Time-Stamping with Binary Linking Schemes

Ahto Buldas[1], Peeter Laud[2], Helger Lipmaa[1], and Jan Villemson[2]

[1] Cybernetica; Akadeemia 21, EE0026 Tallinn, Estonia
[2] Cybernetica, Tartu Lab; Lai 36, EE2400 Tartu, Estonia
{ahtbu,peeter,helger,jan}@cyber.ee

Abstract. We state the basic requirements for time-stamping systems applicable as the necessary support to the legal use of electronic documents. We analyze the main drawbacks of the time-stamping systems proposed to date and present a new system that meets all the stated requirements. We prove that these requirements cannot be significantly tightened.

1 Introduction

Time-stamping ([HS91], [BdM91], [BHS92]) is a set of techniques enabling us to ascertain whether an electronic document was created or signed at a certain time. The real importance of time-stamping becomes clear when there is a need for a legal use of electronic documents with a long lifetime. Without time-stamping we neither can trust signed documents when the cryptographic primitives used for signing have become unreliable nor solve the cases when the signer himself repudiates the signing, claiming that he has accidentally lost his signature key. During the last years, especially in the context of legal regulation of using digital signatures, the organizational and legal aspects of time-stamping itself have become the subject of world-wide attention. In addition to defining the responsibilities of the owner of the signature, duties and responsibilities of the third party (Time-Stamping Service, TSS) must be stated as well. Hence, there is an increasing interest in time-stamping systems where the need to trust the TSS is minimized. In order to make users liable only for their own mistakes, there has to be a possibility to ascertain the offender.

Unlike physical objects, digital documents do not comprise the seal of time. Thus, the association of an electronic document uniquely with a certain moment of time is very complicated, if not impossible. Even by the theory of relativity, no absolute time exists. The best we can achieve with time-stamping is the relative temporal authentication (RTA) based on the complexity-theoretic assumption on the existence of collision-resistant one-way hash functions. RTA enables the verifier given two time-stamped documents to verify which of the two was created earlier.

The main drawbacks of the time-stamping systems proposed to date concern (1) the need to unconditionally trust the TSS and (2) the time-complexity of RTA, which is linear on the number of issued time-stamps.

In the current paper theoretical and practical requirements are discussed and a new time-stamping system is presented (1) in which the need to trust the TSS is significantly diminished and (2) which offers RTA with the complexity proportional to the logarithm of the number of issued time-stamps.

In Sect. 2 the time-stamping solutions proposed to date are analyzed. Sect. 3 clarifies the security objectives of time-stamping by giving essential requirements to the time-stamping systems. In Sect. 4 the protocols of the new time-stamping system are described using the linear linking scheme. In Sect. 5 binary linking schemes are introduced and a scheme with logarithmic verifying time is presented. In Sect. 6 we prove that the requirements stated in Sect. 3 cannot be tightened.

2 Existing Time-Stamping Systems

By a simple time-stamping protocol ([HS91], Sect. 4), the TSS appends the current time t to the submitted document X, signs the composite document (t, X) and returns the two values t and $s = \mathrm{sig}_{\mathrm{TSS}}(t, X)$ to the client. The weaknesses of this scheme are the unreliability of old time-stamps after a possible leakage of the signature key of the TSS and the impossibility of verifying whether s was issued actually at time t stated in the time-stamp, implying that *the TSS has to be unconditionally trusted*. Because of these drawbacks it has been widely accepted that a secure time-stamping system cannot rely solely on keys or on any other secret information. An overview of the existing time-stamping solutions is given in [MQ97].

2.1 Linear Linking Scheme (LLS)

In order to diminish the need for trust, the users may demand that the TSS links all time-stamps together into a chain using a collision-resistant hash function H as was proposed in [HS91], Sect. 5.1 (variant 1). In this case the time-stamp for the n-th submitted document X_n is

$$s = \mathrm{sig}_{TSS}(n, t_n, \mathrm{ID}_n, X_n, L_n) \ ,$$

where t_n is the current time, ID_n is the identifier of the submitter and L_n is the linking information defined by the recursive equation

$$L_n := (t_{n-1}, \mathrm{ID}_{n-1}, X_{n-1}, H(L_{n-1})) \ .$$

There are several complications with the practical implementation of this scheme. At first, the number of steps needed to verify the one-way relationship between two time-stamps is linear with respect to the number of time-stamps between them. Hence, a single verification may be as costly as it was to create the whole chain. This solution has impractical trust and broadcast requirements, as it was pointed out already in [BdM91]. A modification was proposed in [HS91] (Sect. 5.1, variant 2) where every time-stamp is linked with $k > 1$ time-stamps directly preceding it. This variation decreases the requirements for broadcast by increasing the space needed to store individual time-stamps.

2.2 Tree-Like Schemes

Two similar tree-like schemes have been proposed [BdM91, BHS92]. In the Haber-Stornetta scheme [BHS92, HS97], the time-stamping procedure is divided into rounds. The time-stamp R_r for round r is a cumulative hash of the time-stamp R_{r-1} for round $r-1$ and of all the documents submitted to the TSS during the round r. After the end of the r-th round a binary tree T_r is built. Every participant P_i who wants to time-stamp at least one document in this round, submits to the TSS a hash $y_{r,i}$ which is a hash of R_{r-1} and of all the documents he wants to time-stamp in this round. The leafs of T_r are labeled by different $y_{r,i}$. Each inner node k of T_r is recursively labeled by $H_k := H(H_{k_\mathsf{L}}, H_{k_\mathsf{R}})$, where k_L and k_R are correspondingly the left and the right child nodes of k, and H is a collision-resistant hash function. The TSS has to store only the time-stamps R_r for rounds (Fig. 1). All the remaining information, required to verify whether a certain document was time-stamped during a fixed round, is included into the individual time-stamp of the document.

