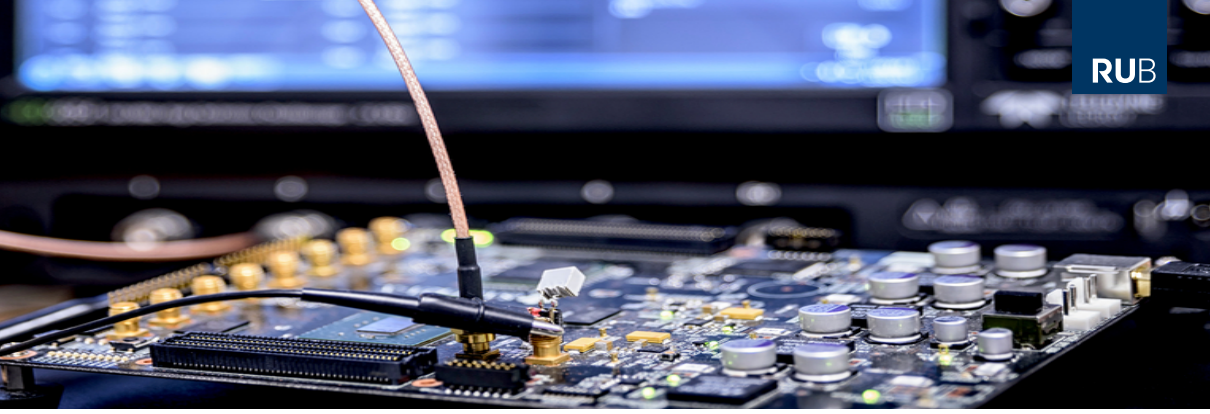


RUHR-UNIVERSITÄT BOCHUM

PQC in the Embedded World

Implementations, Side-Channel Attacks, and Countermeasures

Georg Land

Chair for Security Engineering
Faculty of Computer Science
Ruhr-University Bochum

June 28, 2022

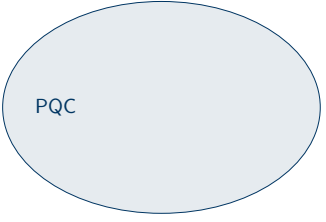
PROMETHEUS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 780701.

What is PQC?

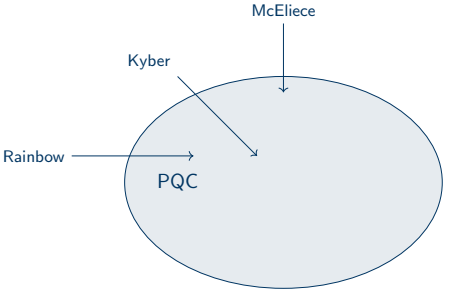
RUB

- PQC = any cryptography that withstands attacks from large-scale quantum computers
- basic (asymmetric) primitives: PKE/KEM and signatures
- advanced primitives: (F)HE, ZKP etc.
- this talk: PQC (only basic primitives) in the embedded world



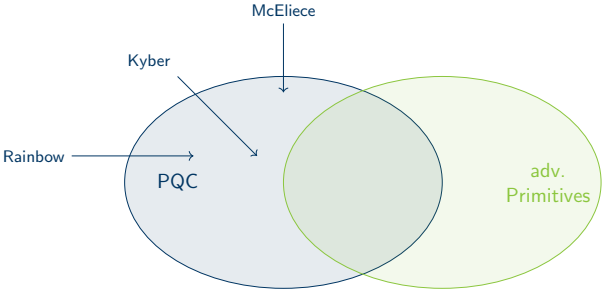
What is PQC?

- PQC = any cryptography that withstands attacks from large-scale quantum computers
- basic (asymmetric) primitives: PKE/KEM and signatures
- advanced primitives: (F)HE, ZKP etc.
- this talk: PQC (only basic primitives) in the embedded world



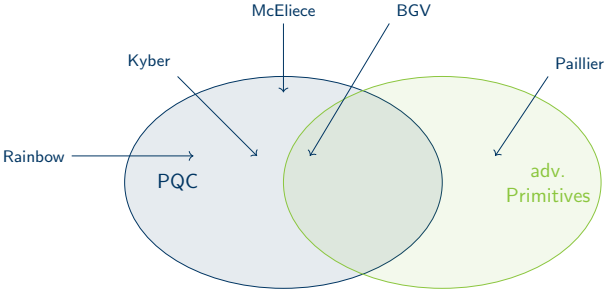
What is PQC?

