Hash-based signatures & Hash-and-sign without collision-resistance

Andreas Hülsing

21.12.2017
Basic Construction
Lamport-Diffie OTS [Lam79]

Message $M = b_1, ..., b_m$, OWF $H$  

* = $n$ bit

```
\begin{align*}
\text{SK} & : sk_{1,0} \quad sk_{1,1} \\
\text{PK} & : pk_{1,0} \quad pk_{1,1} \\
\text{Sig} & : sk_{1,b_1} \\
\end{align*}
```

```
\begin{align*}
\text{SK} & : \ldots \quad \ldots \quad \ldots \\
\text{PK} & : \ldots \quad \ldots \quad \ldots \\
\text{Sig} & : \ldots \quad \ldots \quad \ldots \\
\end{align*}
```

```
\begin{align*}
\text{SK} & : sk_{m,0} \quad sk_{m,1} \\
\text{PK} & : pk_{m,0} \quad pk_{m,1} \\
\text{Sig} & : sk_{m,b_m} \\
\end{align*}
```
Merkle’s Hash-based Signatures

\[ \text{SIG} = (i=2, H, \text{OTS}_1, \text{OTS}_2, \text{OTS}_3, \text{OTS}_4) \]
Security

Theorem:
MSS is eu-cma-secure if OTS is a one-time eu-cma secure signature scheme and H is a random element from a family of collision resistant hash functions.
Reduction

Input: $k, pk_{OTS}$

1. Choose random $0 \leq i < 2^h$
2. Generate key pair using $pk_{OTS}$ as $i$th OTS public key and $H \leftarrow h_k$
3. Answer all signature queries using $sk$ or sign oracle (for index $i$)
4. Extract OTS-forgery or collision from forgery
Reduction (Step 4, Extraction)

Forgery: \((i^*, \sigma^*_{OTS}, pk^*_{OTS}, AUTH)\)

1. If \(pk^*_{OTS}\) equals OTS pk we used for \(i^*\) OTS, we got an OTS forgery.
   - Can only be used if \(i^* = i\).

2. Else adversary used different OTS pk.
   - Hence, different leaves.
   - Still same root!
   - Pigeon-hole principle: Collision on path to root.
Winternitz-OTS
Recap LD-OTS \cite{Lam79}

**Message** \( M = b_1, \ldots, b_m \), OWF \( H \)

\[ * = n \text{ bit} \]

**SK**
- \( sk_{1,0} \)
- \( sk_{1,1} \)
- \( \ldots \)
- \( sk_{m,0} \)
- \( sk_{m,1} \)

**PK**
- \( pk_{1,0} \)
- \( pk_{1,1} \)
- \( \ldots \)
- \( pk_{m,0} \)
- \( pk_{m,1} \)

**Sig**
- \( sk_{1,b_1} \)
- \( \ldots \)
- \( sk_{m,b_m} \)
LD-OTS in MSS

\[ \text{SIG} = (i=2, \text{verify sign}, \text{document}, \text{verify authenticity}) \]

Verification:
1. Verify \text{document}
2. Verify authenticity of \text{document}

We can do better!
Trivial Optimization

Message $M = b_1, \ldots, b_m$, OWF $H$ 

* = n bit
Optimized LD-OTS in MSS

\[ \text{SIG} = (i=2, X, 0, 0, 0) \]

Verification:
1. Compute \( X \) from \( \text{SIG} \)
2. Verify authenticity of \( X \)

Steps 1 + 2 together verify \( X \)
Germans love their „Ordnung“!

**Message** $M = b_1, \ldots, b_m$, OWF $H$

**SK:** $sk_1, \ldots, sk_m, sk_{m+1}, \ldots, sk_{2m}$

**PK:** $H(sk_1), \ldots, H(sk_m), H(sk_{m+1}), \ldots, H(sk_{2m})$

**Encode $M$:** $M' = M \mid \neg M = b_1, \ldots, b_m, \neg b_1, \ldots, \neg b_m$

(Instead of $b_1, \neg b_1, \ldots, b_m, \neg b_m$)

**Sig:** $\text{sig}_i =
\begin{cases} 
  sk_i, & \text{if } b_i = 1 \\
  H(sk_i), & \text{otherwise}
\end{cases}$

Checksum with bad performance!
Optimized LD-OTS

Message \( M = b_1, \ldots, b_m \), OWF \( H \)

SK: \( sk_1, \ldots, sk_m, sk_{m+1}, \ldots, sk_{m+\log m} \)

