# TIMESEC

**Digital Timestamping and the Evaluation of Security Primitives**

Programme of scientific support to standardisation

part II

# Final report

Belgian Federal Office for
SCIENTIFIC,TECHNICAL AND
CULTURAL AFFAIRS

N0101007

# TIMESEC

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

# Final Report

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

December 1999

# Contents

# 1 Introduction

This report summarizes all the results and conclusions that have been obtained for the TIMESEC project. The goal of this project was the development of a complete system for digital timestamping, as well as a methodology for the evaluation of security primitives.

First we will give an overview of the complete system as it was developped, and next we will show the methodology used, by discussing the work done for the different work packages of the project. We also discuss the current situation of standardization in this area and the role played herein by the project partners. We conclude by summing up the publications that were made, which illustrate or continue the work done for this project.

# 2 Overview of the complete system

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, 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 unpartiality of the entity that is in charge of issuing the timestamps. We can also classify those techniques into two different types: those where the third party is completely trusted and those where it is partially trusted.

The technique based on distributed trust consists 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 [MQ97]. 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.

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 [BHS92] and [BdM91]. All the timestamping requests 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 implemented two of them which seemed the most practical (see section 2.1).

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

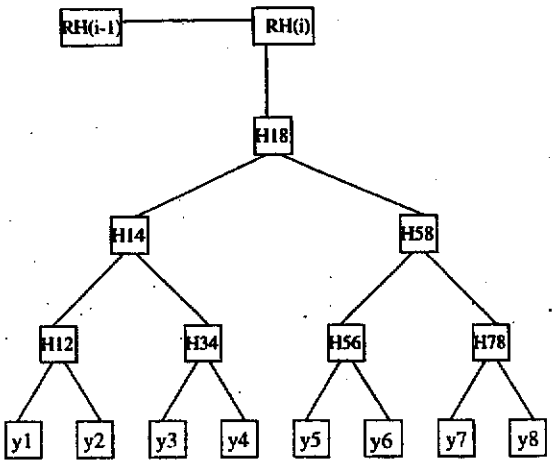


Figure 1: The binary tree structure

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 [BHS92]. It basically consists of re-timestamping the document as well as the original timestamp before the hash function is broken.

## 2.1 Two methods of accumulation

**Using a binary tree.** The first method of accumulation uses a binary tree structure and has been described in [HS91] and [HS97]. 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.

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 [HS91], [MQ97] and [MAQ99a].

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. In that way, the system remains secure in case of an unexpected break of one of the hash functions used.

**Using a one-way accumulator** An alternative method of accumulation was proposed in [BdM94]. 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 a modular exponentiation operation:

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

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

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

Hence an individual timestamp for request $y_j$ consists of the values $Z_j$ and $Z_m$. 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 security of this method relies on the well known RSA problem: to produce a fake timestamp for a value $y'$, one should find a value $Z'$ with

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

If the factorization of the modulus $(N = p \times q)$ is unknown this is unfeasible. As an additional feature we can increase the security by constructing the primes of the modulus as follows: $p = 2p' + 1$ and $q = 2q' + 1$, where $p'$ and $q'$ are also primes. 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 [Mas98].

# 3 Work done for the project

## 3.1 Time and cryptography

The interaction between time and cryptography is a new subject. Nevertheless, the time in cryptography is of high importance, especially for electronic contracts.

First we define what we call *time* in cryptography and the different situations we will deal with. Two of the needs are: timestamping a document to answer the question "When has this document been created?"; and sending information at some point in the future.

An important bibliographic work as well as an analysis has been done. Some solutions were given in the past but they leave some open problems and none have been standardized. There are two kinds of solutions: the ones which need a trusted third party, and the others which are based on distributed trust. A solution to date safely is to link the requests. We presented two solutions based on this idea.

Several protocols have been patented, we have studied these US-patents. We have precisely described the verification protocols and we have made the cryptanalysis of all the algorithms. We have studied in details all the aspects of the protocols and then drawn conclusions.

