TECHNISCHE UNIVERSITÄT DRESDEN

A Latency-optimized Hash-based Digital Signature Accelerator for the Tactile Internet

Tensilica University Day 2019

Robert Wittig / Technische Universität Dresden

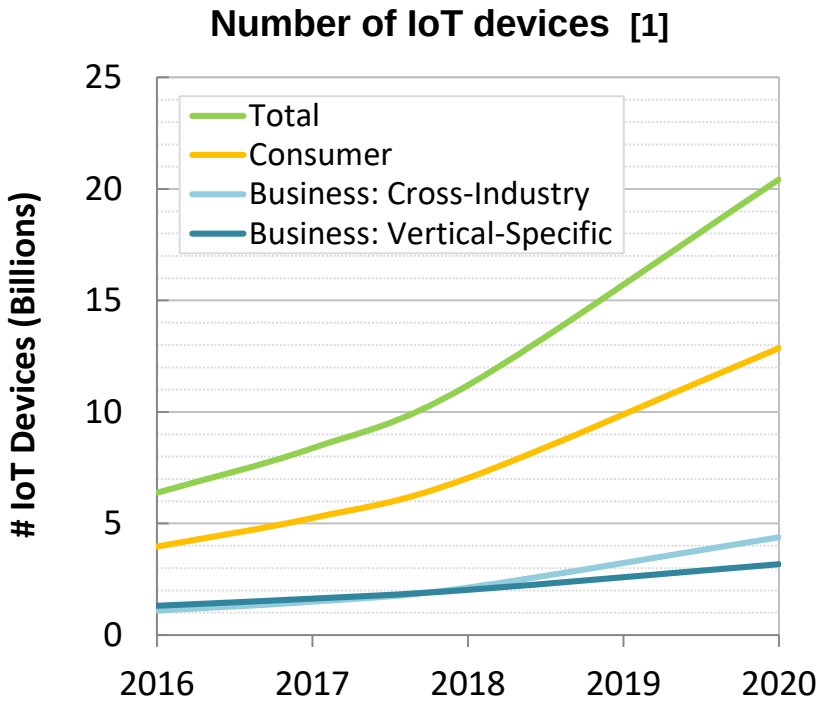
Friedrich Pauls / Barkhausen Institut

Pauls, F., Wittig, R., Matus, E., & Fettweis, G. (2019). A Latency-optimized Hash-based Digital Signature Accelerator for the Tactile Internet. In *SAMOS*. Retrieved from https://link.springer.com/chapter/10.1007/978-3-030-27562-4_7

Tactile Internet

- Tactile Internet as evolution of Internet of Things (IoT)
- Enables real-time interactive systems in areas such as
  - Automation
  - Transportation
  - Gaming
  - Education
  - Healthcare

- Key requirements
  - Extremely high availability and reliability
  - Strong security
    - Encryption
    - Data authentication
  - Ultra-low latency
    - 1 ms



[1] https://www.gartner.com/en/newsroom/press-releases/2017-02-07-gartner-says-8-billion-connected-things-will-be-in-use-in-2017-up-31-percent-from-2016

The Global Security Threat

The New York Times

Cyberattack Shows Vulnerability of Gas Pipeline Network

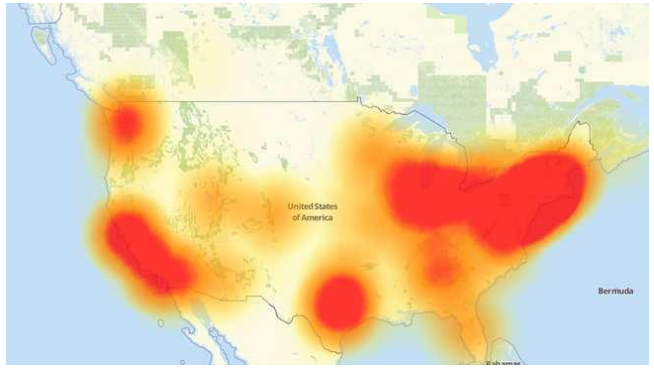
April 4, 2018

International Business Times

"New botnet found targeting billions of ARC-based IoT devices worldwide"

Jan 17, 2018

"Blame the Internet of Things for Destroying the Internet Today"



Oct 21, 2016

Bloomberg

Technology

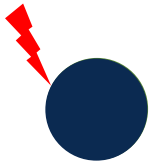
The Cyberattack That Crippled Gas Pipelines Is Now Hitting Another Industry

April 5, 2018

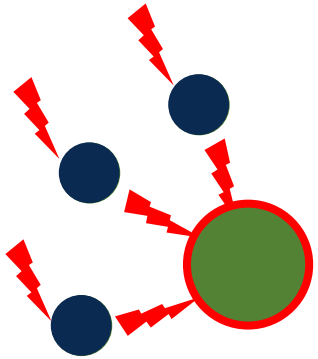
[1] https://www.bloomberg.com/news/articles/2018-04-03/day-after-cyber-attack-a-third-gas-pipeline-data-system-shuts
[2] https://www.ibtimes.co.uk/what-mirai-okiru-new-botnet-found-targeting-billions-arc-based-iot-devices-worldwide-1655491
[3] https://www.nytimes.com/2018/04/04/business/energy-environment/pipeline-cyberattack.html
[4] https://motherboard.vice.com/en_us/article/vv7xg9/blame-the-internet-of-things-for-destroying-the-internet-today

