certificate number appearing in the agency's records immediately prior to the certificate number in question. Only if the alleged document is identical to the original document will the original and repeat certificate numbers match.

13 Claims, 2 Drawing Sheets



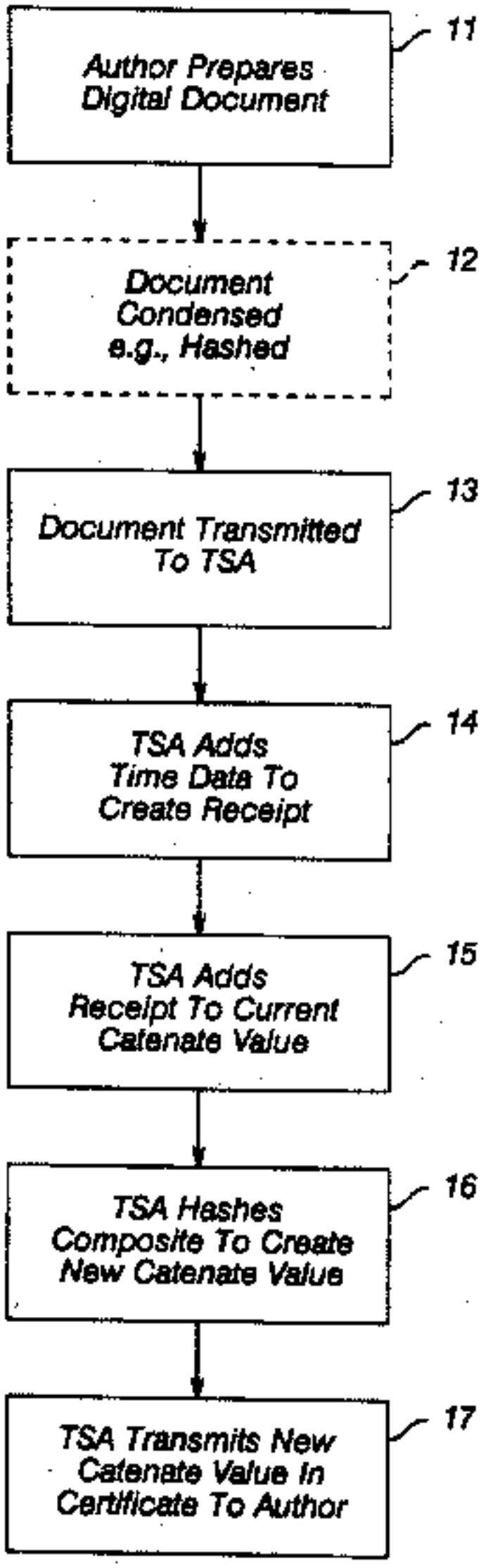

Author Prepares Digital Document
11
Document Condensed e.g., Hashed
12
Document Transmitted To TSA
13
TSA Adds Time Data To Create Receipt
14
TSA Adds Receipt To Current Catenate Value
15
TSA Hashes Composite To Create New Catenate Value
16
TSA Transmits New Catenate Value In Certificate To Author
17

FIG. 1

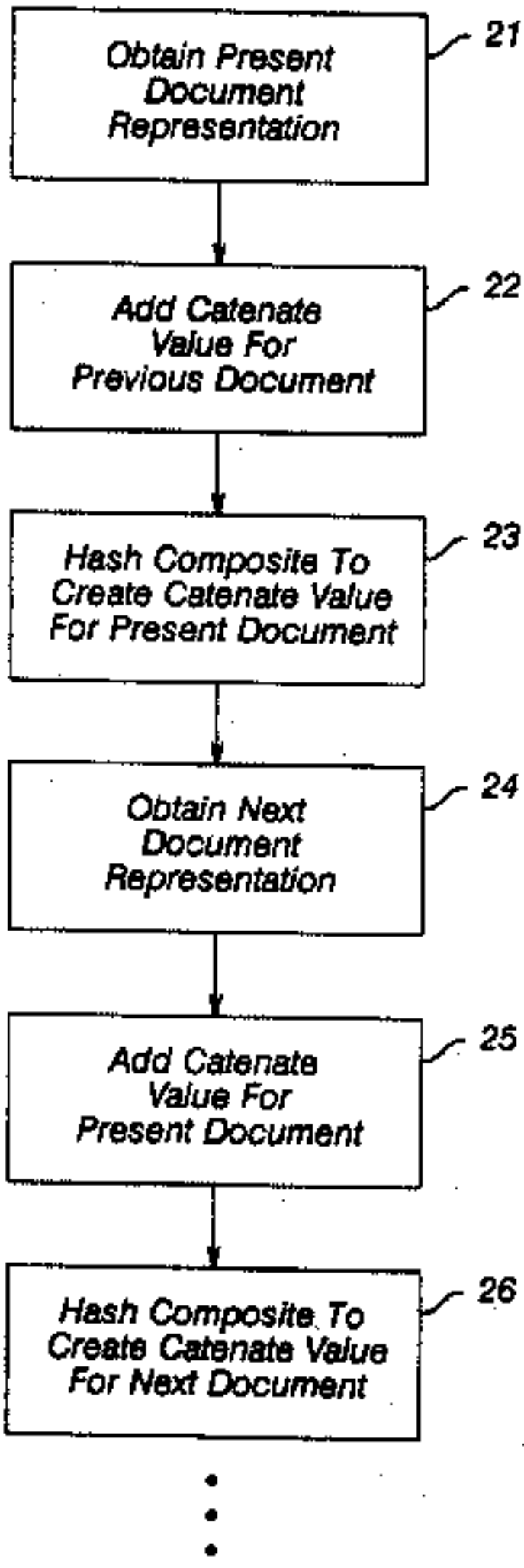

Obtain Present Document Representation
21
Add Catenate Value For Previous Document
22
Hash Composite To Create Catenate Value For Present Document
23
Obtain Next Document Representation
24
Add Catenate Value For Present Document
25
Hash Composite To Create Catenate Value For Next Document
26

FIG. 2

DIGITAL DOCUMENT TIME-STAMPING WITH CATENATE CERTIFICATE

BACKGROUND OF THE INVENTION

In many situations there is a need to establish the date on which a document was created and to prove that the text of a document in question is in fact the same as that of the original dated document. For example, in intellectual property matters it is often crucial to verify the date on which a person first put into writing the substance of an invention. A common procedure for thus "time-stamping" an inventive concept comprises daily notations of one's work in a laboratory notebook. Indelibly dated and signed entries are made one after another on each page of the notebook where the sequentially numbered, sewn-in pages make it difficult to revise the record without leaving telltale signs. The validity of the record is further enhanced by the regular review and signed witnessing by a generally disinterested third party. Should the time of the concept become a matter for later proof, both the physical substance of the notebook and the established recording procedure serve as effective evidence in substantiating the fact that the concept existed at least as early as the notebook witness date.

The increasingly widespread use of electronic documents, which include not only digital representations of readable text but also of video, audio, and pictorial data, now poses a serious threat to the viability of the "notebook" concept of establishing the date of any such document. Because electronic digital documents are so easily revised, and since such revisions may be made without telltale sign, there is available limited credible evidence that a given document truly states the date on which it was created or the message it originally carried. For the same reasons there even arises serious doubt as to the authenticity of a verifying signature. Without an effective procedure for ensuring against the surreptitious revision of digital documents, a basic lack of system credibility prevents the efficiencies of electronic documentation from being more widely implemented.

Some procedures are presently available for verifying electronic document transmissions; however, such procedures are limited in application to bilateral communications. That is, in such communications the sender essentially desires to verify to the receiver the source and original content of the transmitted document. For example, "private key" cryptographic schemes have long been employed for message transmission between or among a limited universe of individuals who are known to one another and who alone know the decrypting key. Encryption of the message ensures against tampering, and the fact that application of the private key reveals the "plaintext" of the transmitted message serves as proof that the message was transmitted by one of the defined universe. The time of creation of the message is only collaterally established, however, as being not later than its receipt by the addressee. This practice thus fails to provide time-stamp evidence that would be useful in an unlimited universe at a later date.

A more broadly applicable verifying communication procedure, that of "public key" cryptography, has been described by Diffie and Hellman ("New Directions in Cryptography", *IEEE Transactions On Information Theory*, Vol IT-22, November 1976, pp. 644–654). While this scheme expands the utilizing universe to a

substantially unlimited number of system subscribers who are unknown to one another, but for a public directory, verifiable communications remain bilateral. These limitations persist, since although a public key "signature", such as that which entails public key decryption of a message encrypted with the private key of the transmitter, provides any member of the unlimited universe with significant evidence of the identity of the transmitter of the message, only a given message recipient can be satisfied that the message existed at least as early as the time of its receipt Such receipt does not, however, provide the whole universe with direct evidence of time of the message's existence Testimony of a such a recipient in conjunction with the received message could advance the proof of message content and time of its existence, but such evidence falls victim to the basic problem of ready manipulation of electronic digital document content, whether by originator or witness

Thus, the prospect of a world in which all documents are in easily modifiable digital form threatens the very substance of existing procedures for establishing the credibility of such documents As a means of providing an answer to this burgeoning problem, we disclosed in our copending U.S. Pat. application Ser. No. 07/561,888, file Aug. 2, 1990, a system of verification by which a digital document may be so fixed in time and content that it can present, at least to the extent currently recognized in tangible documents, direct evidence on those issues.

The method described there entails transmittal of a document to an outside agency where current time data are incorporated with at least a portion of a digital representation of the document. In order to prevent collusive misstamping by the agency, one or more agencies are selected at random or an agency is required to incorporate into the time stamp receipt at least the time and a portion of identifying data from one or more temporally adjacent receipts Although this procedure accomplishes the two-fold goals of effective time-stamping, i.e., to fix the time and content of a document and to prevent collusive misdeeds of author and agent-witness, any subsequent personal interaction between participating authors may be burdensome, particularly in later proof stages where the comparison of contemporary receipts is required.

SUMMARY OF THE INVENTION

The present invention represents an improvement on our above-mentioned system and provides a reliable and more adaptable method of time-stamping digital documents that continues to maintain the two essential characteristics of accepted document verification. First, the content of a document and a time stamp of its existence are "indelibly" incorporated into the digital data of the document so that it is not possible to change any bit of the resulting time-stamped data without such a change being apparent. In this manner, the state of the document content is fixed at the instant of time-stamping. Second, the time at which the digital document is stamped is certified by a cryptographic summary, or catenation, procedure that deters the incorporation of a false time statement. In essence, the method transfers control of the time-stamping step from the author to an independent agent and removes from the author the ability to influence the agent in the application of other than a truthful time stamp.

One embodiment of the present invention presumes a number of document authors distributed throughout a communication network. Such authors may be individuals, companies, company departments, etc., each representing a distinct and identifiable, e.g., by ID number or the like, member of the author universe. This universe would be supported by a central record repository and would, in essence, constitute the clientele of such an outside time-stamping agency (TSA).

In this particular application, as depicted in FIG. 1 of the drawing, the method entails an author's preparation of a digital document, which may broadly comprise any alphanumeric, audio, or pictorial presentation, and the transmission of the document, preferably in a condensed representative form, to the TSA. The TSA time-stamps the document to create a receipt by adding digital data signifying the current time, concatenates the receipt with the current cryptographic catenation of its prior time stamp receipts, and creates a new catenation from the composite document by means of a deterministic function, such as discussed in greater detail below. The resulting catenate value is then included with time and other identifying data in a document, now a certificate of the temporal existence of the original document, which is transmitted back to the author where it will be held for later use in any required proof of such existence.

To ensure against interception of confidential document information during transmission to the TSA, and to reduce the digital bandwidth required for transmission of an entire document, the author may optionally convert the digital document string to a unique value having vastly condensed digital size by means of a deterministic function which may, for example, be any one of a number of algorithms known in the art as "one-way hash functions". Such an application of hash functions has been described, among others, by Damgard in his discussions on the improvement of security in document signing techniques ("Collision-Free Hash Functions and Public Key Signature Schemes", *Advances in Cryptology—Eurocrypt* '87, Springer-Verlag, LNCS, 1988, Vol. 304, pp. 203–217). In practice of the present invention, however, the "one-way" characteristic typical of a hashing algorithm serves an additional purpose; that is, to provide assurance that the document cannot be secretly revised subsequent to the time the TSA applies its time stamp and incorporates the document into the catenate certificate.

A hashing function provides just such assurance, since at the time a document, such as an author's original work or a composite receipt catenation, is hashed there is created a representative "fingerprint" of its original content from which it is virtually impossible to recover that document. Therefore, the time-stamped document is not susceptible to revision by any adversary of the author. Nor is the author able to apply an issued time-stamp certificate to a revised form of the document, since any change in the original document content, even to the extent of a single word or a single bit of digital data, results in a different document that would hash to a completely different fingerprint value. Although a document cannot be recovered from its representative hash value, a purported original document can nonetheless be proven in the present time-stamping procedure by the fact that a receipt concatenation comprising a true copy of the original document representation will always hash to the same catenate value as is contained in the author's certificate, assuming use of the original hashing algorithm.

Any available deterministic function, e.g. a one-way hash function such as that described by Rivest ("The MD4 Message Digest Algorithm", *Advances in Cryptology—Crypto* '90, Springer-Verlag, LNCS, to appear), incorporated herein by reference, may be used in the present procedure. In the practice of the invention, such a hashing operation is optionally employed by the author to obtain the noted benefit of transmission security, although it might be effected by the TSA if the document were received in plaintext form. In whatever such manner the document content and incorporated time data are fixed against revision, there remains the further step, in order to promote the credibility of the system, of certifying to the members of an as yet unidentified universe that the receipt was in fact prepared by the TSA, rather than by the author, and that the time indication is correct, i.e., that it has not, for instance, been fraudulently stated by the TSA in collusion with the author.

To satisfy these concerns, the TSA maintains a record of its sequential time-stamping transactions by adding each new receipt to its current catenation and applying its deterministic function, e.g. hashing, the composite to obtain a new catenation. This catenation, itself a value resulting from the hashing process, is included on the receipt or certificate returned to the author and serves to certify the indicated time stamp. Confirmation of the certificate at a later time involves rehashing the combination of the author's time receipt and the next previous catenate value in the TSA records. The resulting generation of the author's catenate certificate value proves to the author and to the universe at large that the certificate originated with the TSA. This result also proves the veracity of the time-stamp itself, since all original elements of the original receipt must be repeated in order to again generate, by the hashing function, the original catenate certificate value.

The process of the invention relies upon the relatively continuous flow of documents from the universe of authors through the facilities of the TSA. For each given processed document D_k, from an author, A_k, the TSA generates a time-stamp receipt which includes, for example, a sequential receipt transaction number, r_k, the identity of the author, for example by ID number ID_k, or the like, a digital representation, e.g. the hash, H_k, of the document, and the current time, t_k. The TSA then includes these receipt data, or any representative part thereof, with the catenate certificate value, C_{k-1}, of the immediately preceding processed document D_{k-1}, of author, A_{k-1}, thereby bounding the time-stamp of document D_k, by the independently established earlier receipt time, t_{k-1}.

The composite data string, r_k, ID_k, H_k, t_k, C_{k-1}), is then hashed to a new catenate value, C_k, that is entered with transaction number, r_k, in the records of the TSA, and is also transmitted to A_k, as the catenate certificate value, with the time-stamp receipt data. In like manner, a certificate value derived from the hashing of C_k with time stamp elements of the receipt for document D_{k+1}, would be transmitted to author, A_{k+1}. Thus, each of the time-stamped catenate certificates issued by the TSA is fixed in the continuum of time and none can be falsely prepared by the TSA, since any attempt to regenerate a catenate certificate number from a hash with the next prior certificate would reveal the discrepancy.

In a more general application of the invention, as shown in FIG. 2, the representation, e.g., a hash, of a particular document is simply concatenated with the catenate certificate value of the next previous document and the deterministic function representation, again a hash, for example, of this composite is then generated and retained as the record catenate value for the particular document Each subsequent document in the growing series is similarly processed to expand the record which itself would serve as a reliable certification of the position each such document occupies in the series, or more broadly viewed, in the continuum of time. This embodiment of the invention provides a reliable method by which an organization, for instance, could readily certify the sequence and continuity of its digital business documents and records.

