Hash-based Signatures and upcoming topics

Andreas Hülsing
Eindhoven University of Technology

ESA Workshop on
"Secure Communications for Space Missions in the Post-Quantum Era"
June 27, 2022

Outline

<u>Part I – HBS</u>

1. Background
2. Standardization
   - XMSS & LMS
   - SPHINCS$^+$
3. Recent results
   - Updated security proof

<u>Part II - Next steps</u>

1. Formally verified proofs
2. From primitives to communication
   - PQ-WireGuard
   - PQNoise
3. QROM Security
4. A note on QKD

Hash-based Signature Schemes

[Mer89]

- Conservative choice
- Only secure hash function needed
- First standardized PQC signatures (XMSS,LMS)
- One NIST round 3 scheme (SPHINCS$^+$)
- Fast verification
- Small keys

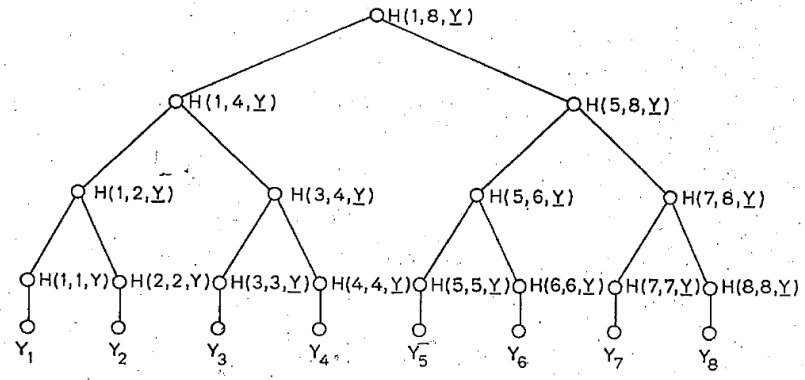
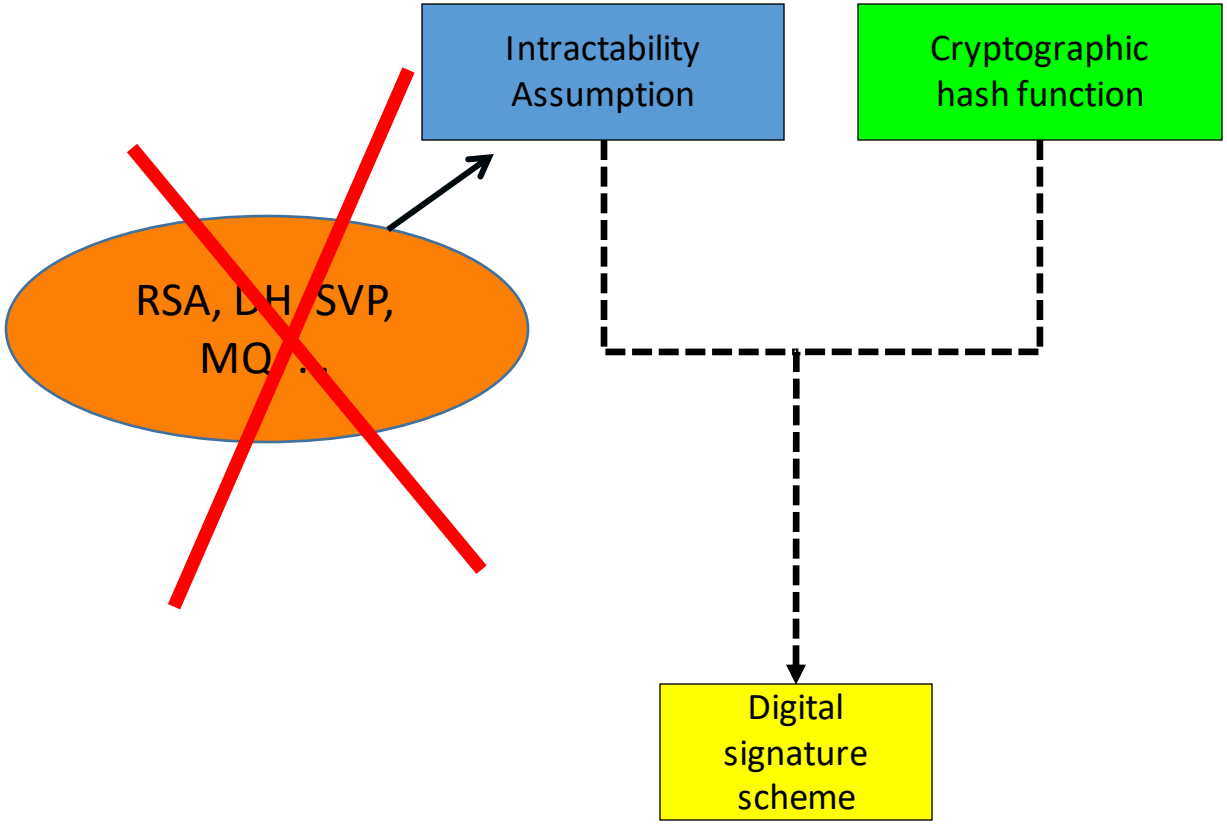


