Certificate Management using Undeniable Status Attestations*
(Submitted Version)

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

[1] Küberneetika AS, Akadeemia tee 21, 12618 Tallinn, Estonia
{ahto.buldas,helger.lipmaa}@cyber.ee
[2] Universität des Saarlandes, FB 14 Informatik Im Stadtwald - Bau 45, Postfach 15 11 50, 66041 Saarbrücken, Germany
laud@cs.uni-sb.de

Abstract. We introduce a new type of *authenticated search trees*, an alternative to Certificate Revocations Lists that enables one to reduce the scope of trusted operations performed by Certificate Authorities. Our construction is similar to the authenticated data structures proposed by Naor and Nissim [NN98] but, as we show, it fulfills some stronger security objectives that seem to be necessary to make the certificate management system accountable.

Keywords: accountable certificate management, authenticated search trees, attesters, public-key infrastructure, search trees.

1 Introduction

Secure electronic commerce necessitates wide-scale employment of public-key cryptography that, in turn, requires secure and efficient methods of certificate management in the *Public-Key Infrastructure* (PKI). Most of the known techniques for the latter involve a public database of valid (or revoked) certificates that originates from a trusted source (known as Certification Authority, CA), but is maintained by a less trusted party (known as a Directory Service, DS) who provides on-line certificate information services to the clients. Neither the CA nor the DS is forced to store old certificates. Anyone having a candidate certificate and a certificate-specific attestation can efficiently verify whether the certificate was included in the database (i.e., whether it was valid) at a given moment in time, even if the third parties have ceased to exist in between.

Certificate management is very often *the* security bottleneck in electronic commerce and other legal applications of digital signatures. Therefore it is highly desirable to minimize the number of possible frauds the third parties involved in PKI can accomplish. This is often done by auditing the protocols followed by the CA. Certificate issuing and revocation are examples of such procedures, during

* Partially supported by Estonian Science Foundation grant 3742.

which the CA is *forced* to return the attestation. To audit each operation the CA performs would be clearly too costly and therefore, for the sake of efficiency, the CA is usually assumed to be trusted in some aspects of certificate management. Additional trust assumptions mean however that the origin of some frauds may stay undetectable.

Still, we want that, in legal disputes arising from the frauds in certificate management, both the frauds themselves and their origins were efficiently detectable and provable to the third parties (i.e., we want the certificate management system to be *accountable*). Ideally, one should be able to distinguish between the frauds done by a CA and the frauds caused by someone capable of breaking the used primitives. This means that the number of frauds that a computationally challenged CA is able to perform undetectably should be minimal.

To explain how to proceed in building an accountable certificate management system, we note that any legal case caused by the problems in certificate management would involve two principals, having a pair of so called *contradictory* attestations, so that verification accepts or rejects the certificate depending on which attestation was used as input to the algorithm. Clearly, if it were tractable for the CA to create contradictory attestations, we would have a non-accountable system.

On the other hand, let us assume that it is intractable for the CA (or anyone else) to create contradictory attestations (we would then say that the attestations are *undeniable*). In this case, one could resolve possible disputes unambiguously if at least one interested party has the corresponding attestation, which is almost always the case. Now if two clients would still present contradictory attestations in court, the judge can deduce that some underlying primitive has been broken and should be replaced. Attestations generated by using broken-by-now primitives can be refreshed by using digital time-stamping [HS91,BHS92]. Note that the used time-stamping system should be accountable [BLLV98,BLS00], since otherwise we would introduce another vulnerability — trust in the time-stamping authority — in the system.

To conclude this brief argumentation, it is desirable to find efficient constructions for creating undeniable attestations. This paper concentrates on the existence of suitable cryptographic primitives — *undeniable attesters*. We propose a very simple and intuitive authenticated search tree based construction of undeniable attesters with attestation lengths and update times comparable to Certificate Revocation Trees (CRTs) [Koc98,NN98] which, as we will demonstrate, are not undeniable attesters. Our construction assigns to every internal node v of a search tree a hash value $S[v]$ taken over the labels of v's children *and* the sorting keys of v. The main difference between this construction and the CRTs is that CRTs do not include the sorting keys in $S[v]$. While this omission shortens the negative attestations by a factor of 2, it also opens a possibility to the CA to perform frauds. It is in several aspects also more intuitive than the CRTs, since it is directly based on the search trees as they are generally understood in computer science, therefore allowing to carry over to cryptography a lot of the research done in the area of algorithms and data structures [Knu98].

Our solution is the first efficient certificate management system that enables one to replace inefficient CRLs without compromising security and has hence, as we think, great practical importance. The described construction can almost always be used instead of the certificate revocation trees in order to reduce trust in the database maintainer, without decreasing efficiency.

As with any new cryptographic primitive, it is good to know how it relates to the previously known primitives. We show, using our construction, that undeniable attesters exist if and only if collision-resistant hash functions exist.

