On Provably Secure Time-Stamping Schemes

Ahto Buldas

University of Tartu / Tallinn University of Technology / Cybernetica AS

Märt Saarepera

Independent

Security of Time Stamps: Overview

Time stamps – proofs that electronic records were created at certain time.

- *Before 1989* – trusted services that manage the security of time stamps
- *1989* – first attempt to construct a secure scheme [Haber, Stornetta]
- *1991* – proof sketch for a broadcast scheme [Benaloh, de Mare]
- *1997* – proof sketch for a centralized scheme [Haber, Stornetta]

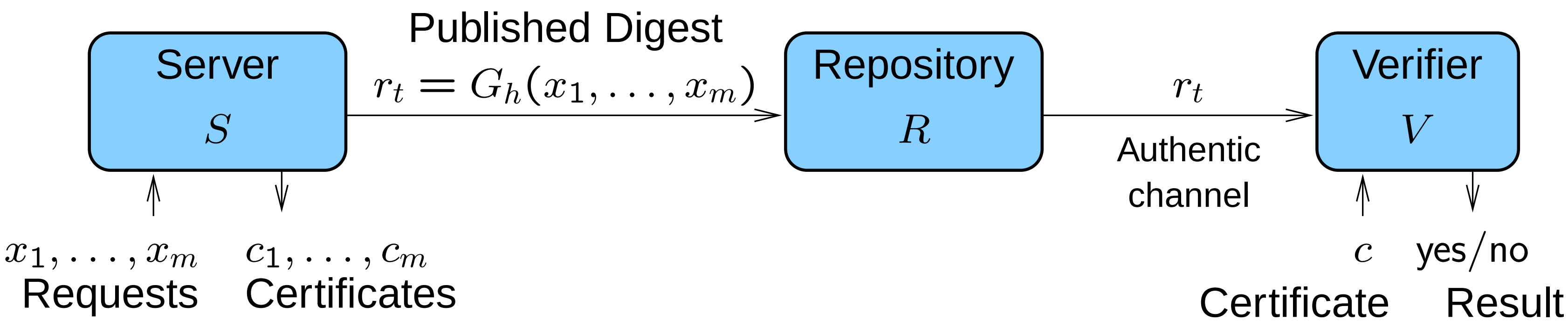
Regardless of the increasing practical importance of time-stamping, no precise security proofs have been presented.

Our Results

Our initial motivation was to complete the security proof outlined by Haber and Stornetta [1997].

- We show that the security condition presented by Haber and Stornetta is unattainable because it overlooks precomputation
- Inspired by a patent scenario, we derive a different security condition
- We modify the time stamp verification procedure
- We present a security proof for the modified scheme
- We argue the necessity of modifications – there are no black-box reductions otherwise