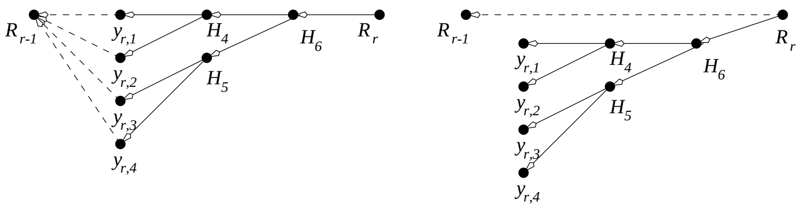


Fig. 1. An example of the time-stamp for round r by the schemes presented in [BdM91] (left) and [BHS92] (right).

For example, the individual time-stamp for $y_{r,3}$ is $[r;\ (y_{r,4}, \mathsf{L}), (H_4, \mathsf{R})]$. The verifying procedure of the time-stamp of $y_{r,3}$ consists of verifying the equality $R_r = H(H(H_4, H(y_{r,3}, y_{r,4})), R_{r-1})$. Here, the size of a single time-stamp is logarithmic with respect to the number of participants submitting their documents to the TSS for the current round.

The Haber–Stornetta linking scheme [BHS92, HS97] differs slightly from the Benaloh–de Mare scheme [BdM91]. Here, the time-stamp R_n for the n-th round is linked directly to R_{n-1}, enabling the verifier to check one-way dependencies between R_i without examining the individual time-stamps of the submitted documents. This is impossible in the Benaloh–de Mare scheme. However, in the Haber–Stornetta scheme the individual time-stamps in the n-th round are not linked to the time-stamp R_{n-1} for previous round.

These schemes are feasible but provide the RTA for the documents issued during the same round only if we unconditionally trust the TSS to maintain the order of time-stamps in T_r. Therefore, this method either increases the need for trust or otherwise limits the maximum temporal duration of rounds to the insignificant units of time (one second in Digital Notary system). However, if the number of submitted documents during a round is too small, the expenses of time-stamping a single document may become unreasonably large (Sect. 3.3).

3 Security Objectives

In the following we give a definition of time-stamping systems applicable in legal situations. Later we will justify our approach and compare it to older systems.

A *time-stamping system* consists of a set of principals with the time-stamping server (TSS) together with a triple $(\mathbf{S}, \mathbf{V}, \mathbf{A})$ of protocols. The stamping protocol $\mathbf{S}$ allows each participant to post a message. The verification protocol $\mathbf{V}$ is used by a principal having two time-stamps to verify the temporal order between those time-stamps. The audit protocol $\mathbf{A}$ is used by a principal to verify whether the TSS carries out his duties. Additionally, no principal (in particular, TSS) should be able to produce fake time-stamps without being caught.

A time-stamping system has to be able to handle time-stamps which are anonymous and do not reveal any information about the content of the stamped data. The TSS is not required to identify the initiators of time-stamping requests.

Our notion of time-stamping system differs from the one given in, e.g., [BdM91] by several important aspects. Below we motivate the differences.

3.1 Relative Temporal Authentication

The main security objective of time-stamping is *temporal authentication* [Jus98]– ability to prove that a certain document has been created at a certain moment of time. Although the creation of a digital data item is an observable event in the physical world, the moment of its creation cannot be ascertained by observing the data itself (moreover, no such thing as the absolute thing exists). The best one can do is to check the *relative* temporal order of the created data items (i.e., prove the *RTA*) using one-way dependencies defining the arrow of time, analogous to the way in which the growth of entropy defines the arrow of time in the physical world ([Haw88], Chap. 9). For example, if H is a collision-resistant one-way hash function, one can reliably use the following "rough" derivation rule: if $H(X)$ and X are known to a principal P at a moment t, then someone (possibly P himself) used X to compute $H(X)$ at a moment prior to t. To date, the existence of one-way functions has not been proved. Therefore, the proposed time-stamping systems make sense only under the hypothesis of the existence of collision free one-way hash functions.

Definition 1. *A* collision-resistant one-way hash function *([MOV96], Sect. 9.2) is a function H which has the properties of* compression*,* ease of computation*,* preimage resistance*,* 2nd-preimage resistance *and* collision resistance*.*

Definition 2. *Let ρ be a binary relation on $\mathbb{N}$, such that $x\ \rho\ y$ implies $x < y$ and H be a collision-resistant one-way hash function. A (ρ, H)*-linking scheme *is* *a procedure to link a family (H_n) of data items together using auxiliary linking items L_n satisfying the recursive formula*

$$L_n := H(H_n, L_{n_1}, \dots , L_{n_{\sharp\rho^{-1}(n)}}) \ , \tag{1}$$

where $n_1 \geq \cdots \geq n_{\sharp\rho^{-1}(n)}$ are exactly the elements of $\rho^{-1}(n) := \{m \mid m\ \rho\ n\}$ (the preimage of n by ρ). A sequence $(m_i)_{i=1}^{\ell}$, where $m_i\ \rho\ m_{i+1}$, is called a verifying chain *between m_1 and m_ℓ with length ℓ.*

In the context of time-stamping $H_n = H(n, X_n)$, where X_n denotes the n-th time-stamped document. The linking item L_n is also referred to as a time-stamp of X_n. Note that a one-way relationship between L_n and L_m $(n < m)$ does not prove that in the moment of creating X_n the bit-string X_m did not exist. All we know is that X_n did exist at the moment of creating L_m.

