Security Proofs for Hash Tree Time-Stamping using Hash Functions with Small Output Size

Ahto Buldas [1] [2]   Risto Laanoja[1] [2]

Guardtime AS, Tammsaare tee 60, 11316 Tallinn, Estonia.

Tallinn University of Technology, Raja 15, 12618 Tallinn, Estonia.

October 26, 2013 — Theory Days at Saka

Motivation

We have an **application** (a linking-based time-stamping system) that is built on **cryptographic hash functions**.

There is also a security proof: if H is $2^{n/2}$-collision resistant (it has n output bits), then our application is 2^s-secure. Proof is *asymptotically optimally tight at our case.*

What to do if practical hash functions with e.g. 160, 256 bit output do not provide reasonable security? Let's consider other abstractions:

Random Oracle	H *is a random function.*
Pseudorandom Oracle	H *is built from an ideal primitive* P *and is indifferentiable from a random oracle.*
Preimage Awareness	*if an adversary* A *commits* y *and later comes up with* x*, such that* $y = H(x)$*, then it is safe to say that* A *knew* x *before committing* y*.*

Previous Results

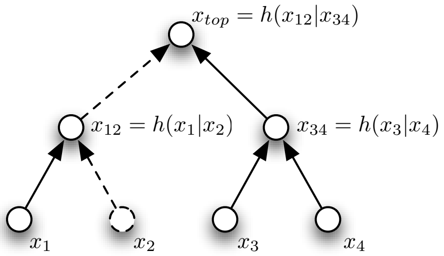
It was proven by Buldas, Niitsoo in 2010 that they have provided asyptotically optimally tight proof based on CR assumption. So when we take configuration of an existing implementation we get following necessary hash function output length:

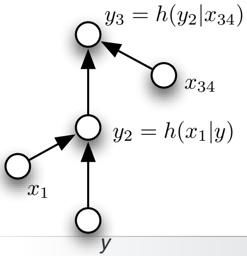
Proof	Output Size $n = n(C, s)$	$n(2^{64}, 80)$
Asiacrypt 2004	$n = 2\log_2 C + 4s + 2$	448
ACISP 2010	$n = \log_2 C + 3s + 8$	**312**

where C is the potential size of the hash tree, which must be bounded.

Hash Tree

- Hash tree (Merkle tree): authenticated dictionary data structure
- Fixing the top does not allow any modifications
- Leaves: hashes of input documents
- Signature token: sufficient data to re-compute the top starting from a particular document, i.e. *membership proof*. Size $\log C$ when C is capacity.





Hash-Tree Based Time-Stamping and its Security

During every time unit t the time-stamping server:

- receives a list $\mathcal{X}_t = (x_1, \ldots, x_m)$ of n-bit requests from clients,
- finds the root hash $r_{(t)} = \mathcal{T}(x_1, \ldots, x_m)$ of a hash tree $\mathcal{T}$ and
- publishes $r_{(t)}$ in a public repository $\mathcal{R} = (r_{(1)}, r_{(2)}, \ldots, r_{(t)})$.

Each request x_i is then provided with a hash chain c_i (the *time stamp token* for x_i) that proves the participance of x_i in the computation of the root hash $r_{(t)}$.

Security: A malicious server is unable to add new (fresh, with high min-entropy) requests to already published hash trees.

Formalizing the Security Notion

We have two-stage unpredictable adversary $A = (A_1, A_2)$ potentially colluding with the server.

- At stage A_1 the server operates normally, creating commitments $\mathcal{R}$
- Then A_2 presents an unpredictable x and a hash chain c so that $x \overset{c}{\rightsquigarrow} r$ for an $r \in \mathcal{R}$.