FIG 1
AN AUTHENTICATION TREE WITH N = 8.

PAGE 41B

RSA – EC-DSA – Falcon - Dilithium…



Hash-based One-time Signatures

[Lam79]

Spy wants to signal HQ that they safely arrived in the field

FaceMePLS from The Hague, The Netherlands - Dutch intelligence agency AIVD

Hash-based One-time Signatures

[Lam79]

$\xrightarrow{\quad y = H(x) \quad}$

FaceMePLS from The Hague, The Netherlands
- Dutch intelligence agency AIVD

When arrived safely,
send z = x.

$\xrightarrow{\quad z \quad}$

If H(z) = y
  Spy arrived
Else
  Problems

Hash-based One-time Signatures

[Lam79]

$y_1 = H(x_1)$ →

$y_2 = H(x_2)$ →

- -

FaceMePLS from The Hague, The Netherlands
- Dutch intelligence agency AIVD

If they are attacking,
send $z = x_1$.
If they are pulling back,
send $z = x_2$.

z →

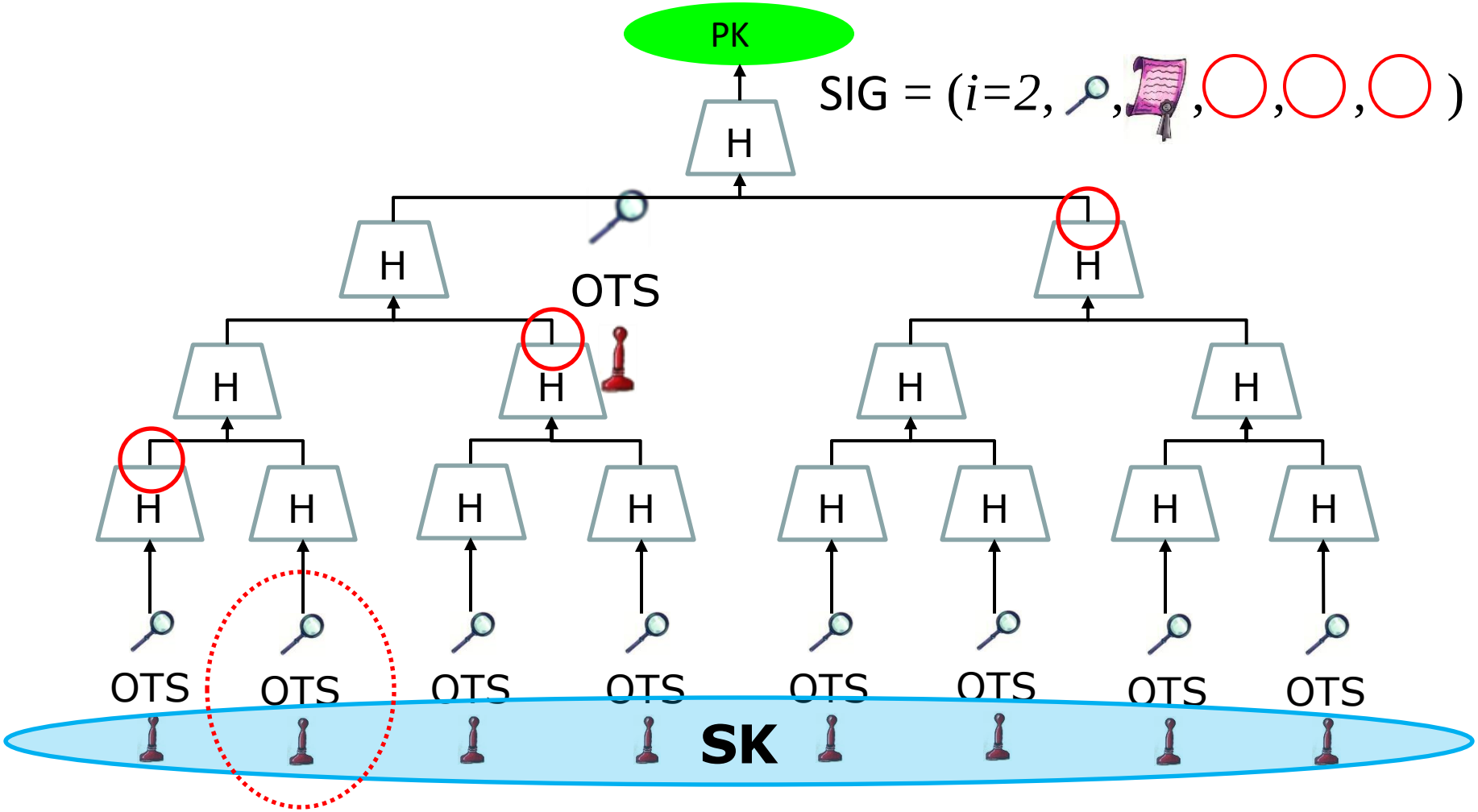
If $H(z) = y_1$
They are attacking!
Else if $H(z) = y_2$