We have omitted the t_n in the formula for H_n, whereas it should not be taken for granted that the value t_n indeed represents the submission time of X_n. The only way for a principal to associate a time-stamp with a certain moment of time is to time-stamp a nonce at this moment. By a nonce we mean a sufficiently long random bit-string, such that the probability it has been already time-stamped is negligible. In order to verify the absolute creating time of a document time-stamped by another principal, the verifier has to compare the time-stamp with the time-stamps of nonces generated by the verifier herself. In this solution there are neither supplementary duties to the TSS nor to the other principals. The use of nonces illustrates the similarity between time-stamping and ordinary authentication protocols, where nonces are used to prevent the possible reuse of old messages from previous communications.

By using RTA it is possible to determine not only the submitting time of the signature but also the time of signing the document. Before signing a document X the principal P generates a nonce N and time-stamps it. He then includes the time-stamp $L(N)$ of N to the document, signs it and obtains the time-stamp $L(\sigma)$ of the signature $\sigma = \mathrm{sig}_P(L(N), X)$. From the view-point of the TSS these stamping events are identical (he need not be aware whether he is time-stamping a nonce or meaningful data). For the verification of the document X, the verifier has to compare both these time-stamps with the time-stamps trusted by her. As there are one-way dependencies between $L(N)$, σ and $L(\sigma)$ the verifier may conclude that the signature was created in the time-frame between the moments of issuance of $L(N)$ and of $L(\sigma)$ respectively. If these moments are close enough, the signing time can be ascertained with necessary precision.

3.2 Detection of Forgeries

A time-stamping system must have properties enabling users to verify whether an arbitrary time-stamp is correct or not. Possession of two documents with corresponding time-stamps is not enough to prove the RTA between the *documents* because everyone is able to produce fake chains of time-stamps.

A time-stamping system should allow (1) to determine whether the time-stamps possessed by an individual have been tampered with; and (2) in the case of tampering, to determine whether the time-stamps were tampered by the TSS or tampered after the issuing (generally by unknown means). In the second case, there is no-one to bring an action against. The principals interested in legal use of time-stamps should themselves verify their correctness immediately after the issuing (using signatures and other techniques discussed later) because if the signature of the TSS becomes unreliable, the signed time-stamps cannot be used as an evidence. In order to increase the trustworthiness of the time-stamping services it should be possible for the clients to periodically inspect the TSS. Also, in the case when the TSS is not guilty he should have a mechanism to prove his innocence, i.e., that he has not issued a certain time-stamp during a certain round.

Additionally, the TSS must publish regularly, in an authenticated manner, the time-stamps for rounds [BdM91] in mass media. If the time-stamping protocol includes (by using collision-resistant one-way hash functions) (1) the message digest of any time-stamp issued during the r-th round into the time-stamp for r-th round, and (2) the message digest of the time-stamp for round $r-1$ into any time-stamp issued during the r-th round, it will be intractable for anyone to undetectably forge a time-stamp. The forgery detection procedures should be simple. Forgeries should be determinable either during the stamping protocol (when the time-stamp, signed by the TSS, fails to be correct) or later when it is unable to establish the temporal order between two otherwise correct time-stamps (see Sect. 4 for details).

3.3 Feasibility Requirements

The time-stamping systems of [BdM91] and [HS97] use nonlinear partial ordering of time-stamps and therefore do not support the RTA. Sect. 4 shows how to modify the linear linking scheme ([HS91], Sect. 5.1) to fulfill the security objectives (RTA and detection of forgeries). On the other hand, *in practice*, in this scheme the detection of forgeries would take too many steps. As noted in [Jus98], it is easy to forge time-stamps when we can assume that the verifier has limited computational power. This leads us to the question of feasibility. In order to make RTA feasible in the case when time-stamps belong to different rounds, it is reasonable to define an additional layer of links between the time-stamps for rounds.

Definition 3. *Assume we are given* (ρ, H) *and* (δ, H) *linking schemes and a monotonically increasing function* $\xi\colon \mathbb{N} \to \mathbb{N}$. *By a* (ρ, ξ, δ, H)*-linking scheme we mean a procedure for linking a family* (H_n) *of data items together using auxiliary linking items* L_n *and* $\mathcal{L}_r$ *satisfying the recursive formulae*

$$\begin{aligned} L_n &:= H(H_n, L_{n_1}, \dots, L_{n_{\sharp\rho^{-1}(n)}}) \qquad \text{if } n \notin \xi(\mathbb{N}) \\ \mathcal{L}_r &:= L_{\xi(r)} = H(\mathcal{H}_r, \mathcal{L}_{r_1}, \dots, \mathcal{L}_{r_{\sharp\delta^{-1}(r)}}) \\ \mathcal{H}_r &:= H(H_m, L_{m_1}, \dots, L_{m_{\sharp\rho^{-1}(n)}}) \ , \end{aligned}$$

where $m = \xi(r)$, $\rho^{-1}(n) = \{m_1, \dots, m_{\sharp\rho^{-1}(n)}\}$ $(m_1 \geq \dots \geq m_{\sharp\rho^{-1}(n)})$ *and* $\delta^{-1}(r) = \{r_1, \dots, r_{\sharp\delta^{-1}(r)}\}$ $(r_1 \geq \dots \geq r_{\sharp\delta^{-1}(r)})$.

The values $\mathcal{L}_r$ are also referred to as the *time-stamps for rounds.* Note that the time-stamps requested from the TSS during the verification protocol should belong to the set of time-stamps for rounds because only these time-stamps are available in the time-stamping server.

