IMES

HASH-BASED SIGNATURE SCHEMES

Dorian Amiet, electrical engineer at IMES
TEMET Conference
About & Beyond PKI
11.06.2018

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Agenda

- **HSR, IMES & Securosys**
- **Quantum computer impact on today's cryptography**
- **Proposals for quantum-safe algorithms**
  - Categories
  - (Dis-) advantages of the proposed algorithms
- **Hash-based signatures**
  - Hash functions
  - OTS (one-time signature)
  - Merkle trees
  - SPHINCS-256

HSR & IMES

Inosuisse (KTI) Project

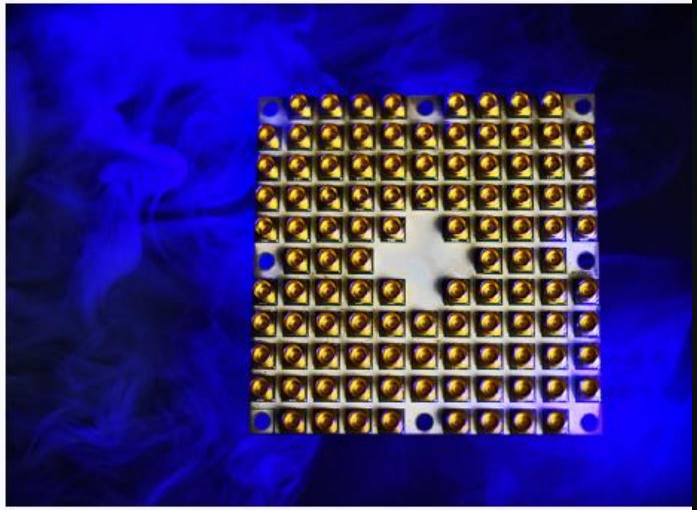
IMES INSTITUTE FOR MICROELECTRONICS AND EMBEDDED SYSTEMS

securosys

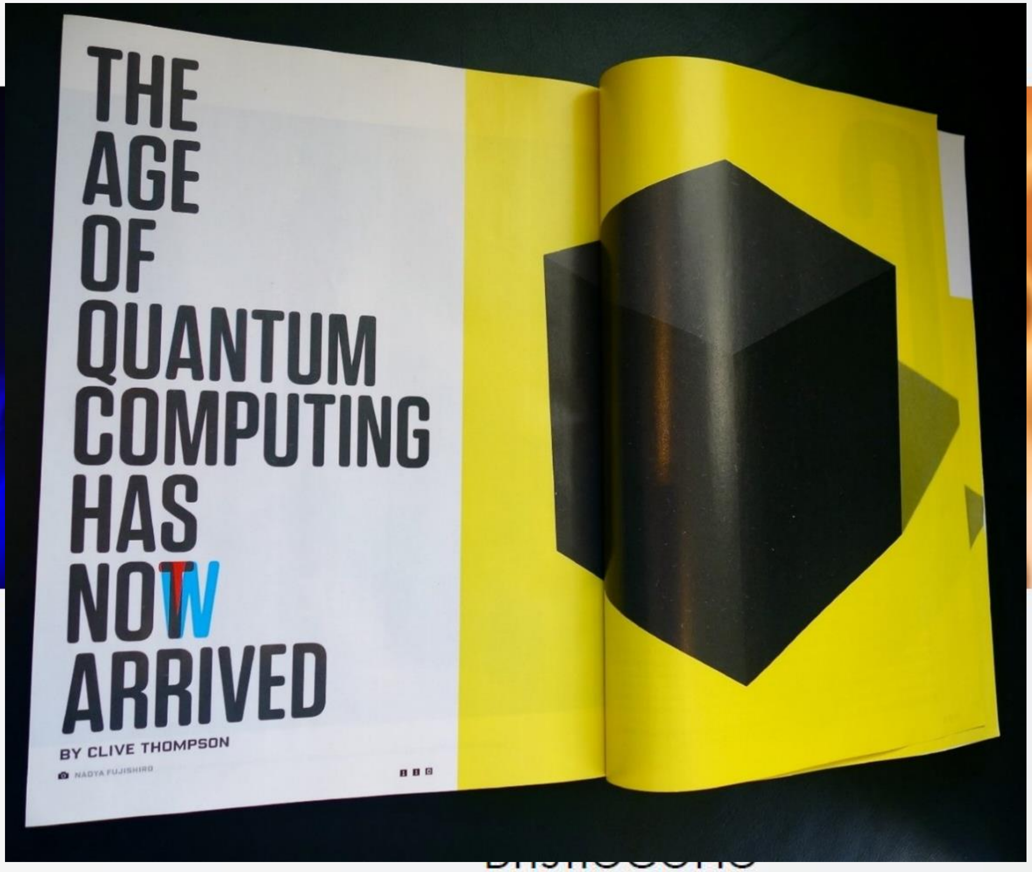
- **Analysis on proposed post-quantum algorithms**
- **Hardware (FPGA) implementation of some algorithms**
- **Implement post-quantum algorithms in Securosys HSM**

