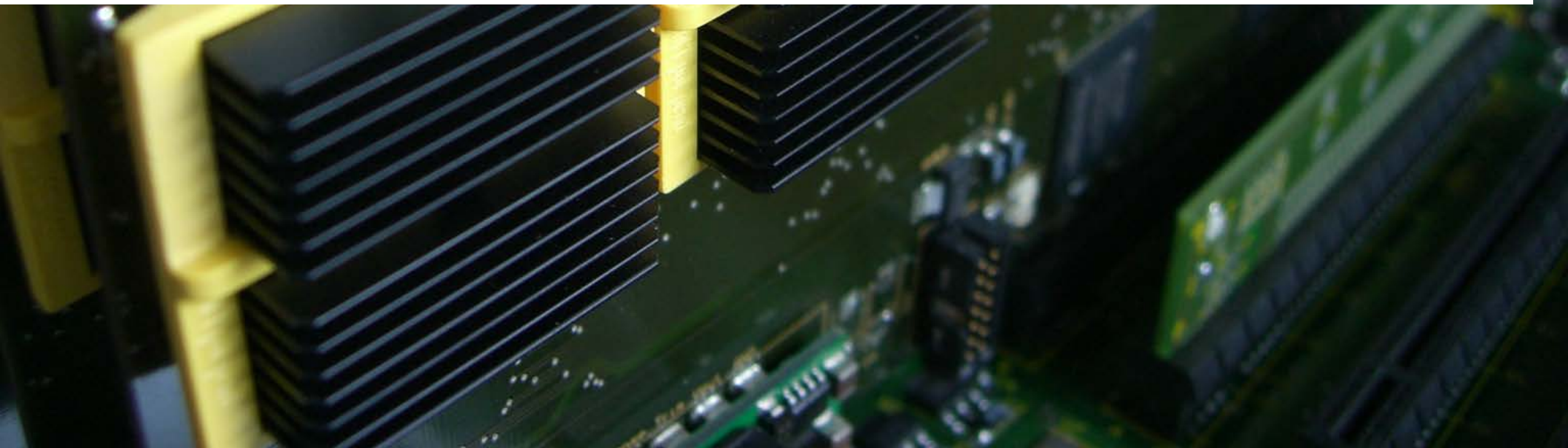
RUHR-UNIVERSITÄT BOCHUM
Worcester Polytechnic Institute

Faster Hash-based Signatures with Bounded Leakage

SAC 2013, Burnaby, Canada

Thomas Eisenbarth[1], Ingo von Maurich[2] and Xin Ye[1]
[1]Worcester Polytechnic Institute, Worcester, MA, USA
[2]Horst Görtz Institute for IT-Security, Ruhr University Bochum, Germany

August 15, 2013



Motivation

Status quo

- Providing efficient PKC in embedded systems is challenging
- Side-channel attacks are a serious threat, protection is costly
- RSA, ECC, (EC-)DSA, …, breakable by quantum computers

→ Need for an efficient, post-quantum signature scheme with limited side-channel leakage

Idea

- Leakage-resilient schemes aim for inherent SCA resistance
- Candidate: Merkle signature scheme (MSS) with Winternitz one-time signatures (W-OTS)
  - Efficient in embedded systems [RED+08, HBB12]
  - Possible choice for a time-limited signature scheme
  - No efficient attacks on quantum computers (with right parameters)

Overview

Motivation

Background

Optimized Authentication Path Computation

Implementation and Leakage Analysis

Conclusions

Overview

Motivation

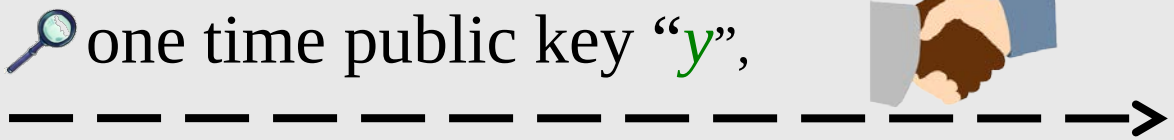
Background

Optimized Authentication Path Computation

Implementation and Leakage Analysis

Conclusions

Hash-based Signatures: Principle



Key Setup

1. Select random x
2. Calculate: $y = f(x)$

Choice: disclose x?

How much information is signed?

One-time Signature Scheme

one time public key "$y_1, y_2, \ldots, y_n$",

 , "x_1, x_4" → signature of "1 , 0"

Key Setup

1. Select random $x_1, x_2, \ldots, x_n$
2. Calculate: $y_i = f(x_i)$

$$\left.\begin{matrix} y_1 \\ y_2 \end{matrix}\right\} \stackrel{?}{=} f(x_1)$$

$$\left.\begin{matrix} y_3 \\ y_4 \end{matrix}\right\} \stackrel{?}{=} f(x_4)$$

$= \overbrace{1}\ \overbrace{0}$

$= \overbrace{x_1, \not{x_2}}, \overbrace{\not{x_3}, x_4}$

Winternitz One-Time Signatures

- Uses hash chains to sign chunks of bits at once
- Reduces signature length (main drawback of Hash-based OTS)

Feature?

