Bounded Pre-Image Awareness and the Security of Hash-Tree Keyless Signatures

Ahto Buldas[1,2], Risto Laanoja[1,2], Peeter Laud[3], and Ahto Truu[1] *

[1] GuardTime AS, Tammsaare tee 60, 11316 Tallinn, Estonia.
[2] Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia.
[3] Cybernetica AS, Mäealuse 2/1, 12618 Tallinn, Estonia.

Abstract. We present a new tighter security proof for unbounded hash tree keyless signature (time-stamping) schemes that use Merkle-Damgård (MD) hash functions with Preimage Aware (PrA) compression functions. It is known that the PrA assumption alone is insufficient for proving the security of unbounded hash tree schemes against back-dating attacks. We show that many known PrA constructions satisfy a stronger *Bounded Pre-Image Awareness (BPrA)* condition that assumes the existence of an extractor $\mathcal{E}$ that is bounded in the sense that for any efficiently computable query string α, the number of outputs y for which $\mathcal{E}(y,\alpha)$ succeeds does not exceed the number of queries in α. We show that blockcipher based MD-hash functions with rate-1 compression functions (such as Davies-Meyer and Miyaguchi-Preneel) of both type I and type II are BPrA. We also show that the compression function of Shrimpton-Stam that uses non-compressing components is BPrA. The security proof for unbounded hash-tree schemes is very tight under the BPrA assumption. In order to have 2^s-security against back-dating, the hash function must have $n = 2s+4$ output bits, assuming that the security of the hash function is close to the birthday barrier, i.e. that there are no structural weaknesses in the hash function itself. Note that the previous proofs that assume PrA gave the estimation $n = 2s+2\log_2 C+2$, where C is the maximum allowed size of the hash tree. For example, if $s = 100$ (2^{100}-security) and $C = 2^{50}$, the previous proofs require $n = 302$ output bits, while the new proof requires $n = 204$ output bits.

1 Introduction

Keyless time-stamping [10] was proposed by Haber *et al* in order to avoid key-based cryptography and trusted third parties so that time stamps become irrefutable proofs of time. A collection of C documents is hashed down to a single digest of few dozen bytes that is then published in widely available media such as newspapers. Merkle hash trees [12] enable to create compact "keyless signatures" of size $O(\log C)$ for each of the C documents. Every such signature consists of all sibling hash values in the path from a document (a leaf of the tree) to the root of the tree. After the root hash value is published, it will be impossible for anyone to *back-date* a new document in terms of creating a hash chain from a new document to the already published hash value. In [1], a global-scale hash-tree scheme was drafted where at every unit of time a large hash tree is created co-operatively by numerous servers all over the globe and the root value is published in newspapers.

The security of hash-tree schemes against back-dating can be reduced to collision-resistance of the hash function. The first correct security proof was published in 2004 [6] but this proof assumes that the size C of the global hash tree (the *capacity* of the scheme) is limited and the number n of the output bits of the hash function needed for 2^s-security was $n = 4s+2\log_2 C+2$, i.e. n depends on C (Tab. 1). This means that if the maximum hash tree size is 2^{60}, then for 2^{100}-security against

* This work has been supported by Estonian Research Council through grant IUT27-1 and by Eestonian Research and Development Foundation (ERDF) through the Centre of Excellence in Computer Science (EXCS).

back-dating one has to use 522-bit hash functions. The practical hash functions in such schemes might be 256-bit and twice larger output size will double the amount of data in the system.

The tightest possible proof of security [5] against back-dating under the collision-resistance assumption requires the output size $n = 3s + \log_2 C + 8$, which in case $C = 2^{60}$ and $s = 100$ gives $n = 368$ (Tab. 1), which is still too large if one desires to use 256-bit hash functions in a global hash tree scheme. The proof in [5] has been shown to be asymptotically optimally tight if the collision-resistance property is used as the security assumption. So, the only way to obtain tighter security proofs is to use stronger (or incomparable) security assumptions for hash functions.

In [4], a tighter security proof was presented that instead of the traditional collision-resistance assumption used a stronger assumption called Pre-Image Awareness (PrA) (first proposed in [9]). The PrA condition makes sense if the hash function uses ideal components (ideal ciphers, random permutations, etc.). The proof under PrA required hash size $n = 2s + 2\log_2 C + 2$. This might be valuable for high security requirements, but for the case of $C = 2^{60}$ and $s = 100$ gives $n = 322$, which is still too large for using 256-bit hash functions, for example.

Therefore, in [4], a new non-standard and seemingly just slightly stronger than PrA security assumption—Strong Pre-Image Awareness (SPrA)—was used to obtain a tighter security proof with required hash size $n = 2s + 2\log_2 \log_2 C + 2$, which in case $C = 2^{60}$ and $s = 100$ gives $n = 214$. However, the SPrA is a new assumption and not sufficiently studied. In contrast to the PrA condition, which is known to hold for many cryptographic constructions of hash functions [9], there are no similar proofs of SPrA. Considering the formal definition of SPrA, such proofs might be hard to construct, mostly because the SPrA condition involves arbitrary "parsing" functions.

In this work, we define another strenghtening of the PrA condition, the so-called *Bounded Pre-Image Awareness (BPrA)* that assumes the existence of an extractor $\mathcal{E}$ that is bounded in the sense that for any efficiently computable query string α, the number of outputs y for which $\mathcal{E}(y, \alpha)$ succeeds does not exceed the number of queries in α. We show that many known PrA constructions actually are BPrA. For example, we show that blockcipher based MD-hash functions with rate-1 compression functions (such as Davies-Meyer and Miyaguchi-Preneel) of both type I and type II are BPrA. We also show that some compression functions with uncompressing components (such as Shrimpton-Stam) are BPrA. Therefore, the BPrA assumption is (at least for now) more justified in practice than the SPrA assumption.

The security proof for unbounded hash-tree schemes is very tight under the BPrA assumption, even tighter than under the SPrA assumption. In order to have 2^s-security against back-dating, the hash function must have $n = 2s + 4$ output bits (Tab. 1), assuming that the security of the hash function is close to the birthday barrier, i.e. that there are no structural weaknesses in the hash function itself. In the case of $s = 100$ this gives $n = 204$.

Tab. 1 summarizes the efficiency of the existing security reductions, in which a t-time backdating adversary with success probability δ is converted to a t'-time collision-finding adversary with success probability δ'. An n-bit hash function is assumed to be $2^{n/2}$-secure, i.e. near to the birthday barrier. The third column of Tab. 1 presents a formula for the required output size n of the hash function for the time-stamping scheme to be 2^s-secure. The last column presents the output size in a particular case, where $s = 100$, $C = 2^{60}$ and $T = 2^{32}$. In addition to the new security proof under the BPrA assumption, we also show in this work that in the RO model bounded schemes are secure beyond the birthday barrier.

The paper is organized as follows. In Sec. 2, we provide readers with necessary preliminary concepts and the state of the art in the security proofs of hash tree schemes. In Sec. 3, we study the

Table 1. Efficiency of security proofs, where n is the required output size of the hash function, assuming that the scheme is 2^s-secure, uses hash trees of size C and is intended for a time period of T units. The results of this work are presented in bold.