We start the paper with necessary preliminaries (Section 2). Thereafter, in Section 3, we give an overview of the existing solutions to the certificate management, point out some weaknesses in them and argue (informally) for the model used in the rest of the paper. Formal definitions of attesters, including the collision-resistant and undeniable ones, are given in Section 4. Section 5 describes two constructions of collision-resistant attesters, closely based on Certificate Revocation Trees [Koc98,NN98]. A construction of authenticated search trees is given in Section 6, where also relations between undeniable attesters, collision-resistant hash functions and collision-resistant attesters are shown. Section 7 focuses on efficiency issues. Final conclusions is presented in Section 8.

2 Preliminaries

Let $\Sigma = \{0,1\}$. As usually, Σ^k denotes the set of k-bit words, $\Sigma^* := \bigcup_{k\geq 0} \Sigma^k$. We assume that NIL is a special symbol encoded differently from any $x \in \Sigma^*$. Let $\mathcal{EA}$ be the class of probabilistic algorithms with execution time polynomial in the length of their input. A probability family $P = (P_k)$, $k \in \mathbb{N}$, is *negligible* if for all $\varepsilon > 0$ there exists a k_ε, such that $|P_k| < k^{-\varepsilon}$, for any $k > k_\varepsilon$. Notation $A \leftarrow \mathsf{S}$ means that A is assigned according to the probability space S that may be generated by some probabilistic algorithm.

A *collision-resistant hash function* (CRHF) $\mathcal{H}$ for some index set $I \subseteq \Sigma^*$ is a pair (G, H), such that 1) $\mathsf{G} \in \mathcal{EA}$ is a *generation algorithm*, so that $\mathsf{G}(1^k) \in \Sigma^k \cap I$; 2) for an index $i \in I$, $\mathsf{H}(i,\cdot) = \mathsf{H}_i(\cdot)$ is a function $\mathsf{H}_i : \Sigma^{p(|i|)} \to \Sigma^{|i|}$, so that $\mathsf{H} \in \mathcal{EA}$, for some polynomial p, where $p(k) > k$. 3) For all algorithms $\mathsf{A} \in \mathcal{EA}$, the next probability family $\mathsf{CRH}_{\mathcal{H}}(\mathsf{A})$ is negligible in k:

$$\mathsf{CRH}_{\mathcal{H},k}(\mathsf{A}) := \Pr[i \leftarrow \mathsf{G}(1^k), (x_1,x_2) \leftarrow \mathsf{A}(1^k,i) : x_1 \neq x_2 \wedge \mathsf{H}_i(x_1) = \mathsf{H}_i(x_2)]$$

(where the probability is taken over the coin tosses of G and A).

A binary tree $T = (V,E)$ is a *search tree* [Knu98, Section 6.2.2] if every node $v \in V$ has a unique *search key* $K[v]$ associated to it, so that if w is a descendant of the left (resp. right) child of v, then $K[w] < K[v]$ (resp. $K[w] > K[v]$).

3 Related Work and Motivations

We follow the model that the database S of valid (or revoked) certificates x originates from an off-line CA, but the on-line certificate information services are per-

formed by a Directory Service (DS). In the simplest setting the DS just responds to the client's query with a (possibly signed by DS) attestation $\mathsf{P}(x, S) = 1$, if $x \in S$, and $\mathsf{P}(x, S) = 0$, otherwise. Here, the DS can easily create contradictory attestations. However, it is widely accepted that the DS as an on-line service should not be trusted and therefore such a solution is not recommended.

In a slight modification of this scheme the DS transmits to a client CA's time-stamped signature on x if $x \in S$. This makes it intractable for the DS to convince that $x \in S$ if it actually does not. However the DS can still convince the client that $x \notin S$ even if it is not the case. Moreover, this solution does not reduce the trust in the CA. The third drawback of this solution is that the attestations cease to have any legal value after revocation of CA's signature key. This is clearly unacceptable in situations where the certificates held in S must be used long after they have been issued.

The next method is a common solution to the said problems. Namely, it is assumed that the whole database S is updated periodically, and a short digest $d = \mathsf{D}(S)$ of it is made available for clients by publishing it in authenticated way so that it is intractable for anyone to alter the data already published. This can be accomplished by following a publication protocol similar to that of [BLS00]. Now, applying a verification algorithm V to a value x, digest $\mathsf{D}(S)$ and attestation $\mathsf{P}(x, S)$, a client can efficiently establish whether $x \in S$. We call such triples $(\mathsf{P}, \mathsf{D}, \mathsf{V})$ *attesters*.

An utterly impractical but still widely used attester underlies the Certificate Revocation Lists (CRLs), where the DS transmits a copy of the whole database, time-stamped and signed by the CA, to the client. Here, $d = \mathsf{D}(S)$ is a succinct efficiently computable value, and $\mathsf{P}(x, S)$ is equal to the set S. Clearly, the clients can unambiguously establish whether $x \in S$, since they have a copy of the whole database. They may bring an action against the CA if different copies of S were transmitted to different users. Note that it is intractable for anyone to create contradictory attestations if D is a collision-resistant hash function. That is, it is intractable to find a pair of attestations $(p, \overline{p})$, so that $\mathsf{V}(x, d, p) = 1$ but $\mathsf{V}(x, d, \overline{p}) = 0$ In what follows we call such attesters $(\mathsf{P}, \mathsf{D}, \mathsf{V})$ *undeniable*. As informally argued before, undeniable attesters are sufficient for accountable certificate management if 1) the certificate issuing and revocation processes are audited and 2) an accountable publication protocol is employed.

