RSAConference2020
San Francisco | February 24 – 28 | Moscone Center

HUMAN ELEMENT

SESSION ID: **CRYP-R02**

Efficient FPGA Implementations of LowMC and Picnic

Roman Walch

PhD Student
IAIK / Know-Center GmbH, Graz University of Technology
@rw0x0

Post-Quantum Digital Signatures

- Shor‘s algorithm for factoring and discrete logarithm
- Quantum computer breaks:
  - Most asymmetric cryptography
  - **RSA**, **DSA**, **ECDSA, …**
- NIST Standardization Project for PQ Signatures
  - Currently second round
  - **Picnic** [Cha+17; Cha+19] (using **LowMC** [Alb+15])
  - Performance optimized implementations required

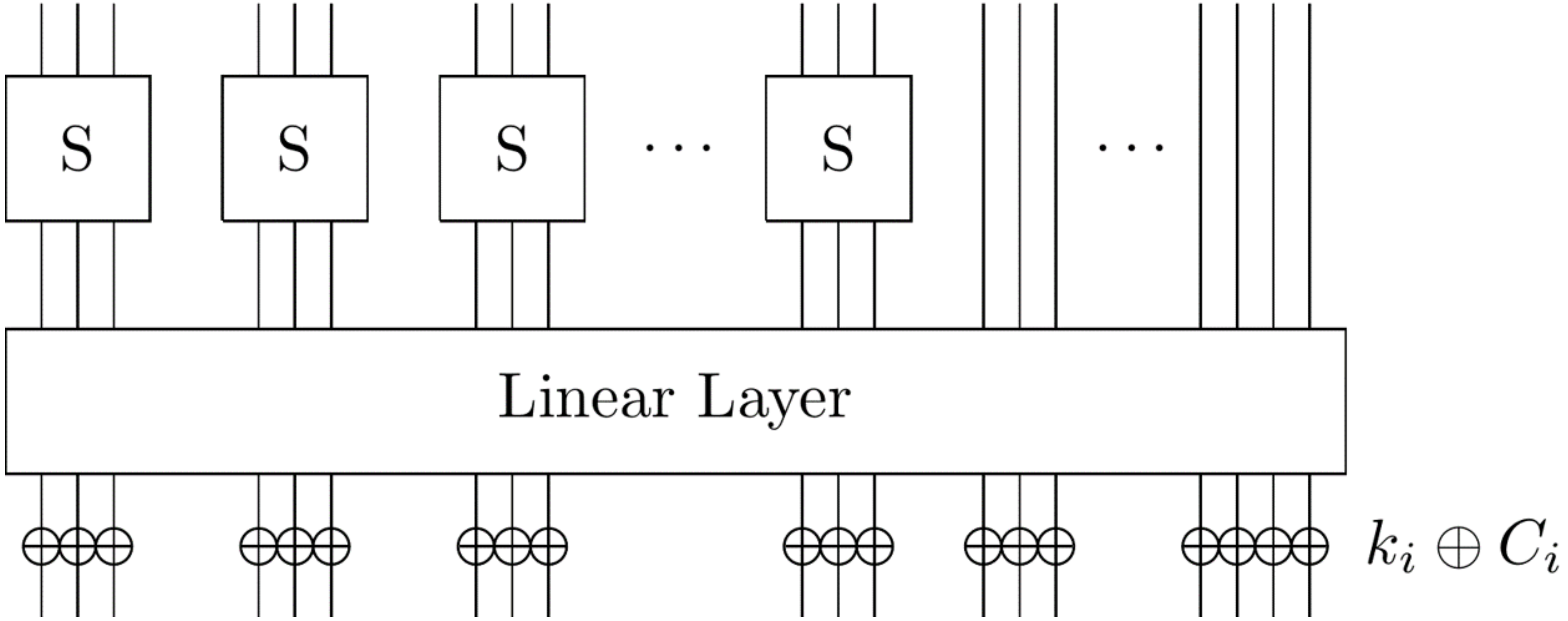
Contribution

- First efficient VHDL implementation of **LowMC**
- First VHDL implementation of **Picnic**
  - **Picnic1-L1-FS:** 128 (64) bit security (PQ)
  - **Picnic1-L5-FS:** 256 (128) bit security (PQ)
- Coprocessors accessible via PCIe interface
  - Communication protocol confrom with NIST recommendation

