

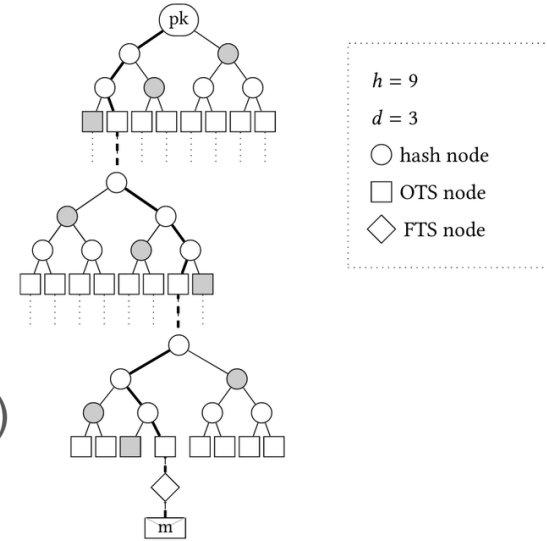
SPHINCS+C

Compressing SPHINCS+ With (Almost) No Cost

Andreas Hülsing, Mikhail Kudinov, **Eyal Ronen**, Eylon Yogev

SPHINCS+

- A **hash-based** signature scheme
- One of the selected **NIST PQ** digital schemes
 - One of the most secure and robust schemes
- Has a “small” and “fast” variant (for each security level)
- Actually allows a wide range of tradeoffs between:
 - Sig-Size, Sig-Gen-Time, Sig-Ver-Time
 - In general: faster Sig-Gen-Time and Sig-Ver-Time -> larger Sig-Size



SPHINCS+C

- Same structure with new more efficient primitives: WOTS+C, FORS+C
- Minor code changes compared to SPHINCS+
- Allows a new realm of better tradeoffs than SPHINCS+:
 - E.g., smaller Sig-Size with same Sig-Gen-Time

| Security Level | Small Signature Size | | Fast Signature Size | |
|----------------|----------------------|--------------|---------------------|--------------|
| | SPHINCS+ | SPHINCS+C | SPHINCS+ | SPHINCS+C |
| 128-bit | 7856 | 6304 (−20%) | 17088 | 14904 (−13%) |
| 192-bit | 16224 | 13776 (−16%) | 35664 | 33016 (−8%) |
| 256-bit | 29792 | 26096 (−13%) | 49856 | 46884 (−6%) |

Intuition for WOTS+C and FORS+C

- Hash and Sign

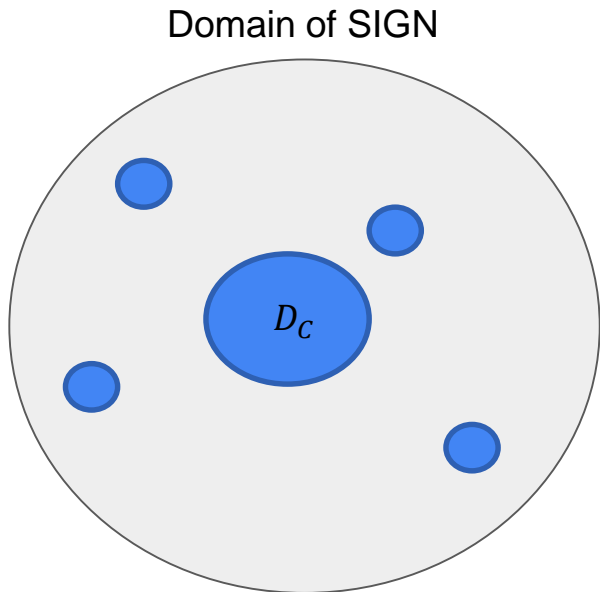
- $d = \text{HASH}(r||m)$, $m \in \{0,1\}^*$, $d \in \{0,1\}^n$, $r \leftarrow \$$
- $\sigma = \text{SIGN}(d)$

- SIGN accepts any value of d

- SIGN may be “compressed” (size, run time) for some sub-domains

- Basic idea

- Find “good” sub-domain D_C and compress SIGN/VER for it
- For signing, we add an incrementing counter to the hash
- Search for cnt value such that $d_C = \text{HASH}(r||cnt||m) \in D_C$
- $\sigma_c = \text{SIGN}_C(d_c)||cnt$
- Verifier checks that $\text{HASH}(r||cnt||m) \in D_C$ and $\text{VER}_C(d, \sigma) = 1$