- PQC = any cryptography that withstands attacks from large-scale quantum computers
- basic (asymmetric) primitives: PKE/KEM and signatures
- advanced primitives: (F)HE, ZKP etc.
- this talk: PQC (only basic primitives) in the embedded world



What is PQC?

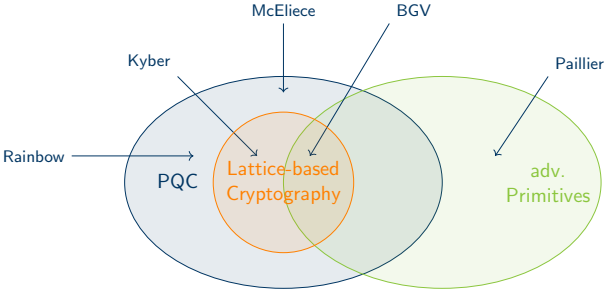
- PQC = any cryptography that withstands attacks from large-scale quantum computers
- basic (asymmetric) primitives: PKE/KEM and signatures
- advanced primitives: (F)HE, ZKP etc.
- this talk: PQC (only basic primitives) in the embedded world



What is PQC?

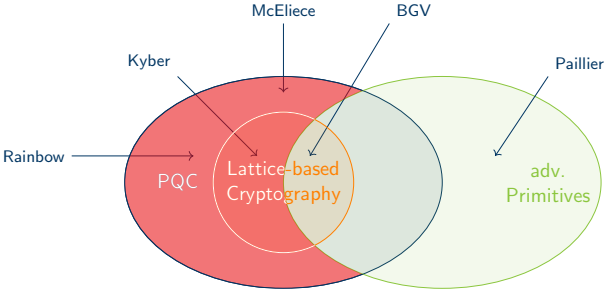
RUB

- PQC = any cryptography that withstands attacks from large-scale quantum computers
- basic (asymmetric) primitives: PKE/KEM and signatures
- advanced primitives: (F)HE, ZKP etc.
- this talk: PQC (only basic primitives) in the embedded world



What is PQC?

- PQC = any cryptography that withstands attacks from large-scale quantum computers
- basic (asymmetric) primitives: PKE/KEM and signatures
- advanced primitives: (F)HE, ZKP etc.
- this talk: PQC (only basic primitives) in the embedded world



The Embedded World

not embedded

servers, laptops

embedded

smartcards, distributed sensor systems (IoT, cars, industry), ...

"Every computer that has no fan."

PQC and Embedded Systems: Requirements

RUB

small

- software: program memory and stack usage
- FPGAs: slices, LUTs, FFs, DSPs, BRAMs
- ASICs: area

fast

- latency
- throughput

secure

- passive side-channel attacks
- active side-channel attacks

PQC and Embedded Systems: Requirements

RUB

small

- software: program memory and stack usage
- FPGAs: slices, LUTs, FFs, DSPs, BRAMs
- ASICs: area

fast

- latency
- throughput

secure

- passive side-channel attacks
- active side-channel attacks

Embedded Software Implementations: KEMs

scheme	NIST security level	key gen (avg. kcycles)	encaps (avg. kcycles)	decaps (avg. kcycles)	program memory (kB)	key gen stack (kB)	encaps stack (kB)	decaps stack (kB)	ref.
ECDH	–	327	906		8.9		$\leq$2		[1]
Kyber	1	442	542	494	15	2.7	2.8	2.8	[2]
Saber	1	422	591	581	20	3.2	3.1	3.1	[2]
NTRU	1	2867	565	538	192	21	14	15	[2]
sNTRUp	1	6714	631	486	238	92	13	16	[2]
SIKE	1	48265	78911	84275	30	6	6	7	[2]
BIKE	1	65551	4963	116657	35	44	32	91	[2]
McEliece	1	1430811	582	2707	621	115	1.4	18	[3]

