Putting wings on SPHINCS

PQCRYPTO Conference

Stefan Kölbl

April 10th, 2018

Technical University of Denmark,
Cybercrypt

SPHINCS

SPHINCS

- Hash-based signature scheme
- **Stateless**
- 128-bit post-quantum security
- Sizes:
  - Public Key: 1KB
  - Secret Key: 1KB
  - Signature: 41KB

`https://sphincs.cr.yp.to/`

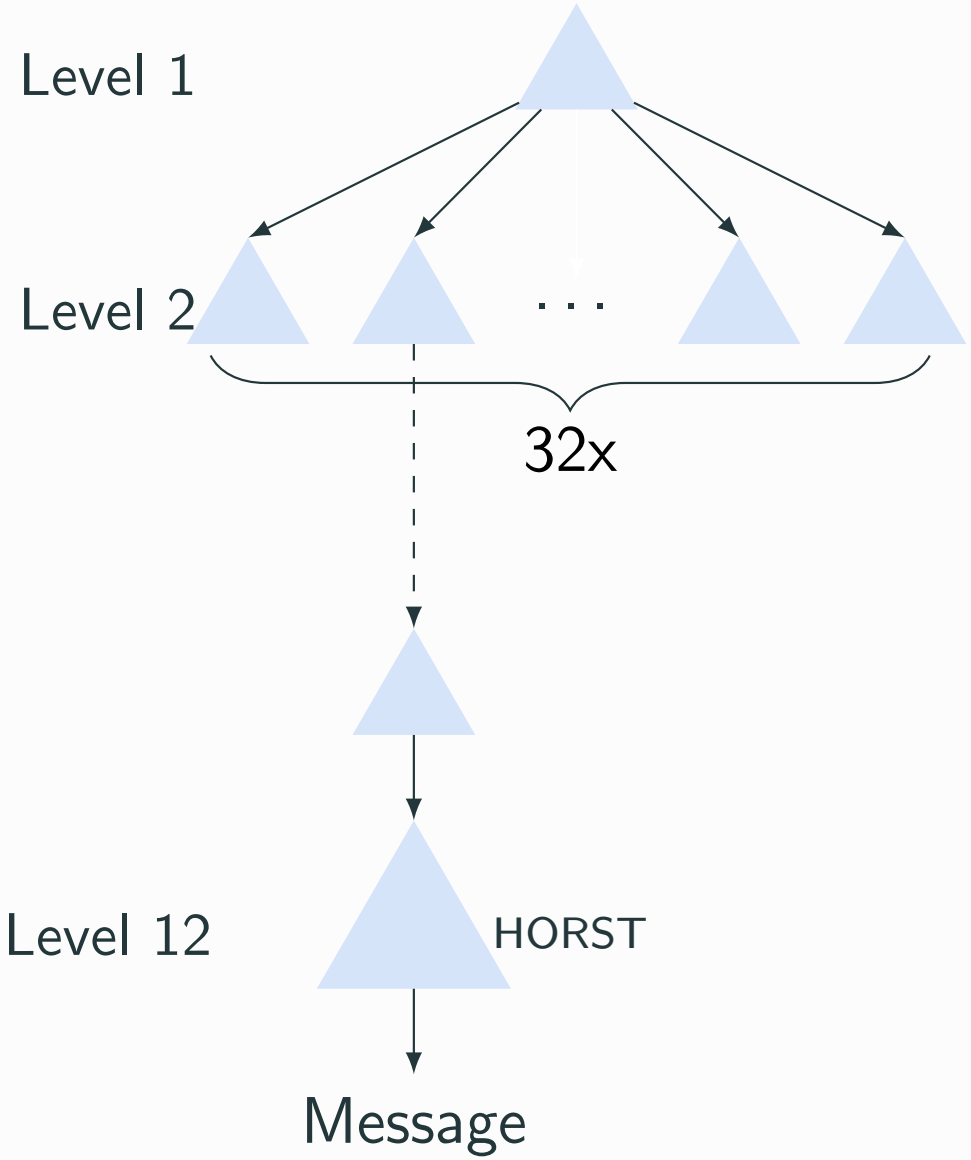
How to instantiate SPHINCS?

