

# SPHINCS: practical stateless hash-based signatures

Daniel J. Bernstein  
Daira Hopwood  
Andreas Hülsing  
Tanja Lange  
Ruben Niederhagen  
Louiza Papachristodoulou  
Michael Schneider  
Peter Schwabe  
Zooko Wilcox-O'Hearn

28 April 2015

# Hash-based signatures [Mer90]

- ▶ Security relies only on secure hash function
  - ▶ Post-quantum
  - ▶ Reliable security estimates
- ▶ Fast [BGD<sup>+</sup>06, BDK<sup>+</sup>07, BDH11]
- ▶ Stateful

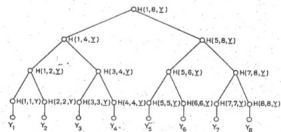
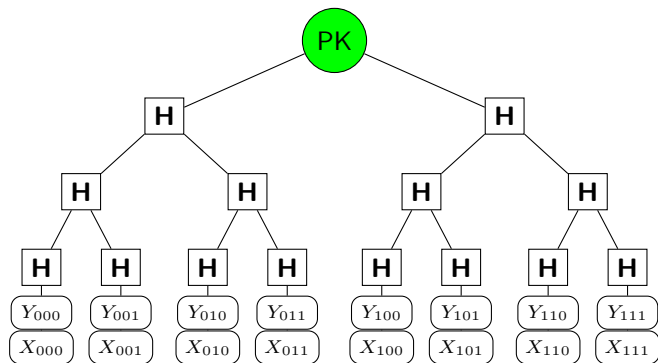


FIG 1  
AN AUTHENTICATION TREE WITH  $n = 8$ .

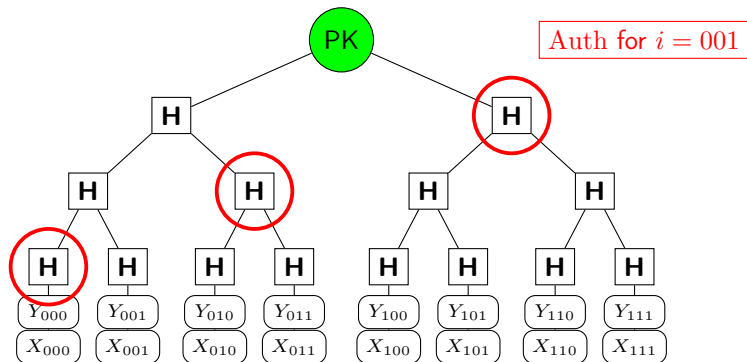
Page 438

# Merkle Trees



- ▶ Merkle, 1979: Leverage one-time signatures to multiple messages
- ▶ Binary hash tree on top of OTS public keys

# Merkle Trees



- ▶ Use OTS keys sequentially
- ▶  $SIG = (i, \text{sign}(M, X_i), Y_i, \text{Auth})$

## About the state

- ▶ Used for *security*:  
Stores index  $i \Rightarrow$  Prevents using one-time keys twice.
- ▶ Used for *efficiency*:  
Stores intermediate results for fast Auth computation.

## About the state

- ▶ Used for *security*:  
Stores index  $i \Rightarrow$  Prevents using one-time keys twice.
- ▶ Used for *efficiency*:  
Stores intermediate results for fast Auth computation.
- ▶ Problems:
  - ▶ Load-balancing
  - ▶ Multi-threading
  - ▶ Backups
  - ▶ Virtual-machine images
  - ▶ ...

