On Provably Secure Time-Stamping Schemes

Ahto Buldas[1,2,3,⋆] and Märt Saarepera[4]

[1] University of Tartu, Liivi 2, 50409 Tartu, Estonia. Ahto.Buldas@ut.ee
[2] Cybernetica, Akadeemia tee 21, 12618 Tallinn, Estonia.
[3] Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia.
[4] Independent researcher. marts@neoteny.com

Abstract. It is almost a folklore-knowledge that hash-based time-stamping schemes are secure if the underlying hash function is *collision-resistant* but still no rigorous proofs have been published. We try to establish such proof and conclude that the existing security conditions are improper because they ignore precomputations by adversaries. After analyzing a simplistic patent filing scenario, we suggest a new security condition for time-stamping schemes that leads to a new security property of hash functions – *chain-resistance*. We observe that if the variety of possible shapes of hash-chains is polynomial (and the verification procedure is suitably improved), then the time-stamping scheme becomes provably secure, assuming that the underlying hash function is collision-resistant. Finally, we show that in some sense, the restrictions in the security definition are necessary – conventional black-box techniques are unable to prove that chain-resistance follows from collision-resistance.

1 Introduction

The main goal of digital time-stamping is to prove that electronic data-items were created or registered at a certain time. A simple method is to use a trusted service (with a precise clock) that provides data items with current time value and digitally signs them. The assumption of unconditionally trusted service hides a risk of possible collusions that may not be acceptable in applications. The risks are especially high in centralized applications like electronic patent- or tax filing as well as in electronic voting, where the possible collusions are related to direct monetary (or even political) interests.

First attempts to eliminate trusted services from time-stamping schemes were made in [4], where cryptographic hash functions and publishing were used to replace electronic signatures. To date, several improvements of hash-based time-stamping schemes have been presented [1–3]. Such schemes have been used in business applications and are even included in international standards [9].

The combined monetary value of electronic content (insured, in particular, with time stamps) increases over time and so does the risk associated with it. A decision of a content manager to start using a certain time-stamping service for protecting electronic records must involve the assessment of long-term security

⋆ Supported by Estonian SF grant no. 5113

risks. Desirably, such assessments should be based on analytical arguments. As an example of such argument, modern cryptography can prove that there are no *structural flaws* (or *principal design errors*) in security solutions, assuming that their basic building blocks (such as hash functions) are secure. The use of provably secure time-stamping schemes can avoid many practical risks.

Regardless of the growing importance of hash-based time-stamping schemes, their security is only superficially studied in scientific literature. In [5], a formal security condition for hash-based time-stamping schemes was presented and an informal sketch of a security proof was outlined. Though no rigorous proofs were presented it has become almost a public myth that the security of hash-based time-stamping schemes can be reduced to the *collision-resistance* of underlying hash functions. Thus far, no more related studies have been published.

In this paper, we revisit the security analysis of hash-based time-stamping schemes [5]. We observe that the formal security condition stated in [5] is unreachably strong because it overlooks pre-computations of the adversary.

Inspired by a simplistic patent filing scenario, we present a new security condition for time-stamping schemes that leads to a new security condition for hash functions – *chain-resistance* – necessary for the scheme [5] to be secure. We show that *additional checks* in the verifying procedure render the conventional time-stamping schemes provably secure in the new sense, based on *collision-resistance* of the hash function. The additions concern an examination of whether the shape of the hash-chain included into a time stamp belongs to a certain (polynomial) set of templates. This may seem a minor detail but as no currently used time-stamping schemes implement it, none of them are provably secure.

We further examine the necessity of said additional checks in the verification procedure and prove that without these checks *it is probably very hard (if not impossible) to prove the security of the schemes of type [5] based on collision-resistance alone*. We present an oracle relative to which there exist collision-resistant hash functions which are not chain-resistant. Almost all security proofs *relativize* – are valid relative to any oracle. Therefore, any security proof of the unmodified schemes should use either non-standard (non-relativizing) proof techniques or stronger/incomparable security assumptions on the underlying hash function. For example, it is easy to prove that entirely random hash functions (*random oracles*) are chain-resistant. In practice, it is often assumed that SHA-1 and other hash functions behave like random oracles which means that in such setting, their use in practical time-stamping schemes is justified. At the same time, it is still possible that a time-stamping scheme that uses SHA-1 is totally insecure while no collisions are found for SHA-1.

2 Notation and Definitions

By $x \leftarrow \mathcal{D}$ we mean that x is chosen randomly according to a distribution $\mathcal{D}$. If A is a probabilistic function or a Turing machine, then $x \leftarrow \mathsf{A}(y)$ means that x is chosen according to the output distribution of A on an input y. By $\mathcal{U}_n$ we denote the uniform distribution on $\{0,1\}^n$. If $\mathcal{D}_1, \ldots, \mathcal{D}_m$ are distributions and

$F(x_1,\ldots,x_m)$ is a predicate, then $\mathsf{Pr}[x_1 \leftarrow \mathcal{D}_1,\ldots,x_m \leftarrow \mathcal{D}_m\colon F(x_1,\ldots,x_m)]$ denotes the probability that $F(x_1,\ldots,x_m)$ is true after the ordered assignment of $x_1,\ldots,x_m$. We write $f(k) = O(g(k))$ if there are $c, k_0 \in \mathbb{R}$, so that $f(k) \leq cg(k)$ $(\forall k > k_0)$. We write $f(k) = \omega(g(k))$ if $g(k) = O(f(k))$ but $f(k) \neq O(g(k))$. A function $f\colon \mathbb{N} \to \mathbb{R}$ is *negligible* if $f(k) = k^{-\omega(1)}$. A Turing machine M is *polynomial-time* (*poly-time*) if it runs in time $k^{O(1)}$, where k denotes the input size. Let F^* be the class of all functions $f\colon \{0,1\}^* \to \{0,1\}^*$. Let FP be the class of all functions $f \in \mathsf{F}^*$ computable by poly-time Turing machines M. A distribution $\mathcal{D}$ on $\{0,1\}^*$ is *polynomially sampleable* if it is an output distribution of a poly-time Turing machine.

By an *oracle Turing machine* we mean an incompletely specified Turing machine S that comprises calls to *oracles*. The description can be completed by defining the oracle as a function $\mathcal{O} \in \mathsf{F}^*$. In this case, the machine is denoted by $S^{\mathcal{O}}$. An oracle $\mathcal{O}$ is not necessarily computable but may still have assigned a conditional (worst-case) running time $t(k)$, which may or may not reflect the actual amount of computations performed by $\mathcal{O}$ internally. Running time of $S^{\mathcal{O}}$ comprises the conditional worst-case running time $t(k)$ of oracle calls – each call takes $t(k)$ steps. An oracle $\mathcal{O}$ is *poly-time* if $t(k) = k^{O(1)}$. We say that S is a *poly-time oracle machine*, if the running time of $S^{\mathcal{O}}$ is polynomial, whenever $\mathcal{O}$ is poly-time. Let $\mathsf{FP}^{\bullet}$ denote the class of all poly-time oracle machines. Let $\mathsf{FP}^{\mathcal{O}}$ be the class of all functions computable by poly-time oracle machines $S^{\mathcal{O}}$.