To finish we proposed a terminology and an informal method based on this work. For details of this work we refer to the first technical report [MQ97]. An overview and analysis of timestamping schemes are also presented in [RPV99].

## 3.2 Evaluation of security primitives

**Definition 1** *The **timestamp** of a document is something added to it which proves that the document has been issued **before, after** or **at** a certain time.*

In practice timestamps will be issued by a *trusted third party* (TTP). In the most elementary protocol, the issuer of a document will send it to the TTP, which will add date and time, and send it back after computing a *digital signature* over it.

A **digital signature** is a security primitive that allows, just like its equivalent on paper, to link a document to the author of it (in this case the TTP). It is clear that the cryptographic security of this primitive is essential for a secure operation of the timestamping service: it must be impossible for a dishonest party to forge signatures from the TTP.

A **hash function** is another security primitive that will be used extensively by the timestamping service. These functions allow to represent a document of arbitrary length by a corresponding hash value of fixed (and generally much shorter) length. They are used for different reasons:

- Instead of sending the document $X$ to the TTP, one will send its hash value $h(X)$. In this way the original document remains secret.
- Digitally signing is a computationally expensive operation. By signing the hash value which is much shorter, the efficiency of the service will be greatly improved.
- In order to increase the security of the protocol, timestamps succeeding in time will be linked to one another (building an unmodifiable chronological chain). To restrict the length of the required linking information, it will be compressed by a hash function.

If a document is to be securely represented by its hash value, the hash function that is used will have to meet certain cryptographic requirements: the function must be uninvertable (unfeasible to find a document which hashes to a given value), and collision-free (unfeasible to find two documents which hash to the same value).

In the remainder of the second technical report [PRQM97] we describe an evaluation methodology for digital signature algorithms and hash functions, as well as the perceived security of existing algorithms. Several candidates are proposed, which are expected to offer the required security for some years to come. An analysis of practical hash functions is also presented in [RPV98].

## 3.3 Design of a timestamping system

In the third technical report [PRQ+98] we have presented a design of a timestamping system which matches the requirements of the TIMESEC project. The originality of this work is all what has been done on top of the timestamping algorithms (tree technique and accumulator technique).

Our main goal was to minimize the trust required in the third party providing the timestamping service. To achieve that we have chosen timestamping techniques which allow 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 users in two parts: the clients who ask for time certificates, and the verifiers who will check the correctness of these certificates to be convinced by them. We also tried to reuse as much as possible the existing Internet technology and infrastructure.

The system works by rounds, so the reliable accuracy of the timestamp is the length of a round, which has to be chosen for the implementation. 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 preceeding round values as well as on the requests that have been submitted during the corresponding round. At fixed time laps that will be chosen for the implementation, a round value will be published on an unmodifiable media (hence everybody can consult this value but nobody, even the STA, can change it).

The security of our system ultimately relies on hash functions, so we use two hash functions in parallel to prevent the failure of the system in case one of them is broken. The time certificates have a limited 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 before both hash functions are broken.

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 transmission issues.

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

### 3.4 Implementation issues

The remarks that we make in this section come from our experience in designing and implementing a complete timestamping system (see [MAQ99a]). Our principal design requirement was to minimize the trust required in the third party issuing the timestamps. This work is described in the fourth and last technical report [QMA+99].

We developped two seperate implementations based on the techniques presented 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, chosen according to the work in [PRQM97]). 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 [MAQ99b], 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.

**Implementation based on the tree technique.** The binary tree is defined for a number of leafs (requests) 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 these nodes "special nodes".

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 thought 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 determined when the request is put in the round queue. The round queue does not accept any requests 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 the issuing of the timestamps will be delayed.

The values timestamped as well as the round values are logged in 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 (the values for the big rounds are published on an unmodifiable and widely witnessed media). With this material, the verifier will be able to check the validity of the timestamp without having to trust the STA at all.