- Kyber and Saber are competitive to ECDH!
- NTRU variants may be used for use cases without key generation
- McEliece suitable for encaps-only use cases
- McEliece: lower memory footprint possible by streaming in the public key from a master device [4]

Embedded Software Implementations: KEMs

scheme	NIST security level	key gen (avg. kcycles)	encaps (avg. kcycles)	decaps (avg. kcycles)	program memory (kB)	key gen stack (kB)	encaps stack (kB)	decaps stack (kB)	ref.
ECDH	–	327	906		8.9		≤2		[1]
Kyber	1	442	542	494	15	2.7	2.8	2.8	[2]
Saber	1	422	591	581	20	3.2	3.1	3.1	[2]
NTRU	1	2867	565	538	192	21	14	15	[2]
sNTRUp	1	6714	631	486	238	92	13	16	[2]
SIKE	1	48265	78911	84275	30	6	6	7	[2]
BIKE	1	65551	4963	116657	35	44	32	91	[2]
McEliece	1	1430811	582	2707	621	115	1.4	18	[3]

- Kyber and Saber are competitive to ECDH!
- NTRU variants may be used for use cases without key generation
- McEliece suitable for encaps-only use cases
- McEliece: lower memory footprint possible by streaming in the public key from a master device [4]

Embedded Software Implementations: KEMs

scheme	NIST security level	key gen (avg. kcycles)	encaps (avg. kcycles)	decaps (avg. kcycles)	program memory (kB)	key gen stack (kB)	encaps stack (kB)	decaps stack (kB)	ref.
ECDH	–	327	906		8.9		≤2		[1]
Kyber	1	442	542	494	15	2.7	2.8	2.8	[2]
Saber	1	422	591	581	20	3.2	3.1	3.1	[2]
NTRU	1	2867	565	538	192	21	14	15	[2]
sNTRUp	1	6714	631	486	238	92	13	16	[2]
SIKE	1	48265	78911	84275	30	6	6	7	[2]
BIKE	1	65551	4963	116657	35	44	32	91	[2]
McEliece	1	1430811	582	2707	621	115	1.4	18	[3]

- Kyber and Saber are competitive to ECDH!
- NTRU variants may be used for use cases without key generation
- McEliece suitable for encaps-only use cases
- McEliece: lower memory footprint possible by streaming in the public key from a master device [4]

Embedded Software Implementations: KEMs

scheme	NIST security level	key gen (avg. kcycles)	encaps (avg. kcycles)	decaps (avg. kcycles)	program memory (kB)	key gen stack (kB)	encaps stack (kB)	decaps stack (kB)	ref.
ECDH	–	327	906		8.9		≤2		[1]
Kyber	1	442	542	494	15	2.7	2.8	2.8	[2]
Saber	1	422	591	581	20	3.2	3.1	3.1	[2]
NTRU	1	2867	565	538	192	21	14	15	[2]
sNTRUp	1	6714	631	486	238	92	13	16	[2]
SIKE	1	48265	78911	84275	30	6	6	7	[2]
BIKE	1	65551	4963	116657	35	44	32	91	[2]
McEliece	1	1430811	582	2707	621	115	1.4	18	[3]

- Kyber and Saber are competitive to ECDH!
- NTRU variants may be used for use cases without key generation
- McEliece suitable for encaps-only use cases
- McEliece: lower memory footprint possible by streaming in the public key from a master device [4]

Embedded Software Implementations: Signature Schemes

scheme	NIST security level	key gen (avg. kcycles)	sign (avg. kcycles)	verify (avg. kcycles)	program memory (kB)	key gen stack (kB)	sign stack (kB)	verify stack (kB)	ref.
ECDSA	–	327	375	976	8.9		≤2		[1]
Dilithium	2	1597	4095	1572	18	38	49	36	[2]
Falcon	1	162463	38999	474	121	1.4	2.6	0.4	[2]
Picnic	1	60	303854	203717	81	0.8	32	32	[2]
SPHINCS+	1	16112	400443	22548	4.5	2.1	2.2	2.7	[2]
XMSS	(1)	243255	247726	3207	–	4.0	4.0	3.8	[5]

- Falcon well-suited for verify-only use cases
- Dilithium has very balanced performance – verify can run with <8 kB [6]