A *primitive* $\mathfrak{P}$ is a class of (not necessarily computable by ordinary Turing machines) functions intended to perform a security related task (e.g. data confidentiality, integrity etc.). Each primitive $\mathfrak{P}$ is characterized by the success $\delta(k)$ of an adversary A. For example, a *collision-resistant hash function* is a function family $\{h_k\}_{k\in\mathbb{N}}$, where $h_k\colon \{0,1\}^{2k} \to \{0,1\}^k$ and

$$\delta(k) = \mathsf{Pr}[(x,x') \leftarrow \mathsf{A}(1^k)\colon x \neq x',\, h_k(x) = h_k(x')] \ .$$

In more rigorous definitions [12], h_k is randomly chosen from a set $\mathfrak{F} \subseteq \mathsf{F}^*$. Otherwise A may output a fixed collision. We write $h(x)$ instead of $h_k(x)$. Sometimes, we need hash functions $\mathcal{H} = \{\mathcal{H}_k\}_{k\in\mathbb{N}}$ of type $\mathcal{H}_k\colon \{0,1\}^* \to \{0,1\}^k$. An adversary $\mathsf{A} \in \mathsf{F}^*$ *breaks* $f \in \mathfrak{P}$ (and write $\mathsf{A} \underset{\mathfrak{P}}{\mathsf{breaks}} f$) if A has non-negligible success. An instance $f \in \mathfrak{P}$ is *secure* if no $\mathsf{A} \in \mathsf{FP}$ breaks f. An instance $f \in \mathfrak{P}$ is *secure relative to an oracle* $\mathcal{O}$ if no $\mathsf{A} \in \mathsf{FP}^{\mathcal{O}}$ breaks f.

3 Time-Stamping Schemes and Their Security

3.1 The Scheme of Haber and Stornetta

A time-stamping scheme [5] involves three parties: a *Client* C, a *Server* S, and a *Repository* $\mathfrak{R}$; and two procedures for *time-stamping* a data item and for *verifying* a time stamp (Fig. 1). It is assumed that $\mathfrak{R}$ is write-only and receives items from S in an authenticated manner.

Time-stamping procedure is divided into rounds of equal duration. During each round, S receives requests from Clients. For simplicity, all requests $x_1, \ldots, x_m$ are assumed to be bit-strings $x_i \in \{0,1\}^k$. If the t-th round is over, S computes a compound hash $r_t \in \{0,1\}^k$ by using a function $h\colon \{0,1\}^{2k} \to \{0,1\}^k$ and a tree-shaped hashing scheme $r_t = G_h(x_1, \ldots, x_m)$. For example, if the requests of the t-th round are x_1, x_2, x_3, x_4, then S may compute $r_t = h(x_1, h(h(x_2, x_3), x_4))$. Next, S sends r_t to $\mathfrak{R}$ (in a secure way), where it is stored as a pair (t, r_t).

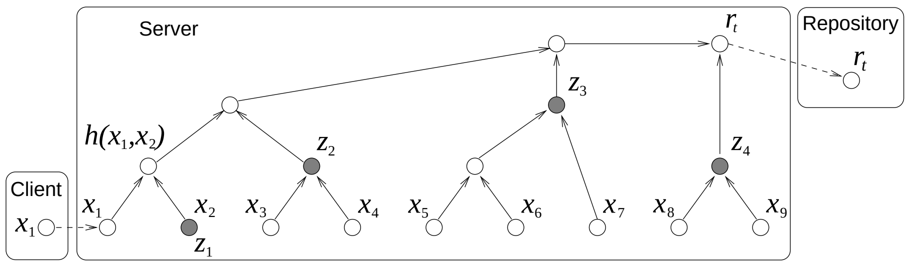

Fig. 1. The scheme of Haber and Stornetta (a simplified model).

After that, S issues for each request x a *time-certificate* $c = (x, t, n, z)$, where t is current time value, n is an identifier $n = n_1 n_2 \ldots n_\ell \in \{0,1\}^\ell$, and z is a sequence $z = (z_1, z_2, \ldots, z_\ell) \in \left(\{0,1\}^k\right)^\ell$. In the scheme of Fig. 1, the time-certificate for x_1 is $(x_1, t, \mathsf{0000}, (z_1, z_2, z_3, z_4))$, where $z_1 = x_2$, $z_2 = h(x_3, x_4)$, $z_3 = h(h(x_5, x_6), x_7)$, and $z_4 = h(x_8, x_9)$.

Verification procedure is performed by C as follows. To verify (x, t, n, z), C computes a hash value r'_t by using h and a $F_h(x; n; z)$, which computes a sequence $y = (y_0, y_1, \ldots, y_\ell) \in \left(\{0,1\}^k\right)^\ell$ inductively, so that $y_0 := x$, and

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

for $i > 0$, and outputs $r'_t = F_h(x; n; z) := y_\ell$. Second, C sends a query t to $\mathfrak{R}$ and obtains r_t as a reply. Finally, C checks whether $r'_t = r_t$. Note that n and z can be equal to empty string $\sqcup$, in which case $F_h(x; n; z) = x$.

Security condition [5] states that the time-stamping scheme above is secure against $\mathsf{A}_{\mathrm{HS}} \in \mathsf{FP}$ that sends requests $x_1, \ldots, x_q$ to S and queries to $\mathfrak{R}$. As a result, A_{HS} outputs a time-certificate (x, t, n, z), where $x \in \{0,1\}^k$, $n \in \{0,1\}^\ell$, and $z \in (\{0,1\}^k)^\ell$. The attack is considered successful, if $x \notin \{x_1, \ldots, x_q\}$ and $F_h(x; n; z) = r_t$, where r_t is assumed to be the correct response of $\mathfrak{R}$ to query t.

3.2 Analysis of the Security Condition

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

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

where $\|$ denotes concatenation. By definition, $F_h(x; \mathsf{0}\|n; z_0\|z) = F_h(x_0; n; z) = r_t$. Hence, the attack is successful whenever $x \neq x_0$ because x_0 was the only request made by A_{HS}. If h has reasonable security properties then $\mathsf{Pr}[x \neq x_0]$ is non-negligible. [5] This "attack" shows that the formal security definition does not follow the intuition behind time-stamping, because it overlooks the possibility of precomputations. As a success criterion, the condition $x \notin \{x_1, \ldots, x_q\}$ is improper because the notion of *already time-stamped items* is not sufficiently precise.

4 New Security Condition and Improved Schemes

4.1 New Security Condition

The new security condition is inspired by the following simplistic attack-scenario, where Bob, a criminal who steals inventions, co-operates with a server S:

- Bob precomputes (not necessarily with G_h) some hash values $r_1, \ldots, r_s$ that may help him to back-date documents in the future. His collaborator S sends the hash values to $\mathfrak{R}$, where they are stored as pairs $(t_1, r_1), \ldots, (t_s, r_s)$.
- Alice, an inventor, creates a description $X_A \in \{0,1\}^*$ of her invention and requests a time certificate for $x_A = \mathcal{H}(X_A)$, where $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^k$ is a (collision-resistant) hash function.
- Some time later, the invention is disclosed to the public and Bob tries to steal the rights to Alice's invention. He creates a slightly modified version X_B of X_A (at least the author's name should be replaced), and tries to back-date it relative to X_A. Bob is successful, if he finds n and z, so that $F_h(x_B; n; z) \in \{r_1, \ldots, r_s\}$. Bob can use (x_B, t, n, z) to claim his rights to the invention.