SPHINCS

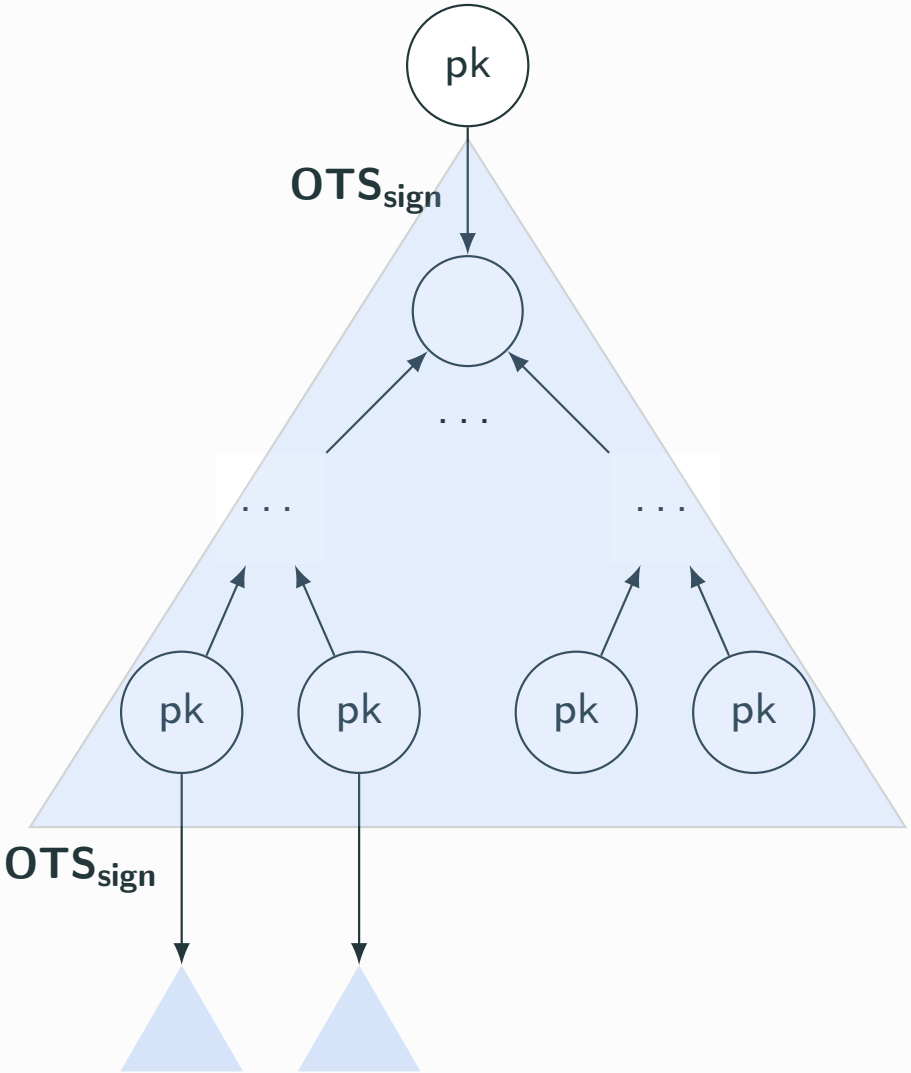
Main components:

- One-time Signature (WOTS)
- Few-time Signature (HORST)
- Merkle-Tree

SPHINCS



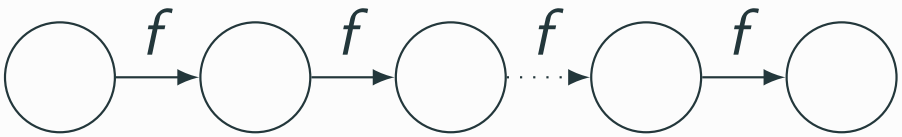
SPHINCS

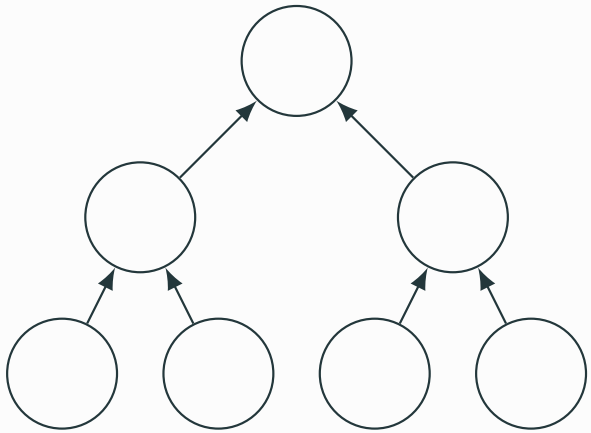


SPHINCS

What is computed?

- Many calls to a hash function...
- ...but using short input only.