Additional variations in the process of the invention might include the accumulation of documents, preferably in hashed or other representative form, generated within an author organization over a period of time, e.g. a day or more depending upon the extent of activity, with the collection being hashed to present a single convenient document for time-stamping and certification. As an alternative, an organizational designee might serve as a resident "outside" agency who would maintain a catenate certificate record of organization documents by means of the present procedure and on a regular basis would transmit the then current catenate certificate to a TSA. In this manner the sequence of an organization's business records would be established both within the organization and externally through the TSA.

Also, the implementation of process embodiments might readily be automated in simple computer programs which would directly carry out the various steps of hashing, transmitting, and concatenating original document representations, applying current time stamps, generating and recording catenate certificate values, and providing receipt certificates.

THE DRAWING

The present invention will be described with reference to the accompanying drawing of which:

FIG. 1 is a flow diagram of an embodiment of the time-stamping process according to the invention; and.

FIG. 2 is a flow diagram of the general catenation process according to the invention.

DESCRIPTION OF THE INVENTION

The following exemplary application of the present invention, as depicted in the steps of the drawing, will serve to further describe the time-stamping process. For convenience in the presentation of this example, the deterministic function employed is the md4 hashing algorithm described by Rivest, as mentioned above; however, the function actually selected by a TSA could be any of various available algorithms. Whatever algorithm is implemented, records of its identity and period of use must be maintained for later proof of certified receipts.

The present time-stamping procedure begins, as at step 11 of the drawing, with the preparation of a digital document by the author, e.g. A_k. As previously noted, this digital document may be the digital form or representation of any alphanumeric text or video, audio, pictorial or other form of fixed data. Although the present process may be used with documents of any length, the following excerpt is amply representative of a document, D_k, for which time-stamping is desired:

> . . . the idea in which affirmation of the world and ethics are contained side by side . . . the ethical acceptance of the world and of life, together with the ideals of civilization contained in this concept . . . truth has no special time of its own. Its hour is now—always.
>
> Schweitzer

If the author so desires, the document, D_k, may, for the purposes of security as well as to reduce the required transmission bandwidth, be condensed by means, for example, of the md4 algorithm. As indicated by the optional, dashed step 12, the document is thus hashed to a value, H_k, of a standard 128 bit format which, expressed in base 16, appears as:

ee2ef3ea60df10cb621c4fb3f8dc34c7

It should be noted at this point that the hexadecimal and other numerical value representations used in this example are not in such form crucial to the implementation of the invention. That is to say, any portion or other distinct representation of those values selected according to a given procedure would function as well.

Author, A_k, whose assigned identification number, ID_k, is 634 in a 1000 member author universe, then transmits the document, at step 13, to the system TSA in the identifying message, (ID_k, H_k), which appears:

634, ee2ef3ea60df10cb621c4fb3f8dc34c7

as a request that the document be time-stamped.

The TSA, at step 14, prepares the receipt for document, D_k, by adding a sequential receipt transaction number, r_k, of 1328, for example, and a statement of the current time, t_k. This time statement might be a standard binary representation of computer clock time or simply a literal statement, e.g., 19:46:28 Greenwich Mean Time on Mar. 6, 1991, in order to allow the final time-stamp certificate to be easily read. The receipt then comprises the string, (r_k, t_k, ID_k, H_k), which appears as follows;

1328, 194628GMT06MAR91, 634,
ee2ef3ea60ef10cb621c4fb3f8dc34c7

In accordance with the invention, the records of the TSA at this time contain a catenation of all its prior receipt transactions in the form, for example, of the values resulting from the hashing of each consecutive receipt with the record catenation to that time. This catenate record would thus have been developed as follows. The receipt of first transaction (r_{k-1}) was hashed with an initial datum value, e.g., the hash of the identification of the TSA, to yield the first catenate value, C_1, which was then used as the certificate value for that first transaction. In the next transaction, the receipt was concatenated with $C_1$and the composite hashed to yield the second catenate certificate value, C_2, and so on through the entire history of the TSA time-stamping operation.

Assume now that the document, D_{k-1}, immediately preceding that of the present example had been processed by the TSA, in its 1327th receipt transaction, to yield as the catenate certificate value, C_{k-1}:

26f54eae925156b1f0d6047c2de6e0fcf

In step 15 of the process, the TSA now concatenates with this value the receipt for D_k to obtain:

26f54eae92516b1f0d6047c2de6e0fcf, 1328,
194628GMT06MAR91, 634,
ee2ef3ea60df10cb621e4fb3f8dc34c7

This composite is then hashed by the TSA, at step 16, to yield as the new catenate certificate value, C_k:

46f7d75f0fbea95e96fc38472aa28ca1

The TSA then adds this value to its records and prepares and transmits to author, A_k, at step 17, a time-stamp certificate, including this catenate certificate value, which might appear as:

Transaction Number:	1328
Client ID Number:	634
Time:	19:46:28 Greenwich Mean Time
Date:	06 March 1991
Certificate Number:	46f7d75f0fbea95e96fc38472aa28ca1

The procedure would be repeated by the TSA for each subsequent time stamp request. Assuming the next request from A_{k+1} was received with the document in the form of its hash H_{k+1}, as:

201, 882653ee04d511dbb5e06883aa27300b

at 19:57:52 GMT on Mar. 6, 1991, the composite concatenation would appear:

46f7d75f0fbea95e96fc38472aa28ca1, 1329,
195752GMT06MAR1991, 201,
882653ee04d511dbb5e06883aa27300b

and the certificate returned to A_{k+1} would read:

Transaction Number:	1329
Client ID Number:	201
Time:	19:57:52 Greenwich Mean Time
Date:	06 March 1991
Certificate Number:	d9bb1b11d58bb09c2763e7915fbb83ad

When, at a later date, author, A_{k+1}, desires to prove the authenticity of document, D_{k+1}, as that which was received and dated by the TSA on Mar. 6, 1991 at 19:57:52, the records of the TSA are examined to obtain the catenate certificate value, C_k, of the next previous transaction, 1328, which appears as:

46f7d75f0fbea95e96fc38472aa28ca1

The alleged document is then reduced to the form in which it was transmitted to the TSA, e.g, as its hash, and this value is then concatenated with C_k and the remaining data from the certificate of A_{k+1}. The resulting composite, assuming the alleged document to be authentic, now appears as:

46f7d75f0fbea95e96fc38472aa28ca1, 1329,
195752GMT06MAR1991, 201,
882653ee04d511dbb5e06883aa27300b

which, when hashed, produces the correct catenate certificate value:

d9bb1b11d58bb09c2763e7915fbb83ad

thereby proving the alleged document to be D_{k+1}. Otherwise, a revised document would hash to a different value and the composite of which it is an element would hash to a catenate certificate value different from that stated in the certificate of transaction number 1329.

If further proof were demanded, for example upon an adversary allegation that C_{k+1} had been falsified after the fact of a document revision, the certificate and the submitted, e.g. hashed, document of A_k, who is identified from TSA records, would be employed in an attempt to regenerate the subsequent, questioned certificate value, C_{k+1}. If that value were correct, D_{k+1} would be proved. As an alternative, the certificate value, C_{k+1}, could be proved by the regeneration of the subsequent catenate certificate value, C_{k+2}, from the certificate data and submitted document of A_{k+2}, since no feasible revision could be made to that later document which would result in a match of C_{k+2} if C_{k+1} were not the same as existed at the time of the transaction, 1330, processing D_{k+2}.

In the more general record catenation procedure depicted in FIG. 2, the documents in a growing series are processed, within an organization or by a TSA, as each is generated. At step 21, a new document representation, such as would be generated by a hashing deterministic function algorithm, becomes available and, at step 22, is concatenated with the current record catenate value that was generated in the processing of the previous document. This composite is then processed, e.g., hashed, at step 23, to generate the new catenate value for the present document. This value may be separately recorded and utilized for inclusion in a certificate, or simply retained in the processing system for application to the next document which is presented at step 24. The subsequent processing steps 25, 26 are applied to this document representation, and the process repeats with each new document in its turn.

The procedures described and variants suggested herein for the practice of this time-stamping process and the various other embodiments which will become apparent to the skilled artisan in the light of the foregoing description are all nonetheless to be included within the scope of the present invention as defined by the appended claims.

What is claimed is:

1. A method of certifying the temporal sequence of digital documents in a series of such documents characterized in that said method comprises:

a) generating a digital representation of a specified one of the documents in said series; and

b) generating a catenate certificate value representation for said specified document by applying a selected deterministic function algorithm to a catenation comprising said digital representation and the catenate certificate value representation for the document immediately prior in said series to said specified document.

2. A method according to claim 1

characterized in that the method further comprises repeating the recited steps with each subsequent document in said series.

3. A method according to claim 2

characterized in that said method further comprises maintaining a sequential record of said series documents with their respective catenate certificate value representations.

4. A method according to claim 2

characterized in that each said digital representation is generated by applying to said document one or another deterministic function algorithm which

may be the same as or different from said selected deterministic function algorithm.

5. A method according to claim 4

characterized in that said one or another deterministic function algorithm is any one-way hashing algorithm.

6. A method according to claim 2

characterized in that said selected deterministic function algorithm is any one-way hashing algorithm.

7. A method of time-stamping a digital document which comprises transmitting a digital representation of said document to an outside agency, creating at said outside agency a receipt comprising a digital representation of then current time and at least a portion of a digital representation of said digital document, and certifying said receipt at said outside agency

characterized in that the certifying of said receipt comprises:

a) concatenating a digital representation of said receipt with a representation of a prior catenate certificate value to form a composite; and

b) generating a catenate certificate value for said receipt by applying a selected deterministic function algorithm to said composite.

8. A method of time-stamping a digital document according to claim 7

characterized in that said outside agency maintains a record comprising the catenate certificate values of prior time-stamping transactions.

9. A method of time-stamping a digital document according to claim 7

characterized in that said prior certificate value representation comprises at least a portion of the catenate certificate value of the immediately preceding recording time-stamping transaction.

10. A method of time-stamping a digital document according to claim 7

characterized in that said selected deterministic function algorithm is any one-way hashing algorithm.

11. A method of time-stamping a digital document according to claim 7

characterized in that said transmitted digital document representation comprises at least a portion of the digital representation of the value derived by applying to said digital document one or another deterministic function algorithm which may be the same as or different from said selected deterministic function algorithm.

12. A method of time-stamping a digital document according to claim 7

characterized in that said receipted digital document representation comprises at least a portion of the digital representation of the value derived by applying to said digital document one or another deterministic function algorithm which may be the same as or different from said selected deterministic function algorithm.

13. A method of time-stamping a digital document according to claim 12

characterized in that said one or another deterministic function is any one-way hashing algorithm.

* * * * *

US005136647A

United States Patent [19]

Haber et al.

[11] Patent Number: 5,136,647

[45] Date of Patent: Aug. 4, 1992

[54] METHOD FOR SECURE TIME-STAMPING OF DIGITAL DOCUMENTS

[75] Inventors: Stuart A. Haber, New York, N.Y.; Wakefield S. Stornetta, Jr., Morristown, N.J.

[73] Assignee: Bell Communications Research, Inc., Livingston, N.J.

[21] Appl. No.: 561,888

[22] Filed: Aug. 2, 1990

[51] Int. Cl.[5] H04L 9/00; H04L 9/30

[52] U.S. Cl. 380/49; 380/23; 380/25; 380/30

[58] Field of Search 364/200, 900; 380/3, 380/4, 30, 49, 850, 5, 9, 10, 28

[56] References Cited

U.S. PATENT DOCUMENTS

4,145,568	3/1979	Ehrat	380/50 X
4,405,829	9/1983	Rivest et al.	380/30
4,972,474	11/1990	Sabin	380/49 X

OTHER PUBLICATIONS

"New Directions in Cryptography", W. Diffie & M. E. Hellman, IEEE *Transactions On Information Theory*, vol. IT-22, Nov. 1976, pp. 644-654.

"Collision-Free Hash Functions & Public Key Signature Schemes", I. B. Damgdard, *Advanaces in Cryptology-Eurocrypt* '87, Springer-Verlag, LNCS, 1988, vol. 304, pp. 203-216.

"Pseudorandom Generation From One-Way Functions", R. Impagliazzo & L. A. Levin, *Proc.* 21st STOC, pp. 12-24, ACM, 1989.

"The MD4 Message Digest Algorithm", R. L. Rivest *Crypto* '90 *Abstracts*, Aug. 1990, pp. 281-291.

Alan G. Konheim, *Cryptography, a Primer*; (John Wiley & Sons, Inc.; 1981); pp. 331-333.

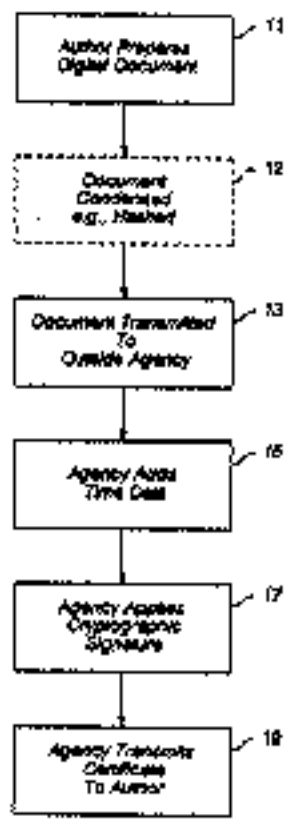
Primary Examiner—Bernarr E. Gregory

Attorney, Agent, or Firm—Leonard Charles Suchyta; Lionel N. White

[57] ABSTRACT

A system for time-stamping a digital document, including for example text, video, audio, or pictorial data, protects the secrecy of the document text and provides a tamper-proof time seal establishing an author's claim to the temporal existence of the document. Initially, the author reduces the document to a number by means of a one-way hash function, thereby fixing a unique representation of the document text. In one embodiment of the invention the number is then transmitted to an outside agency where the current time is added to form a receipt which is certified by the agency using a public key signature procedure before being returned to the author as evidence of the document's existence. In later proof of such existence, the certificate is authenticated by means of the agency's public key to reveal the receipt which comprises the hash of the alleged document along with the time seal that only the agency could have signed into the certificate. The alleged document is then hashed with the same one-way function and the original and newly-generated hash numbers are compared. A match establishes the identify of the alleged document as the time-stamped original. In order to prevent collusion in the assignment of a time stamp by the agency and thus fortify the credibility of the system, the receipt is linked to other contemporary receipts before certification by the agency, thereby fixing a document's position in the continuum of time. In another embodiment, a plurality of agencies are designated by means of random selection based upon a unique seed that is a function of the hash number of the document to be time-stamped. Thus being denied the ability to choose at will the identity of an agent, the author cannot feasibly arrange for falsification of a time stamp.