In order to formalize such attack scenario, a two-staged adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ is needed. The first stage A_1 *precomputes* a set $\mathfrak{R} = \{r_1, \ldots, r_s\}$ after which the second stage A_2 obtains a *new document* $X \in \{0,1\}^*$ ("a document, unknown to mankind before") and tries to back-date it by finding n and z, so that

[5] If h is collision-resistant, $\mathsf{Pr}[x = x_0] = \mathsf{Pr}[x, z_0 \leftarrow \mathcal{U}_k\colon h(x, z_0) = z] = \delta$, and $p_x = \mathsf{Pr}[x' \leftarrow \mathcal{U}_k\colon h(x, x') = x]$, then the collision-finding adversary $(xx', xx'') \leftarrow \mathsf{A}(1^k)$ (where $x, x', x'' \leftarrow \mathcal{U}_k$ are independent random bit-strings) has non-negligible success. Indeed, the probability δ' that A outputs a collision for h (possibly, with $x' = x''$) is $\delta' \geq \sum_x \mathsf{Pr}[x] \cdot p_x^2 \geq \left(\sum_x \mathsf{Pr}[x] \cdot p_x\right)^2 = \delta^2$, where $\mathsf{Pr}[x] = \mathsf{Pr}[X \leftarrow \mathcal{U}_k\colon X = x]$. Hence, the overall success of A is at least $\delta^2 - 2^{-k}$.

$F_h(\mathcal{H}(X);n;z) \in \mathfrak{R}$. The term *new document* is hard to define formally. As we saw, the condition $\mathcal{H}(X) \notin \{x_1,\ldots,x_q\}$ does not guarantee that X is really *new*. We assume that X is chosen according to a distribution $\mathcal{D}$ on $\{0,1\}^*$ that is somewhat unpredictable to A. The success of A is defined as follows:

$$\delta(k) = \mathsf{Pr}[(\mathfrak{R},a)\leftarrow \mathsf{A}_1(1^k), X\leftarrow \mathcal{D}, (n,z)\leftarrow \mathsf{A}_2(X,a)\colon F_h(\mathcal{H}(X);n;z)\in\mathfrak{R}] \ , \quad (2)$$

where a denotes a state information sent from A_1 to A_2. Note that (2) can be simplified by assuming $\mid \mathfrak{R} \mid = 1$, because this reduces $\delta(k)$ only polynomially.

Necessary conditions for $\mathcal{D}$. Intuitively, the prediction of $\mathcal{D}$ must require super-polynomial time-success ratio, i.e. every A with running time $T(k)$ and success $\delta(k) = \mathsf{Pr}[\mathfrak{R} \leftarrow \mathsf{A}(1^k), x \leftarrow \mathcal{D}\colon x \in \mathfrak{R}]$ has time-success ratio $\frac{T(k)}{\delta(k)} = k^{\omega(1)}$. In case $\mathcal{D}$ is *polynomially sampleable*, an equivalent assumption is that

$$\mathsf{P}_{\mathrm{C}}(\mathcal{D}) = \mathsf{Pr}[y \leftarrow \mathcal{D}, x \leftarrow \mathcal{D}\colon x = y] = k^{-\omega(1)}, \quad (3)$$

where $\mathsf{P}_{\mathrm{C}}(\mathcal{D})$ is the *collision probability* of $\mathcal{D}$. Indeed, if for a poly-time A we have $\mathsf{Pr}[\mathfrak{R}\leftarrow\mathsf{A}(1^k), x\leftarrow\mathcal{D}\colon x\in\mathfrak{R}] = T(k)\cdot k^{-O(1)}$, then there is $\mathfrak{R}_\mathsf{o} \subseteq \{0,1\}^k$, so that $|\mathfrak{R}_\mathsf{o}| = k^{O(1)}$ and $\mathsf{Pr}[x\leftarrow\mathcal{D}\colon x\in\mathfrak{R}_\mathsf{o}] = k^{-O(1)}$. Thus,

$$\exists r\in\mathfrak{R}_\mathsf{o}\colon p = \mathsf{Pr}[x\leftarrow\mathcal{D}\colon x = r] = |\mathfrak{R}_\mathsf{o}|^{-1} \cdot k^{-O(1)} = k^{-O(1)}$$

and hence $\mathsf{P}_C(\mathcal{D}) \geq p^2 = k^{-O(1)}$. If, in turn, $\mathsf{P}_{\mathrm{C}}(\mathcal{D}) = k^{-O(1)}$, then every A with output distribution $\mathcal{D}$ has success $\delta(k) = k^{-O(1)}$.

The condition (3) is equivalent to the requirement that $\mathcal{D}$ has *Rényi entropy* $\mathsf{H}_2(\mathcal{D}) = -\log_2 \mathsf{P}_{\mathrm{C}}(\mathcal{D}) = \omega(\log k)$, and is in fact *necessary* for a time-stamping scheme to be secure relative to $\mathcal{D}$. Indeed, if A_1 is defined to output $y \leftarrow \mathcal{D}$ and $(\sqcup, \sqcup) \leftarrow \mathsf{A}_2(x,a)$ (for any x and a), then $(\mathsf{A}_1,\mathsf{A}_2)$ has success $\mathsf{P}_{\mathrm{C}}(\mathcal{D})$.

Chain-resistant hash functions. The security definition (2) implies that h must satisfy the following new security condition for hash functions:

Definition 1. *A hash function h is* chain resistant (relative to a distribution $\mathcal{D}_k$ on $\{0,1\}^k$)*, if for every adversary $\mathsf{A} = (\mathsf{A}_1,\mathsf{A}_2) \in \mathsf{FP}$:*

$$\mathsf{Pr}[(\mathfrak{R},a)\leftarrow\mathsf{A}_1(1^k), x\leftarrow\mathcal{D}_k, (n,z)\leftarrow\mathsf{A}_2(x,a)\colon F_h(x;n;z)\in\mathfrak{R}] = k^{-\omega(1)} \ . \quad (4)$$

It is easy to show that if a time-stamping scheme is secure relative to $\mathcal{D}$, then the hash function h is chain-resistant relative to $\mathcal{H}(\mathcal{D})$.

4.2 Improved Verification Procedure

We will prove later that the conventional black-box techniques are insufficient to imply chain-resistance from collision-resistance, and hence, also the security of time-stamping schemes in the sense of (2) cannot be proved under the collision-resistance condition alone. We modify the verification procedure in a way that prevents the adversary from finding chains for h without finding collisions as by-products. We restrict the set $\mathfrak{N} \subset \{0,1\}^*$ of identifiers (possible shapes of hash chains) that are considered valid by the verification procedure and show that if $|\mathfrak{N}| = k^{O(1)}$, then the collision-resistance of h is sufficient for a time-stamping scheme to be secure. The modified verification procedure is defined as follows:

New verification procedure. To verify a time-certificate (x, t, n, z) for $X \in \{0,1\}^*$, C checks if $x = \mathcal{H}(X)$, computes a hash value r'_t using $F_h(x; n; z)$ defined by (1), sends a query t to $\mathfrak{R}$ to obtain r_t, and checks if $r'_t = r_t$ and $n \in \mathfrak{N}$.

To be usable in practical time-stamping, the condition $n \in \mathfrak{N}$ must be efficiently verifiable. One way to achieve this is to set $\mathfrak{N} = \{0,1\}^{k_0}$, where k_0 is constant, which means that $n \in \mathfrak{N}$ is equivalent to $\|n\| = k_0$ and is naturally efficiently computable. The set $\mathfrak{N}$ can be viewed as a template of a *hashing scheme* that is published by the service provider before the service starts. As we consider service providers as possible adversaries, $\mathfrak{N}$ is created by an adversary. The restrictions above lead us to a weaker condition with the following notion of success:

$$\mathsf{Pr}[(\mathfrak{R},\mathfrak{N},a) \leftarrow \mathsf{A}_1(1^k), X \leftarrow \mathcal{D}, (n, z) \leftarrow \mathsf{A}_2(X, a) \colon F_h(\mathcal{H}(X); n; z) \in \mathfrak{R}, n \in \mathfrak{N}] \ . \quad (5)$$

4.3 Proof of Security

We prove that the security of the modified hash-based time-stamping schemes follows from the collision-resistance of h and $\mathcal{H}$.