Hash-Based Time-Stamping Schemes



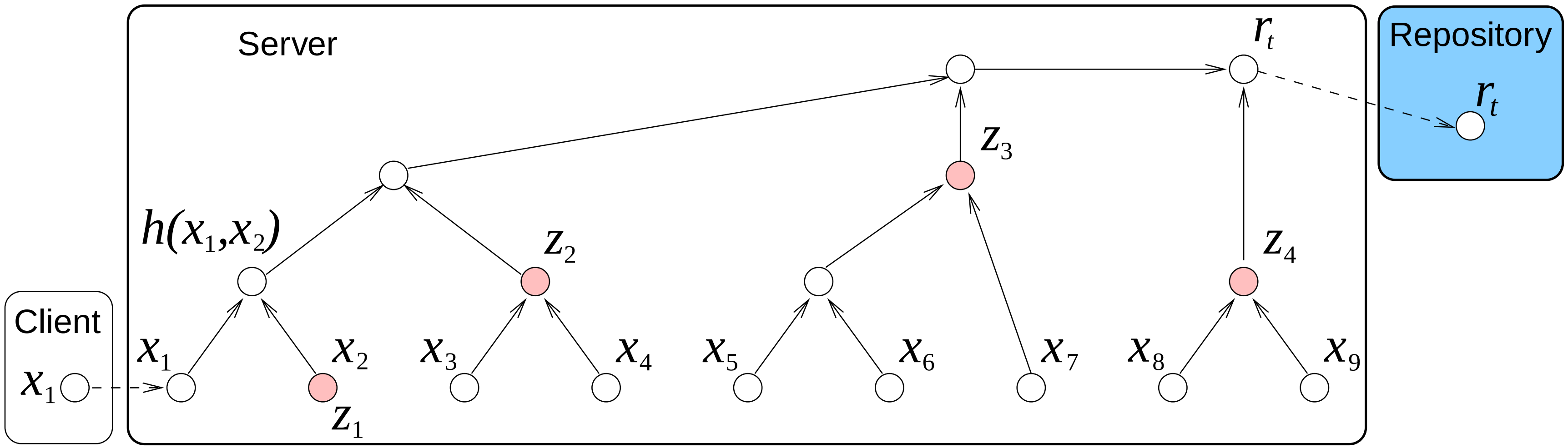
Server S – issues time stamps and publishes roundly digests.

Repository R – a write-only database for publishing roundly digests.

Verifier V – verifies time stamps.

Server Procedure

During the t-th round, S receives a list $x_1, \ldots, x_m$ of k-bit requests and computes the root $r_t = G_h(x_1, \ldots, x_m)$ of a hash tree and sends r_t to R.



S issues *time-certificates* $c = (x, t, n, z)$, where n is a ℓ-bit *identifier*, and $z = (z_1, z_2, \ldots, z_\ell)$.

Example: The certificate for x_1 is $(x_1, t, 0000, (z_1, z_2, z_3, z_4))$.

Verifier Procedure

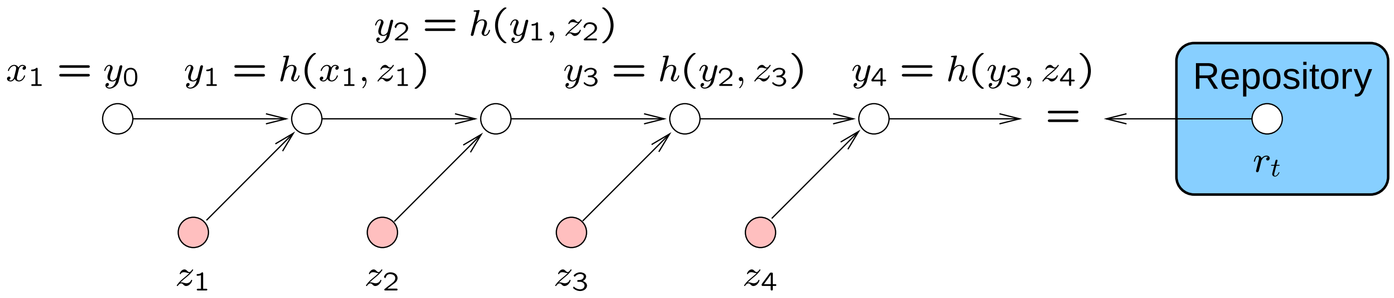
To verify a certificate (x, t, n, z), where $n = n_1 n_2 \ldots n_\ell$, a verifier:

- Obtains an authentic copy of r_t by querying R,
- Computes $(y_0, y_1, \ldots, y_\ell)$, where $y_0 := x$, and for $i = 1, \ldots, \ell$:

$$y_i := \begin{cases} h(z_i, y_{i-1}) & \text{if } n_i = 1 \\ h(y_{i-1}, z_i) & \text{if } n_i = 0 \end{cases} .$$

- Checks if $y_\ell \stackrel{\text{def}}{=} F_h(x; n; z) \stackrel{?}{=} r_t$.