PK: \( H(sk_1), \ldots, H(sk_m), H(sk_{m+1}), \ldots, H(sk_{m+\log m}) \)

Encode \( M \): \( M' = b_1, \ldots, b_m, \neg \sum_{1}^{m} b_i \)

\[ \text{Sig: } \quad \begin{cases} 
  sk_i & \text{, if } b_i = 1 \\
  H(sk_i) & \text{, otherwise}
\end{cases} \]

IF one \( b_i \) is flipped from 1 to 0, another \( b_j \) will flip from 0 to 1
Function chains

Function family: $H_n := \{h_k: \{0,1\}^n \rightarrow \{0,1\}^n\}$

$h_k \leftarrow H_n$

Parameter $w$

Chain:
$c^i(x) = h_k(c^{i-1}(x)) = \underbrace{h_k \circ h_k \circ \ldots \circ h_k}_\text{i-times} (x)$

$c^0(x) = x$

$c^1(x) = h_k(x)$

$c^{w-1}(x)$
WOTS

Winternitz parameter $w$, security parameter $n$, message length $m$, function family $H_n$

**Key Generation:** Compute $l$, sample $h_{\kappa}$

- $c^0(sk_1) = sk_1$
- $c^1(sk_1)$
- $c^1(sk_i)$
- $c^0(sk_i) = sk_i$

$pk_1 = c^{w-1}(sk_1)$

$pk_i = c^{w-1}(sk_i)$
WOTS Signature generation

Signature:
\[ \sigma = (\sigma_1, \ldots, \sigma_\ell) \]
WOTS Signature Verification

Verifier knows: $M$, $w$

Signature:
$\sigma = (\sigma_1, \ldots, \sigma_\ell)$
WOTS Function Chains

For $x \in \{0,1\}^n$ define $c^0(x) = x$ and

- WOTS: $c^i(x) = h_k(c^{i-1}(x))$
- WOTS$: c^i(x) = h_{c^{i-1}(x)}(r)$
- WOTS$^+ : c^i(x) = h_k(c^{i-1}(x) \oplus r_i)$
WOTS Security

**Theorem (informally):**

\textbf{W-OTS} is strongly unforgeable under chosen message attacks if \( H_n \) is a \textit{collision resistant family of undetectable one-way functions}.

\textbf{W-OTS} is existentially unforgeable under chosen message attacks if \( H_n \) is a \textit{pseudorandom function family}.

\textbf{W-OTS} is strongly unforgeable under chosen message attacks if \( H_n \) is a \textit{2^{nd}-preimage resistant family of undetectable one-way functions}. 
XMSS
XMSS

Tree: Uses bitmasks

Leafs: Use binary tree with bitmasks

OTS: WOTS$^+$

Message digest:
Randomized hashing

Collision-resilient
-> signature size halved
Multi-Tree XMSS

Uses multiple layers of trees

-> Key generation
(= Building first tree on each layer)
Θ(2^h) → Θ(d2^{h/d})

-> Allows to reduce worst-case signing times
Θ(h/2) → Θ(h/2d)
How to Eliminate the State
Protest?
Few-Time Signature Schemes
Recap LD-OTS

Message $M = b_1, \ldots, b_n$, OWF $H$  

$\text{SK}$  
$\text{PK}$  
$\text{Sig}$
HORS [RR02]

Message M, OWF H, CRHF H’

Parameters t=2^a,k, with m = ka (typical a=16, k=32)
HORS mapping function

Message $M$, OWF $H$, CRHF $H'$

Parameters $t=2^a,k$, with $m = ka$ (typical $a=16, k=32$)
**HORS**

Message $M$, OWF $H$, CRHF $H'$

Parameters $t=2^a, k$, with $m = ka$ (typical $a=16$, $k=32$)

```
<table>
<thead>
<tr>
<th>SK</th>
<th>pk_1</th>
<th>pk_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>sk_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sk_2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>sk_{t-1}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sk_t</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>PK</th>
<th>pk_1</th>
<th>pk_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>pk_1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>pk_2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>pk_{t-1}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>pk_t</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>H'(M)</th>
<th>b_1</th>
<th>b_2</th>
<th>b_a</th>
<th>b_{a+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Mux</th>
<th>i_1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>sk_{i1}</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>Mux</th>
<th>i_k</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>sk_{i_k}</td>
<td></td>
</tr>
</tbody>
</table>
```

* = $n$ bit
HORS Security

- $M$ mapped to $k$ element index set $M^i \in \{1, \ldots, t\}^k$
- Each signature publishes $k$ out of $t$ secrets
- Either break one-wayness or...