Security Considerations

- Security vulnerabilities → Dangerous consequences:
  - Direct attack:



  - Indirect attack:



- Strong security of future devices is a must!
  - Encryption (important, may be optional)
  - Data Authentication (essential for security)
    → Provided by digital signatures

Digital signatures widely used today

- Digital signatures are fundamental building block for:
  - Banking, E-Mail, Online Shopping, Social Media, Cloud Storage, …
  - Software updates
  - Sim card, TPM
  - V2X, IoT

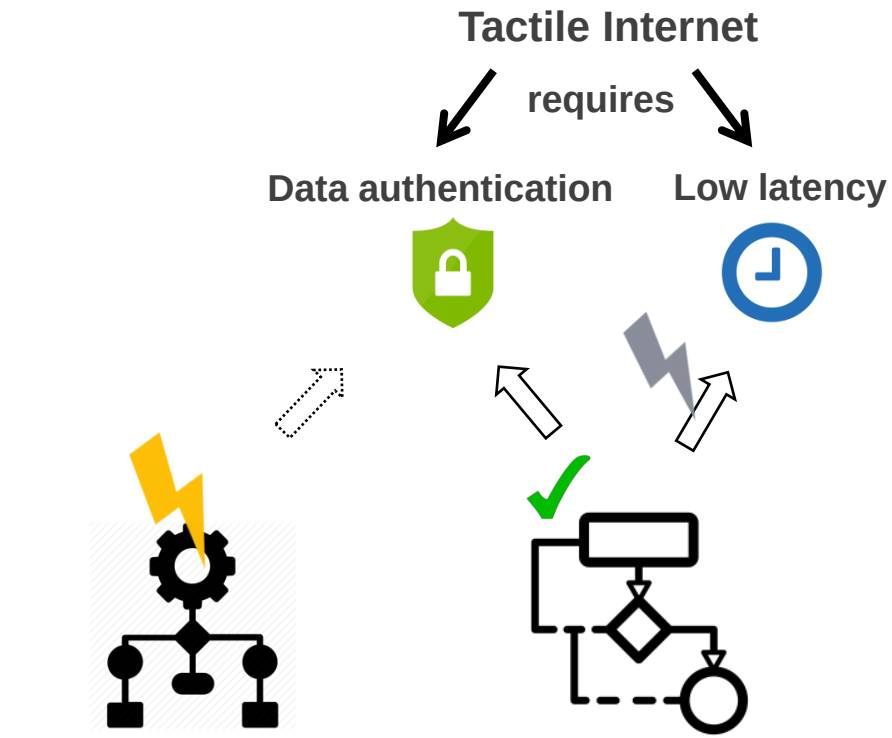
- Digital Signatures used today are based on problems which are (currently) hard to solve:
  - Factorization (RSA)
  - Discrete Logarithm (Elliptic Curve Cryptography)

- Both algorithms not suitable for post-quantum cryptography [Shor ’94]

- Quantum computers in development all over the world (e.g. Intel, IBM, Microsoft, Google)
  → Unknown when powerful QC will become available; > 10 years
  → Once they are there, most of our current security primitives will become useless

