A note on SPHINCS+ parameters

Stefan Kölbl, Jade Philipoom
Fifth PQC Standardization Conference
April 11th, 2024
Background  NIST calls requires supporting $q = 2^{64}$ without security degradation.

In practice  Use cases where SPHINCS+ fits well do not need that many signatures.

Goal  What do we get if we target a lower $q$?
SPHINCS+ Parameter Space

- $2^{10}$ Signatures
- $2^{20}$ Signatures
- $2^{10}$ Signatures ($w = 256$)
- $2^{20}$ Signatures ($w = 256$)
- $2^{10}$ Signatures (100-bit sec for $2^{20}$)
- $2^{20}$ Signatures (100-bit sec for $2^{30}$)
- $2^{30}$ Signatures
- $2^{40}$ Signatures
- $2^{50}$ Signatures
- $2^{64}$ Signatures

Round 3 F parameter set

Round 3 S parameter set
SPHINCS+ Parameter Space

- Too many choices, trade-offs which may fit specific use cases better.
- Our proposal: Focus on use cases where SPHINCS+ will likely find usage:
  - Firmware signing.
  - Limit on $q = 2^{20}$.
- We don’t see much value in having fast signing, targeting low $q$:
  - SPHINCS+ signing is slow, or huge signatures.
  - Low q and fast signing $\rightarrow$ Higher risk of misuse.
Our proposal

<table>
<thead>
<tr>
<th></th>
<th>n</th>
<th>h</th>
<th>d</th>
<th>b</th>
<th>k</th>
<th>w</th>
<th>bitsec</th>
<th>sig bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPHINC$^+$-128s</td>
<td>16</td>
<td>63</td>
<td>7</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>128</td>
<td>7856</td>
</tr>
<tr>
<td>SPHINC$^+$-128s-q20</td>
<td>16</td>
<td>18</td>
<td>1</td>
<td>24</td>
<td>6</td>
<td>16</td>
<td>128</td>
<td>3264</td>
</tr>
<tr>
<td>SPHINC$^+$-192s</td>
<td>24</td>
<td>63</td>
<td>7</td>
<td>14</td>
<td>17</td>
<td>16</td>
<td>192</td>
<td>16224</td>
</tr>
<tr>
<td>SPHINC$^+$-192s-q20</td>
<td>24</td>
<td>20</td>
<td>1</td>
<td>21</td>
<td>10</td>
<td>16</td>
<td>192</td>
<td>7008</td>
</tr>
<tr>
<td>SPHINC$^+$-256s</td>
<td>32</td>
<td>64</td>
<td>8</td>
<td>14</td>
<td>22</td>
<td>16</td>
<td>255</td>
<td>29792</td>
</tr>
<tr>
<td>SPHINC$^+$-256s-q20</td>
<td>32</td>
<td>19</td>
<td>1</td>
<td>21</td>
<td>14</td>
<td>16</td>
<td>256</td>
<td>12640</td>
</tr>
</tbody>
</table>

- Target $q=2^{20}$
- >50% reduction signature size
- Very fast verification, very slow signing (~1 min)
Benchmarks

<table>
<thead>
<tr>
<th>Parameters</th>
<th>signature size (bytes)</th>
<th>verification speed (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPHINCS(^+)-SHAKE-128s</td>
<td>7 856</td>
<td>1 298 047</td>
</tr>
<tr>
<td>SPHINCS(^+)-SHAKE-128s-q20</td>
<td>3 264</td>
<td>277 852</td>
</tr>
<tr>
<td>SPHINCS(^+)-SHAKE-192s</td>
<td>16 224</td>
<td>2 089 772</td>
</tr>
<tr>
<td>SPHINCS(^+)-SHAKE-192s-q20</td>
<td>7 008</td>
<td>462 991</td>
</tr>
<tr>
<td>SPHINCS(^+)-SHAKE-256s</td>
<td>29 792</td>
<td>3 390 932</td>
</tr>
<tr>
<td>SPHINCS(^+)-SHAKE-256s-q20</td>
<td>12 640</td>
<td>695 937</td>
</tr>
</tbody>
</table>

- Benchmarks on OpenTitan (open source silicon root of trust)
- Verification speed competitive with RSA/ECDSA
Risks

Main risks

- Tracking signature count = stateful?
- Low usage limits have been problematic in the past (e.g. AES-GCM).

Mitigations

1) Security degrades very slowly.
2) Backing up keys is much simpler (no synchronization on import/export).
3) Concurrent use of keys is much simpler (no synchronization).
Risks
Risks

~2 years on 1 CPU
Risks

~2 years on 1 CPU

~64 years on 1 CPU

n=18, d=1, b=24, k=6
Conclusion

We think such parameter sets will find use in practice:

- Significantly more efficient.
- Provide a good alternative to stateful HBS.

Open questions

- Are there other use cases which would benefit from this?
- Should there be more parameter sets?
Thank you

https://eprint.iacr.org/2022/1725