**Implementation based on the accumulator technique.** The second implementation 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 2.1 for the notations used in this discussion) we need to generate the corresponding value $Z_j$. We also need the accumulated value $Z_m$, 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 $Z_m = Z_j^{y_j} \mod N$, and finally checking the linking of succeeding rounds until trusted (published) round values are obtained.

# 4 Standardization

For the interactions with standardization committees, we mainly focused on two major standardization organizations: ISO[1] and IETF[2].

A work item entitled "Information technology - Security techniques - Time stamping services" has been added to the ISO/IEC JTC1/SC27 working group. The reference of this working draft is ISO/IEC WD 18014. We take an active part in the development of this new standard by making the comments for Belgium as well as participating to the discussions.

A familiarization work has been done for the IETF because we had no experience with this standardization committee. The work done by the PKIX[3] and the SPKI[4] groups has been followed. A working group concerning timestamping has been created inside PKIX. We have participated to this working group mainly by making comments and discussing with the people in charge of writing the internet draft.

# 5 Publications

Several papers that illustrate or continue the work done for TIMESEC have been published and presented at conferences:

- Henri Massias, Xavier Serret and Jean-Jacques Quisquater: **Timestamps: Main issues on their use and implementation.** In *Proceedings of IEEE 8th International Workshop on enabling Technologies: Infrastructure for Collaborative Enterprises - Fourth International Workshop on Enterprise Security*, ISBN 0-7695-0365-9, pp. 178-183, June 1999.
- Henri Massias, Xavier Serret and Jean-Jacques Quisquater: **Design of a secure timestamping service with minimal trust requirement.** In *A. Barbé, E.C. van der Meulen and P. Vanroose, eds, Proceedings of the 20th symposium on Information Theory in the Benelux*, pp. 79-86, May 1999.
- Bart Van Rompay, Bart Preneel and Joos Vandewalle, **On the security of dedicated hash functions.** In *P.H.N. de With and M. v.d. Schaar-Mitrea, eds, Proceedings of the 19th Symposium on Information Theory in the Benelux*, pp. 103-110, May 1998.
- Bart Van Rompay, Bart Preneel and Joos Vandewalle, **The digital timestamping problem.** In *A. Barbé, E.C. van der Meulen and P. Vanroose, eds, Proceedings of the 20th Symposium on Information Theory in the Benelux*, pp. 71-78, May 1999.

# 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 decentralized alternative to digital signatures. In Tor Helleseth, editor, *Advances in Cryptology - Proceedings of Eurocrypt'93*, number 765, pages 274–285. Springer-Verlag, 1994.


[1] International Organization for Standardization
[2] Internet Engineering Task Force
[3] Public Key Infrastructure X.509
[4] Simple Public Key Infrastructure

[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.

[MAQ99a] 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.

[MAQ99b] 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.

[Mas98] H. Massias. A survey of accumulators. Technical report, UCL Crypto Group Technical Report, February 1998.

[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`.

[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`.

[QMA+99] J.-J. Quiquater, H. Massias, X. Serret Avila, B. Preneel, and B. Van Rompay. Specification and implementation of a timestamping system. Technical report, TIMESEC Project (Federal Governement Project, Belgium), 1999. To be available at `http://www.dice.ucl.ac.be/crypto/TIMESEC.html`.

[RPV98] B. Van Rompay, B. Preneel, and J. Vandewalle. On the security of dedicated hash functions. In P.H.N. de With and M. v.d. Schaar-Mitrea, editors, *Nineteenth Symposium on Information Theory in the Benelux*, pages 103–110, May 1998.

[RPV99] B. Van Rompay, B. Preneel, and J. Vandewalle. The digital timestamping problem. In A. Barbé, E.C. van der Meulen, and P. Vanroose, editors, *Twentieth Symposium on Information Theory in the Benelux*, pages 71–78, May 1999.