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:

SK	PK	Sig (M=0)	Sig (M=1)
X_0	$Y_0 = H(X_0)$	X_0	
X_1	$Y_1 = H(X_1)$		X_1

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

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!

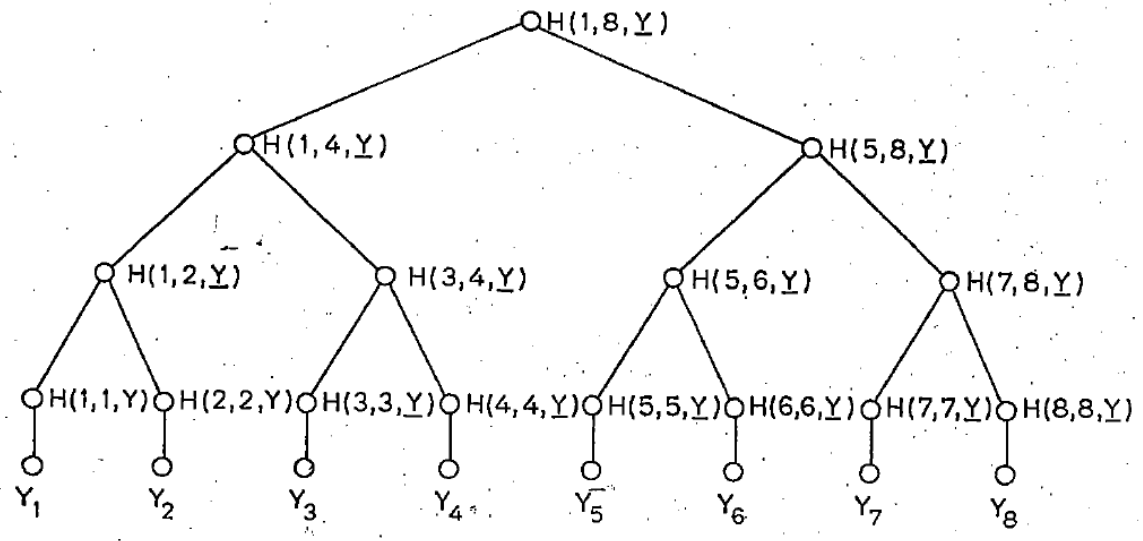
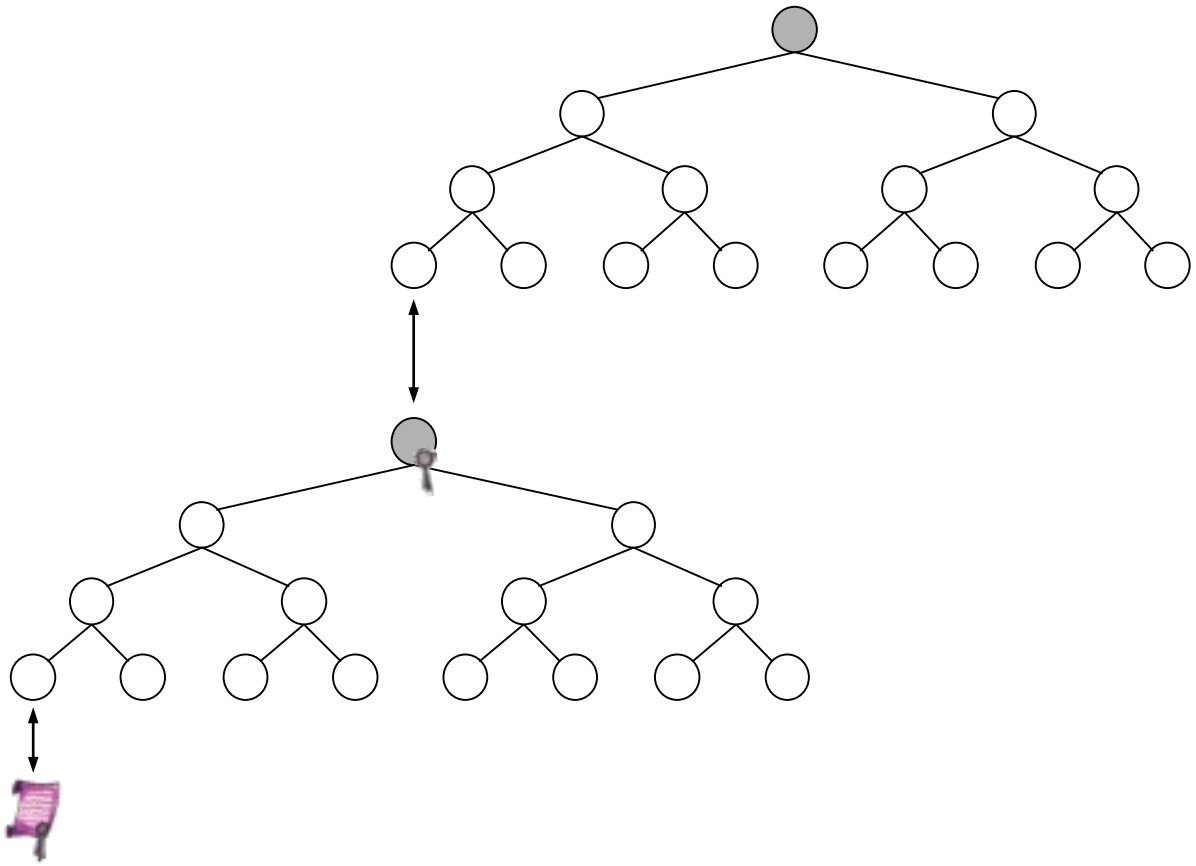