Intuition for WOTS+C and FORS+C

- Wait a minute, the Sig-Gen running time is not constant!
 - Yes, but this is actually OK
- Is this secure against **side-channel attacks**?
 - Yes, if original SPHINCS+ is “Constant time” crypto then so is SPHINCS+C
 - “Constant time” means independence of running time and secret inputs
 - Our run time variance only depends on the message and public values
 - Variance is independent from any secret values and doesn't leak any information about them

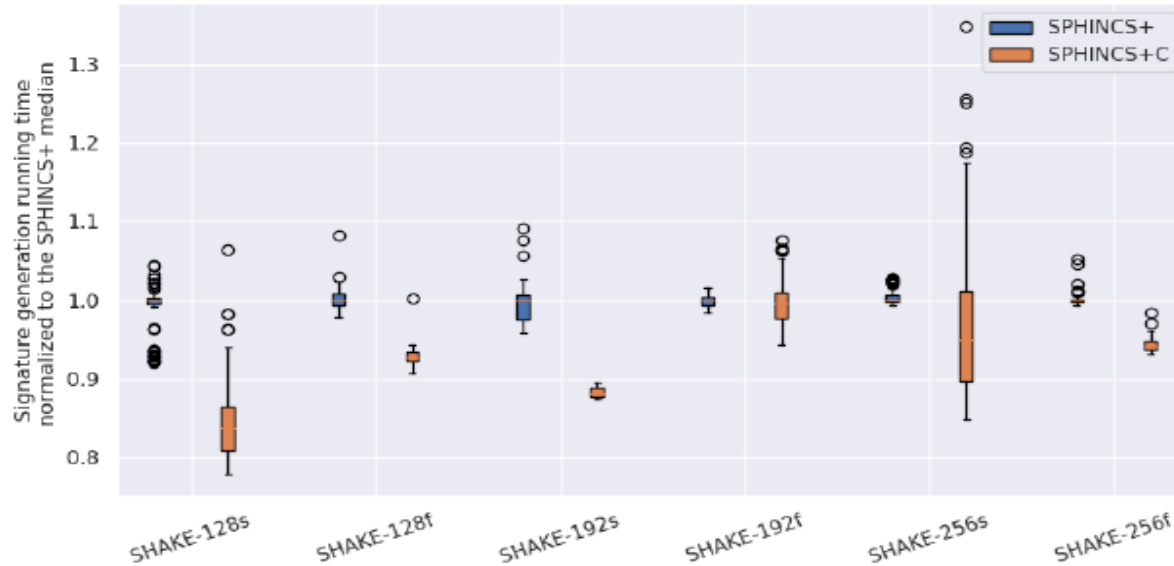
Intuition for WOTS+C and FORS+C

- Won't some signatures take a really long time?
 - We bound the probability p for s signature will run more than $f(p)$ time the expected time
 - E.g., $f(2^{-16}) < 3$, $f(2^{-32}) < 5$ and $f(2^{-64}) < 9$
 - Can optimize for parameter sets with lower variance.
 - E.g., for SPHINCS+C-256f $f(2^{-64}) < 1.2$

| | expected | | $f(p)$ for probability | | | | | | | |
|----------------|------------|------------|------------------------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| | hash calls | $\log(t')$ | 2^{-8} | 2^{-16} | 2^{-24} | 2^{-32} | 2^{-40} | 2^{-48} | 2^{-56} | 2^{-64} |
| SPHINCS+C-128s | $2^{20.9}$ | 18 | 1.6 | 2.3 | 3.1 | 3.8 | 4.5 | 5.3 | 6.0 | 6.7 |
| SPHINCS+C-128f | $2^{16.7}$ | 8 | 1.0 | 1.0 | 1.1 | 1.1 | 1.1 | 1.1 | 1.1 | 1.1 |
| SPHINCS+C-192s | $2^{21.7}$ | 12 | 1.0 | 1.0 | 1.1 | 1.1 | 1.1 | 1.1 | 1.1 | 1.1 |
| SPHINCS+C-192f | $2^{17.4}$ | 13 | 1.2 | 1.5 | 1.8 | 2.0 | 2.3 | 2.6 | 2.9 | 3.1 |
| SPHINCS+C-256s | $2^{21.5}$ | 19 | 1.8 | 2.8 | 3.7 | 4.7 | 5.7 | 6.6 | 7.6 | 8.6 |
| SPHINCS+C-256f | $2^{18.4}$ | 10 | 1.0 | 1.0 | 1.1 | 1.1 | 1.1 | 1.1 | 1.1 | 1.2 |