- **Can only be used once → leaks information only once**

Practicalities

- Secret signing key X is generated using PRNG
- Generation of public verification key Y requires generation of X and all hash chains

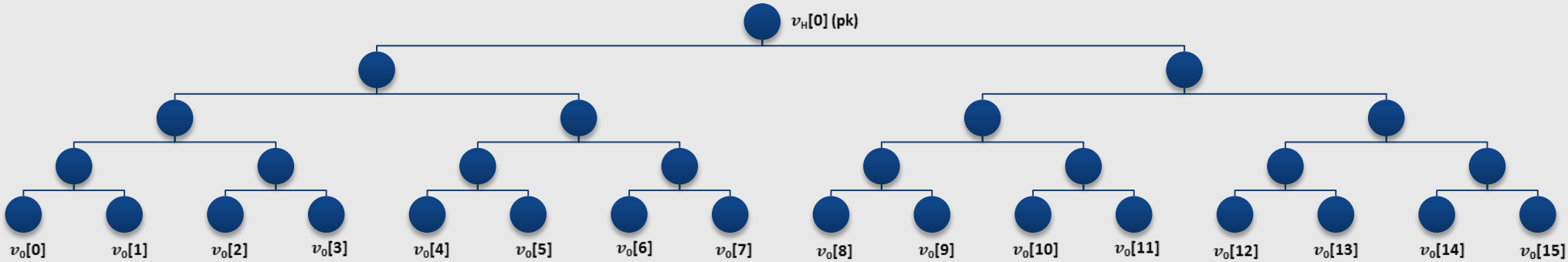
→ **Leakage:** Each generation of Y causes one full leakage of X

Background on Hash-based Signatures

- Given some one-time signature scheme (OTSS)
- Hash together OTS verification keys in a binary hash tree
- Cryptographic hash function $g : \{0,1\}^* \rightarrow \{0,1\}^m$
- Choose tree height H, allows 2^H signatures

MSS Key Generation

- Leaves are hashed one-time verification keys $\nu_0[i] = g(Y_i)$
- Generate root of Merkle tree (MSS verification key)



Background on Hash-based Signatures

MSS Signature Generation

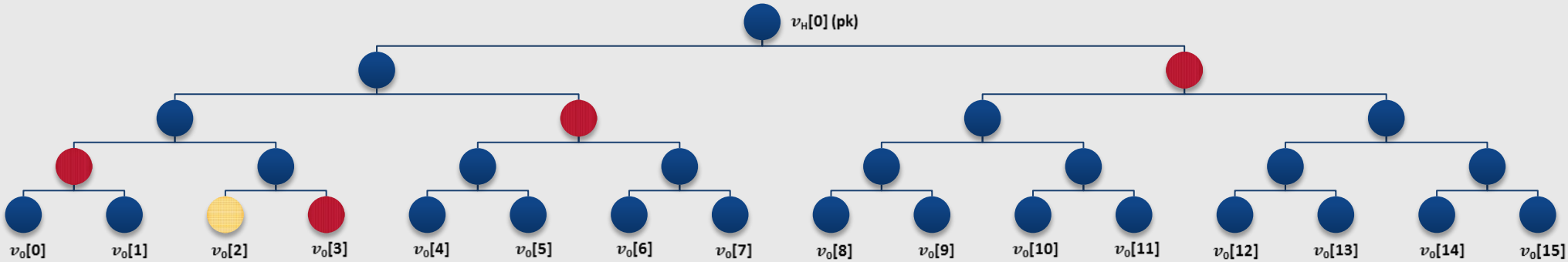
- Compute message digest $d = g(M)$, choose signature index s
- Compute OTS $\sigma_{\mathrm{OTS}} = \mathrm{Sign}_{\mathrm{OTS}}(d, X_s)$ under signing key X_s
- Find all sibling nodes on the way to the root $(\mathrm{AUTH}_0, \ldots, \mathrm{AUTH}_{H-1})$

$$\sigma_s(d) = (s, \sigma_{\mathrm{OTS}}, Y_s, (\mathrm{AUTH}_0, \ldots, \mathrm{AUTH}_{H-1}))$$