- Strong need for post-quantum digital signatures

Research Objective



Quantum Computers break current signatures schemes

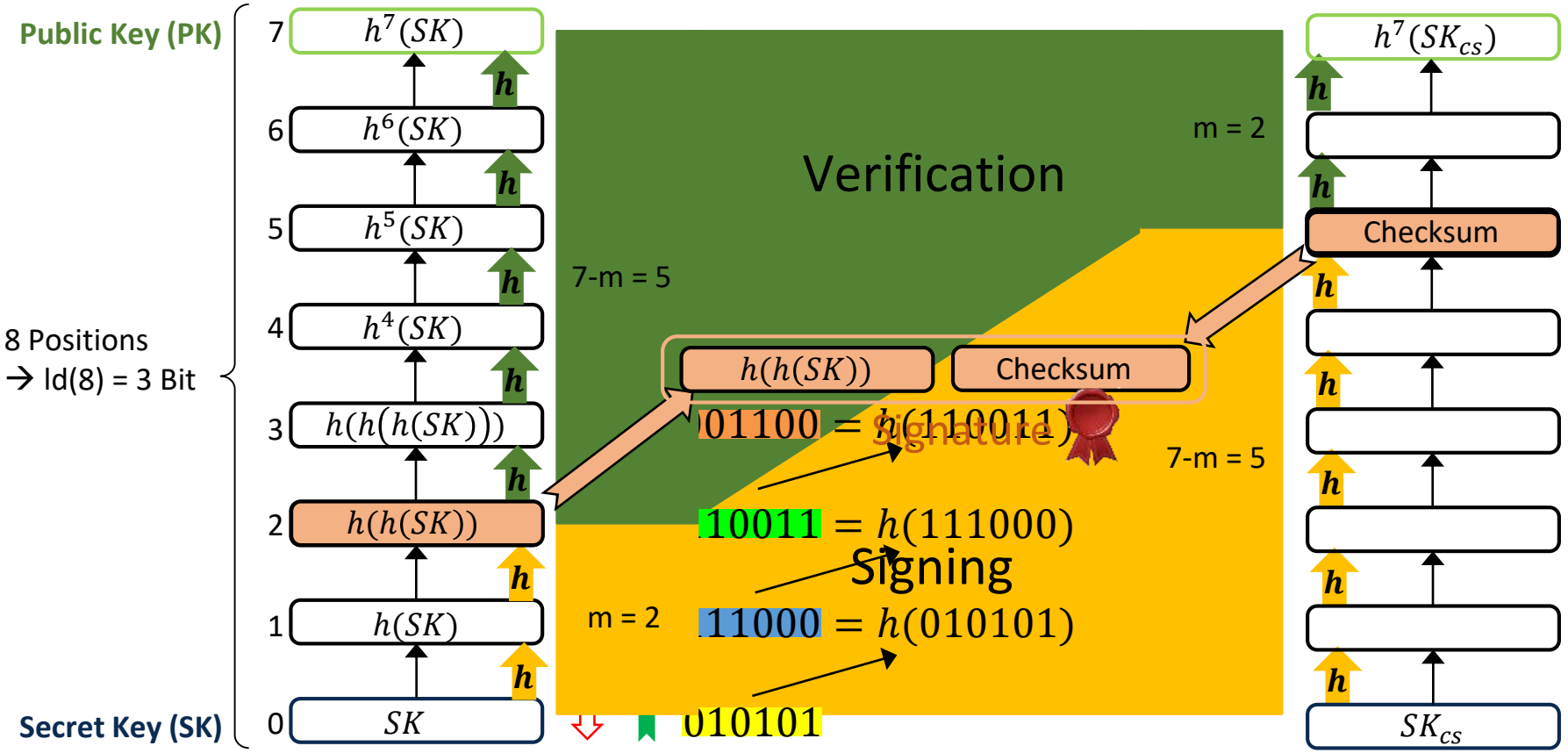
Hash-based Digital Signatures

- **Large signatures**
- **High processing demand**

Can Hash-based Digital Signatures meet the low-latency requirements of the tactile internet?

- Achievable latencies?
- Trade-offs?
- What are most efficient solutions?
- What use cases can be served?