18 Claims, 3 Drawing Sheets



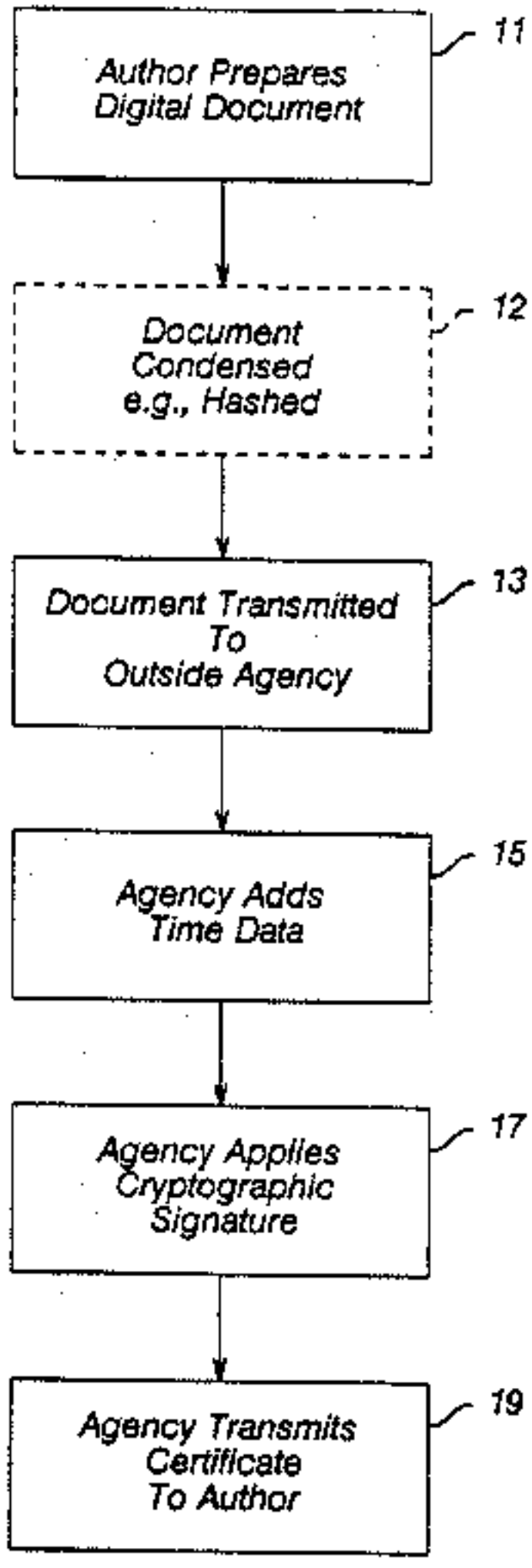

Author Prepares Digital Document
11
Document Condensed e.g., Hashed
12
Document Transmitted To Outside Agency
13
Agency Adds Time Data
15
Agency Applies Cryptographic Signature
17
Agency Transmits Certificate To Author
19

FIG. 1

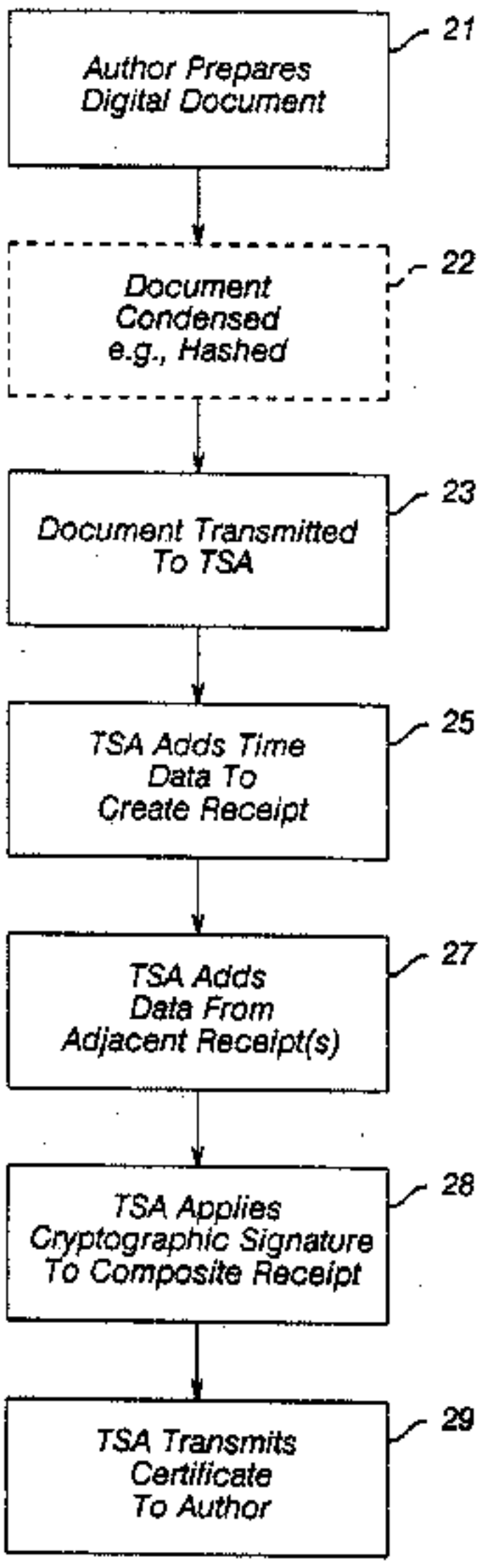

Author Prepares Digital Document
21
Document Condensed e.g., Hashed
22
Document Transmitted To TSA
23
TSA Adds Time Data To Create Receipt
25
TSA Adds Data From Adjacent Receipt(s)
27
TSA Applies Cryptographic Signature To Composite Receipt
28
TSA Transmits Certificate To Author
29

FIG. 2

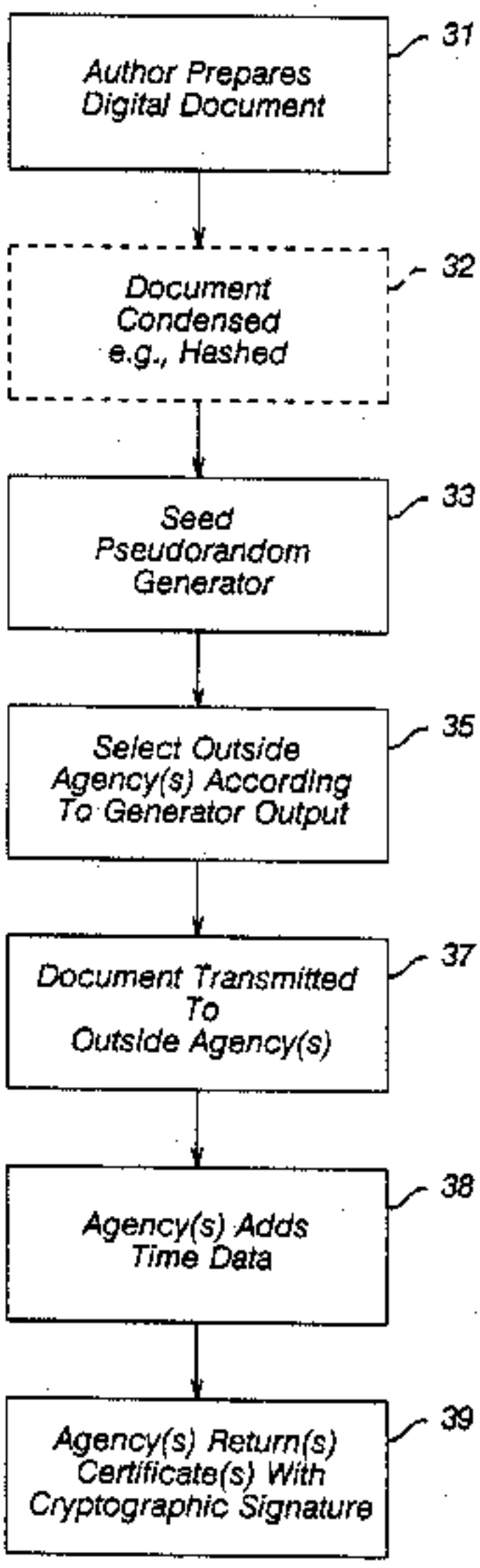

Author Prepares Digital Document
31
Document Condensed e.g., Hashed
32
Seed Pseudorandom Generator
33
Select Outside Agency(s) According To Generator Output
35
Document Transmitted To Outside Agency(s)
37
Agency(s) Adds Time Data
38
Agency(s) Return(s) Certificate(s) With Cryptographic Signature
39

FIG. 3

METHOD FOR SECURE TIME-STAMPING OF DIGITAL DOCUMENTS

BACKGROUND OF THE INVENTION

In many situations there is a need to establish the date on which a document was created and to prove that the text of a document in question is in fact the same as that of the original dated document. For example, in intellectual property matters it is often crucial to verify the date on which a person first put into writing the substance of an invention. A common procedure for thus "time-stamping" an inventive concept comprises daily notations of one's work in a laboratory notebook. Indelibly dated and signed entries are made one after another on each page of the notebook where the sequentially numbered, sewing pages make it difficult to revise the record without leaving telltale signs. The validity of the record is further enhanced by the regular review and signed witnessing by a generally disinterested third party. Should the time of the concept become a matter for later proof, both the physical substance of the notebook and the established recording procedure serve as effective evidence in substantiating the fact that the concept existed at least as early as the notebook witness date.

The increasingly widespread use of electronic documents, which include not only digital representations of readable text but also of video, audio, and pictorial data, now poses a serious threat to the viability of the "notebook" concept of establishing the date of any such document. Because electronic digital documents are so easily revised, and since such revisions may be made without telltale sign, there is available limited credible evidence that a given document truly states the date on which it was created or the message it originally carried. For the same reasons there even arises serious doubt as to the authenticity of a verifying signature. Without an effective procedure for ensuring against the surreptitious revision of digital documents, a basic lack of system credibility prevents the efficiencies of electronic documentation from being more widely implemented.

Some procedures are presently available for verifying electronic document transmissions; however, such procedures are limited in application to bilateral communications. That is, in such communications the sender essentially desires to verify to the receiver the source and original content of the transmitted document. For example, "private key" cryptographic schemes have long been employed for message transmission between or among a limited universe of individuals who are known to one another and who alone know the decrypting key. Encryption of the message ensures against tampering, and the fact that application of the private key reveals the "plaintext" of the transmitted message serves as proof that the message was transmitted by one of the defined universe. The time of creation of the message is only collaterally established, however, as being not later than its receipt by the addressee. This practice thus fails to provide time-stamp evidence that would be useful in an unlimited universe at a later date.

A more broadly applicable verifying communication procedure, that of "public key" cryptography, has been described by Diffie and Hellman ("New Directions in Cryptography", *IEEE Transactions On Information Theory*, Vol. IT-22, November 1976, pp. 644-654) and more recently implemented by Rivest et al. in U.S. Pat.

No. 4,405,829, issued Sept. 20, 1983. While this scheme expands the utilizing universe to a substantially unlimited number of system subscribers who are unknown to one another, but for a public directory, verifiable communications remain bilateral. These limitations persist, since although a public key "signature", such as that which entails public key decryption of a message encrypted with the private key of the transmitter, provides any member of the unlimited universe with significant evidence of the identity of the transmitter of the message, only a given message recipient can be satisfied that the message existed at least as early as the time of its receipt. Such receipt does not, however, provide the whole universe with direct evidence of time of the message's existence. Testimony of such a recipient in conjunction with the received message could advance the proof of message content and time of its existence, but such evidence falls victim to the basic problem of ready manipulation of electronic digital document content, whether by originator or witness.

Thus, the prospect of a world in which all documents are in easily modifiable digital form threatens the very substance of existing procedures for establishing the credibility of such documents. There is clearly a significant present need for a system of verification by which a digital document may be so fixed in time and content that it can present, at least to the extent currently recognized in tangible documents, direct evidence on those issues.

SUMMARY OF THE INVENTION

The present invention yields such a reliable system in a method of time-stamping digital documents that provides the equivalent of two essential characteristics of accepted document verification. First, the content of a document and a time stamp of its existence are "indelibly" incorporated into the digital data of the document so that it is not possible to change any bit of the resulting time-stamped data without such a change being apparent. In this manner, the state of the document text is fixed at the instant of time-stamping. Second, the time at which the digital document is stamped is verified by a "witnessing" digital signature procedure that deters the incorporation of a false time statement. In essence, the method transfers control of the time-stamping step from the author to an independent agent and removes from the author the ability to influence the agent in the application of other than a truthful time stamp.

The method of the present invention presumes a number of document authors distributed throughout a communication network. Such authors may be individuals, companies, company departments, etc. each representing a distinct and identifiable, e.g. by ID number or the like, member of the author universe. In one embodiment of the invention, this universe may constitute the clientele of a time-stamping agency (TSA), while in another embodiment the distributed authors may serve as agents individually performing the time-stamping service for other members of the universe.

In its general application as depicted in FIG. 1 of the drawing, the present method entails an author's preparation of a digital document, which may broadly comprise any alphanumeric, audio, or pictorial presentation, and the transmission of the document, preferably in a condensed representative form, to the TSA. The TSA time-stamps the document by adding digital data signifying the current time, applying the agency's crypto-

graphic signature scheme to the document, and transmitting the resulting document, now a certificate of the temporal existence of the original document, back to the author where it is held for later use in required proof of such existence.

To ensure against interception of confidential document information during transmission, and to reduce the digital bandwidth required for transmission of the entire document, the author may optionally convert the digital document string to a unique number having vastly reduced digital size by means of a deterministic function which may, for example, be any one of a number of algorithms known in the art as "oneway hash functions". Such an application of hash functions has been described, among others, by Damgard in his discussions on the improvement of security in document signing techniques ("Collision-Free Hash Functions and Public Key Signature Schemes", *Advances in Cryptology—Eurocrypt* '87, Springer-Verlag, LNCS, 1988, Vol. 304, pp. 203–217). In practice of the present invention, however, the "one-way" characteristic typical of a hashing algorithm serves an additional purpose; that is, to provide assurance that the document cannot be revised subsequent to the time the TSA applies its time stamp.

A hashing function provides just such assurance, since at the time a document is hashed there is created a representative "fingerprint" of its original content from which it is virtually impossible to recover that document. Therefore, the time-stamped document is not susceptible to revision by any adversary of the author. Nor is the author able to apply an issued time-stamp certificate to a revised form of the document, since any change in the original content, even to the extent of a single word or a single bit of digital data, results in a different document that would hash to a completely different fingerprint number. Although the original document can thus not be recovered from the hashed document, a purported original document can nonetheless be proven by the fact that a true copy of the original document will always hash, assuming use of the same hashing algorithm, to the original number contained in the certificate.

Any available deterministic function, e.g. a one-way hash function such as that described by Rivest ("The MD4 Message Digest Algorithm", *Advances in Cryptology—Crypto*, '90, Springer-Verlag, LNCS, to appear), may be used in the present procedure. In the practice of the invention, such a hashing operation would normally be employed by the author to obtain the noted benefit of transmission security, although it might be effected by the TSA if the document were received in plaintext form. In whatever such manner the document content and incorporated time data are fixed against revision, there remains the further step, in order to promote the credibility of the system, of certifying to the members of an as yet unidentified universe that the receipt was in fact prepared by the TSA, rather than by the author, and that the time indication is correct, i.e. that it has not, for instance, been fraudulently stated by the TSA in collusion with the author.

To satisfy the former concern, the TSA uses a verifiable signature scheme, of a type such as the public key method earlier noted, to certify the time-stamp prior to its transmittal to the author. Confirmation of the signature at a later time, such as by decryption with the TSA's public key, proves to the author and to the universe at large that the certificate originated with the TSA. Proof of the veracity of the time-stamp itself, however, relies upon a following additional aspect of the invention.

One embodiment of this segment of the process, as generally depicted in FIG. 2, draws upon the relatively continuous flow of documents from the universe of authors through the facilities of the TSA. For each given processed document D_k, the TSA generates a time-stamp receipt which includes, for example, a sequential receipt number, r_k, the identity of the author, A_k, by ID number ID_k, or the like, the hash, H_k, of the document, and the current time, t_k. In addition, the TSA includes the receipt data of the immediately preceding processed document, D_{k-1}, of author, A_{k-1}, thereby bounding the timestamp of document, D_k, in the "past" direction by the independently established earlier receipt time, t_{k-1}. Likewise, the receipt data of the next received document, D_{k+1}, are included to bound the time-stamp of document, D_k, in the "future" direction. The composite receipt, now containing the time data of the three, or more if desired, sequential time-stamp receipts, or identifying segments thereof, is then certified with the cryptographic TSA signature and transmitted to the author, A_k. In like manner, a certificate containing identifiable representations of D_k and D_{k+2} would be transmitted to author, A_{k+1}. Thus, each of the time-stamp certificates issued by the TSA is fixed in the continuum of time and none can be falsely prepared by the TSA, since a comparison of a number of relevant distributed certificates would reveal the discrepancy in their sequence. So effective is such a sequential fixing of a document in the time stream that the TSA signature could be superfluous in actual practice.

A second embodiment of the invention, shown generally in FIG. 3, distributes the time-stamping task randomly among a broad universe, for example the multiplicity of authors utilizing the time-stamping process. A TSA could still be employed for administrative purposes or the requesting author could communicate directly with the selected time-stamping author/agents. In either event, the above-mentioned need for assurance that a time-stamp has not been applied to a document through collusion between the author and the stamping agency is met in the combination of the reasonable premise that at least some portion of the agency universe is incorruptible or would otherwise pose a threat of exposure to an author attempting falsification, and the fact that the time-stamping agencies for a given document are selected from the universe entirely at random. The resulting lack of a capability on the part of the author to select a prospective collusive agent of the author's own choosing substantially removes the feasibility of intentional time falsification.

The selection of the individual universe members who will act as the predetermined number of agents is accomplished by means of a pseudorandom generator of the type discussed by Impagliazzo, Levin, and Luby ("Pseudorandom Generation From One-Way Functions", *Proc. 21st STOC*, pp. 12–24, ACM, 1989) for which the initial seed is a deterministic function, such as a hash, of the document being time-stamped. Given as a seed input the document hash or other such function, the implemented pseudorandom generator will output a series of agency IDs. This agency selection is for all practical purposes unpredictable and random.

