Optimally Efficient Accountable Time-Stamping

Ahto Buldas*[1], Helger Lipmaa*[1], and Berry Schoenmakers**[2]

[1] Küberneetika AS, Akadeemia tee 21, 12618 Tallinn, Estonia
{ahtbu,helger}@cyber.ee
[2] Dept. of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
berry@win.tue.nl

Abstract. Efficient secure time-stamping schemes employ a 2-level approach in which the time-stamping service operates in rounds. We say that a time-stamping service is *accountable* if if it makes the TSA and other authorities accountable for their actions by enabling a principal to detect and later prove to a judge any frauds, including attempts to reorder time-stamps from the *same* round. We investigate the paradigm of time-stamping services based on simply connected graphs, and propose a simple, yet optimal, accountable time-stamping service, using what we call threaded tree schemes. We improve upon the previously best scheme by Buldas and Laud by reducing the size of a time stamp by a factor of about 3.786 and show that our construction is optimal in a strict sense. The new protocols also increase the trustworthiness of the publication process, which takes place at the end of each round.

1 Introduction

A time-stamping service is a collection of protocols providing long-term authentication of digital documents together with the moment in time at which they were submitted for authentication. Time-stamping is expected to become an integral part of the legal framework for digital documents, as it enables one to prove in court that a document existed at some given point in time. In addition, time-stamping helps to significantly lower the level of trust currently required of a public-key infrastructure by making it possible to prove that a document was signed before the revocation act of the corresponding signature key was stamped. As such, one will frequently depend on time-stamping to resolve the status of documents and corresponding signature keys, which motivates the need for a clear understanding of the security and efficiency requirements of time-stamping.

In the context of time-stamping it is common to regard time as a relative notion. We say that a (time-)stamping service provides *relative temporal authentication* if one is able to decide which stamp has been issued first for each pair

* Supported partially by Estonian Science Foundation grant 3742.
** Part of this work was done while visiting Küberneetika AS.

of time-stamps. Following Haber and Stornetta [HS90,HS91], a cryptographic way to achieve such a relative order for the documents is to create dependencies between the time certificates attached to the documents, where later certificates are obtained by applying a collision-resistant function to earlier ones. A short description of the existing time-stamping schemes is given in Section 3.

We consider time-stamping services that operate in rounds, where clients will issue requests to a central time-stamping authority (TSA) which in turn hands over the cumulative round stamp to the publication authority (PA) so the latter may publish it. In order that the time-stamps are meaningful in court, it is required that one is able to actually *prove* that a relative temporal ordering holds for any pair of time-stamps (even for time-stamps from the same round), or to prove that an authority deviated from the protocols (be it accidentally or intentionally). It is important to note that no single player in the scheme (including the TSA and PA) is required to store all of the time stamps for the rounds in which it takes part.

A time-stamping service is *accountable*, if it makes the TSA and other authorities accountable for their actions by enabling a principal to detect and later prove to a judge any frauds, including attempts to reorder time-stamps from the same round. Moreover, if a honest party followed the protocol but is still accused of forgeries, she can explicitly disavow any false accusations. Section 4 addresses the security objectives in more depth, where we will rely on notions such as simply connected graphs and authentication graphs, which are introduced in Section 2.

In Sections 5–7 we present our main contributions. In Section 5 we introduce a notion of general ϕ-schemes, based on simply connected graphs. In particular, we define two instances of it: anti-monotone schemes [BLLV98,BL98], and our *threaded tree schemes*. We show that while the latter scheme is not anti-monotone it satisfies the same security objectives as achieved by anti-monotone schemes. The important difference is that the size of the time certificates becomes approximately 3.786 times shorter than for the optimal anti-monotone schemes of [BL98], and overall the scheme also becomes simpler.

Section 6 presents an optimized set of time-stamping protocols using simply connected graphs. As a novel feature, we present a *publication protocol*, in addition to the *stamping protocol* and the *stamp completion protocol* as described for other time-stamping services. Finally, in Section 7, we prove that under general assumptions our scheme based on threaded tree schemes is tightly optimal (i.e., optimality is not just asymptotic).

2 Preliminaries

2.1 Cryptographic Primitives

The basic cryptographic tools used in a time-stamping service are hash functions and digital signatures. We let $h : \{0,1\}^* \to \{0,1\}^k$ denote some fixed *collision-resistant hash function*, where k is a security parameter. Furthermore, we let

$\mathsf{sig}_A(M)$ denote A's signature on message M. Since during stamping protocols the principals must sign arbitrary data, we require the signature scheme to be existentially unforgeable against adaptive chosen message attack.

2.2 Authentication Graphs

Let $G = (V, E)$ be a directed acyclic graph such that $V = \{1, 2, \ldots, |V|\}$ and $v < w$ whenever $(v, w) \in E$, i.e., the vertices of G are topologically sorted (since G is a directed acyclic graph, such a numbering always exists). For any $W \subseteq V$ we define $E(W) = \{v : (\exists w \in W)(w, v) \in E\}$ and $E^{-1}(W) = \{v : (\exists w \in W)(v, w) \in E\}$. We assume that $v = |V|$ is the unique vertex with $E(\{v\}) = \emptyset$, which we call the *root* of G. Finally, we let $\mathcal{S}(G) = \{v \in V : E^{-1}(\{v\}) = \emptyset\}$ be the set of *source* vertices of G.