Principle of Hash-based Digital Signatures



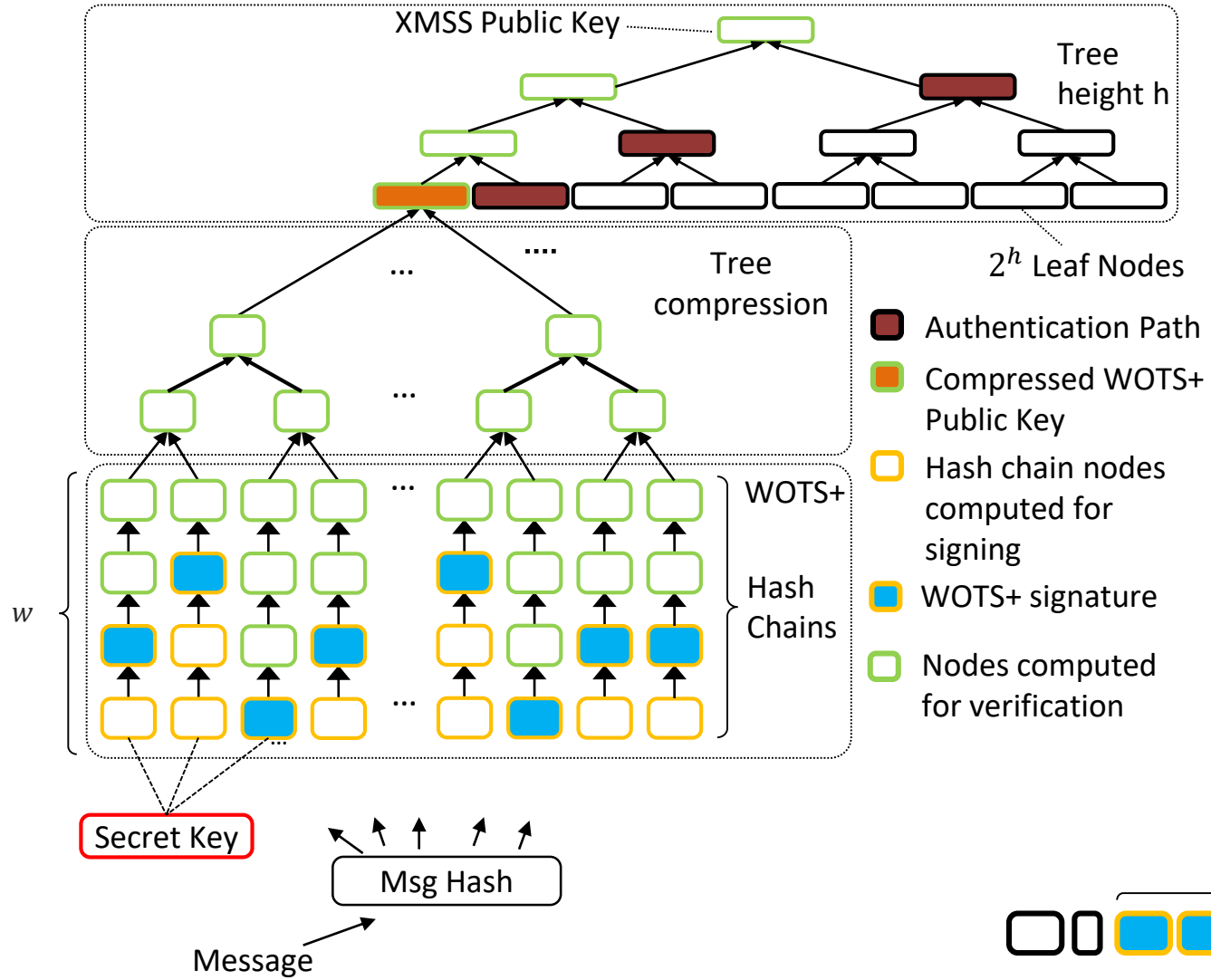
Message m: 0 1 0 m = 2

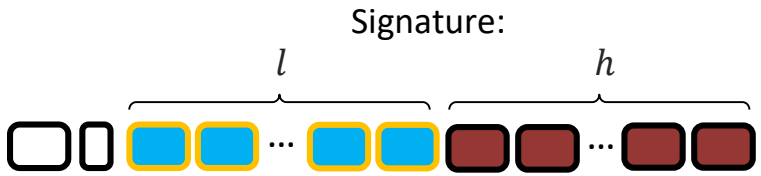
$$out = h(in)$$

$$in = h^{-1}(out)$$

Hash function is not invertible

eXtended Merkle Signature Scheme (XMSS)





State-of-the-Art XMSS Implementations

- High end desktop PC, Oliveira et al. [15]
  - Intel Core i7 @ 4 GHz
  - XMSS Hash Algorithm: SHAKE128
  - **Sign operation: 3.7 ms**
  - **Verify operation: 0.3 ms**

x40

- Our initial Implementation:
  - XMSS Reference Implementation, SHAKE128
  - Cadence Tensilica LX6 @ 1 GHz
  - **Sign operation: 149 ms**
  - **Verify operation: 36 ms**

x1850

- RISC-V Embedded XMSS Hardware Accelerator, Wang, W. et al. [17]
  - RISC-V (FPGA @ ~85 MHz)
  - XMSS Hash Algorithm: SHA-256
  - **Sign operation: 23.5 ms**
  - **Verify operation: 6.54 ms**

x300

- Objective: End-to-end Latency of Application of a 1ms
  - Time budget includes: sensor input, processing, data transmission
  - Time budget **for authentication 0.1 ms**

→ **Hardware acceleration necessary**

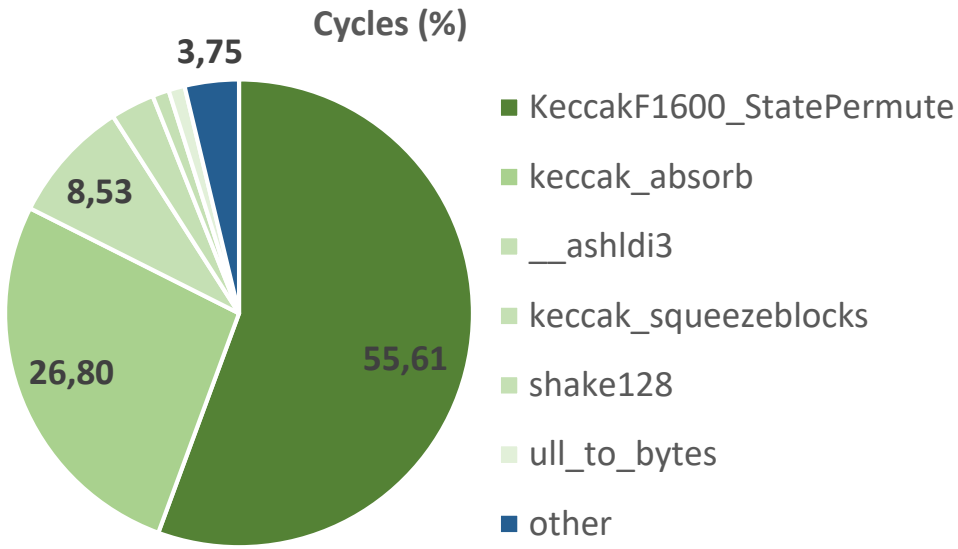
[15] Ana Karina D. S. de Oliveira et el, 2017. High performance of hash-based signature schemes.