Assumption	Formula	Required Output Size n	$n(2^{60}, 2^{32}, 100)$
CR [6]	$\frac{t'}{\delta'} \approx 2C\left(\frac{t}{\delta}\right)^2$	$n = 2\log_2 C + 4s + 2$	522
CR [5]	$\frac{t'}{\delta'} \approx 14\sqrt{C}\left(\frac{t}{\delta}\right)^{1.5}$	$n = \log_2 C + 3s + 8$	368
PrA [4]	$\frac{t'}{\delta'} \approx 2C\frac{t}{\delta}$	$n = 2(\log_2 C + s + 1)$	322
SPrA [4]	$\frac{t'}{\delta'} \approx 4\log_2 C\frac{t}{\delta}$	$n = 2(\log_2\log_2 C + s + 2)$	216
BPrA	$\frac{t'}{\delta'} \approx 4\frac{t}{\delta}$	$n = 2s + 4$	**204**
RO (bounded)	$\frac{t}{\delta} \geq \frac{2^{n-1}}{CT}$	$n = s + \log_2 C + \log_2 T + 1$	**193**
RO (unbounded) [4]	$\frac{t}{\delta} \geq 2^{\frac{n-1}{2}}$	$n = 2s + 1$	201

security proofs in the random oracle model and present the motivation behind the new BPrA security condition. In Sec. 4, we show that many of the block cipher based hash function constructions (and also some constructions with non-compressing ideal components, e.g. Shrimpton-Stam [14]) that have been proved to be PrA are actually BPrA and hence these hash functions are probably much more secure for hash-tree time-stamping than the previously known security proofs might suggest.

2 Preliminaries

2.1 Tightness of Security Proofs

The security of cryptographic schemes is measured by the amount of resources needed for an adversary to break the primitive. A scheme is said to be S-secure, if it can be broken by no adversaries with less than S units of resources available. Considering that the running time t and the success probability δ of the known practical attacks against the scheme may vary, Luby [11] proposed the *time-success ratio* $\frac{t}{\delta}$ as a measure for attacking resources. A scheme is said to be S-secure, if the success probability of any t-time adversary does not exceed $\frac{t}{S}$.

In a typical security proof for a scheme $\mathcal{P}$ built from a primitive $\mathcal{Q}$, it is shown that if $\mathcal{Q}$ is S_q-secure, then $\mathcal{P}$ is S_p-secure. Bellare and Rogaway [2, 3] first emphasized the importance of the *tightness* S_p/S_q of security proofs in practical applications. Informally, tightness shows how much security of the primitive is retained by the scheme. Security proofs are mostly *reductions*: an adversary for $\mathcal{P}$ with running time t and success probability δ is transformed to an adversary for $\mathcal{Q}$ with running time t' and success probability δ'. This means that for having $\frac{t}{\delta}$-secure $\mathcal{P}$, we have to use a $\frac{t'}{\delta'}$-secure $\mathcal{Q}$.

2.2 Security Properties of Hash Functions

In this paper, we study the security properties of hash functions H^P that use some kind of ideal functionality P (random permutations, random functions, ideal ciphers, etc.) as an oracle. For example, in case of the Merkle-Damgård hash functions, the compression function and the output transform are often assumed to be ideal objects. In this section, we describe some of the properties of hash functions, starting from the strongest ones.

Random Oracles. By a *random oracle* $\mathcal{R}$, we mean a function that is chosen randomly from the set of all functions of type $\{0,1\}^m \to \{0,1\}^n$. By the *random oracle heuristic* we mean a security argument when an application of a hash function (e.g. a time-stamping scheme, a signature scheme) is proved to be secure in the so-called *random oracle model*, where the hash function is replaced with a random oracle. The random oracle heuristic was first introduced by Bellare and Rogaway [2]. Although it was proved later by Canetti et al [7] that the random oracle heuristic fails in certain theoretical cases, proofs in the random oracle model are still considered valuable security arguments, especially if no better security proofs are known.

Pre-Image Awareness. Pre-Image Awareness (PrA) of a (hash) function H means, that if we first commit an output y and later come up with an input x, such that $y = H(x)$, then it is safe to conclude that we knew x before committing y. This notion was first formalized by Dodis et al. [9] for hash functions H^P that are built using an ideal primitive P as a black box. For H^P being PrA, there has to be an efficient deterministic extractor $\mathcal{E}$ which when given y and the list α of all previously made P-calls, outputs an input x, such that $H^P(x) = y$, or $\perp$ if $\mathcal{E}$ was unable to find such an x. The adversary tries to find x and y so that $x \neq \mathcal{E}(\alpha, y)$ and $y = H^P(x)$. A weaker form of PrA (so-called WPrA) allows $\mathcal{E}$ output a set $\mathcal{L}$ of inputs x, and the adversary tries to find x, such that the query $\mathcal{L} \leftarrow \mathcal{E}(\alpha, y)$ was made, $y = H^P(x)$, but $x \notin \mathcal{L}$. Obviously, WPrA becomes PrA if the number of elements in $\mathcal{L}$ is limited to one, i.e. $|\mathcal{L}| \leq 1$. To define pre-image awareness of H^P

$\mathbf{Exp}^{\mathrm{pra}}_{H,P,\mathcal{E},B}$:
$x \leftarrow B^{\mathsf{P},\mathsf{Ex}}$
$y \leftarrow H^P(x)$
If $\mathtt{Q}[y] = 1$ and $\mathtt{V}[y] \neq x$ **return** 1,
else return 0

oracle $\mathsf{P}(m)$:
$c \leftarrow P(m)$
$\alpha \leftarrow \alpha || (m, c)$
return c

oracle $\mathsf{Ex}(y)$:
$\mathtt{Q}[y] \leftarrow 1$
$\mathtt{V}[y] \leftarrow \mathcal{E}(y, \alpha)$
return $\mathtt{V}[y]$

Fig. 1. Preimage awareness experiment with the oracles P and Ex.

in a precise way, we set up an experiment **Exp** (see Fig. 1), specified as a game which an attacker B is trying to win. B is constrained to oracle access to P, via a wrapper oracle P, which records all P-calls made by B as an advise string α. Likely, the extractor $\mathcal{E}$ is also accessible through another wrapper oracle Ex, which uses global arrays $\mathtt{Q}$ (initially $\perp$ everywhere) and $\mathtt{V}$ (initially blank). $\mathtt{Q}$ is used to record all input parameters to $\mathcal{E}$; $\mathtt{V}$ is used to store all successfully extracted values corresponding to $\mathcal{E}$'s inputs. The adversary B tries to output a value x such that $H^P(x) = y$, $\mathtt{Q}[y] = 1$ and $\mathtt{V}[y] \neq x$, i.e. $\mathcal{E}$ tried to invert y, but was unsuccessful. As P- and Ex-calls are unit cost, the running time of B does not depend on the running time of $\mathcal{E}$. Note that PrA implies collision-resistance [9], but WPrA does not.

Definition 1 (Pre-Image Awareness). *A function H^P is S-secure* pre-image aware *(PrA) if there is an efficient extractor $\mathcal{E}$, so that for every t-time B:*

$$\mathbf{Adv}^{\mathrm{pra}}_{H,P,\mathcal{E}}(B) = \mathsf{Pr}\left[1 \leftarrow \mathbf{Exp}^{\mathrm{pra}}_{H,P,\mathcal{E},B}\right] \leq \frac{t}{S} \ . \tag{1}$$

In [4], a stronger notion of *Strong Pre-Image Awareness* (SPrA) was presented in which the Ex-oracle is allowed to use the "oldest" possible α. For example, if we obtain $x \leftarrow \mathsf{Ext}(y)$ (where $x = x_1 x_2$ and $x_1, x_2 \in \{0,1\}^n$) for which the oracle uses α, and later we call $\mathsf{Ext}(x_1)$, the same α

is used for extraction, because the oracle remembers that x_1 was created by just "parsing" x and it is thereby as old as x and the use of α is justified. This new notion allows one to establish more tight security proofs for hash-tree time-stamping than the PrA would allow.