Quantum-computer-safe algorithms

Quantum Computer Progress



Intel’s 49-qubit chip
“Tangle-Lake”
January 2018



THE AGE OF QUANTUM COMPUTING HAS NOW ARRIVED

BY CLIVE THOMPSON

March 2018

IBM’s 50-qubit
quantum computer
November 2017

Impact on Current Algorithms [NISTIR]

Function	Algorithm	Key length/ Hash length (bits)	Security level (bits)		Quantum Algorithm
			Classical	Quantum	
PK: Signing, Key Exchange, Asymmetric Encryption	RSA-1024	1024	80	0	[Shor]
	RSA-2048	2048	112	0	[Shor]
	ECC-256	256	128	0	[Shor]
	ECC-512	512	256	0	[Shor]
Symmetric Encryption	AES-128	128	128	64	[Grover]
	AES-256	256	256	128	[Grover]
Hash	SHA256, SHA3-256	256	256	128 [Ber09]	[Grover]
	SHA384, SHA3-384	384	384	192 [Ber09]	[Grover]

Theorem [Mosca]

- ***X*: How much time to re-tool the existing infrastructure?**
- ***Y*: How long do you need your keys to be secure?**
- ***Z*: How long until large-scale quantum computer is built?**

- **Theorem [Mosca]: If X + Y > Z, then panic**

- **How big is Z?**
- **Mosca: 1/7 chance of breaking RSA-2048 by 2026 and a 1/2 chance by 2031**

Requirements for Post-Quantum Public-Key Algorithms

- **Security**
  - Reducible to NP-hard problems (=> no known fast attack)
  - Classifiable attack complexity
- **Efficiency comparable to RSA**
  - Size of keys and signatures
  - Processing time
  - Implementation complexity
    - Attacks on Implementations
    - Parameter choice
- **Usability**
  - Signing
  - Asymmetric encryption
  - Key exchange
  - Homomorphism

Lattice-Based Algorithms

Requirements

- **Security**
- **Efficiency comparable to RSA**
- **Implementation complexity**
- **Usability**

Lattice-based algorithms

- **Great usability**
  - Hash functions
  - Signing
  - Key exchange
  - Asymmetrical encryption
  - Homomorphism
- **Efficient processing**
  - Reasonable key sizes (<10KB)
  - >2000 op/s on a desktop processor
- **Doubt in cryptanalysis**
  - Many schemes and parameters
  - Hard to classify security

Code-Based Algorithms

Requirements

- **Security**
- **Efficiency comparable to RSA**
- **Implementation complexity**
- **Usability**

Code-based algorithms

- **Usability**
  - Signing
  - Asymmetrical encryption
  - Key exchange
- **Fast processing (1000 op/s)**
- **Fair cryptanalysis**
  - Security-levels somewhat predictable
- **Very big keys (>1MB)**

Hash-Based Algorithms

Requirements

- **Security**
- **Efficiency comparable to RSA**
- **Implementation complexity**
- **Usability**

Hash-based algorithms

- **Security very well analyzed and understood**
- **Small keys (<1KB)**
    - Fair signature sizes (<40KB)
- **Fair processing time (comparable to RSA)**
    - Fair signing (200 op/s)
    - Fast verification (>1000 op/s)
- **Signing only**
- **State-based**

Others

Requirements

- **Security**
- **Efficiency comparable to RSA**
- **Implementation complexity**
- **Usability**

