Google | Privacy, Safety & Security

A note on SPHINCS+ parameters

Stefan Kölbl, Jade Philipoom

Fifth PQC Standardization Conference

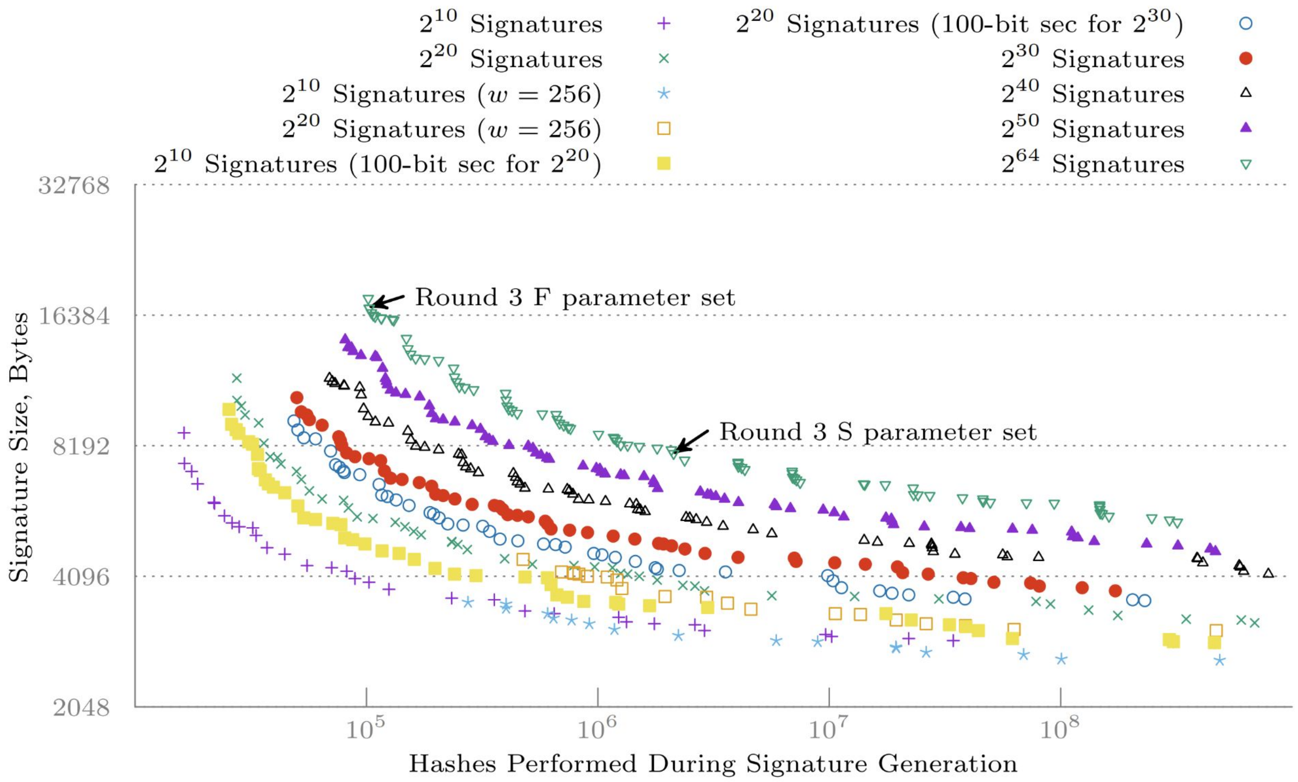
April 11th, 2024

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



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 -> Higher risk of misuse.

Our proposal

	n	h	d	b	k	w	bitsec	sig bytes	
SPHINCS$^+$-128s	16	63	7	12	14	16	128	7 856	-58%
SPHINCS$^+$-128s-q20	16	18	1	24	6	16	128	3 264	
SPHINCS$^+$-192s	24	63	7	14	17	16	192	16 224	-57%
SPHINCS$^+$-192s-q20	24	20	1	21	10	16	192	7 008	
SPHINCS$^+$-256s	32	64	8	14	22	16	255	29 792	-57%
SPHINCS$^+$-256s-q20	32	19	1	21	14	16	256	12 640	

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

Benchmarks

Parameters	signature size (bytes)	verification speed (cycles)	
SPHINCS$^+$-SHAKE-128s	7 856	1 298 047	-79%
SPHINCS$^+$-SHAKE-128s-q20	3 264	277 852	
SPHINCS$^+$-SHAKE-192s	16 224	2 089 772	-78%
SPHINCS$^+$-SHAKE-192s-q20	7 008	462 991	
SPHINCS$^+$-SHAKE-256s	29 792	3 390 932	-79%
SPHINCS$^+$-SHAKE-256s-q20	12 640	695 937	