Definition 4. *A (ρ, ξ, δ, H)-linking scheme is said to be an* Accumulated Linking Scheme (ALS) *with* rank m, *if*

1. *If $\xi(r) < n \leq \xi(r+1)$ then $\rho^{-1}(n) \subset [\xi(r), \xi(r+1)] \cup \xi(\mathbb{N})$;*
2. *$\xi(r+1) - \xi(r) \geq m$.*

We say that a (ρ, H)-linking scheme enables accumulated time-stamping *if for arbitrary positive m there exists ξ, such that the (ρ, ξ, ρ, H)-scheme is an ALS with rank m.*

If the linking scheme used enables accumulated time-stamping, the duration of the rounds can be flexibly enlarged in order to guarantee that only a negligible fraction of the time-stamps are kept in the memory of the time-stamping server.

Let n be the total number of time-stamps issued till the moment of the current run of stamping/verification protocol. The feasibility requirements can be summarized with the following:

1. The number of the evaluations of the hash function during the verification protocol should be $O(\log n)$. In particular, the number of time-stamps examined during a single run of the verification protocol should be $O(\log n)$;
2. There should be a conveniently small upper bound to the length of rounds, whereas the clients want to get their time-stamps in reasonable time. It seems to be sensible to require that the stamping protocol of the n-th document must terminate before the TSS has received additional $O(\log n)$ time-stamp requests. In real applications it is desirable for the average length of rounds to be constant (this would guarantee that for an arbitrary constant c there would be only negligible fraction of rounds with length greater than c).
3. The size of an individual time-stamp should be small.

As we will show later (Thm. 2), there is a trade-off between these quantities. In Sect. 5 and the following sections we present an improvement of the scheme of Sect. 4.

4 First Version of Our System: Linear Linking

For pedagogical reasons, we outline the protocols and the basic organizational principles of our system using the linear linking scheme. This scheme fulfills all the trust requirements but is impractical. Further, the described scheme is significantly improved by replacing the linear scheme with a binary linking scheme.

Let the number M of time-stamps per round be a constant known to the participants (clients) and all the data items X_n be of fixed size. Therefore, in the case of the linear linking scheme, the time-stamp for the r-th round has a number $\xi_r = M \cdot r$.

4.1 Role of the TSS

The TSS maintains the following three databases:

1. the database $\mathcal{D}_{\mathrm{c}}$ of the time-stamps of the current round.
2. the database $\mathcal{D}_{\mathrm{p}}$ of the time-stamps of the previous round.
3. the database $\mathcal{D}_{\mathrm{r}}$ of the time-stamps for rounds.

These databases are considered to be on-line in the sense that any client can make requests into them at any moment. The fourth database (the complete data-base of time-stamps) is also stored but not on-line (it may be stored into an archive of CD-s). Requests to this database are possible, but costly (e.g., requiring human interaction). After the end of each round, the time-stamps in $\mathcal{D}_{\mathrm{p}}$ are stored to a separate CD (this process *may* be audited). Thereafter, $\mathcal{D}_{\mathrm{p}}$ is emptied. The time-stamp R_r for the current round is computed, added to $\mathcal{D}_{\mathrm{r}}$ and published in a newspaper (two processes which *should* be audited). The database $\mathcal{D}_{\mathrm{c}}$ is copied into $\mathcal{D}_{\mathrm{p}}$ and a new database $\mathcal{D}_{\mathrm{c}}$ is created.

4.2 Stamping Protocol

Suppose, the current round number is r.

1. Client sends X_n to the TSS.
2. The TSS finds $H_n = H(n, X_n)$ and $L_n = (H_n, L_{n-1})$, and adds the pair (H_n, L_n) to $\mathcal{D}_{\mathrm{c}}$.
3. The TSS signs the pair (n, L_n) and sends $(n, L_n, \mathrm{sig}_{\mathrm{TSS}}(n, L_n))$ back to the client.
4. The TSS sends the tuple $\mathrm{head}(n) = (H_{n-1}, H_{n-2}, \ldots, H_{\xi_{r-1}+1})$ to the client.
5. The client verifies the signature of TSS and checks, whether

$$H(H_n, H(H_{n-1}, \ldots H(H_{\xi_{r-1}+1}, L_{\xi_{r-1}}) \ldots)) = L_n \ , \tag{2}$$

   where the true values L_{ξ_i} can be found either from the newspaper or by requesting for their values from the on-line database $\mathcal{D}_{\mathrm{r}}$ of the TSS.

After the M requests have been answered the TSS finishes the round by finding $L_{\xi_r} = H(H'_{\xi_r}, L_{\xi_{r-1}})$ (where $H'_{\xi_r} = H(H_{\xi_r}, L_{\xi_r - 1})$) and publishing L_{ξ_r} and his public key K_{TSS} in the newspaper. The client may now continue, during a limited period, the protocol in order to get the complete individual time-stamp for X_n.

6. The client sends a request to the TSS.
7. Let $\mathrm{tail}(n) = (H_{\xi_r - 1}, H_{\xi_r - 2}, \ldots, H_{n+2}, H_{n+1})$ The TSS answers by sending $(\mathrm{tail}(n), \mathrm{sig}_{\mathrm{TSS}}(\mathrm{tail}(n)))$ to the client.
8. The client checks whether

$$L_{\xi_r} = H(H_{\xi_r - 1}, H(H_{\xi_r - 2}, \ldots H(H_{n+2}, H(H_{n+1}, L_n)) \ldots)) \ . \tag{3}$$

Definition 5. *The* complete individual time-stamp s_n *for the* n*-th document is*

$$s_n := (\mathrm{tail}(n), \mathrm{head}(n), n, L_n, \mathrm{sig}_{\mathrm{TSS}}(n, L_n)) \ .$$

Every client who is interested in the legal use of a time-stamp, should validate it during the stamping protocol. In a relatively short period between the 1st and the 3rd step and between the 4th and 6th step, the signature key of TSS is trusted to authenticate him and therefore, his signature on an invalid head(n) or tail(n) can be used as an evidence in the court. But the client is responsible for doing it when the signature key of TSS can still be trusted. Later, the signature of TSS may become unreliable and therefore only the one-way properties can be used.