FIG 1
AN AUTHENTICATION TREE WITH N = 8.

PAGE 41B

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

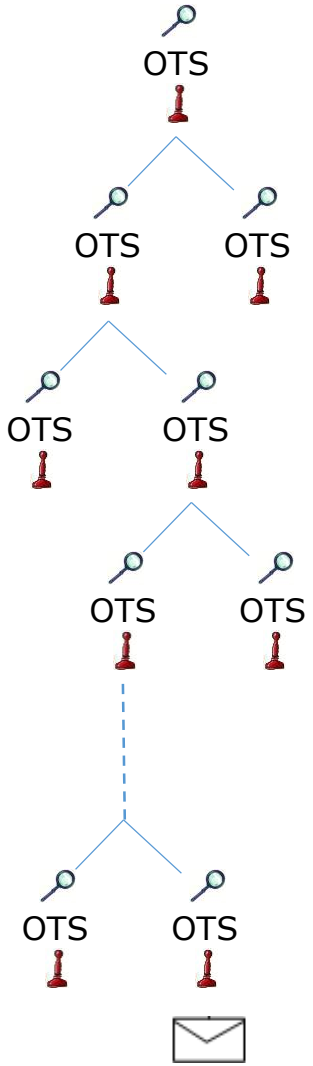
[NY89,Gol87,Gol04]

Goldreich's approach [Gol04]:

Security parameter $\lambda = 128$

Use binary tree as in Merkle, but...

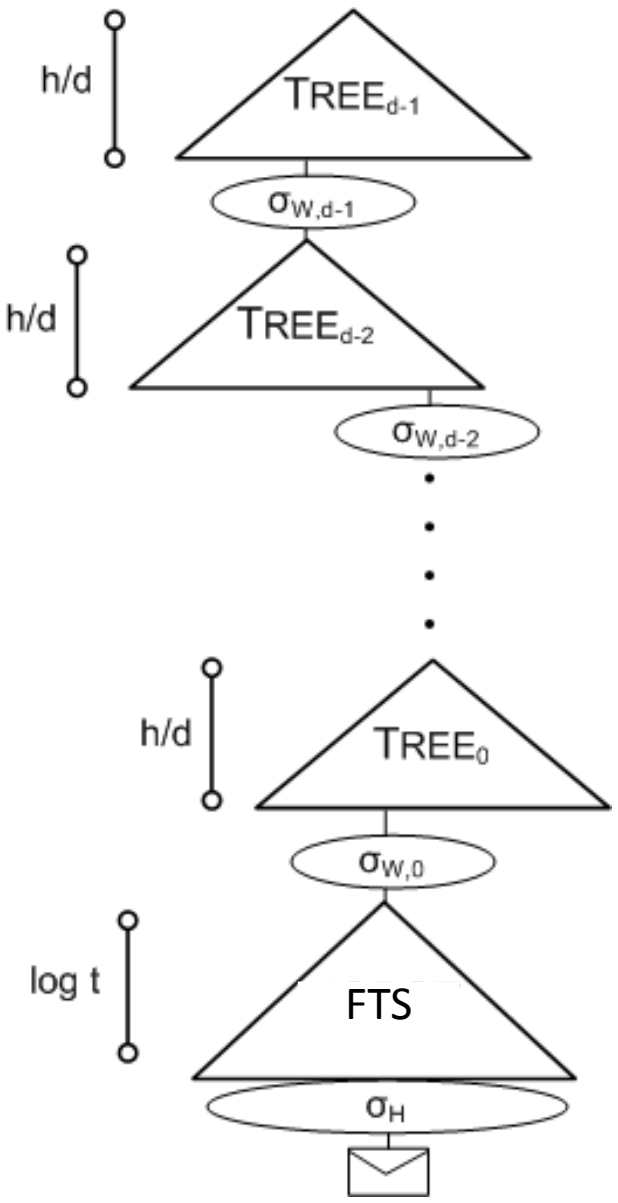
- ...for statelessness
  - pick index i at random;
  - requires huge tree to avoid index collisions (e.g., height $\mathrm{h} = 2\lambda = 256$).
- ...for efficiency:
  - use binary certification tree of OTS key pairs (= Hypertree with $d = h$),
  - all OTS secret keys are generated pseudorandomly.