- $r$-Subset-Resilience: After seeing index sets $M_j^i$ for $r$ messages $msg_j$, $1 \leq j \leq r$, hard to find $msg_{r+1} \neq msg_j$ such that $M_{r+1}^i \in \bigcup_{1 \leq j \leq r} M_j^i$.

- Best generic attack: $\text{Succ}_{r-SSR}(A, q) = q \left( \frac{rk}{t} \right)^k$
  $\rightarrow$ Security shrinks with each signature!
HORST

Using HORS with MSS requires adding PK (tn) to MSS signature.

HORST: Merkle Tree on top of HORS-PK
- New PK = Root
- Publish Authentication Paths for HORS signature values
- PK can be computed from Sig
- With optimizations: \( tn \rightarrow (k(\log t - x + 1) + 2^x)n \)
  - E.g. SPHINCS-256: 2 MB → 16 KB
- Use randomized message hash
SPHINCS

• Stateless Scheme
• \( \text{XMSS}^{\text{MT}} + \text{HORST} \) + (pseudo-)random index
• Collision-resilient
• Deterministic signing
• SPHINCS-256:
  • 128-bit post-quantum secure
  • Hundrest of signatures / sec
  • 41 kb signature
  • 1 kb keys
Hash & sign without collision resistance
PKC with variable-length inputs

- Hard to build variable input length constructions (Schoolbook RSA)
- Direct constructions are inefficient (See the hash-based constructions for long messages)
- Full adversarial control over input can harm security (Schoolbook RSA)

- For encryption: Use hybrid encryption -> PKE only encrypts “short” symmetric key.
- For signatures: Hash-and-sign -> SIG only signs “short” message digest
Security of Hash & Sign

Theorem.

Given an EU-CMA-secure Signature scheme for fixed-length inputs and a collision-resistant family of hash functions. The hash & sign signature scheme for variable-length inputs, obtained by first applying a hash function from the family and then signing the resulting digest, is EU-CMA-secure.
Reminder EU-CMA

\[ \text{PK}, 1^n \rightarrow A \rightarrow (\sigma^*, M^*) \rightarrow \text{SIGN} \]

\[ (\sigma_i, M_i) \rightarrow q \rightarrow \text{SK} \]

Success if \( M^* \neq M_i \), \( \forall i \in [0, q] \) and \( \text{Verify}(pk, \sigma^*, M^*) = \text{Accept} \)
Security of Hash & Sign Reduction

Input: pk & access to SIGN, hash function key k.
Output: Forgery or collision.

- Give (pk, k) as public key to A.
- Answer signing queries $M_i$ computing $md_i = h_k(M_i)$ and sending $md_i$ to SIGN.
- Given forgery $(M^*, \sigma^*)$ compute $md^* = h_k(M^*)$
- If $\exists i \leq q: md^* = md_i$, return collision $(M^*, M_i)$
- Else, return forgery $(md^*, \sigma^*)$
ARE WE
DONE YET?
Recall: Attacks on Hash Functions

- **MD5** Collisions (theo.)
- **SHA-1** Collisions (theo.)
- **MD5** Collisions (practical!)
- **MD5 & SHA-1** No (Second-) Preimage Attacks!

Timeline:
- 2004
- 2005
- 2008
- 2015
Target-collision resistance (TCR)

\[ H_n := \{ h_k : \{0,1\}^{m(n)} \rightarrow \{0,1\}^n \} \]

\[ h_k \leftarrow H_n \]

Success if
\[ h_k(x_c) = h_k(x^*) \]

Run 1
- \( x_c, k, S \)
- \( A \)
- \( x_c, S \)

Run 2
- \( x^* \)
Target-collision resistance

Collision-Resistance

2^{nd}-Preimage-Resistance

One-way

TCR

Pseudorandom

Assumption / Attacks

stronger / easier to break

weaker / harder to break
TCR vs. SPR

- TCR -> SPR: Note that A in „Run 2“ is just an SPR adversary!
  - Every TCR function family must be SPR!
- SPR -> TCR: Per construction
  - Given a SPR function, use key of length $m$
  - Key the function by XORing key to message:
  - $h_k'(x) = h(x \oplus k)$
TCR of „XOR construction“
Reduction