The "only" problem with CRLs is their inefficiency: namely, if k is a security parameter then $|\mathsf{D}(S)| = \Theta(k)$ and $|\mathsf{P}(x, S)| = \Theta(k|S|)$. As pointed out in many sources (see, e.g., [Mic96] for some detailed discussion), CRLs are extremely costly in practice since the lengthy attestations $\mathsf{P}(x, S)$.We focus here on constructing undeniable attesters with shorter attestations while not increasing tre trust in the third parties.

RSA accumulator [BdM93,BP97] has succinct attestations of length $\Theta(k)$. As pointed out by Nyberg [Nyb96a,Nyb96b], succinctness of the attestations is caused by the built-in trapdoor information that is known to some coalition of participants, which should therefore to be trusted. The best known method [San99] of making the RSA accumulator trapdoorless introduces attestation

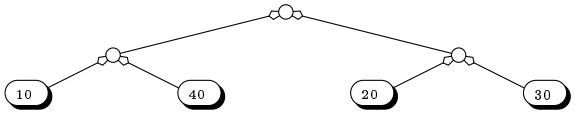

Fig. 1. A toy example of an improperly created CRT.

lengths of order $\Theta(k^2)$. RSA accumulator provides positive attestations: it is intractable, for any S, to find an attestation p so that $\mathsf{V}(x, \mathsf{D}(S), p)$ accepts, if $x \notin S$. We call *collision-resistant provers* the cryptographic primitives satisfying this objective.

Another well-known construction — hash tree — has been widely used as a collision-resistant prover since [Mer80] in a broad variety of applications (to name a few, incremental cryptography, certificate revocation, memory checking, time-stamping, one-time signatures, auditable e-cash). In most of the hash trees, $|\mathsf{D}(S)| = \Theta(k)$ and $|\mathsf{P}(x, S)| = \Theta(k|S|)$.

The opposite question of finding *collision-resistant disprovers* where, for any set S and for any $x \in S$, it is intractable to find an attestation p such that verification $\mathsf{V}(x, \mathsf{D}(S), p)$ rejects, has got much less attention. Micali's Certificate Revocation Status (CRS, [Mic96], later refined in [ALO98]) provided both succinct positive and negative attestations, so that $|\mathsf{P}(x, S)| = \Theta(k)$. However, in this solution the CA-to-directory communication size (or $|\mathsf{D}(S)|$ in our model) is $\Theta(|S|)$.

A better solution is due to Kocher [Koc98] and Naor and Nissim [NN98] who assumed that the values stored at hash tree leaves are sorted. In this case, the resulting hash-tree construction will become both a collision-resistant prover and a collision-resistant disprover (we will call such primitives *collision-resistant attesters*). Their construction is until now the only known *efficient* collision-resistant attester.

However, their construction (Certificate Revocation Tree, CRT) is *not* undeniable. We demonstrate this with the Naor-Nissim style CRT [NN98] depicted by Figure 1. Here the leaves are labeled (from left to right) by data structures representing the certificates with serial numbers 10, 40, 20 and 30. The pair of the authentication paths of two rightmost (resp. leftmost) leaves is a valid attestation that the certificate with serial number 20 is (resp. is not) valid.

Verifier, given the digest d (root of the hash tree), will accept or reject depending on what pair of authentication paths was submitted to her. The only way to reliably detect this "non-sorting" attack is to verify the validity of *all* authentication paths, a solution that is clearly impractical and even impossible to perform, if some paths are inaccessible (if, to lessen the storage requirements, the old versions of the certificate database are not stored).

4 Formal Definitions

Definition 1. *A quadruple* $\mathcal{A} = (\mathsf{G}, \mathsf{P}, \mathsf{D}, \mathsf{V})$ *is an* attester *for an index set* $I \subseteq \Sigma^*$*, if there is a polynomial* p*,* $p(k) > k$*, such that:*

1. Generating algorithm $\mathsf{G} \in \mathcal{EA}$ *takes as input a security parameter* 1^k *and outputs an index* $i \in \Sigma^k \cap I$*.*
2. Proving algorithm $\mathsf{P} \in \mathcal{EA}$ *takes as input an index* i*, an element* $x \in \Sigma^k$ *and a set* $S \subseteq \Sigma^k$*,* $|S| \leq p(k)$*, and outputs an* attestation $\mathsf{P}_i(x, S) = \mathsf{P}(i, x, S)$*.*
3. Digest algorithm D *takes as input an index* i*, a set* $S \subseteq \Sigma^k$*,* $|S| \leq p(k)$*, and outputs a digest* $\mathsf{D}_i(S) = \mathsf{D}(i, S)$*.*
4. Verification algorithm V *takes as input an index* i*, a candidate element* $x \in \Sigma^k$*, a digest* d *and a proof* C*, and outputs* $\mathsf{V}_i(x, d, C) = \mathsf{V}(i, x, d, C) \in \{1, 0, \mathsf{Error}\}$*. We require that for any* $S \subseteq \Sigma^k$*,* $|S| \leq p(k)$*, and any* $x \in \Sigma^k$*,* $\mathsf{V}_i(x, \mathsf{D}_i(S), \mathsf{P}_i(x, S))$ *outputs* 1 *if* $w \in S$ *and* 0*, otherwise. If* $S \not\subseteq \Sigma^k$*,* $|S| > p(k)$ *or* $x \notin \Sigma^k$ *then for any* C*,* $\mathsf{V}_i(x, \mathsf{D}_i(S), C) = \mathsf{Error}$*.*

