Simplified security arguments for hash-based signatures

Andreas Hülsing
Eindhoven University of Technology
The quantum threat

• **Shor’s algorithm** breaks RSA, (EC)DSA, (EC)DH,…

• **Grover’s algorithm** asymptotically reduces complexity of brute-force search attacks by a square-root factor.
Why care today

• **EU** launched a one billion Euro project on quantum technologies

• Similar range is spent in **China**

• **US** administration passed a bill on spending $1.275 billion US dollar on quantum computing research

• **Google**, **IBM**, **Microsoft**, **Alibaba**, and others run their own research programs.
It’s a question of risk assessment
Real world cryptography development

- Develop systems
- Analyze security
- Implement systems
- Analyze implementation security
- Select best systems and standardize them
- Integrate systems into products & protocols
- Role out secure products

18.06.2019

https://huelsing.net
Who would store all encrypted data traffic? That must be expensive!
Long-lived systems

- Development time easily 10+ years
- Lifetime easily 10+ years

- At least make sure you got a secure update channel!
Hash-based signatures

[Lam79,Mer89]

No new hardness assumptions*

Provably (post-quantum) secure if (post-quantum) secure hash function is used

Basic concept extremely easy

Stateful

* We only assume hash functions do not show non-random behaviour.
Basic construction
Lamport OTS [Lam79]

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

* = n bit
Merkle’s Hash-based Signatures

SIG = (i=2, , , , , )
Winternitz-OTS
Lamport-OTS in MSS

SIG = (i=2, □, □, ○, ○, ○)

Verification:

1. Verify □
2. Verify authenticity of ○

We can do better!
WOTS in MSS

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

Verification:
1. Compute \( X \) from
2. Verify authenticity of \( O \)

Steps 1 + 2 together verify
Function chains

Hash function \( h : \{0,1\}^n \rightarrow \{0,1\}^n \)

Parameter \( w \)

Chain: \( c^i(x) = h \left( c^{i-1}(x) \right) = h \circ h \circ \cdots \circ h(x) \)

\[ c^0(x) = x \]
\[ c^1(x) = h(x) \]

\[ c^{w-1}(x) \]
WOTS

Winternitz parameter $w$ (usually a power of 2), security parameter $n$, message length $m$, hash function $h$

**Key Generation:** Compute $l$, sample $h_k$

$c^0(sk_1) = sk_1$

$c^1(sk_1)$

$c^1(sk_2)$

$c^0(sk_r) = sk_r$

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

$pk_r = c^{w-1}(sk_r)$
WOTS Signature generation

\[ c^0(\text{sk}_1) = \text{sk}_1 \]

\[ \sigma_i = c^{b_1}(\text{sk}_1) \]

\[ c^0(\text{sk}_\ell) = \text{sk}_\ell \]

\[ \sigma_\ell = c^{b_\ell}(\text{sk}_\ell) \]

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

Verifier knows: M, w

Signature:
\( \sigma = (\sigma_1, \ldots, \sigma_\ell) \)
Multi-Tree MSS
Multi-Tree MSS / Hypertree

Uses multiple layers of trees to reduce key generation time

- Key state generation & stateless signing
  (= Building one tree on each layer)
  \[ \Theta(2^h) \rightarrow \Theta\left(d \cdot 2^{h/d}\right) \]

- Worst-case stateful signing times
  \[ \Theta(h/2) \rightarrow \Theta(h/2d) \]

- Increases signature size by \(d-1\) one-time signatures
SPHINCS

Joint work with Daniel J. Bernstein, Daira Hopwood, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O’Hearn
Stateless hash-based signatures

[NY89,Gol87,Gol04]

Goldreich’s approach [Gol04]:
Security parameter $\lambda = 128$
Use binary tree as in Merkle, but...

• ...for security
  • pick index $i$ at random;
  • requires huge tree to avoid index collisions (e.g., height $h = 2\lambda = 256$).