4.3 Verification Protocol

Let $r(n)$ denote the round where s_n was issued. Assume, the verifier has two time-stamped documents (X_m, s_m) and (X_n, s_n) where $m < n$.

1. The verifier checks the validity of the equations (2) and (3) for both time-stamps.
2. If $r(m) = r(n)$ then the data hold in tail(m) and head(n) will be enough to check whether

$$L_n = H(H_n, H(H_{n-1}, \ldots H(H_{m+1}, L_m) \ldots)) \ .$$

3. If $r(m) < r(n)$, the verifier sends a request to the TSS.
4. The TSS answers by sending the tuple

$$v_{mn} = (H'_{\xi_{r(n)-1}}, H'_{\xi_{r(n)}-2}, \ldots, H'_{\xi_{r(m)}})$$

   and the signature $\mathrm{sig}_{\mathrm{TSS}}(v_{mn})$ to the verifier.
5. The verifier validates the signature, finds $L_{\xi_{r(m)}}$ using (3), finds $L_{r(n)-1}$ using the formula

$$L_{r(n)-1} = H(H'_{\xi_{r(n)-1}}, H(H'_{\xi_{r(n)-2}}, \ldots H(H'_{\xi_{r(m)}}, L_{\xi_{r(m)}}) \ldots))$$

   and finally, compares the value of L_n in s_n with the value given by (2).

4.4 Audit Protocol

Because of the possible legal importance of the time-stamps issued by the TSS, there should be some mechanism to audit TSS. One easy way to do it is to periodically ask time-stamps from the TSS and verify them. If these time-stamps are linked inconsistently (i.e., the Eq. (2) and (3) hold for both time-stamps but the verification protocol fails), the TSS can be proven to be guilty. Also, there has to be a mechanism for the TSS to prove that he has not issued a certain time-stamp S in a certain round r. This can be done if the TSS presents all the time-stamps issued during the r-th round, shows that S is not among them and that the time-stamp for the r-th round, found by using these time-stamps and the linking rules, coincides with the published time-stamp.

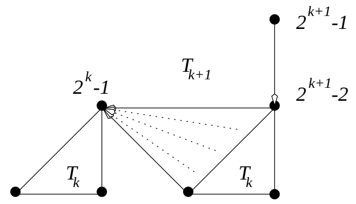
5 Binary Linking Schemes

In the current section we give a construction of a practical linking scheme with logarithmic upper bound to the length of the shortest verifying chain between any two time-stamps.

Definition 6. *Let f and g be functions from $\mathbb{N}$ to $\mathbb{N}$ satisfying the condition $f(n) \leq g(n) < n$ for any n. A $(f, g, H)-$*binary linking scheme (BLS) *is a (ρ, H)-linking scheme where for any n, $\rho^{-1}(n) = \{f(n), g(n)\}$. In order to guarantee the existence of a verifying chain between arbitrary x and y, we have to take $g(n) := n - 1$. In those cases we omit $n - 1$ and talk just about a (f, H)-BLS.*

A binary linking scheme can alternatively be defined as a directed countable graph which is connected, contains no cycles and where all the vertices have two outgoing edges (links). Let us construct an infinite family of such graphs T_k in the following way:

1. T_1 consists of a single vertex which is labeled with the number 1. This vertex is both the *source* and the *sink* of the graph T_1.
2. Let T_k be already constructed. Its sink is labeled by $2^k - 1$. The graph T_{k+1} consists of two copies of T_k, where the sink of the second copy is linked to the source of the first copy, and an additional vertex labeled by $2^{k+1} - 1$ which is linked to the source of the second copy. Labels of the second copy are increased by $2^k - 1$. The *sink* of T_{k+1} is equal to the sink of the first copy, the *source* of T_{k+1} is equal to the vertex labeled by $2^{k+1} - 1$.
Thereafter, link all the vertices of the second copy which have less than two outgoing links, to the source of the first copy. Note that there is now a double link from the sink of the second copy to the source of the first copy.



The sequence (T_k) defines a binary linking scheme with the vertices labeled by natural numbers which contains each scheme T_k as its initial segment. After the construction of this binary linking scheme, add links from the sources of any such initial segment to a special vertex labeled by 0 (Fig. 2). Here (see also Rem. 1), $f(n) = n - 2^{h(n)} + 1$, where $h(n)$ is given recursively by the equation

$$h(n) = \begin{cases} k \ , & \text{if } n = 2^k - 1 \ , \\ h(n + 1 - 2^{k-1}) \ , & \text{if } 2^{k-1} \leq n < 2^k - 1 \ . \end{cases}$$

Theorem 1. *Let $\ell(a, b)$ be the length of the shortest verifying chain from b to a. If $k > 2$ and $0 < a \leq b < 2^k$ then $\ell(a, b) \leq 3k - 5$. (See Appendix A)*

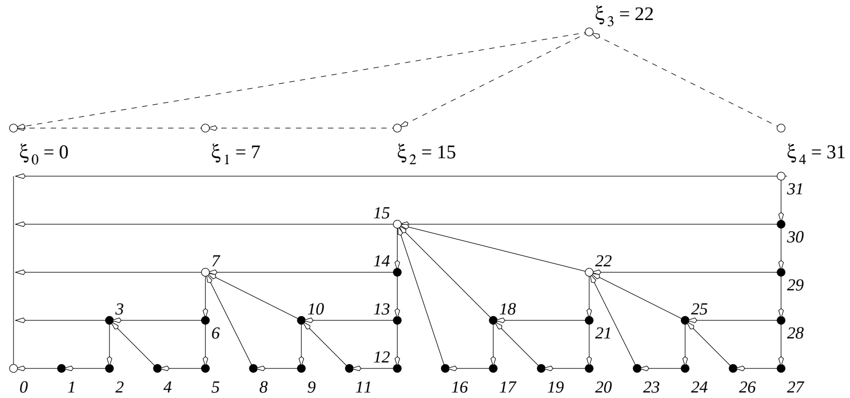


Fig. 2. The ALS structure built on T_5 with $m = 7$.