Algorithms

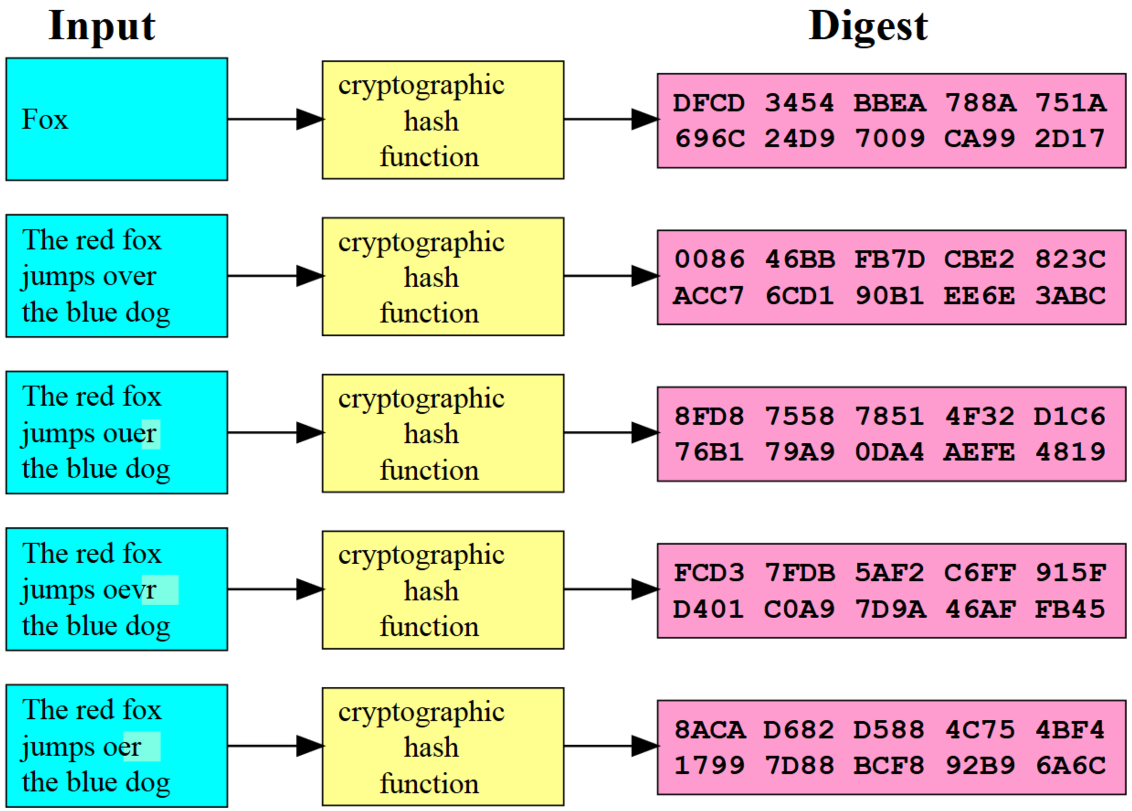
- **Multivariate-quadratic**
  - Efficient processing (>2000 op/s)
  - Small Signatures (<1KB)
  - Fair key sizes (50KB)
  - Very complex
  - Cryptanalysis is hard
- **Quantum-based**
  - Security based on quantum physics
  - Expensive and slow
  - No Signing

Summary on Signature Schemes

Type	Code	Lattice	Multivariate-quadratic	Hash	RSA	ECC
Operations/s	1000	>2000	>2000	200	200	1000
Key sizes	2 MB	7 KB	200KB	1KB	2KB	250 B
Signature sizes	500 B	6 KB	100 B	40 KB	2KB	500 B
Quantum security	+	?	?	+++	---	---
Functions	PK	PK and more	Signing (encryption)	Signing	PK	PK
Signing algorithm	[MCELIECE]	[BLISS]	[RAINBOW]	[SPHINCS]	[RSA]	[ECDSA]
Comments	Huge keys		Complex	Most conservative security	Broken by quantum computer	Broken by quantum computer

Cryptographic Hash Function

- **Input** *X* **is a bit-stream of arbitrary length**
- **Digest** *Y* = *h*(*X*) **has a fix size**
- **Fast computation:**
  - Find *Y*, given *X*
- **Hard Problems:**
  - Find *X*, given *Y*
  - Find X_2, such that $h(X_1) == h(X_2)$