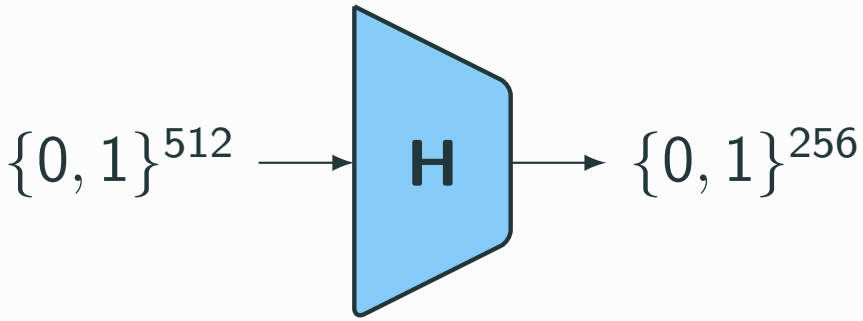


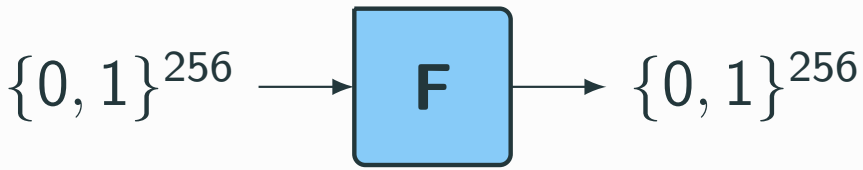


SPHINCS

For one signature

- ≈450.000 times **F**
- ≈90.000 times **H**





Cryptographic Hash Functions

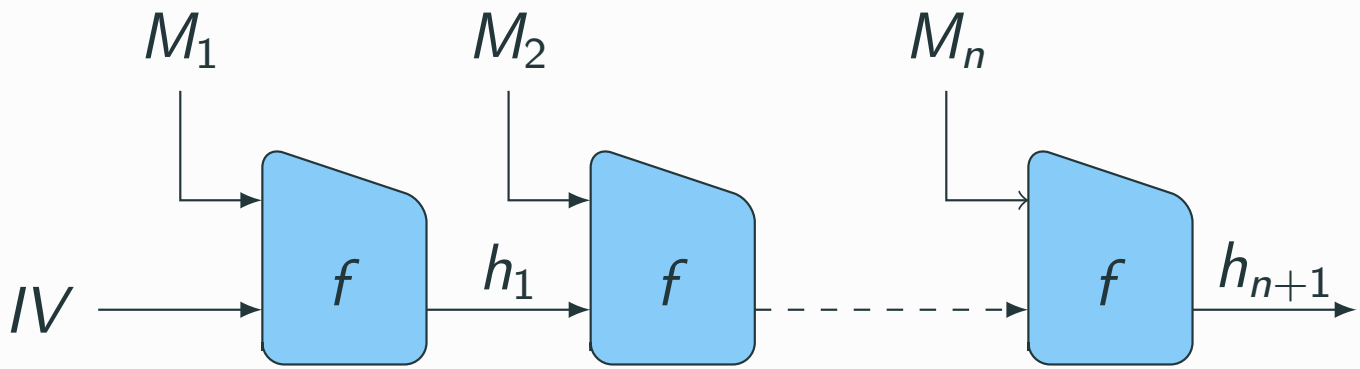
Which hash function could we use?

- Standards
  - SHA256
  - SHA-3
- ChaCha12 permutation
- Keccak
- Haraka
- Simpira

Cryptographic Hash Functions

SHA-2 (FIPS PUB 180-4)

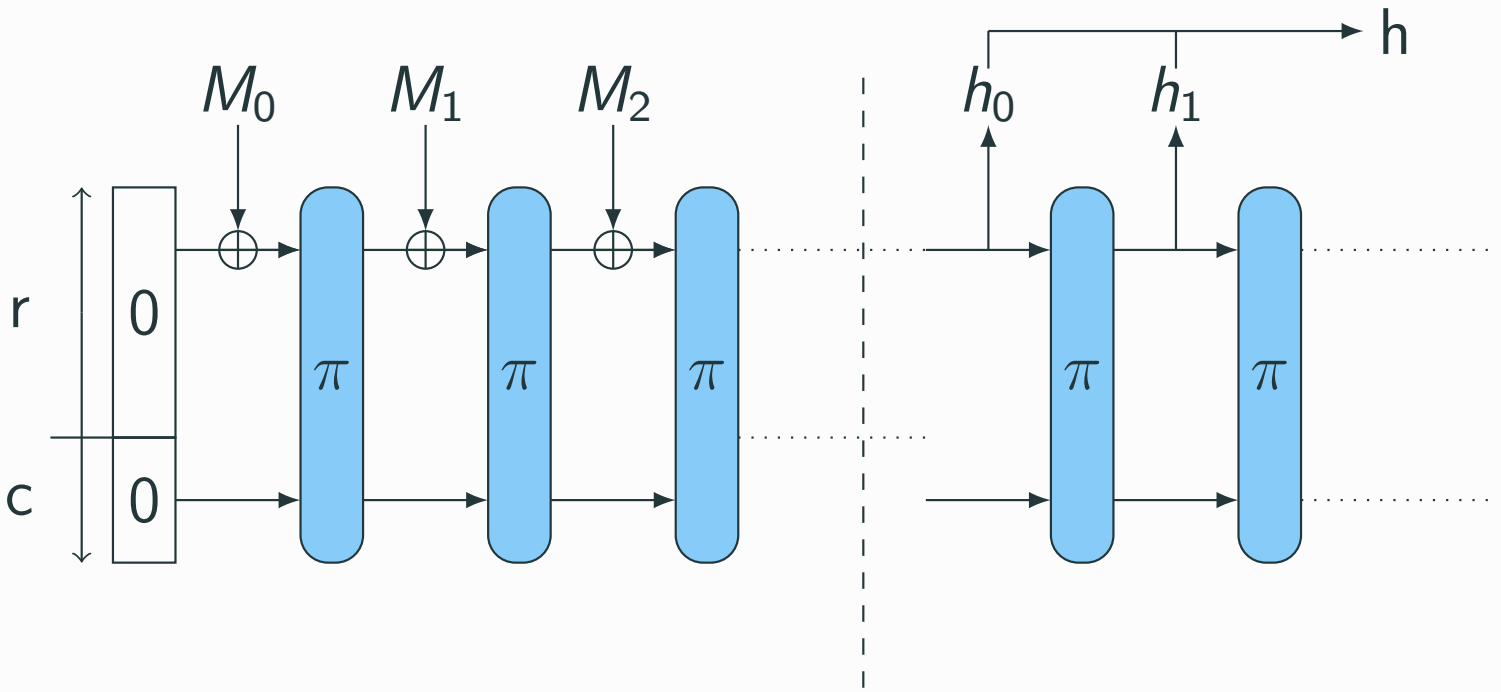
- 512-bit Message Blocks
- Padding...



