SPHINCS$^+$

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 standardization
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 \([\text{BHH}^+15]\)

- 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+ in 1st Round

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$^+$ vs 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

https://sphincs.org/
23.08.2019
SPHINCS$^+$ in 2nd Round

Joint work with Jean-Philippe Aumasson, 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
2nd Round changes

• One new team member: 1/2 Gravity-SPHINCS (J.-P. Aumasson)
• “Simple”-instantiations (of tweakable hash functions)
• Performance optimization for SHA-256 instantiations
• Fixed tight security reduction

23.08.2019
Instantiations

- SPHINCS⁺-SHAKE256-robust
- SPHINCS⁺-SHAKE256-simple NEW!
- SPHINCS⁺-SHA-256-robust
- SPHINCS⁺-SHA-256-simple NEW!
- SPHINCS⁺-Haraka-robust
- SPHINCS⁺-Haraka-simple NEW!

Simple instantiations inspired by LMS: All security reasoning in (Q)ROM
## Instantiations (cycles for SHA2)

<table>
<thead>
<tr>
<th>Scheme</th>
<th>sec</th>
<th>keypair</th>
<th>Cycles</th>
<th>verify</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPHINCS⁺-SHA-256-128s-simple</td>
<td>L1</td>
<td>49 078 104</td>
<td>835 272 076</td>
<td>2 348 916</td>
<td>8 080 32 64</td>
</tr>
<tr>
<td>SPHINCS⁺-SHA-256-128s-robust</td>
<td>L1</td>
<td>94 988 100</td>
<td>1 624 566 118</td>
<td>4 700 588</td>
<td>8 080 32 64</td>
</tr>
<tr>
<td>SPHINCS⁺-SHA-256-128f-simple</td>
<td>L1</td>
<td>1 602 368</td>
<td>51 805 308</td>
<td>5 676 578</td>
<td>16 976 32 64</td>
</tr>
<tr>
<td>SPHINCS⁺-SHA-256-128f-robust</td>
<td>L1</td>
<td>2 978 018</td>
<td>96 974 576</td>
<td>11 401 188</td>
<td>16 976 32 64</td>
</tr>
<tr>
<td>SPHINCS⁺-SHA-256-192s-simple</td>
<td>L3</td>
<td>69 860 954</td>
<td>1 737 629 602</td>
<td>3 662 790</td>
<td>17 064 48 96</td>
</tr>
<tr>
<td>SPHINCS⁺-SHA-256-192s-robust</td>
<td>L3</td>
<td>134 664 612</td>
<td>3 024 929 742</td>
<td>7 784 118</td>
<td>17 064 48 96</td>
</tr>
<tr>
<td>SPHINCS⁺-SHA-256-192f-simple</td>
<td>L3</td>
<td>2 116 010</td>
<td>66 380 214</td>
<td>9 611 814</td>
<td>35 664 48 96</td>
</tr>
<tr>
<td>SPHINCS⁺-SHA-256-192f-robust</td>
<td>L3</td>
<td>4 390 738</td>
<td>133 192 018</td>
<td>19 219 918</td>
<td>35 664 48 96</td>
</tr>
<tr>
<td>SPHINCS⁺-SHA-256-256s-simple</td>
<td>L5</td>
<td>85 946 882</td>
<td>1 121 074 298</td>
<td>4 903 926</td>
<td>29 792 64 128</td>
</tr>
<tr>
<td>SPHINCS⁺-SHA-256-256s-robust</td>
<td>L5</td>
<td>350 260 762</td>
<td>4 064 645 574</td>
<td>13 790 402</td>
<td>29 792 64 128</td>
</tr>
<tr>
<td>SPHINCS⁺-SHA-256-256f-simple</td>
<td>L5</td>
<td>5 298 662</td>
<td>133 374 038</td>
<td>9 408 596</td>
<td>49 216 64 128</td>
</tr>
<tr>
<td>SPHINCS⁺-SHA-256-256f-robust</td>
<td>L5</td>
<td>21 672 826</td>
<td>495 051 104</td>
<td>26 825 462</td>
<td>49 216 64 128</td>
</tr>
</tbody>
</table>
SHA-256 optimization

\[ F(PK.\ seed, ADRS, M_1) = \text{SHA-256}(PK.\ seed \ || \ \text{toByte}(0, 64 - n) \ || \ ADRS^c \ || \ M_1^\oplus) \]