A time-stamping scheme is S-secure against back-dating if for every t-time strongly unpredictable A:

$$\mathsf{Pr}\left[(\mathcal{R}, a) \leftarrow A_1, (x, c) \leftarrow A_2(\mathcal{R}, a) : \; x \overset{c}{\rightsquigarrow} \mathcal{R}, \; \ell(c) \in \mathcal{S}\right] \le \frac{t}{S} \tag{1}$$

Proof on RO Assumption

- Showing probability of finding $x = x_1 \| x_2$ so that A_1 ‘haven”t seen’ x_1 or x_2, but ‘have seen’ $h(x)$ during time t.
- $\frac{t}{\delta} \geq 2^{\frac{n-1}{2}}$.
- Do not care if hash tree is bounded or not!
- Idealized construction, indicating what could be achieved on stronger assumptions

Assumption	Output Size $n = n(C, s)$	$n(2^{64}, 80)$
CR	$n = \log_2 C + 3s + 8$	312
RO	$n = 2s + 1$	161

Proof on Preimage Awareness Assumption

- PrA was introduced by Dodis, Ristenpart et al at Eurocrypt 2009.
- Concept similar to plaintext awareness.
- Formal model using extractor and some wrapper oracles, so that all h-calls are recorded in advice string a.
- RO $>>$ PrA $>>$ CR
- Preserved by Merkle-Damgård construct

Proof outline: We run experiment where let the extractor loose on $\mathcal{R}$, then try to hit any extracted hash value with x. The attacker's time-success ratio is $\frac{t}{\delta} \geq \frac{S}{2C}$.

Assumption	Output Size $n = n(C, s)$	$n(2^{64}, 80)$
CR	$n = \log_2 C + 3s + 8$	312
PrA	$n = 2(\log_2 C + s + 1)$	**290**
RO	$n = 2s + 1$	161

Stong Preimage Awareness

- The result on PrA assumption is not impressive, we got linear dependence on C. Intuitively $\log_2 C$ relation should be possible.
- Let's find reasonable strenghtenings of the PrA definition. Extracting full hash tree is not necessary, we should restrict it to hash chain.
- We chose to limit the amount of information available in advice string a, using the "oldest possible" version of a.

Strong PrA: If an adversary A:

- commits y and y', and later
- outputs x and x' such that $y = H(x)$ and $y' = H(x')$, and
- y' can be extracted (parsed) from x in an obvious way,

then we may assume that *A knew x' before y was committed*.

Example: In time-stamping application this parser de-concatenates a pair of hashes.

Proof on SPrA assumption

The formal experiment setup is even uglier than standard PrA, thus not displayed here. Sketch of proof: We are able to change the order of things:

1. Simulate: $(\mathcal{R}, a) \leftarrow A_1$
2. Call $\mathsf{Ext}(r)$ for every $r \in \mathcal{R}$; thanks to modifications it 'fixes' a
3. Simulate A_2
4. use the Ext-oracle to extract the hash values along c (less than $2\log_2 C$)

Time-success ratio is here $\frac{t}{\delta} \geq \frac{S}{4\log_2 C}$.

Assumption	Output Size $n = n(C, s)$	$n(2^{64}, 80)$
CR	$n = \log_2 N + 3s + 8$	344
PrA	$n = 2(\log_2 C + s + 1)$	290
SPrA	$n = 2(\log_2 \log_2 C + s + 2)$	**176**
RO	$n = 2s + 1$	161

Other results

- Show that SPrA implies PrA, thus RO $>>$ SPrA $>>$ PrA $>>$ CR.
- Necessity of C:
  - Construct a PrA hash function that is insecure for unbounded time-stamping schemes,
  - ... same for SPrA.

Next Steps

Honestly, SPrA is not exactly elegant construction.

Bounded Preimage Awareness: number of inputs y where extractor outputs something is bounded by number of P-calls (size of a).

- Tight proof, almost RO
- Security loss does not depend on capacity,
- Found a stronger property which is:
  - preserved by Merkle Damgård construct,
  - provided by tested compression functions: Davis-Meyer, etc.

Proof in standard model almost as strong as RO!

Thank You!