The LowMC Block Cipher

LowMC – Round

- Substitution-Permutation Network (SPN) with reduced SboxLayer:



LowMC – Details

- Designed to minimize AND gates (3 ANDs / Sbox)
  - $S(a, b, c) = (a \oplus (b \wedge c), a \oplus b \oplus (a \wedge c), a \oplus b \oplus c \oplus (a \wedge b))$
- Linear Layer:
  - State multiplied with matrix over $GF(2)$
  - $n \times n$ matrix per round
- Roundkey schedule
  - Key multiplied with matrix over $GF(2)$
  - $n \times k$ matrix per round + inital key whitening

n … blocksize
k … keysize

LowMC – Constants per Instance

- Naive implementaion:
  - L1: ~82 kiB
  - L5: ~617 kiB

- Optimizations by [Din+19]:
  - L1: ~29 kiB
  - L5: ~117 kiB

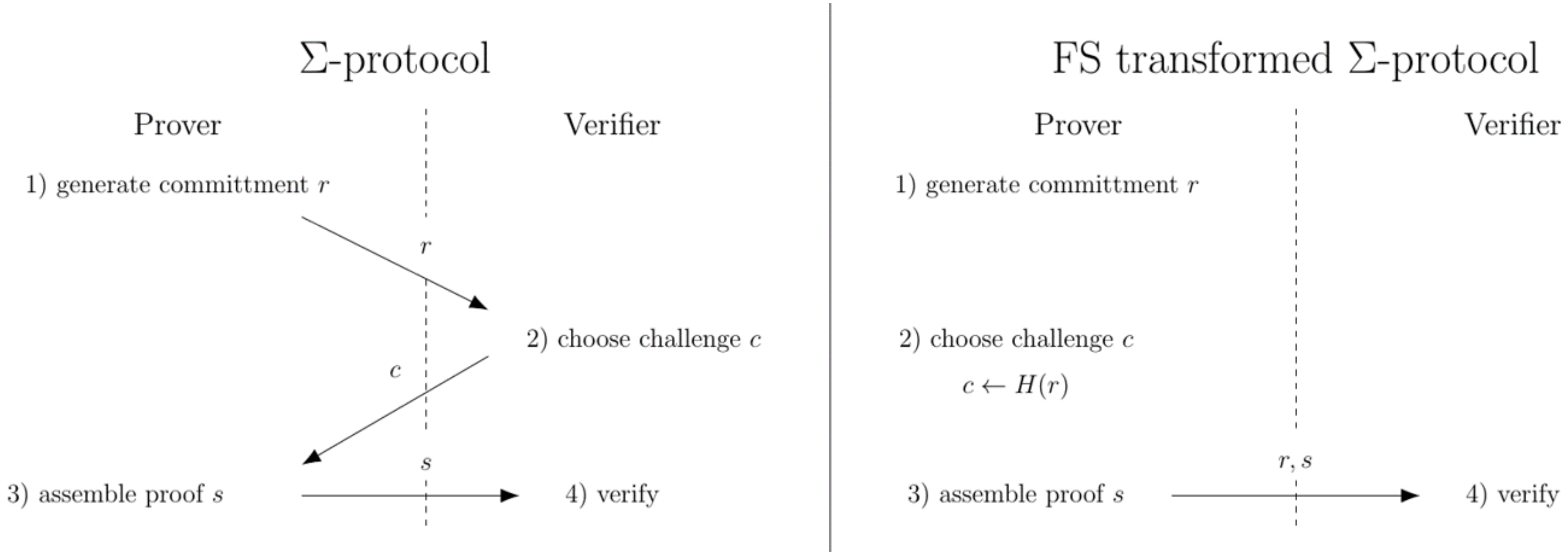
- Impact on hardware utilization

	LowMC				without opt.		with opt.		Improv. %
nr.	n	k	m	r	LUTs	% LUTs	LUTs	% LUTs	
L1	128	128	10	20	42 395	**20.80%**	13 558	**6.65%**	68.02%
L5	256	256	10	38	209 348	**102.72%**	44 431	**21.8 %**	78.78%

The Picnic Signature Scheme

Σ-protocol and Fiat-Shamir

- Σ-protocol for proof of knowledge
- Fiat-Shamir (FS) transformation:
  - Proof becomes non-interactive
  - Secure in the random oracle model (ROM)