- Authentication path independent of message → precomputable

Current (hashed) verification key

Authentication nodes



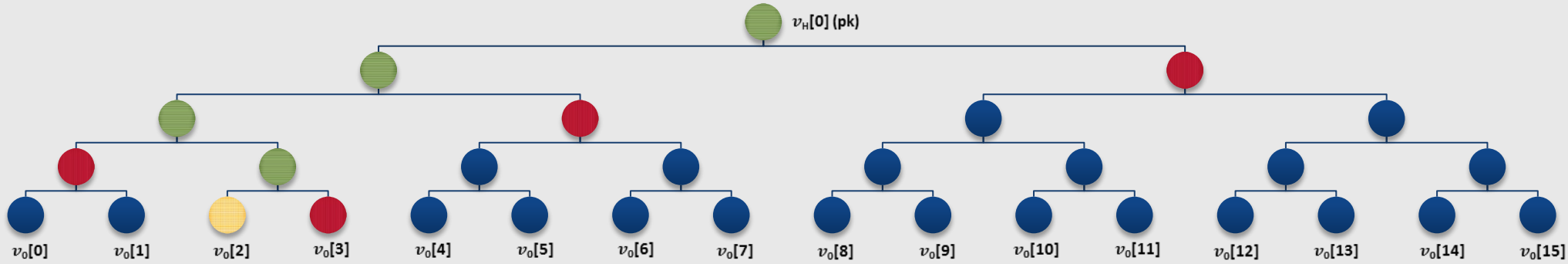
Background on Hash-based Signatures

MSS Signature Verification

- Given digest $d = g(M)$ and signature $\sigma_s(d)$
- Verify underlying one-time signature

→ Reconstruct root of the Merkle tree

- Current (hashed) verification key
- Authentication nodes
- Reconstructed nodes



Practical MSS

Leakage Resilient MSS

- Simply generating and storing all OTS keys independently will yield a leakage resilient signature scheme
- All computations in Merkle tree are public ➔ **No Leakage**
- Memory consumption is too high, so:

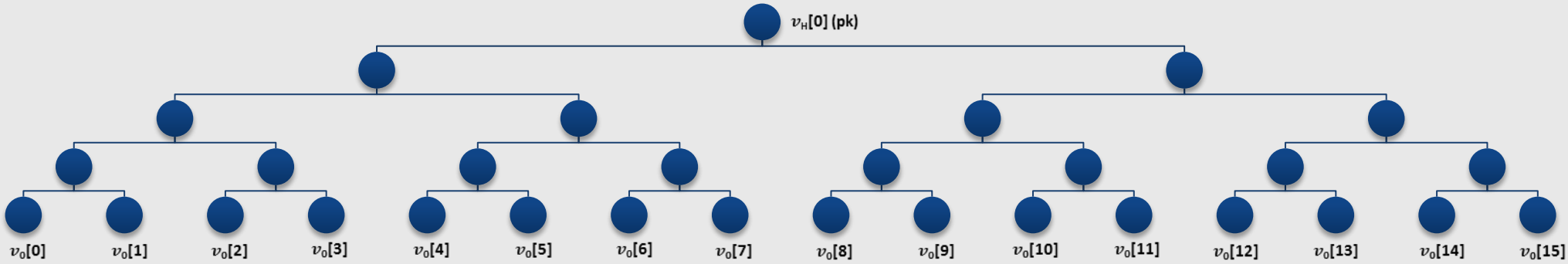
Practical MSS

- Uses PRNG to generate OTS keys
- Allows for just-in-time generation of OTS verification keys for authentication path
- Several optimized algorithms for efficient authentication path generation have been proposed

Authentication Path Computation

Currently best solution: BDS algorithm [BDS09]