Once the agents are selected, the time-stamping proceeds as previously indicated with the exception that each agent individually adds the current time data to the

representative document it receives, certifies the resulting separate time-stamped receipt with its own verifiable cryptographic signature, and transmits the certificate back to the author. This transmittal may be directly to the requesting author or by way of the administrative TSA where the receipts are combined with or without further certification by the TSA. The combination of signature scheme and a published directory of author IDs provides verification of the utilization of the agents that were in fact selected by the pseudorandom generator. This distributed agent embodiment of the invention presents some advantages over the receiptlinking procedure in that a certified time-stamp is provided more quickly and a given author's later proof of a document is less reliant upon the availability of the certificates of other authors.

Additional variations in the process of the invention might include the accumulation of documents, preferably in hashed or other representative form, generated within an author organization over a period of time, e.g. a day or more depending upon the extent of activity, with the collection being hashed to present a single convenient document for time-stamping and certification. Also, the initial seed for the pseudorandom generator may be based upon a function of time or previously receipted documents, as well as of the document. The implementation of the process may be automated in simple computer programs which would directly carry out the described steps of hashing and transmitting original documents, selecting time-stamping agents, applying current time stamps, and returning certified receipts.

THE DRAWING

The present invention will be described with reference to the accompanying drawing of which:

FIG. 1 is a flow diagram of the general process of time-stamping a document according to the invention;

FIG. 2 is a flow diagram of a specific embodiment of the process; and

FIG. 3 is a flow diagram of another specific embodiment of the process.

DESCRIPTION OF THE INVENTION

The following examples of the application of embodiments of the present invention will serve to further describe the involved process. For convenience in the presentation of these examples, the deterministic function selected is the md4 hashing algorithm described by Rivest, as mentioned above, and the verifiable signature scheme is the public key method suggested by Diffie and Hellman, as implemented by Rivest et al. in U.S. Pat. No. 4,405,829. Further, in order to simplify explanation of the process and for the additional reasons noted below, only representative segments of the entire numbers will be employed.

The receipt-linking embodiment of the invention shown in FIG. 2 is initially considered. Although the present process may be used with documents of any length, the following apt excerpt is amply representative of a document, D_k, which an author prepares at step 21 and for which time-stamping is desired:

> Time's glory is to calm contending kings, To unmask falsehood, and bring truth to light, To stamp the seal of time in aged things, To wake the morn, and sentinel the night, To wrong the wronger till he render right;
>
> The Rape of Lucrece

By means of the md4 algorithm, the document is hashed, at optional, dashed step 22, to a number, H_k, of a standard 128 bit format which expressed in base 16 appears as:

ef6dfded833f3a43d4515a9fb5ce3915

The author, A_k, whose system identification number ID_k, is 172 in a 1000 member author universe, transmits the thus-identified document to the system TSA, at step 22, as the message, (ID_k,H_k), which appears:

172, ef6dfded833f3a43d4515a9fb5ce3915

as a request that the document be time-stamped.

The TSA then prepares the receipt for document, D_k, by adding, at step 25, a sequential receipt number, r_k, of 132, for example, and a statement of the current time, t_k. This time statement might include a standard 32 bit representation of computer clock time plus a literal statement, i.e. 16:37:41 Greenwich Mean Time on Mar. 10, 1990, in order to allow the final time-stamp certificate to be easily readable by the author, A_k. The receipt would then comprise the string, (r_k,t_k,ID_k,H_k).

At this point it would be appropriate to further consider the earlier-mentioned reduction of number size to representative segments. As is described by Rivest et al. in U.S. Pat. No. 4,405,829, the cryptographic public key scheme to be employed in this example (generally known in the field as the "RSA" signature scheme) requires the division of an extended message into blocks that may each be represented by a number not exceeding the encoding key number element, n. Each such block is then signed with the RSA algorithm, to be reassembled after transmission. Therefore, in order to be able to use a number, n, of reasonable size in this example while maintaining a single block for the final receipt string to be certified with the RSA scheme, each element of the receipt string will be reduced to a representative eight bits, typically the last eight bits of any overlong string, and those bits will be stated in base 16 to present a two hexadecimal character string. Thus, for instance, the 128 bit document hash, H_k, will be represented by its last eight bits, i.e. 0001 0101, stated as 15 (base 16). Likewise, ID_k, 172, is 1010 1100 and is represented by ac (base 16). Without actually undertaking the calculation, it will suffice to assume that the time statement, t_k, is represented as 51. The receipt number, 132, would be represented as 84. The receipt string to this point, i.e. (r_k,t_k,ID_k,H_k) now appears as 8451ac15.

Assume now that the immediately preceding document, D_{k-1}, was processed by the TSA as the request:

201, d2d67232a61d616f7b87dc146c575174

at 16:32:30 on Mar. 10, 1990 (t_{k-1} being represented as 64). The TSA adds these data at step 27, to the receipt string for D_k to yield the hexadecimal representation, 8451ac1564c974. This receipt R_k, now contains data fixing the time for D_k and a time, t_{k-1}, before which author, A_k, cannot claim that D_k existed. This limitation on A_k is established by the fact that the previous author, A_{k-1}, holds a time certificate, C_{k-1}, that fixes t_{k-1} as subsequent to the linked time data, t_{k-2}, in the certificate of author, A_{k-2}, and so on for as long as a proof requires.

To establish that TSA in fact originated the receipt for document, D_k, that receipt is transmitted, at step 29,

to author, A_k, after TSA signing, at step **28**, with the public key cryptographic signature scheme and becomes the certified receipt, or certificate, C_k. With the data derived above, and assuming that TSA has the RSA signature key set, in decimal:

$$<n, e> = <432006778214281O9, 191> \text{ (Public)}$$

$$<n, d> = <432006778214281O9, 29403602422449791> \text{ (Private)}$$

the signed certificate for R_k, 8451ac1564c974, would compute as:

$$R^e \bmod n = 39894704664774392$$

When author, A_k, receives this certificate, C_k, along with the literal statement of R_k, it may be readily confirmed as being correct by application of the TSA public key to verify that:

$$C_k^e \bmod n = R_k$$

and that R_k in fact contains the data representing the document hash, H_k.

The procedure shown in this simple one-link example results in a certificate which, being bounded in time by the data from document, D_k, provides author, A_{k-1} with reliable evidence that document, D_{k-1}, was not backdated to a time significantly prior to the existence of document, D_k. When the certificate of A_k is expanded with additional data from the subsequently processed document, D_{k+1}, it will likewise be effectively bounded to substantiate the time stamp claimed by A_k. In an alternative of the same effect, A_k could simply be advised of the identity of A_{k+1} and could confirm from that author that the one-link certificate, C_{k+1}, contained the element, H_k. The procedure could also be varied to provide certified receipts which include data from any number of authors, with each addition providing a further degree of assurance against falsification.

Another embodiment of the invention, as shown in FIG. 3, which utilizes randomly selected members of the author universe as time-stamping agents, or witnesses, i.e. a "distributed trust" procedure, would proceed in the following manner. Although these numbers are not so limited in actual practice, for purposes of the example it will be assumed that the universe consists of 1000 authors, having IDs 0–999, and that three witnesses will be sufficient to establish the veracity of the time stamp. Also, in this example the earlier-noted variation including the services of a TSA is being implemented. The hashing function, md4, utilized in the above example is employed here also, in optional step **32**, as an example of a deterministic document function which will seed the pseudorandom selection of the three witnesses from the author universe.

As in the previous example, the author transmits the document to the TSA, normally in hashed form, as the identified request:

(72, ef6dfded833f3a43d4515a9fb5ce3915

The TSA now uses this document hash string, in step **33**, as the seed to generate the ID number of the first witness, at step **35**, according to the selection algorithm:

ID=[md4(seed)]mod(universe size)

The resulting seed hash:

26f54eae92511dbb5e06e7c2de6e0fcf

represents the 128 bit number which mod **1000** is 487, the ID of the first selected witness. The next witness is likewise chosen using this seed hash representation as the seed in the second selection computation to yield:

882653ee04d16b1f0d604883aa273006

which mod **1000**, is 571, the second witness ID. A repeat of the computation, again seeding with the prior seed hash, selects the final witness as **598**, which is:

2fe8768ef3532f15c40acf1341902cle mod 1000

The TSA now sends, at step **37**, a copy of the original request to each of these three witnesses who individually, at step **38**, add a current time statement and ID, and certify the resulting receipts by signing with the RSA cryptographic signature scheme and transmitting them, at step **39**, directly to the author or through the TSA who may assemble the certificates into a file to be delivered to the author. By virtue of the fact that the pseudorandom generation prevents the exercise of a personal choice in the selection of witnesses, the author is deterred by the risk of encountering a non-cooperative witness from attempting any communication prior to time stamp certification for the purpose of arranging for a false time entry. In a process variant where the author is allowed to transmit the request directly to witnesses, the random selection of such witnesses which is keyed essentially to the involved document itself frustrates any attempt by the author to direct the document to a known cooperative witness. The group of resulting certificates may thus be employed with confidence in later proofs employing signature verification in the manner earlier described.

The procedures described and variants suggested herein for the practice of this time-stamping process and the various other embodiments which will become apparent to the skilled artisan in the light of the foregoing description are all nonetheless to be included within the scope of the present invention as defined by the appended claims.

What is claimed is:

1. A method of time-stamping a digital document which comprises:

a) transmitting a digital representation of said document from an originator to an outside agency;

b) creating at said outside agency a receipt comprising a digital representation of then current time and at least a portion of a digital representation of said digital document; and

c) certifying said receipt at said outside agency by means of a verifiable digital cryptographic signature scheme.

2. A method of time-stamping a digital document according to claim **1** wherein said transmitted digital document representation comprises at least a portion of the digital representation of the number derived by application of a deterministic function algorithm to said digital document.

3. A method of time-stamping a digital document according to claim **1** wherein said receipted digital document representation comprises at least a portion of the digital representation of the number derived by

application of a deterministic function algorithm to said digital document.

4. A method of time-stamping a digital document according to claim 3 wherein said digital number representation is derived from the application of a one-way hashing algorithm to said digital document.

5. A method of time-stamping a digital document according to claim 1 wherein said receipt further comprises the time representation and digital document representation specific to at least one other digital document receipted by said outside agency.

6. A method of time-stamping a digital document according to claim 5 wherein the receipt of said at least one other digital document was created by said outside agency earlier than that of the currently receipted digital document.

7. A method of time-stamping a digital document according to claim 5 wherein the receipt of said at least one other digital document was created by said outside agency later than that of the currently receipted digital document.

8. A method of time-stamping a digital document according to claim 1 wherein said outside agency is selected at random from a predetermined universe.

9. A method of time-stamping a digital document according to claim 8 wherein said outside agency is selected by means of a pseudorandom generator seeded with at least a portion of the digital representation of the number derived by application of a deterministic function algorithm to said digital document.

10. A method of time-stamping a digital document according to claim 9 wherein said pseudorandom generation seed is derived from the application of a one-way hashing algorithm to said digital document.

11. A method of time-stamping a digital document according to claim 10 which further comprises the like preparation of a time-stamp certificate by at least one additional outside agency selected by said pseudorandom generation and wherein the input for each additional outside agency selection is at least a portion of the digital representation of the output derived from the application of said one-way hashing algorithm to a digital representation of the previously generated output.

12. A method of time-stamping a digital document according to claim 9 which further comprises the like preparation of a time-stamp certificate by at least one additional outside agency selected by said pseudorandom generation.

13. A method for the secure time-stamping of a digital document

characterized in that

a) a digital representation of said document is transmitted from an originator to an outside agency;

b) said outside agency creates a receipt comprising a digital representation of then current time and at least a portion of a digital representation of said digital document; and

c) said receipt is certified at said outside agency by means of a verifiable digital cryptographic signature scheme.

14. A method for the secure time-stamping of a digital document according to claim 13

characterized in that said receipt further comprises the time representation and digital document representation specific to at least one other digital document receipted by said outside agency.

15. A method for the secure time-stamping of a digital document according to claim 14

characterized in that the receipt of said at least one other digital document was created by said outside agency later than that of the currently receipted digital document.

16. A method for the secure time-stamping of a digital document according to claim 13

characterized in that said outside agency is selected at random from a predetermined universe by means of a pseudorandom generator seeded with at least a portion of the digital representation of the number derived from the application of a deterministic function algorithm to said digital document.

17. A method for the secure time-stamping of a digital document according to claim 16

characterized in that said seed is derived from the application of a one-way hashing algorithm to said digital document.

18. A method for the secure time-stamping of a digital document according to claim 16

characterized in that a time-stamp certificate for said digital document is likewise prepared by at least one additional outside agency selected by means of said pseudorandom generation.

* * * * *

US005373561A

United States Patent [19]

Haber et al.

[11] Patent Number: 5,373,561

[45] Date of Patent: Dec. 13, 1994

[54] **METHOD OF EXTENDING THE VALIDITY OF A CRYPTOGRAPHIC CERTIFICATE**

[75] Inventors: **Stuart A. Haber**, New York, N.Y.; **Wakefield S. Stornetta, Jr.**, Morristown, N.J.

[73] Assignee: **Bell Communications Research, Inc.**, Livingston, N.J.

[21] Appl. No.: **992,883**

[22] Filed: **Dec. 21, 1992**

[51] **Int. Cl.**[5] **H04L 9/00**; H04L 9/30

[52] **U.S. Cl.** **380/49**; 380/23; 380/25; 380/30

[58] **Field of Search** 380/3–5, 380/9, 10, 23, 24, 25, 28, 30, 49, 50

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,405,829	9/1983	Rivest et al.	380/30
4,625,076	11/1986	Okamoto et al.	380/30 X
4,868,877	9/1989	Fischer	380/25
4,881,264	11/1989	Merkle	380/25
4,972,474	11/1990	Sabin	380/28
5,001,752	3/1991	Fischer	380/23
5,136,646	8/1992	Haber et al.	380/49
5,136,647	8/1992	Haber et al.	380/49

OTHER PUBLICATIONS

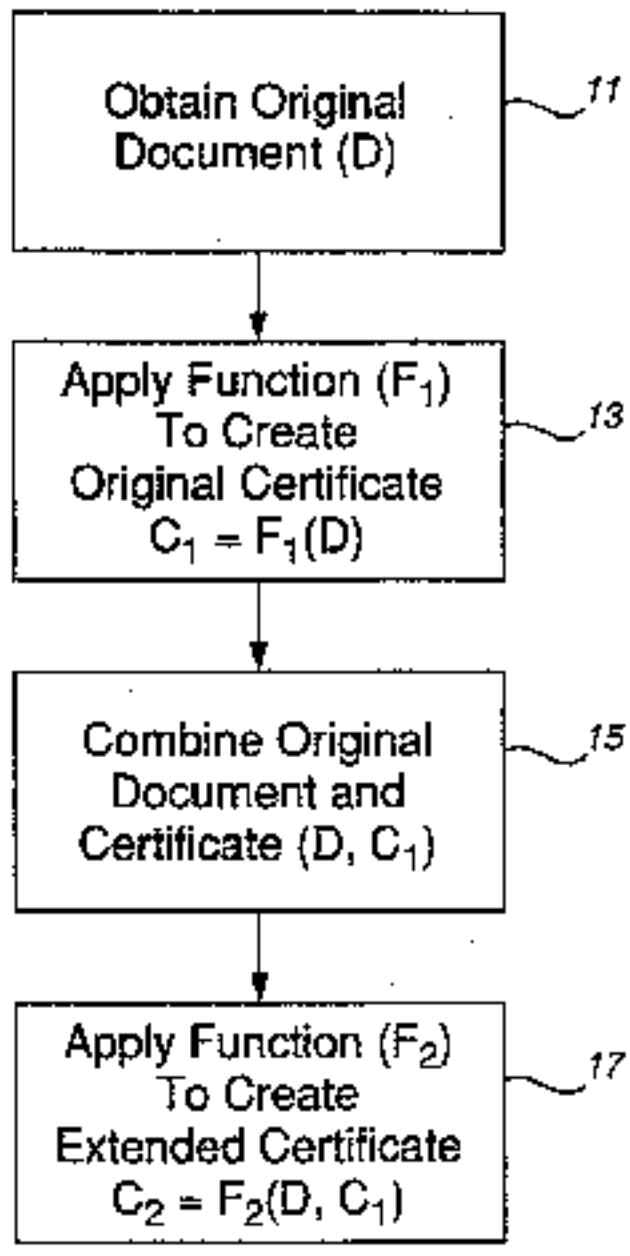
Cryptography and Data Security, D. E. R. Denning, pp. 170–171, Addison–Wesley Publishing Company (1982).