Cryptographic Hash Functions

SHA3-256 (FIPS PUB 202)

- 1600-bit Permutation
- 1088-bit Message Blocks



Cryptographic Hash Functions

Other Keccak variants:

- Use 800-bit permutation?
- Use less rounds (Kangaroo12[1]).
- Best preimage attack on 4 rounds[2].

[0] see `https://eprint.iacr.org/2016/770`

[0] Linear Structures: Applications to Cryptanalysis of Round-Reduced Keccak, Asiacrypt 2016

Cryptographic Hash Functions

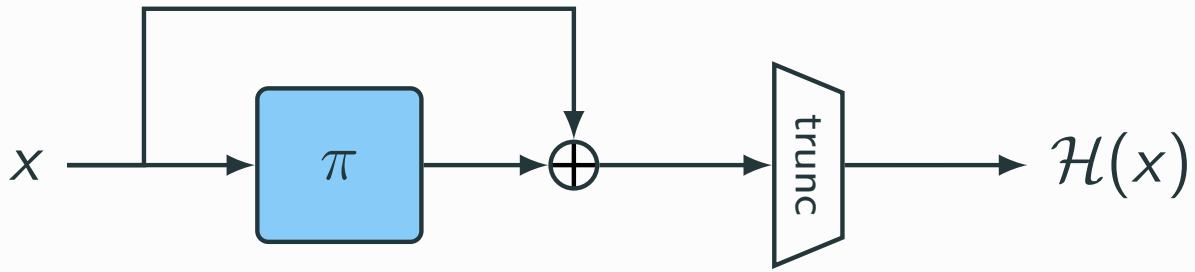
ChaCha12

- Suggested in SPHINCS paper.
- Use ChaCha12 permutation in sponge.
- Great software performance with vectorization.

Cryptographic Hash Functions

Haraka: A short-input hash function[3]

- Permutation based on AES rounds.
- SPN construction.
- 256- and 512-bit permutation.

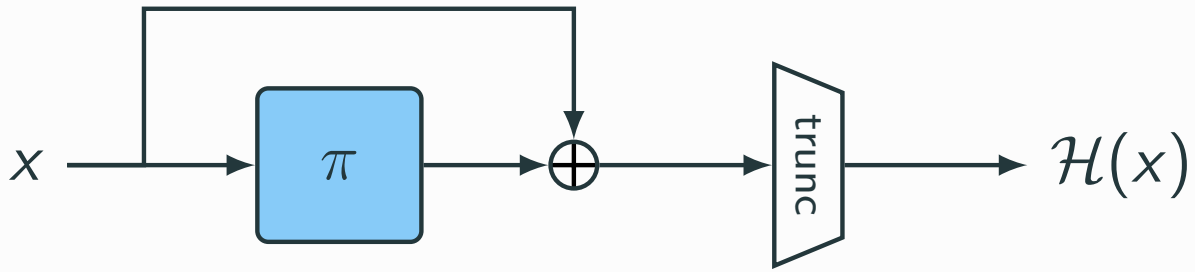


[3] https://eprint.iacr.org/2016/098

Cryptographic Hash Functions

Simpira[4]

- Permutation based on AES rounds.
- Feistel construction.
- 256- and 512-bit permutation.



[4] https://eprint.iacr.org/2016/122

Microarchitectures

SPHINCS not well suited for small devices[5]

- Signature size larger than RAM for some devices.
- Computational costs for signing high...
- ... but verification is cheap.