It's safe to send buses.
Else
Something went wrong

Hash-based One-time Signatures

[Lam79]

- Send n pairs --> Allows to sign hashes of length n
- Can be optimized using "hash chains" (WOTS)
- See my homepage for detailed summer school talks

Merkle's Hash-based Signatures



Generic Security

	OW	SPR	CR	UD	PRF
Classical	$\Theta(2^n)$	$\Theta(2^n)$	$\Theta(2^{n/2})$	$\Theta(2^n)$	$\Theta(2^n)$
Quantum	$\Theta(2^{n/2})$	$\Theta(2^{n/2})$	$\Theta(2^{n/3})$	$\Theta(2^{n/2})$	$\Theta(2^{n/2})$

Can be tightly related to very basic quantum distinguishing problems over boolean functions.

Categorizing hash-based signatures

Stateful vs Stateless

Stateful
(LMS & XMSS)

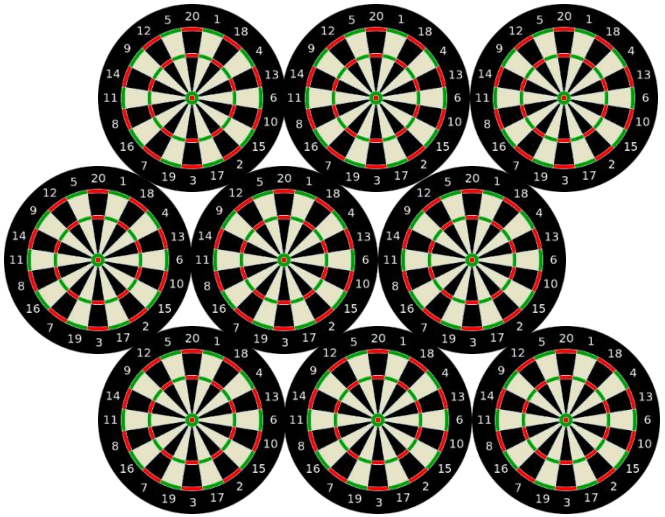
- Secret key has to be updated after every signature
- Requires advanced & <u>secure</u> key management:
  - No copies == no backup!
  - No parallel use (of same key)
- Fast & small