Graph G is called *simply connected* if there is a directed path between each pair of vertices v and w (going either from v to w or from w to v). For topologically sorted G it is equivalent to state that $(v, v+1) \in E$ for all v, $1 \leq v < |V|$. For each pair $v < w$ of vertices of a simply connected graph G we define the *distance* $d(v, w)$ as the length of the shortest path from v to w.

We say that $U \subseteq V$ is *computable* from $W \subseteq V$ if either (1) $U \subseteq W$ or (2) $U \cap \mathcal{S}(G) \subseteq W$ and $E^{-1}(U \setminus W)$ is computable from W. For any subset $W \subseteq V \setminus \mathcal{S}(G)$, we say that the set $\mathsf{A}_G(W) = E^{-1}(W) \setminus W$ is the *authenticator* of W. Clearly, W is computable from $\mathsf{A}_G(W)$.

Recall that h denotes a collision-resistant hash function. As a generalization of Merkle's authentication trees [Mer80], we introduce *authentication graphs* G as labeled directed acyclic graphs with the labels assigned in the following manner. Each source v of G is labeled by $L_v = H_v$, where H_v is a given string (data element) specific to source v. The label of a non-source vertex v is computed as a function of the labels of its predecessors: $L_v = h(\boldsymbol{L}_{E^{-1}(v)})$. Here $\boldsymbol{L}_S = (L_{v_1}, \ldots, L_{v_k})$, where $v_1, \ldots, v_k$ are the elements of S in strictly increasing order. We often identify the labels of an authentication graph with its vertices and say that a vertex is computed from its predecessors rather than the label of a vertex is computed from the labels of its predecessors.

3 State of the Art

Modern time-stamping services provide *relative temporal authentication*, where one verifies the relative order of stamp issuing, given any two time stamps. In all such schemes the time certificate of a later issued stamp is dependent on earlier stamps (a paradigm first proposed in [HS90]).

Most of the proposed schemes are built upon authentication graphs. *Linear linking schemes* [HS90] use a chain as the underlying graph. This solution is clearly impractical, since certificate length increases linear with the number of stamps issued so far. In the worst case, verification of a single time certificate takes as much time as it takes the TSA to compute the stamps for all documents issued so far.

Tree schemes [BdM91,BHS92] (using a tree as the underlying graph) were motivated by the desire to minimize the storage and communication requirements. Since trees are not simply connected, this type of schemes do not provide relative temporal authentication within rounds. To overcome this, the stamping procedure was divided into rounds, so that the cumulative round stamp of every round was published in an authenticated medium. Additionally, the duration of rounds was assumed to be small enough to think about the stamping requests submitted during the same round as simultaneous. Unfortunately, bounding the round lengths reduces the compression effect.

Anti-monotone schemes [BLLV98,BL98] combine the simply connectedness of linear schemes and efficiency of tree schemes, therefore sharing the best features of these schemes. Anti-monotonicity makes the schemes slightly less efficient (and also, more difficult to understand and implement) than the tree schemes.

Accumulator schemes [BdM93] (the only known viable alternative to the graph based schemes) further reduce the storage requirements, by introducing a trapdoor into the scheme. However, accumulator schemes share all drawbacks of tree schemes. They have also another principal weakness: one has to trust the coalition of parties who generated the trapdoor information. Hence, usage of such schemes would directly violate the objective of reducing trust in stamping services. To date, there is no known efficient construction of trapdoorless accumulators (see [San99] for recent work in this area).

4 Time-Stamping Objectives

A *stamping service* consists of a set of principals with the time-stamping authority (TSA) and the publication authority (PA) together with a quadruple (S, C, V, P) of protocols. The *stamping protocol* S is used by a participant to hand over a message to the TSA for time-stamping. During the *stamp completion protocol* C a participant obtains a time certificate from the TSA. The *verification algorithm* V is used by a principal having two time certificates to verify the temporal order of the corresponding stamping events. The *publication protocol* P is used by a TSA to handle the round stamp (short fingerprint of the round) to the PA who will publish it on some authenticated and easily accessible medium.

We say a time-stamping service is *accountable* if the next two properties hold whenever there is at least one uncorrupted party (say, one honest client interested in legal value of a particular time stamp) during the creation of stamped information:

- Fraud detection. The service makes the trusted third parties accountable for their actions by enabling a principal to detect and later prove to a judge any frauds affecting the relative ordering between time-stamps.
- Anti-framing. If a party has honestly followed the protocol but is still accused in forgeries, she can explicitly disavow any false accusations.