Source: Wikipedia

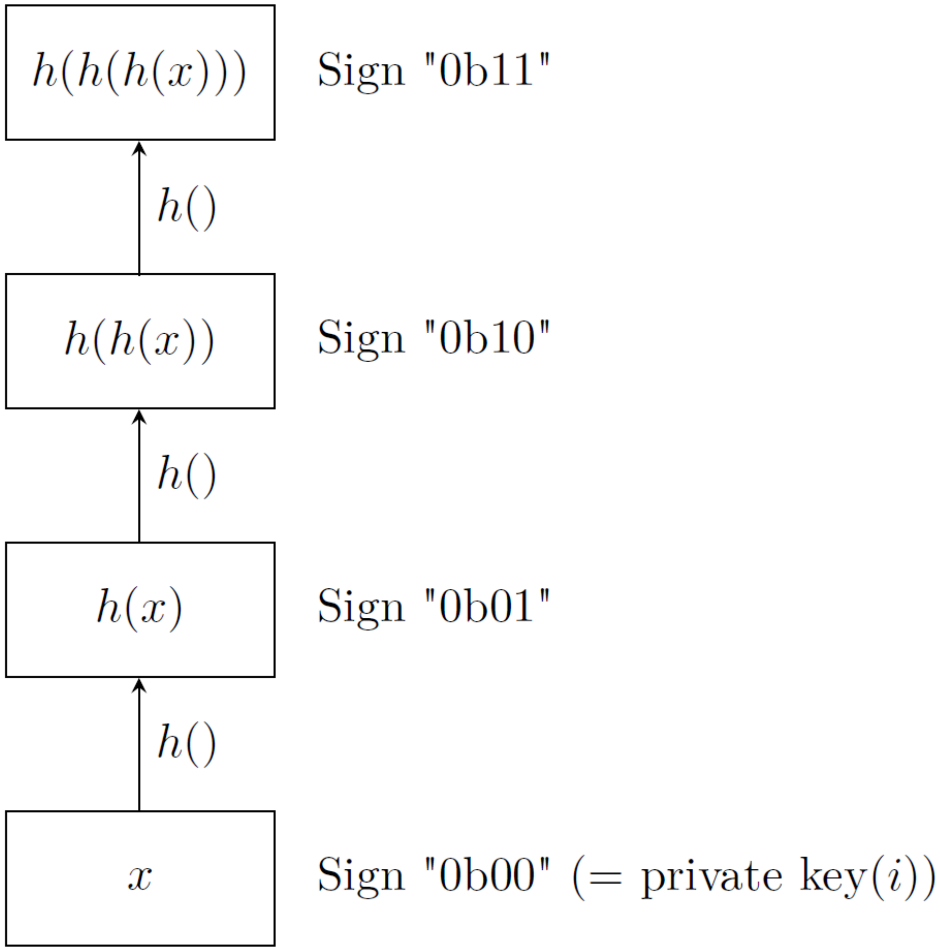
One-Time Signature (OTS)

Example: OTS with 256 bit security

1. **Generate 2x256 random numbers, each 256 bits**
   - $X_{0,0}$, $X_{0,1}$, $X_{2,0}$ …$X_{255,1}$
   - $X_{i,j}$ = private key
2. **Calculate all digests from random Numbers**
   - $Y_{0,0} = H(X_{0,0})$, $Y_{0,1} = H(X_{0,1})$,…,$Y_{255,1} = H(X_{255,1})$
   - $Y_{i,j}$ = public key
3. **Sign:**
   1. Calculate digest from message d = H(m)
   2. For i = 0 to 255
      1. If $d_i = 0$, then $\upsilon_i <= X_{i,0}$
      2. Else $\upsilon_i <= X_{i,1}$

PRN 0	H(PRN 0)	PRN 1	H(PRN 1)
$X_{0,0}$	$Y_{0,0}$	$X_{0,1}$	$Y_{0,1}$
$X_{1,0}$	$Y_{1,0}$	$X_{1,1}$	$Y_{1,1}$
$X_{2,0}$	$Y_{2,0}$	$X_{2,1}$	$Y_{2,1}$
$X_{\dots,0}$	$Y_{\dots,0}$	$X_{\dots,1}$	$Y_{\dots,1}$
$X_{255,0}$	$Y_{255,0}$	$X_{255,1}$	$Y_{255,1}$

W-OTS+ Shorter Signatures for Hash-Based Signature Schemes [WOTS]



$h(h(h(x)))$ Sign "0b11"

$h()$

$h(h(x))$ Sign "0b10"

$h()$

$h(x)$ Sign "0b01"