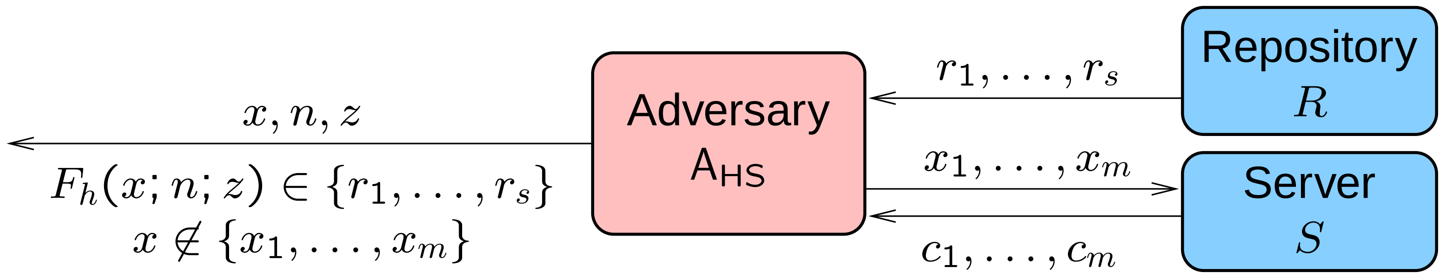
Example: The verification of$(x_1, t, 0000, (z_1, z_2, z_3, z_4))$:



Security condition

Adversary (Haber, Stornetta): Adversary A$_{\text{HS}}$ sends requests $x_1, \ldots, x_m$ to S, obtains digests $r_1, \ldots, r_s$ form R, and tries to find (x, t, n, z) so that

$$x \notin \{x_1, \ldots, x_m\} \quad \text{and} \quad F_h(x; n; z) = r_t \in \{r_1, \ldots, r_s\}.$$

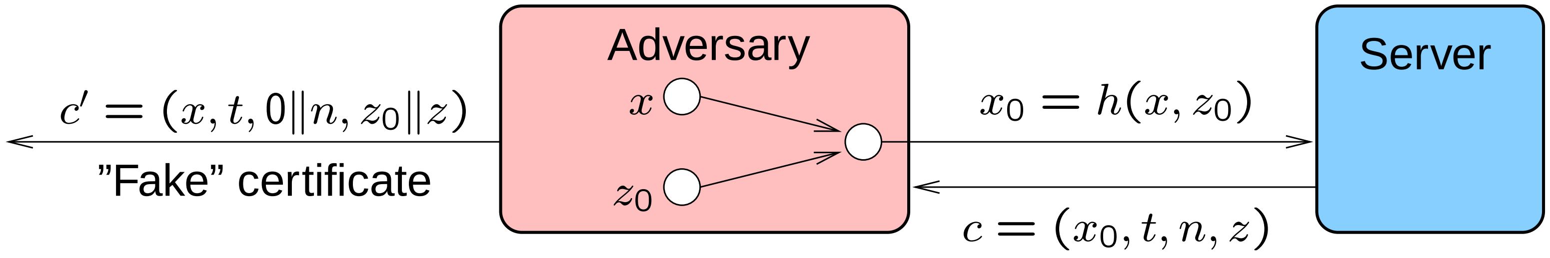


Security condition: Every poly-time A$_{\text{HS}}$ has negligible success probability.

The Security Condition is not Attainable!

The scheme above is insecure against the following behavior of A_{HS}:

- A_{HS} picks x and z_0 uniformly at random.
- A_{HS} sends $x_0 = h(x, z_0)$ to S and obtains $c = (x_0, t, n, z)$.
- A_{HS} computes a "fake" certificate $(x, t, 0\|n, z_0\|z)$.



By definition, $F_h(x; 0\|n; z_0\|z) = F_h(x_0; n; z) = r_t$. Hence, the attack is successful whenever $x \neq x_0$ (as far as $\{x_1, \ldots, x_q\} = \{x_0\}$).

If h has reasonable security properties then $\Pr[x \neq x_0]$ is non-negligible.