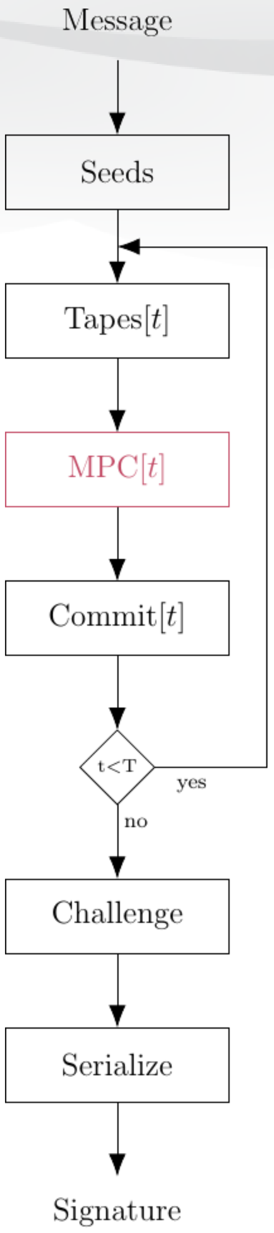


Picnic – Building Blocks

- FS transformed Σ-protocol
- Σ-protocol: **ZKB++** or **KKW**
- Proof system:
  - Multi-party computation (MPC) of **LowMC**
  - Random oracle: **SHAKE** (Keccak)
- Keys:
  - Relation: $C = \mathrm{LowMC}(p, k)$
  - Public Key: $pk = (C, p)$
  - Secret Key: $sk = k$

Picnic – MPC

- MPC of 3 **LowMC** encryptions
  - $sk = sk_0 \oplus sk_1 \oplus sk_2$
  - $C_i = \mathrm{LowMC_{MPC}}(p, sk_i)$
  - $C_0 \oplus C_1 \oplus C_2 = C$
- Repeat T times
  - Reduce probability to cheat
  - **Picnic1-L1-FS:** $T = 219$
  - **Picnic1–L5–FS:** $T = 438$



Picnic – MPC contd.

- 3 players calculate:
  - $C_i = \mathrm{LowMC}_{\mathrm{MPC}}(p, ski)$
- MPC rules to ensure $C_0 \oplus C_1 \oplus C_2 = C$:
  - XOR with constant only for one player
  - Players calculate AND gates $(c = a \wedge b)$ jointly:
  - $c_i = (ai \wedge bi_{\ +\ 1}) \oplus (a_{i\ +\ 1} \wedge b_i) \oplus (ai \wedge bi) \oplus (r_i \wedge r_{i\ +\ 1})$

⇒ Special Sbox implementation

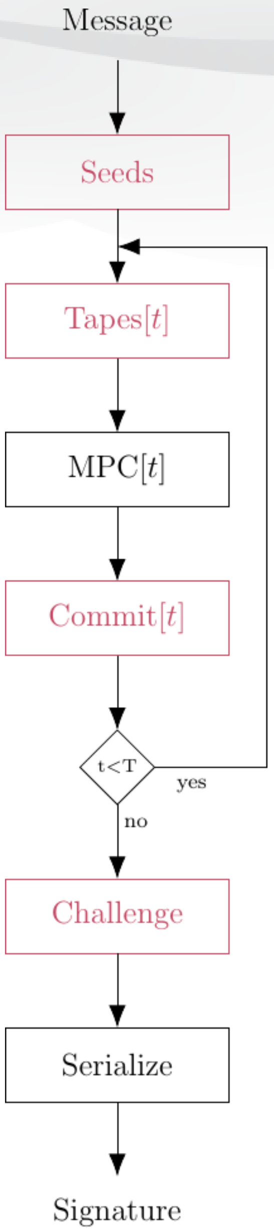
Picnic – MPC Implementation

- 3 players calculated in parallel
- Further improvement
  - Precomputation of one share
  - Only 2 **LowMC** instances on FPGA
- Sign / Verify use same LUTs for matrices
  - But different Sbox implementation

Picnic – Other Submodulues

- Seeds and Tapes
  - Provide Pseudorandomness
- Commitments
  - Players commit to results
  - Part of signature
- Challenge (Random Oracle)

⇒ All using **SHAKE** (different configurations)