$h()$

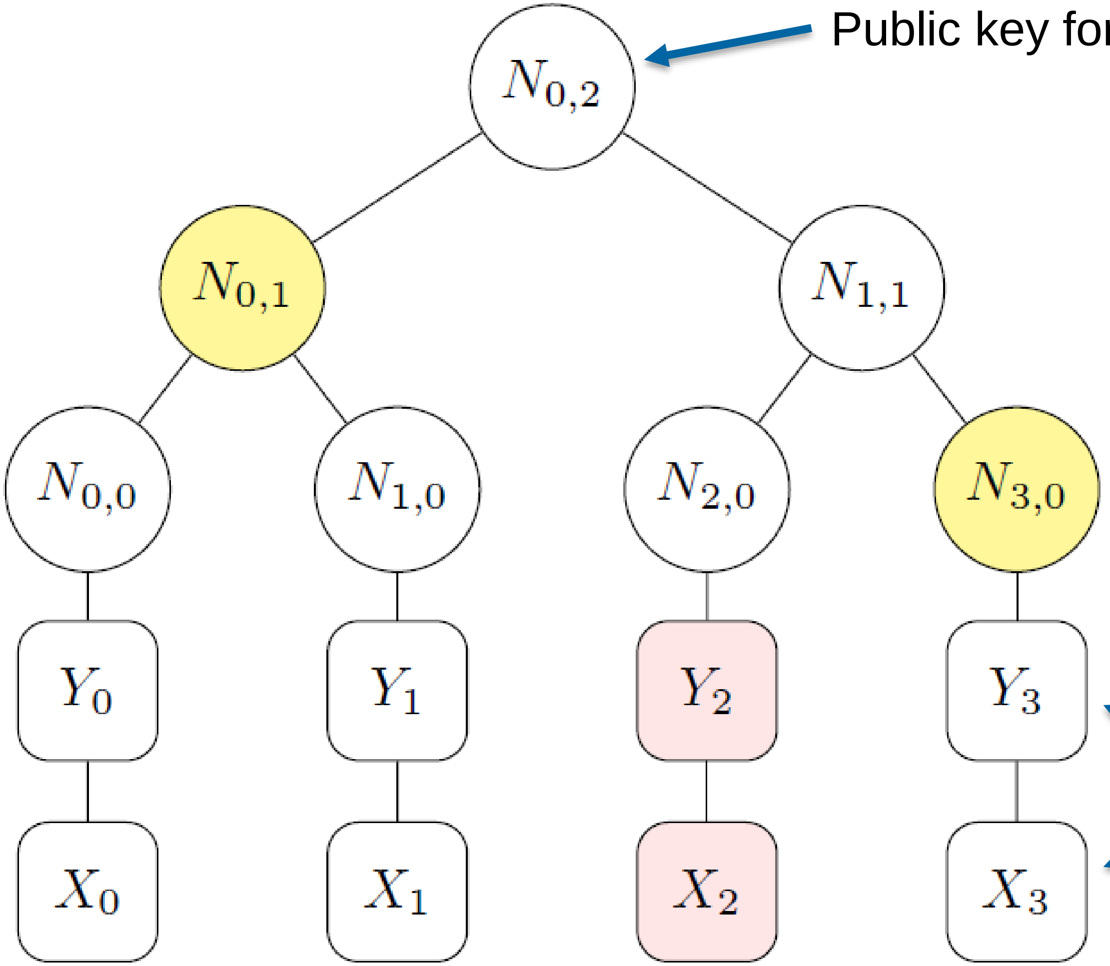
x Sign "0b00" $(=$ private key$(i))$

- **Sign a few bits per random number**
- **Needs a checksum**
- **Increases processing time**
- **Decreases key and signature sizes**

W-OTS+ Summary

- **Signature system which security is based only on security of hash function**
- **Quantum secure**
- **Very fast**
- **Only one signature per key pair!**