Intuition for WOTS+C and FORS+C

- Won't some signatures take a really long time?
 - We also run experiments to compare variability in SIG-GEN time with SPHINCS+



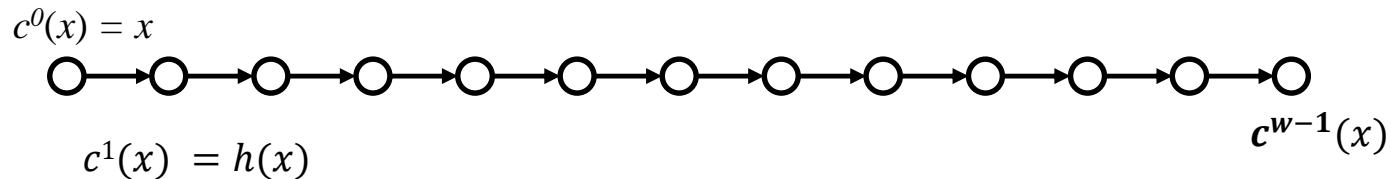
From WOTS+ and FORS to
WOTS+C and FORS+C

Function chains in WOTS*

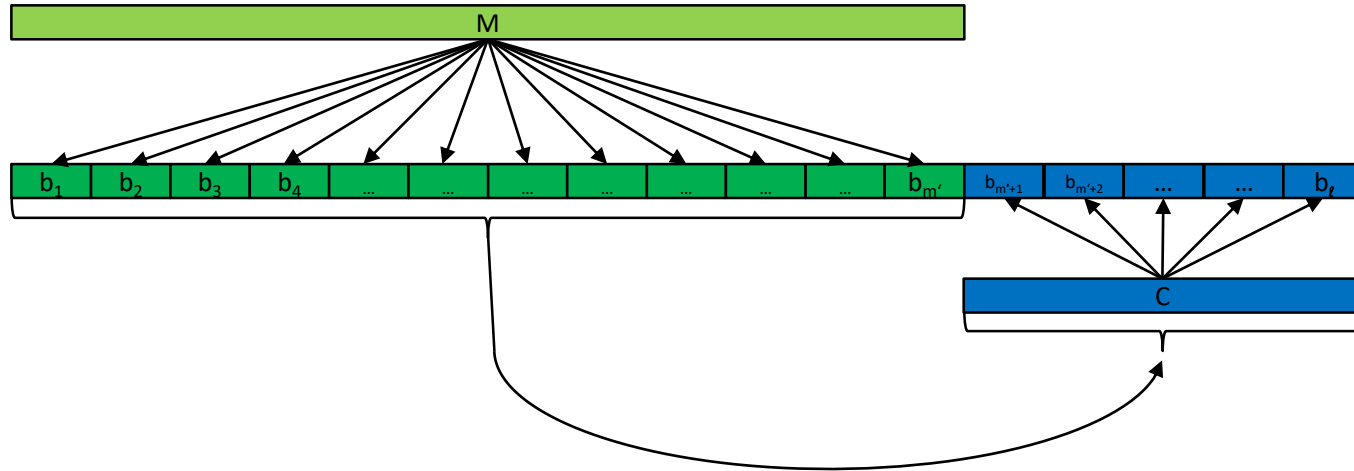
Hash function $h : \{0,1\}^n \rightarrow \{0,1\}^n$