Definition 2. *Let $y = (y_0, y_1, \ldots, y_\ell)$ and $y' = (y'_0, y'_1, \ldots, y'_{\ell'})$ be two sequences produced by $F_h(x; n; z)$ and $F_h(x'; n'; z')$ respectively, by using (1). Let $z = (z_1, \ldots, z_\ell)$, $z' = (z'_1, \ldots, z'_{\ell'})$, $n = n_1 \ldots n_\ell$, and $n' = n'_1 \ldots n'_{\ell'}$. We say that sequences y and y' comprise a collision, if for some indices $i \in \{1, \ldots, \ell\}$ and $j \in \{1, \ldots, \ell'\}$, $h(a, b) = y_i = y'_j = h(a', b')$, but $(a, b) \neq (a', b')$, where*

$$(a, b) = \begin{cases} (z_i, y_{i-1}) \text{ if } n_i = 1 \\ (y_{i-1}, z_i) \text{ if } n_i = 0 \end{cases} \quad \text{and} \quad (a', b') = \begin{cases} (z'_j, y'_{j-1}) \text{ if } n'_j = 1 \\ (y'_{j-1}, z'_j) \text{ if } n'_j = 0 \end{cases} .$$

Lemma 1. *If $x \neq x'$ and $F_h(x; n; z) = F_h(x'; n; z')$, then the sequences y and y' computed internally by $F_h(x; n; z)$ and $F_h(x'; n; z')$ comprise a collision.*

Proof. Let ℓ be the bit-length of n. As $x = y_0 \neq y'_0 = x'$ and $y_\ell = F_h(x; n; z) = F_h(x'; n; z') = y'_\ell$, there exists $i \in \{1, \ldots, \ell\}$, such that $y_i = y'_i$ and $y_{i-1} \neq y'_{i-1}$. Hence, either $h(z_i, y_{i-1}) = h(z'_i, y'_{i-1})$ or $h(y_{i-1}, z_i) = h(y'_{i-1}, z'_i)$. In both cases, we have a collision. $\square$

Theorem 1. *If h and $\mathcal{H}$ are collision-resistant, then the time-stamping scheme is secure in the sense of (5) relative to every polynomially sampleable $\mathcal{D}$ with Rényi entropy $\mathsf{H}_2(\mathcal{D}) = \omega(\log k)$.*

Proof. Let $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ be an adversary with running time $T(k)$ that breaks the time-stamping scheme in the sense of (5) with success $\delta(k)$. Assuming the collision-resistance of $\mathcal{H}$, we construct a collision-finding adversary A' for h with running time $T'(k) = k^{O(1)}T(k)$ and with success $\delta'(k) \geq \frac{\delta^2(k)}{T^2(k)} - k^{-\omega(1)}$, and hence, $\frac{T'(k)}{\delta'(k)} = k^{O(1)} \left(\frac{T(k)}{\delta(k)}\right)^2$. The adversary A':

– calls A_1 and obtains $\mathfrak{R} = \{r_1, \ldots, r_m\}$, $\mathfrak{N} \subset \{0,1\}^*$, and $a \in \{0,1\}^*$;
– generates two independent random strings $X, X' \leftarrow \mathcal{D}$ and calls A_2 twice to obtain $(n, z) \leftarrow \mathsf{A}_2(X, a)$ and $(n', z') \leftarrow \mathsf{A}_2(X', a)$;
– 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 by Lemma 1 above, A' is able to find a collision for h. By Lemma 2 below, the probability that all these conditions hold is at least $\frac{\delta^2(k)}{T^2(k)} - 2^{-\mathsf{H}_2(\mathcal{H}(\mathcal{D}))}$.

It remains to show that $2^{-\mathsf{H}_2(\mathcal{H}(\mathcal{D}))} = \mathsf{P}_{\mathrm{C}}(\mathcal{H}(\mathcal{D})) = k^{-\omega(1)}$. Let C be a collision-finding adversary that on input 1^k generates $X \leftarrow \mathcal{D}$ and $X' \leftarrow \mathcal{D}$ independently at random and outputs (X, X'). Let E_1 denote the event that $X = X'$. Hence, $\Pr[\mathsf{E}_1] = \mathsf{P}_{\mathrm{C}}(\mathcal{D}) = k^{-\omega(1)}$. Let E_2 be the event that $\mathcal{H}(X) = \mathcal{H}(X')$. As $\mathcal{H}$ is collision-resistant, the success of C is $\Pr[\mathsf{E}_2 \backslash \mathsf{E}_1] = k^{-\omega(1)}$ and due to $\mathsf{E}_1 \subseteq \mathsf{E}_2$, we have $\mathsf{P}_{\mathrm{C}}(\mathcal{H}(\mathcal{D})) = \Pr[\mathsf{E}_2] = \Pr[\mathsf{E}_1] + \Pr[\mathsf{E}_2 \backslash \mathsf{E}_1] = k^{-\omega(1)} + k^{-\omega(1)} = k^{-\omega(1)}$. $\square$

Lemma 2. *The success of* A' *is at least* $\frac{\delta^2(k)}{T^2(k)} - 2^{-\mathsf{H}_2(\mathcal{H}(\mathcal{D}))}$. *(See Appendix)*

Remark. Using the improved verification procedure, it is possible to achieve the original security condition of Haber and Stornetta, assuming that the server S is honest and the set $\mathfrak{N}$ is a prefix-free code with a polynomial number of words.

5 Necessity of the Improved Verification

In this section, we prove that the conventional proof techniques used in theoretical cryptography – *black-box reductions* and *semi black-box reductions* – are unable to prove that collision-resistance implies chain-resistance. Hence, in some sense the modifications in time-stamping schemes are necessary for establishing their provable security. For the self-containedness of this paper, we introduce some basic results about *oracle separation*, which have been used to prove several "impossibilities" in theoretical cryptography [6–8, 13].

5.1 Cryptographic Reductions and Oracle Separation

Almost all known constructions of a new primitive $\mathfrak{P}_2$ from another $\mathfrak{P}_1$ belong to one of the following two types:

Definition 3. *A* semi black-box reduction *from* $\mathfrak{P}_1$ *to* $\mathfrak{P}_2$ *is a machine* $P \in \mathsf{FP}^\bullet$*, so that (1)* $P^f \in \mathfrak{P}_2$ *(*$\forall f \in \mathfrak{P}_1$*); and (2) for any* $\mathsf{A}_2 \in \mathsf{FP}^\bullet$ *there exists* $\mathsf{A}_1 \in \mathsf{FP}^\bullet$*, so that* $\mathsf{A}_2^f \underset{\mathfrak{P}_2}{\mathsf{breaks}} P^f$ *implies* $\mathsf{A}_1^f \underset{\mathfrak{P}_1}{\mathsf{breaks}} f$ *(*$\forall f \in \mathfrak{P}_1$*).*

Definition 4. *A* (fully) black-box reduction *from $\mathfrak{P}_1$ to $\mathfrak{P}_2$ is a pair of machines $P, S \in \mathsf{FP}^{\bullet}$, so that (1) $P^f \in \mathfrak{P}_2$ ($\forall f \in \mathfrak{P}_1$); and (2) $\mathsf{A} \underset{\mathfrak{P}_2}{\mathsf{breaks}} P^f$ implies $S^{\mathsf{A},f} \underset{\mathfrak{P}_1}{\mathsf{breaks}} f$ ($\forall f \in \mathfrak{P}_1, \forall \mathsf{A} \in \mathsf{F}^*$).*

Note that the universal quantifiers apply over F^* instead of FP. The reason is that uniform reductions stay valid if the quantifiers' range is extended from FP to F^* and this is exactly what expresses the *black-box nature* of f and A in these reductions. We will use the following folklore lemmas about oracle separation.