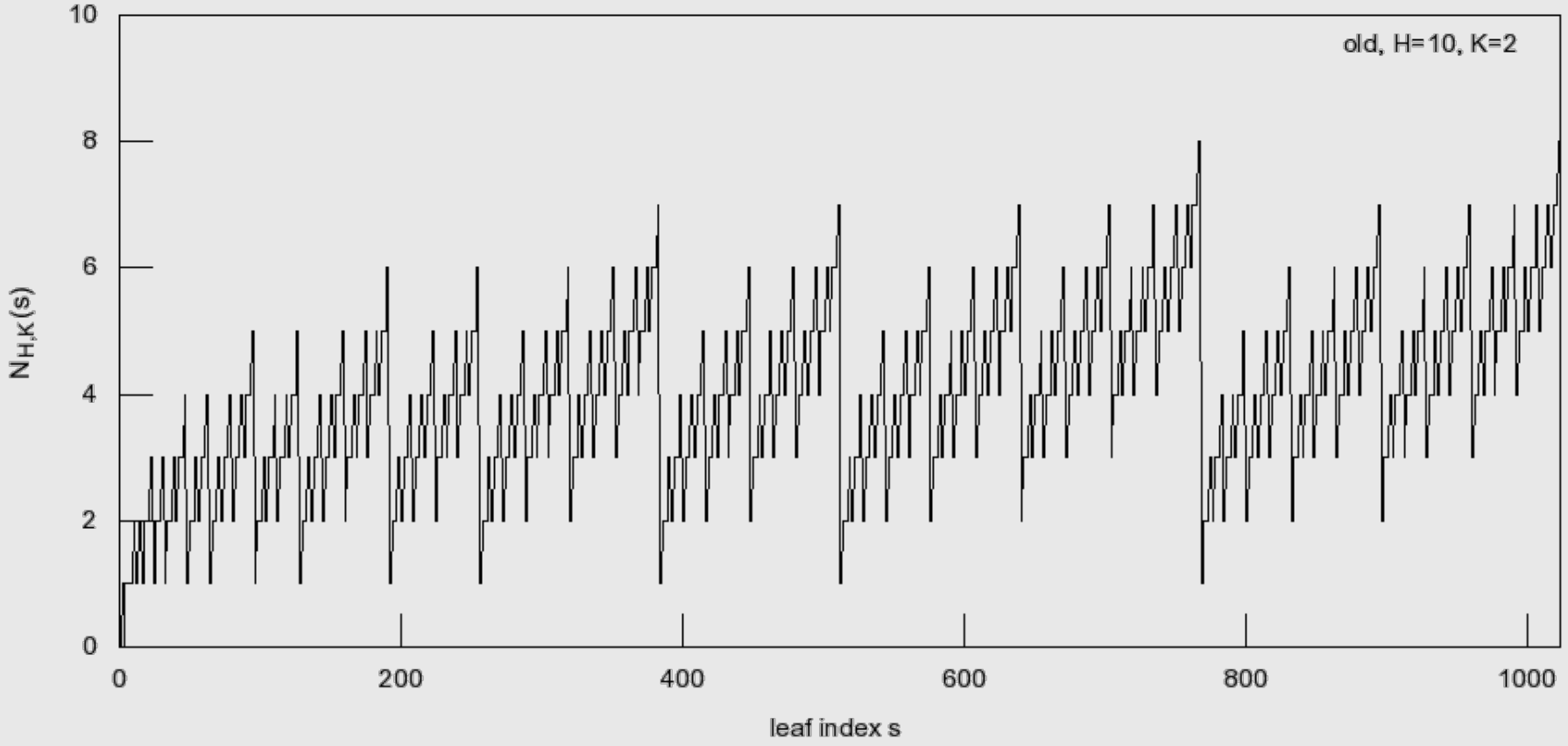
- *Left* authentication nodes are easy to compute:
  either leaf or both child nodes are part of previous authentication paths → store and reuse
- *Right* authentication nodes computed from scratch
- Two ways to determine right authentication nodes
  - Nodes close to the top are most expensive to compute → store them
  - Use instances of the Treehash algorithm [Mer89, Szy04] to compute lower right nodes — one instance per tree level



Drawbacks of the BDS Algorithm

Unbalanced leaf computations

- Some leaves are generated various times, others are barely touched
- Each computation means additional leakage!



Overview

Motivation

Background

Optimized Authentication Path Computation

Implementation and Leakage Analysis

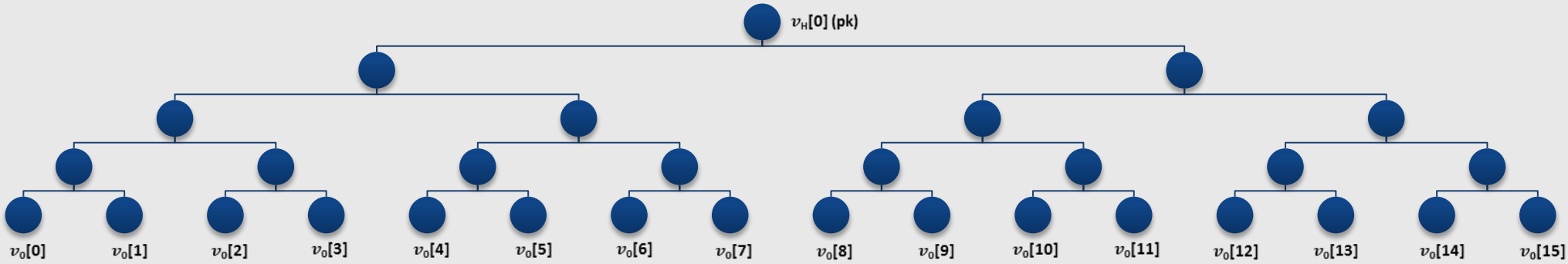
Conclusions

Optimized Authentication Path Computation

- Rightmost nodes of each Treehash instance are computed most frequently → store and reuse them in lower Treehash instances
- Excluding the root of each Treehash instance and Treehash_0
- Every second reinitialization of Treehash_h, $0 \leq h \leq H - K - 2$ can now directly copy the authentication node