In Sect. 4 we presented an outline of a time-stamping system that fulfills our trust requirements. In the next we show how to make this system feasible by using a BLS.

In order to issue the individual time-stamp for the n-th document, the TSS has to find the shortest verifying chains between $\xi_{r(n)-1}$ and n and between n and $\xi_{r(n)}$. The n-th individual time-stamp consists of the minimal amount of data (Sect. 4.2) necessary to verify the mutual one-way dependencies between all L_j which lay on these chains. It can be shown that if f satisfies the implication

$$m > n \Rightarrow (f(m) \leq f(n) \vee f(m) \geq n) \tag{4}$$

then (f, H) enables accumulated time-stamping (the proof has been omitted because of its technicality). In particular, the binary linking scheme described in Sect. 5 enables accumulated time-stamping. For a fixed m let $k := \lceil \log_2 m \rceil$, $\xi_0 := 0$, $\xi_1 := 2^k - 1$ (the source of T_k) and for arbitrary $i > 1$,

$$\xi(i) := \begin{cases} \xi_{2^j} + \xi_{i-2^j} \ , & \text{if } i \neq 2^j \\ 2 \cdot \xi_{i/2} + 1 \ , & \text{if } i = 2^j \ , \end{cases}$$

where $j := \lfloor \log_2 i \rfloor$. The length of the n-th time-stamp in this scheme does not exceed $2 \cdot 3 \cdot \log(n) \cdot \chi$ bits, where χ is the output size of the hash function H.

The maximum length of rounds grows proportionally to $O(\log n)$. However, the average length of rounds is constant and therefore it is practical to publish the time-stamps for rounds after constant units of time. This can be achieved easily with the following procedure. If the "deadline" for round is approaching and there are still q time-stamps not issued yet, assign random values to the remaining data items H_n.

Remark 1. Denote by $\operatorname{ord} n$ the greatest power of 2 dividing n. In the ALS presented above, it is reasonable to label time-stamps in the lexicographical order with pairs (n,p), where $0 \le p \le \operatorname{ord} n$ and $n > 0$. Then,

$$f(n,p) := \begin{cases} (0,p) \ , & n = 2^p \\ (n-2^p, \operatorname{ord}(n-2^p)) \ , & \text{otherwise} \end{cases}$$

and $g(n,p) := (n, p-1)$ if $p > 0$ and $g(n,0) := (n-1, \operatorname{ord}(n-1))$. Also, the formulas of ξ_i will simplify: in this case, $\xi(i) := (2^{k-1}i, k-1+\operatorname{ord} i)$, for $i \ge 1$.

It is easy to show that for each n and m the shortest verifying chain between n and m is uniquely defined. The data v_{mn} necessary to verify the one-way dependence is computed by the procedure $\mathit{TSData}(m,n)$:

proc $\mathit{TSData}(m,n) \ \equiv$
  $\mathit{Data} := \mathit{nil}$
  while $n > m$ **do**
    $\mathit{Data} := \mathit{append}(\mathit{Data}, H_n)$
    if $f(n) \ne n-1 \ \wedge \ f(n) \ge m$
      then $\mathit{Data} := \mathit{append}(\mathit{Data}, L_{n-1})$; $n := f(n)$
      else $\mathit{Data} := \mathit{append}(\mathit{Data}, L_{f(n)})$; $n := n-1$
    fi
  od.

Here, $\operatorname{head}(n) := \mathit{TSData}(\xi_{r(n-1)}, n)$ and $\operatorname{tail}(n) := \mathit{TSData}(n, \xi_{r(n)})$.

Example 1. Let $\xi_0 = 0$ and $\xi_1 = 15$ (Fig. 2). In order to compute the fourth and the tenth time-stamps we need

$$\begin{aligned} \operatorname{tail}(10) &:= (H_{15}, L_0, H_{14}, L_7, H_{13}, L_{12}) \ , \\ \operatorname{head}(10) &:= (H_{10}, L_9, H_7, L_6) \ , \\ \operatorname{tail}(4) &:= (H_{15}, L_0, H_{14}, L_{13}, H_7, L_0, H_6, L_3, H_5, L_4) \ , \\ \operatorname{head}(4) &:= (H_4, L_3, H_3, L_2) \ . \end{aligned}$$

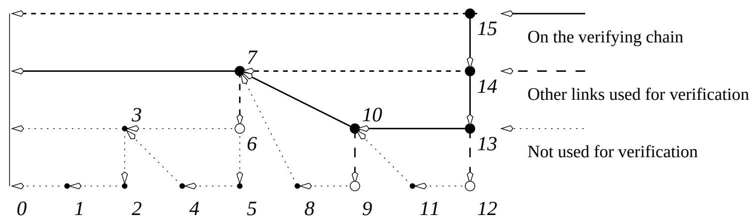


Fig. 3. The time-stamp of X_{10} in the proposed system.