Parameter w

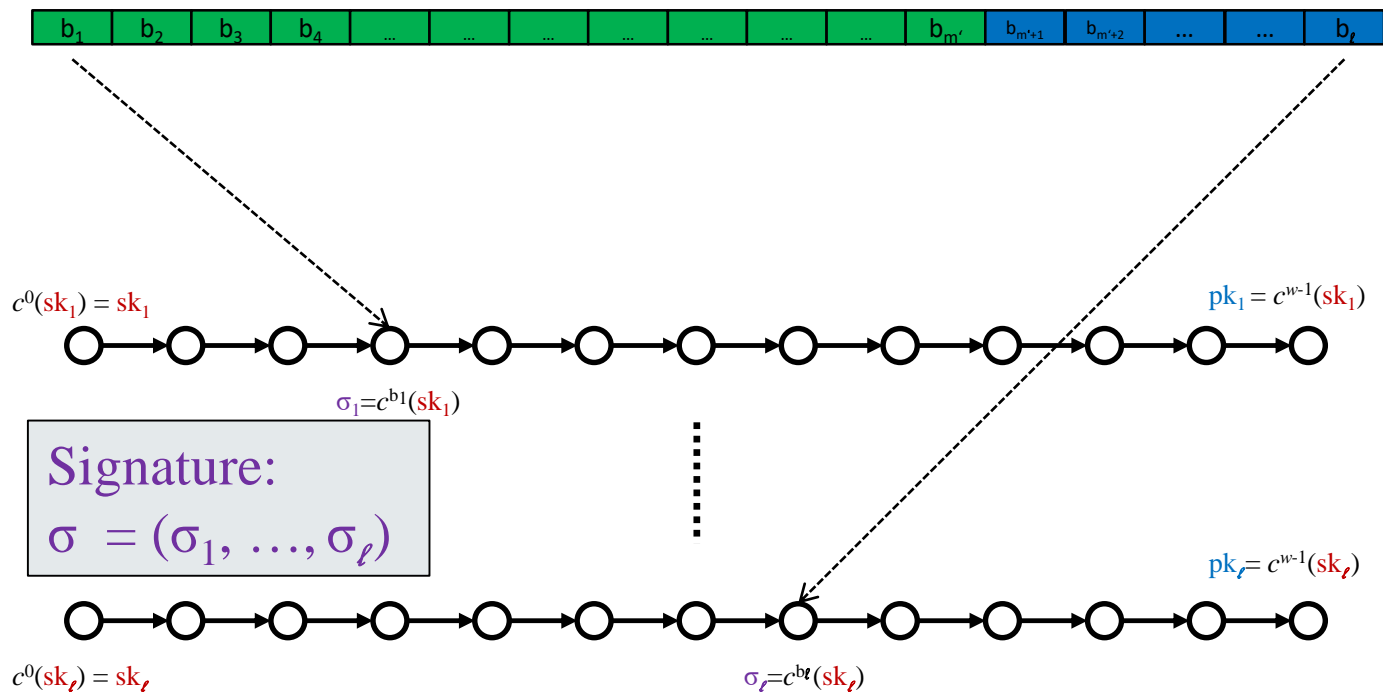
$$\text{Chain: } c^i(x) = h\left(c^{i-1}(x)\right) = \underbrace{h \circ h \circ \dots \circ h}_{i\text{-times}}(x)$$



WOTS Signature generation



WOTS Signature generation



WOTS+C

We **remove the checksum** chains by forcing it to always a pre-defined value

Signing:

Instead of signing the message m , we sign $d = h(s||m)$, where s a salt.

Search for s s.t. d has a checksum S , add s to the signature

S is pre-defined to be the expected checksum.

Signer run-time is usually **reduced!** More work to find salt, but no checksum chains to calculate

Verifying:

Verifier run-time is **reduced**.

No need to verify the checksum chains

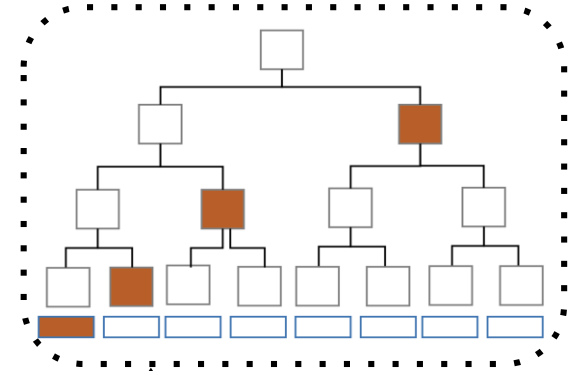
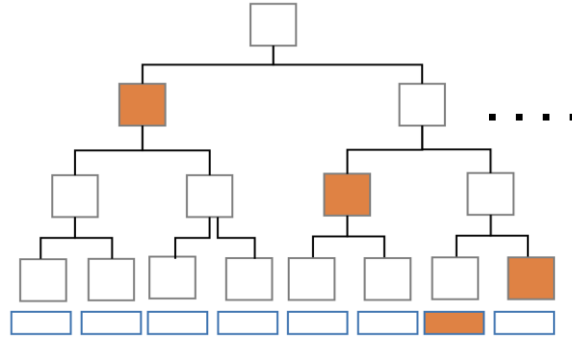
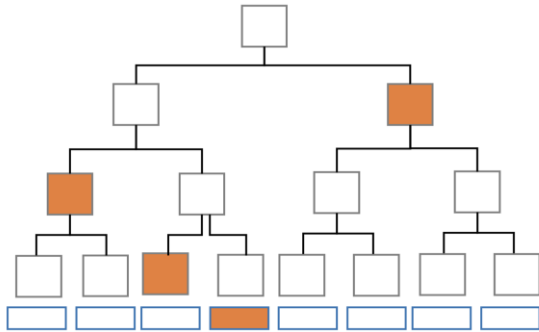
Only compute $d = h(s||m)$ and verify that d has checksum S (and verify the signature)