Focus on highend platforms:

- Intel Haswell/Skylake, AMD Ryzen
- ARM Cortex A57/A72

[5] see https://eprint.iacr.org/2015/1042

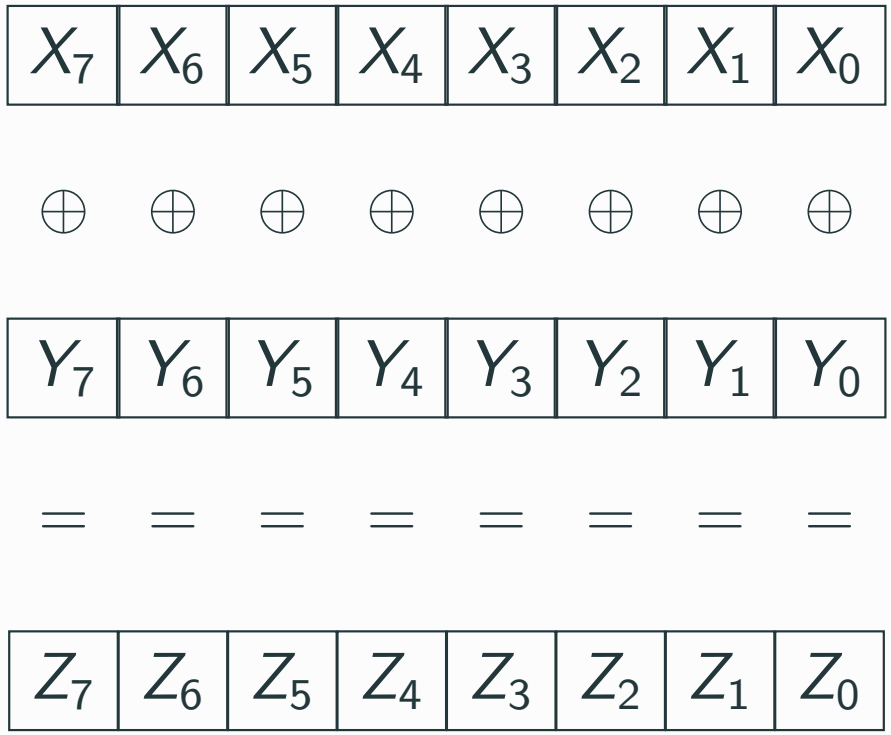
Microarchitectures

How to get a fast implementation?

- Vectorization (AVX2, NEON, AVX-512)
- Hardware Support (AES, SHA-2, SHA-3)
- Utilize pipeline

Microarchitectures

Vector Instructions



- Apply same operation on all elements of the vector.
- Use independet inputs.