One of the crucial requirements of an accountable time-stamping service is to withstand *reordering attacks*, where the TSA assigns an earlier stamp to a document submitted later than another document, i.e., they should provide *relative temporal authentication*. While network latency makes it impossible to detect microscale reordering attacks, it should be intractable for the stamping service to forge the order of time stamps for documents submitted in significantly different time moments. An ideal time-stamping service should provide relative temporal authentication between any pair of documents.

Simple time-stamping services (like hash-and-sign and simple tree schemes) not ensuring relative ordering of time stamps within rounds are highly vulnerable to reordering attacks. However, these attacks can partially be eliminated by using the stamping protocols defined in [BLLV98], where directly after the vth stamping request H_v the TSA returns a signed receipt on some value $L_v = h(H_1, \ldots, H_v)$ to the client, where h is a collision-resistant hash function (informally, we say then that L_v is *(one-way) dependent on* H_i, $i = 1, \ldots, v$). If L_v is additionally dependent on L_i, $1 \leq i < v$, the client can use TSA's signatures on the values L_v to prove any frauds.

Unfortunately, this methodology helps only until TSA's signature can be trusted (in the worst case, only until the end of the current round). If all L_v were not authenticated by the round stamp, there would be no way to detect the frauds after TSA's key becomes invalid. Hence, a time-stamping service is accountable only if the round stamp authenticates all the values L_v, where every L_v is dependent on L_{v-1}. Moreover, every L_v should be dependent on the round stamp of the previous round. The resulting structure can be seen as a *dependency graph* that has the round stamp as a label of its root and as a subgraph a simply connected graph with vertices labeled by values L_v, where the edges correspond to applying a collision-resistant hash function, and the sources correspond to the data elements H_v. For efficiency we assume that each L_v is a k-bit output of a fixed hash function h. We additionally assume that L_1 is equal to the stamp for the previous round, since the rounds have to be connected. A time certificate has to contain sufficient data to prove that the mentioned dependencies hold. We shall make these informal notions precise in the next sections.

It is important to realize what time-stamping does and what it does not. For example, there is a type of reordering attack that is often used as an argument against linking schemes. Let $H_1, \ldots, H_n$ be the stamping requests during a round. Before "officially" closing the round, the TSA may issue additional time stamps for $H_n, H_{n-1}, \ldots, H_1$ in reverse order. After that the TSA is able, for any pair of stamps of H_i and H_j, to present proofs for the statements "H_i was stamped before H_j" and "H_j was stamped before H_i". It may seem, therefore, that using linking schemes and signed receipts does not give any advantage compared to simple hash-and-sign scheme.

The following example shows a weakness in this argument (cf. [BLLV98, Section 3.1]). First, a client requests a stamp L_n of a nonce from the TSA. Subsequently, she stamps her signature σ on the actual document X and L_n. The TSA may then issue additional stamps for X and σ. However, no one is able

to back-date new documents relative to σ without cooperating with the signer. Note that this attack is hard to prevent if the underlying graph is not simply connected, and that usually the TSA is trusted not to mount such an attack.

5 Schemes Based on Simply Connected Graphs

5.1 General Construction

The first efficient accountable time-stamping service proposed in [BLLV98]is based on *anti-monotone* schemes. Briefly, the intuition behind this scheme is to define the time certificates in such a way that for any v and w, $v < w$, the union of the vth and wth time certificates *contains* set-theoretically a dependent path between the corresponding stamps L_v and L_w. In what follows we show that up to 3.786 times shorter certificates can be achieved if we instead assume that said union contains only information sufficient to *compute* this path. For this we shall define a general framework that includes in particular anti-monotone schemes and the new optimal threaded authentication schemes.

Let $G = (V, E)$ be a rooted simply connected graph with a single source. Let $\phi : V \to \{\mathsf{red}, \mathsf{black}\}$ be an arbitrary function, such that $\phi(1) = \mathsf{red}$. We regard ϕ as a red/black-coloring of G. The *ϕ-scheme* $[\![G]\!]_\phi$ is constructed from G by adding individual data elements, that is, a new vertex v_ϕ and an edge (v_ϕ, v), to every black vertex v. Then we say that v_ϕ is a *data element connected to vertex* ϕ. We say that $\mathcal{P} : V \to 2^V$ is a *pass-through function* if it maps an arbitrary vertex v to a path $(w, \ldots, v, \ldots, |V|)$, such that $(1, w) \in E$.

Definition 1. *Let $[\![G]\!]_\phi$ be a ϕ-scheme, where $G = (V, E)$, and let $\mathcal{P} : V \to 2^V$ be a pass-through function. Now, let $v \in V$ be a black vertex. We define* $\mathsf{head}(v)$ *to be the set of vertices in $\mathsf{A}_{[\![G]\!]_\phi}(\mathcal{P}(v))$ that can be computed from the set of data elements connected to black vertices $w \leq v$, and we set* $\mathsf{tail}(v) = \mathsf{A}_{[\![G]\!]_\phi}(\mathcal{P}(v)) \setminus \mathsf{head}(v)$. *The* time certificate $\mathsf{Cert}(v)$ *of a source $v \in \mathcal{S}([\![G]\!]_\phi)$ is defined as* $\mathsf{Cert}(v) = (v, \boldsymbol{L}_{\mathsf{head}(v)}, \boldsymbol{L}_{\mathsf{tail}(v)})$.