Primary Examiner—Bernarr E. Gregory
Attorney, Agent, or Firm—Leonard Charles Suchyta; Lionel N. White

[57] **ABSTRACT**

A cryptographic certificate attesting to the authenticity of original document elements, such as time of creation, content, or source, will lose its value when the cryptographic function underlying the certifying scheme is compromised. The present invention provides a means for extending the reliability of such a certificate by subjecting, prior to any such compromise, a combination of the original certificate and the document digital representation from which that certificate was derived to a scheme based on a different and ostensibly less vulnerable function. The new certificate resulting from this procedure extends the validity of the original authenticity by implacably incorporating the original certificate at a time when that certificate could only have been derived by legitimate means.

20 Claims, 2 Drawing Sheets



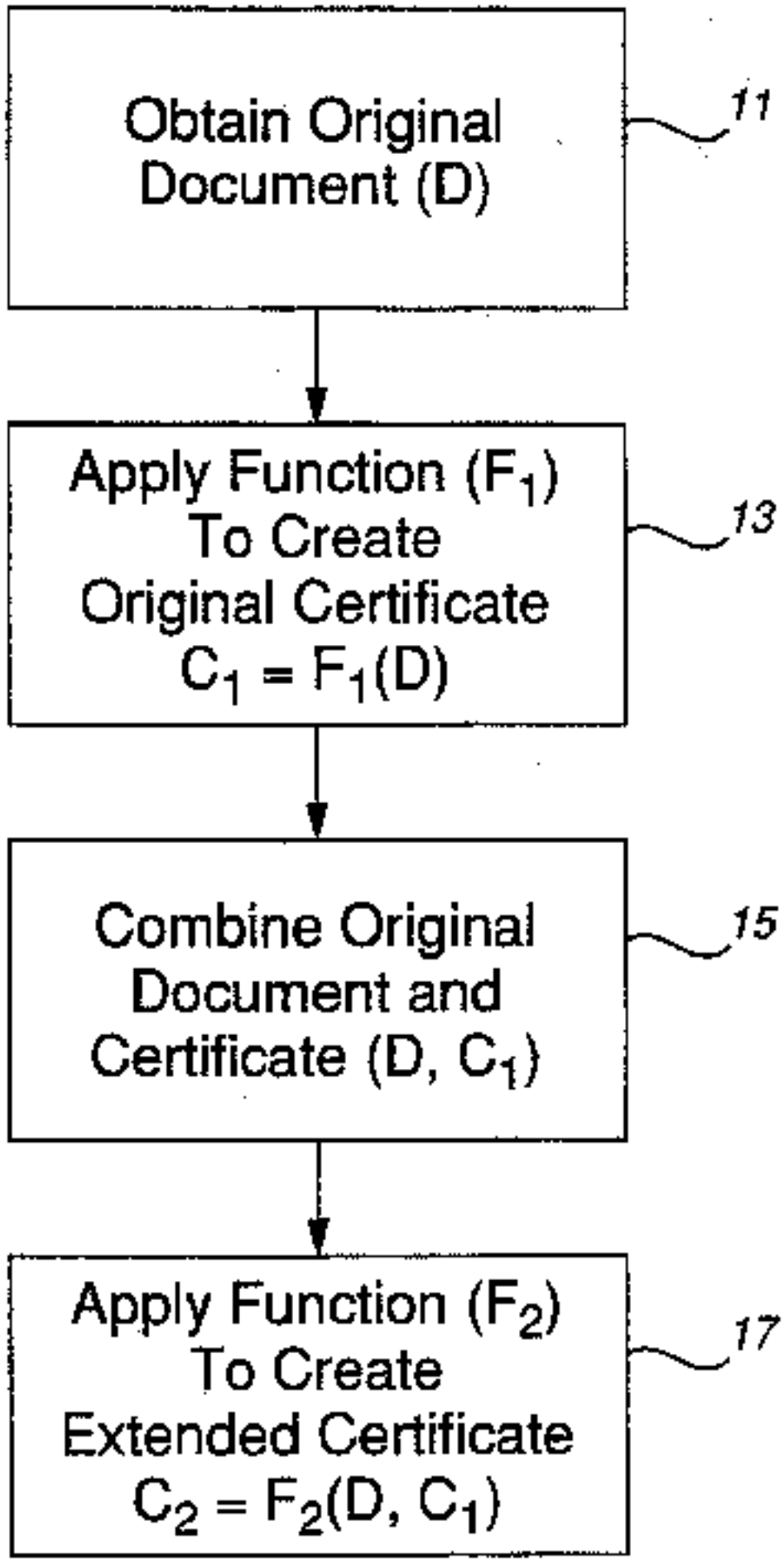

Obtain Original Document (D)
11
Apply Function (F_1) To Create Original Certificate $C_1 = F_1(D)$
13
Combine Original Document and Certificate (D, C_1)
15
Apply Function (F_2) To Create Extended Certificate $C_2 = F_2(D, C_1)$
17

FIG. 1

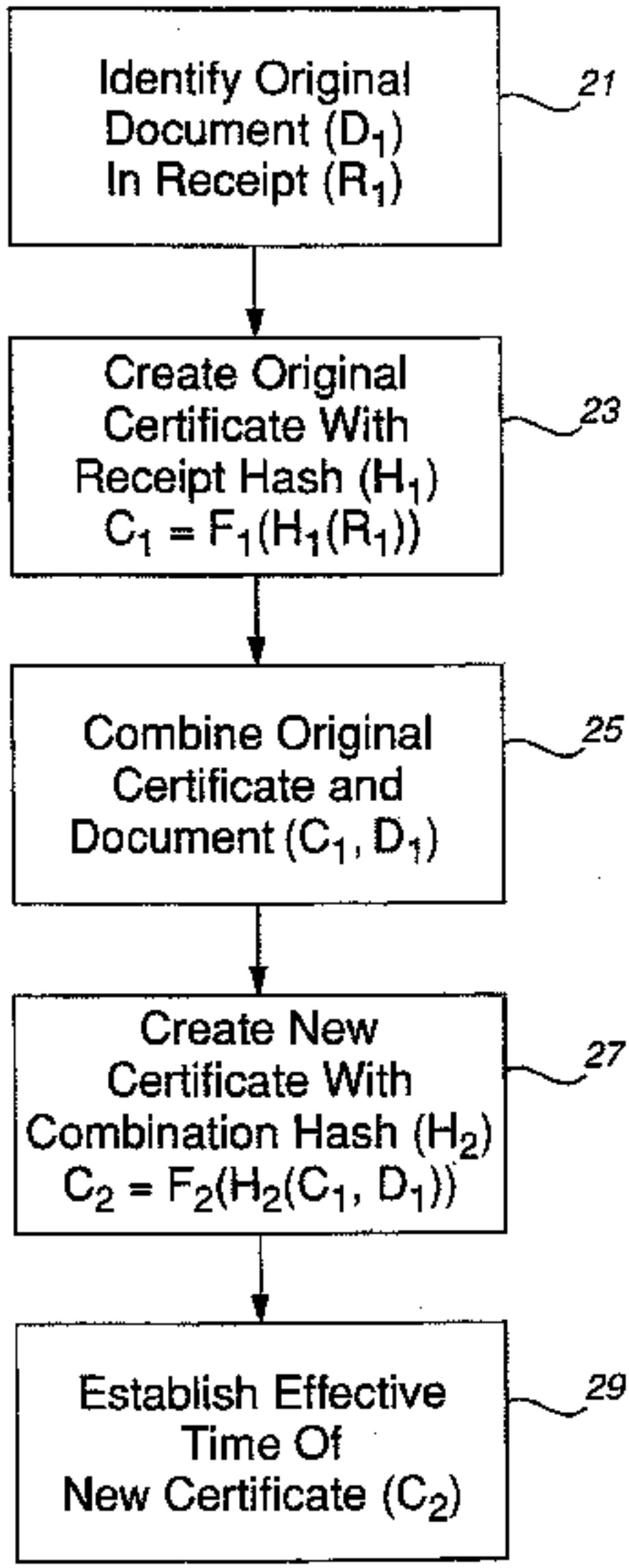

Identify Original Document (D_1) In Receipt (R_1)
21
Create Original Certificate With Receipt Hash (H_1) $C_1 = F_1(H_1(R_1))$
23
Combine Original Certificate and Document (C_1, D_1)
25
Create New Certificate With Combination Hash (H_2) $C_2 = F_2(H_2(C_1, D_1))$
27
Establish Effective Time Of New Certificate (C_2)
29

FIG. 2

METHOD OF EXTENDING THE VALIDITY OF A CRYPTOGRAPHIC CERTIFICATE

BACKGROUND OF THE INVENTION

This invention relates to methods for certifying or validating the existence or occurrence of a recorded document or event, particularly methods which rely upon cryptographic assumptions to establish the basis for such a certification or validation. More specifically, the invention relates to a method for reconfirming an original certificate in order to maintain its validity for a significant period of time beyond the probable compromise of an underlying cryptographic assumption or step in the original certification procedure.

Time-stamping procedures described in U.S. Pat. Nos. 5,136,646 and 5,136,647 are representative of a type of certification for which the present method is adapted. Such schemes for setting a reliable time of creation of a document, or providing indisputable evidence against the alteration of a document, generally digital computer data in alphanumeric, pictorial, video, or audio form, depend upon the assumption that there exist cryptographic functions which, when applied to a digital representation of such a document, defy any manner of manipulation which might permit undetectable alterations or falsifications of the original state of document elements. The functional procedures generally exemplified in those disclosures typically provide this required property, since they generate unique certificate statements which essentially can not be duplicated other than from an identical document representation. This security arises from the fact that the derivation or reconstruction of these functions from the products of their application is computationally infeasible. Ultimate achievement of such derivations must be anticipated, however, since a given function or procedure may be fatally flawed or, as is becoming more probable, advancements in computer technology and algorithmic techniques are likely to make more readily available a level of calculating power which enables such derivation.

With compromise of a step or algorithm in a procedural certification function, the possibility arises of generating duplicate certificates or parts thereof from different digital representations, i.e., creating "collisions", and thereby defeating the previously reliable basis for a certification scheme. Substitution of a newer and presumably less vulnerable function in the certification procedure may prevent for some finite time the compromise of future certificates, but the value of past certificates in establishing original creation dates, for example, is all but lost. The present invention, however, provides a means for bridging the technological gap and extending into the era of a newer function or procedure the validity of the original certification.

SUMMARY OF THE INVENTION

Historically, there has usually been an overlap period between the time spans of reliability of an established cryptographic function and one which has been newly implemented with improved resistance to compromise. As computational power increases and algorithmic techniques improve, the evolution and phasing of cryptographic certification procedures or functions, for example, can generally be foreseen, It is possible, therefore, to anticipate the final stages of reliability provided by an existing certification scheme and to initiate a

procedure, such as provided by the present invention, to ensure the continuity of original certificate validity.

In essence, this invention entails generating from the original document a new document certificate during the viable term of the original certification scheme, such as may be based upon a cryptographic signature key procedure or a time-stamping procedure. This new certification process comprises applying a different cryptographic function, e.g., a time-stamping procedure, to a combination including the original certificate and the original digital document from which the certificate was derived. Such a different function is preferably a new and presumably more reliable algorithm or procedure, or at least one upon which the original certification did not rely. The resulting certificate, generated by means of a function or procedure having a significant expected remaining term of reliability, now implacably embodies the original certificate elements at a time prior to any likely compromise of the original certification function. Since these original elements have as yet been exposed to no threat of compromise and are now bound by the new time stamp within the protective cloak of a far more relatively invulnerable certification function, their original veracity has been extended for at least the reliable term of this new function.

BRIEF DESCRIPTION OF THE DRAWING

The present invention will be described with reference to the accompanying drawing of which:

FIG. 1 presents a flow chart of steps embodying a general procedure implementing the certificate extension process of the invention; and

FIG. 2 presents a flow chart of steps embodying a rudimentary time-stamping procedure implementing the certificate extension process of the invention.

DESCRIPTION OF THE INVENTION

The extension procedure of the present invention is applicable to any manner of certificate digitally derived by cryptographic means. For instance, the process may be used to support the veracity of a document transmittal originally certified with a cryptographic key signature algorithm or function beyond a time when that function might be compromised, whether due to misappropriation of a secret key or to advances in computer technology and algorithmic techniques. A digital time-stamp certificate could similarly benefit by application of the invention to prevent its coming into question after compromise of the scheme or function underlying the time-stamping procedure. In general, the process of the invention is useful to ensure the continued viability of any certificate produced by a digital scheme or function which is capable of compromise.

The steps comprising a basic application of the certificate extension process are shown in FIG. 1. There, initial steps 11, 13 are intended to depict any certification procedure, such as a signature scheme or time-stamping process, in which a digital document, D_1, e.g., a body of text or alphanumeric representations, a picture, an audio recording, or the like, is subjected to a cryptographic scheme or procedure, generally a "function", F_1, to produce a certificate, C_1, which will serve later as evidence of the original existence and substance of D_1. The value of certificate, C_1, will persist, however, only until a compromise of the certification function, as a whole or in a component step or algorithm, since, as a result of such a compromise, the certificate

might thereafter be duplicated by an imposter or through the use of a counterfeit document.

The basic steps of the invention are therefore effected prior to any such compromise, as projected, for example, on the basis of the current state of computational technology, and comprise combining, at **15**, the original document, D, with the original certificate, C_1, and applying to that combination, at **17**, a different and presumably more secure scheme or function to obtain a new certificate, C_2, which will later attest to the validity of original certificate, C_1, at a time when its generating function, F_1, was as yet uncompromised and secure. The essential element of this process resides in the application of the new certification function to the conjunction of original document, D, with original certificate, C_1. This step avoids the error inherent in the naive and ineffectual procedure of merely recertifying either the original certificate or the original document alone; namely, that of perpetuating a compromise which reflects directly upon the veracity of the original document, D.

As an example, one might consider application of the present invention to extend the valid lifetime of a digitally signed document where, in keeping with usual practices, a digital signature, σ, is derived by application of some cryptographic signature scheme to a document, D. To avoid invalidation of such a signed document by subsequent compromise of the scheme, for instance, due to misappropriation of a user's private key, the pre-compromise generation of a certificate, C, by application of a time-stamp function, T, to a combination of the signature and the document:

$$C=T(\sigma,D)$$

will provide continuing proof that the signature was created prior to the compromise, i.e., at a time when only a legitimate user could have produced it. Such a certificate might also be used to establish original authorship of the document.

The invention is broadly useful, as well, as a means of extending or "renewing" time-stamp certificates, generally. For example, a simple scheme for certifying an event, such as time-stamping the creation of a document, comprises establishing a digital representation of the document content, adding data denoting current time, and permanently fixing the resulting digital statement against subsequent revision, all under trustworthy circumstances, to yield a certificate which will provide irrefutable evidence of the event at a later time. Means for ensuring the original veracity of the certificate have been described in our earlier-noted patent specifications as including use of trusted outside agencies, arbitrary selection of agencies, linking of certificates in temporal chains, and similar practices which remove substantially all influence a document author might have upon the certification process. Other methods of establishing the authenticity of original certification procedures might also include private and public key cryptographic communications.

Common to certification procedures is the application of some manner of cryptographic function by which the document, related identifying data, or digital representations of these elements may be algorithmically reduced to a unique statement or cipher which can not feasibly be duplicated from different representative elements by computational means. Any of the general class of one-way hashing algorithms, for example, may be used in such a procedure or function applied to a digital representation of a time-receipted document to produce an inimitable certificate, usually in the form of a cryptic string of alphanumeric characters, which can only be generated by such an application of that same function to exactly that digital representation. The additional characteristic property of the one-way function is that of possessing such mathematical complexity as to discourage the computational derivation or reconstruction of the original digital representation from the resultant certificate, as well as to discourage the generation of a matching certificate from a different representation.