## About the state

- ▶ Used for *security*:  
Stores index  $i \Rightarrow$  Prevents using one-time keys twice.
- ▶ Used for *efficiency*:  
Stores intermediate results for fast Auth computation.
- ▶ Problems:
  - ▶ Load-balancing
  - ▶ Multi-threading
  - ▶ Backups
  - ▶ Virtual-machine images
  - ▶ ...
- ▶ “Huge foot-cannon” (Adam Langley, Google)

## About the state

- ▶ Used for *security*:  
Stores index  $i \Rightarrow$  Prevents using one-time keys twice.
- ▶ Used for *efficiency*:  
Stores intermediate results for fast Auth computation.
- ▶ Problems:
  - ▶ Load-balancing
  - ▶ Multi-threading
  - ▶ Backups
  - ▶ Virtual-machine images
  - ▶ ...
- ▶ “Huge foot-cannon” (Adam Langley, Google)
- ▶ Not only a hash-based issue!



ELIMINATE



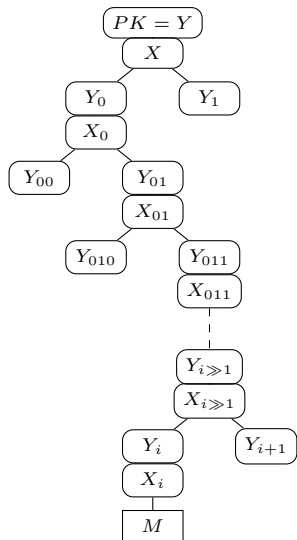
THE STATE

# Protest?



# Stateless hash-based signatures [NY89, Gol87, Gol04]

Goldreich's approach [Gol04]:  
Security parameter  $\lambda = 128$   
Use binary tree as in Merkle, but...



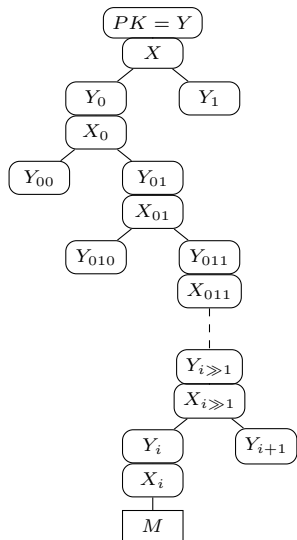
# Stateless hash-based signatures [NY89, Gol87, Gol04]

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  $h = 2\lambda = 256$ ).



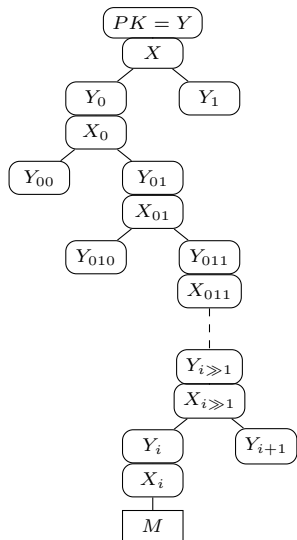
# Stateless hash-based signatures [NY89, Gol87, Gol04]

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;
  - ▶ all OTS secret keys are generated pseudorandomly.



# It works, but signatures are painfully long

- ▶ 0.6 MB for Goldreich signature using short-public-key Winternitz-16 one-time signatures.
- ▶ Would dominate traffic in typical applications, and add user-visible latency on typical network connections.

# It works, but signatures are painfully long

- ▶ 0.6 MB for Goldreich signature using short-public-key Winternitz-16 one-time signatures.
- ▶ Would dominate traffic in typical applications, and add user-visible latency on typical network connections.
- ▶ Example:
  - ▶ Debian operating system is designed for frequent upgrades.
  - ▶ At least one new signature for each upgrade.
  - ▶ Typical upgrade: one package or just a few packages.
  - ▶ 1.2 MB average package size.
  - ▶ 0.08 MB median package size.