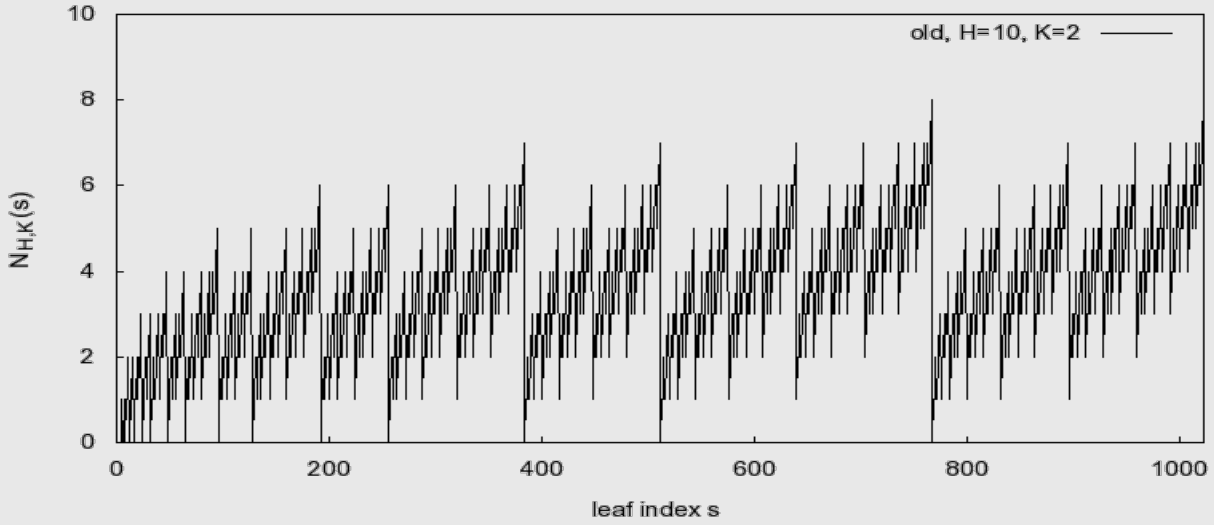
➔ Number of updates is (almost) **halved**!

- Additional storage requirement: $\binom{H-K}{2}$ nodes



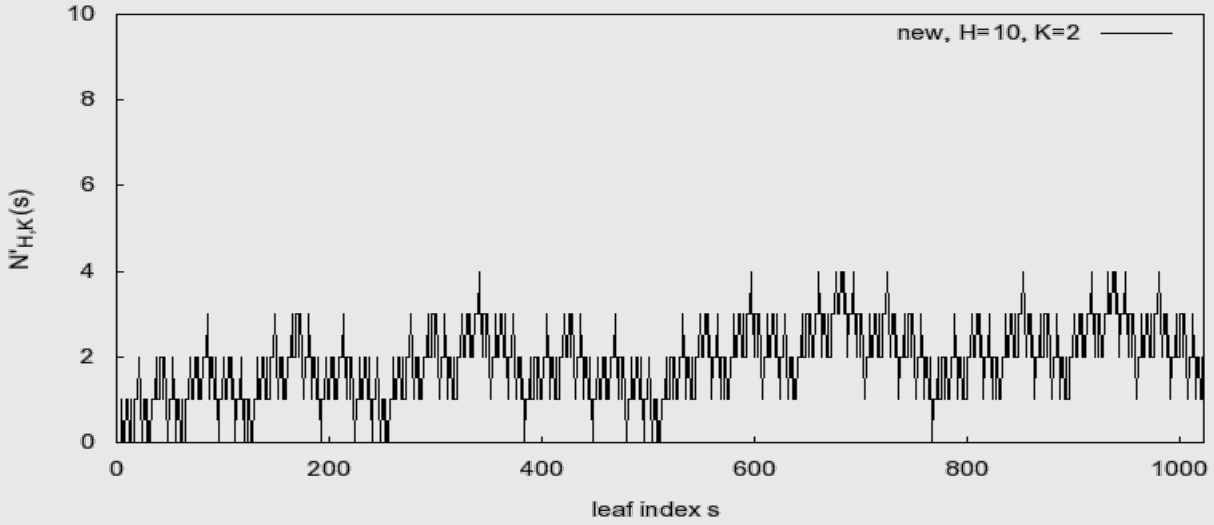
Comparison

Before



Average:
$\approx 1/2(H-K)$

Now



Average:
$\approx 1/4(H-K+1)$

Leakage is halved

Overview

Motivation

Background

Optimized Authentication Path Computation

Implementation and Leakage Analysis

Conclusions

Implementation Choices

- General purpose hash functions have bad performance
- Block cipher constructions can make use of AES coprocessor