A simple certification procedure utilizing such a one-way hashing algorithm is represented in FIG. **2** at steps **21–23**. There, digital document, D_1, of step **21** is identified, e.g., annotated with author data, to yield a receipt, R_1, that, in a rudimentary procedure which may be simply stated as:

$$C_1=F_1(H_1(R_1))$$

is in turn reduced at step **23** to a certificate, C_1, by application of a time-stamping function, F_1, comprising a current hash algorithm, H_1.

As a result of computational or algorithmic developments over time, or in the event of a flaw in the function itself, hash, H_1, may become compromised with the result that a falsified receipt, R_x, could produce a duplicate, or "collision", certificate, C_1. The veracity of original certificate, C_1, and its value as probative evidence of the contents of document, D, and other elements of receipt, R_1, would thus be destroyed, since there would no longer exist a singular certificate cipher that could be traced solely to the original document and its once-unique receipt, R_1.

Advent of the collision need not denigrate the worth of the initial certificate back to the time of its creation, however, but only for the period subsequent to the compromise. The value of the certificate during its earlier term could be preserved and extended into the future if means were available to link into a time prior to such compromise with a trustworthy scheme for deriving a new certificate at least as unique and intractable as was the initial certificate. The problem, therefore, has been to "recertify" the original certificate in a manner which would verify the facts that had been securely bound into that certificate until the first collision occurred.

A naive solution to this problem would appear to be just that simple; that is, to recertify the original certificate, for example by applying a new and more robust hash, H_2. The fallacy in this approach becomes apparent, however, when one considers that after the instance of a collision the condition exists where:

$$H_1(R_1)=C_1=H_1(R_x).$$

The hashing of certificate, C_1, with a new function, H_2, would therefore not produce a renewal certificate cipher, C_2, unique only to receipt, R_1, since:

$$C_2=H_2(C_1)=H_2(H_1(R_1))=H_2(H_1(R_x))$$

and, thus, there is no reliable distinction between those resulting certificates.

The present invention, however, does provide such a unique certificate which serves to extend the veracity of an original certificate beyond subsequent compromise

of the original function or algorithm. This is accomplished, as in the representative of FIG. 2, by combining, at step 25, the original certificate, C_1, with the original document, D_1, from which it was generated and which is to be later proven, and applying to that composite statement, at step 27, a different certification function, F_2, e.g., comprising a new hashing algorithm, H_2, to yield the extended certificate:

$$C_2=F_2(H_2(C_1,D_1))=F_2(H_2(H_1(R_1),D_1)).$$

The final represented step, 29, in which it is established that the new certificate, C_2, was created during the valid term of original certificate, C_1, i.e., prior to any compromise of the original certification function, may be effected along with step 27, for example in the course of applying an earlier-described time-stamping procedures, to generate certificate, C_2. Alternatively, the effective time of the new certificate, C_2, may be established simply by publication, e.g., in a widely-distributed newspaper, either alone or as incorporated into a derivative representation similar to the "authentication tree" noted by D. E. R. Denning in *Cryptography and Data Security*, pp. 170–171, Addison-Wesley (1982).

In the ultimate utilization of this new certificate, C_2, to prove the original document, D_1, by recomputing certificate, C_2, from its elements, such proof will fail unless original document, D_1, rather than a bogus document, D_x, is an included element. Even though a collision due to compromised function, H_1, may exist at the time of using certificate, C_2, in a proof, the as yet invulnerable state of hash function, H_2, ensures against any collision with the expanded statement, i.e., one comprising document element, D_1, which is used to generate that new certificate. During a normal proofing process, the original certificate, C_1, will also be recomputed using the document in question. Unless the document then employed to recompute original certificate, C_1, matches precisely the document similarly employed with new certificate, C_2, the proof will not be realized. A false document, D_x, therefore can not be substituted surreptitiously for an original document as long as the applied hash function, H_2, remains uncompromised, since for any document, D_x, which one could feasibly compute:

$$H_2(C_1,D_1)\neq H_2(C_1,D_x).$$

When advancements in computation portend a compromise situation, yet a different time-stamp function, e.g., one utilizing algorithm, H_3, with longer life expectancy may be employed in the same procedure to again extend the involved certificate.

As an example of the implementation of the present invention, one might consider first an initial certificate prepared in the manner described in our earlier U.S. Pat. No. 5,136,646 employing the one-way hash algorithm specified by R. L. Rivest in "The MD4 Message Digest Algorithm", Advances in Cryptology—Crypro '90, Lecture Notes in Computer Science, Vol. 537 (ed. A. J. Menezes et al.), pp. 303–311, Springer-Verlag (Berlin, 1991). In that earlier example, elements of the receipt, R_1, identifying the quotation "document" appeared as:

1328, 194628GMT06MAR91, 634,

ec2cf3ca60ef10cb621c4f63f8dc34c7

and with additional data representing a prior transaction formed the basic statement to which the function comprising MD4 hash algorithm, H_1, was applied to yield the unique cipher:

46f7d75f0fbea95e96fc38472aa28ca1

which is held by the author as a time-stamp certificate, C_1.

In the event of an anticipated compromise of the MD4 hash function algorithm, the procedure of this invention would be initiated utilizing a different time-stamping certification function comprising, for example, a new algorithm, H_2, such as the MD5 hashing function described by Rivest and Dusse, "The MD5 Message Digest Algorithm", Network Working Group, Internet Draft, RSA Data Security, Inc. (July 1991); RFC 1321, Internet Activities Board (April 1992).

As an initial step in this procedure, the document representation, D_1, to be proven at a later time is combined with original certificate, C_1, either in original digital form or, preferably, as the convenient, condensed output of hash function, H_2, viz.:

.D9776652kDAj2.M5191CAD7

thus forming the combination statement, (C_1, D_1), as:

46f7d75f0fbea95e96fc38472aa28ca1,

.D9776652kDAj2.M5191CAD7.

Applying to this statement hashing algorithm, H_2, comprising the new function, F_2, produces:

656h//FDDM60M9/qDDt85F56

which in a time-stamping procedure, for instance, may be transmitted to an outside agency for the inclusion of current time data and authenticating cryptographic signature to yield extended certificate, C_2. As earlier noted, the effective date of a new certificate, C_2, may otherwise be established, such as in other time-stamping schemes or by public display or notoriety.

A variation on the foregoing embodiment provides an even more reliable practice in that it substantially eliminates the uncertainties associated with estimating the onset of a certification function compromise. This is accomplished by using a plurality of different cryptographic functions, e.g., F_a and F_b, to derive a compound original certificate, C_α:

$$C_\alpha=F_a(D_1),F_b(D_1).$$

which will remain valid even after the confirmed compromise of one of those function due to the likely continued viability of the other. Thus a period of security continues during which one may select a new certification function, F_α to be employed in the extension of certificate, C_α as:

$$C_\beta=F_b(C_\alpha, D_1), F_c(C_\alpha, D_1).$$

Subsequent compromise of any current cryptographic function can be remedied in like manner.

It is anticipated that other variants will become apparent to the skilled artisan in the light of the foregoing disclosure, and such embodiments are likewise consid-

ered to be encompassed within the scope of the invention defined by the appended claims.

What is claimed is:

1. A method of extending the validity of a first cryptographic certificate derived by applying a first cryptographic function to a digital document, which method comprises:

a) combining a digital representation of said document with a digital representation of said certificate; and

b) applying to the resulting combination during the valid term of said first certificate a different cryptographic function to thereby generate a second certificate attesting to the then current validity of said first certificate.

2. A method according to claim 1 wherein said first function is a cryptographic signature scheme.

3. A method according to claim 2 wherein said different function is a time-stamping procedure.

4. A method according to claim 3 wherein said different function comprises a one-way hashing algorithm.

5. A method according to claim 1 wherein said first function is a time-stamping procedure.

6. A method according to claim 5 wherein said first function comprises a one-way hashing algorithm.

7. A method according to claim 5 wherein said different function is a time-stamping procedure.

8. A method according to claim 7 wherein said first function comprises a first one-way hashing algorithm and said different function comprises a different one-way hashing algorithm.

9. A method according to claim 1 wherein said different function is a time-stamping procedure.

10. A method of certifying a digital representation of a document which comprises:

a) generating a first certificate by applying to said digital representation at least a first cryptographic function;

b) combining said first certificate with said digital representation; and

c) generating a second certificate by applying to said combination at least one cryptographic function which is different from said first function.

11. A method according to claim 10 wherein said first function is a cryptographic signature scheme.

12. A method according to claim 11 wherein said different function is a time-stamping procedure.

13. A method according to claim 12 wherein said different function comprises a one-way hashing algorithm.

14. A method according to claim 10 wherein said first function is a time-stamping procedure.

15. A method according to claim 14 wherein said first function comprises a one-way hashing algorithm.

16. A method according to claim 14 wherein said different function is a time-stamping procedure.

17. A method according to claim 16 wherein said first function comprises a first one-way hashing algorithm and said different function comprises a different one-way hashing algorithm.

18. A method according to claim 10 wherein:

a) said first certificate is generated by applying to said digital representation at least first and second different cryptographic functions; and

b) said second certificate is generated by applying to said combination at least one cryptographic function which is different from said first and second functions.

19. A certificate authenticating a digital representation of a document, said certificate consisting of a second certificate generated according to the method of claim 10.

20. A certificate according to claim 19 wherein:

a) said first certificate is generated by applying to said digital representation at least first and second different cryptographic functions; and

b) said second certificate is generated by applying to said combination at least one cryptographic function which is different from said first and second functions.

* * * * *

22. A method for time stamping a particular digital document in a relatively continuous flow of digital documents comprising the steps of

(a) forming a receipt comprising at least a portion of a digital representation of said particular digital document,

(b) forming at least one different receipt comprising at least a portion of a digital representation of at least one different digital document in said relatively continuous flow of digital documents, and

(c) utilizing said receipts to form a composite receipt which is thereby fixed in the continuum of time.

23. A method for time stamping a particular digital document comprising the steps of

(a) at an originator, forming a particular digital value representative of said particular digital document by applying a deterministic function to said particular digital document,

(b) transmitting said particular digital value to a time stamping agency,

(c) at said time stamping agency, forming a collection of digital values, including said particular digital value, and forming a single digital value from the digital values in said collection, and

(d) creating a time stamp receipt for said particular digital document by associating with said single digital value a representation of then current time.

24. The method of claim 23 wherein said particular digital value representative of said particular digital document is a particular hash value, said deterministic function is a hash function, said collection of digital values including said particular digital value is a collection of hash values including said particular hash value, and said single digital value is a single hash value representative of said collection of hash values.

25. A method for time stamping a particular digital document comprising the steps of

(a) at an originator, forming a particular digital representation of said particular digital document,

(b) transmitting said particular digital representation to a time stamping agency,

(c) at said time stamping agency, forming a receipt comprising at least a portion of said particular digital representation of said particular digital document,

(d) at said time stamping agency, forming at least one different receipt comprising at least a portion of a digital representation of at least one different digital document, and

(e) forming from said receipts a composite receipt which is thereby fixed in time.

* * * * *

US00RE34954E

United States Patent [19]

Haber et al.

[11] E **Patent Number: Re. 34,954**

[45] **Reissued Date of Patent: May 30, 1995**

[54] **METHOD FOR SECURE TIME-STAMPING OF DIGITAL DOCUMENTS**

[75] Inventors: **Stuart A. Haber**, New York, N.Y.; **Wakefield S. Stornetta, Jr.**, Morristown, N.J.

[73] Assignee: **Bell Communications Research, Inc.**, Livingston, N.J.

[21] Appl. No.: **156,120**

[22] Filed: **Nov. 22, 1993**

Related U.S. Patent Documents

Reissue of:

[64] Patent No.: **5,136,647**
Issued: **Aug. 4, 1992**
Appl. No.: **561,888**
Filed: **Aug. 2, 1990**

[51] **Int. Cl.**[6] **H04L 9/00**; H04L 9/30

[52] **U.S. Cl.** **380/49**; 380/23; 380/25; 380/30

[58] **Field of Search** 380/3–5, 380/9, 10, 28, 30, 49, 50

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,145,568	3/1979	Ehrat	380/50 X
4,405,829	9/1983	Rivest et al.	380/30
4,625,076	11/1986	Okamoto et al.	380/23
4,868,877	9/1989	Fischer	380/25
4,881,264	11/1989	Merkle	380/25
4,972,474	11/1990	Sabin	380/49 X
5,001,752	3/1991	Fischer	380/23

OTHER PUBLICATIONS

"New Directions in Cryptography", W. Diffie & M. E. Hellman, IEEE Trans. on Info. Theory, (vol. IT–22, Nov. 1976, pp. 644–654).

"Collision–Free Hash Functions & Public Key Signature Schemes", I. B. Damgdard, Advances in Cryptology–Eurocrypt '87, Springer–Verlag, LNCS, 1988, vol. 304, pp. 203–216.

"Pseudorandom Generation From One–Way Functions", R. Impagliazzo & L. A. Levin, Proc. 21st STOC, pp. 12–24, ACM, 1989.

"The MD4 Message Digest Algorithm", R. L. Rivest, Crypto '90 Abstracts, Aug. 1990, pp. 281–291.

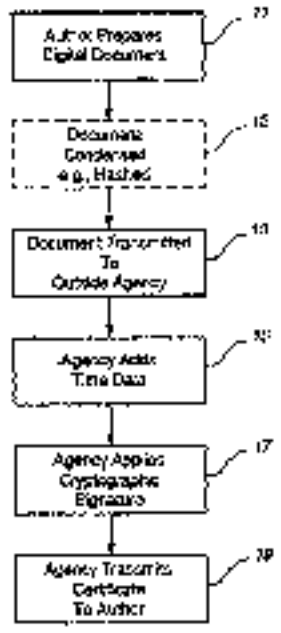
Alan G. Konheim, Cryptography, A Primer; (John Wiley & Sons, Inc., 1981), pp. 331–333.

Primary Examiner—Bernarr E. Gregory
Attorney, Agent, or Firm—Leonard Charles Suchyta; Lionel N. White

[57] **ABSTRACT**

A system for time-stamping a digital document, including for example text, video, audio, or pictorial data, protects the secrecy of the document text and provides a tamper-proof time seal establishing an author's claim to the temporal existence of the document. Initially, the author reduces the document to a number by means of a one-way hash function, thereby fitting a unique representation of the document text. In one embodiment of the invention the number is then transmitted to an outside agency where the current time is added to form a receipt which is certified by the agency using a public key signature procedure before being returned to the author as evidence of the document's existence. In later proof of such existence, the certificate is authenticated by means of the agency's public key to reveal the receipt which comprises the hash of the alleged document along with the time seal that only the agency could have signed into the certificate. The alleged document is then hashed with the same one-way function and the original and newly-generated hash numbers are compared. A match establishes the identify of the alleged document as the time-stamped original. In order to prevent collusion in the assignment of a time stamp by the agency and thus fortify the credibility of the system, the receipt is linked to other contemporary receipts before certification by the agency, thereby fixing a document's position in the continuum of time. In another embodiment, a plurality of agencies are designated by means of random selection based upon a unique seed that is a function of the hash number of the document to be time-stamped. Thus being denied the ability to choose at will the identity of an agent, the author cannot feasibly arrange for falsification of a time stamp.

