The SPHINCS$^+$ Signature Framework

Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, Peter Schwabe
The SPHINCS + team

Jean-Philippe Aumasson, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe
Hash-based signatures

Boring crypto:

• Dates back to beginning of public-key cryptography
• No fancy new mathematical assumption: Only requires a secure hash function („minimal security assumptions“)
• Stateful schemes already in RFCs (LMS & XMSS)
OTS

• 1-bit Lamport:

<table>
<thead>
<tr>
<th>SK</th>
<th>PK</th>
<th>Sig (M=0)</th>
<th>Sig (M=1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_0$</td>
<td>$Y_0 = H(X_0)$</td>
<td>$X_0$</td>
<td>$X_1$</td>
</tr>
<tr>
<td>$X_1$</td>
<td>$Y_1 = H(X_1)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• N-bit Lamport: Use N pairs of secret values.

https://sphincs.org/
Merkle Signatures (from OTS to MTS)

- Take $2^h$ OTS key pairs
- Use the $H(PK_{OTS})$ as leaves of a binary hash tree
- Signature = OTS.sig + authentication path
- Can be used for $2^h$ signatures
- Requires to remember used OTS key pairs / indices!

![Authentication tree with n = 8](https://sphincs.org/)

14.11.2019
https://sphincs.org/
Hypertree: A tree of trees

• Single tree:
  Root & authpath computation costs $\Theta(2^h)$ time (w/o state)

• Hypertree = Certification tree (think of a PKI):
  Root & authpath computation
  costs $\Theta(2^h) \rightarrow \Theta(2^{h/d})$ (w/o state)
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 statelessness
  • 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\textsuperscript{+}15]

- Select index pseudorandomly
- Use a few-time signature key-pair on leaves to sign messages
  - Few index collisions allowed
  - Allows to reduce tree height
- Use hypertree: Use $d < h$. (SPHINCS-256: $h=60$, $d=12$)
This work
Our contribution

• Introduce SPHINCS\(^+\)
  (currently in 2\(^{nd}\) round of NIST PQC competition)

• Introduce tweakable hash functions
  (Allow for modular proof of security)

• Comparison with SPHINCS, Gravity-SPHINCS, and Picnic
SPHINCS+
From SPHINCS to SPHINCS$^+$

• Allow for $2^{64}$ instead of $2^{50}$ signatures per key pair
• Add multi-target attack mitigation (Tweakable hash functions)
• New few-time signature scheme FORS
• Verifiable index selection
• Optional non-deterministic signatures (better SCA protection)
FTS (HORST → FORS)

• Basic idea of FTS:
  • $M \rightarrow I_M = (i_0, i_1, \ldots, i_k)$
  • Signature consists of $(SK_{i_0}, SK_{i_1}, \ldots, SK_{i_k})$
  • Generic attack: Find $M$ such that all $SK_j$ selected by $I_M$ have been part of previous signatures.

• HORST: Draw all $SK_j$ from same “pool” of secret values
  • Issue: Possible to have $i_0 = i_1 = \cdots = i_k = j$ so signature only requires knowledge of single value $SK_j$

• FORS: Use $k$ independent „pools“ of secret values
  • Generic attacks get harder → can choose smaller parameters → smaller signatures
Tweakable hash functions

A tool for modular proofs for hash-based signatures

14.11.2019
https://sphincs.org/
Hashing for hash-based signatures
(Change driven by goal to minimize security assumptions)

Gravity-SPHINCS

SPHINCS

SPHINCS+ robust

SPHINCS+

simple

https://sphincs.org/
Tweakable hash function

\[ \text{Th}(P, T, X) \rightarrow Y \]

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

All of the previous modes are instantiations!
Required security properties
(see paper for formal definitions)

• (PQ-)SM-TCR for distinct tweaks:
  - Single-function (= random but fixed P),
  - Multi-target (= adversary A can adaptively define targets \((T, M)\), using oracle initialized with P)
  - Target-Collision Resistance (= when targets defined, A receives P, has to find collision \((T, M^*)\) for one of the targets)
  - For distinct tweaks (= only one target per tweak)

• (PQ-)SM-DSPR for distinct tweaks:
  - ...
  - Decisional Second-Preimage Resistance (= decide for one target if it has a second preimage)
  - ...
In paper