Lemma 3. *(A) If there is $f \in \mathfrak{P}_1 \cap \mathsf{FP}^{\mathcal{O}}$ secure relative to $\mathcal{O}$ but no $g \in \mathfrak{P}_2 \cap \mathsf{FP}^{\mathcal{O}}$ is secure relative to $\mathcal{O}$, then there exist no (fully) black-box reductions from $\mathfrak{P}_1$ to $\mathfrak{P}_2$. (B) If in addition, $\mathcal{O} = \pi^f$ (equality of functions) for a $\pi \in \mathsf{FP}^{\bullet}$, then there exist no semi black-box reductions from $\mathfrak{P}_1$ to $\mathfrak{P}_2$.*

Proof. (A) Suppose (S, P) is a black-box reduction from $\mathfrak{P}_1$ to $\mathfrak{P}_2$. According to the assumptions, $g = P^f \in \mathfrak{P}_2 \cap \mathsf{FP}^{\mathcal{O}}$ and g is insecure relative to $\mathcal{O}$. Hence, $\mathsf{A} \underset{\mathfrak{P}_2}{\mathsf{breaks}}\ g = P^f$ for some $\mathsf{A} \in \mathsf{FP}^{\mathcal{O}} \subset \mathsf{F}^*$. It follows that $S^{f,\mathsf{A}} \underset{\mathfrak{P}_1}{\mathsf{breaks}} f$, contradicting $S^{f,\mathsf{A}} \in \mathsf{FP}^{\mathcal{O}}$. (B) Suppose P is a semi black-box reduction from $\mathfrak{P}_1$ to $\mathfrak{P}_2$. Let $f \in \mathfrak{P}_1 \cap \mathsf{FP}^{\mathcal{O}}$ be a secure (relative to $\mathcal{O}$) instance of $\mathfrak{P}_1$. Let $\mathcal{O} = \pi^f$ for some $\pi \in \mathsf{FP}^{\bullet}$. According to the assumptions, $g = P^f \in \mathfrak{P}_2 \cap \mathsf{FP}^{\mathcal{O}}$ and g is insecure relative to $\mathcal{O}$. Hence, $\mathsf{A} \underset{\mathfrak{P}_2}{\mathsf{breaks}}\ g = P^f$ for some $\mathsf{A} \in \mathsf{FP}^{\mathcal{O}}$. Therefore, taking $\mathsf{A}_2 = \mathsf{A}^{\pi} \in \mathsf{FP}^{\bullet}$ we have that $\mathsf{A} = \mathsf{A}^{\mathcal{O}} = \mathsf{A}^{\pi^f} = \mathsf{A}_2^f \underset{\mathfrak{P}_2}{\mathsf{breaks}} P^f$. Hence, there exists $\mathsf{A}_1 \in \mathsf{FP}^{\bullet}$, so that $\mathsf{A}_1^f \underset{\mathfrak{P}_1}{\mathsf{breaks}} f$, which contradicts $\mathsf{A}_1^f \in \mathsf{FP}^{\mathcal{O}}$. □

Definition 5. *A (semi/fully) black-box reduction is said to be a* self reduction *if P is a trivial machine, i.e. $P^f = f$ (for every f).*

Lemma 4. *(A) If relative to $\mathcal{O}$ there is a secure instance of $f \in \mathfrak{P}_1$, which is also an insecure instance of $\mathfrak{P}_2$, then there exist no (fully) black-box self reductions from $\mathfrak{P}_1$ to $\mathfrak{P}_2$. (B) If in addition, $\mathcal{O} = \pi^f$ (equality of functions) for a $\pi \in \mathsf{FP}^{\bullet}$, then there exist no semi black-box self reductions from $\mathfrak{P}_1$ to $\mathfrak{P}_2$.*

The proof of Lemma 4 is completely analogous to the proof of Lemma 3.

5.2 Non-Existence of Fully Black-Box Self Reductions

We define an oracle $\mathcal{O}$, relative to which there exist a collision-resistant hash function $H\colon \{0,1\}^{2k} \to \{0,1\}^k$ (chosen randomly from a set $\mathfrak{F}$ of functions) that is not chain-resistant. The oracle $\mathcal{O}$ responds to the following queries:

- H-queries that given as input $(x_1, x_2) \in \{0,1\}^{2k}$ return $H(x_1, x_2) \in \{0,1\}^k$.
- A_1-queries that given as input 1^k return the root r_k of a Merkle tree [11] M_k, the leaves of which are all k-bit strings in lexicographic order (Fig. 2).
- A_2-queries that given as input a bit string $x \in \{0,1\}^k$ find $z \in (\{0,1\}^k)^k$, based on M_k, so that $F_H(x; x; z) = r_k$ and output a pair (x, z).

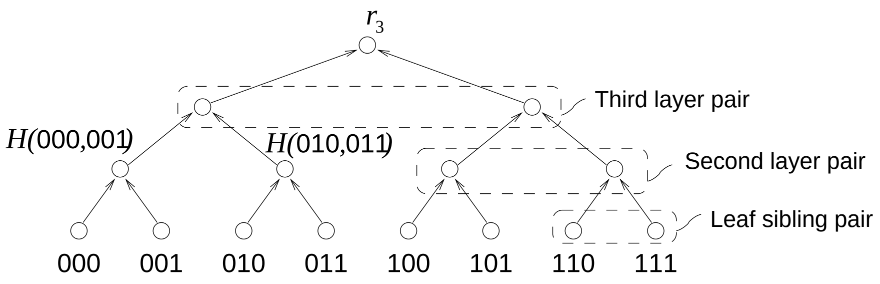

Fig. 2. Computations performed by $A_1(1^k)$ in case $k = 3$.

We assume that $\mathcal{O}$-queries are of unitary cost and hence H is not chain-resistant relative to $\mathcal{O}$. We define $\mathfrak{F}$ so that $\mathcal{O}$ is insufficient for finding collisions for H.

Let $\mathfrak{F}$ be the set of all functions H, such that for all k: (1) all non-leaf vertices in M_k contain different elements of $\{0,1\}^k$ and (2) all sibling-pairs (including the leaves) are different. Hence, the argument-value pairs in M_k do not comprise collisions and A_1- and A_2-queries do not help in finding collisions for H.

Lemma 5. *Every collision finding adversary* $\mathsf{A}^{\mathcal{O}}$ *for* H *that makes* $p(k) = k^{O(1)}$ *oracle calls, has success probability* $k^{-\omega(1)}$.

Proof. Let $S \subseteq \{0,1\}^{2k}$ denote the set of all pairs in the tree M_k. There are exactly $2^k - 1$ of such pairs. Hence, there are $2^{2k} - 2^k + 1$ pairs in the complement $\overline{S} = \{0,1\}^{2k} \backslash S$. The restriction of H to $\overline{S}$ behaves like a uniformly random function while the restriction of H to S is injective. Hence, if $\mathsf{A}^{\mathcal{O}}$ finds a collision (p_1, p_2) for H, then one or both of the pairs p_1, p_2 belong to $\overline{S}$.

Let $K \subset \{0,1\}^{2k}$ be the set of all pairs for which the value of H is released. If $p_1, p_2 \in \overline{S}$, then the probability of finding collisions does not exceed $\frac{|K \cap \overline{S}|^2}{2^{k+1}} \leq \frac{p^2(k)}{2^{k+1}} = k^{-\omega(1)}$, because the values of $H|_S$ can only be obtained via H-queries.