Let (f,H) be a BLS satisfying the implication (4). Let $x < y < z < w$ and C_1 , C_2 be verifying chains from z to x and w to y respectively. It is obvious that

C_1 and C_2 have a common element. Thus, if $m < n$ then the verifying chains $\text{tail}(m)$ and $\text{head}(n)$ have a common element c which implies the existence of a verifying chain

$$(m = n_0, n_1, \ldots, n_{i-1}, n_i = c, n_{i+1}, \ldots, n_{\ell-1}, n_\ell = n) \ .$$

This chain can be found by a simple algorithm and is of logarithmic length. Let $r(m)$ denote the round into which m belongs. The proof of the last claim for the case $r(m) = r(n)$ is given in Appendix A. If m and n belong to different rounds, the verifying is straightforward, because of the similar structure of the second layer of links. The verifying chain from n to m is of the form

$$(m, \ldots, m', \xi_{r(m)}, \ldots, \xi_{r(n)-1}, n', \ldots, n) \ ,$$

where the number of ξ_j-s is logarithmic due to the fact that the time-stamps for rounds are linked together in a way similar to the linking of all time-stamps (Fig. 2). The length of the sequences $(m, \ldots, m')$ and $(n', \ldots, n)$ is also logarithmic (Appendix A).

Example 2. For the chains given in Example 1, the common element is 7 and the verifying chain between 4 and 10 is $(4, 5, 6, 7, 10)$.

Corollary 1. *Due to the similarity between the verification and the stamping procedure, for an arbitrary pair of time-stamped documents the number of steps executed (and therefore, also the number of time-stamps examined) during a single run of the verification protocol is* $O(\log n)$.

6 Optimality

Our solution meets asymptotically the feasibility requirements, but could these requirements be refined? Mostly not, an insight into this is given below. Namely, we show that for any linking scheme there does not exist a time-stamping solution where (1) the length of the time-stamps is $O(\log n)$, (2) for any m and n there exists a verifying chain between m and n with the length $O(\log n)$ that is completely contained in the union $S(m) \cup S(n)$ of the corresponding individual time-stamps and (3) the stamping protocol will end in a logarithmic time.

We prove this under the assumptions (1) that an individual time-stamp is a subset of $\mathbb{N}$ and (2) that the size of a time-stamp is proportional to the size of $\sharp S(n) + \sharp\rho^{-1}(S(n)) = O(\sharp\rho^{-1}(S(n)))$ (holds if the transitive closure ρ^* of ρ coincides with the natural order $<$, i.e, the time-stamp $S(n)$ consists of $\text{tail}(n)$ and $\text{head}(n)$)).

Theorem 2. *Let ρ be a binary relation on $\mathbb{N}$ satisfying $\rho^* = <$. There does not exist a function $S : \mathbb{N} \to 2^{\mathbb{N}}$ such that*

1. $|\ \rho^{-1}\ (S(n))| < c_1 \log n$ *for some* c_1, *for any* n.

2. *For every* m *and* n *there exists a* ρ*-chain* $(m = m_1, m_2, \ldots, m_k = n)$ *where* $m_i \in S(m) \cup S(n)$ *(that is, the number of time-stamps necessary to examine during the verification protocol is greater than 2).*
3. *For any* n*,* $\max\{S(n)\} - n \leq c_2 \log n$ *for some constant* c_2*.*

Proof. Assume that there exists such S. Let n be a sufficiently large positive integer. For a $m \in \mathbb{N}$ let $\Phi(m) := [m, m + \lceil c_2 \log m \rceil]$. The intervals $\Phi(1 + ic_2 \log n)$, $i \in 0, \ldots, \lfloor \frac{n - c_2 \log n - 2}{c_2 \log n} \rfloor$ do not intersect.

Let $m < n - c_2 \log n - 1$. In this case $\lceil m + c_2 \log m \rceil < n$. As the set $S(m) \cup S(n)$ contains a ρ-chain from m to n there should exist such $m_1 \in \Phi(m)$ and $n_1 \in \mathbf{S}_(n)$ on this chain that $m_1 \, \rho \, n_1$. Thus, for every $m < n - c_2 \log n - 1$ the set $\Phi(m) \cap \rho^{-1}(S(n)))$ is nonempty. Hence, the set $\rho^{-1}(S(n))$ has at least $\lfloor \frac{n - c_2 \log n - 2}{c_2 \log n} \rfloor = \Theta(n / \log n)$ elements. A contradiction with Condition 1. □

The Thm. 2 can be straightforwardly generalized to claim that the number of examined time-stamps must be greater than any fixed constant.

7 Acknowledgements and Further Work

We would like to thank Stuart Haber for his patience, without his help this paper would have been totally unreadable. We are grateful to Philip Hawkes and anonymous referees for valuable remarks.

The reasoning about the time-stamping procedures creates the need for a formal apparatus capable to prove the security of time-stamping protocols, in a way similar to how the BAN-style logics [BAN89] are used for reasoning about ordinary authentication protocols. The renewing protocols and technical specifications need to be elaborated.

References

[BAN89] Michael Burrows, Martín Abadi, and Roger Needham. A Logic of Authentication. SRC Research Reports 39, DEC's System Research Center, February 1989.

[BdM91] Josh Benaloh and Michael de Mare. Efficient Broadcast time-stamping. Technical Report 1, Clarkson University Department of Mathematics and Computer Science, August 1991.

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

[Haw88] Stephen W. Hawking. *A Brief History of Time: From the Big Bang to Black Holes.* Bantam Books, April 1988.

[HS91] Stuart Haber and W.-Scott Stornetta. How to Time-Stamp a Digital Document. *Journal of Cryptology*, 3(2):99–111, 1991.

[HS97] Stuart Haber and W.-Scott Stornetta. Secure Names for Bit-Strings. In *Proceedings of the 4th ACM Conference on Computer and Communications Security*, pages 28–35, April 1997.

[Jus98] Mike Just. Some Timestamping Protocol Failures. In *Internet Society Symposium on Network and Distributed System Security*, 1998. Available at `http://www.scs.carleton.ca/~just/`.