Embedded Software Implementations: Signature Schemes

RUB

scheme	NIST security level	key gen (avg. kcycles)	sign (avg. kcycles)	verify (avg. kcycles)	program memory (kB)	key gen stack (kB)	sign stack (kB)	verify stack (kB)	ref.
ECDSA	–	327	375	976	8.9		≤2		[1]
Dilithium	2	1597	4095	1572	18	38	49	36	[2]
Falcon	1	162463	38999	474	121	1.4	2.6	0.4	[2]
Picnic	1	60	303854	203717	81	0.8	32	32	[2]
SPHINCS+	1	16112	400443	22548	4.5	2.1	2.2	2.7	[2]
XMSS	(1)	243255	247726	3207	–	4.0	4.0	3.8	[5]

- Falcon well-suited for verify-only use cases
- Dilithium has very balanced performance – verify can run with <8 kB [6]

Embedded Software Implementations: Signature Schemes

scheme	NIST security level	key gen (avg. kcycles)	sign (avg. kcycles)	verify (avg. kcycles)	program memory (kB)	key gen stack (kB)	sign stack (kB)	verify stack (kB)	ref.
ECDSA	–	327	375	976	8.9		≤2		[1]
Dilithium	2	1597	4095	1572	18	38	49	36	[2]
Falcon	1	162463	38999	474	121	1.4	2.6	0.4	[2]
Picnic	1	60	303854	203717	81	0.8	32	32	[2]
SPHINCS+	1	16112	400443	22548	4.5	2.1	2.2	2.7	[2]
XMSS	(1)	243255	247726	3207	–	4.0	4.0	3.8	[5]

- Falcon well-suited for verify-only use cases
- Dilithium has very balanced performance – verify can run with <8 kB [6]

Hardware Implementations: Targets

RUB

FPGA

- "field-programmable gate array"
- reconfigurable hardware
- area metrics: lookup tables, flip flops, DSPs, block RAM
- speed metrics: latency, throughput

ASIC

- "application-specific integrated circuit"
- not reconfigurable
- area metrics: area in mm^2
- speed metrics: latency, throughput

SW / HW Codesign

- idea 1: new instructions for CPUs that speed up PQC schemes (instruction set extension)
- idea 2: whole co-processors
- proof of concept usually in FPGAs
- emerging: RISC-V is easily exensible

Hardware Implementations: Targets

RUB

FPGA

- "field-programmable gate array"
- reconfigurable hardware
- area metrics: lookup tables, flip flops, DSPs, block RAM
- speed metrics: latency, throughput

ASIC

- "application-specific integrated circuit"
- not reconfigurable
- area metrics: area in mm^2
- speed metrics: latency, throughput

SW / HW Codesign

- idea 1: new instructions for CPUs that speed up PQC schemes (instruction set extension)
- idea 2: whole co-processors
- proof of concept usually in FPGAs
- emerging: RISC-V is easily exensible

Hardware Implementations: Targets

FPGA

- "field-programmable gate array"
- reconfigurable hardware
- area metrics: lookup tables, flip flops, DSPs, block RAM
- speed metrics: latency, throughput

ASIC

- "application-specific integrated circuit"
- not reconfigurable
- area metrics: area in mm^2
- speed metrics: latency, throughput

SW / HW Codesign

- idea 1: new instructions for CPUs that speed up PQC schemes (instruction set extension)
- idea 2: whole co-processors
- proof of concept usually in FPGAs
- emerging: RISC-V is easily exensible

Hardware Implementations: Targets

RUB

FPGA

- "field-programmable gate array"
- reconfigurable hardware
- area metrics: lookup tables, flip flops, DSPs, block RAM
- speed metrics: latency, throughput

ASIC

- "application-specific integrated circuit"
- not reconfigurable
- area metrics: area in mm^2
- speed metrics: latency, throughput

SW / HW Codesign

- idea 1: new instructions for CPUs that speed up PQC schemes (instruction set extension)
- idea 2: whole co-processors
- proof of concept usually in FPGAs
- emerging: RISC-V is easily exensible

Easy: Fast, but large area.

Easy: Small area, but slow.

The art: Low area-latency product.