In practical applications we want the attesters to have "succinct" attestations and digests and fast (amortized) update time. Informally, we say an attester is *dynamic* if it has $O(k \log |S|)$ (amortized) time per insertion and deletion. We say an attester is *succinct* if $|\mathsf{D}_i(S)| = O(|i|)$ and $|\mathsf{P}_i(x, S)| = O(|i| \log |S|)$

Definition 2. *Let*

$$\mathsf{CRP}_{\mathcal{A},k}(\mathsf{A}) := \Pr[i \leftarrow \mathsf{G}(1^k), (x, S, p) \leftarrow \mathsf{A}(1^k, i) : x \notin S \wedge \mathsf{V}_i(x, \mathsf{D}_i(S), p) = 1] \ ,$$

$$\mathsf{CRD}_{\mathcal{A},k}(\mathsf{A}) := \Pr[i \leftarrow \mathsf{G}(1^k), (x, S, \overline{p}) \leftarrow \mathsf{A}(1^k, i) : x \in S \wedge \mathsf{V}_i(x, \mathsf{D}_i(S), \overline{p}) = 0] \ ,$$

$$\mathsf{UN}_{\mathcal{A},k}(\mathsf{A}) := \Pr[i \leftarrow \mathsf{G}(1^k), (x, d, p, \overline{p}) \leftarrow \mathsf{A}(1^k, i) : \mathsf{V}_i(x, d, p) = 1 \wedge \mathsf{V}_i(x, d, \overline{p}) = 0] \ .$$

Attester $\mathcal{A}$ *is a* collision-resistant prover *(resp.* collision-resistant disprover*) if* $\forall \mathsf{A} \in \mathcal{EA}$*,* $\mathsf{CRP}_{\mathcal{A}}(\mathsf{A})$ *(resp.* $\mathsf{CRD}_{\mathcal{A}}(\mathsf{A})$*) is negligible. Attester* $\mathcal{A}$ *is a* collision-resistant attester *if for any* $\mathsf{A} \in \mathcal{EA}$*, both* $\mathsf{CRP}_{\mathcal{A}}(\mathsf{A})$ *and* $\mathsf{CRD}_{\mathcal{A}}(\mathsf{A})$ *are negligible. An attester* $\mathcal{A} = (\mathsf{G}, \mathsf{P}, \mathsf{D}, \mathsf{V})$ *is* undeniable *if for any* $\mathsf{A} \in \mathcal{EA}$*,* $\mathsf{UN}_{\mathcal{A}}(\mathsf{A})$ *is negligible.*

5 Collision-Resistant Attesters

We proceed by giving an efficient construction [Koc98,NN98] of collision-resistant attesters based on Merkle's hash-tree construction [Mer80] that is by itself an collision-resistant prover but not a collision-resistant disprover. The latter is since a candidate string x can be a label of any leaf. That means a negative attestation should incorporate all positive attestations. However, one can make the negative attestations succinct if we assume that the values at hash tree leaves are sorted.

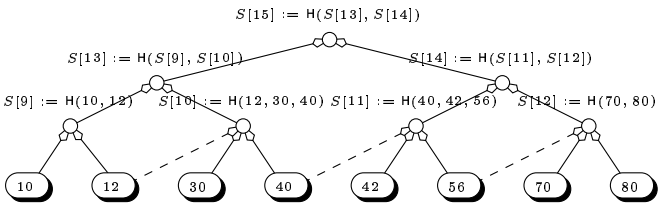

Fig. 2. Toy example of a collision-resistant attester.

For the sake of simplicity we will describe the attesters for a fixed $i \in \Sigma^k \cap I$. Suppose that $S = \{S[1], ..., S[n]\}$ is a set of k-bit integers such that $S[j] < S[j+1]$ for any $0 < j < n$. Let T be a binary tree with n leaves, with its j-th leftmost leaf j is labeled by $S[j]$. A non-leaf vertex v is labeled with an auxiliary hash value

$$S[v] = \mathsf{H}(S[v_{\mathsf{L}}], S[v_{\mathsf{R}}]) \ ,$$

where v_{L} (v_{R}) denotes the left (right) child of v. The digest $d = \mathsf{D}(S)$ of S is equal to the label of the root vertex v_n.