Merkle Tree



Public key for 4 signatures

4 OTS key pairs

Merkle Tree Summary

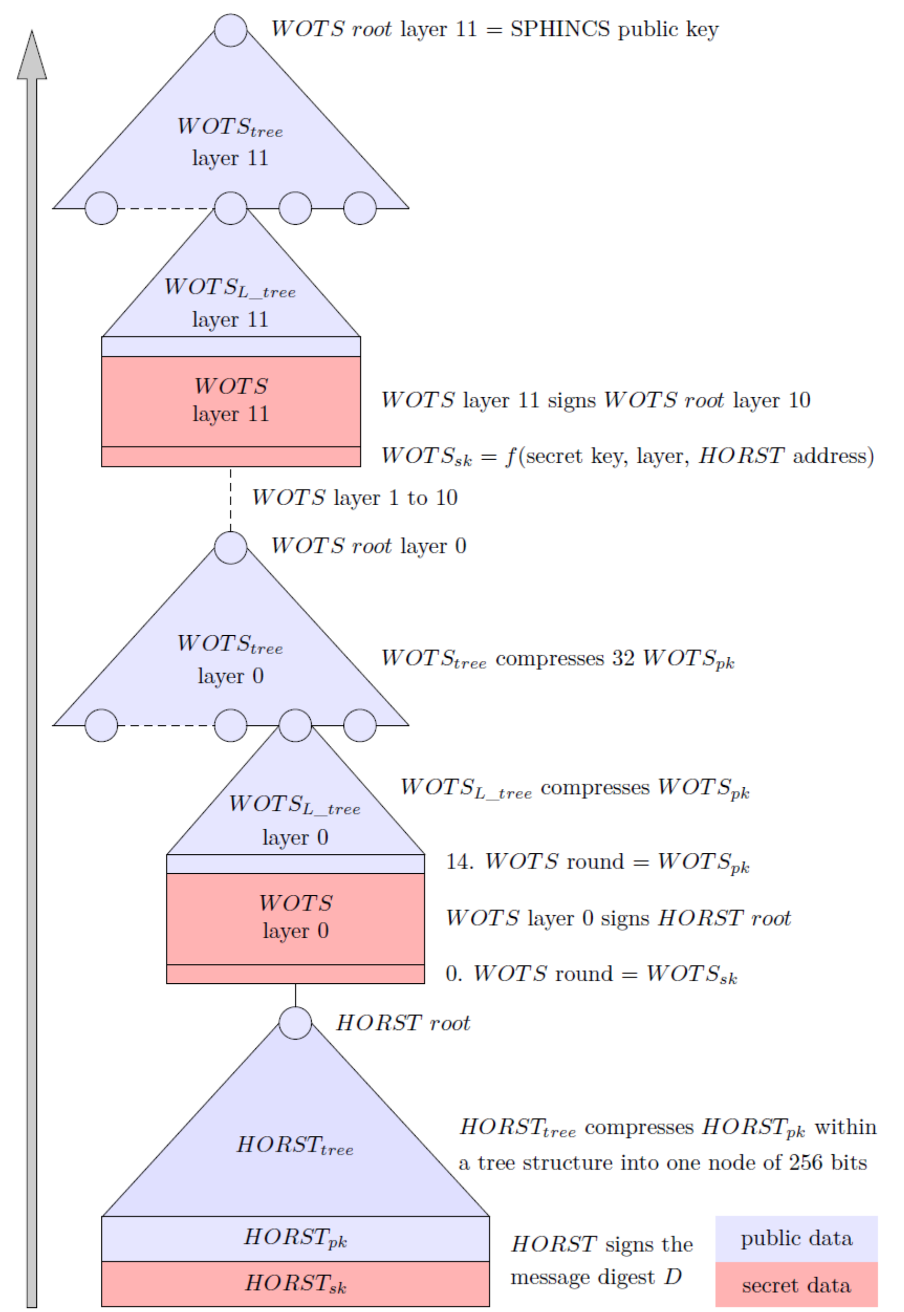
- **Signature system which security is based only on security of hash function**
- **Quantum secure**
- **Fast operations**
- **Problem: State-based**
  - Check-list required: Which OTS keys are already used?

Merkle Tree Evolution

- **Make a hyper-tree (tree of trees)**
  - Increase number of leaves (OTSs)
- **Use a FTS (few-time signature) at bottom layer instead of OTS**
- **Choose starting point at random**

=> Stateless, practical, hash-based, incredibly nice cryptographic signatures (SPHINCS)

SPHINCS-256



$WOTS$ $root$ layer 11 = SPHINCS public key

$WOTS$ layer 11 signs $WOTS$ $root$ layer 10

$WOTS_{sk} = f($secret key, layer, $HORST$ address$)$

$WOTS$ layer 1 to 10

$WOTS$ $root$ layer 0

$WOTS_{tree}$ compresses 32 $WOTS_{pk}$

$WOTS_{L\_tree}$ compresses $WOTS_{pk}$

14. $WOTS$ round = $WOTS_{pk}$

$WOTS$ layer 0 signs $HORST$ $root$

0. $WOTS$ round = $WOTS_{sk}$

$HORST$ $root$

$HORST_{tree}$ compresses $HORST_{pk}$ within a tree structure into one node of 256 bits

$HORST$ signs the message digest D

public data
secret data

Summary

- **Impact from quantum computer: public key cryptography**
- **There are some proposals to replace RSA and ECC**
  - Key and signature sizes may increase
  - Processing time may decreases
  - **Different algorithms for different tasks**
  - Protocols may change
- **SPHINCS-256 is a promising candidate to replace signature schemes**
  - Based on the security of hash functions
  - Stateless
  - FPGA Implementation: >600 sign/s, >15000 verifications/s
- **SPHINCS+ (SPHINCS-256 follower) is part of the NIST Post-Quantum Cryptography Standardization**

What can we do now?