Recall that every source v of an authentication graph G corresponds to some string H_v. Intuitively, $\mathsf{Cert}(v)$ consists of the minimal amount of data necessary to verify whether v lies on a path between 1 (for which the label is equal to the last round stamp) and $|V|$ (for which the label is equal to the current round stamp). If v is represented by k bits then $|\mathsf{Cert}(v)| = k(|\mathsf{A}_{[\![G]\!]_\phi}(\mathcal{P}(v))| + 1)$.

Jumping ahead a little, in the time-stamping protocols $\mathsf{head}(v)$ corresponds to the maximum subset of $\mathsf{A}_{[\![G]\!]_\phi}(\mathcal{P}(v))$ that the stamping authority knows directly after stamping the vth element and $\mathsf{tail}(v)$ corresponds to the remainder of the authenticator. Since $\boldsymbol{L}_{\mathsf{head}(v)}$ is dependent on all data elements issued thus far, by returning his signature on $\boldsymbol{L}_{\mathsf{head}(v)}$ (or on some value that is one-way dependent on $\boldsymbol{L}_{\mathsf{head}(v)}$) to a client directly after issuing the vth stamp, the stamping authority commits himself to all data elements stamped before the vth element. Transmission of the certificate to a client at the end of the round commits the authority to every data element of this round even if the authority's

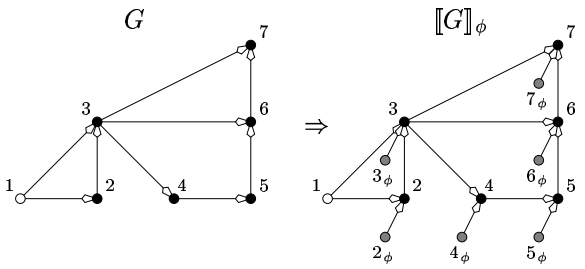

Fig. 1. Construction of $[\![G]\!]_\phi$ for anti-monotone G.

signing key gets compromised later on. (The time-stamping protocols are given in Section 6.)

5.2 Anti-Monotone Schemes

Next we show that even if using exactly the same base graphs as in [BLLV98] one can get a significant improvement over their solution. A simply connected graph $G = (V, E)$ is *anti-monotone* if the in-degree of each vertex is at most 2, and if for all $(v_1, w_1), (v_2, w_2) \in E$ we have that $v_1 < w_2 < w_1$ implies $v_1 \leq v_2$. Let $\phi : V \to \{\mathsf{red}, \mathsf{black}\}$ be such that $\phi(v) = \mathsf{black}$ iff $v \neq 1$. The *anti-monotone ϕ-scheme* $[\![G]\!]_\phi$ is a ϕ-scheme constructed from anti-monotone graph G.

As shown in [BLLV98], in this case $\mathsf{tail}(v) \cap \mathsf{head}(w)$ is nonempty for any $v < w$. Since $\mathsf{Cert}(v)$ (respectively $\mathsf{Cert}(w)$) contains by definition information necessary to compute all the elements in $\boldsymbol{L}_{\mathsf{tail}(v)}$ (respectively in $\boldsymbol{L}_{\mathsf{head}(w)}$), then for any $u \in \mathsf{tail}(v) \cap \mathsf{head}(w)$, the union of $\mathsf{Cert}(v)$ and $\mathsf{Cert}(w)$ contains sufficient information to verify that L_u is dependent on L_v and L_w is dependent on L_u.

Let $d_{pt}(G) = \max_{v \in V}(d(1, v) + d(v, |V|))$. Buldas and Laud proved in [BL98] that for any anti-monotone graph G, $d_{pt}(G) \geq 1.893 \log_2 |\mathcal{S}(G)| + O(1)$. Using the ϕ-scheme $[\![G]\!]_\phi$ we get that the length of a certificate $\mathsf{Cert}(v)$ is equal to $\log_2 |\mathcal{S}(G)| + |\mathsf{head}(v)| + |\mathsf{tail}(v)|$, and hence, $\max_v |\mathsf{Cert}(v)| \geq (k + o(1)) \cdot d_{pt}(G)$. However, the binary linking schemes of [BLLV98,BL98] used redundant certificates such that $\max_v |\mathsf{Cert}(v)| \geq (2k + o(1)) \cdot d_{pt}(G)$ and hence casting the problem in a more general view has helped to reduce the length of the certificates by a factor of two.

5.3 Threaded Authentication Trees

The security of the last construction depends only on the property that there is a vertex u that can be computed from the union of $\mathsf{Cert}(v)$ and $\mathsf{Cert}(w)$. As such element always exists (take the root), the anti-monotonicity property is not necessary for the stamping service to be accountable. Below we present a new construction based on *threaded authentication trees*. Optimality of this scheme in the general case will be proven later.