Let $p = (b; h_1, h_2, \ldots, h_m)$, so that $h_j \in \Sigma^k$ and $b = b_1 \ldots b_m$, $b_j \in \{0, 1\}$. The verification algorithm $\mathsf{V}(x, d, p)$ computes d_m by assigning $d_0 := x$ and then recursively, for all $j > 0$,

$$d_j := \begin{cases} \mathsf{H}(d_{j-1}, h_j), & \text{if } b_j = 0 \ , \\ \mathsf{H}(h_j, d_{j-1}), & \text{if } b_j = 1 \ . \end{cases}$$

Verification returns 1, if $d_m = d$, and Error, otherwise. Proving that $x \in S$ is equivalent to finding a p such that $\mathsf{V}(x, d, p)$ accepts. Proving that $x \notin S$ is equivalent to finding to finding a quadruple (x_1, p_1, x_2, p_2), such that $\mathsf{V}(x_1, d, p_1) = \mathsf{V}(x_2, d, p_2) = 1$, $x_1 < x < x_2$, and x_1 and x_2 correspond to two neighboring leaves in the tree T.

Negative attestations can be shortened by adding edges to the underlying tree as follows (cf also [Koc98,NN98]): if the parents of a leaf $v \neq 1$ and its left neighbor leaf w are different, then add an edge from w to v's parent as in Figure 2. Build the attester upon the resulting graph, by modifying the algorithms P, D and V to account with the new edges. Both attesters are succinct and dynamic collision-resistant. However, as shown in Section 3, they are not undeniable attesters.

6 Undeniable Attesters

Next we will give a construction of what we call *authenticated search trees*. Thereafter we show that the resulting construction is an undeniable attester, and finish the Section with some discussion.

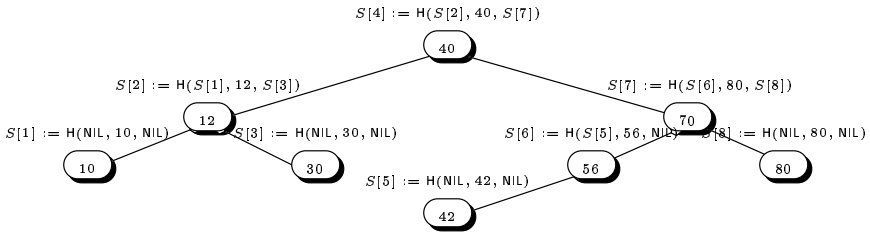

Fig. 3. A toy example of authenticated search trees.

We do it as previously for a fixed k and a $i \in \Sigma^k \cap I$. Let S be a set and let T be a binary search tree with $|S|$ vertices. Each vertex v of T is labeled by a pair $(K[v], S[v])$. Here the elements $K[v]$ belong to the set S and $K[v_1] \neq K[v_2]$, if $v_1 \neq v_2$. Moreover, the tree T together with keys $K[v]$ is a search tree. The values $S[v]$ are computed as follows:

$$S[v] := \mathsf{H}(S_{\mathsf{L}}, K[v], S_{\mathsf{R}}) \ ,$$

where S_{L} (S_{R}) is equal to the label S of the v's left (right) child if the corresponding child exists, or NIL, otherwise. For example, if v is a leaf then $S[v] = \mathsf{H}(\mathsf{NIL}, K[v], \mathsf{NIL})$. Once again, the digest $\mathsf{D}(S)$ is defined as $S[r]$, where r is the root vertex.

Let $p = (h_{\mathsf{L}}, k_0, h_{\mathsf{R}}; k_1, h_1; k_2, h_2; ...; k_m, h_m)$. Verification $\mathsf{V}(x, d, p)$ returns Error if (1) $h_{\mathsf{L}} \neq \mathsf{NIL}$ and $x < k_0$, or (2) $h_{\mathsf{R}} \neq \mathsf{NIL}$ and $x > k_0$. Otherwise, V assigns $d_0 := \mathsf{H}(h_{\mathsf{L}}, k_0, h_{\mathsf{R}})$ and for all $0 < j < m$:

$$d_j := \begin{cases} \mathsf{H}(d_{j-1}, k_j, h_j) & \text{if } x < k_j \ , \\ \mathsf{H}(h_j, k_j, d_{j-1}) & \text{if } x > k_j \ . \end{cases}$$

Now, V outputs Error if $d_m \neq d$ or $x = k_j$, for some $j \geq 1$. Otherwise it returns 1 or 0, depending on whether $k_0 = x$. The algorithm $\mathsf{P}(x, S)$ outputs a list p such that $\mathsf{V}(x, \mathsf{D}(S), p)$ accepts. Therefore, an attestation concerning $x \in \Sigma^k$ is principally equivalent to searching x from a search tree, where the usage of hash functions in the vertices guarantees that the CA has to work with the same tree during each query.