Collision Resistance. Informally, the collision resistance of a hash function H^P means that it is infeasible for adversaries to find two different inputs x and x' that have the same hash value, i.e. $H^P(x) = H^P(x')$. This definition makes sense only if the ideal primitive P contains some randomness, as the collisions of fixed functions can always be "wired" into the adversary.

Definition 2 (Collision Resistance). *A function H^P is S-secure* collision resistant *(CR) if for every adversary B with running time t:*

$$\mathbf{Adv}^{\mathrm{cr}}_{H,P}(B) = \mathsf{Pr}\left[x, x' \leftarrow B^P : x \neq x',\ H^P(x) = H^P(x')\right] \leq \frac{t}{S}\ . \tag{2}$$

Due to the so-called Birthday bound, functions with n-bit output can only be up to $2^{\frac{n}{2}}$-secure collision resistant.

2.3 Merkle-Damgård Hash Functions

Merkle-Damgård (or iterated) hash functions use a compression function $F(m, v)$ to iteratively compute a hash of an arbitrary size message m divided into equal blocks $m_1, \ldots, m_\ell$ of suitable size. The hash $h = H(m)$ is computed as follows: (1) $h \leftarrow IV$; (2) for $i \in \{1, \ldots, \ell\}$ do: $h \leftarrow F(m_i, h)$; (3) and output $H(m) = h$. Here, IV is a public and standard initial value. It has been proved [9] that if F is PrA, then so is H.

2.4 Blockcipher-Based Hash Functions

Many hash functions are constructed from secure blockciphers. The most common approach for creating a $2n \to n$ hash function is to use a blockcipher with n-bit block and n-bit key and make a compression function that makes only a single call to the blockcipher. Such constructions were first analyzed by Preneel et al. [13] and are called *rate-1 schemes*. The most general approach is that of Stam [15], where the compression function is defined by the following three steps:

1. Prepare key and plaintext: $(k, x) \leftarrow C_{\mathrm{pre}}(m, v)$;
2. Use the blockcipher: $y \leftarrow E_k(x)$;
3. Output the digest: $w \leftarrow C_{\mathrm{post}}(m, v, y)$.

There are two types of rate-1 compression functions.

Definition 3. *A blockcipher-based rate-1 compression function F^E is called Type-I iff: (1) C_{pre} is bijective; (2) $C_{\mathrm{post}}(m, v, \cdot)$ is bijective for all m, v; and (3) $C_{\mathrm{aux}}(\cdot) = C_{\mathrm{post}}(C^{-1}_{\mathrm{pre}}(k, \cdot), y)$ is bijective for all k, y.*

Definition 4. *A blockcipher-based rate-1 compression function F^E is called Type-II iff: (1) C_{pre} is bijective; (2) $C_{\mathrm{post}}(m, v, \cdot)$ is bijective for all m, v; and (3) $C^{-1}_{\mathrm{pre}}(k, \cdot)$ (restricted to its second output v) is bijective for all k.*

Type-I functions are preimage aware [9] and thus also collision-resistant. Type-II functions become preimage-aware (and collision-resistant) when iterated as Merkle-Damgård hash functions [9].

2.5 Hash-Tree Schemes and their Security Against Back-Dating

Hash trees were introduced by Merkle [12]. Let $h\colon\{0,1\}^{2n} \to \{0,1\}^n$ be a hash function. By a *hash-tree* we mean a tree-shaped data structure that consists of nodes labeled with n-bit hash values. Each node is either a *leaf* which means it has no children, or an *internal node* with two child nodes (the left and the right child). The hash value y of an internal node is computed as a hash $y = h(y_0, y_1)$, where y_0 and y_1 are the hash values of children. There is one *root node* that is not a child of any node. By $r = \mathcal{T}(x_1, \ldots, x_m)$ we mean that r is the root label of a hash tree $\mathcal{T}$ with leaves labeled with hash values $x_1, \ldots, x_m$.

Encoding the Leaves of a Hash Tree. Nodes of a hash tree can be named in a natural way with finite bit-strings. The root node is named by the empty string $\lfloor\rfloor$. If a node is named by ℓ, then its left and right child nodes are named by $\ell 0$ and $\ell 1$, respectively. The name ℓ of a node is also an "address" of the node, considering that one starts searching from the root node, and then step by step, chooses one of the children depending on the corresponding bit in ℓ.

Shape of a Hash Tree. Hash tree has a particular *shape* by which we mean the set of all names of the leaf-nodes. For example, a balanced complete tree with four nodes (Fig. 2, left) has the shape $\{00, 01, 10, 11\}$. If the root hash value is denoted by r (instead of $r_{\lfloor\rfloor}$) and r_ℓ denotes the hash value of a node with name ℓ, then in this example, the relations between the nodes are the following: $r = h(r_0, r_1)$, $r_0 = h(r_{00}, r_{01})$, and $r_1 = h(r_{10}, r_{11})$. The shape $\{000, 001, 01, 1\}$ represents a tree with four leaves (Fig. 2, right) with the hash values being in the following relations: $r = h(r_0, r_1)$, $r_0 = h(r_{00}, r_{01})$, and $r_{00} = h(r_{000}, r_{001})$. Note also that the shape is a *prefix-free code*.

Fig. 2. Trees with shape $\{00, 01, 10, 11\}$ (left) and $\{000, 001, 01, 1\}$ (right).

Hash Chains. In order to prove that a hash value r_ℓ (where $\ell_1\ell_2\ldots\ell_m$ is the binary code of ℓ) participated in the computation of the root hash r, it is sufficient to present all the sibling hashes of the nodes on the unique path from r_ℓ to the root r. For example, in the left tree of Fig. 2, to prove that r_{01} belongs to the tree, one has to present the hashes r_{00} and r_1 that enable us to compute $r_0 = h(\underline{r_{00}}, r_{01})$ and $r = h(r_0, \underline{r_1})$. Hash chains are defined as follows [4]:

Definition 5 (Hash-Chain). *A* hash-link *from x to r (where $x, r \in \{0,1\}^n$) is a pair (s, b), where $s \in \{0,1\}^n$ and $b \in \{0,1\}$, such that either $b = 0$ and $r = h(x\|s)$, or $b = 1$ and $r = h(s\|x)$. A* hash-chain *from x to r is a (possibly empty) list $c = ((s_1, b_1), \ldots, (s_m, b_m))$, such that either $c = ()$ and $x = r$; or there is a sequence $x_0, x_1, \ldots, x_m$ of hash values, such that $x = x_0$, $r = x_m$, and (s_i, b_i) is a hash-link from x_{i-1} to x_i for every $i \in \{1, \ldots, m\}$. We denote by $x \overset{c}{\rightsquigarrow} r$ the proposition that c is a hash chain from x to r. Note that $x \overset{()}{\rightsquigarrow} x$ for every $x \in \{0,1\}^n$. By the* shape $\ell(c)$ *of c we mean the m-bit string $b_1 b_2 \ldots b_m$.*

Hash-Tree Keyless Signature Schemes. The signing (time-stamping) procedure runs as follows. During every time unit t (e.g. one second) the server receives a list $\mathcal{X}_t = (x_1, \ldots, x_m)$ of requests (n-bit hash values) from clients, computes the root hash value $r_{(t)} = \mathcal{T}(x_1, \ldots, x_m)$ of a hash tree $\mathcal{T}$ and publishes $r_{(t)}$ in a public repository $\mathcal{R} = (r_{(1)}, r_{(2)}, \ldots, r_{(t)})$ organized as an append-only list. Each request x_i is then provided with a hash chain c_i (the *signature* for x_i) that proves the participation of x_i in the computation of the root hash value $r_{(t)}$. A request $x \in \mathcal{X}_t$ is said to precede another request $x' \in \mathcal{X}_{t'}$ if $t < t'$. The requests of the same batch are considered simultaneous. In order to verify the hash chain c_i (the signature) of a request x_i, one computes the output hash value of c_i (the last hash value x_m in the sequence) and checks whether $x_m = r$.