SPHINCS [BHH+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 2nd 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:
  - $M \rightarrow I_M = (i_0, i_1, \dots, i_k)$
  - Signature consists of $\left(SK_{i_0}, SK_{i_1}, \dots, SK_{i_k}\right)$
  - Generic attack: Find M such that all SK_j selected by I_M have been part of previous signatures.

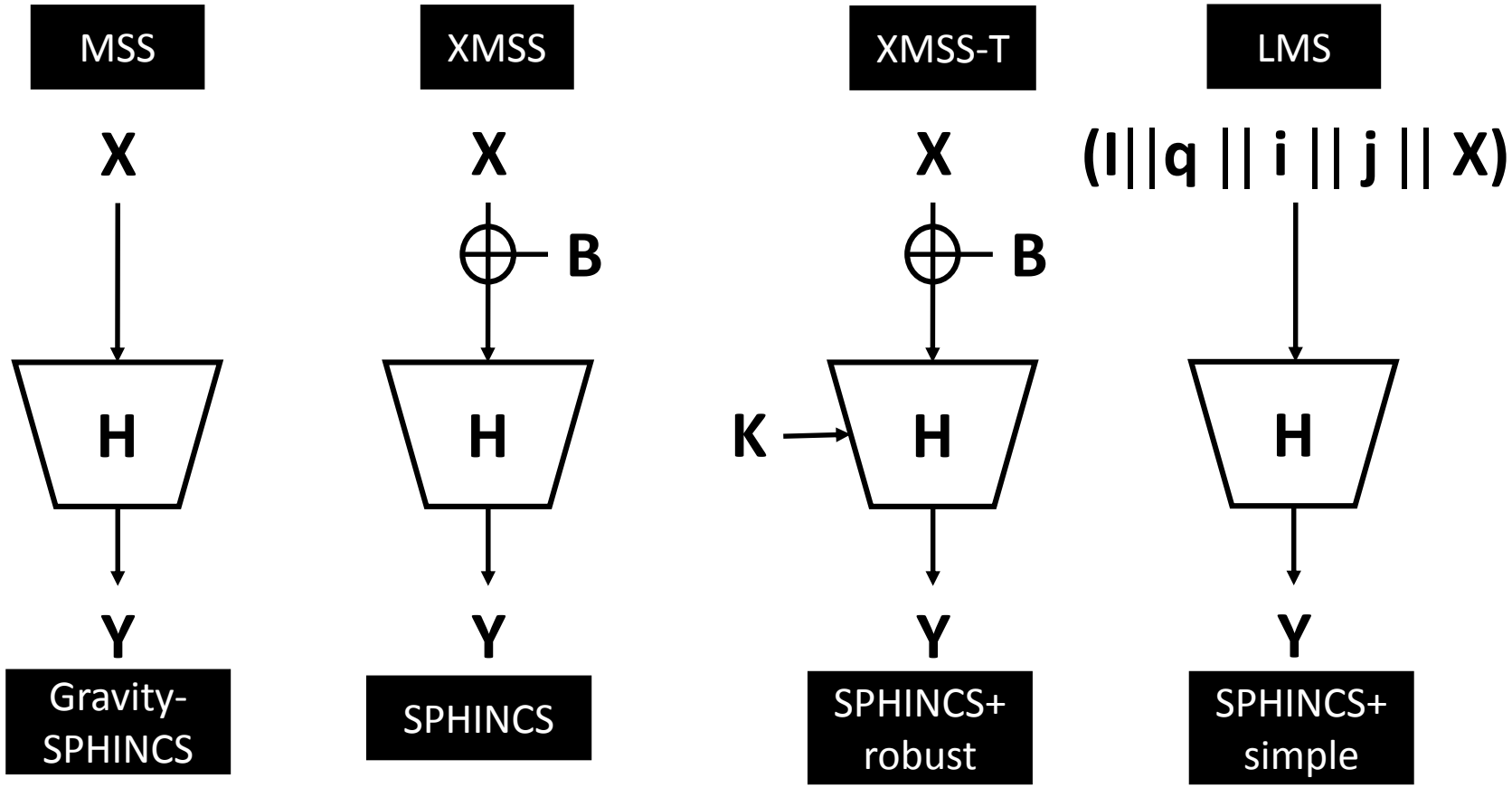
- HORST: Draw all SK_j from same "pool" of secret values
  - Issue: Possible to have $i_0 = i_1 = \cdots = i_k = j$ so signature only requires knowledge of single value SK_j
- FORS: Use k independent „pools" of secret values
  - Generic attacks get harder → can choose smaller parameters → smaller signatures