For Merkle Tree

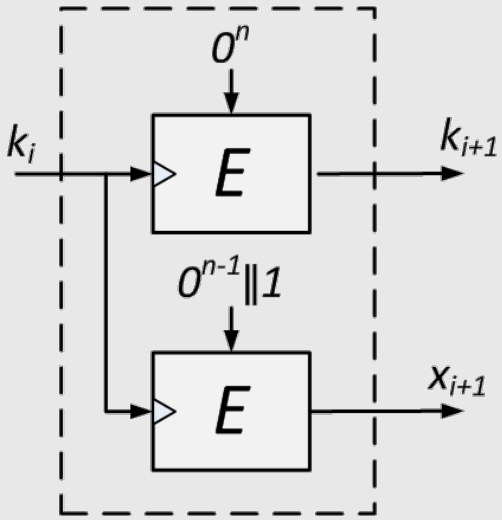
- g **uses AES-128 MJH construction** [LM11] w\ 256/160-bit output

For Winternitz OTS

- f **uses AES-128 MMO construction** [MMO85] w\ 128-bit output

PRNGs

- Leakage-2-limiting PRNG proposed in [SPY+10]
  - Each key used on 2 different inputs only
  - Only static (i.e. non-adaptive leakage allowed)

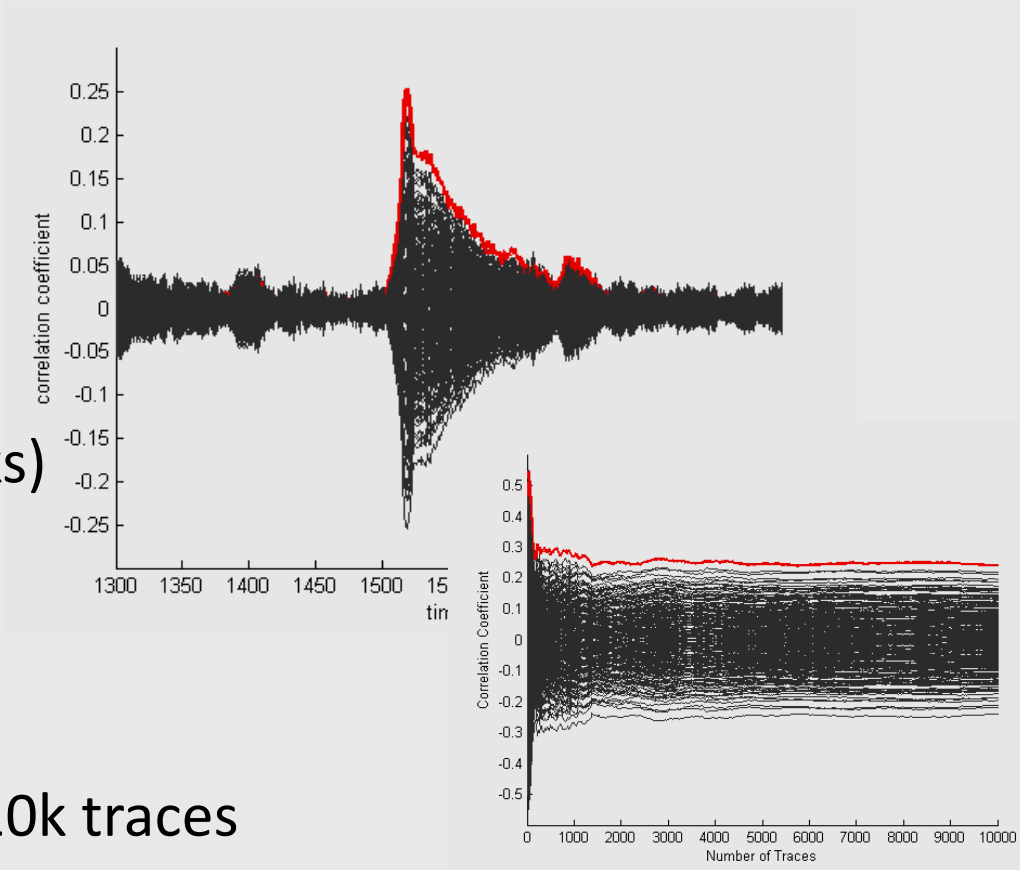


Side Channel Analysis of XMEGA-AES

- AVR XMEGA has unprotected AES co-processor

DPA results

- **[Kiz09]** suggests approx. 2500 traces for key recovery
- No strong leakage at S box output
- Instead HD of 2 inputs (ghost peaks)



- Our results match **[Kiz09]**
- Our correlation is slightly higher:

.27 instead of .18 @10k traces

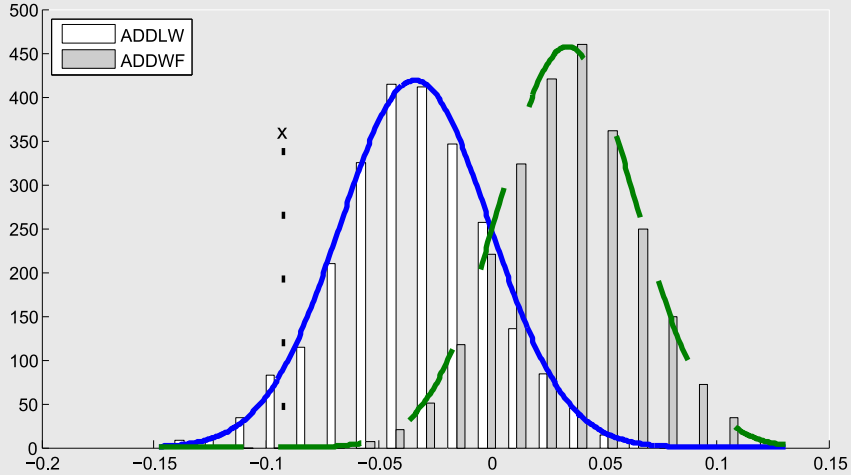
This implies slightly better measurement setup

300 traces suffice

Quantifying Single Measurements

Template Attack

- Univariate templates
- Built from 10k traces
- Point selection via DPA

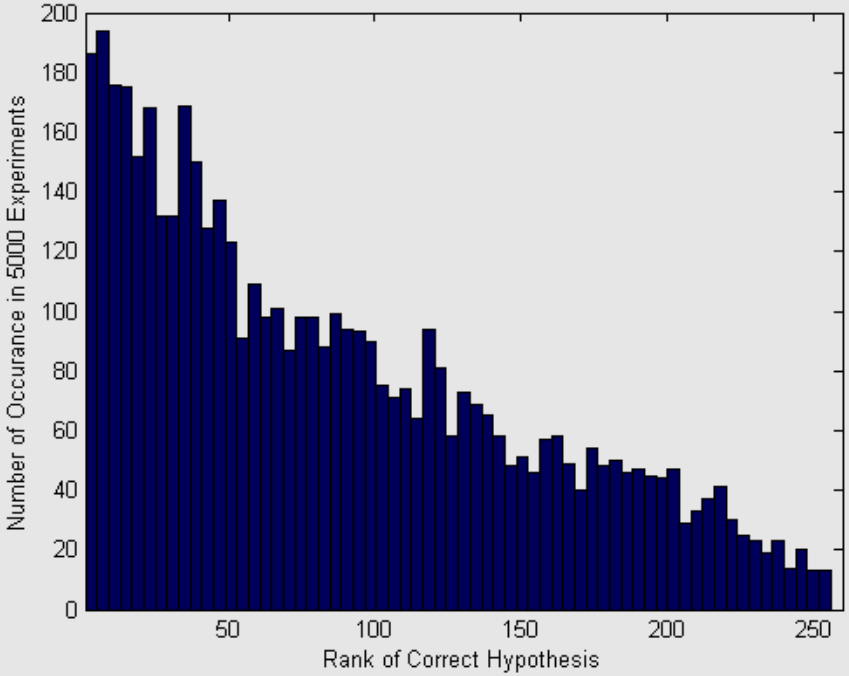


PRNG Leakage Quantification

- 10 leakages on 2 different inputs
- 5000 experiments
- **Guessing Entropy** (av. key rank)

  85.06 or 6.41 bit

→ **Less than one bit per byte**



Implementation Results

CPU Performance (Intel Core i7-2620M, 64-bit)

- Parameters $H = 16, K = 2, w = 2$
- Disabled Turbo Boost and Hyper-threading

Microcontroller Performance (Atmel AVR ATxmega128A1, 8-bit)

- Parameters $H = 10, K = 2, w = 2$
- KeyGen infeasible for reasonable tree heights

Hash g		MJH-256 w/ AES-128			MJH-160 w/ AES-128		
Target		[22]	our	impr.	[22]	our	impr.
Core i7	KeyGen	6546.9 ms	6037.5 ms	8%	4218.7 ms	3886.3 ms	8%
Core i7	Sign	743.9 us	401.3 us	46%	487.1 us	256.2 us	47%
Core i7	Verify	76.1 us	78.1 us	-3%	50.8 us	49.3 us	3%
AVR	Sign	110.0 ms	64.9 ms	41%	70.7 ms	41.7 ms	41%
AVR	Verify	18.4 ms	18.4 ms	0%	11.0 ms	11.0 ms	0%

Conclusions

- Algorithmic improvement for auth. path computation in MSS
- Balanced leaf computations→Reduced side-channel leakage
- Efficient implementations on two common platforms
- Practically verified the theoretic performance gains and bounded leakage on an embedded device