Bounded and Unbounded Schemes. A hash-tree keyless signature (time-stamping) scheme is said to be C-bounded, if the shape $\mathcal{S}$ of the hash tree is assumed to be upper-bounded: $|\mathcal{S}| \leq C$ and while verifying a hash chain c it is checked if $\ell(c) \in \mathcal{S}$. A hash-tree keyless signature scheme is (C, T)- strongly bounded if it is C-bounded and also $|\mathcal{R}| \leq T$.

Security Against Back-Dating. Informally, we want that no efficient adversary can *back-date* any request x, i.e. first publishing a hash value r, and only after that generating a new "fresh" x (not pre-computed by the adversary), and a hash chain c, so that $x \overset{c}{\rightsquigarrow} r$. We use the formal security condition from [4] that involves a two-stage back-dating adversary $A = (A_1, A_2)$. The first stage A_1 creates a public repository $\mathcal{R}$ of hash values that may be created in an arbitrary way, not necessary by using hash trees. The second stage A_2 of A presents a high-entropy x and a hash chain $x \overset{c}{\rightsquigarrow} r$ with $r \in \mathcal{R}$. The high entropy of x is crucial because otherwise x could have been pre-computed or guessed by A_1 before r is published and hence x could be in fact older than r and thereby not really back-dated by A_2. Therefore, only *unpredictable* adversaries (that produce high-entropy x) are considered, i.e. x must be hard to guess for A_2 even if the contents of $\mathcal{R}$ and all the internal computations of A_1 are known. There are many ways to define unpredictability. We use the so-called *strong unpredictability* [4]:

Definition 6 (k-Strong Unpredictability). *A back-dating adversary* (A_1,A_2) *is* strongly unpredictable *if the conditional min-entropy* $H_\infty[x \mid \mathcal{R}, a]$ *of* x *(back-dated by* A_2*) is at least* k *bits, i.e. for every input of* A_2 *and for any possible value* x_0 *of* x*, the probability of* $x = x_0$ *is upper bounded by* $\frac{1}{2^k}$.

Definition 7 (Security against Back-Dating). *A hash-tree scheme is* S*-secure against* k*-strongly unpredictable back-dating adversaries* (A_1, A_2) *if for every* t*-time* k*-strongly unpredictable adversary* :

$$\delta = \mathsf{Pr}\left[(\mathcal{R}, a) \leftarrow A_1, (x, c) \leftarrow A_2(\mathcal{R}, a) \colon x \overset{c}{\rightsquigarrow} \mathcal{R},\ \ell(c) \in \mathcal{S}\right] \leq \frac{t}{S}\ , \tag{3}$$

where by $x \overset{c}{\rightsquigarrow} \mathcal{R}$ *we mean that* $x \overset{c}{\rightsquigarrow} r$ *for some* $r \in \mathcal{R}$*, and* a *is an advice string that contains possibly useful information that* A_1 *stores for* A_2.

In the rest of this paper, we will restrict our back-dating adversaries to be $(n-1)$-strongly unpredictable. This restriction is in practice justified by (i) the time-stamped values x being hashes of much longer documents, containing significant amounts of new information, and (ii) the cryptographic hash functions supposedly being good entropy extractors [4].

Existing Security Proofs and their Tightness. The tightness of the existing security proofs is summarized in Tab. 1. The proofs of [6, 5] use the collision-resistance assumption and apply only

to bounded time-stamping schemes. Their tightness depends on the capacity C. Both proofs are in the form of a reduction: a t-time backdating adversary with success probability δ is converted to a t'-time collision-finding adversary with success probability δ'. An n-bit hash function is assumed to be $2^{n/2}$-secure, i.e. near to the birthday barrier. The third column of Tab. 1 presents a formula for the required output size n of the hash function for the time-stamping scheme to be 2^s-secure. The last column presents the output size in a particular case, where $s = 100$, $C = 2^{60}$ and $T = 2^{32}$. Note that 2^{32} seconds is about one hundred years. The proof under PrA assumption is from [4]. We see that even though PrA seems to be much stronger than CR, the required output length is not much smaller. This is because the security loss is linear in C and not in $\sqrt{C}$ as in the case of the CR assumption. SPrA [4] allows much more tight security reductions but has not been sufficiently studied yet. We also see that the random oracle (RO) assumption makes proofs very tight and also to hold for unbounded schemes. The RO proof for unbounded schemes is from [4]. The bounded version is proved in this work.

Security Proofs for Unbounded Schemes. It is known that neither collision-resistance [6] nor PrA [4] is insufficient for proving the security of *unbounded* time-stamping schemes. The only known proof for unbounded schemes [4] uses the random oracle assumption. In order to move forward in this direction, we first examine the main ideas of the proofs in the random oracle model and see how to generalize them for the assumptions weaker than RO.

3 Security in the Random Oracle Model

We first show that for bounded schemes the random oracle model enables security beyond the birthday barrier, i.e. even when using a hash function with n output bits, the security (against back-dating) we achieve is far beyond $2^{n/2}$.

Theorem 1. *If $h : \{0,1\}^{2n} \to \{0,1\}^n$ is a random oracle, then the corresponding (C,T)-strongly bounded hash-tree time-stamping schemes are $\frac{2^{n-1}}{CT}$-secure against $(n-1)$-strongly unpredictable back-dating adversaries.*

Proof. Let $A = (A_1, A_2)$ be a t-time strongly unpredictable adversary (Def. 6) and with success δ as defined in (3). Let t_1, t_2 be the running times of A_1 and A_2, respectively. Considering that $(\mathcal{R}, a) \leftarrow A_1$, and $r \in \mathcal{R}$ is an arbitrary element of $\mathcal{R}$, let $R_1^r \subseteq \{0,1\}^n$ be the set of all x-s so that the h-calls performed by A_1 induce a proper shape hash-chain from x to an r. Let $R_1 = \cup_{r\in\mathcal{R}} R_1^r$. Note that $\mathcal{R} \subseteq R_1$, as an empty hash-chain is always induced by any set of h-calls. Note also that $|\,R_1\,| \le CT$, because $|\,\mathcal{R}\,| \le T$ and $|\,R_1^r\,| \le C$ for any $r \in \mathcal{R}$.