Ensure that key dependent input fills first block (precompute & reuse state)

Compress ADRS to fit padding in second block

Only 1 compression function call per F call!
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 keypair</th>
<th>Cycles sign</th>
<th>Cycles verify</th>
<th>Bytes sig</th>
<th>Bytes pk</th>
<th>Bytes sk</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Comparison to SPHINCS-256</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPHINCS-256 [8]</td>
<td>2 868 464⁵</td>
<td>50 462 856⁵</td>
<td>1 672 652⁵</td>
<td>41 000</td>
<td>1 056</td>
<td>1 088</td>
</tr>
<tr>
<td>(Haraka, robust) (n = 192, h = 51, d = 17, b = 7, k = 45, w = 16)</td>
<td>1 254 968⁵</td>
<td>29 015 002⁵</td>
<td>2 739 770⁵</td>
<td>30 696</td>
<td>48</td>
<td>96</td>
</tr>
<tr>
<td><strong>Comparison to Gravity-SPHINCS</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Haraka, robust) (n = 192, h = 66, d = 22, b = 8, k = 33, w = 16)</td>
<td>1 257 826⁵</td>
<td>38 840 268⁵</td>
<td>3 467 192⁵</td>
<td>35 664</td>
<td>48</td>
<td>96</td>
</tr>
<tr>
<td>SPHINCS+ (Haraka, simple) (n = 192, h = 64, d = 16, b = 7, k = 49, w = 16)</td>
<td>1 892 462⁵</td>
<td>35 029 380⁵</td>
<td>1 460 204⁵</td>
<td>30 552</td>
<td>48</td>
<td>96</td>
</tr>
<tr>
<td><strong>Comparison to Picnic</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Picnic2-L5-FS [16]</td>
<td>18 244⁶</td>
<td>904 189 188⁶</td>
<td>268 485 212⁶</td>
<td>max: 54 732 avg: 46 282</td>
<td></td>
<td>65</td>
</tr>
<tr>
<td>SPHINCS+ (SHA-256, simple) (n = 256, h = 63, d = 9, b = 12, k = 29, w = 16)</td>
<td>43 317 320⁶</td>
<td>527 413 100⁶</td>
<td>5 463 884⁶</td>
<td>33 408</td>
<td>64</td>
<td>128</td>
</tr>
</tbody>
</table>

⁵ As reported by SUPERCOP [10] from 3.5GHz Intel Xeon E3-1275 V3 (Haswell)
⁶ 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

Tight security reduction

“2. This claim is incorrect, because the theorem apparently does not apply to the proposed instantiations: it requires the component function F to have a structural property that it almost certainly does not have. (Indeed, it seems hard to find *any* suitable instantiation having the property.)”

Chris Peikert, May 24, 2018
Ugly fixes:

a) Rely on loose security reduction

b) Build artificial hash with property

\[ F = \text{SHA2−256}(K, \text{SHA2−256}(K, M)_{0...248}) \]

Cost of factor 2 speed penalty and slightly decreased security
More elegant fix

Replace statistical property by new computational assumption:

Decisional second-preimage resistance (DSPR) [BH19]

Intuition:
Given domain element it is hard to decide if second-preimage exists (with probability better than guessing)
Conclusion

• SPHINCS$^+$ beats performance of other symmetric crypto based signatures for comparable parameters.
• Possible synergies with standardizing stateful hash-based signatures
• New *simple* instantiations integrate well with an LMS-like stateful scheme.
• Re-established tight security proof.
• *The* most conservative submission in the competition.
Thank you!
Questions?
Structural property

• “Every n bit string has at least one colliding n bit string under $F_K$“

Use in proof:
We can turn preimage finder $A$ for $F_K$ into second-preimage finder $B$ for $F_K$.
• $B(x)$: On input $x$ return $A(F_K(x))$
• If property holds,
\[
\Pr[A(F_K(x)) \neq x] \geq \frac{1}{2}
\]