A toy example with $S = \{10, 12, 30, 40, 42, 56, 70, 80\}$ is depicted by Figure 3. Here, $\mathsf{D}(S) := S[4]$ and

$$\mathsf{P}(42, S) = \mathsf{P}(43, S) := (\mathsf{NIL}, 42, \mathsf{NIL}; 56, \mathsf{NIL}; 70, S[8]; 40, S[2]) \ .$$

Theorem 1. *If there exists* $\mathsf{A} \in \mathcal{EA}$ *with* $\mathsf{UN}_{\mathcal{A}}(\mathsf{A}) = \varepsilon$ *then there exists* $\mathsf{A}' \in \mathcal{EA}$ *with* $\mathsf{CRH}_{\mathcal{H}}(\mathsf{A}') = \varepsilon$.

Proof. Given an index i and the security parameter 1^k the adversary A' performs a query to $\mathsf{A}(1^k, i)$ that, with probability ε outputs a tuple $(x, d, p, \overline{p})$ such that $\mathsf{V}_i(x, d, p) = 1$ and $\mathsf{V}_i(x, d, \overline{p}) = 0$.

It follows from the assumptions that in both verifications $d_m = d$, i.e. (1) we can either find a collision during the recursive verification at some step $j > 1$ or (2) d_0 has the same value in both verifications. The latter leads to the collision because by the description of V the value of k_0 cannot be the same in both verifications. Therefore by simulating the verification procedure with arguments (x, d, p) and $(x, d, \overline{p})$ the adversary A' finds a collision to $\mathcal{H}$ with probability ε. □

The proof of the next Theorem is given in Appendix A.

Theorem 2. *1) Any undeniable attester is also collision-resistant attester, but the opposite is not true. 2) Undeniable attesters exists if and only if CRHFs exist.*

Given construction generalizes to the case when the underlying tree is a multiway search tree [Knu98, Section 6.2.4]. However, if we wish the attestations to be of length $O(k \log |S|)$, we are (as in the case of collision-resistant attesters) restricted to the trees where the number of children of every node is upper-bounded with some constant that does not depend on k. Therefore we cannot base our construction on exponential search trees and other related data structures that have been lately extensively used in sublogarithmic search algorithms.

Authenticated search trees can be made dynamic as in [NN98] by requiring that the CA stores the whole hash tree and after each database update, updates all the necessary hash values in the tree, including the value $d = \mathsf{D}(S)$. Updating can be done in time $O(k \log |S|)$ if by using appropriate dynamic search trees.

There are a number of other possible constructions of undeniable attesters. For example, one could add a number of edges to a binary tree as follows: for any non-leaf node v, add an edge (if it already does not exist) from its left child's rightmost descendant leaf to v. We stress that the main difference between the described constructions of collision-resistant and undeniable attesters is that in the first case the choice between the left and the right subtree is just done by an explicitly given bit b_i, while in the latter case an explicit key is given, so that based on this key the verifier can additionally check that the element returned in a query is in the correct location in this tree.

7 Efficiency

Authenticated search trees result in the shortest attestation length if built upon a complete binary tree. If we assume that the sorting keys have length k (we could store at leaves the hash values of certificates that are generally longer than k bits), the worst case attestation length would then be $k(2\log(n+1)-1)$, where $n = 2^{d+1} - 1$ is the number of leaves. A simple calculation shows that the attestations $\mathsf{P}_i(x, S)$ have in total $\frac{1}{k}\sum_{i=1}^{2^{d+1}-1} |\mathsf{P}_i(x, S)| = 2^{d-1}(2d+2) - 3 + 2\sum_{i=0}^{d-2} 2^i i = 2^{d+1}(d-1) + 1$ elements, which makes the amortized attestation

length equal to

$$k \cdot \frac{2^{d+1}(d-1)+1}{2^d-1} \approx k \cdot 2d \approx 2k \log n \ .$$

Actually, our construction has two times longer attestations than the optimal construction of collision-resistant attesters. In the case of the dynamic AVL trees [Knu98, Section 6.2.3], the worst case certificate length is $2.88 \cdot k \log n$.

The next compression technique, similar to arithmetic coding, helps to further shorten the attestations in authenticated search trees. Let T be a fixed search tree, and let k be the security parameter. Then we assign a range (ℓ_v, u_v) to every node v, so that (1) for the left child v_{L} of v, $\ell_{v_{\mathsf{L}}} = \ell_v$ and $u_{v_{\mathsf{L}}} = K[v]-1$; and (2) for the right child v_{L} of v, $\ell_{v_{\mathsf{L}}} = K[v]+1$ and $u_{v_{\mathsf{L}}} = u_v$. In particular, for the root vertex v, $(\ell_v, u_v) = (0, 2^k - 1)$. Now, we store with each node v the value $K[v] - \ell_v$ encoded with $\lceil \log_2(u_v - \ell_v) \rceil$ bits. The values (ℓ_v, u_v) can be recalculated every time the attestation $\mathsf{P}(x, S)$ is used in verification. By using this method, the attestations are never longer than $(2k - \frac{1}{2}\log|S|)\log|S|$, being therefore *always* shorter than the "uncompressed" attestations. In many cases the amortized length of attestation lengths may become relatively low. Also, additional compression techniques can be applied. Detailed calculations are omitted from the submitted version of paper.