- **PKI: Prepare for software/firmware updates, replace algorithms when standards are ready**
- **Already adopt post-quantum algorithms for cases where long-time security (>10 y) is required**
- **Contribute to the NIST post-quantum “not-contest” standardization**
- **Symmetric encryption: use 256 bit keys (e.g. AES-256)**
- **Hash functions: use hash lengths >= 256 bits**

- **Interested in Projects (including post-quantum security)?**

  => Contact us: https://www.imes.hsr.ch/

Thank You

[Shor] P.W. Shor. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. 1994

[Grover] L. K. Grover. A fast quantum mechanical algorithm for database search. 1996

[Mosca] M. Mosca. Cybersecurity in an era with quantum computers: will we be ready?. 2015

[Ber09] D. J. Bernstein. Cost analysis of hash collisions: Will quantum computers make SHARCS obsolete?. 2009

[NISTIR] NISTIR 8105 Report on Post-Quantum Cryptography. 2016, http://dx.doi.org/10.6028/NIST.IR.8105

[MiR09] D. Miccianacio and O. Regev. Lattice-based Cryptography in Post-Quantum Cryptography. 2009. ISBN: 978-3-540-88701-0

[XMSS] J. Buchmann, E. Dahmen, and A. Hülsing. XMSS - A Practical Forward Secure Signature Scheme based on Minimal Security Assumptions. 2011

[SPHINCS] D. J. Bernstein, D. Hopwood, A. Hülsing, T. Lange, R. Niederhagen, L. Papachristodoulou, M.Schneider, P. Schwabe, and Z. Wilcox-O'Hearn. SPHINCS: Practical Stateless Hash-Based Signatures. 2015

[BLISS] T. Pöppelmann, L. Ducas, and T. Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. 2014

[RSA] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. 1987

[MCELIECE] N. Courtois, M. Finiasz, N. Sendrier. How to achieve a McEliece-based digital signature scheme. 2001

[RAINBOW] J. Ding and D. Schmidt. Rainbow, a new multivariate polynomial signature scheme. 2005

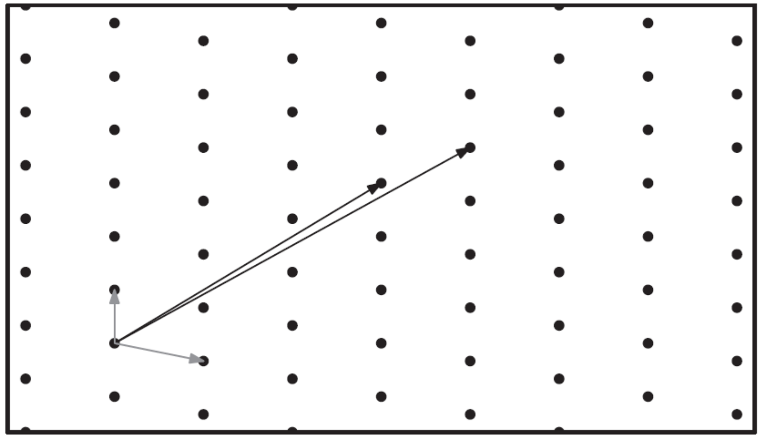
[ECDSA] G. Locke and P. Gallagher. "FIPS PUB 186-3 Digital Signature Standard" Federal Information Processing Standards Publication. vol. 3. 2009

[WOTS] A. Hülsing. W-OTS+ - Shorter Signatures for Hash-Based Signature Schemes. 2013

[AZC18] D. Amiet, A. Curiger and P. Zbinden. FPGA-based Accelerator for Post-Quantum Signature Scheme SPHINCS-256. 2018

Lattice-Based Algorithms

- **Mathematical problem: shortest vector, closest vector (SVP, CVP)**
- **Principle:**
  - Private key is a lattice with a “good” basis **B**
  - Public key is the same lattice given in a “bad” basis **H**
  - Encryption: encode plaintext using **H** => point **v** in the lattice => add error **r** => **v+r**
  - Decryption: Solve CVP using **B** => point **v** => decode => plaintext



Two possible bases in a two-dimensional lattice
Source: [MiR09]

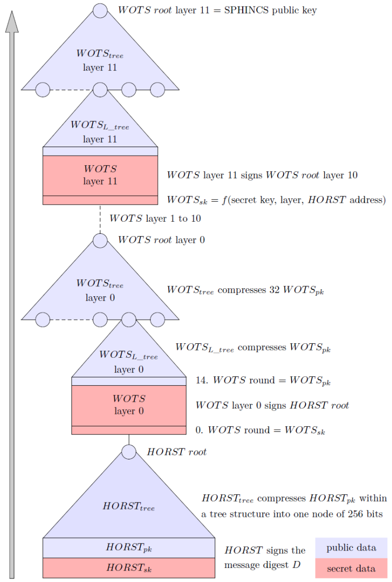
Code-Based Algorithms

- **Mathematical problem: Decoding a defective bit-stream**
- **Principle:**
  - Generator matrix ***G*** is hided by multiplication with permutation matrix ***P*** and encryption matrix ***S***
  - Random errors ***e*** are added during encryption
  - Efficient decryption is only possible with $\boldsymbol{G}$, $\boldsymbol{P^{-1}}$ and $\boldsymbol{S^{-1}}$
- **Public key: *G' (= SGP)***
- **Private key: *S, G, P***