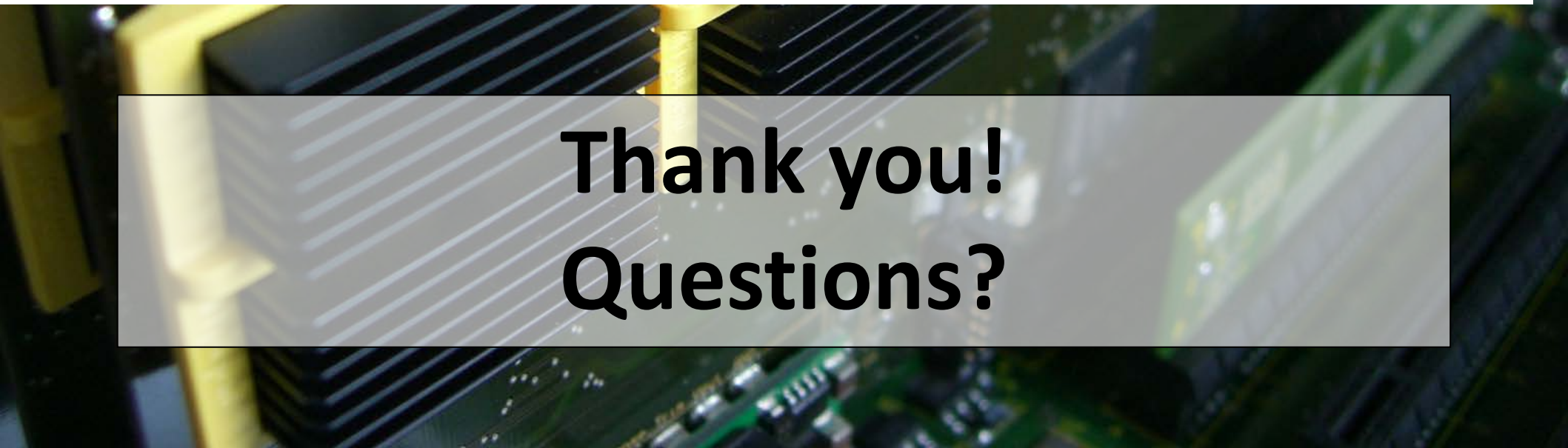
RUHR-UNIVERSITÄT BOCHUM
Worcester Polytechnic Institute

Faster Hash-based Signatures with Bounded Leakage

SAC 2013, Burnaby, Canada

Thomas Eisenbarth[1], Ingo von Maurich[2] and Xin Ye[1]
[1]Worcester Polytechnic Institute, Worcester, MA, USA
[2]Horst Görtz Institute for IT-Security, Ruhr University Bochum, Germany

August 15, 2013



Thank you!
Questions?

References

[BDS09] J. Buchmann, E. Dahmen, and M. Szydlo. Hash-based Digital Signature Schemes. In D. J. Bernstein, J. Buchmann, and E. Dahmen, editors, *Post-Quantum Cryptography*, pages 35–93. Springer Berlin Heidelberg, 2009.

[HBB12] A. Hülsing, C. Busold, and J. Buchmann. Forward Secure Signatures on Smart Cards. In L. R. Knudsen and H. Wu, editors, *Selected Areas in Cryptography*, volume 7707 of *Lecture Notes in Computer Science*, pages 66–80. Springer, 2012.

[Kiz09] I. Kizhvatov. Side Channel Analysis of AVR XMEGA Crypto Engine. In Proceedings of the 4th Workshop on Embedded Systems Security, WESS '09, pages 8:1–8:7, New York, NY, USA, 2009. ACM.

[LM11] J. Lee and M. Stam. MJH: A Faster Alternative to MDC-2. In A. Kiayias, editor, *Topics in Cryptology CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 213–236. Springer Berlin / Heidelberg, 2011.

[MMO85] S. M. Matyas, C. H. Meyer, and J. Oseas. Generating strong one-way functions with cryptographic algorithm. *IBM Technical Disclosure Bulletin*, 27(10A):5658–5659, 1985.

References

[Mer89] R. C. Merkle. A Certified Digital Signature. In G. Brassard, editor, *CRYPTO*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer, 1989.

[RED+08] S. Rhode, T. Eisenbarth, E. Dahmen, J. Buchmann, and C. Paar. Fast Hash-based Signatures on Constrained Devices. In *Smart Card Research and Advanced Applications – CARDIS 2008*, pages 104-117. Springer, 2008.

[SPY+10] F.-X. Standaert, O. Pereira, Y. Yu, J.-J. Quisquater, M. Yung, and E. Oswald. Leakage Resilient Cryptography in Practice. In A.-R. Sadeghi, D. Naccache, D. Basin, and U. Maurer, editors, *Towards Hardware-Intrinsic Security, Information Security and Cryptography*, pages 99–134. Springer Berlin Heidelberg, 2010.

[Szy04] M. Szydlo. Merkle Tree Traversal in Log Space and Time. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 541–554. Springer, 2004.