Tweakable hash functions

A tool for modular proofs for hash-based signatures

Hashing for hash-based signatures

(Change driven by goal to minimize security assumptions)



Tweakable hash function

$$\mathbf{Th}(P, T, X) \rightarrow Y$$

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:
  - **S**ingle-function (= random but fixed P),
  - **M**ulti-target (= adversary A can adaptively define targets (T, M), using oracle inititalized with P)
  - **T**arget-**C**ollision **R**esistance (= 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:
  - ...
  - **D**ecisional **S**econd-**P**reimage **R**esistance (= 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)

Security using tweakable hash

THEOREM 17. *For parameters n, w, h, d, m, t, k as described above,* SPHINCS$^+$ *is* PQ-EU-CMA *secure if*

- $\mathbf{Th}$ *(and thereby also* $\mathbf{F}$ *and* $\mathbf{H}$*) is post-quantum single-function multi-target-collision resistant for distinct tweaks (with tweak advice),*
- $\mathbf{F}$ *is post-quantum single-function multi-target decisional second-preimage resistant for distinct tweaks (with tweak advice),*
- $\mathbf{PRF}$ *and* $\mathbf{PRF_{msg}}$ *are post-quantum pseudorandom function families, and*
- $\mathbf{H_{msg}}$ *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.

Scheme	Cycles			Bytes		
	keypair	sign	verify	sig	pk	sk
Comparison to SPHINCS-256						
SPHINCS-256 [8]	2 868 464[a]	50 462 856[a]	1 672 652[a]	41 000	1 056	1 088
SPHINCS$^+$ (Haraka, robust) ($n = 192, h = 51, d = 17, b = 7, k = 45, w = 16$)	1 254 968[b]	29 015 002[b]	2 739 770[b]	30 696	48	96
Comparison to Gravity-SPHINCS						
Gravity-SPHINCS [5] (parameter-set L)	30 729 044 392[a]	32 564 796[a]	625 752[a]	max: 35 168 avg: ?[c]	32	1 048 608
SPHINCS$^+$ (Haraka, robust) ($n = 192, h = 66, d = 22, b = 8, k = 33, w = 16$)	1 257 826[b]	38 840 268[b]	3 467 192[b]	35 664	48	96
SPHINCS$^+$ (Haraka, simple) ($n = 192, h = 64, d = 16, b = 7, k = 49, w = 16$)	1 892 462[b]	35 029 380[b]	1 460 204[b]	30 552	48	96
Comparison to Picnic						
Picnic2-L5-FS [16]	18 244[c]	904 189 188[c]	268 485 212[c]	max: 54 732 avg: 46 282	65	97
SPHINCS$^+$ (SHA-256, simple) ($n = 256, h = 63, d = 9, b = 12, k = 29, w = 16$)	43 317 320[b]	527 413 100[b]	5 463 884[b]	33 408	64	128

[a] As reported by SUPERCOP [10] from 3.5GHz Intel Xeon E3-1275 V3 (Haswell)
[b] 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`
[c] As reported by SUPERCOP [10] from 3.1GHz Intel Xeon E3-1220 V3 (Haswell)
[d] Neither [5] nor [6] report the average size of signatures; the analysis in [4] suggests that it is about 1KB smaller than the worst-case size.

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 *the* most conservative signature scheme in the NIST PQC competition.

"If you're signing something for the long-term future, and ~~40~~ 30KB sigs is not a problem, use (stateless) hash-based sigs e.g. SPHINCS"

Vadim Lyubashevsky, 2017