The classical *predecessor problem* requires one to maintain a set S in such a way that queries of the form "Is j an element of S and, if not, what element of S, if any, is just before it in sorted order?" may be answered efficiently. *Membership problem* only requires that the question "Is j an element of S?" may be answered efficiently. Note that this attester has the property that $\forall v > 0$, $S[v-1]$ is implicitly in $\mathsf{P}(v, S)$, i.e., it explicitly solves the predecessor problem.

Note that there exist extremely efficient dynamic attesters if one does not require them to be collision-resistant. From one side, let $\mathcal{A}$ be an arbitrary attester so that f_i, $f \in \{\mathsf{G}, \mathsf{P}, \mathsf{D}, \mathsf{V}\}$, works in the worst-case time $t_{f,|i|}$. Straightforwardly, there exists a search algorithm working in time $t_{\mathsf{D}} + t_{\mathsf{P}} + t_{\mathsf{V}} + O(1)$, that solves the membership problem. On the other hand, by the results of [DKM+94] for search problems solving the membership problem, there exists a dynamic attester so that for any $S \subseteq \Sigma^k$, $\forall x \in S$, $t_{\mathsf{P}}, t_{\mathsf{D}}, t_{\mathsf{V}} = O(1)$, $|\mathsf{P}_i(x, S)| = 1$ and $|\mathsf{D}_i(S)| = 0$. (define $\mathsf{P}_i(x, S) = 1$ iff $x \in S$, and fix $\mathsf{D}_i(S)$ to be the empty string). However, both the Certificate Revocation Trees and our autenticated search trees solve the predecessor problem.

8 Conclusions

We defined the notions of collision-resistant attesters and undeniable attesters. We proposed a simple and efficient construction (authenticated search trees) of undeniable attesters. The latter construction is also the first known efficient construction of undeniable attesters in the literature. As we showed, undeniable attesters have important applications in certificate management.

Note that undeniable attesters are actually equivalent to collision-resistant hash functions in their strength. Collision-resistant attesters behave more like chameleon hash functions in the context of certificate management, since the only party who can find the collisions is the CA.

Most of the "reasonable" data structures for searching can be seen as search trees. Since our construction is just slight reformulations of what is usually meant by search trees, it can also be used in combination with dynamic data structures like the 2-3 trees (which were used for public-key infrastructure in [NN98]) and the AVL trees. The resulting structures support efficient delete and insert operations (more precisely, the update time increases not more than twice, compared to the construction of collision-resistant attesters).

Further Work

An open question is whether inclusion of trapdoor information helps to decrease the attestation length for collision-resistant attesters. Strict optimality of our constructions is left as another open question. For example, since it is known that it is easier to solve membership problem [DKM$^+$94] than predecessor problem [BF99], it is interesting to know whether succinct undeniable attesters can be built upon the search algorithms solving the membership problem.

References

[ALO98] William Aiello, Sachin Lodha, and Rafail Ostrovsky. Fast Digital Identity Revocation. In Hugo Krawczyk, editor, *Advances on Cryptology — CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 137–152, Santa Barbara, USA, 23–27 August 1998. Springer-Verlag.

[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.

[BF99] Paul Beame and Faith Fich. Optimal Bounds for the Predecessor Problem. In *Proceedings of the Thirty-First Annual ACM Symposium on the Theory of Computing*, pages 295–304, New York, 1–4 May 1999.

[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.

[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.

[BLS00] Ahto Buldas, Helger Lipmaa, and Berry Schoenmakers. Optimally Efficient Accountable Time-Stamping. In *Public Key Cryptography '2000*, volume ? of *Lecture Notes in Computer Science*, pages ??–?? Springer-Verlag, 18–20 January 2000. To appear.

[BP97] Niko Barić and Birgit Pfitzmann. Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees. In Walter Fumy, editor, *Advances on Cryptology — EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 480–494, Konstanz, Germany, May 1997. Springer-Verlag.

[Dam89] Ivan Damgård. A Design Principle for Hash Functions. In Gilles Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer-Verlag, 1990, 20–24 August 1989.

[DKM+94] Martin Dietzfelbinger, Anna Karlin, Kurt Mehlhorn, Friedhelm Meyer Auf Der Heide, Hans Rohnert, and Robert E. Tarjan. Dynamic perfect hashing: Upper and lower bounds. *SIAM Journal on Computing*, 23(4):738–761, August 1994.

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

[Knu98] Donald E. Knuth. *The Art of Computer Programming. Volume 3: Sorting and Searching.* Addison-Wesley, 2 edition, 1998.

[Koc98] Paul Kocher. On Certificate Revocation and Validation. In Rafael Hirschfeld, editor, *Financial Cryptography — Second International Conference*, volume 1465 of *Lecture Notes in Computer Science*, pages 172–177, Anguilla, British West Indies, 23–25 February 1998. Springer-Verlag.

[Mer80] Ralph Charles 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.

[Mer89] Ralph Charles Merkle. One Way Hash Functions and DES. In Gilles Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer-Verlag, 1990, 20–24 August 1989.

[Mic96] Silvio Micali. Efficient Certificate Revocation. Technical report, Massachusetts Institute of Technology, 22 March 1996.