• ...for efficiency:
  • use binary certification tree of OTS key pairs (= Hypertree with $d = h$),
  • all OTS secret keys are generated pseudorandomly.
SPHINCS \([BHH^{+15}]\)

- Select index pseudo-randomly
- Use a few-time signature key-pair on leaves to sign messages
  - Few index collisions allowed
  - Allows to reduce tree height
- Use hypertree: Use \(d << h\).
Security arguments
Requirements

Reductions should lead to
• collision-resilience,
• multi-target attack protection,
• tight security reductions,
and allow for
• easy verification, and
• maintainability.
Multi-target attacks

• WOTS & Lamport need hash function $h$ to be one-way
• Hypertree of total height 60 with WOTS ($w=16$) leads $> 2^{60} \cdot 67 \approx 2^{66}$ images.
• Inverting one of them allows existential forgery (at least massively reduces complexity)
• $q$-query brute-force succeeds with probability $\Theta \left( \frac{q}{2^{n-66}} \right)$ conventional and $\Theta \left( \frac{q^2}{2^{n-66}} \right)$ quantum
• We loose 66 bits of security! (33 bits quantum)
Multi-target attacks: Mitigation

• Mitigation: Separate targets [HRS16]
• Common approach:
  • In addition to hash function description and „input“ take
    • Hash „Address“ (uniqueness in key pair)
    • Hash „key“ used for all hashes of one key pair (uniqueness among key pairs)

18.06.2019
https://huelsing.net
Multi-target attacks: Mitigation

• Mitigation: Separate targets [HRS16]

• Common approach:
  • In addition to hash function description and „input“ take
    • Hash „Address“ (uniqueness in key pair)
    • Hash „key“ used for all hashes of one key pair (uniqueness among key pairs)
New intermediate abstraction: Tweakable Hash Function [SPHINCS+]

• Tweakable Hash Function:

\[ \text{Th}(P, T, M) \rightarrow MD \]

P: Public parameters (one per key pair)
T: Tweak (one per hash call)
M: Message
MD: Message Digest

• Security in two steps:
1. Prove security of SPHINCS(+), XMSS, LMS,..... using tweakable hash functions
2. Prove tweakable hash function security

So what properties do we need?
Single-function multi-target collision resistance for distinct tweaks

• Intuition:
  • Adversary gets black box access to $\text{Th}(P, \cdot, \cdot)$ for random $P$.
  • Adversary can adaptively query with restriction to use each tweak only once.
  • Adversary receives $P$ and has to find second-preimage for one of its previous queries (such that $P$ and $T$ are the same).

• This is what the hashing in [HRS16] already tightly achieves!
  • Generating pseudorandom bitmasks & function keys from $P$ and $T$. 
Decisional second-preimage resistance
(https://ia.cr/2019/492)

• (actually: Single-function multi-target decisional second preimage resistance for distinct tweaks)

• [HRS16] required statistical property: Every message input has to have a sibling (colliding value) under \( \text{Th}(P, \cdot, \cdot) \) for the length-preserving case (\(|M| = |MD|\)).

• Reason: Want reduction using SPR instead of OW.
WOTS reduction from PRE
(assume adversary that always inverts one of the signature query elements)

Signature:
\[ \sigma = (\sigma_1, \ldots, \sigma_l) \]
Decisional second-preimage resistance
(https://ia.cr/2019/492)

• (actually: Single-function multi-target decisional second preimage resistance for distinct tweaks)

• HRS16 required statistical property: Every message input has to have a sibling (colliding value) under $\text{Th}(P, \cdot, \cdot)$ for the length-preserving case ($|M| = |MD|$).