Now let x denote the hash-value back-dated by A_2. The probability that $x \in R_1$ is upper-bounded by $\frac{|R_1|}{2^{n-1}}$ because A is $(n-1)$-strongly unpredictable. In case of $x \notin R_1$, in order to be successful, A_2 has to make additional h-calls so that a chain from x to $r \in \mathcal{R}$ is induced. A necessary condition that A_2 has to satisfy is that it has to find $x' = x'_1 \| x'_2$ so that $x'_1 \notin R_1$ or $x'_2 \notin R_1$ (this means that A_1 did not make h-calls with input x'), but $h(x') \in R_1$. The probability of this condition does not exceed $t_2 \frac{|R_1|}{2^n} \le t_2 \frac{CT}{2^n}$, hence, considering that $|\,R_1\,| \le CT$, and $t_1 \ge 1$, the overall success probability of A is:

$$\delta \le \frac{|\,R_1\,|}{2^{n-1}} + \left(1 - \frac{|\,R_1\,|}{2^{n-1}}\right) t_2 \frac{|\,R_1\,|}{2^n} \le \frac{CT}{2^{n-1}} + t_2 \frac{CT}{2^{n-1}} \le \frac{CT}{2^{n-1}} \cdot (1 + t_2) \le t \frac{CT}{2^{n-1}} \ .$$

Hence, $\frac{t}{\delta} \ge \frac{2^{n-1}}{CT}$. □

The next theorem is from [4]. We repeat their proof in order to draw conclusions about why it holds in the RO model but does not in the PrA-environment.

Theorem 2. *If $h : \{0,1\}^{2n} \to \{0,1\}^n$ is a random oracle, then the corresponding unbounded hash-tree schemes are $2^{\frac{n-1}{2}}$-secure against $(n-1)$-strongly unpredictable back-dating adversaries.*

Proof. Let $A = (A_1, A_2)$ be a t-time strongly unpredictable adversary (Def. 6). Let t_1, t_2 denote the running times of A_1 and A_2, respectively. Assuming that $(\mathcal{R}, a) \leftarrow A_1$, let $R_1 \subseteq \{0,1\}^n$ be the set of all values of x such that the h-calls performed by A_1 induce a hash-chain $x \overset{c}{\rightsquigarrow} r$ with $r \in \mathcal{R}$. Note that $\mathcal{R} \subseteq R_1$ and we assume without loss of generality that the advice a contains R_1.

Let x denote the (back-dated) hash value produced by A_2. Due to the strong unpredictability of A, we have $\mathsf{Pr}[x \in R_1] \leq \frac{|R_1|}{2^{n-1}}$. If $x \notin R_1$ then A_2 has to make h-calls that induce a chain $x \rightsquigarrow r \in \mathcal{R}$. For that, A_2 has to find $x' = x'_1 \| x'_2$ such that $x'_1 \notin R_1$ or $x'_2 \notin R_1$ (i.e. A_1 did not make h-calls with x'), but $h(x') \in R_1$. This happens with probability $\leq t_2 \frac{|R_1|}{2^n}$. Hence, as $|R_1| \leq 2t_1$ and $t_1, t_2 \geq 1$, the success probability of A is $\delta \leq \frac{|R_1|}{2^{n-1}} + \left(1 - \frac{|R_1|}{2^{n-1}}\right) t_2 \frac{|R_1|}{2^n} \leq \frac{2t_1}{2^{n-1}} + \frac{t_1 t_2}{2^{n-1}} \leq \frac{(t_1+t_2)^2}{2^{n-1}} = \frac{t^2}{2^{n-1}}$, and as $\delta^2 \leq \delta \leq \frac{t^2}{2^{n-1}}$, we have $\frac{t}{\delta} \geq 2^{\frac{n-1}{2}}$. □

The key factor of success of this proof is the ability to define the set R_1 and to estimate its size by $|R_1| \leq 2t_1$. In the PrA-type environment where the hash function H^P is not a random oracle, the computable (given the query-sequence α) hash chains that lead to the hash values in $\mathcal{R}$ can be constructed via the extractor $\mathcal{E}$. We start applying $\mathcal{E}$ to the elements of $\mathcal{R}$ and after each try $x \leftarrow \mathcal{E}(\alpha, y)$ apply $\mathcal{E}$ also to the right and the left halves of x, until we reach $\perp$ in every branch. The problem is that the standard PrA assumption does not guarantee that this iterative procedure will end. The new security condition presented in the next section is motivated by the need to make this iterative extraction-tree generation procedure to end eventually. This means that $\mathcal{E}(\alpha, y) \neq \perp$ is allowed to hold only for a limited number of outputs y. This leads to the following new variation of the Pre-Image Awareness security condition.

4 Bounded Pre-Image Awareness

We show that many known PrA constructions actually satisfy a new stronger security condition called *Bounded Pre-Image Awareness* (BPrA) that assumes the existence of a PrA-extractor $\mathcal{E}$ that is bounded in the sense that for efficiently computable query strings α, the number of outputs y for which $\mathcal{E}(y, \alpha) \neq \perp$ does not exceed the number of queries in α.

The security proof for unbounded hash-tree schemes turns out to be very tight under BPrA. In order to have 2^s-security against back-dating, the hash function must have $k = 2s + 4$ output bits, assuming that the security of the hash function is close to the birthday barrier, i.e. that there are no structural weaknesses in the hash function itself. In the case of $s = 100$, this gives $k = 204$.

4.1 Formal Security Condition

The BPrA security condition can be formalized as follows. We have to consider the case where the output size of H^P is larger than the input size of P. Thus, in the extreme case where α contains all possible P-queries, it might be the case that $\mathcal{E}$ is able to determine the inputs of more than $|\alpha|$ of outputs. Hence, instead of requiring the condition $|\{y : \mathcal{E}(y, \alpha) \neq \perp\}| \leq |\alpha|$ unconditionally, we require this condition to hold for efficiently computable query-strings α.

Definition 8. *A function $H^P\colon\{0,1\}^{2n}\to\{0,1\}^n$ is S-secure* Bounded Pre-Image Aware *(BPrA) if it is S-secure PrA, and for any t-time adversary $\alpha \leftarrow A^P$ that produces a P-query list α the probability that $|\{y\colon \mathcal{E}(y,\alpha)\neq\perp\}| > |\alpha|$ does not exceed $\frac{t^2}{2^n}$, where $\mathcal{E}$ is the extractor from the PrA condition.*

This means that efficient adversaries with oracle access to P can only produce query strings α such that the number of outputs y for which $\mathcal{E}(y,\alpha)\neq\perp$ is bounded by the number of P-queries in α.

The bound $\frac{t^2}{2^n}$ may seem ad hoc, but this is actually the natural birthday bound, because in case of output collisions that may occur with probability $\frac{t^2}{2^n}$, a single P-query in α may contribute to computing several different output values.

4.2 Security Proof under BPrA

In order to establish a security proof with measurable tightness, we have to assume a concrete BPrA-security of H^P. As BPrA implies PrA and PrA implies Collision Resistance, by using the birthday bound, no hash function with n-bit output can be more than $2^{n/2}$-secure BPrA. Therefore, in the next proof, we assume that the security of H^P lies between $2^{n/3}$ and $2^{n/2}$.

```
Procedure ExTree^Ex(y):
    If y ∉ 𝒯 and N > 0, then
        𝒯 := 𝒯 ∪ {y}
        N := N − 1
        If ⊥ ≠ Ex(y) = (y0, y1) then
            Define y0, y1 as children of y
            ExTree^Ex(y0)
            ExTree^Ex(y1) .
        endif
    endif
```

```
Procedure ExForest^Ex():
    𝒯 := ∅
    N := 2 × "a time bound for A1"
    For all r ∈ ℛ do
        ExTree^Ex(r) .
```

Fig. 3. Procedures for extracting the set $\mathcal{T}$ from the published hash database $\mathcal{R}$.