New Security Condition

- Bob, a criminal who steals inventions (in cooperation with S), computes $r_1, \ldots, r_s$ (not necessarily using G_h) that are stored in R.

- Alice, an inventor, creates a description X_A of her invention and time-stamps $x_A = \mathcal{H}(X_A)$. Some time later, X_A is disclosed to the public.

- Bob creates a slightly modified version X_B of the description (inventor's name should be replaced!) and computes $x = \mathcal{H}(X_B)$

- Bob tries to find (n, z), so that $F_h(x; n; z) \in \{r_1, \ldots, r_s\}$.

New security condition: For every poly-time $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ and for every poly-sampleable distribution $\mathcal{D}$ with Rényi entropy $\mathsf{H}_2(\mathcal{D}) = \omega(\log k)$:

$$\Pr[(\mathfrak{R}, a) \leftarrow \mathsf{A}_1(1^k), X \leftarrow \mathcal{D}, (n, z) \leftarrow \mathsf{A}_2(X, a) : F_h(\mathcal{H}(X); n; z) \in \mathfrak{R}] = k^{-\omega(1)}.$$

Security

Let $\mathfrak{N} \subset \{0,1\}^*$ (*set of valid identifiers*) and $|\mathfrak{N}| = k^{O(1)}$. $\mathfrak{N}$ can be viewed as a *hashing scheme* published by S before the service starts.

New verification procedure: To verify $c = (x, t, n, z)$ for $X \in \{0,1\}^*$, the verifier checks if $x = \mathcal{H}(X)$, $F_h(x; n; z) = r_t$, and $n \in \mathfrak{N}$.

New definition for the success probability of A:

$$\Pr[(\mathfrak{R}, \mathfrak{N}, a) \leftarrow \mathsf{A}_1(1^k),\ X \leftarrow \mathcal{D},\ (n, z) \leftarrow \mathsf{A}_2(X, a) : F_h(\mathcal{H}(X); n; z) \in \mathfrak{R},\ n \in \mathfrak{N}]$$

Theorem 1: If h and $\mathcal{H}$ are collision-resistant, then the time-stamping scheme is secure relative to every polynomially sampleable $\mathcal{D}$ with Rényi entropy $\mathsf{H}_2(\mathcal{D}) = \omega(\log k)$.

Proof Sketch

Oops ... again a sketch!?

Proof of Theorem 1: Having A $=$ (A$_1$, A$_2$) with ratio $T(k)/\delta(k)$, we construct a collision-finder A$'$ for h with ratio $\frac{T'(k)}{\delta'(k)} = k^{O(1)}\left(\frac{T(k)}{\delta(k)}\right)^2$.
- A$'$ calls A$_1$ to obtain $\mathfrak{R}$, $\mathfrak{N}$, and a;
- A$'$ picks $X, X' \leftarrow \mathcal{D}$ and computes $(n, z) \leftarrow$ A$_2(X, a)$, $(n', z') \leftarrow$ A$_2(X', a)$;
- A$'$ simulates $F_h(\mathcal{H}(X); n; z)$ and $F_h(\mathcal{H}(X'); n'; z')$.
- If $F_h(\mathcal{H}(X); n; z) = F_h(\mathcal{H}(X'); n'; z')$, $\mathcal{H}(X) \neq \mathcal{H}(X')$, and $n = n'$ then A$'$ checks the h-calls and outputs a collision for h.

We prove (Lemma 1) that if $x \neq x'$ and $F_h(x; n; z) = F_h(x'; n; z')$ then the h-calls of $F_h(x; n; z)$ and $F_h(x'; n; z')$ comprise a collision.