• Reason: Want reduction using SPR instead of OW.
  • WOTS reduction fails if guess was incorrect (Recall, in SPHINCS we have to make $\approx 2^{66}$ guesses)
  • When reducing SPR, we know full chain -> no guesses

• WOTS reduction gives us Inverter with non-negligible success probability
SPR ⇒ PRE for length-preserving functions

• Reduction idea: Return $x' \leftarrow A(H(x))$
  • If $x$ has sibling, reduction loss $\leq 1/2$
    ($x$ is information-theoretically hidden in set of size $\geq 2$)

• Canonical counter example: Identity function

• What about random functions?
  • About $1/e$ of inputs has no second preimage
  • Unbounded adversary might only return preimage if it is a singleton

• Reduction works if $A$ cannot tell singletons from values with siblings...
  • ... better than guessing.

• This is formalized in DSPR.
**Definition 3 (SPexists).** The second-preimage-exists predicate $\text{SPexists}(H)$ for a hash function $H$ is the function $P : \mathcal{X} \to \{0, 1\}$ defined as follows:

$$P(x) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } |H^{-1}(H(x))| \geq 2 \\ 0 & \text{otherwise.} \end{cases}$$

**Definition 4 (SPprob).** The second-preimage-exists probability $\text{SPprob}(H)$ for a hash function $H$ is $\Pr[x \leftarrow_R \mathcal{X} : P(x) = 1]$, where $P = \text{SPexists}(H)$.

**Definition 5 (DSPR).** The advantage of an algorithm $A$ against the decisional second-preimage resistance of a hash function $H$ is

$$\text{Adv}_{H}^{\text{DSPR}}(A) \overset{\text{def}}{=} \max \{0, \Pr[x \leftarrow_R \mathcal{X}; b \leftarrow A(x) : P(x) = b] - p\}$$

where $P = \text{SPexists}(H)$ and $p = \text{SPprob}(H)$. 
DSPR

• Result: **DSPR + SPR ⇒ PRE**

More:

• Best generic attack we found needs a high probability second-preimage finder (probability \( \approx \text{SPprob} \)).
• Quantum query complexity is the same as for SPR
• Almost all length preserving functions have \( \text{SProb} > 0.6 \)
• Strongly compressing random functions have \( \text{SPprob} \) negligibly close to 1
  ⇒ DSPR advantage can only be negligible.
Instantiating the tweakable hash (for SHA2)

SPHINCS\(^+\)-robust (\(\approx\) XMSS)
- \(BM = SHA2(pad(P) || T+1), \quad MD = SHA2(P || T || M \oplus BM)\)
- Standard model proof if BM were random,
- (Q)ROM proof when generating BM as above (modeling those SHA2 invocations as RO)
- Tight proof

SPHINCS\(^+\)- simple (\(\approx\) LMS)
- \(MD = SHA2(P || T || M)\)
- QROM proof assuming SHA2 is QRO
- Tight proofs conjectured (LMS has tight proof)
Conclusion

• Tweakable hash functions provide an abstraction to split proofs in two parts and simplify the analysis of new constructions
• SPHINCS$^+$-simple is factor 3 faster
• SPHINCS$^+$-simple makes somewhat stronger assumptions about the security properties of the used hash function
Thank you!

Questions?

For references, literature & longer lectures see https://huelsing.net
SPHINCS+

Joint work with Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe
SPHINCS\(^+\) (our NIST submission)

- Strengthened security gives smaller signatures
- Collision- and multi-target attack resilient (XMSS tweakable hash)
- Fixed length signatures
- Small keys, medium size signatures (lv 3: 17kB)
- Sizes can be much smaller if q\(_{\text{sign}}\) gets reduced
- The **conservative choice**
FORS (Forest of random subsets)

- Parameters $t$, $a = \log t$, $k$ such that $ka = m$
Verifiable index selection
(and optionally non-deterministic randomness)

• SPHINCS:

\[(\text{idx}||\mathbf{R}) = PRF(\mathbf{SK}. \text{prf}, M)\]
\[\text{md} = H_{\text{msg}}(\mathbf{R}, \mathbf{PK}, M)\]