Microarchitectures

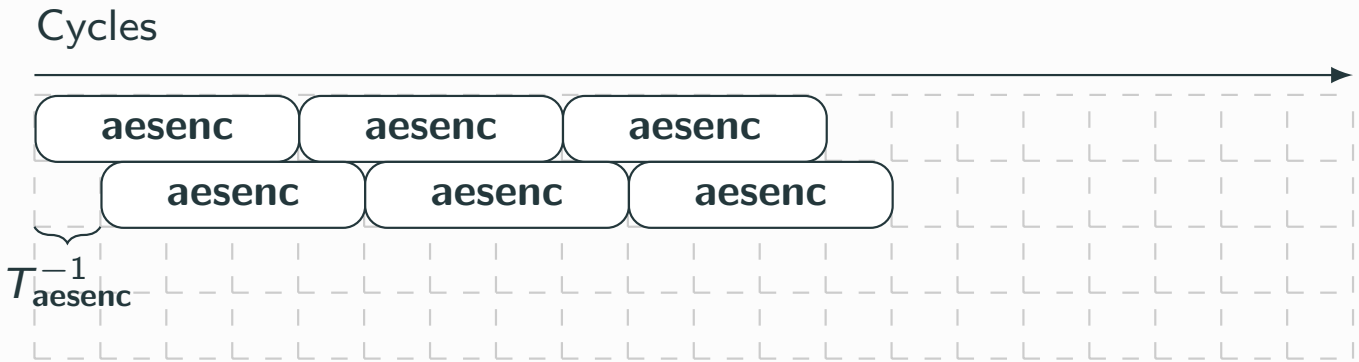
Pipelining

- Latency
- Inverse Throughput

Microarchitectures

Pipelining

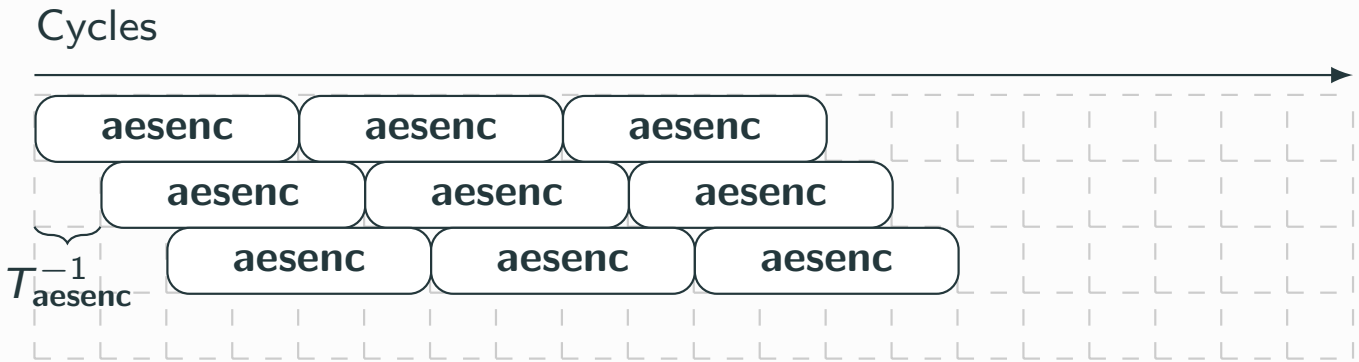
- Latency
- Inverse Throughput



Microarchitectures

Pipelining

- Latency
- Inverse Throughput



Platforms

Performance varies a lot depending on the platform

Platform	**Instruction**	Latency	inv. Throughput
Skylake	vectorized XOR	1	0.33
Ryzen	vectorized XOR	1	0.5
Cortex A57	vectorized XOR	3	2

Implementations

How to implement those functions efficiently?

- SHA-2
- Keccak[$b = 800$]
- ChaCha12
- Haraka
- Simpira

Implementations

How to implement those functions efficiently?

- SHA-2
  - 32-bit word oriented
  - Vectorize
  - Hardware Support
- Keccak[$b = 800$]
- ChaCha12
- Haraka
- Simpira

Implementations

How to implement those functions efficiently?

- SHA-2
- Keccak[$b = 800$]
  - 32-bit word oriented
  - Vectorize
- ChaCha12
- Haraka
- Simpira

Implementations

How to implement those functions efficiently?

- SHA-2
- Keccak[$b = 800$]
- ChaCha12
  - 32-bit word oriented
  - Vectorize
- Haraka
- Simpira

Implementations

How to implement those functions efficiently?

- SHA-2
- Keccak[$b = 800$]
- ChaCha12
- Haraka
  - AES + permute
- Simpira

Implementations

How to implement those functions efficiently?

- SHA-2
- Keccak[$b = 800$]
- ChaCha12
- Haraka
- Simpira
    - AES

Tour de SPHINCS

Tour de SPHINCS

Intel Skylake

- AVX2 (256-bit vector)
- AES-NI

Tour de SPHINCS

Intel Skylake

- AVX2 (256-bit vector)
- AES-NI

Signing (million cycles)

Design	**Skylake**
ChaCha12	
Haraka	
Keccak	
SHA-256	
Simpira	

Tour de SPHINCS

Intel Skylake

- AVX2 (256-bit vector)
- AES-NI

Signing (million cycles)

Design	**Skylake**
ChaCha12	
Haraka	
Keccak	
SHA-256	142.06
Simpira	

Tour de SPHINCS

Intel Skylake

- AVX2 (256-bit vector)
- AES-NI

Signing (million cycles)	
Design	**Skylake**
ChaCha12	
Haraka	
Keccak	108.62
SHA-256	142.06
Simpira	

Tour de SPHINCS

Intel Skylake

- AVX2 (256-bit vector)
- AES-NI

Signing (million cycles)

Design	**Skylake**
ChaCha12	43.49
Haraka	
Keccak	108.62
SHA-256	142.06
Simpira	

Tour de SPHINCS

Intel Skylake

- AVX2 (256-bit vector)
- AES-NI

Signing (million cycles)

Design	**Skylake**
ChaCha12	43.49
Haraka	
Keccak	108.62
SHA-256	142.06
Simpira	28.40

Tour de SPHINCS

Intel Skylake

- AVX2 (256-bit vector)
- AES-NI

Signing (million cycles)

Design	**Skylake**
ChaCha12	43.49
Haraka	20.78
Keccak	108.62
SHA-256	142.06
Simpira	28.40

Tour de SPHINCS

AMD Ryzen

- AVX2 (256-bit vector)
- AES-NI (2 ports)
- SHA256 instructions

Tour de SPHINCS

AMD Ryzen

- AVX2 (256-bit vector)
- AES-NI (2 ports)
- SHA256 instructions

Signing (million cycles)

Design	**Ryzen**
ChaCha12	
Haraka	
Keccak	
SHA-256	
Simpira	

Tour de SPHINCS

AMD Ryzen

- AVX2 (256-bit vector)
- AES-NI (2 ports)
- SHA256 instructions

Signing (million cycles)

Design	**Ryzen**
ChaCha12	
Haraka	
Keccak	189.98
SHA-256	
Simpira	

Tour de SPHINCS

AMD Ryzen

- AVX2 (256-bit vector)
- AES-NI (2 ports)
- SHA256 instructions

Signing (million cycles)

Design	**Ryzen**
ChaCha12	63.42
Haraka	
Keccak	189.98
SHA-256	
Simpira	

Tour de SPHINCS

AMD Ryzen

- AVX2 (256-bit vector)
- AES-NI (2 ports)
- SHA256 instructions

Signing (million cycles)