[MOV96] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.

[MQ97] Henry Massias and Jean Jacques Quisquater. Time and Cryptography. Technical report, Université catholique de Louvain, March 1997. TIMESEC Technical Report WP1.

A Proof of Theorem 1

We will prove an upper bound for the length of the verifying chain for the linking scheme described in Sect. 5. Let $e_k = 2^k - 1$, i.e. e_k is the number of the last vertex of T_k. To simplify the proof we add the vertex 0 to the scheme and link it with all the vertices that have less than two outgoing links. These are exactly the vertices e_k. Let $\ell(a,b)$ denote the length of the shortest path between a and b. The equations $\ell(0,e_k) = 1$, $\ell(e_{k-1},e_k) = 2$ and $e_k - e_{k-1} = e_{k-1} + 1$ follow immediately from the definition.

Lemma 1. *If* $0 < a \le e_k \le b$ *then* $\ell(a,b) = \ell(a,e_k) + \ell(e_k,b)$. *If* $e_{k-1} \le a < e_k$ *then* $\ell(a,e_k) = \ell(a,e_k-1) + \ell(e_k-1,e_k)$.

The claims above follow immediately from the structural properties of the linking scheme.

Lemma 2. *If* $e_{k-1} \le a \le b < e_k$ *then* $\ell(a,b) = \ell(a-e_{k-1}, b-e_{k-1})$.

Proof. This follows from the construction of T_k from the two copies of T_{k-1}. Here a and b are vertices in the second copy of T_{k-1} (or the last vertex of the first copy), and $a - e_{k-1}$ and $b - e_{k-1}$ are the same vertices in the first copy of T_{k-1} (or the vertex 0). □

Lemma 3. *If* $0 \le a < e_k$ *then* $\ell(0,a) \le k$.

Proof. Induction on k.
Base: $k = 1$. Then $a = 0$ and $\ell(0,a) = 0 < k$.
Step: $k > 1$. Observe the following cases:

– If $0 \le a < e_{k-1}$ then the induction assumption gives $\ell(0,a) \le k-1 < k$.
– If $e_{k-1} \le a < e_k$ then $\ell(0,a) = \ell(0,e_{k-1}) + \ell(e_{k-1},a) = 1 + \ell(0,a-e_{k-1})$ by Lemma 2. Observe the following cases:
  - $a = e_k - 1$. Then $\ell(0,a) = 1 + \ell(0,a-e_{k-1}) = 1 + \ell(0,e_{k-1}) = 2 \le k$.
  - $a < e_k - 1$. Then $\ell(0,a) = 1 + \ell(0,a-e_{k-1}) \le 1 + (k-1) = k$ by induction assumption.

□

Lemma 4. *If* $0 < a \le e_k$ *then* $\ell(a,e_k) \le 2(k-1)$.

Proof. Induction on k.
Base: $k = 1$. Then $a = 1$ and $\ell(a, e_k) = \ell(1, 1) = 0 = 2(k - 1)$.
Step: $k > 1$. Observe the following cases:

- If $0 < a \leq e_{k-1}$ then $\ell(a, e_k) = \ell(a, e_{k-1}) + \ell(e_{k-1}, e_k) \leq 2(k-2) + 2 = 2(k-1)$ by induction assumption.
- If $e_{k-1} < a \leq e_k$ then observe the following cases:
  - $a = e_k$. Then $\ell(a, e_k) = 0 \leq 2(k - 1)$.
  - $a < e_k$. Then $\ell(a, e_k) = \ell(a, e_k - 1) + \ell(e_k - 1, e_k) = \ell(a - e_{k-1}, e_{k-1}) + 1$ by the Lemma 2. Induction assumption now gives $\ell(a, e_k) = \ell(a - e_{k-1}, e_{k-1}) + 1 \leq 2(k-2) + 1 < 2(k-1)$.

□

Proof (Theorem 1). Induction on k.
Base: $k = 3$. In this case one can directly verify that $\ell(a, b) \leq 4$.
Step: $k > 3$. Observe the following cases:

- If $0 < a \leq b \leq e_{k-1}$ then the induction assumption gives us $\ell(a, b) \leq 3(k-1) - 5 < 3k - 5$.
- If $0 < a \leq e_{k-1} < b \leq e_k$ then $\ell(a, b) = \ell(a, e_{k-1}) + \ell(e_{k-1}, b) \leq 2(k-2) + \ell(e_{k-1}, b)$ by the Lemma 4. The following cases are possible:
  - $b = e_k$. Then $\ell(e_{k-1}, b) = 2 < k - 1$.
  - $b = e_k - 1$. Then $\ell(e_{k-1}, b) = 1 < k - 1$.
  - $b < e_k - 1$. Then the lemmas 2 and 3 give
    $\ell(e_{k-1}, b) = \ell(0, b - e_{k-1}) \leq k - 1$.

  Thus $\ell(a, b) \leq 2(k-2) + k - 1 = 3k - 5$.
- If $e_{k-1} < a \leq b \leq e_k$ then observe the following cases:
  - $b = e_k$. Then $\ell(a, b) = \ell(a, e_k) \leq 2(k-1) < 3k - 5$ by Lemma 4.
  - $b < e_k$. Then $\ell(a, b) = \ell(a - e_{k-1}, b - e_{k-1}) \leq 3(k-1) + 5 < 3k - 5$ by Lemma 2 and induction assumption.

□

As $\lceil \log b \rceil = k$ iff $e_{k-1} + 1 < b \leq e_k + 1$ we get $k < \lceil \log b \rceil + 1$ and thus

$$\ell(a, b) \leq 3\lceil \log b \rceil - 2 \ .$$