25 Claims, 3 Drawing Sheets



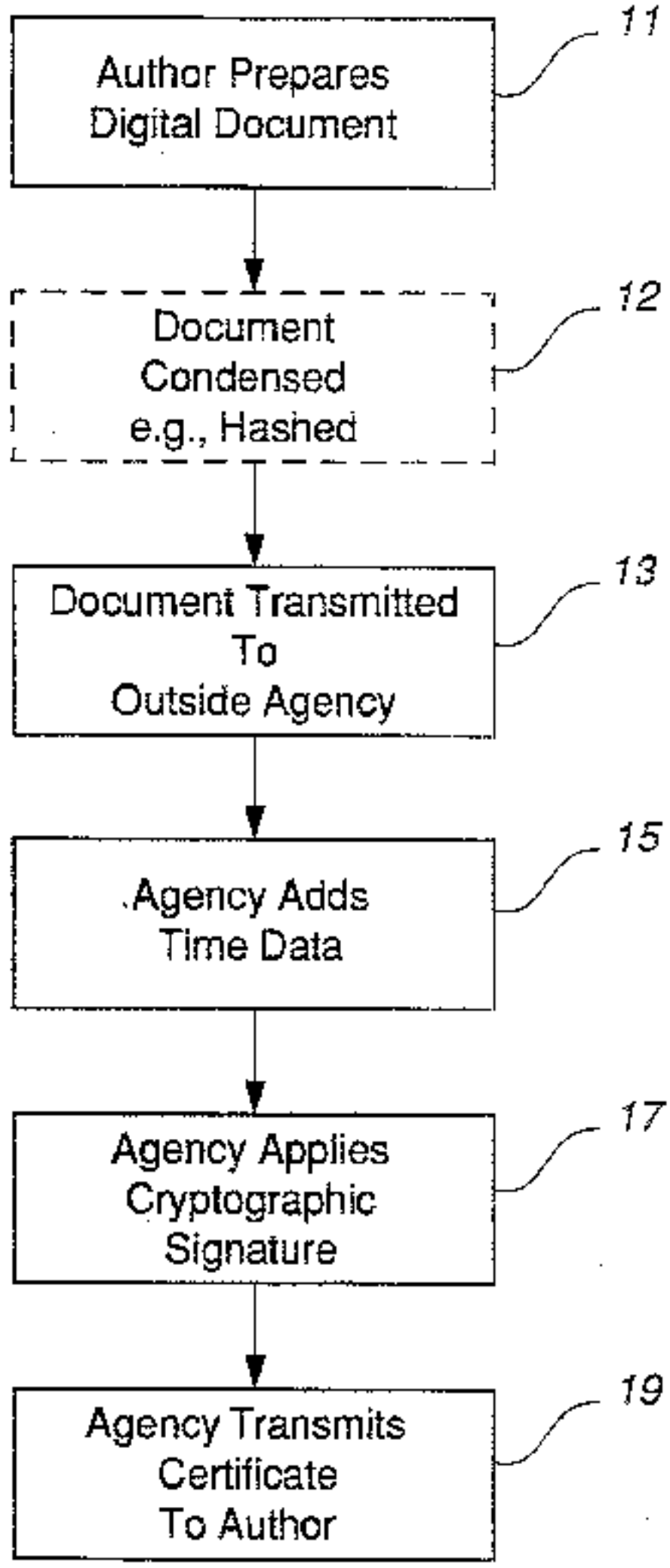

Author Prepares Digital Document
11
Document Condensed e.g., Hashed
12
Document Transmitted To Outside Agency
13
Agency Adds Time Data
15
Agency Applies Cryptographic Signature
17
Agency Transmits Certificate To Author
19

FIG. 1

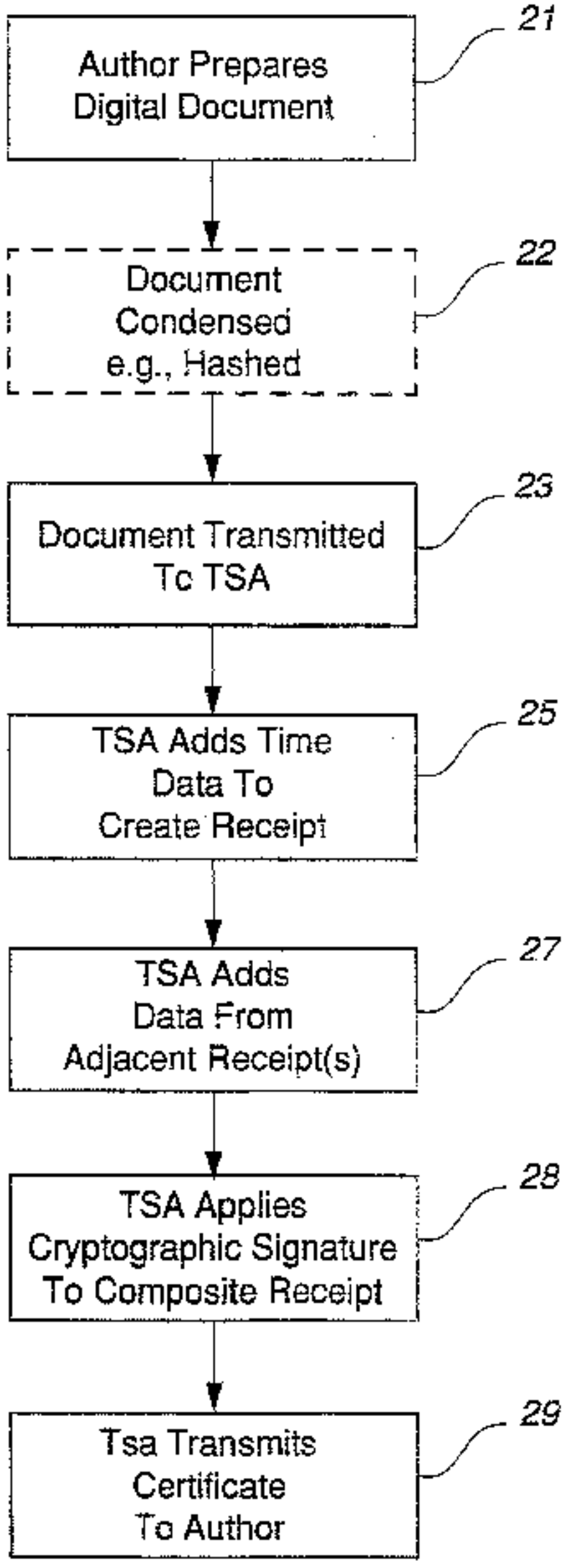

Author Prepares Digital Document
21
Document Condensed e.g., Hashed
22
Document Transmitted Tc TSA
23
TSA Adds Time Data To Create Receipt
25
TSA Adds Data From Adjacent Receipt(s)
27
TSA Applies Cryptographic Signature To Composite Receipt
28
Tsa Transmits Certificate To Author
29

FIG. 2

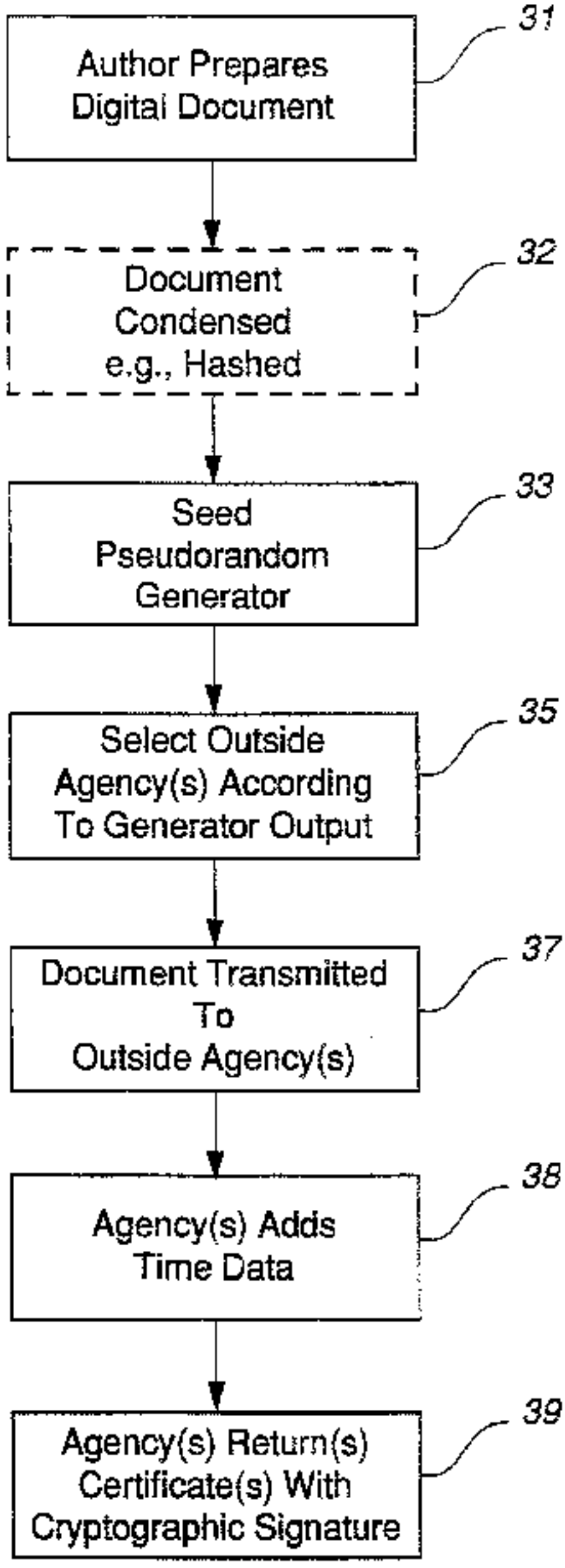

Author Prepares Digital Document
31
Document Condensed e.g., Hashed
32
Seed Pseudorandom Generator
33
Select Outside Agency(s) According To Generator Output
35
Document Transmitted To Outside Agency(s)
37
Agency(s) Adds Time Data
38
Agency(s) Return(s) Certificate(s) With Cryptographic Signature
39

FIG. 3

METHOD FOR SECURE TIME-STAMPING OF DIGITAL DOCUMENTS

Matter enclosed in heavy brackets [ ] appears in the original patent but forms no part of this reissue specification; matter printed in italics indicates the additions made by reissue.

BACKGROUND OF THE INVENTION

In many situations there is a need to establish the date on which a document was created and to prove that the text of a document in question is in fact the same as that of the original dated document. For example, in intellectual property matters it is often crucial to verify the date on which a person first put into writing the substance of an invention. A common procedure for thus "time-stamping" an inventive concept comprises daily notations of one's work in a laboratory notebook. Indelibly dated and signed entries are made one after another on each page of the notebook where the sequentially numbered, sewing pages make it difficult to revise the record without leaving telltale signs. The validity of the record is further enhanced by the regular review and signed witnessing by a generally disinterested third party. Should the time of the concept become a matter for later proof, both the physical substance of the notebook and the established recording procedure serve as effective evidence in substantiating the fact that the concept existed at least as early as the notebook witness date.

The increasingly widespread use of electronic documents, which include not only digital representations of readable text but also of video, audio, and pictorial data, now poses a serious threat to the viability of the "notebook" concept of establishing the date of any such document. Because electronic digital documents are so easily revised, and since such revisions may be made without telltale sign, there is available limited credible evidence that a given document truly states the date on which it was created or the message it originally carried. For the same reasons there even arises serious doubt as to the authenticity of a verifying signature. Without an effective procedure for ensuring against the surreptitious revision of digital documents, a basic lack of system credibility prevents the efficiencies of electronic documentation from being more widely implemented.

Some procedures are presently available for verifying electronic document transmissions; however, such procedures are limited in application to bilateral communications. That is, in such communications the sender essentially desires to verify to the receiver the source and original content of the transmitted document. For example, "private key" cryptographic schemes have long been employed for message transmission between or among a limited universe of individuals who are known to one another and who alone know the decrypting key. Encryption of the message ensures against tampering, and the fact that application of the private key reveals the "plaintext" of the transmitted message serves as proof that the message was transmitted by one of the defined universe. The time of creation of the message is only collaterally established, however, as being not later than its receipt by the addressee. This practice thus fails to provide time-stamp evidence that would be useful in an unlimited universe at a later date.

A more broadly applicable verifying communication procedure, that of "public key" cryptography, has been described by Diffie and Hellman ("New Directions in Cryptography", IEEE Transactions On Information Theory, Vol. IT-22, November 1976, pp. 644–654) and more recently implemented by Rivest et al. in U.S. Pat. No. 4,405,829, issued Sept. 20, 1983. While this scheme expands the utilizing universe to a substantially unlimited number of system subscribers who are unknown to one another, but for a public directory, verifiable communications remain bilateral. These limitations persist, since although a public key "signature", such as that which entails public key decryption of a message encrypted with the private key of the transmitter, provides any member of the unlimited universe with significant evidence of the identity of the transmitter of the message, only a given message recipient can be satisfied that the message existed at least as early as the time of its receipt. Such receipt does not, however, provide the whole universe with direct evidence of time of the message's existence. Testimony of such a recipient in conjunction with the received message could advance the proof of message content and time of its existence, but such evidence falls victim to the basic problem of ready manipulation of electronic digital document content, whether by originator or witness.

Thus, the prospect of a world in which all documents are in easily modifiable digital form threatens the very substance of existing procedures for establishing the credibility of such documents. There is clearly a significant present need for a system of verification by which a digital document my be so fixed in time and content that it can present, at least to the extent currently recognized in tangible documents, direct evidence on those issues.

SUMMARY OF THE INVENTION

The present invention yields such a reliable system in a method of time-stamping digital documents that provides the equivalent of two essential characteristics of accepted document verification. First, the content of a document and a time stamp of its existence are "indelibly" incorporated into the digital data of the document so that it is not possible to change any bit of the resulting time-stamped data without such a change being apparent. In this manner, the state of the document text is fixed at the instant of time-stamping. Second, the time at which the digital document is stamped is verified by a "witnessing" digital signature procedure that deters the incorporation of a false time statement. In essence, the method transfers control of the time-stamping step from the author to an independent agent and removes from the author the ability to influence the agent in the application of other than a truthful time stamp.

The method of the present invention presumes a number of document authors distributed throughout a communication network. Such authors may be individuals, companies, company departments, etc. each representing a distinct and identifiable, e.g. by ID number or the like, member of the author universe. In one embodiment of the invention, this universe may constitute the clientele of the time-stamping agency (TSA), while in another embodiment the distributed authors may serve as agents individually performing the time-stamping service for other members of the universe.

In its general application as depicted in FIG. 1 of the drawing, the present method entails an author's preparation of a digital document, which may broadly com-

prise any alphanumeric, audio, or pictorial presentation, and the transmission of the document, preferably in a condensed representative form, to the TSA. The TSA time-stamps the document by adding digital data signifying the current time, applying the agency's cryptographic signature scheme to the document, and transmitting the resulting document, now a certificate of the temporal existence of the original document, back to the author where it is held for later use in required proof of such existence.

To ensure against interception of confidential document information during transmission, and to reduce the digital bandwidth required for transmission of the entire document, the author may optimally convert the digital document string to a unique number having vastly reduced digital size by means of a deterministic function which may, for example, be any one of a number of algorithms known in the art as "oneway hash functions". Such an application of hash functions has been described, among others, by Damgard in his discussions on the improvement of security in document signing techniques ("Collision-Free Hash Functions and Public Key Signature Schemes", Advances in Cryptology—Eurocrypt '87, Springer-Verlag, LNCS, 1988, Vol. 304, pp. 203–217). In practice of the present invention, however, the "one-way" characteristic typical of a hashing algorithm serves an additional purpose;, that is, to provide assurance that the document cannot be revised subsequent to the time the TSA applies its time stamp.

A hashing function provides just such assurance, since at the time a document is hashed there is created a representative "fingerprint" of its original content from which it is virtually impossible to recover that document. Therefore, the time-stamped document is not susceptible to revision by any adversary of the author. Nor is the author able to apply an issued time-stamp certificate to a revised form of the document, since any change in the original content, even to the extent of a single word or a single bit of digital data, results in a different document that would hash to a completely different fingerprint number. Although the original document can thus not be recovered from the hashed document, a purported original document can nonetheless be proven by the fact that a true copy of the original document will always hash, assuming use of the same hashing algorithm, to the original number continued in the certificate.

Any available deterministic function. e.g. a one-way hash function such as that described by Rivest ("The MD4 Message Digest Algorithm", Advances in Cryptology—Crypto, '90, Springer-Verlag, LNCS, to appear), may be used in the present procedure. In the practice of the invention, such a hashing operation would normally be employed by the author to obtain the noted benefit of transmission security, although it might be effected by the TSA if the document were received in plaintext form. In whatever such manner the document content and incorporated time data are fixed against revision, there is, mains the further step, in order to promote the crucibility of the system, of certifying to the members of an as yet unidentified universe that the receipt was in fact prepared by the TSA, rather than by the author, and that the time indication is correct, i.e. that it has not, For instance, been fraudulently stated by the TSA in collusion with the author.

To satisfy the former concern, the TSA uses a verifiable signature scheme, of a type such as the public key method earlier noted, to certify the time-stamp prior to its transmittal to the author. Confirmation of the signature at a later time, such as by decryption with the TSA's public key, proves to the author and to the universe at large that the certificate originated with the TSA. Proof of the veracity of the time-stamp itself, however, relies upon a following additional aspect of the invention.