[17] Wang, W. et al., 2018 XMSS and Embedded Systems - XMSS Hardware Accelerators for RISC-V

Bottleneck Analysis

- Benchmark Implementation
  - Key generation
  - Sign operation
  - Verify operation
- Software profiling
- Over 90% of operations are related to hash operations

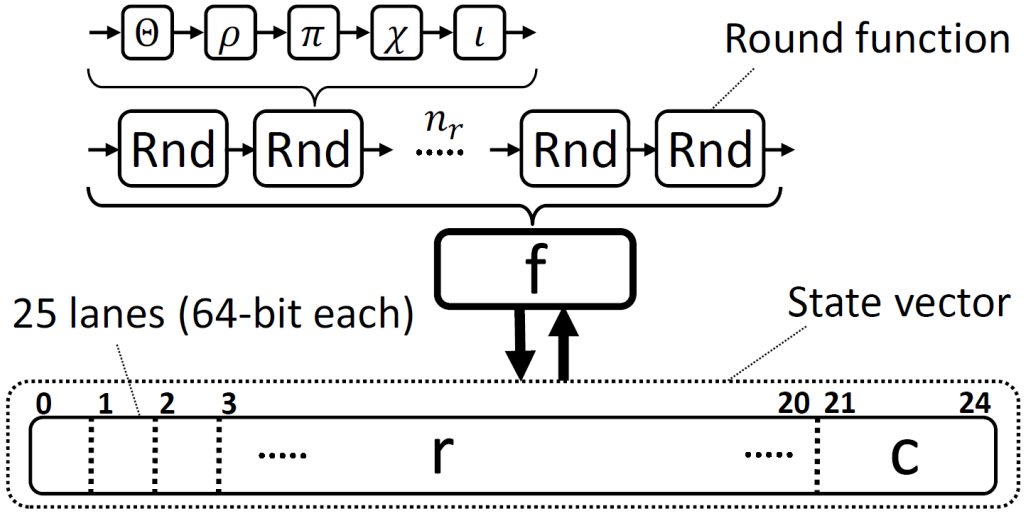
  → **Hash function is the bottleneck**
- Hardware Acceleration
  - ASIP Approach
  - Instruction set extensions
  - Cadence Tensilica LX6
    - 7-stage Pipeline
    - 64-bit memory interface
  - Hash instance for XMSS: SHAKE128 (SHA-3 family, a.k.a Keccak)



Generic Keccak Accelerator

- Choosing state vector implementation
  - 25 lanes of 64 bit each as state needed
    → 1600 bit
    → needs to be mapped to HW
  - Low latency consideration:
    - Rnd computation should be computed in 1 cycle
    - Only possible if full state (1600 bit) is available