Design	**Ryzen**
ChaCha12	63.42
Haraka	
Keccak	189.98
SHA-256	53.33
Simpira	

Tour de SPHINCS

AMD Ryzen

- AVX2 (256-bit vector)
- AES-NI (2 ports)
- SHA256 instructions

Signing (million cycles)

Design	**Ryzen**
ChaCha12	63.42
Haraka	
Keccak	189.98
SHA-256	53.33
Simpira	20.43

Tour de SPHINCS

AMD Ryzen

- AVX2 (256-bit vector)
- AES-NI (2 ports)
- SHA256 instructions

Signing (million cycles)

Design	**Ryzen**
ChaCha12	63.42
Haraka	15.54
Keccak	189.98
SHA-256	53.33
Simpira	20.43

Tour de SPHINCS

ARM Cortex A57

- NEON (128-bit vector)
- AES
- SHA256 support

Tour de SPHINCS

ARM Cortex A57

- NEON (128-bit vector)
- AES
- SHA256 support

Signing (million cycles)	
Design	**Cortex A57**
ChaCha12	
Haraka	
Keccak	
SHA-256	
Simpira	

Tour de SPHINCS

ARM Cortex A57

- NEON (128-bit vector)
- AES
- SHA256 support

Signing (million cycles)

Design	**Cortex A57**
ChaCha12	
Haraka	
Keccak	376.90
SHA-256	
Simpira	

Tour de SPHINCS

ARM Cortex A57

- NEON (128-bit vector)
- AES
- SHA256 support

Signing (million cycles)	
Design	**Cortex A57**
ChaCha12	193.51
Haraka	
Keccak	376.90
SHA-256	
Simpira	

Tour de SPHINCS

ARM Cortex A57

- NEON (128-bit vector)
- AES
- SHA256 support

Signing (million cycles)

Design	**Cortex A57**
ChaCha12	193.51
Haraka	
Keccak	376.90
SHA-256	92.08
Simpira	

Tour de SPHINCS

ARM Cortex A57

- NEON (128-bit vector)
- AES
- SHA256 support

Signing (million cycles)

Design	**Cortex A57**
ChaCha12	193.51
Haraka	
Keccak	376.90
SHA-256	92.08
Simpira	63.48

Tour de SPHINCS

ARM Cortex A57

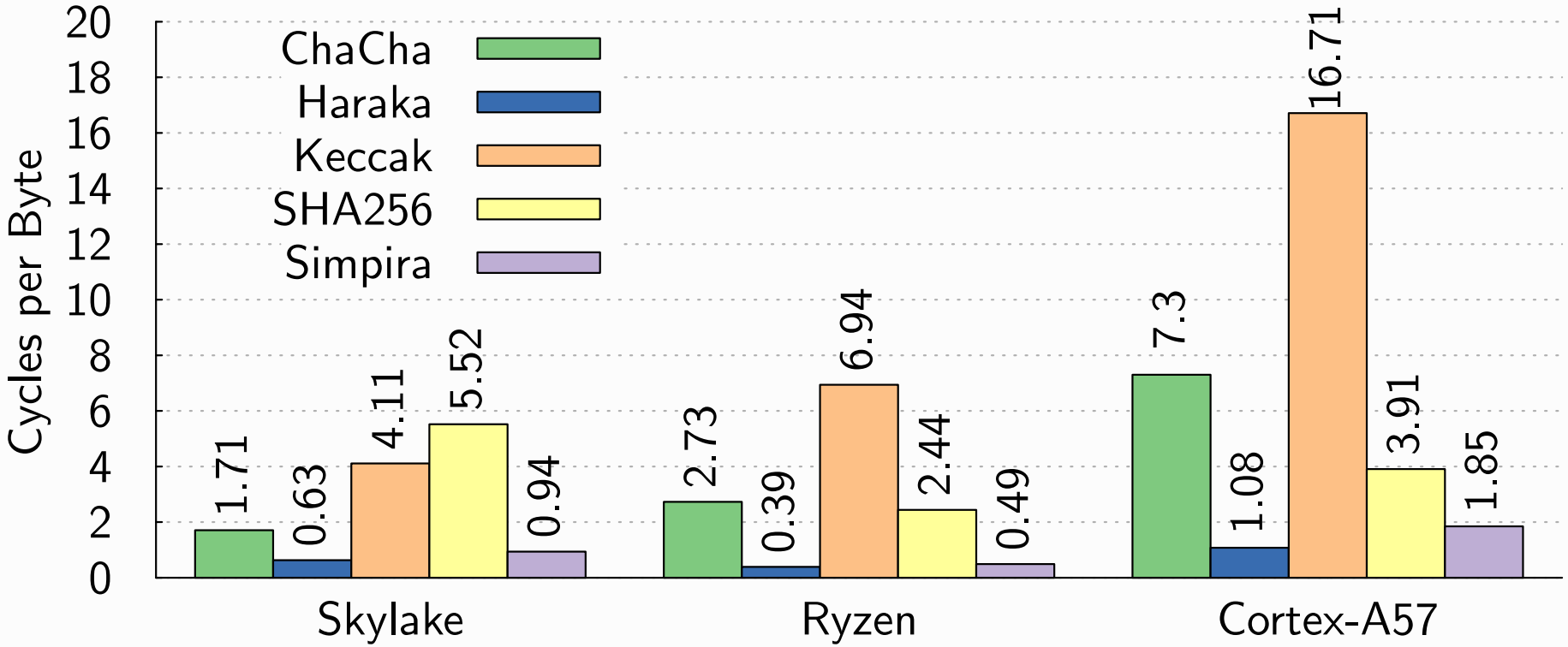
- NEON (128-bit vector)
- AES
- SHA256 support