It can be shown (Lemma 2) that the success of A$'$ is at least

$$\frac{\delta^2(k)}{T^2(k)} - 2^{-\mathsf{H}_2(\mathcal{H}(\mathcal{D}))} = \frac{\delta^2(k)}{T^2(k)} - k^{-\omega(1)}. \qquad \square$$

Security Proofs and Oracle Separation

Semi black-box reduction: $\forall_{\mathsf{pol}} A_2 \exists_{\mathsf{pol}} \mathsf{A}_1$: A_2^h breaks $\mathsf{TS}^h \Rightarrow \mathsf{A}_1^h$ breaks h.

Black-box reduction: $\exists_{\mathsf{pol}} S \forall \mathsf{A}$: A breaks $\mathsf{TS}^h \Rightarrow S^{\mathsf{A},h}$ breaks h.

Separation: If h is collision-resistant relative to $\mathcal{O}$ but TS^h is insecure relative to $\mathcal{O}$, then there exist no black-box reductions.
Strong separation: If in addition, $\mathcal{O} = \pi^h$ for a poly-time π, then there exist no semi black-box reductions.

For more details: Omer Reingold, Luca Trevisan, and Salil Vadhan. Notions of reducibility between cryptographic primitives. In TCC'04, LNCS 2951, pp.1–20. Feb. 2004.

Necessity of the Modified Verification

We prove that semi black-box reductions are insufficient for proving the security of the unmodified time-stamping scheme, based on the collision-resistance of h (and $\mathcal{H}$).

We construct an oracle $\mathcal{O}$ relative to which there exists a collision-resistant hash function $h^{\mathcal{O}}\colon \{0,1\}^{2k} \to \{0,1\}^{k}$ and a poly-time $(\mathsf{A}_1^{\mathcal{O}}, \mathsf{A}_2^{\mathcal{O}})$ with

$$\Pr[r \leftarrow \mathsf{A}_1^{\mathcal{O}}, x \leftarrow \mathcal{D}, (n,z) \leftarrow \mathsf{A}_2^{\mathcal{O}}(x,r)\colon F_{h^{\mathcal{O}}}(x;n;z) = r] = 1$$

for every distribution $\mathcal{D}$ on $\{0,1\}^k$. Hence, $h^{\mathcal{O}}$ makes the unmodified time-stamping scheme insecure. (*Rules out black-box reductions*)

We construct a hash function oracle $\mathfrak{H}_k\colon \{0,1\}^{2k} \to \{0,1\}^{k}$, which is collision-resistant relative to itself but $\mathfrak{H}_{4k}$ can be used to break the time-stamping scheme that uses $\mathfrak{H}_k$. (*Rules out semi black-box reductions*)