Can use the same technique to reduce more chains (at the cost of increasing Sig-Gen-Time)

FORS+C

- FORS includes multiple Merkle trees, opening one leaf in each tree
- Using similar techniques, we **remove the last tree** of the FORC+ signature
- **Idea:** force the hash for the last tree to always open the first leaf (leaf index 0)
- Find a salt s that satisfies the above
- **Tweak:** we can make the last tree larger than other to gain savings
- Verifier run-time is **reduced** (simply check that last tree has index 0)

FORS+C



Last tree removed
in FORS+C

SPHINCS+C

SPHINCS+C Parameter Sets

- As a starting point we can use the original SPHINCS+ parameter sets
- This results in a “compressed” version of SPHINCS+ that is strictly better
 - Faster Key-Gen-Time, Sig-Gen-Time, Sig-Ver-Time
 - Smaller Sig-Size

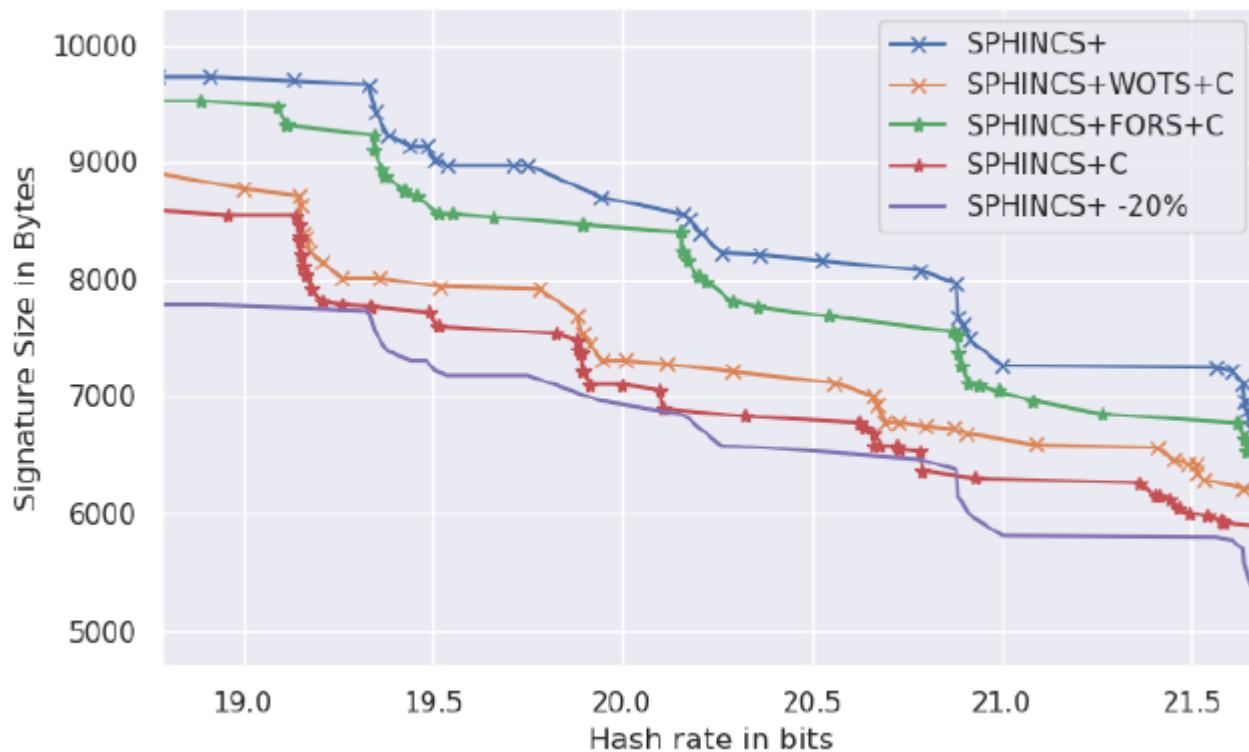
| | Key Generation | | Signature | | Verification | | Size | |
|------------|----------------|--------------|-----------|--------------|--------------|-------------|----------|-------------|
| | SPHINCS+ | Compressed | SPHINCS+ | Compressed | SPHINCS+ | Compressed | SPHINCS+ | Compressed |
| SHAKE-128s | 721.4 | 649.8 (-10%) | 5398.0 | 4964.5 (-8%) | 5.0 | 4.9 (-1%) | 7856 | 7344 (-7%) |
| SHAKE-128f | 10.8 | 9.7 (-11%) | 256.3 | 232.4 (-9%) | 16.4 | 14.2 (-13%) | 17088 | 16012 (-7%) |
| SHAKE-192s | 1068.6 | 962.3 (-10%) | 9133.3 | 8283.6 (-9%) | 8.5 | 7.5 (-12%) | 16224 | 15392 (-6%) |
| SHAKE-192f | 15.1 | 13.4 (-12%) | 380.6 | 347.6 (-9%) | 22.0 | 19.5 (-12%) | 35664 | 33956 (-5%) |
| SHAKE-256s | 652.6 | 648.2 (-1%) | 7573.0 | 7367.8 (-3%) | 11.4 | 11.3 (-1%) | 29792 | 28580 (-5%) |
| SHAKE-256f | 44.4 | 38.5 (-13%) | 860.3 | 763.4 (-11%) | 24.1 | 21.1 (-12%) | 49856 | 47976 (-4%) |