Hash-Based Algorithms

- **Mathematical Problem: Find a hash function input to a given output (digest)**
  - Collision attack
  - Preimage attack
  - Brute-force (and birthday) attack
- **Private Key: Random data packets**
- **Public key: Digests of each data packets**
- **Signature: A selection of the random data packets**

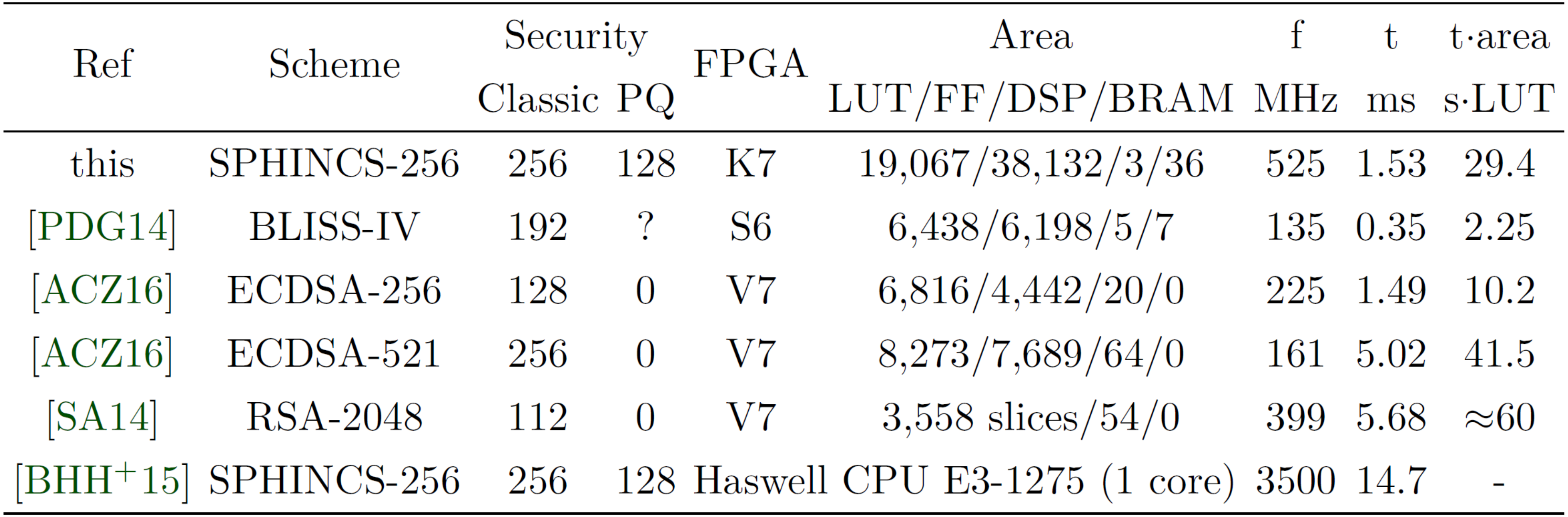
SPHINCS-256 Implementation



Function	Signing					Verification
Part	Start	HORST	12·WOTS	Overhead	**Total**	Total
BLAKE-256	0	1	384	12	**397**	0
ChaCha12	0	32,768	13,056	408	**46,232**	0
π_{ChaCha}	0	19,3410	437,352	≈9,000	**≈640,000**	≈9,000
BLAKE-512	2	0	0	0	**2**	1

Source: [AZC18]

Implementation Results



Ref	Scheme	Security		FPGA	Area	f	t	t·area
		Classic	PQ		LUT/FF/DSP/BRAM	MHz	ms	s·LUT
this	SPHINCS-256	256	128	K7	19,067/38,132/3/36	525	1.53	29.4
[PDG14]	BLISS-IV	192	?	S6	6,438/6,198/5/7	135	0.35	2.25
[ACZ16]	ECDSA-256	128	0	V7	6,816/4,442/20/0	225	1.49	10.2
[ACZ16]	ECDSA-521	256	0	V7	8,273/7,689/64/0	161	5.02	41.5
[SA14]	RSA-2048	112	0	V7	3,558 slices/54/0	399	5.68	≈60
[BHH$^+$15]	SPHINCS-256	256	128	Haswell CPU E3-1275 (1 core)		3500	14.7	-

Simple Power Analysis