If $p_1 \in S$ and $p_2 \in \overline{S}$, then the probability of finding a collision does not exceed $\frac{|K \cap S| \cdot |K \cap \overline{S}|}{2^k} \leq \frac{mk \cdot (p(k) - m)}{2^k}$, where m is the number of A_2-queries, each of which releases no more than k values of H. The maximum of the last function is achieved if $m \approx \frac{p(k)}{2}$, and hence the success is $\frac{k \cdot p(k)^2}{2^{k+2}} = k^{-\omega(1)}$. □

Corollary 1. *Fully black-box self reductions cannot prove that collision-resistance of* h *implies chain-resistance of* h.

5.3 Non-Existence of Semi Black-Box Self Reductions

The oracle $\mathcal{O}$ defined above does not yet prove the non-existence of semi-black box self reductions because H does not provide full access to $\mathcal{O}$, i.e. $\mathcal{O} \neq \pi^H$. Hence, we have to "embed" $\mathcal{O}$ into H. We define a new hash function (oracle) $\mathcal{O}\colon \{0,1\}^{2n} \to \{0,1\}^n$ recursively for all $n > 0$, assuming that the values of it are already defined for smaller indices.

Let M_n be a complete Merkle tree, the leaves of which are all n-bit strings in the lexicographic order. Each internal vertex v in M_n is computed as a hash $\mathcal{O}_n(v_L, v_R)$ of the child vertices v_L, v_R of v. Note that as we have not yet defined $\mathcal{O}_n\colon \{0,1\}^{2n} \to \{0,1\}^n$, the tree M_n is not yet defined either. We divide the domain $\{0,1\}^{2n} = \{(y_1, y_2)\colon y_1, y_2 \in \{0,1\}^n\})$ into two non-intersecting parts:

- The set S of all sibling pairs in M_n that occur as inputs to $\mathcal{O}$ during the computation of M_n. It contains *leaf-sibling pairs* of the form $(y\mathsf{0}, y\mathsf{1})$, where $y \in \{0,1\}^{n-1}$, *second-layer pairs* of the form $(\mathcal{O}(t\mathsf{00}, t\mathsf{01}), \mathcal{O}(t\mathsf{10}, t\mathsf{11}))$, where $t \in \{0,1\}^{n-2}$ etc. (Fig. 2)
- The set P of all other pairs.

Hence, to define $\mathcal{O}_n$, we have to define two functions: $\mathcal{O}_n^S\colon S \to \{0,1\}^n$ and $\mathcal{O}_n^P\colon P \to \{0,1\}^n$. The function $\mathcal{O}_n^S$ is defined in a deterministic way and is injective (no collisions can be found inside S), while $\mathcal{O}_n^P$ is a random oracle (obviously collision-resistant!). In addition, if $n = 4k$, then we embed a chain-finding adversary for $\mathcal{O}_k$ into $\mathcal{O}_n^S$, which means that $\mathcal{O}$ can find chains for itself and is thereby not chain-resistant.

First of all, we define (for $n = 4k$) an oracle $\mathcal{A}_n\colon \{0,1\}^{2n} \to \{0,1\}^n$ that can be used to find chains for $\mathcal{O}_k$. The oracle $\mathcal{A}_n$ allows input pairs of the form $(\mathsf{0}^{2k}x\mathsf{0}^{k-m}\mathsf{1}^m, \mathsf{0}^k\mathsf{1}^k x\mathsf{0}^{k-m}\mathsf{1}^m)$, where $x \in \{0,1\}^k$ and $m \in \{0,\ldots,k\}$. The set D of all such pairs has exactly $(k+1)2^k$ elements. Let r_k be the root of M_k (which has been already defined). On input of such form, the oracle $\mathcal{A}_n$ finds (based on M_k) $z \in (\{0,1\}^k)^k$, such that $F_{\mathcal{O}}(x;x;z) = r_k$. We define $\mathcal{A}_n$ as follows:

$$\mathcal{A}_n(\mathsf{0}^{2k}x\mathsf{0}^{k-m}\mathsf{1}^m, \mathsf{0}^k\mathsf{1}^k x\mathsf{0}^{k-m}\mathsf{1}^m) = \begin{cases} \mathsf{1}^k x\mathsf{0}^{k-m}\mathsf{1}^m x & \text{if } m = 0 \ , \\ \mathsf{1}^k x\mathsf{0}^{k-m}\mathsf{1}^m z_m & \text{if } m \in \{1,\ldots,k\} \ . \end{cases}$$

Obviously, $\mathcal{A}_n$ is injective and its values never coincide with the allowed inputs.

Now we are ready to define $\mathcal{O}_n^S$. We begin with the case $n \neq 4k$, which is considerably easier, because there is no need to embed $\mathcal{A}_n$ into $\mathcal{O}_n^S$. To define $\mathcal{O}_n^S$ as an injection, it is sufficient to assign different n-bit strings to all $2^n - 1$ internal vertices of M_n. However, care must be taken that no sibling pairs (including the leaf sibling pairs) coincide with other pairs, because otherwise we may have a contradictory definition – different values are assigned to the same input pair. Such contraditions can be easily avoided if, as opposed to the leaf sibling pairs, the elements of internal sibling pairs are in the decreasing order.

If $n = 4k$, then we have to embed $\mathcal{A}_n$ as a function into $\mathcal{O}_n^S$. There are $2^{n-2} = 2^{4k-2}$ second layer pairs in M_n and $(k+1)2^k$ arguments of $\mathcal{A}_n$ (elements of D). As $(k+1)2^k \le 2^{4k-2}$ for any $k > 0$, there is an injection $e\colon D \to \{0,1\}^{n-2}$ and we can embed D into the set of second layer pairs of M_n, so that for each $x \in \{0,1\}^k$ and $m \in \{0,\ldots,k\}$ there is $t = e(x,m) \in \{0,1\}^{n-2}$, such that $\mathcal{O}(t\mathsf{00}, t\mathsf{01}) = \mathsf{0}^{2k}x\mathsf{0}^{k-m}\mathsf{1}^m$ and $\mathcal{O}(t\mathsf{10}, t\mathsf{11}) = \mathsf{0}^k\mathsf{1}^k x\mathsf{0}^{k-m}\mathsf{1}^m$. Now we apply $\mathcal{A}_n$ to the second layer pairs in $e(D)$ and store the values into M_n as third layer vertices. Note that if $k > 1$, then there are still some second layer pairs for which the value of $\mathcal{O}$ has not yet been defined. Note also that all non-leaf vertices defined thus far are different and hence to conclude the definition of $\mathcal{O}_n^S$, we define (in arbitrary way) the values of other vertices (not yet defined) so that all non-leaf vertices are different and hence $\mathcal{O}_n^S$ is injective.

As said above, for every n we choose $\mathcal{O}_n^P$ uniformly at random from the set of all functions $P \to \{0,1\}^n$. Now we can do it because P is fixed after the procedure above. Like in Lemma 5, we can show in a similar fashion that $\mathcal{O}$ is collision resistant but not chain-resistant, because $\mathcal{O}_{4k}$ can be used to find chains for $\mathcal{O}_k$ (for any $k > 0$) and therefore also a time-stamping scheme that uses $\mathcal{O}$ as a hash function (and (1) for verification) is insecure.