• SPHINCS⁺:

\[\mathbf{R} = PRF(\mathbf{SK}. \text{prf}, \text{OptRand}, M)\]
\[(\text{md}||\text{idx}) = H_{\text{msg}}(\mathbf{R}, \mathbf{PK}, M)\]
Verifiable index selection

Improves FORS security

• SPHINCS: Attacks can target „weakest“ HORST key pair
• SPHINCS\(^+\): Every hash query also selects FORS key pair
  • Leads to notion of interleaved target subset resilience

https://sphincs.org
Instantiations
(after second round tweaks)

• SPHINCS\(^+\)-SHAKE256-robust
• SPHINCS\(^+\)-SHAKE256-simple NEW!
• SPHINCS\(^+\)-SHA-256-robust
• SPHINCS\(^+\)-SHA-256-simple NEW!
• SPHINCS\(^+\)-Haraka-robust
• SPHINCS\(^+\)-Haraka-simple NEW!
# Instantiations (small vs fast)

<table>
<thead>
<tr>
<th></th>
<th>$n$</th>
<th>$h$</th>
<th>$d$</th>
<th>$\log(t)$</th>
<th>$k$</th>
<th>$w$</th>
<th>bitsec</th>
<th>sec level</th>
<th>sig bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPHINCS$^+$-128s</td>
<td>16</td>
<td>64</td>
<td>8</td>
<td>15</td>
<td>10</td>
<td>16</td>
<td>133</td>
<td>1</td>
<td>8080</td>
</tr>
<tr>
<td>SPHINCS$^+$-128f</td>
<td>16</td>
<td>60</td>
<td>20</td>
<td>9</td>
<td>30</td>
<td>16</td>
<td>128</td>
<td>1</td>
<td>16976</td>
</tr>
<tr>
<td>SPHINCS$^+$-192s</td>
<td>24</td>
<td>64</td>
<td>8</td>
<td>16</td>
<td>14</td>
<td>16</td>
<td>196</td>
<td>3</td>
<td>17064</td>
</tr>
<tr>
<td>SPHINCS$^+$-192f</td>
<td>24</td>
<td>66</td>
<td>22</td>
<td>8</td>
<td>33</td>
<td>16</td>
<td>194</td>
<td>3</td>
<td>35664</td>
</tr>
<tr>
<td>SPHINCS$^+$-256s</td>
<td>32</td>
<td>64</td>
<td>8</td>
<td>14</td>
<td>22</td>
<td>16</td>
<td>255</td>
<td>5</td>
<td>29792</td>
</tr>
<tr>
<td>SPHINCS$^+$-256f</td>
<td>32</td>
<td>68</td>
<td>17</td>
<td>10</td>
<td>30</td>
<td>16</td>
<td>254</td>
<td>5</td>
<td>49216</td>
</tr>
</tbody>
</table>
Comparison to SPHINCS-128 at same security level

<table>
<thead>
<tr>
<th></th>
<th>Signing median cycles</th>
<th>Verifying median cycles</th>
<th>Signature bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPHINCS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n = 24, h = 55, d = 11, b = 8, ) ( k = 30, w = 16 )</td>
<td>67 017 940</td>
<td>1 911 684</td>
<td>21 288</td>
</tr>
<tr>
<td><strong>SPHINCS</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n = 24, h = 51, d = 17, b = 9, ) ( k = 30, w = 16 )</td>
<td>40 117 282</td>
<td>2 724 094</td>
<td>29 256</td>
</tr>
<tr>
<td><strong>SPHINCS-128</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( n = 32, h = 60, d = 12, t = 2^{16}, ) ( k = 32, w = 16 )</td>
<td>51 636 372</td>
<td>1 451 004</td>
<td>41 000</td>
</tr>
</tbody>
</table>
Hash-based Signatures in NIST „Competition“

• SPHINCS$^+$
  • FORS as few-time signature
  • XMSS-T tweakable hash

• Gravity-SPHINCS (R.I.P.)
  • PORS as few-time signature
  • Requires collision-resistance -> no tweakable hash

• (PICNIC)