Picnic – Implementation

- Custom **SHAKE** implementation
- 3 players parallel per run t
- BRAM for intermediate values
  - ~ 400 kiB for **Picnic1-L5-FS**
- **Picnic1-L1-FS** and **Picnic1-L5-FS** implementations for
  - Sign / Verify only
  - Sign and Verify combined

Practical Evaluation

FPGA and PCIe

- Xilinx Kintex-7 FPGA KC705 Evaluation Kit
- PCIe/DMA subsystem
  - Manages FPGA/PC interface
- AXI4-Stream
  - High data throughput master/slave bus interface
  - Handshake parallel to data transfer
  - Connects our design to PCIe/DMA
- Developed C-Library for PC/FPGA communication

Hardware Utilization

- Lookup tables (LUTs) and BRAM utilization (% available)

Design Part	LUTs	%	BRAM	%
LowMC-MPC-L1	32 224	**15.81 %**	0	0 %
LowMC-MPC-L5	98 319	**48.24 %**	0	0 %
Picnic1-L1	90 037	**44.18 %**	52.5	11.80 %
Picnic1-L1-Sign	76 472	**37.52 %**	52.5	11.80 %
Picnic1-L1-Verify	68 614	**33.67 %**	33.5	7.53 %
Picnic1-L5	167 530	**82.20 %**	98.5	22.13 %
Picnic1-L5-Sign	149 456	**73.33 %**	98.5	22.13 %
Picnic1-L5-Verify	138 547	**67.98 %**	62.5	14.04 %
PCIe/DMA	22 216	**10.90 %**	42.5	9.55 %

Runtime Comparison

- Software platform:
  - Ubuntu 18.04.1, GCC 7.3.0, 16 GB RAM
  - CPU: Intel i7-4790, 3.6 GHz

Coprocessor	clock frequency	clock cycles	FPGA runtime	C-Access runtime	Software	
					SIMD	No SIMD
	MHz	k cycles	ms	ms	ms	ms
Picnic1-L1-Sign	125	~31.3	**0.25**	**0.35**	1.44	2.82
Picnic1-L1-Verify	125	~29.6	**0.24**	**0.40**	1.15	2.34
Picnic1-L5-Sign	125	~154.5	**1.24**	**1.38**	5.87	12.37
Picnic1-L5-Verify	125	~146.6	**1.17**	**2.13**	4.92	10.59

Design Choices – Reducing LUT Utilization

- Implementation is optimized for speed
- **LowMC** matrices encoded in LUTs
  - 1 multiplication per clock cycle
  - High LUT utilization
- Reduce LUT utilization
  - Store **LowMC** matrices in BRAM

  ... reduces performance
  - **LowMC** same matrix each round?
  - **GMiMC** [Alb+19] instead of **LowMC**?

Conclusion

- First efficient VHDL implementation **LowMC**
- First VHDL implementation of **Picnic**
  - **Picnic1-L1-FS** and **Picnic1-L5-FS**
- Extended to FPGA-based coprocessor (PCIe Interface)
- Good runtime
  - Trade off with high hardware utilization

RSAConference2020

Efficient FPGA Implementations of LowMC and Picnic

Questions?

Bibliography I

[Alb+15] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. **Ciphers for MPC and FHE.** EUROCRYPT (1). Vol. 9056. LNCS. Springer, 2015, pp. 430–454.

[Alb+19] Martin R. Albrecht, Lorenzo Grassi, Léo Perrin, Sebastian Ramacher, Christian Rechberger, Dragos Rotaru, Arnab Roy, and Markus Schofnegger. **Feistel Structures for MPC, and More.** ESORICS (2). Vol. 11736. LNCS. Springer, 2019, pp. 151–171.

[Cha+17] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. **Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives.** ACM CCS. ACM, 2017, pp. 1825-1842.

Bibliography II

[Cha+19] Melissa Chase et al. **The Picnic Signature Scheme Design Document (version 2).** 2019. URL: https://github.com/microsoft/Picnic/blob/master/spec/design-v2.0.pdf.

[Din+19] Itai Dinur, Daniel Kales, Angela Promitzer, Sebastian Ramacher, and Christian Rechberger. **Linear Equivalence of Block Ciphers with Partial Non-Linear Layers: Application to LowMC.** EUROCRYPT (1). Vol. 11476. LNCS. Springer, 2019, pp. 343–372.