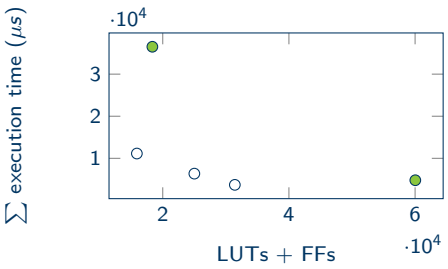
Hardware Implementations: FPGA Implementations

- Kyber [7]
- Saber [8]
- McEliece [9]
- Dilithium [10–14]
- Falcon [13]
- Rainbow [15]
- BIKE [16, 17]
- sNTRU Prime [18, 19]
- ...

FPGA Implementations: BIKE

- first set of implementations: feasibility shown, already high speed and low area variant
- second set: vast improvements in both dimensions, additional mid-range core

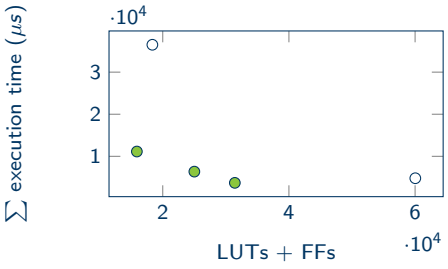
LUTs	FFs	KeyGen μs	Encaps μs	Decaps μs	Ref.
13k	5354	21903	1252	13349	[16]
53k	7035	2691	127	1972	[16]
12k	3896	3797	443	6896	[17]
20k	5008	1870	280	4210	[17]
26k	5426	1672	132	1892	[17]



FPGA Implementations: BIKE

- first set of implementations: feasibility shown, already high speed and low area variant
- second set: vast improvements in both dimensions, additional mid-range core

LUTs	FFs	KeyGen μs	Encaps μs	Decaps μs	Ref.
13k	5354	21903	1252	13349	[16]
53k	7035	2691	127	1972	[16]
12k	3896	3797	443	6896	[17]
20k	5008	1870	280	4210	[17]
26k	5426	1672	132	1892	[17]

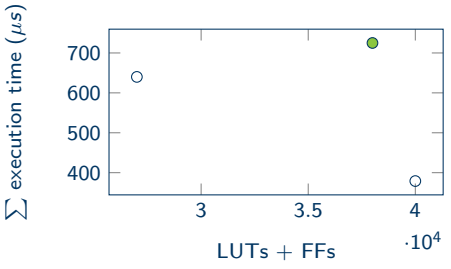


FPGA Implementations: Dilithium

RUB

- initial implementation [10] shows feasibility
- follow-up [11]: faster, supports all security levels
- novel idea [14]: run Saber and Dilithium with the same polynomial multiplier

LUTs	FFs	KeyGen μs	Sign μs	Verify μs	Ref.
27k	11k	134	470	121	[10]
30k	10k	43	290	46	[11]
18k	9k	71	494	75	[14]

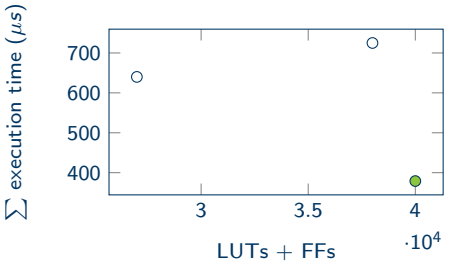


FPGA Implementations: Dilithium

RUB

- initial implementation [10] shows feasibility
- follow-up [11]: faster, supports all security levels
- novel idea [14]: run Saber and Dilithium with the same polynomial multiplier

LUTs	FFs	KeyGen μs	Sign μs	Verify μs	Ref.
27k	11k	134	470	121	[10]
30k	10k	43	290	46	[11]
18k	9k	71	494	75	[14]

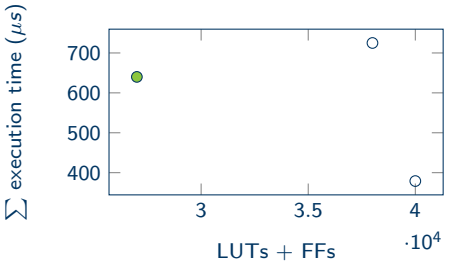


FPGA Implementations: Dilithium

RUB

- initial implementation [10] shows feasibility
- follow-up [11]: faster, supports all security levels
- novel idea [14]: run Saber and Dilithium with the same polynomial multiplier