SPHINCS+C Parameter Sets

- However, we can do better.
- We can optimize parameter sets for different constraints and use cases
- E.g., we optimized SPHINCS+C parameters to:
 - Minimize Sig-Size
 - Keeping Sig-Gen-Time at least as fast as SPHINCS+

| | Key Generation | | Signature | | Verification | | Size | |
|------------|----------------|--------------|-----------|---------------|--------------|--------------|----------|--------------|
| | SPHINCS+ | SPHINCS+C | SPHINCS+ | SPHINCS+C | SPHINCS+ | SPHINCS+C | SPHINCS+ | SPHINCS+C |
| SHAKE-128s | 721.4 | 341.1 (-53%) | 5398.0 | 4602.4 (-15%) | 5.0 | 30.8 (+518%) | 7856 | 6304 (-20%) |
| SHAKE-128f | 10.8 | 8.6 (-20%) | 256.3 | 237.6 (-7%) | 16.4 | 12.0 (-27%) | 17088 | 14904 (-13%) |
| SHAKE-192s | 1068.6 | 501.2 (-53%) | 9133.3 | 8107.8 (-11%) | 8.5 | 45.0 (+429%) | 16224 | 13776 (-16%) |
| SHAKE-192f | 15.1 | 12.9 (-15%) | 380.6 | 379.4 (-0%) | 22.0 | 18.6 (-15%) | 35664 | 33016 (-8%) |
| SHAKE-256s | 652.6 | 432.0 (-34%) | 7573.0 | 7339.4 (-3%) | 11.4 | 36.1 (+218%) | 29792 | 26096 (-13%) |
| SHAKE-256f | 44.4 | 37.7 (-15%) | 860.3 | 810.5 (-6%) | 24.1 | 19.9 (-18%) | 49856 | 46884 (-6%) |

Improved Tradeoff with SPHINCS+C



SPHINCS+C Parameter Sets

- SPHINCS+C can provide better tradeoffs compare to SPHINCS+
- We are looking for feedback on real-world requirements and tradeoffs
 - Sig-Gen-Time Vs. Sig-Ver-Time
 - Sig-Size Vs. Sig-Ver-Time
 - Low q_{sign} variants
- The paper includes a sage script for finding suitable parameter sets

Future Work

- In the paper we propose other optimization that require bigger code changes and are not included in SPHINCS+C
 - Interleaved Trees for better FORS compression
 - Small trees of FORS+C
 - Soft-state-full variant to XMSS based on a tree of FORS+C
 - Only need to make sure you don't pass the signature number limit
- Fine-tuning parameter sets choice

Conclusion

- We presented SPHINCS+C a “compressed” variant of SPHINCS+
 - Based on WOTS+C and FORS+C variants of WOTS+ and FORS used in SPHINCS+
 - Full **tight** security proof as in SPHINCS+
- SPHINCS+C allows for better tradeoffs and optimization of parameter sets
- WOTS+C optimizations can also be used in XMSS
- Improved tradeoffs and optimization also for low q_{sign} variants

- Paper available at: ia.cr/2022/778
- Code: <https://github.com/eyalr0/sphincsplusC/>
- Any questions?