- Benchmarks on OpenTitan (open source silicon root of trust)
- Verification speed competitive with RSA/ECDSA
- Full details: https://github.com/jadephilipoom/opentitan/tree/spx-benchmark/spx-benchmark

Risks

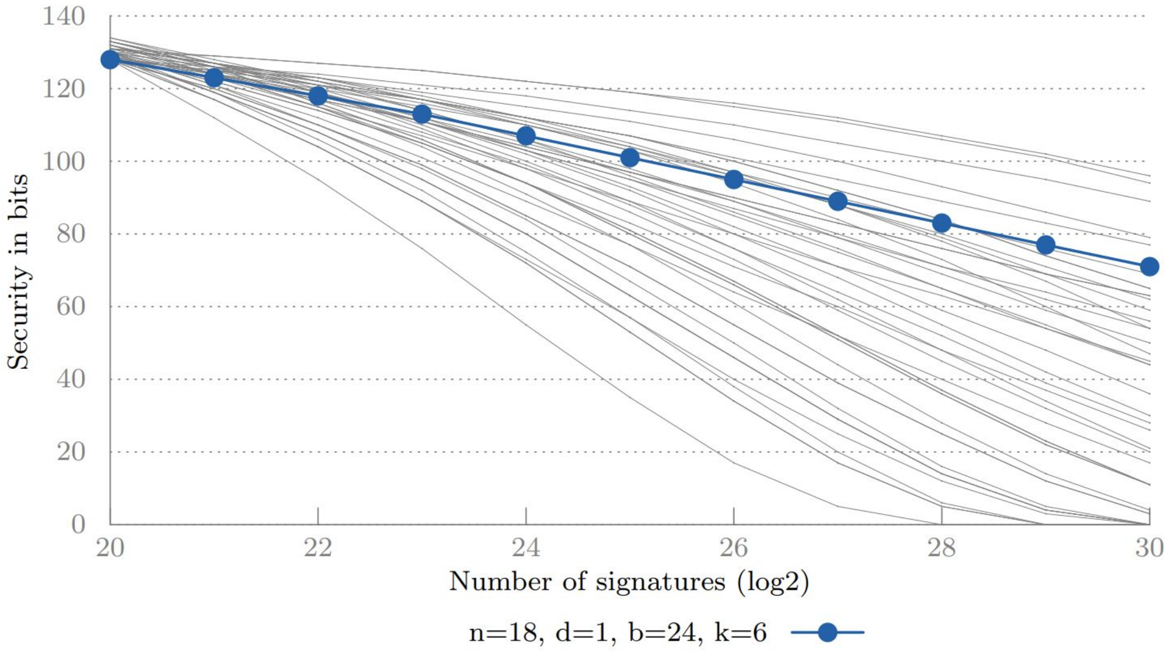
Main risks

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

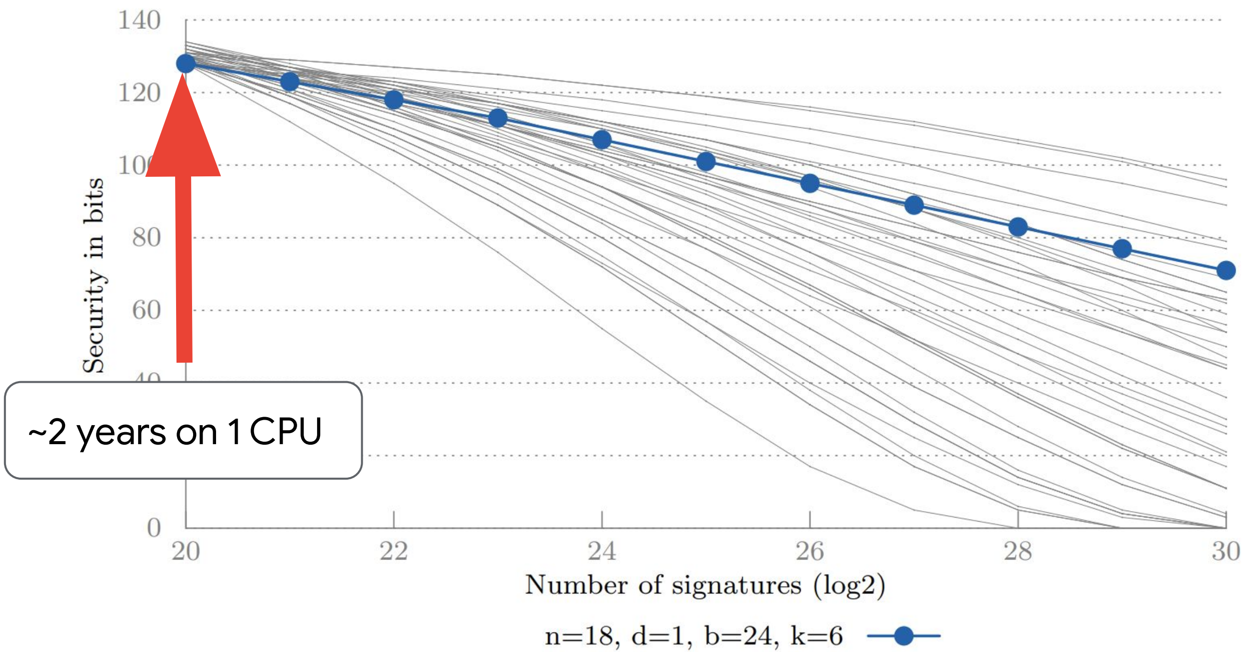
Mitigations

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