Input: SPR function $h$, target $x_c$
Output: Second-preimage $x^*$ for $x_c$

- Run TCR adversary $A$ to obtain $x'_c$
- Compute $k = x'_c \oplus x_c$
- Run $A$ on $k$ to obtain $x''$
- Output $x^* = x'' \oplus k$

Note:

$h'_k(x'_c) = h(x'_c \oplus k) = h(x'_c \oplus x'_c \oplus x_c) = h(x_c)$

$= h'_k(x'^{*}) = h(x'^{*} \oplus k) = h(x^*)$
TCR Hash & Sign

Input: Fixed input-length signature scheme SIG=(KeyGen,Sign, Verify), TCR function family
Output: Variable input length signature scheme SIG’

• KeyGen’ = KeyGen
• Sign’: Given $M, sk$ sample random function key $k$, compute $md = h_k(M)$, compute $\sigma = Sign((k||md), sk)$, return $\sigma’ = (k, \sigma)$
• Verify’: Compute $md = h_k(M)$, output $Verify((k||md), \sigma, pk)$
Security of TCR Hash & Sign Reduction

Input: pk & access to SIGN, hash function key k „oracle“. Output: Forgery or target-collision.

• Give pk as public key to A.
• Answer signing queries $M_i$ sending $M_i$ to oracle to get $k_i$, computing $md_i = h_{k_i}(M_i)$ and sending $(k_i, md_i)$ to SIGN.
• Given forgery $(M^*, \sigma^*) = (M^*, (k^*, \sigma^*))$ compute $md^* = h_{k^*}(M^*)$
• If $\exists i \leq q: md^* == md_i$ & $k_i == k^*$, return collision $(k_i, M^*, M_i)$
• Else, return forgery $((k^* || md^*), \sigma^*)$
Note:

• If the $k_i$ weren’t signed, the second-to-last step wouldn’t work!

• An adversary could output a forgery $(M^*, \sigma^*) = (M^*, (k^*, \sigma^*))$ such that

$$h_{k^*}(M^*) = md^* = md_i = h_{k_i}(M_i),$$

for some $i \leq q$, but $k^* \neq k_i$. Hence, $(k_i, M^*, M_i)$ is no target-collision! However, $\sigma_i$ still works as signature for $md^*$.

$\Rightarrow$ EU-CMA of fixed-length input SIG not violated!

• Really need to sign keys!
Problem using XOR-construction: Key-length == Message-length

\[ \text{Sign}((k||md), sk) \]

- We have to sign messages of length hash function key-length + digest length
- Even decreased possible message length!
**Bellare & Rogway:**

**XOR-tree**

- Reduce key-length to $2n \log_2 \frac{m}{n}$. (m-bit message, n-bit digest)
- Given SPR hash $h: \{0,1\}^{2n} \rightarrow \{0,1\}^n$ split message into $\frac{m}{n}$, n-bit blocks
- Build binary hash tree of height $t = \log_2 \frac{m}{n}$ using message blocks as leaves. (Assume t is integer)
- Keying:
  - Split key into t 2n-bit blocks $b_j$ called bitmasks.
  - XOR values on level $j$ with j-th bitmask before applying $h$
- $md$ is root of tree
XOR-tree
Reduction

Input: SPR function $h$, target $x_c$
Output: Second-preimage $x^*$ for $x_c$

- Select random node $N_{i',j'}$ in tree
- Given message $M$, compute tree up to children $N_{2i',j'-1}, N_{2i'+1,j'-1}$ of $N_{i',j'}$, sampling the required $b_j$ at random.
- Select $b_{j'} = (N_{2i',j'-1} \parallel N_{2i'+1,j'-1}) \oplus x_c$
XOR-tree
Reduction, cont’d

Input: SPR function $h$, target $x_c$
Output: Second-preimage $x^*$ for $x_c$

- Select $b_{j'} = (N_{2i'}, j' - 1 \parallel N_{2i' + 1, j' - 1}) \oplus x_c$

\[
\Rightarrow N_{i', j'} = h \left( (N_{2i'}, j' - 1 \parallel N_{2i' + 1, j' - 1}) \oplus b_{j'} \right)
\]
\[
= h \left( \bigoplus (N_{2i'}, j' - 1 \parallel N_{2i' + 1, j' - 1}) \oplus x_c \right) = h(x_c)
\]
XOR-tree
Reduction, cont’d