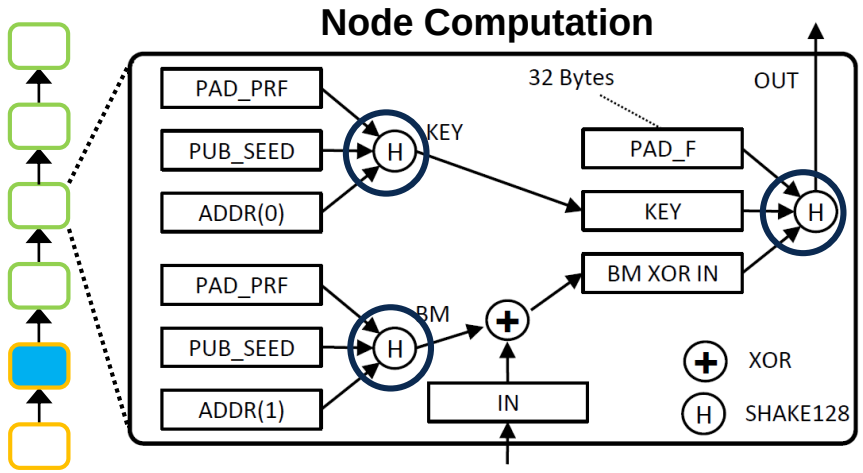
    → 25x64 bit HW state register



Generic Keccak Accelerator

Instruction Histogram of Node Computation

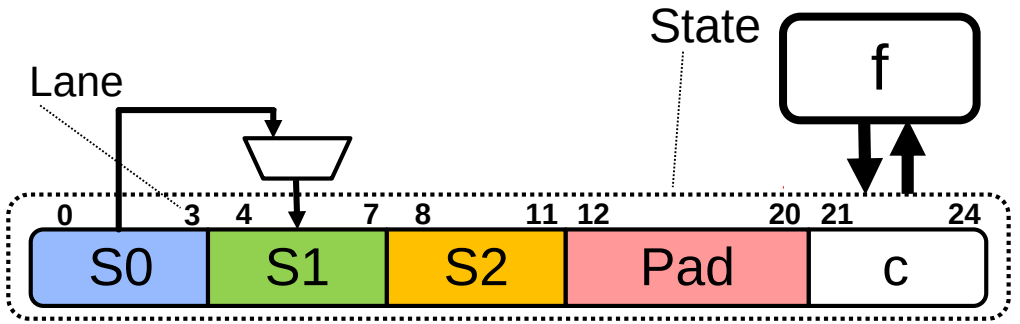
```
Cycles per call: 257.0
kec_wr_state    82.0 (31.9%) (2.2 cpi)
kec_round       73.0 (28.4%) (1.0 cpi)
movi            19.0 ( 7.4%) (0.3 cpi)
s32i.n          17.0 ( 6.6%) (1.0 cpi)
kec_rd_lane     12.0 ( 4.7%) (1.0 cpi)
movi.n           9.0 ( 3.5%) (0.4 cpi)
s32i             9.0 ( 3.5%) (0.5 cpi)
mov.n            7.7 ( 3.0%) (0.4 cpi)
l32i             7.3 ( 2.9%) (0.7 cpi)
nop              5.0 ( 1.9%) (0.3 cpi)
addi             4.0 ( 1.6%) (0.5 cpi)
entry            4.0 ( 1.6%) (4.0 cpi)
kec_init         3.0 ( 1.2%) (1.0 cpi)
s8i              2.0 ( 0.8%) (0.5 cpi)
l32r             1.0 ( 0.4%) (1.0 cpi)
retw.n           1.0 ( 0.4%) (1.0 cpi)
addi.n           1.0 ( 0.4%) (1.0 cpi)
```



- Intermediate performance analysis
  → 257 cycles to compute a node
- Script to extract profiling data → instr. histogram
- Instruction histogram analysis of a node computation
  - High pressure on memory subsystem
  - Loads (35% of cycles)
  - Stores (16% of cycles)
  - >50% of cycles for memory operations

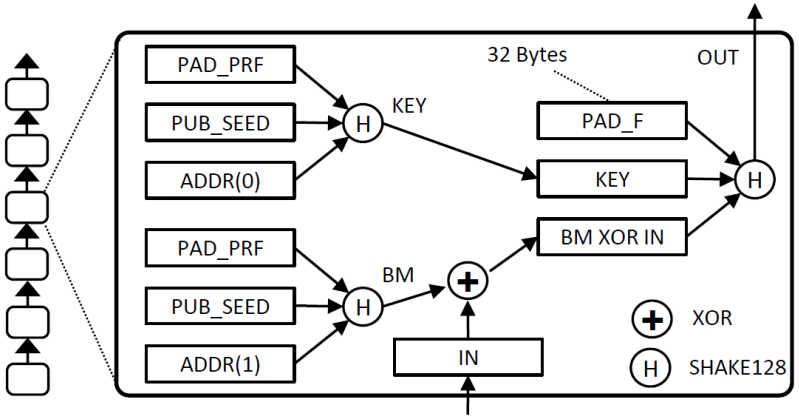
XMSS Specific Accelerator: Results Shift

- Result Shift Mechanism
  - Reduce load/stores of intermediate results
- HW Padding and Result Shift
  - Reduces cycles from 257 → 201
  - No buffers → cheap in area
  - Still 42% of cycles for Load/Store



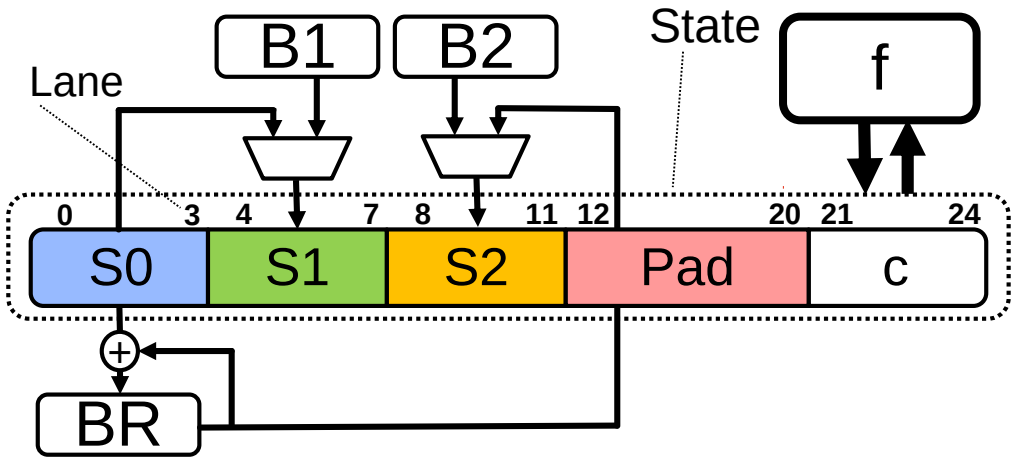
Instruction Histogram of Node Computation