Corollary 2. *Semi black-box self reductions cannot prove that collision-resistance of h implies chain-resistance of h.*

6 Discussion and Open Problems

More Efficient Reductions. The reduction established in the proof of Theorem 1 does not give sufficient security guarantees for practical time-stamping schemes. To show this, assume that $k = 160$ (output size of SHA-1) and that there is an adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ with running time $T(k) = 2^{16}$ and with success probability $\delta(k) = 2^{-16} \approx 1/65000$. Hence, the time-success ratio is $T(k)/\delta(k) = 2^{32}$. If the time unit denotes the time elapsed for one hash operation and a computer performs 10,000 hash operations per second, then $T(k)$ is about six seconds. For practical time-stamping schemes, an attack with such ratio is considered very serious. Now let us examine the consequences of Theorem 1. Assume that the collision-finding adversary A' is implemented very efficiently, so that $T'(k) = 2T(k)$. By Lemma 2, the time-success ratio of A' is $\frac{T'(k)}{\delta'(k)} \approx \frac{2\times T(k)}{\frac{\delta^2(k)}{T^2(k)} - 2^{\mathsf{H}_2(\mathcal{D})}} \geq \frac{2T^3(k)}{\delta^2(k)} = 2^{81}$, which is close to the *birthday barrier* and says nothing essential about security – any 160-bit hash function can be broken with that amount (2^{81}) of computational resources. Hence, even the highest security of h does not exclude the attacks with ratio 2^{32}. The reduction gives practical security guarantees only in case $k > 400$, which is much larger than used in the existing schemes. Therefore, it would be very desirable to find more efficient reductions, say the *linear-preserving* ones [10], in which $\frac{T'(k)}{\delta'(k)} = k^{O(1)} \cdot \frac{T(k)}{\delta(k)}$.

Constructions of Chain-Resistant Hash Functions. We leave open the existence of efficient constructions of chain-resistant hash functions, possibly as atomic primitives. While we proved that collision-resistance does not imply chain-resistance, it is still unknown whether there exist more general black-box constructions ($g = P^h$) of chain-resistant hash functions (g) based on a collision-resistant one (h). In case such constructions exist, it would be sufficient to just replace the hash functions in the existing schemes.

Another interesting research topic is attempts at the opposite: to prove that there exist no general black-box constructions of chain-resistant hash-functions based on collision-resistant ones. It would be sufficient to find an oracle $\mathcal{O}$ relative to which there exist collision-resistant hash functions while no function is chain-resistant. Inspired by the work of Simon [13] it may seem tempting to define an oracle $\mathcal{O}$ capable of finding chains to any computable $f\colon \{0,1\}^{2k} \to \{0,1\}^k$, the description of which is given to $\mathcal{O}$ as an argument. However, there seem to be no obvious ways of doing this. For example, if $\mathcal{O}$ is able to compute the root of the complete Merkle tree M_k^f for any (computable) hash function f, then one can show that such $\mathcal{O}$ can also be "abused" to find collisions for any hash function.

At the same time, it seems very likely that the oracle used by Simon [13] (to prove that collision-resistant hash functions are not black-box constructible from one-way functions) is also sufficient for showing that collision-resistant hash-functions cannot be constructed from the chain-resistant ones.

Stronger Security Conditions. The chain-resistance condition is still too simplistic, considering some scenarios that are very likely to happen in practical implementations of time-stamping schemes. Instead of having unconditional uncertainty about x, it is possible that A_1 has some partial knowledge $y = f(x)$ about x (e.g. ciphertexts or signatures). This suggests a stronger condition:

Definition 6. *A function h is* universally chain-resistant *if for any (probabilistic) function f and for any poly-time adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ with success $\delta = \mathsf{Pr}[x \leftarrow \mathcal{D}, (\mathfrak{R}, a) \leftarrow \mathsf{A}_1(f(x)), (n, z) \leftarrow \mathsf{A}_2(x, a)\colon F_h(x; n; z) \in \mathfrak{R}] = k^{-O(1)}$ there is a poly-time A' with success $\mathsf{Pr}[x \leftarrow \mathcal{D}, x' \leftarrow \mathsf{A}'(f(x))\colon x' = x] = k^{-O(1)}$.*

Loosely speaking, if x can be time-stamped based on $y = f(x)$, then x can be efficiently computed based on y, and hence the time stamp is "legitimate". *This condition implies chain-resistance* if we define $f(x) \equiv 1^k$.

Though the universal chain resistance condition seems natural, it is probably not achievable. To see this, assume that h is one-way even if one of the arguments is revealed to the adversary, i.e. every $\mathsf{A}' \in \mathsf{FP}$ has success

$$\delta'(k) = \mathsf{Pr}[(x, z) \leftarrow \mathcal{U}_{2k}, x' \leftarrow \mathsf{A}'(h(x, z), z)\colon x = x'] = k^{-\omega(1)} \ . \qquad (6)$$

This assumption is intuitively assumed to hold in the case of conventional hash functions. Let f be a probabilistic function such that $(h(x, z), z) \leftarrow f(x)$, where $z \leftarrow \mathcal{U}_k$; and let $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ be defined as follows: $(\{y\}, z) \leftarrow \mathsf{A}_1(y, z)$, and $(\mathsf{0}, z) \leftarrow \mathsf{A}_2(x, z)$. Clearly, the success of A (in the sense of universal chain resistance) is $\delta = 1$, while no adversary A' can efficiently invert f. Therefore, no functions that are one-way in the sense of (6) are universally chain resistant, which means that this is a very strong security requirement. Even if h is defined as a *random oracle*, it is still insufficient for the universal chain-resistance. Nothing changes if the set of valid identifiers is polynomially restricted.

Acknowledgements

The authors are grateful to Matthew Franklin, to Peeter Laud, to Berry Schoenmakers, and to anonymous referees for their valuable remarks and suggestions that helped to improve the quality and readability of the paper, as well as to Estonian Science Foundation for supporting the study.

References

1. Dave Bayer, Stuart Haber, and W.-Scott Stornetta. Improving the efficiency and reliability of digital time-stamping. In *Sequences II: Methods in Communication, Security, and Computer Science*, pp.329-334, Springer-Verlag, New York 1993.
2. Josh Benaloh and Michael de Mare. Efficient broadcast time-stamping. Tech. report 1, Clarkson Univ. Dep. of Mathematics and Computer Science, August 1991.
3. Ahto Buldas, Peeter Laud, Helger Lipmaa, and Jan Villemson. Time-Stamping with Binary Linking Schemes. In *Advances in Cryptology – CRYPTO'98*, *LNCS* 1462, pp. 486-501, 1998.
4. Stuart Haber and W.-Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, Vol. 3, No. 2, pp. 99-111 (1991).
5. Stuart Haber and W.-Scott Stornetta. Secure Names for Bit-Strings. In *ACM Conference on Computer and Communications Security*, pp. 28–35, 1997.
6. Yael Gertner, Sampath Kannan, Tal Malkin, Omer Reingold, and Mahesh Viswanathan. The relationship between public key encryption and oblivious transfer. In *FOCS 2000, 41st IEEE Symposium on the Foundations of Computer Science*, pp. 325–335, 2000.
7. Susan Rae Hohenberger. The Cryptographic Impact of Groups with Infeasible Inversion. Master Thesis. Massachusetts Institute of Technology. May 2003.
8. Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. *Proceedings of 21st Annual ACM Symposium on the Theory of Computing*, 1989, pp. 44 – 61.
9. ISO IEC 18014-3,Time-stamping services – Part 3: Mechanisms producing linked tokens.
10. Michael Luby. *Pseudorandomness and cryptographic applications.* Princeton University Press, 1996.
11. Ralph C. Merkle. Protocols for public-key cryptosystems. *Proceedings of the 1980 IEEE Symposium on Security and Privacy*, pp.122-134, 1980.
12. Alexander Russell. Necessary and sufficient conditions for collision-free hashing. *Journal of Cryptology* (1995) 8: 87–99.
13. Daniel Simon. Finding collisions on a one-way street: can secure hash functions be based on general assumptions? In *Advances in Cryptology – EUROCRYPT'98, LNCS 1403*, pp.334–345. Springer-Verlag, 1998.