[NN98] Moni Naor and Kobbi Nissim. Certificate Revocation and Certificate Update. In *7th USENIX Security Symposium*, 1998.

[Nyb96a] Kaisa Nyberg. Commutativity in Cryptography. In *Proceedings of the First International Workshop on Functional Analysis at Trier University*, pages 331–342, Berlin, 1996.

[Nyb96b] Kaisa Nyberg. Fast Accumulated Hashing. In Dieter Grollman, editor, *Fast Software Encryption: Third International Workshop*, volume 1039 of *Lecture Notes in Computer Science*, pages 83–87, Cambridge, UK, 21–23 February 1996. Springer-Verlag.

[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.

A Proof of Theorem 2

1) Let $\mathcal{A} = (\mathsf{G}, \mathsf{P}, \mathsf{D}, \mathsf{V})$, and let A be a machine such that $\mathsf{CRP}_{\mathcal{A}}(\mathsf{A}) = \varepsilon$. We now build an efficient machine M with $\mathsf{UN}_{\mathcal{A}}(M) = \varepsilon$. Let $i \leftarrow \mathsf{G}(1^k)$. Adversary $M(1^k, i)$ lets $(w, S, C) \leftarrow \mathsf{A}(1^k, i)$. With probability ε_k, either a) $w \notin S \subseteq \Sigma^k$,

but $\mathsf{V}_i(w, \mathsf{D}_i(S), C) = 1$; or b) $w \in S \subseteq \Sigma^k$, but $\mathsf{V}_i(w, \mathsf{D}_i(S), C) = 0$. M distinguishes between these cases by letting $d \leftarrow \mathsf{D}_i(S)$, $v \leftarrow \mathsf{V}_i(w, d, C)$ and returning $(w, d, \mathsf{P}_i(w, S), C)$ (case a), if $v = 1$, and $(w, d, C, \mathsf{P}_i(w, S))$ (case b), otherwise. Hence, M queries once $\mathsf{G}(1^k)$, $\mathsf{A}(1^k, i)$, P_i, D_i and V_i, has success probability $\mathsf{UN}_{\mathcal{A}}(M) = \varepsilon$ and works otherwise in constant time. As for opposite, the construction in Section 5 showed that not every collision-resistant attester is undeniable.

2) Let $\mathcal{A} = (\mathsf{G}, \mathsf{P}, \mathsf{D}, \mathsf{V})$ be an undeniable attester. By 1), $\mathcal{A}$ is also collision-resistant. Next we show that if $\mathcal{A}$ is collision-resistant, then $\mathcal{D} = (\mathsf{G}, \mathsf{D})$ is a CRHF on 2^{Σ^k} (i.e., on the subsets of Σ^k). Let $\mathsf{A} \in \mathcal{EA}$ be an adversary such that $\mathsf{CRH}_{\mathcal{D}}(\mathsf{A}) = \varepsilon$. Let M be the next machine. For an $i \in \mathsf{G}(1^k)$, M lets $S_1, S_2 \leftarrow \mathsf{A}(1^k, i)$. With probability ε_k, $S_1 \neq S_2$ but $\mathsf{D}_i(S_1) = \mathsf{D}_i(S_2) =: d$. Since $|S_1|, |S_2| = k^{O(1)}$, we can find efficiently an element w (w.l.o.g.) in $S_1 \setminus S_2$. Let $C := \mathsf{P}_i(w, S_1)$. By the definition of attesters, $\mathsf{V}_i(w, d, C) = \mathsf{V}_i(w, \mathsf{D}_i(S_1), \mathsf{P}_i(w, S_1)) = 1$. Thus, we have found a tuple (w, S_2, C) such that $w \notin S_2$ but $\mathsf{V}_i(w, \mathsf{D}_i(S_2), C) = 1$. A contradiction, and thus D_i is a CRHF on sets (i.e., on $2^{\Sigma^{|i|}}$, or alternatively, on concatenated strings $S[1]||\ldots||S[|S|]$, where $|S[j]| = |i|$ and for any $j < |S|$, $S[j] < S[j+1]$).

We finish the proof by constructing a CRHF $\mathcal{H} = (\mathsf{G}, \mathsf{H})$ on on input domain Σ^* as follows. Let $S = S[1]||S[2]||\ldots||S[n]$, $n \leq p(k)$, be an arbitrary string such that $|S[j]| = k - \log_2 n \leq k - \log_2 p(k)$ (it is sufficient to look at strings with length dividing $k - \log_2 p(k)$, due to the constructions [Dam89,Mer89]). Now define $\mathsf{H}_i(S[1]||\ldots||S[n]) := \mathsf{D}_i(x[1]||\ldots||x[n])$, where $x[j] = \langle j \rangle_{\log_2 p(k)}||S[j]$, where $\langle i \rangle_k$ denotes a k-bit binary fixed representation of $i \in \mathbb{N}$. Clearly if D is a CRHF on the domain 2^{Σ^k}, then $\mathcal{H}$ is a CRHF on domain Σ^*.

The opposite was implicitly proven by the construction in Section 5.