• Select remaining $b_j$ at random
• Run $M^* = A(M, k, S)$ with $k = (b_1, ..., b_t)$
• Compute $x^* = (N_{2i',j'-1} \parallel N_{2i'+1,j'-1}) \oplus b_{j'}$ from $M^*$ and output it.
• Remember MSS reduction? As $M^* \neq M$ but $h_k(M^*) = h_k(M)$ there must exist at least one node which is equal in both trees but has different children in the trees obtained from $M^*$ and $M$.
• We selected a random node, so hit it with probability $(\#nodes)^{-1} = \left(\frac{2m}{n} - 1\right)^{-1}$
XOR-tree

- $2n \log_2 \frac{m}{n}$ still pretty large!

- Can be improved, but not asymptotically.

- For a large group of constructions, key length $\Theta(\log_2 m)$ is optimal!
WE LOST THE FIGHT
BUT NOT THE
WAAAAAAAAAAAR
Approach 1: ROM

• An RO, keyed by prefixing with key, is a TCR hash function.

• No way to get TCR from Merkle-Damgard construction without (implicitly) assuming collision resistance.
Approach 2: eTCR

\[ H_n := \{h_k: \{0,1\}^{m(n)} \rightarrow \{0,1\}^n\} \]

\[ h_k \leftarrow H_n \]

Success if

\[ h_k(x_c) = h_{k^*}(x^*) \]
Note:

• If the $k_i$ weren’t signed, the second-to-last step wouldn’t work!

• An adversary outputs a forgery $(M^*, \sigma^*) = (M^*, (k^*, \sigma^*))$

\[
h_{k^*}(M^*) = m_{\sigma^*} = m_{\sigma^*} = h_{k_i}(M_i),
\]

for some $i \leq q$, but $k^* \neq k_i$. Hence, $(k_i, M^*, M_i)$ is no target-collision! However, $\sigma_i$ still works as signature for $md^*$.

$\Rightarrow$ EU-CMA of fixed-length input SIG not violated!

• Really need to sign keys!

Now ok! Don’t need to sign keys!

Don’t need to send key along with signature!
Approach 2: eTCR

• Can get short-key-constructions in standard model (from pretty strong, non-common assumptions)

• RO with prefixed key has this property

• Just require it as property from new hash functions (SHA-3 competition).
Real life version: Randomized Hashing

• eTCR-hash & sign:
  \[ \text{SIGN}'(M, sk) = (R, \text{SIGN}(H(R \parallel M), sk_a)) \] where
  \[ R = \text{PRF}(sk_b, M) \]

• Not “provably weaker assumption than collision resistance”.

• In practice: Known collision does not help as R is unpredictable.
The multi-target issue

• What‘s the attack target?
• A special single Person?
• Wouldn‘t anyone in the company work?
• Wouldn‘t the ability to forge for any of Google‘s servers be sufficient?
• ...
The multi-target issue

• Neglected in EU-CMA model
• Relevant in practice!
• Can significantly harm security!
• Assume security of signature relies on hardness of finding preimage...
• ... Finding single preimage takes $\mathcal{O}(2^n)$ time
• ... Finding one out of $p$ preimages takes $\mathcal{O}\left(\frac{2^n}{p}\right)$ time
Multi-target eTCR attack

- Collect as many signatures of as many possible victims as possible.
- Store the message digests used in the sigs in sorted DB.
- Start search for $k, x$ such that $h_k(x)$ matches entry in DB.
- Complexity goes down from $O(2^n)$ to $O\left(\frac{2^n}{p}\right)$.
Preventing multi-target attacks, state of the art

• Not known how to prevent this for single user (without state / signing k)
• Can reduce #targets p:
  \[ \text{SIGN'}(M, sk) = (R, \text{SIGN}(H(R \parallel pk \parallel M), sk_a)) \]
  where \( R = \text{PRF}(sk_b, M) \)
• Makes hash depending on pk
• For „good“ hash function H, attacker has to decide on target for each evaluation of H.
• For stateful schemes: Also add signature index.
Multi-target attack prevention

• Used in recent Internet Drafts (e.g. draft-irtf-cfrg-eddsa-08, draft-irtf-cfrg-xmss-hash-based-signatures-07)

• If scheme already uses randomness (EC-DSA: $R = [r]B$) might reuse that -> authenticates R!

• Price for avoiding collision resistance...
Thank you!

Questions?

For references & further literature see
https://huelsing.wordpress.com/hash-based-signature-schemes/literature/