Stateless
(SPHINCS / SPHINCS$^+$)

- Secret key stays the same
- Key management like today
- Penalty in performance

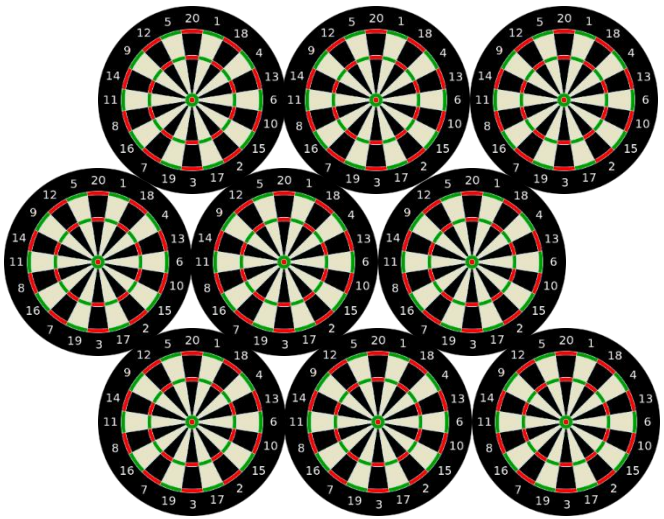
Multi-target attacks

- WOTS & Lamport need hash function h to be one-way
- Hypertree of total height 60 with WOTS (w=16) leads $> 2^{60} \cdot 67 \approx 2^{66}$ images.
- Inverting one of them allows existential forgery (at least massively reduces complexity)
- q-query brute-force succeeds with probability $\Theta\left(\frac{q}{2^{n-66}}\right)$ conventional and $\Theta\left(\frac{q^2}{2^{n-66}}\right)$ quantum
- We lose 66 bits of security! (33 bits quantum)



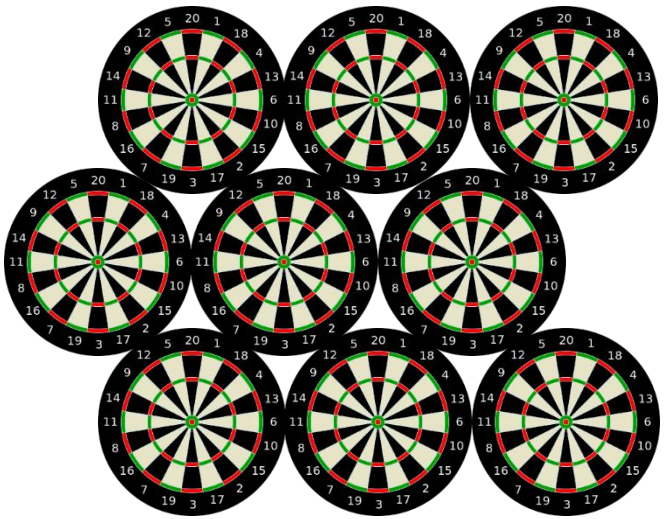
Multi-target attacks: Mitigation

- Mitigation: Separate targets [HRS16]
- Common approach:
  - In addition to hash function description
    and „input“ take
    - Hash „Address“
      (uniqueness in key pair)
    - Hash „key“ used for all hashes of one key pair
      (uniqueness among key pairs)



Multi-target attacks: Mitigation

- Mitigation: Separate targets [HRS16]
- Common approach:
  - In addition to hash function description
    and „input“ take
    - Hash „Address“
      (uniqueness in key pair)
    - Hash „key“ used for all hashes of one key pair
      (uniqueness among key pairs)



New intermediate abstraction: Tweakable Hash Function

- Tweakable Hash Function:

$$\mathbf{Th}(P, T, M) \rightarrow MD$$

P: Public parameters (one per key pair)
T: Tweak (one per hash call)
M: Message
MD: Message Digest

Security properties are determined by instantiation of tweakable hash!