Signing (million cycles)

Design	**Cortex A57**
ChaCha12	193.51
Haraka	47.10
Keccak	376.90
SHA-256	92.08
Simpira	63.48

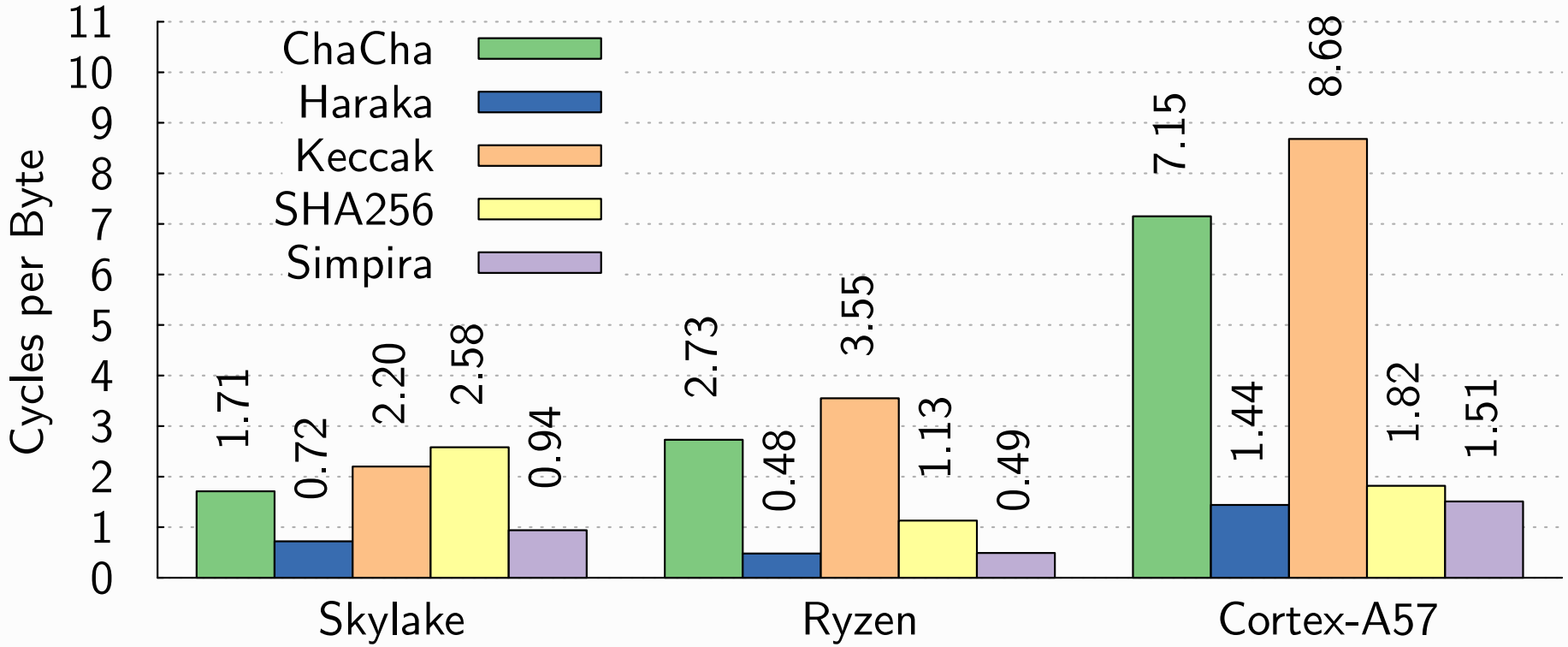
Formula SPHINCS

Hash Performance for **F**



Formula SPHINCS

Hash Performance for **H**



NIST PQ Competition

Two variants of SPHINCS in NIST PQ competition:

- Gravity-SPHINCS
  - Results directly apply.
  - Already uses Haraka.
- SPHINCS+
  - Tweakable Hash.
  - Needs to process slightly larger inputs.

Conclusion

Summary

- Gap between fastest and slowest up to 10x.
- Verification can be really fast, e.g. 258.660 cycles on Ryzen.

Future Platforms:

- Larger Vectors (AVX512)
- Vectorized AES (Intel Icelake)
- SHA-3 instructions (ARMv8.4-a)

Questions?

https://github.com/kste/sphincs