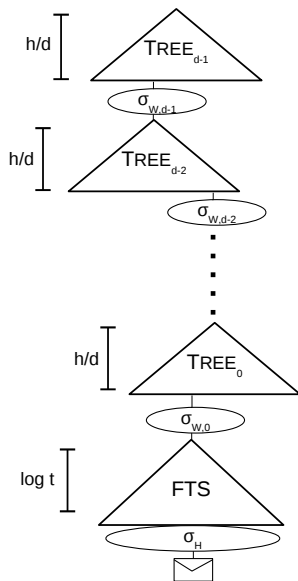
# It works, but signatures are painfully long

- ▶ 0.6 MB for Goldreich signature using short-public-key Winternitz-16 one-time signatures.
- ▶ Would dominate traffic in typical applications, and add user-visible latency on typical network connections.
- ▶ Example:
  - ▶ Debian operating system is designed for frequent upgrades.
  - ▶ At least one new signature for each upgrade.
  - ▶ Typical upgrade: one package or just a few packages.
  - ▶ 1.2 MB average package size.
  - ▶ 0.08 MB median package size.
- ▶ Example:
  - ▶ HTTPS typically sends multiple signatures per page.
  - ▶ 1.8 MB average web page in Alexa Top 1000000.



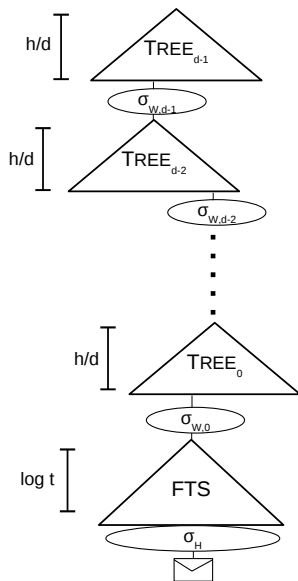
# The SPHINCS approach

- ▶ Use a “hyper-tree” of total height  $h$
- ▶ Parameter  $d \geq 1$ , such that  $d \mid h$
- ▶ Each (Merkle) tree has height  $h/d$
- ▶  $(h/d)$ -ary certification tree



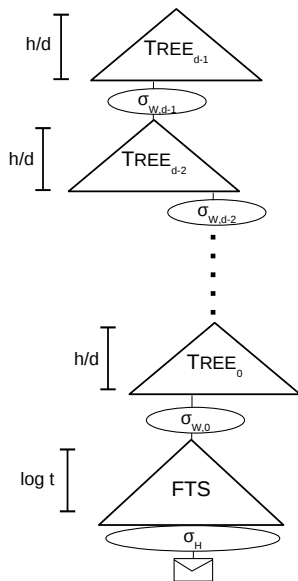
# The SPHINCS approach

- ▶ Pick index (pseudo-)randomly
- ▶ Messages signed with *few-time* signature scheme
- ▶ Significantly reduce total tree height
- ▶ Require  $\Pr[r\text{-times Coll}] \cdot \Pr[\text{Forgery after } r \text{ signatures}] = \text{negl}(n)$



# The SPHINCS approach

- ▶ Designed to be collision-resilient
- ▶ Trees: MSS-SPR trees [DOTV08]
- ▶ OTS: WOTS<sup>+</sup> [Hül13]
- ▶ FTS: HORST (HORS [RR02] with tree)



# SPHINCS-256

- ▶ Designed for 128 bits of post-quantum security  
(**yes, we did the analysis!**)
- ▶ 12 trees of height 5 each

# SPHINCS-256

- ▶ Designed for 128 bits of post-quantum security (**yes, we did the analysis!**)
- ▶ 12 trees of height 5 each
- ▶  $n = 256$  bit hashes in WOTS and HORST
- ▶ Winternitz parameter  $w = 16$
- ▶ HORST with  $2^{16}$  expanded-secret-key chunks (total: 2 MB)

# SPHINCS-256

- ▶ Designed for 128 bits of post-quantum security (**yes, we did the analysis!**)
- ▶ 12 trees of height 5 each
- ▶  $n = 256$  bit hashes in WOTS and HORST
- ▶ Winternitz parameter  $w = 16$
- ▶ HORST with  $2^{16}$  expanded-secret-key chunks (total: 2 MB)
- ▶  $m = 512$  bit message hash (BLAKE-512 [[ANWOW13](#)])
- ▶ ChaCha12 [[Ber08](#)] as PRG