One embodiment of this segment of the process, as generally depicted in FIG. 2, draws upon the relatively continuous flow of documents from the universe of authors through the facilities of the TSA. For each given processed document D_k, the TSA generates a time-stamp receipt which includes, for example, a sequential receipt number, r_k, the identity of the author, A_k, by ID number ID_k, or the like, the hash, H_k, of the document, and the current time, t_k. In addition, the TSA includes the receipt dam of the immediately preceding processed document, D_{k-1}, of author, A_{k-1}, thereby bounding the timestamp of document, D_k, in the "past" direction by the malependently established earlier receipt time, t_{k-1}. Likewise, the receipt data of the next received document, D_{k+1}, are included to bound the time-stamp of document. D_k, in the "future" direction. The composite receipt, now containing the time data of the three, or more if desired, sequential time-stamp receipts, or identifying segments thereof, is then certified with the cryptographic TSA signature and transmitted to the author, A_k. In like manner, a certificate containing identifiable representations of D_k and D_{k+2} would be transmitted to author, A_{k+1}. Thus, each of the time-stamp certificates issued by the TSA is fixed in the continuum of time and none can be falsely prepared by the TSA, since a comparison of a number of relevant distributed certificates would reveal the discrepancy in their sequence. So effective is such a sequential fixing of a document in the me stream that the TSA signature could be superfluous in actual practice.

A second embodiment of the invention, shown generally in FIG. 3, distributes the time-stamping task randomly among a broad universe, for example the multiplicity of authors utilizing the time-stamping process. A TSA could still be employed for administrative purposes or the requesting author could communicate directly with the selected time-stamping author/agents. In either event, the above-mentioned need for assurance that a time-stamp has not been applied to a document through collusion between the author and the stamping agency is met in the combination of the reasonable premise that at least some portion of the agency universe is incorruptible or would otherwise pose a threat of exposure to an author attempting falsification, and the fact that the time-stamping agencies for a given document are selected from the universe entirely at random. The resulting lack of a capability on the part of the author to select a prospective collusive agent of the author's own choosing substantially removes the feasibility of intentional time falsification.

The selection of the individual universe members who will act as the predetermined number of agents is accomplished by means of a pseudorandom generator of the type discussed by Impagliazzo, Levin, and Luby ("Pseudorandom Generation From One-Way Functions", Proc. 21st STOC, pp. 12–24, ACM, 1989) for which the initial seed is a deterministic function, such as a hash, of the document being time-stamped. Given as a seed input the document hash or other such function, the implemented pseudorandom generator will output a

series of agency IDs. This agency selection is for all practical purposes unpredictable and random.

Once the agents are selected, the time-stamping proceeds as previously indicated with the exception that each agent individually adds the current time data to the representative document it receives, certifies the resulting separate time-stamped receipt with its own verifiable cryptographic signature, and transmits the certificate back to the author. This transmittal may be directly to the requesting author or by way of the administrative TSA where the receipts are combined with or without further certification by the TSA. The combination of signature scheme and a published directory of author IDs provides verification of the utilization of the agents that were in fact selected by the pseudorandom generator. This distributed agent embodiment of the invention presents some advantages over the receiptlinking procedure in that a certified time-stamp is provided more quickly and a given author's later proof of a document is less reliant upon the availability of the certificates of other authors.

Additional variations in the process of the invention might include the accumulation of documents, preferably in hashed or other representative form, generated within an author organization over a period of time. e.g. a day or more depending upon the extent of activity, with the collection being hashed to present a single convenient document for time-stamping and certification. Also, the initial seed for the pseudorandom generator may be based upon a function of time or previously receipted documents, as well as of the document. The implementation of the process my be automated in simple computer programs which would directly carry out the described steps of hashing and transmitting original documents, selecting time-stamping agents, applying current time stamps, and returning certified receipts.

THE DRAWING

The present invention will be described with reference to the accompanying drawing of which:

FIG. 1 is a flow diagram of the general process of time-stamping a document according to the invention;

FIG. 2 is a flow diagram of a specific embodiment of the process; and

FIG. 3 is a flow diagram of another specific embodiment of the process.

DESCRIPTION OF THE INVENTION

The following examples of the application of embodiments of the present invention will serve to further describe the involved process. For convenience in the presentation of these examples, the deterministic function selected is the md4 hashing algorithm described by Rivest, as mentioned above, and the verifiable signature scheme is the public key method suggested by Diffie and Hellman as implemented by Rivest et al. in U.S. Pat. No. 4,405,829. Further, in order to simplify explanation of the process and for the additional reasons noted below, only representative segments of the entire numbers will be employed.

The receipt-linking embodiment of the invention shown in FIG. 2 is initially considered. Although the present process may be used with documents of any length, the following apt excerpt is amply representative of a document, D_k, which an author prepares at step 21 and for which time-stamping is desired:

> Time's glory is to calm contending kings, To unmask falsehood, and bring truth to light, To stamp the seal of time in aged things, To wake the morn, and sentinel the night, To wrong the wronger till he render right;
>
> The Rape of Lucrece

By means of the md4 algorithm, the document is hashed, at optional dashed step 22, to a number, H_k, of a standard 128 bit format which expressed in base 16 appears as:

ef6dfdcd833f3a43d4515a9fb5ce3915

The author, A_k, whose system identification number ID_k, is 172 in a 1000 member author universe, transmits the thus-identified document to the system TSA, at step 22, as the message, (ID_k,H_k), which appears:

172, ef6dfdcd833f3a43d4515a9fb5ce3915

as a request that the document be time-stamped.

The TSA then prepares the receipt for document, D_k, by adding, at step 25, a sequential receipt number, r_k, of 132, for example, and a statement of the current rinse, t_k. This time statement might include a standard 32 bit representation of computer clock time plus a literal statement, i.e. 16:37:41 Greenwich Mean Tune on Mar. 10, 1990, in order to allow the final time-stamp certificate to be easily readable by the author, A_k. The receipt would then comprise the string, (r_k,t_k,ID_k,H_k).

At this point it would be appropriate to further consider the earlier-mentioned reduction of number size to representative segments. As is described by Rivest et al. in U.S. Pat. No. 4,405,829, the cryptographic public key scheme to be employed in this example (generally known in the field as the "RSA" signature scheme) requires the division of an extended message into blocks that may etch be represented by a number not exceeding the encoding key number element, n. Each such block is then signed with the RSA algorithm, to be reassembled after transmission. Therefore, in order to be able to use a number, n, of reasonable size in this example while maintaining a single block for the final receipt string to be certified with the RSA scheme, each element of the receipt string will be reduced to a representative eight bits, typically the last eight bits of any overlong string, and those bits will be sated in base 16 to present a two hexadecimal character string. Thus, for instance, the 128 bit document hash, H_k, will be represented by its last eight bits, i.e. 0001 0101, stated as 15 (base 16). Likewise, ID_k, 172, is 1010 1100 and is represented by ac (base 16). Without actually undertaking the calculation, it will suffice to assume that the time statement, t_k, is represented as 51. The receipt number, 132, would be represented as 84. The receipt string to this point, i.e. (r_k,t_k,ID_k,H_k) now appears as 8451ac15.

Assume now that the immediately preceding document, D_{k-1}, was processed by the TSA as the request:

201, d2d67232a61d616f7b87dc146c575174

at 16:32:30 on Mar. 10, 1990 (t_{k-1} being represented as 64). The TSA adds these data at step 27, to the receipt string for D_k to yield the hexadecimal representation, 8451ac1564c974. This receipt R_k, now contains data fixing the time for D_k and a time, t_{k-1}, before which author, A_k, cannot claim that D_k existed. This limitation on A_k is established by the fact that the previous author, A_{k-1}, holds a time certificate, C_{k-1}, that fixes t_{k-1} as subsequent to the linked time data, t_{k-2}, in the certifi-

cate of author, A_{k-2}, and so on for as long as a proof requires.

To establish that TSA in fact originated the receipt for document, D_k, that receipt is transmitted, at step **29** to author, A_k, after TSA signing, at step **28**, with the public key cryptographic signature scheme and becomes the certified receipt, or certificate, C_k. With the data derived above, and assuming that TSA has the RSA signature key set, in decimal:

$$<n,e> = <432006778214281O9, 191> \text{(Public)}$$

$$<n/d> = <43200677821428109, 2940360242244979 1> \text{(Private)}$$

the signed certificate for R_k, 8451ac1564c974, would compute as:

$$R^d \bmod n = 39894704664774392$$

When author, A_k, receives this certificate, C_k, along with the literal statement of R_k, it may be readily confirmed as being correct by application of the TSA public key to verify that:

$$C_k^e \bmod n = R_k$$

and that R_k in fact contains the data representing the document hash, H_k.

The procedure shown in this simple one-link example results in a certificate which, being bounded in time by the data from document, D_k, provides author, A_{k-1} with reliable evidence that document, D_{k-1}, was not backdated to a time significantly prior to the existence of document, D_k. When the certificate of A_k is expanded with additional data from the subsequently processed document, D_{k+1}, it will likewise be effectively bounded to substantiate the time stamp claimed by A_k. In an alternative of the same effect, A_k could simply be advised of the identity of A_{k+1} and could confirm from that author that the one-link certificate, C_{k+1}, contained the element, H_k. The procedure could also be varied to provide certified receipts which include data from any number of authors, with each addition providing a further degree of assurance against falsification.

Another embodiment of the invention, as shown in FIG. 3, which utilizes randomly selected members of the author universe as time-stamping agents, or witnesses, i.e. a "distributed trust" procedure, would proceed in the following manner. Although these numbers are not so limited in actual practice, for purposes of the example it will be assumed that the universe consists of 1000 authors, having IDs 0–999, and that three witnesses will be sufficient to establish the veracity of the time stamp. Also, in this example the earlier-noted variation including the services of a TSA is being implemented. The hashing function, **md4**, utilized in the above example is employed here also, in optional step **32**, as an example of a deterministic document function which will seed the pseudorandom selection of the three witnesses from the author universe.

As in the previous example, the author transmits the document to the TSA, normally in hashed form, as the identified request:

172, ef6dfded833f3a43d4515a9fb5ce3915

The TSA now uses this document hash string, in step **33**, as the seed to generate the ID number of the first witness, at step **35**, according to the selection algorithm:

$$ID = \{\text{md4(seed)}\} \bmod (\text{universe size})$$

The resulting seed hash:

26f34eae9251 1d6b5e06e7c2de6e0fcf

represents the 128 bit number which mod **1000** is 487, the ID of the tint selected witness. The next witness is likewise chosen using this seed hash representation as the seed in the second selection computation to yield:

882653cc04d16b1f0d604883aa273006

which rood **1000**, is 571, the second witness ID. A repeat of the computation, again seeding with the prior seed hash, selects the final witness as **598**, which is:

2fe8768ef3532f15c40acf341902ele mod 1000

The TSA now sends, at step **37**, a copy of the original request to each of these three witnesses who individually, at step **38**, add a current time statement and ID, and certify the resulting receipts by signing with the RSA cryptographic signature scheme and transmitting them, at step **39**, directly to the author or through the TSA who may assemble the certificates into a file to be delivered to the author. By virtue of the fact that the pseudorandom generation prevents the exercise of a personal choice in the selection of witnesses, the author is deterred by the risk of encountering a non-cooperative witness from attempting any communication prior to time stamp certification for the purpose of arranging for a false time entry. In a process variant where the author is allowed to transmit the request directly to witnesses, the random selection of such witnesses which is keyed essentially to the involved document itself frustrates any attempt by the author to direct the document to a known cooperative witness. The group of resulting certificates my thus be employed with confidence in later proofs employing signature verification in the manner earlier described.

The procedures described and variants suggested herein for the practice of this time-stamping process and the various other embodiments which will become apparent to the skilled artisan in the light of the foregoing description are all nonetheless to be included within the scope of the present invention as defined by the appended claims.

What is claimed is:

1. A method of time-stamping a digital document which comprises:

a) transmitting a digital representation of said document from an originator to an outside agency;

b) creating at said outside agency a receipt comprising a digital representation of then current time and at least a portion of a digital representation of said digital document; and

c) certifying said receipt at said outside agency by means of a verifiable digital cryptographic signature.

2. A method of time-stamping a digital document according to claim **1** wherein said transmitted digital document representation comprises at least a portion of the digital representation of the number derived by

application of a deterministic function algorithm to said digital document.

3. A method of time-stamping a digital document according to claim **1** wherein said receipted digital document representation comprises at least a portion of the digital representation of the number derived by application of a deterministic function algorithm to said digital document.

4. A method of time-stamping a digital document according to claim 3 wherein said digital number representation is derived from the application of a one-way hashing algorithm to said digital document.

5. A method of time-stamping a digital document according to claim **1** wherein said receipt further comprises the time representation and digital document representation specific to at least one other digital document receipted by said outside agency.

6. A method of time-stamping a digital document according to claim **5** whereto the receipt of said at least one other digital document was created by said outside agency earlier than that of the currently receipted digital document.

7. A method of time-stamping a digital document according to claim **5** wherein the receipt of said at least one other digital document was created by said outside agency later than that of the currently receipted digital document.

8. A method of time-stamping a digital document according to claim **1** wherein aid outside agency is selected at random from a predetermined universe.

9. A method of time-stamping a digital document according to claim **8** wherein said outside agency is selected by means of a pseudorandom generator seeded with at least a portion of the digital representation of the number derived by application of a deterministic function algorithm to said digital document.

10. A method of time-stamping a digital document according to claim **9** wherein said pseudorandom generation seed is derived from the application of a one-way hashing algorithm to said digital document.

11. A method of time-stamping a digital document according to claim **10** which further comprises the like preparation of a time-stamp certificate by at least one additional outside agency selected by said pseudorandom generation and wherein the input for each additional outside agency selection is at least a portion of the digital representation of the output derived from the application of said one-way hashing algorithm to a digital representation of the previously generated output.

12. A method of time-stamping a digital document according to claim **9** which further comprises the like preparation of a time-stamp certificate by at least one additional outside agency selected by said pseudorandom generation.

13. A method for the secure time-stamping of a digital document

characterized in that

a) a digital representation of said document is transmitted from an originator to an outside agency;

b) said outside agency creates a receipt comprising a digital representation of then current time and at least a portion of a digital representation of said digital document; and

c) said receipt is certified al said outside agency by means of a verifiable digital cryptographic signature scheme.

14. A method for the secure time-stamping of a digital document according to claim **13**

characterized in that said receipt further comprises the time representation and digital document representation specific to at least one other digital document receipted by said outside agency.

15. A method for the secure time-stamping of a digital document according to claim **14**

characterized in that the receipt of said at least one other digital document was created by said outside agency later than that of the currently receipted digital document.

16. A method for the secure time-stamping of a digital document according to claim **13**

characterized in that said outside agency is selected at random from a predetermined universe by means of a pseudorandom generator seeded with at least a portion of the digital representation of the number derived from the application of a deterministic function algorithm to said digital document.

17. A method for the secure time-stamping of a digital document according to claim **16**

characterized in that said seed is derived from the application of a one-way hashing algorithm to said digital document.

18. A method for the secure time-stamping of a digital document according to claim **16**

characterized in that a time-stamp certificate for said digital document is likewise prepared by at least one additional outside agency selected by means of said pseudorandom generation.

19. A method for time stamping a particular digital document comprising

(a) forming a collection of hash values by hashing each group comprised of one or more digital documents in an accumulation of digital documents, including said particular digital document, generated over a period of time,

(b) forming a single hash value representative of said collection of hash values, and

(c) creating a time stamp receipt for said particular digital document by associating an indication of then current time with said single hash value.

20. A method for time stamping a particular digital document comprising

(a) forming a collection of digital values by applying a deterministic function to each digital document or group of digital documents in an accumulation of digital documents, including said particular document, generated over a period of time,

(b) forming a single digital value from said digital values in said collection, and

(c) creating a time stamp receipt for said particular digital document by associating an indication of then current time with said single digital value.

21. A method for time stamping a particular digital document in a series of digital documents comprising the steps of

(a) forming a receipt comprising at least a portion of a digital representation of said particular digital document and a digital representation of time then current as of the forming of said receipt,

(b) forming at least one different receipt comprising at least a portion of a digital representation of at least one different digital document in said series of digital documents and a digital representation of time then current as of the forming of said at least one different receipt, and

(c) utilizing said receipts to form a composite receipt which is thereby fixed in the continuum of time.