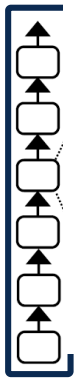
```
Cycles per call: 201.0
kec_round        73.0 (36.3%) (1.0 cpi)
kec_wr_state     62.0 (30.8%) (2.6 cpi)
movi             18.3 ( 9.1%) (0.3 cpi)
s32i.n           12.0 ( 6.0%) (1.0 cpi)
kec_rd_lane       8.0 ( 4.0%) (1.0 cpi)
movi.n            6.3 ( 3.2%) (0.4 cpi)
nop               5.3 ( 2.7%) (0.3 cpi)
mov.n             4.3 ( 2.2%) (0.3 cpi)
entry             4.0 ( 2.0%) (4.0 cpi)
kec_init          3.0 ( 1.5%) (1.0 cpi)
s32i              2.3 ( 1.2%) (0.3 cpi)
l32r              1.0 ( 0.5%) (1.0 cpi)
retw.n            1.0 ( 0.5%) (1.0 cpi)
l32i              0.3 ( 0.2%) (0.3 cpi)
```



XMSS Specific Buffer Architecture

- 2 x Buffers (B1 and B2)
  - 32 byte each
  - Serve Slot 1 and 2
- Result Buffer
  - 32 byte
  - Serves Slot 2





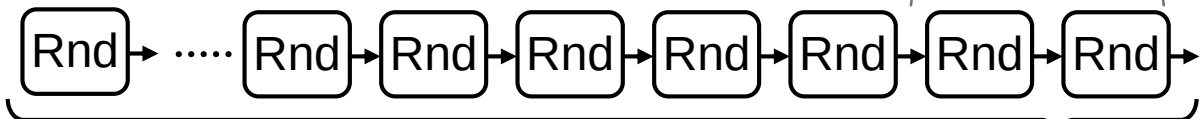
Architecture	Number of Cycles (% of Total)				
	Total	Load	Store	f	Other
Keccak HW	257	90 (35%)	40 (16%)	73 (28%)	54 (21%)
+ Padding Gen./Result Shift	201	63 (31%)	22 (11%)	73 (36%)	42 (21%)
+ Tailored Buffers (FB-EL1)	92	6 (7%)	2 (2%)	73 (79%)	11 (12%)

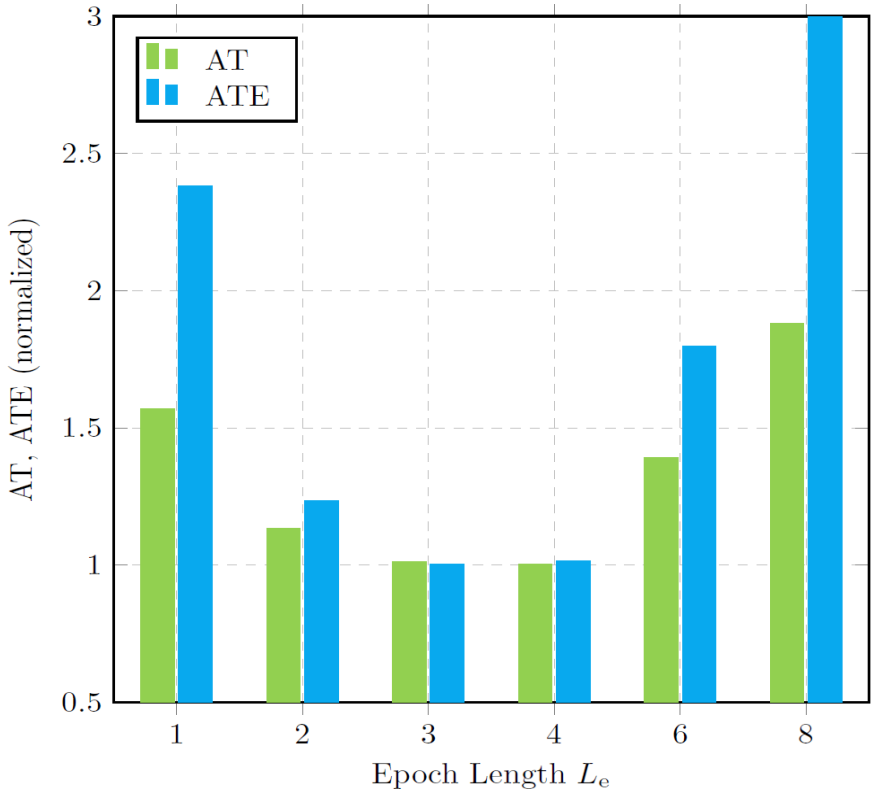
Improving the Round Function Computation