# Cost of SPHINCS-256 signing

- ▶ Three main components:
  - ▶ PRG for HORST secret-key expansion to 2 MB
  - ▶ Hashing in WOTS and HORS public-key generation:  
 $F : \{0, 1\}^{256} \rightarrow \{0, 1\}^{256}$
  - ▶ Hashing in trees (mainly HORST public-key):  
 $H : \{0, 1\}^{512} \rightarrow \{0, 1\}^{256}$
- ▶ Overall: 451 456 invocations of  $F$ , 91 251 invocations of  $H$

# Cost of SPHINCS-256 signing

- ▶ Three main components:
  - ▶ PRG for HORST secret-key expansion to 2 MB
  - ▶ Hashing in WOTS and HORS public-key generation:  
 $F : \{0, 1\}^{256} \rightarrow \{0, 1\}^{256}$
  - ▶ Hashing in trees (mainly HORST public-key):  
 $H : \{0, 1\}^{512} \rightarrow \{0, 1\}^{256}$
- ▶ Overall: 451 456 invocations of  $F$ , 91 251 invocations of  $H$
- ▶ Full hash function would be overkill for  $F$  and  $H$
- ▶ Construction in SPHINCS-256:
  - ▶  $F(M_1) = \text{Chop}_{256}(\pi(M_1||C))$
  - ▶  $H(M_1||M_2) = \text{Chop}_{256}(\pi(\pi(M_1||C) \oplus (M_2||0^{256})))$



# Cost of SPHINCS-256 signing

- ▶ Three main components:
  - ▶ PRG for HORST secret-key expansion to 2 MB
  - ▶ Hashing in WOTS and HORS public-key generation:  
 $F : \{0, 1\}^{256} \rightarrow \{0, 1\}^{256}$
  - ▶ Hashing in trees (mainly HORST public-key):  
 $H : \{0, 1\}^{512} \rightarrow \{0, 1\}^{256}$
- ▶ Overall: 451 456 invocations of  $F$ , 91 251 invocations of  $H$
- ▶ Full hash function would be overkill for  $F$  and  $H$
- ▶ Construction in SPHINCS-256:
  - ▶  $F(M_1) = \text{Chop}_{256}(\pi(M_1||C))$
  - ▶  $H(M_1||M_2) = \text{Chop}_{256}(\pi(\pi(M_1||C) \oplus (M_2||0^{256})))$
- ▶ Use fast ChaCha12 permutation for  $\pi$
- ▶ All building blocks (PRG, message hash,  $H$ ,  $F$ ) built from very similar permutations

# SPHINCS-256 speed and sizes

## SPHINCS-256 sizes

- ▶ 0.041 MB signature ( $\approx 15\times$  smaller than Goldreich!)
- ▶ 0.001 MB public key
- ▶ 0.001 MB private key

# SPHINCS-256 speed and sizes

## SPHINCS-256 sizes

- ▶ 0.041 MB signature ( $\approx 15\times$  smaller than Goldreich!)
- ▶ 0.001 MB public key
- ▶ 0.001 MB private key

## High-speed implementation

- ▶ Target Intel Haswell with 256-bit AVX2 vector instructions
- ▶ Use  $8\times$  parallel hashing, vectorize on high level
- ▶  $\approx 1.6$  cycles/byte for  $H$  and  $F$

# SPHINCS-256 speed and sizes

## SPHINCS-256 sizes

- ▶ 0.041 MB signature ( $\approx 15\times$  smaller than Goldreich!)
- ▶ 0.001 MB public key
- ▶ 0.001 MB private key

## High-speed implementation

- ▶ Target Intel Haswell with 256-bit AVX2 vector instructions
- ▶ Use  $8\times$  parallel hashing, vectorize on high level
- ▶  $\approx 1.6$  cycles/byte for  $H$  and  $F$

## SPHINCS-256 speed

- ▶ Signing:  $< 52$  Mio. Haswell cycles ( $> 200$  sigs/sec, 4 Core, 3GHz)
- ▶ Verification:  $< 1.5$  Mio. Haswell cycles
- ▶ Keygen:  $< 3.3$  Mio. Haswell cycles

# SPHINCS: Stateless Practical Hash-based Incredibly Nice Collision-resilient Signatures



<http://sphincs.cr.yp.to>

# References I



Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein.