A Proof of Lemma 2

Let $\mathsf{Pr}[\mathfrak{R},\mathfrak{N},a] = \mathsf{Pr}[(\overline{\mathfrak{R}},\overline{\mathfrak{N}},\overline{a}) \leftarrow \mathsf{A}_1(1^k)\colon \overline{\mathfrak{R}} = \mathfrak{R},\ \overline{\mathfrak{N}} = \mathfrak{N},\ \overline{a} = a]$ and

$$\mathsf{Pr}[\mathsf{Brk} \mid \mathfrak{R},\mathfrak{N},a] = \mathsf{Pr}[X \leftarrow \mathcal{D}, (n,z) \leftarrow \mathsf{A}_2(X,a)\colon F_h(\mathcal{H}(X);n;z) \in \mathfrak{R},\ n \in \mathfrak{N}] \ .$$

By definition, $\mathsf{Pr}[\mathsf{Brk} \mid \mathfrak{R},\mathfrak{N},a]$ is the conditional success of $(\mathsf{A}_1,\mathsf{A}_2)$, assuming that A_1 outputs $(\mathfrak{R},\mathfrak{N},a)$. Thus, $\delta(k) = \sum_{\mathfrak{R},\mathfrak{N},a} \mathsf{Pr}[\mathfrak{R},\mathfrak{N},a] \cdot \mathsf{Pr}[\mathsf{Brk} \mid \mathfrak{R},\mathfrak{N},a]$, where the sum is computed over all possible outputs of $\mathsf{A}_1(1^k)$. Let

$$\mathsf{Pr}[\mathsf{Brk}(\rho,n) \mid \mathfrak{R},\mathfrak{N},a] = \mathsf{Pr}[X \leftarrow \mathcal{D}, (\overline{n},z) \leftarrow \mathsf{A}_2(X,a)\colon F_h(\mathcal{H}(X);\overline{n};z) = \overline{\rho} \in \mathfrak{R}, \overline{n} = n \in \mathfrak{N}]$$

be the conditional probability of success with additional condition that the identifier (output by A_2) is n and the result of the hash chain is $\rho \in \mathfrak{R}$. Now assume that A' has finished and hence the following computations have been performed:

$$(\mathfrak{R}, \mathfrak{N}, a) \leftarrow \mathsf{A}_1(1^k), \begin{array}{l} X \leftarrow \mathcal{D} \\ X' \leftarrow \mathcal{D} \end{array}, \begin{array}{l} (\overline{n}, z) \leftarrow \mathsf{A}_2(X, a) \\ (\overline{n'}, z') \leftarrow \mathsf{A}_2(X', a) \end{array}, \begin{array}{l} \overline{\rho} = F_h(\mathcal{H}(X); \overline{n}; z) \\ \overline{\rho'} = F_h(\mathcal{H}(X'); \overline{n'}; z') \end{array} .$$

Let Coll denote the event that A' finds a collision and let Coll' denote the event that A' finds a collision so that $\overline{\rho} = \overline{\rho'} \in \mathfrak{R}$, $\overline{n} = \overline{n'}$. By Lemma 1, the event Coll is a superset of $\mathsf{Coll}' \cap (\mathcal{H}(X) \neq \mathcal{H}(X'))$. Hence, the success $\delta'(k) = \mathsf{Pr}[\mathsf{Coll}']$ of A' satisfies

$$\begin{aligned} \delta'(k) &\geq \mathsf{Pr}[\mathsf{Coll}' \cap (\mathcal{H}(X) \neq \mathcal{H}(X'))] = 1 - \mathsf{Pr}[(\neg\mathsf{Coll}') \cup (\mathcal{H}(X) = \mathcal{H}(X'))] \\ &\geq 1 - (1 - \mathsf{Pr}[\mathsf{Coll}']) - \mathsf{Pr}[\mathcal{H}(X) = \mathcal{H}(X')] = \mathsf{Pr}[\mathsf{Coll}'] - \mathsf{P}_{\mathrm{C}}(\mathcal{D}) \ . \end{aligned}$$

Therefore, it remains to estimate $\mathsf{Pr}[\mathsf{Coll}']$. Let $\mathsf{Coll}'(n, \rho)$ denote the product event $\mathsf{Coll}' \cap (\overline{n} = \overline{n'} = n) \cap (\overline{\rho} = \overline{\rho'} = \rho)$. From the independence of the two runs of A_2 it follows that $\mathsf{Pr}[\mathsf{Coll}'(n, \rho) \mid \mathfrak{R}, \mathfrak{N}, a] = \mathsf{Pr}^2[\mathsf{Brk}(n, \rho) \mid \mathfrak{R}, \mathfrak{N}, a]$ and hence,

$$\begin{aligned} \mathsf{Pr}[\mathsf{Coll}' \mid \mathfrak{R}, \mathfrak{N}, a] &= \sum_{n,\rho} \mathsf{Pr}[\mathsf{Coll}'(n, \rho) \mid \mathfrak{R}, \mathfrak{N}, a] = \sum_{n,\rho} \mathsf{Pr}^2[\mathsf{Brk}(n, \rho) \mid \mathfrak{R}, \mathfrak{N}, a] \\ &= \sum_{n,\rho} \mathsf{Pr}^2[\mathsf{Brk}|\mathfrak{R},\mathfrak{N},a] \cdot \mathsf{Pr}^2[n,\rho|\mathfrak{R},\mathfrak{N},a,\mathsf{Brk}] = \mathsf{Pr}^2[\mathsf{Brk}|\mathfrak{R},\mathfrak{N},a] \cdot \sum_{n,\rho} \mathsf{Pr}^2[n,\rho|\mathfrak{R},\mathfrak{N},a,\mathsf{Brk}] \\ &\geq \mathsf{Pr}^2[\mathsf{Brk} \mid \mathfrak{R}, \mathfrak{N}, a] \cdot \frac{1}{|\mathfrak{R}| \cdot |\mathfrak{N}|} \geq \mathsf{Pr}^2[\mathsf{Brk} \mid \mathfrak{R}, \mathfrak{N}, a] \cdot \frac{1}{T^2(k)} \ , \end{aligned}$$

where the first inequality holds because $\sum_{n,\rho} \mathsf{Pr}[n, \rho \mid \mathfrak{R}, \mathfrak{N}, \mathsf{Brk}, a] = 1$ and for any probability space $\mathcal{X}$, we have $\sum_{x \in \mathcal{X}} \overset{2}{\mathsf{Pr}}[x] \geq \frac{1}{|\mathcal{X}|}$. The second inequality follows from the observation that $\mathfrak{R}$ and $\mathfrak{N}$ are produced by the adversary and hence their size cannot exceed the running time. Therefore,

$$\mathsf{Pr}[\mathsf{Coll}'] = \frac{\sum_{\mathfrak{R},\mathfrak{N},a} \mathsf{Pr}[\mathfrak{R},\mathfrak{N},a] \cdot \mathsf{Pr}^2[\mathsf{Brk} \mid \mathfrak{R},\mathfrak{N},a]}{T^2(k)} \geq \frac{(\mathsf{Pr}[\mathfrak{R},\mathfrak{N},a] \cdot \mathsf{Pr}[\mathsf{Brk} \mid \mathfrak{R},\mathfrak{N},a])^2}{T^2(k)} = \frac{\delta^2(k)}{T^2(k)},$$

which follows from the Jensen inequality. $\square$