Does Secure Time-Stamping Imply Collision-Free Hash Functions?

Ahto Buldas[1,2,3,⋆] and Aivo Jürgenson[2,4]

[1] Cybernetica, Akadeemia tee 21, 12618 Tallinn, Estonia
[2] Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia
Aivo.Jurgenson@eesti.ee
[3] University of Tartu, Liivi 2, 50409 Tartu, Estonia
Ahto.Buldas@ut.ee
[4] Elion Enterprises Ltd, Endla 16, 15033 Tallinn, Estonia

Abstract. We prove that there are no black-box reductions from Collision-Free Hash Functions to secure time-stamping schemes, which means that in principle secure time-stamping schemes may exist even if there exist no collision-resistant hash functions. We show that there is an oracle relative to which there exist secure time-stamping schemes but no hash function is collision-free. The oracle we use is not new — a similar idea was already used by Simon in 1998 to show that collision-free hash functions cannot be constructed from one-way permutations in a black-box way. Our oracle contains a random hash function family f and a universal collision-finder A. We show that hash-tree time-stamping schemes that use f as a hash function remain secure even in the presence of A. From more practical view, our result is an implicit confirmation that collision-finding attacks against hash functions will tell us quite little about the security of hash-tree time-stamping schemes and that we need more dedicated research about back-dating attacks against practical hash functions.

1 Introduction

Cryptographic hash functions transform a message X of an arbitrary length into a digest $h(X)$ of a fixed length. They have several applications, such as electronic signatures, fast Message Authentication Codes (MACs), secure registries, time-stamping schemes, etc. Though the range of hash function applications is growing rapidly, not much is known either about suitable design criteria or about how to formalize the security conditions for hash functions demanded by applications.

Security proofs of applications often assume the *collision-freedom* of hash functions and this gives an impression as if such a strong security requirement was necessary. Recent success in finding collisions for practical hash functions (MD4,MD5, RIPEMD, SHA-0) by Wang et al. [13,14,16] and later improvements [11,15] forced us to revisit the security proofs in order to clarify for which practical implementations the collisions are a real threat.

This paper focuses on one particular application of cryptographic hash functions – time-stamping. For a long time it was believed that collision-freedom is a necessary

⋆ Partially supported by Estonian SF grant no. 6944, and by EU FP6-15964: "AEOLUS".

W. Susilo, J.K. Liu, and Y. Mu. (Eds.): ProvSec 2007, LNCS 4784, pp. 138–150, 2007.

and sufficient condition for the security of hash-tree based time-stamping schemes. In 2004, it was shown by Buldas and Saarepera [3] that collision-freedom is probably *insufficient* to prove that unbounded hash-tree time-stamping schemes [1,5] (without explicit restrictions to the length of hash-chains) are secure. Buldas and Laur [2] then showed that collision-freeness (and even one-wayness) is also *unnecessary*—once there are hash functions that are secure for time-stamping, then there also exist hash functions which are secure for time-stamping but are not even one-way (and hence, not collision-resistant). This result shows that breaking a particular hash function in terms of collisions (even if they are meaningful textual documents) does not necessarily mean that this particular hash function is insecure for time-stamping.

In this paper we go even further. We show that collision-free hash functions cannot be constructed from secure time-stamping schemes in a black-box way. This means that even if one finds a universal collision-finder that "breaks" all known hash functions, there may still exist hash functions that remain secure for time-stamping. We show that there is an oracle relative to which there exist secure time-stamping schemes but no hash function is collision-free. The oracle we use is not new — a similar construction was already used by Simon [12] to show that collision-free hash functions cannot be constructed from one-way permutations in a black-box way. Our oracle contains a random hash function family $f\colon \{0,1\}^{2k} \to \{0,1\}^k$ and a universal collision-finder A. To prove the main result of this paper we will show that hash-tree time-stamping schemes with the random hash function f remain secure in the presence of the universal collision-finding oracle A.

From the practical point of view, our result is another confirmation that collision-finding attacks against hash functions will tell us quite little about the security of time-stamping schemes that use these "broken" hash functions. Therefore, to study the security of hash-based time-stamping schemes, it is insufficient to study the feasibility of collision-finding attacks. Instead, we need dedicated research on practical back-dating attacks. To our knowledge, there is only one work published on this issue [8].

The paper is organized as follows. Section 2 gives necessary notations and definitions. Section 3 outlines the basics of secure hash-based time-stamping schemes. In Section 4, for the self-consistency of this paper, we provide the reader with basic definitions and results about oracle separation. In Section 5, we define the separating oracles and prove the main result of this work.

2 Preliminaries and Notation

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 $\mathsf{Pr}\left[x_1 \leftarrow \mathcal{D}_1, \ldots, x_m \leftarrow \mathcal{D}_m\colon F(x_1, \ldots, x_m)\right]$ denotes the probability that $F(x_1, \ldots, x_m)$ is true after the ordered assignment of $x_1, \ldots, x_m$. For functions $f, g\colon \mathbb{N} \to \mathbb{R}$, 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 $\lim\limits_{k\to\infty} \frac{g(k)}{f(k)} = 0$. If $f(k) = k^{-\omega(1)}$, then f is *negligible*. A Turing machine M is *polynomial-time* (*poly-time*) if it runs in time