Theorem 3. *For unbounded time-stamping schemes with $H^P\colon\{0,1\}^{2n}\to\{0,1\}^n$ to be S-secure ($S \le 2^{\frac{n-1}{2}} - 2$) against $(n-1)$-strongly unpredictable back-dating adversaries, it is sufficient that the hash function H^P is $4S$-secure BPrA.*

Proof. Due to the BPrA assumption there exists an efficient bounded extractor $\mathcal{E}$. Let $A^P = (A_1^P, A_2^P)$ be a strongly unpredictable back-dating adversary with running time t and success probability δ, such that $\frac{t}{\delta} \le 2^{\frac{n-1}{2}} - 2$. We construct a PrA-adversary $B^{\mathsf{P},\mathsf{Ex}}$ that first simulates $(\mathcal{R}, a) \leftarrow A_1^P$ so that all P-calls of are executed through the P-oracle. Let α be the query string after such simulation.

After that, the adversary builds a hash-forest $\mathcal{T}$ by using the $\mathsf{ExForest}$ procedure described in Fig. 3 using the bound $N = 2t$. Due to the boundedness, with probability $1 - \frac{t^2}{2^n}$ the number of non-leaf vertices of $\mathcal{T}$ is bounded by $|\alpha| \le t_1$ and hence, $|\mathcal{T}| \le 2t_1 + 1 \le 2t = N$ and hence, such bound is never applied during the procedure, which means that for all $y \in \mathcal{T}$, the extraction call $\mathcal{E}(y)$ has indeed been performed.

Finally, B simulates A_2^P so that all its P calls are executed through the P-oracle. With probability δ we obtain a hash value x and a hash chain c such that $x \overset{c}{\rightsquigarrow} r$ for some $r \in \mathcal{R}$. Due to the strong unpredictability of A, the probability that x coincides with some of the extracted hash values r_ℓ is upper bounded by $\frac{2t}{2^{n-1}} = \frac{4t}{2^n}$. Hence, with probability at least $\delta - \frac{t^2}{2^n} - \frac{4t}{2^n}$ we have a hash value $x \notin \mathcal{T}$ and a hash chain $c = \{(c_1, b_1), (c_2, b_2), \ldots, (c_m, b_m)\}$ with output hash value $r \in \mathcal{R} \subseteq \mathcal{T}$. Let $x_0, x_1, \ldots, x_m$ be the intermediate hash values (outputs of hash links) as described in Def. 5. Let k be the smallest index such that $x_{k-1} \notin \mathcal{T}$ but $x_k \in \mathcal{T}$. For such k,

$$\begin{cases} \mathsf{H}^{\mathsf{P}}(c_k \| x_{k-1}) = x_k \text{ and } \mathsf{Ex}(x_k) \neq (c_k \| x_{k-1}) & \text{if } b_k = 0 \ ; \\ \mathsf{H}^{\mathsf{P}}(x_{k-1} \| c_k) = x_k \text{ and } \mathsf{Ex}(x_k) \neq (x_{k-1} \| c_k) & \text{if } b_k = 1 \ . \end{cases}$$

The output of B is $(c_k \| x_{k-1})$ if $b_k = 0$ or $(x_{k-1} \| c_k)$ if $b_k = 1$. Hence, B with time $t' \leq 2t$ has success $\delta' \geq \delta - \frac{t^2}{2^n} - \frac{4t}{2^n} = \frac{\delta}{2}\left(2 - \frac{1}{2^{n-1}}\frac{t^2+4t}{\delta}\right)$. Hence,

$$\frac{t'}{\delta'} \leq 4\frac{t}{\delta} \cdot \frac{1}{2 - \frac{1}{2^{n-1}}\frac{t^2+4t}{\delta}} \leq 4\frac{t}{\delta} \cdot \frac{1}{2 - \frac{1}{2^{n-1}}\left(\frac{t}{\delta}+2\right)^2} \leq 4\frac{t}{\delta} \cdot \frac{1}{2 - \frac{1}{2^{n-1}}\left(2^{\frac{n-1}{2}}\right)^2} = 4\frac{t}{\delta} \ .$$

Hence, if H^P is $4S$-secure SPrA, then $\frac{t'}{\delta'} \geq 4S$ and $\frac{t}{\delta} \geq S$, which means that H^P is S-secure against strongly unpredictable back-dating adversaries. □

Corollary 1. *Unbounded hash-tree schemes are 2^s-secure against back-dating if one uses 2^{s+2}-secure BPrA hash functions with $2s + 4$ output bits.*

This is close to the tightness achieved in the random oracle model. In our example with $s = 100$, we conclude that 204 output bits are sufficient.

5 Existing PrA Constructions are BPrA

We show that blockcipher based MD-hash functions with rate-1 compression functions (such as Davies-Meyer and Miyaguchi-Preneel) of both type I and type II are BPrA. We also show that some compression functions with uncompressing components (such as Shrimpton-Stam [14]) are BPrA.

It is unknown whether a BPrA compression function is sufficient for the Merkle-Damgård construction to be BPrA. We define a new *Unique P-query (UPQ)* property for H^P, which as we show, the MD-construction preserves.

5.1 Unique P-Query Property (UPQ)

We model the compression function F^P as a boolean (or arithmetic) circuit with P-gates. The Merkle-Damgård structure is modeled as a cascade of such circuits. For every input x define α_x as the set of P-queries that the cascade of F^P circuits makes in case of input x.

Fox every set α of P-queries, we define H^α as a function that is computed exactly like H^P, but instead of making P-queries, the answers are taken from α. Obviously, H^α is only defined for those inputs x, for which $\alpha_x \subseteq \alpha$. We denote by D_α the set of all such inputs x. This is called the domain of H^α. The range R_α is defined as $H^\alpha(D_\alpha)$. Hence, H^α is a function of type $D_\alpha \to R_\alpha$.

Definition 9. *A hash function H^P has the unique P-query property (UPQ), if for every set α, there is a function $\varphi_\alpha \colon R_\alpha \to \alpha$, such that for any efficient adversary $\alpha \leftarrow A^P$, the function φ_α is injective with overwhelming probability.*

5.2 Merkle-Damgård is UPQ-Preserving

We show that if the compression function used in the Merkle-Damgård construction has the UPQ property, then so does the iterated hash function.

Theorem 4. *The Merkle-Damgård transform is UPQ-preserving.*

Proof. We use the property of the MD-transform that for every input $x \in \{0,1\}^*$ and $\alpha_x \subseteq \alpha$ there is an input x' of the last compression round such that $H^\alpha(x) = F^\alpha(x')$. Hence, $R_\alpha = R_\alpha^H \subseteq R_\alpha^F$. As F is UPQ, there is a (computably injective) function $\varphi_\alpha^F \colon R_\alpha^F \to \alpha$. We simply define φ_α as the restriction of φ_α^F to R_α. Obviously, φ_α is injective if φ_α^F is injective. □

5.3 UPQ and PrA imply BPrA for Honest Extractors

We show that practical PrA constructions that are UPQ are also BPrA, but for this we have to assume that the PrA-extractor for H^P is *honest*:

Definition 10. *An extractor $\mathcal{E}$ for H^P is said to be* honest *if for every y and for every query string α, it holds that $\mathcal{E}(y,\alpha) \neq \perp$ only if $y \in R_\alpha$, i.e. if there is x such that $H^\alpha(x) = y$. We say that a function H^P is* honest preimage aware *(HPrA) if it is PrA with a honest extractor.*

It is easy to verify that most extractors that have been constructed in the PrA framework (like those in [9]) are honest in this sense. This is because given the output value y and the P-query string α, the extractors (e.g. in [9]) mostly traverse α in order to find suitable P-queries that together lead to y, and only in that case, output the corresponding input x. The practical extractors never try to just guess x and hope for being lucky. Note that the notion of honesty defined in [9] is somewhat weaker than in Def. 10 and require the statement $H^P(x) = y$ instead of $H^\alpha(x) = y$, but it is easy to see that the extractors in [9] satisfy the stronger version too.