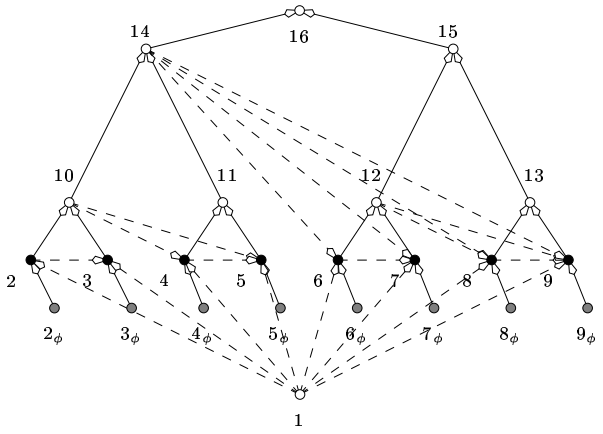

Fig. 2. Threaded tree scheme $[\![W_3]\!]_\phi$.

For Merkle's authentication tree T_d of depth d, the certificate length for a vertex v is $k(d+2)$, where k is the size of the outputs of hash function h. In their stamping system, Benaloh and de Mare [BdM93] added a new vertex 1 with edges $(1, w)$, for any $w \in \mathcal{S}(T_d)$, to Merkle's authentication tree. The resulting graph has certificate length $k(d+3)$, and the vth certificate contains dependency from R_{r-1} to L_v to R_r. We will proceed further by adding edges to the Benaloh-de Mare scheme in such a way that the resulting graph will be simply connected, but *without* enlarging the certificates. The method presented below is not unique in this respect, but has the advantage that it permits an efficient implementation.

Let $T_d = (V, E)$ be the complete binary tree of depth d. We number the vertices of T_d level by level, starting at the lowest level (at depth d) and from left to right so that the leaves get numbers $2, \ldots, 2^d + 1$, and the root gets number $|V| = 2^{d+1}$. Let $\phi : V \to \{\mathsf{red}, \mathsf{black}\}$ color the sources of T_d (i.e., vertices $2 \le v \le 2^d + 1$) black and inner nodes red.

We define the *threaded authentication tree* W_d as the graph built from T_d by applying the next two steps:

1. for each $v \in \mathcal{S}(T_d)$ consider the unique root path $\mathcal{P}(v)$ starting from v and add an edge (w, v) for each left child w of a vertex on $\mathcal{P}(v)$ provided $w \notin \mathcal{P}(v)$, and
2. add a vertex 1 and, for each $v \in \mathcal{S}(T_d)$, add an edge $(1, v)$. Vertex 1 is labeled by $L_1 = R_{r-1}$, where r is number of the current round. We color 1 red, since it corresponds to a data element.

Note that the set of vertices w for which the edge (w, v) was added in the step 1 is equal to $\mathsf{head}(v)$. We define the dth *threaded tree scheme* to be the ϕ-scheme $[\![W_d]\!]_\phi$. Clearly, a threaded authentication tree W_d is simply connected (consider a postorder traversal of the binary tree T_d), but it is not anti-monotone.

INPUT: $(C, L_v, R_{r-1}, R_r, H_v)$, where $C = (v, \chi, \tau)$, $\chi = (\chi_1, \ldots, \chi_m)$, $\tau = (\tau_1, \ldots, \tau_n)$.
Reject if $\chi_1 \neq R_{r-1}$ or $\chi_2 \neq H_v$ or $\tau_n \neq R_r$.
Reject if $L_v \neq h(\chi)$.
$\ell := L_v$, $p := 1$; $q := 3$.
For $i := 0$ to $\log_2 v$ do:
    if $(v-1)$ & $2^i = 0$ then $\ell := h(\ell, \tau_p)$; $p := p+1$ else $\ell := h(\chi_q, \ell)$; $q := q+1$.
Reject if $\ell \neq R_r$. Otherwise accept.

Fig. 3. Certificate verification algorithm VfyCert for threaded tree schemes.

As an example, consider the threaded tree scheme $[\![W_3]\!]_\phi$ of Figure 2. Graph W_3 is simply connected because it contains the postorder traversal $2 \to 3 \to 10 \to 4 \to 5 \to 11 \to 14 \to 6 \to 7 \to 12 \to 8 \to 9 \to 13 \to 15 \to 16$ as a directed path. Then $\mathsf{head}(4) = (1, 4_\phi, 10)$, $\mathsf{tail}(4) = (5, 15)$, $\mathsf{head}(8) = (1, 8_\phi, 12, 14)$, $\mathsf{tail}(8) = (9)$, $\mathsf{Cert}(4) = (4; L_1, L_{4_\phi}, L_{10}; L_5, L_{15})$ and $\mathsf{Cert}(8) = (8; L_1, L_{8_\phi}, L_{12}, L_{14}; L_9)$. The set $\mathsf{Cert}(4) \cup \mathsf{Cert}(8)$ contains sufficient information to first compute and then verify the following path from R_{r-1} to L_4 to L_8 to R_r, where r is the number of the current round:

$$\underbrace{\mathbf{R_{r-1}} = L_1}_{\mathsf{head}(4)\cap\mathsf{head}(8)} \to \mathbf{L_4} = h(\underbrace{R_{r-1}, L_{4_\phi}, L_{10}}_{\mathsf{head}(4)}) \to L_{11} = h(L_4, \underbrace{L_5}_{\mathsf{tail}(4)}) \to$$

$$L_{14} = h(\underbrace{L_{10}}_{\mathsf{head}(4)}, L_{11}) \to \mathbf{L_8} = h(\underbrace{R_{r-1}, L_{8_\phi}, L_{12}, L_{14}}_{\mathsf{head}(8)}) \to$$

$$L_{13} = h(L_8, \underbrace{L_9}_{\mathsf{tail}(8)}) \to L_{15} = h(\underbrace{L_{12}}_{\mathsf{head}(8)}, L_{13}) \to L_{16} = h(\underbrace{L_{14}}_{\mathsf{head}(8)}, \underbrace{L_{15}}_{\mathsf{tail}(4)}) = \mathbf{R_r} \ .$$

Certificate verification algorithm VfyCert (which is used in the verification algorithm) is shown in Figure 3.

For any vertices $v, w \in W_d$, $v \leq w \Rightarrow d(v,w) \leq 2d-1$. Thus, the verification can be done very efficiently, and $|\mathsf{Cert}(v)| = k(d+3)$, for all v. It will be proven in Section 7 that this is also the lower bound. While in the case of schemes used in [BLLV98,BL98], the union of two time certificates contained the whole "proof" of dependency, in the current case the proof itself is not contained in, but can still be computed from the union. That is, intuitively, the main source of the redundancy in the anti-monotone schemes compared to the new scheme. We think that this result is interesting in its own right in graph theory.

6 Protocols

As mentioned in Section 4, protocols are crucial for the overall security of a time-stamping service. In this section we briefly describe the three protocols constituting our time-stamping service. The stamping protocol and the stamp completion protocol are improvements over the protocols of [BLLV98, Section 4.2], using threaded authentication trees. The publication protocol is an

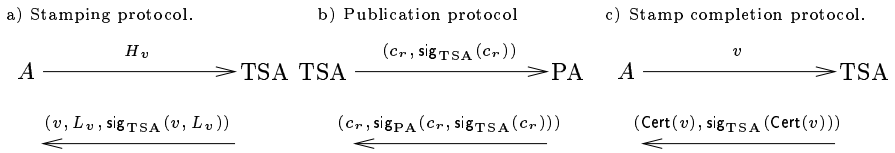

Fig. 4. Protocols of a time-stamping service.

additional protocol required in our model of time-stamping. See Figure 4 for an overview of the protocols. We assume the base graph G and the functions $\mathcal{P}$ and ϕ to be fixed.

The following algorithm is used in the stamping protocols to verify whether $v_1, v_2 \in \mathcal{S}([\![G]\!]_\phi)$ and $v_1 < v_2$ (i.e., whether the corresponding documents H_{v_1} and H_{v_2} were stamped during the same round and further that H_{v_1} was stamped before H_{v_2} was). Note that the algorithm proposed in [BLLV98] included an unnecessary checking whether $\mathsf{tail}(v_1) \cap \mathsf{head}(v_2) \neq \emptyset$ as a separate step.

Verification Algorithm. Let $C = (v, \boldsymbol{L}_\chi, \boldsymbol{L}_\tau)$, where v is a non-source vertex and χ and τ are paths in $[\![G]\!]_\phi$, such that v is computable from χ and $|V|$ is computable from τ. Let $\mathsf{VfyCert}(C, L_v, L_{|V|})$ be a function that accepts iff χ is authenticator of some path from 1_ϕ to v and τ is authenticator of some path from v_ϕ to $|V|$, and the value of a candidate v computed from the values in $\boldsymbol{L}_\chi$ is equal to v (the same verification is done for $|V|$ and $\boldsymbol{L}_\tau$). The algorithm V gets as input a tuple $(C_1, C_2, L_{v_1}, L_{v_2}, R_{r-1}, R_r, H_v)$, where $C_i = (v_i, \chi_i, \tau_i)$ and accepts iff (1) $\mathsf{VfyCert}(C_i, L_{v_i}, R_{r-1}, R_r, H_v)$ holds, for $i \in \{1, 2\}$; and (2) $v_1 < v_2$.

Stamping Protocol. See Fig. 4(a). The client A begins the protocol by sending hash H_v of the vth document to the TSA. The TSA calculates $L_v = h(\boldsymbol{L}_{\mathsf{head}(v)})$, adds L_v to the database of stamps, and sends A the second message. The client checks if the message is of the right form (note that he cannot yet verify whether L_v was computed correctly).