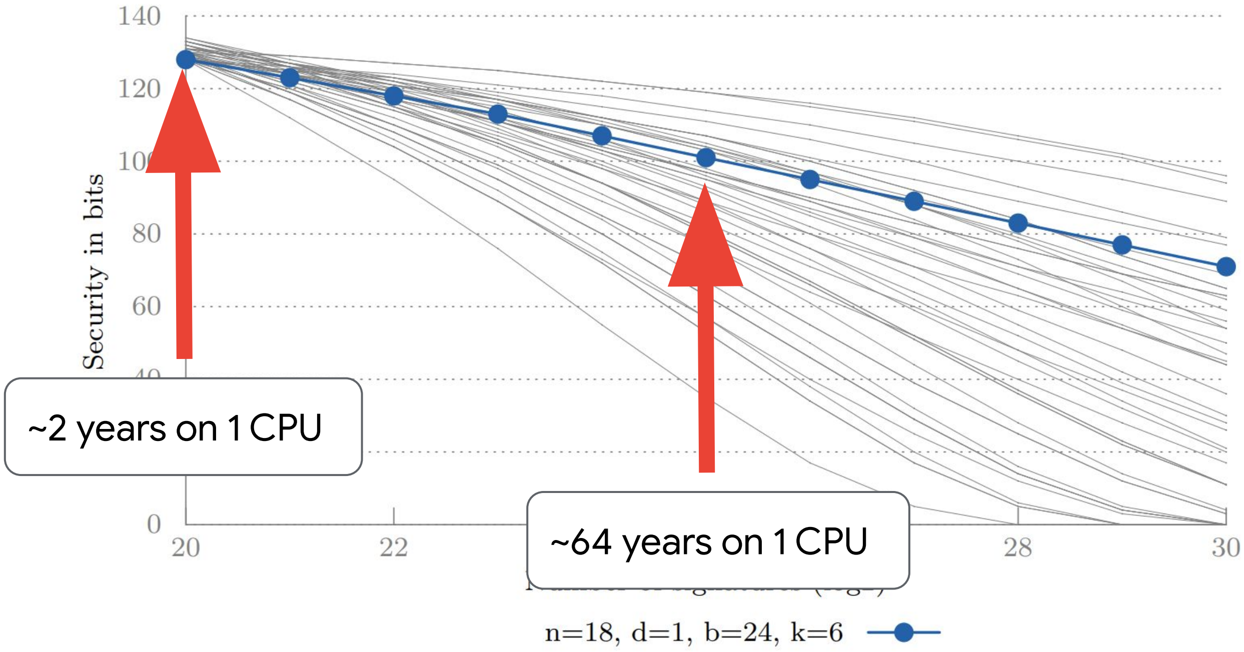

140
120
100
80
60
40
20
0
Security in bits
20
22
24
26
28
30
Number of signatures (log2)
n=18, d=1, b=24, k=6

Risks



140
120
100
80
60
40
0
Security in bits
20
22
24
26
28
30
Number of signatures (log2)
n=18, d=1, b=24, k=6
~2 years on 1 CPU

Risks



140
120
100
80
60
40
0
Security in bits
20
22
28
30
~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