Categorizing HBS

	Stateful	**Stateless**
Constructing tweakable HF assuming SHA2 / SHA3 are good keyed HF	XMSS / Pyramid-robust	SPHINCS / SPHINCS^{+}-robust
Assuming SHA2 / SHA3 is a good tweakable HF	LMS / Pyramid-simple	SPHINCS+-simple

The stateful schemes

[Docs] [txt|pdf] [draft-irtf-cfrg...] [Tracker] [Diff1] [Diff2] [Errata]

INFORMATIONAL
Errata Exist

Internet Research Task Force (IRTF)
Request for Comments: 8391
Category: Informational
ISSN: 2070-1721

A. Huelsing
TU Eindhoven
D. Butin
TU Darmstadt
S. Gazdag
genua GmbH
J. Rijneveld
Radboud University
A. Mohaisen
University of Central Florida
May 2018

XMSS: eXtended Merkle Signature Scheme

Abstract

This note describes the eXtended Merkle Signature Scheme (XMSS), a hash-based digital signature system that is based on existing descriptions in scientific literature. This note specifies Winternitz One-Time Signature Plus (WOTS+), a one-time signature scheme; XMSS, a single-tree scheme; and XMSS^MT, a multi-tree variant of XMSS. Both XMSS and XMSS^MT use WOTS+ as a main building block. XMSS provides cryptographic digital signatures without relying on the conjectured hardness of mathematical problems. Instead, it is proven that it only relies on the properties of cryptographic hash functions. XMSS provides strong security guarantees and is even secure when the collision resistance of the underlying hash function is broken. It is suitable for compact implementations, is relatively simple to implement, and naturally resists side-channel attacks. Unlike most other signature systems, hash-based signatures can so far withstand known attacks using quantum computers.

[Docs] [txt|pdf] [draft-mcgrew-ha...] [Tracker] [Diff1] [Diff2]

INFORMATIONAL

Internet Research Task Force (IRTF)
Request for Comments: 8554
Category: Informational
ISSN: 2070-1721

D. McGrew
M. Curcio
S. Fluhrer
Cisco Systems
April 2019

Leighton-Micali Hash-Based Signatures

Abstract

This note describes a digital-signature system based on cryptographic hash functions, following the seminal work in this area of Lamport, Diffie, Winternitz, and Merkle, as adapted by Leighton and Micali in 1995. It specifies a one-time signature scheme and a general signature scheme. These systems provide asymmetric authentication without using large integer mathematics and can achieve a high security level. They are suitable for compact implementations, are relatively simple to implement, and are naturally resistant to side-channel attacks. Unlike many other signature systems, hash-based signatures would still be secure even if it proves feasible for an attacker to build a quantum computer.

This document is a product of the Crypto Forum Research Group (CFRG) in the IRTF. This has been reviewed by many researchers, both in the research group and outside of it. The Acknowledgements section lists many of them.

LMS vs XMSS

- Difference mostly limited to "tweakable hash function"
- LMS is factor 3-4 faster
- LMS makes somewhat stronger assumptions about the security properties of the used hash function
- More research on construction of "tweakable hash functions" needed

XMSS / XMSS-T Implementation

C Implementation, using OpenSSL [HRS16]

	Sign (ms)	Signature (kB)	Public Key (kB)	Secret Key (kB)	Bit Security classical/ quantum	Comment
XMSS	3.24	2.8	1.3	2.2	236 / 118	h = 20, d = 1,
XMSS-T	9.48	2.8	**0.064**	2.2	**256 / 128**	h = 20, d = 1
XMSS	3.59	8.3	1.3	14.6	196 / 98	h = 60, d = 3
XMSS-T	10.54	8.3	**0.064**	14.6	**256 / 128**	h = 60, d = 3