• Tweakable hash constructions that achieve PQ-SM-TCR & PQ-SM-DSPR
  • Construction 1: Standard model proof but massive public parameters
  • Construction 2: Construction 1 with compressed public parameters (compression needs QROM, approx. XMSS-T construction)
  • Construction 3: All QROM proof (simplified LMS construction)
Theorem 17. For parameters n, w, h, d, m, t, k as described above, SPHINCS+ is PQ-EU-CMA secure if

- \( \textbf{Th} \) (and thereby also \( \textbf{F} \) and \( \textbf{H} \)) is post-quantum single-function multi-target-collision resistant for distinct tweaks (with tweak advice),
- \( \textbf{F} \) is post-quantum single-function multi-target decisional second-preimage resistant for distinct tweaks (with tweak advice),
- \( \textbf{PRF} \) and \( \textbf{PRF}_{\text{msg}} \) are post-quantum pseudorandom function families, and
- \( \textbf{H}_{\text{msg}} \) is post-quantum interleaved target subset resilient.
Comparison
Table 2: Performance comparison of different symmetric-crypto-based signature schemes on the Intel Haswell microarchitecture. All software is optimized using architecture-specific optimizations such as AESNI or AVX2 instructions.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Cycles</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>keypair</td>
<td>sign</td>
</tr>
<tr>
<td><strong>Comparison to SPHINCS-256</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPHINCS-256 [8]</td>
<td>2 868 464&lt;sup&gt;a&lt;/sup&gt;</td>
<td>50 462 856&lt;sup&gt;a&lt;/sup&gt;</td>
</tr>
<tr>
<td>SPHINCS&lt;sup&gt;+&lt;/sup&gt; (Haraka, robust)</td>
<td>1 254 968&lt;sup&gt;b&lt;/sup&gt;</td>
<td>29 015 002&lt;sup&gt;b&lt;/sup&gt;</td>
</tr>
<tr>
<td>(n = 192, h = 51, d = 17, b = 7, k = 45, w = 16)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Comparison to Gravity-SPHINCS**

| Gravity-SPHINCS [5] (parameter-set L) | 30 729 044 392<sup>a</sup> | 32 564 796<sup>a</sup> | 625 752<sup>a</sup> | max: 35 168 | avg: 32 564 | 1 048 608 |
| SPHINCS<sup>+</sup> (Haraka, robust) (n = 192, h = 66, d = 22, b = 8, k = 33, w = 16) | 1 257 826<sup>b</sup> | 38 840 268<sup>b</sup> | 3 467 192<sup>b</sup> | 35 664 | 48  | 96  |
| SPHINCS<sup>+</sup> (Haraka, simple) (n = 192, h = 64, d = 16, b = 7, k = 49, w = 16) | 1 892 462<sup>b</sup> | 35 029 380<sup>b</sup> | 1 460 204<sup>b</sup> | 30 552 | 48  | 96  |

**Comparison to Picnic**

| Picnic2-L5-FS [16] | 18 244<sup>c</sup> | 904 189 188<sup>c</sup> | 268 485 212<sup>c</sup> | max: 54 732 | avg: 46 282 | 65  | 97  |
| SPHINCS<sup>+</sup> (SHA-256, simple) (n = 256, h = 63, d = 9, b = 12, k = 29, w = 16) | 43 317 320<sup>b</sup> | 527 413 100<sup>b</sup> | 5 463 884<sup>b</sup> | 33 408 | 64  | 128 |

<sup>a</sup> As reported by SUPERCOP [10] from 3.5GHz Intel Xeon E3-1275 V3 (Haswell)
<sup>b</sup> Median of 100 runs on 3.5GHz Intel Xeon E3-1275 V3 (Haswell), compiled with gcc-5.4 -O3 -march=native -fomit-frame-pointer -flto
<sup>c</sup> As reported by SUPERCOP [10] from 3.1GHz Intel Xeon E3-1220 V3 (Haswell)
Conclusion

• SPHINCS\(^+\) beats performance of other symmetric crypto based signatures for comparable parameters.

• Tweakable hash functions allow to modularize proofs (and implementation) of hash-based signatures.

• SPHINCS\(^+\) is \textit{the} most conservative signature scheme in the NIST PQC competition.
“If you’re signing something for the long-term future, and 40KB sigs is not a problem, use (stateless) hash-based sigs e.g. SPHINCS”

Vadim Lyubashevsky, 2017