LUTs	FFs	KeyGen μs	Sign μs	Verify μs	Ref.
27k	11k	134	470	121	[10]
30k	10k	43	290	46	[11]
18k	9k	71	494	75	[14]



Hardware Implementations: ASICs and SW/HW Codesigns

ASICs

- only few implementations published: Kyber [20], Saber [21]
- FPGA implementations already good indicator of hardware cost

HW / SW Codesign

- RISC-V instruction set extensions for finite field arithmetic: low-level acceleration applicable to many schemes
- multiple works on this [22–24]
- biggest improvement would be: SHA-3 accelerator (some lattice-based schemes have up to 86% hashing, hash-based up to 95% [2])

Implementations

RUB

Don't sleep on these numbers, there is progress.

PQC and Side-Channel Attacks

RUB

devices everywhere

PQC and Side-Channel Attacks

devices everywhere, attackers everywhere

PQC and Side-Channel Attacks

devices everywhere, attackers everywhere

security?

PQC and Side-Channel Attacks: Attacker Models

secure scheme (under certain assumptions)

$$\neq$$

secure implementation!

Kyber is mathematically secure under certain assumptions, but there are side-channel attacks that find the secret key.

PQC and Side-Channel Attacks: Attacker Models

RUB

Has an attacker pyhsical access to the device?

No? Then we still need execution time independent from secret data.

Side-Channel Attacks

RUB

no physical access		
timing depends on secrets	$\Rightarrow$	measure execution time

physical access, passive		
processing a 1 drains more power / emanates more electromagnetic radiation than a 0	$\Rightarrow$	measure power / EM

physical access, active
induce a fault, learn information about the secret key from the output

Side-Channel Attacks

RUB

no physical access		
timing depends on secrets	⇒	measure execution time

physical access, passive		
processing a 1 drains more power / emanates more electromagnetic radiation than a 0	⇒	measure power / EM

physical access, active
induce a fault, learn information about the secret key from the output

Side-Channel Attacks: Countermeasures

- CPU / hardware processes secrets
- attacker measures side-channel, finds out secrets
- hide secrets from CPU / hardware
- split secret into uniform random shares, perform computation on shares
- highly algorithm-specific!

Side-Channel Resistant Implementations

RUB

good: many schemes have masked SW implementations

Kyber [25, 26], Saber [27, 28], ...

Side-Channel Resistant Implementations

good: many schemes have masked SW implementations

Kyber [25, 26], Saber [27, 28], ...

bad: some of them are already broken again

Kyber [29], Saber [30]

Side-Channel Resistant Implementations

RUB

good: we already know some fault attacks

Kyber [31–33], Survey for lattice-based schemes [34]

Side-Channel Resistant Implementations

RUB

good: we already know some fault attacks

Kyber [31–33], Survey for lattice-based schemes [34]

bad: for some, it is unclear how to prevent them efficiently

Side-Channel Resistant Implementations

bad: for some components, we do not yet know how to mask them

Gaussian sampling? fixed-weigth sampling?

Side-Channel Resistant Implementations

bad: for some components, we do not yet know how to mask them

Gaussian sampling? fixed-weigth sampling?

good: there is progress

masking polynomial inversion [35]

Side-Channel Resistant Implementations

bad: micro-architectural leakage often kills secure-proven gadgets

Side-Channel Resistant Implementations

bad: micro-architectural leakage often kills secure-proven gadgets

bad: there are only few HW masked implementations

Side-Channel Resistant Implementations

bad: micro-architectural leakage often kills secure-proven gadgets

bad: there are only few HW masked implementations

more research necessary

Conclusion

Use cases must be planned carefully.

Memory – Bandwidth – Latency – Throughput – Attack Scenarios

Research just started in many fields.

It will narrow and intensify once there are standards.

Questions? Ideas? Suggestions?

Ask me, contact me: georg.land@rub.de

References I

[1] https://github.com/Emill. *P256-Cortex-M4: P256 ECDH and ECDSA for Cortex-M4, Cortex-M33 and other 32-bit ARM processors*. https://github.com/Emill/P256-Cortex-M4.