Intel(R) Core(TM) i7 CPU @ 3.50GHz
XMSS-T uses message digest from Internet-Draft
All using SHA2-256, w = 16 and k = 2

LMS / XMSS performance

[LMS vs XMSS: Comparion of two Hash-Based Signature Standards
Panos Kampanakis & Scott Fluhrer, eprint 2017-349]

Operation	LMS	XMSS	XMSS / LMS ratio
	XMSSMT_SHA2-256_W16_H20_D2		
PK Gen	0.89 s	3.26 s	3.66
Sign	1.21 ms	4.72 ms	3.90
Verify	0.339 ms	1.76 ms	5.19
	XMSSMT_SHA2-256_W16_H40_D2		
PK Gen	720 s	3340 s	4.64
Sign	1.91 ms	7.70 ms	4.03
Verify	0.350 ms	1.75 ms	5.00

Table 6: Measured time per operation for LMS and XMSS

Machine unknown. Guess: general purpose CPU from ~2017

LMS vs XMSS: Comparison of Stateful Hash-Based Signature Schemes on ARM Cortex-M4.

Fabio Campos, Tim Kohlstadt, Steffen Reith, and Marc Stöttinger. Africacrypt '20

Table 6: Performance comparison for SHA-256, $n = 256$, $w = 16$, and $h = 10$.

	LMS	XMSS_ROBUST	ratio[a]	XMSS_SIMPLE	ratio[b]	XMSS_SIMPLE+PRE	ratio[c]
key gen[d]	3774.88	23631.70	6.26	7792.23	2.06	7586.15	2.01
sign[d]	3791.15	23642.03	6.23	7796.39	2.05	7596.24	2.00
verify[d]	2.65	13.07	4.93	3.57	1.34	4.20	1.58

[a] XMSS_ROBUST/LMS
[b] XMSS_SIMPLE/LMS
[c] XMSS_SIMPLE+PRE/LMS
[d] All results (apart from ratio) are given in 10^6 clock cycles.

SPHINCS

Stateless hash-based signatures

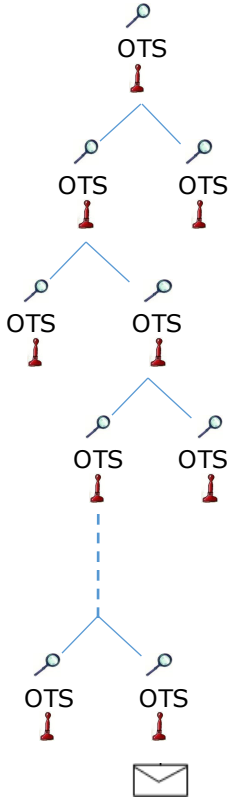
[NY89,Gol87,Gol04]

Goldreich's approach [Gol04]:

Security parameter $\lambda = 128$

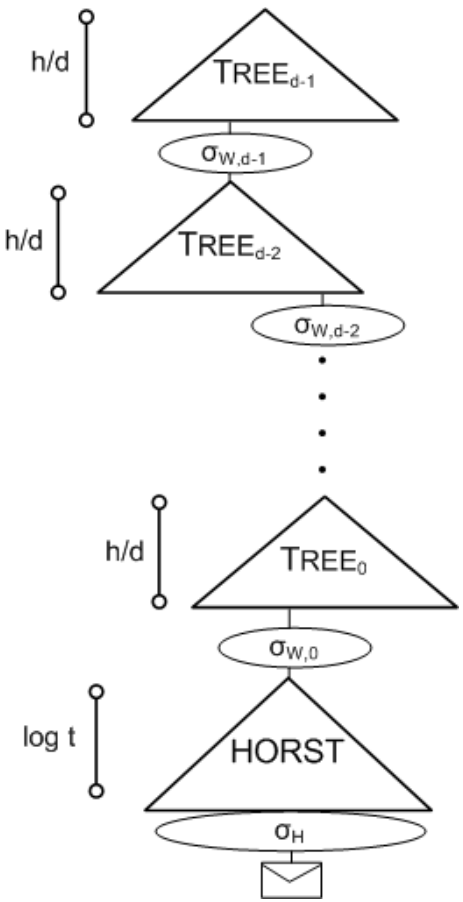
Use binary tree as in Merkle, but...