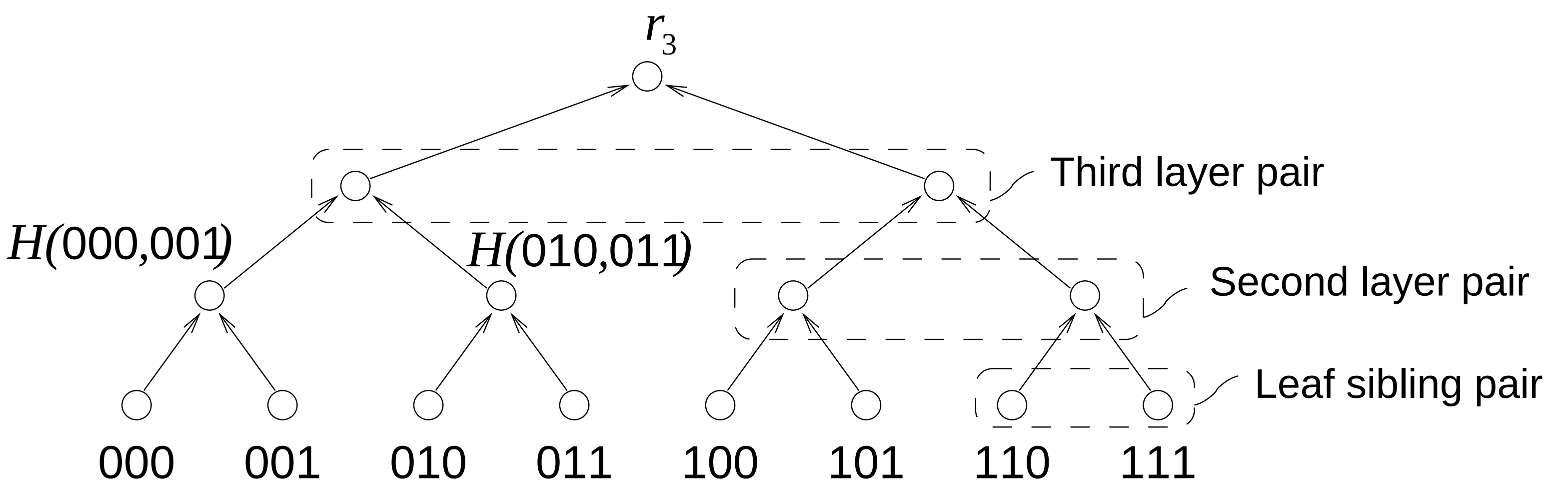
Construction of $\mathcal{O}$

$\mathcal{O}$ comprises a random function $H \leftarrow \mathfrak{F}$ and responds to:

• *H-queries:* on input $(x_1, x_2) \in \{0,1\}^{2k}$ return $H(x_1, x_2) \in \{0,1\}^k$.

• A$_1$*-queries:* on input 1^k return the root r_k of the *complete Merkle tree* M_k, the leaves of which are all k-bit strings in lexicographic order.

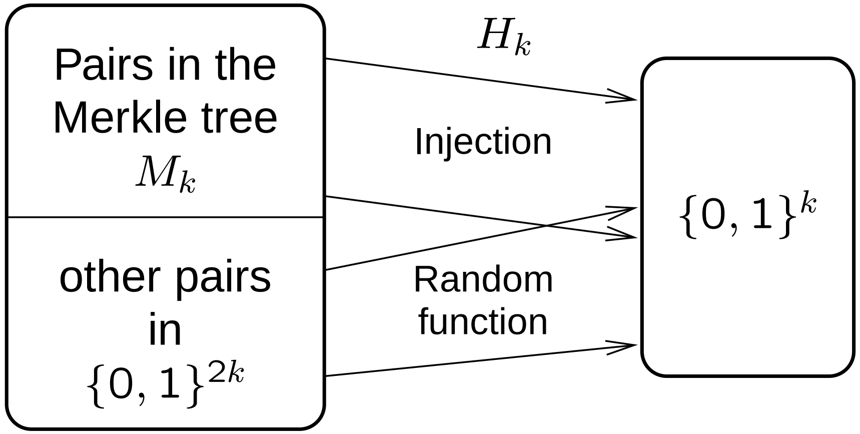
• A$_2$*-queries:* on input $x \in \{0,1\}^k$ find $z \in (\{0,1\}^k)^k$ (based on M_k) so that $F_H(x; x; z) = r_k$ and output (x, z).



Choice of H

We define $\mathfrak{F}$ as a set of all functions H, such that for all k:

- all non-leaf vertices in M_k contain different elements of $\{0,1\}^k$
- all sibling-pairs in M_k are different.