Publication Protocol. See Fig. 4(b). Here $c_r = (L_{|V|}, r)$ and $R_r = L_{|V|}$ is the cumulative round stamp of the rth round. After the end of the rth round the TSA begins the publication protocol by sending the tuple $(c_r, \mathsf{sig}_{\mathrm{TSA}}(c_r))$ to the PA. The PA checks if the message is of the right form. If it is, he sends the TSA the second message and publishes

$$\alpha_r = (c_r, \mathsf{sig}_{\mathrm{TSA}}(c_r), \mathsf{sig}_{\mathrm{PA}}(c_r, \mathsf{sig}_{\mathrm{TSA}}(c_r)))$$

on an accessible authenticated medium (e.g., in a newspaper). The TSA checks if the second message is of the right form. If it is, he waits for the publication and then checks if the published value α'_r is equal to α_r. If it is not, he sends α_r and the newspaper with α'_r to the judge.

Stamp Completion Protocol. See Fig. 4(c). At the end of the rth round, A gets the value α_r from the medium, checks if it is of correct form, and sends the request v to the TSA. The TSA returns the second message m. The client checks

if the signature is correct. If it is, he computes the cumulative hash $L_{|V|}$ from $\mathsf{Cert}(v)$ and compares it against the value in c_r. If the verification algorithm V rejects on input $\mathsf{Cert}(v)$, the client sends α_r and the message m to the court.

The above of protocols rebalances the communication, compared these in [BLLV98], by moving the burden from the stamping protocol to the stamp completion protocol, with the intuition that a lightweight stamping protocol helps to avoid trivial reordering attacks. Because of the collision-resistance of hash function h, transmission of L_v is sufficient if G is a simply connected authentication graph.

7 Optimality

As informally argued in Section 4, the round graph $G = (V, E)$ of a time-stamping service has to be based on a simply connected graph for which the corresponding ϕ-scheme is such that for any source v, L_1 is included in v's certificate. More precisely, for any source v, there has to be a node w in every root path of v, such that $(1, w) \in E$ and w is computable from $(1, \ldots, v)$ but *not* computable from $(1, \ldots, v-1)$. We say that a graph satisfying these conditions is *good*.

For a good graph G we define $\mathsf{a}(G) = \max_{v \in \mathcal{S}(G)} \min_{\mathcal{P}} |\mathsf{A}_G(\mathcal{P}(v))|$, where $\mathcal{P}$ ranges over all pass-through functions. Recall that $h : \{0,1\}^* \to \{0,1\}^k$. Then by the definition of certificates, $|\mathsf{Cert}(v)| = k(\min_{\mathcal{P}} |\mathsf{a}_G(\mathcal{P})| + 1)$, and thus $\max_v |\mathsf{Cert}(v)| = k(\mathsf{a}(G) + 1)$. Hence, a tight lower bound on $\mathsf{a}(G)$ yields a tight lower bound on $\max_v |\mathsf{Cert}(v)|$. If G is a round graph, then the number of time-stamps issued per round is equal to $n = |\mathcal{S}(G)| - 1$, since one source corresponds to the previous round stamp. Next we prove that for any good graph G, $\mathsf{a}(G) \geq \log_2 n + 2$. Hence, time-stamping service based on threaded tree schemes is certificate length optimal if we cannot assume anything more but existence of (one) collision-resistant hash-function.

Theorem 1. *(1) For a rooted directed acyclic graph G, $\mathsf{a}(G) \geq \log_2 |\mathcal{S}(G)| + 1$. (2) For a good graph G, $\mathsf{a}(G) \geq \log_2(|\mathcal{S}(G)| - 1) + 2$.*

Proof. (1) We show in several steps how a given rooted directed acyclic graph G can be transformed by local modifications to a complete binary tree T_d such that $|\mathcal{S}(T_d)| \geq |\mathcal{S}(G)|$ and $\mathsf{a}(T_d) \leq \mathsf{a}(G)$.
First step (eliminating fan-in > 2). Replace any vertex with $s > 2$ incoming edges with a binary tree with s sources (Fig. 5, 1). Let G_1 be the resulting graph. Then clearly $|\mathcal{S}(G_1)| = |\mathcal{S}(G)|$ and $\mathsf{a}(G_1) \leq \mathsf{a}(G)$.
Second step (eliminating fan-out > 1). Every vertex with fan-out > 1 can be replicated (Fig. 5, 2). An application of this step pushes the vertices with fan-out > 1 "downwards" in the graph. The final applications just replicate the sources, without adding any vertices with fan-out > 1. Thus, we can repeat the procedure until we get a binary tree G_2 with no vertex having fan-out > 1. Trivially, $|\mathcal{S}(G_2)| \geq |\mathcal{S}(G_1)|$ and $\mathsf{a}(G_2) = \mathsf{a}(G_1)$.

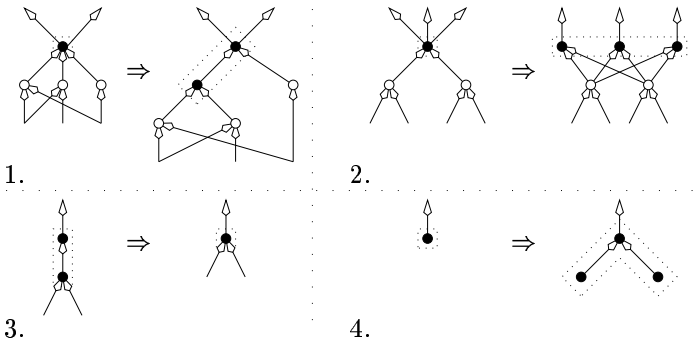

Fig. 5. Local graph modifications in Theorem 1.