- ...for security
  - pick index i at random;
  - requires huge tree to avoid index collisions (e.g., height $\mathrm{h} = 2\lambda = 256$).
- ...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 pseudo-randomly
- 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$^+$

Joint work with Jean-Philippe Aumasson, Daniel J. Bernstein,Ward Beullens, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Bas Westerbaan

Instantiations (small vs fast)

Table 3: Example parameter sets for SPHINCS$^+$ targeting different security levels and different tradeoffs between size and speed. Note that these parameter sets have been update for round 3. The column labeled "bitsec" gives the bit security computed as described in Section 9; the column labeled "sec level" gives the security level according to the levels specified in Section 4.A.5 of the Call for Proposals. As explained later, **for Haraka the security level is limited to 2:** i.e., it is 1 for $n = 16$, and 2 for $n = 24$ or $n = 32$.

	n	h	d	$\log(t)$	k	w	bitsec	sec level	sig bytes
SPHINCS$^+$-128s	16	63	7	12	14	16	133	**1**	7 856
SPHINCS$^+$-128f	16	66	22	6	33	16	128	**1**	17 088
SPHINCS$^+$-192s	24	63	7	14	17	16	193	**3**	16 224
SPHINCS$^+$-192f	24	66	22	8	33	16	194	**3**	35 664
SPHINCS$^+$-256s	32	64	8	14	22	16	255	**5**	29 792
SPHINCS$^+$-256f	32	68	17	9	35	16	255	**5**	49 856

Speed

	key generation	signing	verification
SPHINCS$^+$-SHA2-128s-simple	84 964 790	644 740 090	861 478
SPHINCS$^+$-SHA2-128s-robust	175 257 460	1 328 848 352	1 827 104
SPHINCS$^+$-SHA2-128f-simple	1 334 220	33 651 546	2 150 290
SPHINCS$^+$-SHA2-128f-robust	2 748 026	68 541 846	4 801 338
SPHINCS$^+$-SHA2-192s-simple	125 310 788	1 246 378 060	1 444 030
SPHINCS$^+$-SHA2-192s-robust	260 903 972	2 517 396 082	3 103 732
SPHINCS$^+$-SHA2-192f-simple	1 928 970	55 320 742	3 492 210
SPHINCS$^+$-SHA2-192f-robust	4 063 066	113 484 456	7 552 358
SPHINCS$^+$-SHA2-256s-simple	80 943 202	1 025 721 040	1 986 974
SPHINCS$^+$-SHA2-256s-robust	339 101 780	3 912 132 754	8 294 732
SPHINCS$^+$-SHA2-256f-simple	5 067 546	109 104 452	3 559 052
SPHINCS$^+$-SHA2-256f-robust	21 327 470	435 984 168	14 938 510

Table 6: Runtime benchmarks for SPHINCS$^+$-SHA2 on AVX2

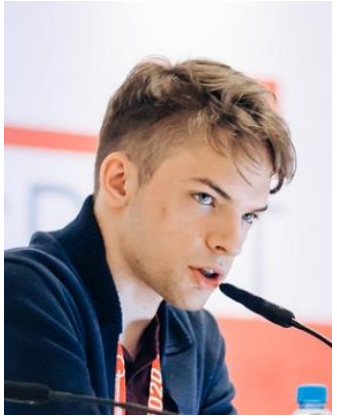
Single core of a 3.1 GHz Intel Xeon E3-1220 CPU (Haswell)

Recent results

New tight security analysis

with Mike Kudinov (eprint)

- Tight security analysis from weaker properties [HRS16] was flawed
- No attack / non-tight proof not concerned
- New tight security proof