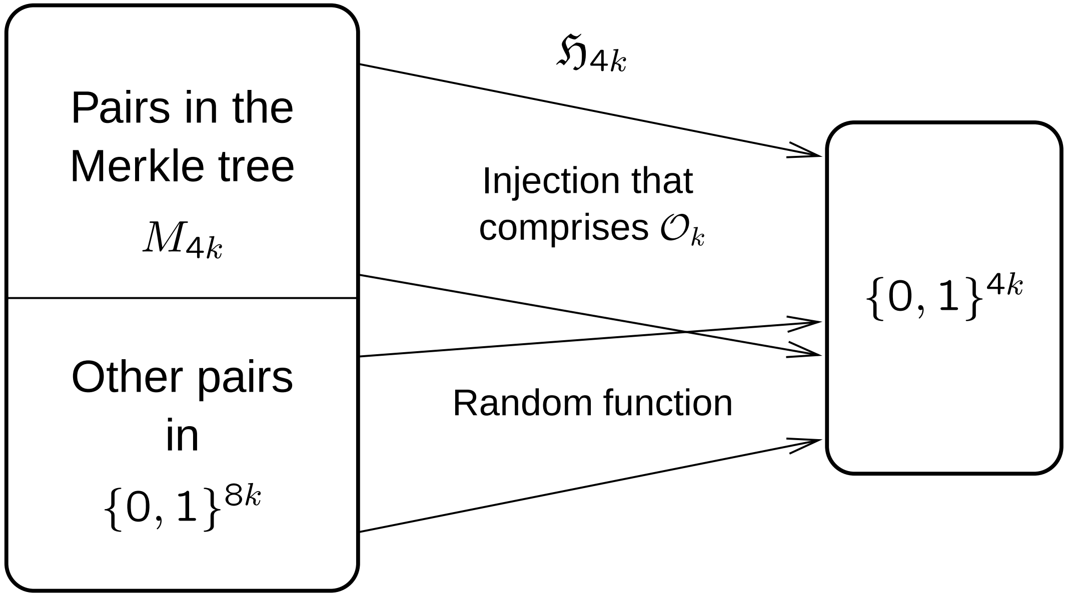


Lemma 5: Every collision-finding adversary $A^{\mathcal{O}}$ for H that makes polynomial number of oracle calls, has negligible success.

Construction of $\mathfrak{H}$

The oracle $\mathcal{O}$ does not yet rule out semi black-box reductions – computation of $\mathcal{O}$ requires an exponential number of H-calls, and hence $\mathcal{O} \neq \pi^H$.

We embed $\mathcal{O}_k$ into a hash function $\mathfrak{H}_{4k}$: $\{0,1\}^{8k} \rightarrow \{0,1\}^{4k}$:



Open question 1: More Efficient Reductions?

The reduction obtained is *poly-preserving*: $\frac{T'(k)}{\delta'(k)} = k^{O(1)} \cdot \left(\frac{T(k)}{\delta(k)}\right)^2$.

Practical guarantees are limited: If the time-stamping scheme is broken with ratio $\frac{T(k)}{\delta(k)} = 2^{32}$ (very efficiently!) then the reduction implies that h with 160-bit output can be broken with ratio 2^{81}, which is trivially true.

The reduction gives practical security guarantees only in case $k > 400$ – much larger than in the existing schemes.

Question: Are there more efficient reductions?
For example, *linear-preserving reductions*: $\frac{T'(k)}{\delta'(k)} = k^{O(1)} \cdot \frac{T(k)}{\delta(k)}$.

Open question 2: General black-box constructions?

Is it possible to construct a hash function $H = P^h$ so that if h is collision resistant then the hash-based time-stamping schemes constructed from H are secure?

Can we prove that there are no general black-box reductions of secure time-stamping schemes to collision-resistant hash functions?
An obstacle: If an oracle $\mathcal{O}$ is able to compute the root of the complete Merkle tree M_k^f for any (computable) f, then $\mathcal{O}$ can be "abused" to find collisions for any hash function.

Open question 3: Stronger security conditions?

In our security condition, A has unconditional uncertainty about $x \leftarrow \mathcal{D}$.

In practice, it is possible that A_1 has some partial knowledge $y = f(x)$ about x (e.g. ciphertexts or signatures).

This suggests conditions of type: If x can be time-stamped based on $y = f(x)$, then x can be efficiently computed based on y.

Main problem:

- $x_1 = h(x, z_0)$ (where $z_0 \leftarrow_{\mathsf{R}} \{0,1\}^k$) is *partial knowledge about* x and is sufficient to time stamp x.
- If h is one-way, x cannot be computed from x_1.

T h a n k

Y o u !