Third step (making G_2 complete). Any vertex $v \in V(G_2)$ with only one predecessor can be deleted (Fig. 5, 3). Let the resulting graph be G_3. Trivially, $|\mathcal{S}(G_3)| = |\mathcal{S}(G_2)|$ and $\mathsf{a}(G_3) = \mathsf{a}(G_2)$.
Fourth step (balancing G_3). If G_3 is not yet a complete binary tree, there exists a source $v \in V(G_3)$ such that $d(v, |V(G_3)|)$ is less than the height of G_3. We proceed by adding two vertices as predecessors of v (Fig. 5, 4). Let the resulting graph be G_4. Trivially, $|\mathcal{S}(G_4)| \geq |\mathcal{S}(G_3)|$ and $\mathsf{a}(G_4) = \mathsf{a}(G_3)$.

Thus for any graph G there exists a complete binary tree T_d such that $|\mathcal{S}(T_d)| \geq |\mathcal{S}(G)|$ and $\mathsf{a}(T_d) \leq \mathsf{a}(G)$. But $|\mathcal{S}(T_d)| = 2^d$ and $\mathsf{a}(T_d) = d + 1$, thus for any graph G, $\mathsf{a}(G) \geq \mathsf{a}(T_d) = \log_2 |\mathcal{S}(T_d)| + 1 \geq \log_2 |\mathcal{S}(G)| + 1$. □

(2) Let $G = (V, E)$ be a good graph with $n = |\mathcal{S}(G)| - 1$. W.l.o.g. we can assume that the minimal authenticator for any $v \in \mathcal{S}(G)$ has the cardinality $\mathsf{a}(G)$. Now let G_1 be the subgraph of G spanned by $V \setminus \{1\}$. By the definition of good graphs, $\mathsf{a}(G_1) = \mathsf{a}(G) - 1$. Since $\mathsf{a}(G_1) \geq \log_2 |\mathcal{S}(G_1)| + 1$, we get that $\mathsf{a}(G) \geq \log_2 n + 2$. □

8 Conclusion

The new time-stamping scheme achieves optimal lengths of time certificates in a general sense ($k \log_2 n$ versus $3.786k \log_2 n$ compared to the optimal anti-monotone scheme [BL98], where n is the number of time stamps issued during a round). As compared to the formerly known most efficient tree schemes, the new scheme has the same space complexity and only somewhat larger time complexity, while being accountable like the scheme of [BLLV98]. We also presented the first accountable publication protocol that makes it unnecessary to audit the publication process.

In practice, a reliable time-stamping service is virtually impossible to achieve if the TSA and the PA are not replicated. Further research in this area is definitely necessary. The formal notion of temporal security is not yet developed and needs additional research; it is not even clear if it could be defined separately from the security of other primitives. An interesting question is whether

more efficient time-stamping services could be built on stronger cryptographic assumptions.

Acknowledgements

The authors are thankful to Stuart Haber, Markus Jakobsson, Peeter Laud, Kaisa Nyberg and to the anonymous referees for feedback and helpful discussions.

References

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

[BdM93] Josh Benaloh and Michael de Mare. One-Way Accumulators: A Decentralized Alternative to Digital Signatures (Extended Abstract). In Tor Helleseth, editor, *Advances in Cryptology—EUROCRYPT 93*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285. Springer-Verlag, 1994, 23–27 May 1993.

[BHS92] Dave Bayer, Stuart A. Haber, and Wakefield 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.

[BL98] Ahto Buldas and Peeter Laud. New Linking Schemes for Digital Time-Stamping. In *The 1st International Conference on Information Security and Cryptology*, pages 3–14, Seoul, Korea, 18–19 December 1998. Korea Institute of Information Security and Cryptology.

[BLLV98] Ahto Buldas, Peeter Laud, Helger Lipmaa, and Jan Villemson. Time-Stamping with Binary Linking Schemes. In Hugo Krawczyk, editor, *Advances on Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 486–501, Santa Barbara, USA, August 1998. Springer-Verlag.

[HS90] Stuart Haber and Wakefield Scott Stornetta. How to Time-Stamp a Digital Document. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology—CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 437–455. Springer-Verlag, 1991, 11–15 August 1990.

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

[Mer80] Ralph C. Merkle. Protocols for Public Key Cryptosystems. In IEEE, editor, *Proceedings of the 1980 Symposium on Security and Privacy, April 14–16, 1980 Oakland, California*, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1980. IEEE Computer Society Press.

[San99] Tomas Sander. Efficient Accumulators without Trapdoor. In *The Second International Conference on Information and Communication Security*, Sydney, Australia, 9–11 November 1999. To appear.