Part II - Next steps

Formally verified proofs

with Matthias Meijers

- Proofs of security for PQC
  - are more complicated
  - make use of new tools
  - fewer knowledgeable reviewers
- Consequences
  - More error prone
  - Reduced confidence
- Solution:
  - Use computers to verify computational proofs
  - Reduces verification effort to verification of statements

Formally verified security of Saber

with Matthias Meijers & Pierre-Yves Strub (CRYPTO '22)

Saber

- Lattice-based KEM
- Finalist in NIST competition

Formal analysis of

- CPA-security proof for PKE, and
- correctness proof using EasyCrypt
- sufficient to apply FO-transform

Formal verification of HBS proofs

with Matthias Meijers & several more people from the Formosa team

- Formal verification using EasyCrypt
- Progress:
  - Traditional MSS scheme (schoolbook version) already verified
  - XMSS: Close to being finished (also covers critical proof part for SPHINCS+)
  - SPHINCS+: Starting
- Far more advanced than Saber proof

From PQ-secure primitives to PQ-secure communication

with Florian Weber & many more

- Process so far largely limited to new primitives
- This is cutting short
  - No plug'n'play for DH
  - Changed performance
  - Classical analysis does not necessarily apply
- We need new / updated protocols

From PQ-secure primitives to PQ-secure communication

with Florian Weber & many more

PQ-WireGuard (S&P 21)

- PQ variant of the VPN protocol
- Symbolic analysis using Tamarin
- Computational security proof
- Optimized implementation using Classic McEliece & Dagger

PQNoise (eprint)

- Generic AKE framework
- Using KEMs in place of DH
- Generic computational security proof for arbitrary patterns

QROM security

with Kathrin Hövelmanns & many more

- Practical schemes & protocols are mostly analyzed in ROM
- Considering quantum computers, we must move to quantum accessible ROM (QROM)
- Existing proof tools are still falling behind classical counterparts
- Recent results:
  - Adaptive reprogramming for the QROM [Asiacrypt 2021]
  - New One-way to hiding lemmas [TCC 2019 & eprint]

A note on QKD

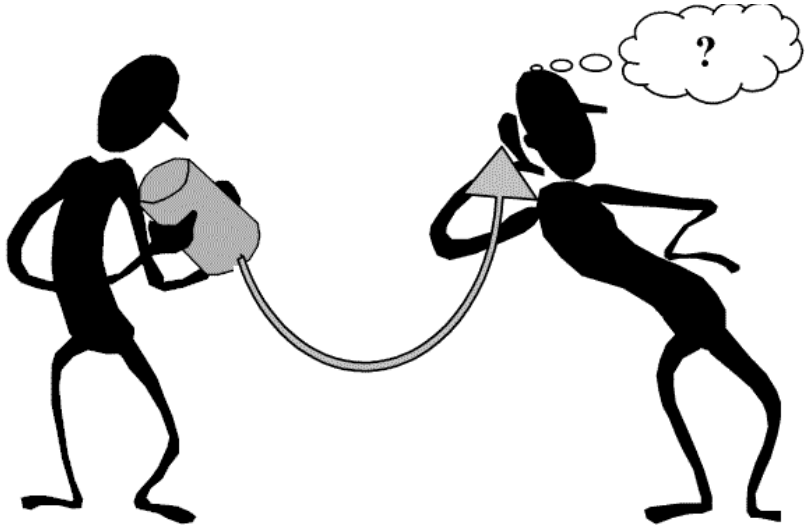
(or lasers are cool, but...)

- Quantum computers break our solutions for the key exchange problem:
  - two parties that never met and need a shared secret key
- QKD solves a different problem:
  - Two parties that already share a secret key can generate more shared secret keying material

Conclusion

- HBS are a suitable alternative if you
  - can handle state, or
  - can live with suboptimal performance
- There are more topics to solve
  - Verify our proofs
  - Make our protocols post-quantum secure
  - Develop the tools to do so

Thank you!
Questions?



For references & further literature see
https://huelsing.net/wordpress/?page_id=165