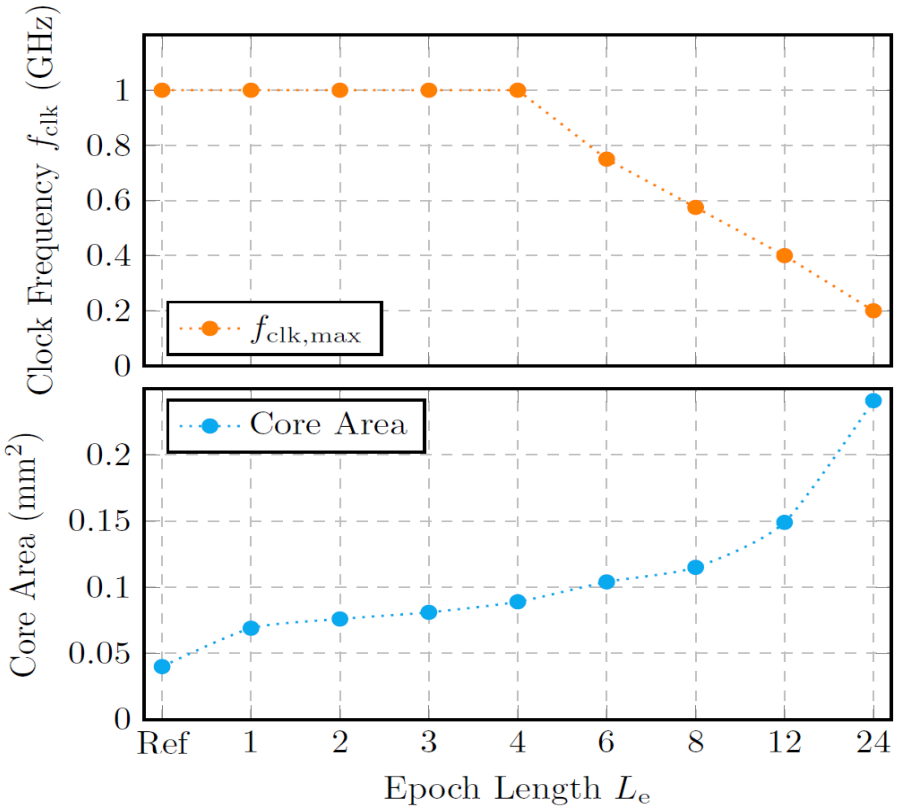
Epoch Length L_e

$L_e = 4$ → 6 cycles (24x Rnd)

$L_e = 2$ → 12 cycles (24x Rnd)







Results

Architecture	Number of Cycles (% of Total)									
	Total		Load		Store		f		Other	
Keccak HW	257		90	(35%)	40	(16%)	73	(28%)	54	(21%)
+ Padding Gen./Result Shift	201		63	(31%)	22	(11%)	73	(36%)	42	(21%)
+ Tailored Buffers (FB-EL1)	92		6	(7%)	2	(2%)	73	(79%)	11	(12%)
+ Round Function (FB-EL3)	43		6	(14%)	2	(4%)	24	(56%)	11	(26%)

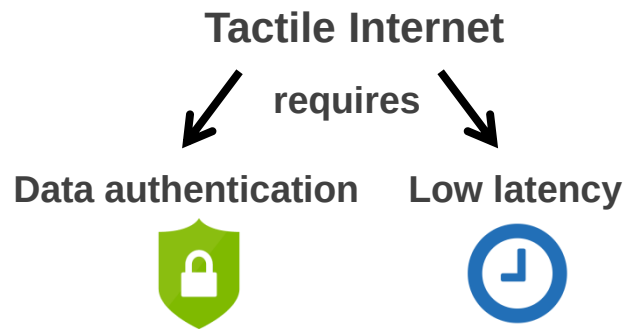
	Cells (10^3)	ΔCells (10^3)	ΔCells (rel.)	Area (mm^2)	Power (mW)	Sign (µs)	Verify (µs)	KeyGen (s)
CT-LX6	88.8	0	(+ 0%)	0.040	9	149k	35.9k	-
FB-EL1	143.8	54.9	(+62%)	0.069	21	330	79	-
FB-EL3	173.1	84.2	(+95%)	0.081	25	176	42	178
MultiBuffer [15]						3740	300	410
Wang *et al.* [17]						(23.5k)[a]	(6.5k)[a]	-[a]

[a] In [17] $h = 10$ is used. Comparison of key generation times would be nonproductive.

Conclusion

- Latency goal 0.1 ms

- State-of-the-Art implementations two slow:
  4ms up to 30ms



- System design for low-latency XMSS processing
  - Hash function specific: Generic Keccak (SHA-3) Accelerator
  - Algorithm specific: XMSS Accelerator
    - HW padding generator, result shift
    - Tailored Buffer Architecture ($T_{sign} = 330\mu s, T_{verify} = 79\mu s$)
    - Optimization of round function ($T_{sign} = 176\mu s, T_{verify} = 42\mu s$)
  - Further latency optimizations possible ($T_{sign} = 39\mu s$) → $T_{Total} = 81\mu s$

- Low-latency data authentication in sub-millisecond range feasible