Formally we can construct functions H^P that may be PrA in the general sense but not PrA when the extractor is required to be honest by Def. 10. For example, in constructions like $H^P(x) = P(x) \oplus P(x+1) \oplus P(x+1)$ if $(x, P(x)) \in \alpha$ but $(x+1, P(x+1)) \notin \alpha$ then honest extractors on input $y = P(x)$ are forced to output $\perp \leftarrow \mathcal{E}(y,\alpha)$ because $y \notin R_\alpha$. To avoid such dummy oracle queries, we may assume that the constructions H^P have the property that once $y \notin R_\alpha$, for every x the probability $\mathsf{Pr}_{P \leftarrow \Omega|\alpha}[H^P(x) = y]$ is negligible, where $\Omega \mid \alpha$ denotes the probability space of all P-oracles consistent with α. This means that whenever y is not an output that can (formally) be computed from an input x with the query string α then there are no inputs x that will lead to y with high probability and hence cannot be guessed by dishonest extractors.

Theorem 5. *If H^P is UPQ and HPrA then it is BPrA.*

Proof. If H^P is HPrA then there is a honest PrA-extractor $\mathcal{E}$. Hence, for every α that is produced by an efficient adversary, $\{y \colon \mathcal{E}(y,\alpha) \neq \perp\} \subseteq R_\alpha$. Hence, by the UPQ property as $R_\alpha \hookrightarrow \alpha$, we have $|\{y \colon \mathcal{E}(y,\alpha) \neq \perp\}| \leq |R_\alpha| \leq |\alpha|$. □

Hence, to show that a Merkle-Damgård hash function is BPrA, it is sufficient to show that its compression function satisfies UPQ. In the following, we show that many hash functions that have been proved to be PrA are actually BPrA.

5.4 The Type-I and Type-II Compression Functions are BPrA

We prove that the rate-1 blockcipher-based hash functions that have been proved to be PrA [9] are UPQ, which by Thm. 5 means that they are also BPrA.

Theorem 6. *The rate-1 block-cipher based Type-I and Type-II compression functions are UPQ.*

Proof. Assume that H^P is a rate-1 block-cipher based compression function that is either of Type-I or Type-II. In both cases, the function C_{pre} is bijective and has a inverse-function C_{pre}^{-1} that transforms a pair (x, k) (as input of an E-query) to the input (m, v) of the compression function H^P.

Let α be any P-query string that consists of ideal cipher calls in the form (x_i, k_i, y_i), where $y_i = E_{k_i}(x_i)$, or equivalently $x_i = E_{k_i}^{-1}(y_i)$. We define a function φ_α as follows. For any given output $w \in R_\alpha$, the function $\varphi_\alpha(w)$ returns the first query (x_i, k_i, y_i) in α, such that $C_{\text{post}}(C_{\text{pre}}^{-1}(x_i, k_i), y_i) = w$. Such a query must exist because of $w \in R_\alpha$. Therefore, φ_α is correctly defined.

If $\varphi_\alpha(w) = (x, k, y) = \varphi_\alpha(w')$ for some $w, w' \in R_\alpha$, then by the definition of φ_α, we have $w' = C_{\text{post}}(C_{\text{pre}}^{-1}(x_i, k_i), y_i) = w$, which means φ_α is injective. □

Consequently, the Davies-Meyer, the Matyas-Meyer-Oseas, and the Miyaguchi-Preneel compression functions as well as many others are UPQ. Due to the fact that these constructions are HPrA, we conclude based on Thm. 5 that all Type-I and iterated Type-II constructions are BPrA.

5.5 Shrimpton-Stam is BPrA

The Shrimpton-Stam [14] compression function $F^P\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ involves independent random oracles f_1, f_2 and f_3 of type $\{0,1\}^n \to \{0,1\}^n$:

$$F^P(c, x) = f_3(f_1(x) \oplus f_2(c)) \oplus f_1(x) \ .$$

Theorem 7. *The Shrimpton-Stam compression function is BPrA.*

Proof. We define the mapping φ as follows to show that Shrimpton-Stam compression function is UPQ. For any query string α and any input $y \in R_\alpha = F^\alpha(D_\alpha)$ we search from α an f_3-query $(z_3; y_3) \in \alpha$ for which there exists an f_1-query $(x_1; y_1) \in \alpha$ such that $y = y_3 \oplus y_1$ and an f_2-query $(c; y_2)$ such that $y_1 \oplus y_2 = z_3$. There must be such a query because of $y \in R_\alpha$. We define $\varphi(y)$ as the first such f_3-query in α. Now, if $\varphi(y) = (z_3; y_3) = \varphi(y')$, then there are f_1-queries $(x_1; y_1)$ and $(x_1'; y_1')$ such that $y = y_3 \oplus y_1$ and $y' = y_3 \oplus y_1'$, and f_2-queries $(c; y_2)$ and $(c'; y_2')$ such that $y_1 \oplus y_2 = z_3$ and $y_1' \oplus y_2' = z_3$. But then

$$f_1(x_1) \oplus f_2(c) = f_1(x_1') \oplus f_2(c') \ , \tag{4}$$

which is hard to satisfy for efficient adversaries, because this is equivalent of finding collisions for the Dodis-Pietrzak-Punyia (DPP) compression function $H^{f_1,f_2}(m, v) = f_1(m) \oplus f_2(v)$ [8] which is about $2^{n/4}$-secure collision-free (Appendix B). □

Note that the DPP compression function itself is not UPQ, because knowing only five P-queries, say $y_1 = f_1(m_1)$, $y_2 = f_1(m_2)$, $y_3 = f_1(m_3)$, $y_1' = f_2(v_1)$, and $y_2' = f_2(v_2)$ allows one to compute six different outputs of H^{f_1,f_2}. We can show in a similar way that DPP is not BPrA (Appendix A). □