BLAKE2: Simpler, smaller, fast as MD5.

In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *Applied Cryptography and Network Security*, volume 7954 of LNCS, pages 119–135. Springer, 2013.



Johannes Buchmann, Erik Dahmen, and Andreas Hülsing.

XMSS - a practical forward secure signature scheme based on minimal security assumptions.

In Bo-Yin Yang, editor, *Post-Quantum Cryptography*, volume 7071 of LNCS, pages 117–129. Springer, 2011.



Johannes Buchmann, Erik Dahmen, Elena Klintsevich, Katsuyuki Okeya, and Camille Vuillaume.

Merkle signatures with virtually unlimited signature capacity.

In Jonathan Katz and Moti Yung, editors, *Applied Cryptography and Network Security*, volume 4521 of LNCS, pages 31–45. Springer, 2007.



Daniel J. Bernstein.

ChaCha, a variant of Salsa20.

SASC 2008: The State of the Art of Stream Ciphers, 2008.

# References II



Johannes Buchmann, L. C. Coronado García, Erik Dahmen, Martin Döring, and Elena Klintsevich.

CMSS - an improved Merkle signature scheme.

In Rana Barua and Tanja Lange, editors, *Progress in Cryptology – INDOCRYPT 2006*, volume 4329 of *LNCS*, pages 349–363. Springer, 2006.



Erik Dahmen, Katsuyuki Okeya, Tsuyoshi Takagi, and Camille Vuillaume.

Digital signatures out of second-preimage resistant hash functions.

In Johannes Buchmann and Jintai Ding, editors, *Post-Quantum Cryptography*, volume 5299 of *LNCS*, pages 109–123. Springer, 2008.



Oded Goldreich.

Two remarks concerning the goldwasser-micali-rivest signature scheme.

In Andrew M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86*, volume 263 of *LNCS*, pages 104–110. Springer, 1987.



Oded Goldreich.

*Foundations of Cryptography: Volume 2, Basic Applications.*

Cambridge University Press, Cambridge, UK, 2004.

# References III



**Andreas Hülsing.**

W-OTS+ – shorter signatures for hash-based signature schemes.

In Amr Youssef, Abderrahmane Nitaj, and Aboul-Ella Hassanien, editors, *Progress in Cryptology – AFRICACRYPT 2013*, volume 7918 of *LNCS*, pages 173–188. Springer, 2013.



**Ralph Merkle.**

A certified digital signature.

In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO '89*, volume 435 of *LNCS*, pages 218–238. Springer, 1990.



**M. Naor and M. Yung.**

Universal one-way hash functions and their cryptographic applications.

In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, page 43. ACM, 1989.



**Leonid Reyzin and Natan Reyzin.**

Better than BiBa: Short one-time signatures with fast signing and verifying.

In Lynn Batten and Jennifer Seberry, editors, *Information Security and Privacy 2002*, volume 2384 of *LNCS*, pages 1–47. Springer, 2002.



## Picture sources

- ▶ “Black Bloc Hamburg” by Autonome NewsflasherInnen - <http://de.indymedia.org/2007/12/202692.shtml>. Licensed under CC BY-SA 2.0 de via Wikimedia Commons - [http://commons.wikimedia.org/wiki/File:Black\\_Bloc\\_Hamburg.jpg#/media/File:Black\\_Bloc\\_Hamburg.jpg](http://commons.wikimedia.org/wiki/File:Black_Bloc_Hamburg.jpg#/media/File:Black_Bloc_Hamburg.jpg)