References

1. Bayer, D., Haber, S., Stornetta, W.-S.: Improving the efficiency and reliability of digital timestamping. In: Sequences II: Methods in Communication, Security, and Computer Sci., pp. 329–334. Springer, Heidelberg (1993)
2. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: the 1st ACM conference on Computer and Communications Security: CCS'93, pp. 62–73. ACM (1993)
3. Bellare, M., Rogaway, P.: The exact security of digital signatures - How to sign with RSA and Rabin. In: Maurer, U.M. (ed.): EUROCRYPT '96. LNCS 1070, pp. 399–416. Springer, Heidelberg (1996)
4. Buldas, A., Laanoja, R.: Security proofs for hash tree time-stamping using hash functions with small output size. In: Boyd, C., Simpson, L. (eds.): ACISP 2013. LNCS 7959, pp. 235–250. Springer, Heidelberg (2013)
5. Buldas, A., Niitsoo, M.: Optimally tight security proofs for hash-then-publish time-stamping. In: Steinfeld, R., Hawkes, P. (eds.): ACISP 2010. LNCS 6168, pp. 318–335. Springer, Heidelberg (2010)
6. Buldas, A., Saarepera, M.: On provably secure time-stamping schemes. In: Lee, P.J. (ed.): ASIACRYPT 2004. LNCS 3329, pp. 500–514. Springer, Heidelberg (2004)
7. Canetti, R., Goldreich, O., Halevi, S.: The random oracle methodology, revisited. JACM 51 (4), 557–594 (2004)
8. Dodis, Y., Pietrzak, K., Puniya, P.: A new mode of operation for blockciphers and length-preserving MACs. In: Smart, N. (ed.): EUROCRYPT 2008. LNCS 4965, pp. 198–219, Springer (2008)
9. Dodis, Y., Ristenpart, T., Shrimpton, T.: Salvaging Merkle-Damgård for practical applications. In: Joux, A. (ed.): Eurocrypt 2009. LNCS 5479, pp. 371–388. Springer, Heidelberg (2009)
10. Haber, S., Stornetta, W.-S.: How to time-stamp a digital document. Journal of Cryptology 3(2), 99–111 (1991)
11. Luby, M.: Pseudorandomness and Cryptographic Applications. Princeton University Press, Princeton (1996)
12. Merkle, R.C.: Protocols for public-key cryptosystems. In: Proceedings of the 1980 IEEE Symposium on Security and Privacy, pp. 122–134 (1980)
13. Preneel, B., Govaerts, R., Vandewalle, J.: Hash functions based on block ciphers: s synthetic approach. In: Stinson, D.R. (ed.): CRYPTO'93. LNCS 773, pp. 368–378. Springer (1993)
14. Shrimpton, T., Stam, M.: Building a collision-resistant compression function from non-compressing primitives. In: Aceto, L. et al (Eds.): ICALP 2008, Part II. LNCS 5126, pp. 643–654. Springer Heidelberg (2008)
15. Stam, M.: Blockcipher-based hashing revisited. In: Dunkelman, O. (Ed.): FSE 2009. LNCS 5665, pp. 67–83. Springer (2009)

A The DPP Hash Function is not BPrA

As DPP is PrA [9], let $\mathcal{E}$ be any suitable extractor. We define two adversaries: a PrA adversary B and a query-list adversary A.

The PrA adversary $B^{f_1,f_2,\mathrm{Ex}}$:

1. Computes $y_1 \leftarrow f_1(m_1)$, $y_2 \leftarrow f_1(m_2)$, $y_3 \leftarrow f_1(m_3)$, $y'_1 \leftarrow f_2(v_1)$, and $y'_2 \leftarrow f_2(v_2)$ by using 5 queries to oracles f_1, f_2, after which $|\,\alpha\,| = 5$, assuming that all m_1, m_2, m_3, v_1, v_2 are different.
2. Checks if there are collision-pairs for H^{f_1,f_2} among the six inputs $X_1 = (m_1, v_1)$, $X_2 = (m_1, v_2)$, $X_3 = (m_2, v_1)$, $X_4 = (m_2, v_2)$, $X_5 = (m_3, v_1)$, $X_6 = (m_3, v_2)$.
3. Composes 6 inputs $w_1 = y_1 \oplus y'_1$, $w_2 = y_1 \oplus y'_2$, $w_3 = y_2 \oplus y'_1$, $w_4 = y_2 \oplus y'_2$, $w_5 = y_3 \oplus y'_1$, $w_6 = y_3 \oplus y'_2$.
4. Finds the smallest $i \in \{1, \ldots, 6\}$, such that $\mathrm{Ex}(w_i) \neq X_i$ and outputs X_i. If there is no such i, gives up and outputs $\perp$.

The query-list adversary A^{f_1,f_2} just computes $y_1 \leftarrow f_1(m_1)$, $y_2 \leftarrow f_1(m_2)$, $y_3 \leftarrow f_1(m_3)$, $y'_1 \leftarrow f_2(v_1)$, and $y'_2 \leftarrow f_2(v_2)$ by using 5 queries to oracles f_1, f_2, assuming that all m_1, m_2, m_3, v_1, v_2 are different and the same that B uses. After that, A outputs the query list α and stops.

If there is i such that $\mathrm{Ex}(w_i) \neq X_i$, then B will fool the extractor, and if there is no such i, we have that $|\,\{y\colon \mathcal{E}(y, \alpha) \neq \perp\}\,| \geq 6 > 5 = |\,\alpha\,|$, which means that A breaks the extractor.

B Collision-Freeness Bounds for DPP

Lemma 1. *The function $F^{f_1,f_2}(x_1,x_2) = f_1(x_1) \oplus f_2(x_2)$ where f_1 and f_2 are independent random oracles of type $\{0,1\}^n \to \{0,1\}^n$, is $2^{n/4}$-collision resistant.*

Proof. If an adversary A^{f_1,f_2} makes t_1 calls to f_1 and t_2 calls to f_2, then it has $t_1 t_2$ known values of F, and $\frac{t_1 t_2 (t_1 t_2 - 1)}{2}$ pairs of inputs (potential collisions). Each pair is a collision with probability 2^{-n}. Hence, by the union bound, the collision probability δ is upper bounded by

$$\delta \leq \frac{t_1 t_2 (t_1 t_2 - 1)}{2^{n+1}} \leq \frac{t^4}{2^{n+1}} \ .$$

Hence, $\frac{t}{\delta} \geq 2^{n/4}$ for any t-time adversary with success probability δ. □

Lemma 2. *There is a collision finder for $F^{f_1,f_2}(x_1,x_2) = f_1(x_1) \oplus f_2(x_2)$ with time-success ratio $5 \cdot 2^{n/4}$.*

Proof. Let A^{f_1,f_2} be an adversary that calls both f_1 and f_2 exactly $m = 2 \cdot 2^{n/4}$ times. Let $x_1, \ldots, x_m$ and $x'_1, \ldots, x'_m$ be the f_1- and f_2-calls respectively. Then it searches a collision among the possible $\frac{m^2(m^2-1)}{2}$ pairs

$$(f_1(x_i) \oplus f_2(x'_{i'}), \quad f_1(x_j) \oplus f_2(x'_{j'}))$$

of outputs of F^{f_1,f_2}, where either $i \neq j$ or $i' \neq j'$. Let $X_{i,j,i',j'}$ denote a 0/1 random variable which is 1 if and only if $f_1(x_i) \oplus f_2(x'_{i'}) = f_1(x_j) \oplus f_2(x'_{j'})$. Let $X = \sum_{i,j,i',j'} X_{i,j,i',j'}$, where the summation is over all indices in the range $[1...m]$ such that either $i \neq j$ or $i' \neq j'$. Note that the random variables $X_{i,j,i',j'}$ are pairwise independent and their mathematical expectation is $p = \frac{1}{2^n}$. Therefore, we can apply the Chebyshev bound:

$$\mathsf{Pr}\left[\,|\,X - \mathrm{E}\,[X]\,|\,> \tau\right] \leq \frac{\mathrm{Var}(X)}{\tau^2} \ ,$$

where $\mathrm{E}\,[X] = \frac{m^2(m^2-1)}{2^{n+1}} = \frac{16 \cdot 2^{n/2}(2^{n/2} - \frac{1}{4})}{2^{n+1}} > 7$ and $\mathrm{Var}(X) = \frac{m^2(m^2-1)(1-2^{-n})}{2^{n+1}} \leq 8$. Hence,

$$\mathsf{Pr}\left[X = 0\right] \leq \mathsf{Pr}\left[\,|\,X - \mathrm{E}\,[X]\,|\,> 7\right] \leq \frac{8}{49} < \frac{1}{6} \ ,$$

Hence, the adversary that makes $t = 4 \cdot 2^{n/4}$ oracle calls succeeds in creating a collision with probability $\delta > \frac{5}{6}$. Hence, $\frac{t}{\delta} \leq \frac{24}{5} \cdot 2^{n/4} < 5 \cdot 2^{n/4}$. □