[2] Matthias J. Kannwischer et al. *PQM4: Post-quantum crypto library for the ARM Cortex-M4*. https://github.com/mupq/pqm4.

[3] Ming-Shing Chen and Tung Chou. "Classic McEliece on the ARM Cortex-M4". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* (2021), pp. 125–148.

[4] Johannes Roth, Evangelos G. Karatsiolis, and Juliane Krämer. "Classic McEliece Implementation with Low Memory Footprint". In: *Smart Card Research and Advanced Applications - 19th International Conference, CARDIS 2020, Virtual Event, November 18-19, 2020, Revised Selected Papers*. Ed. by Pierre-Yvan Liardet and Nele Mentens. Vol. 12609. Lecture Notes in Computer Science. Springer, 2020, pp. 34–49. DOI: 10.1007/978-3-030-68487-7_3. URL: https://doi.org/10.1007/978-3-030-68487-7%5C_3.

[5] Fabio Campos et al. "LMS vs XMSS: Comparison of Stateful Hash-Based Signature Schemes on ARM Cortex-M4". In: *Progress in Cryptology - AFRICACRYPT 2020 - 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20-22, 2020, Proceedings*. Ed. by Abderrahmane Nitaj and Amr M. Youssef. Vol. 12174. Lecture Notes in Computer Science. Springer, 2020, pp. 258–277. DOI: 10.1007/978-3-030-51938-4_13. URL: https://doi.org/10.1007/978-3-030-51938-4%5C_13.

[6] Ruben Gonzalez et al. "Verifying Post-Quantum Signatures in 8 kB of RAM". In: *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20-22, 2021, Proceedings*. Ed. by Jung Hee Cheon and Jean-Pierre Tillich. Vol. 12841. Lecture Notes in Computer Science. Springer, 2021, pp. 215–233. DOI: 10.1007/978-3-030-81293-5_12. URL: https://doi.org/10.1007/978-3-030-81293-5%5C_12.

[7] Yufei Xing and Shuguo Li. "A Compact Hardware Implementation of CCA-Secure Key Exchange Mechanism CRYSTALS-KYBER on FPGA". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.2 (2021), pp. 328–356. DOI: 10.46586/tches.v2021.i2.328-356. URL: https://doi.org/10.46586/tches.v2021.i2.328-356.

[8] Sujoy Sinha Roy and Andrea Basso. "High-speed Instruction-set Coprocessor for Lattice-based Key Encapsulation Mechanism: Saber in Hardware". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.4 (2020), pp. 443–466. DOI: 10.13154/tches.v2020.i4.443-466. URL: https://doi.org/10.13154/tches.v2020.i4.443-466.

References II

[9] Po-Jen Chen et al. "Complete and Improved FPGA Implementation of Classic McEliece". In: *IACR Cryptol. ePrint Arch.* (2022), p. 412. URL: https://eprint.iacr.org/2022/412.

[10] Georg Land, Pascal Sasdrich, and Tim Güneysu. "A Hard Crystal - Implementing Dilithium on Reconfigurable Hardware". In: *Smart Card Research and Advanced Applications - 20th International Conference, CARDIS 2021, Lübeck, Germany, November 11-12, 2021, Revised Selected Papers*. Ed. by Vincent Grosso and Thomas Pöppelmann. Vol. 13173. Lecture Notes in Computer Science. Springer, 2021, pp. 210–230. DOI: 10.1007/978-3-030-97348-3_12. URL: https://doi.org/10.1007/978-3-030-97348-3%5C_12.

[11] Cankun Zhao et al. "A Compact and High-Performance Hardware Architecture for CRYSTALS-Dilithium". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022.1 (2022), pp. 270–295. DOI: 10.46586/tches.v2022.i1.270-295. URL: https://doi.org/10.46586/tches.v2022.i1.270-295.

[12] Luke Beckwith, Duc Tri Nguyen, and Kris Gaj. "High-Performance Hardware Implementation of CRYSTALS-Dilithium". In: *International Conference on Field-Programmable Technology, (IC)FPT 2021, Auckland, New Zealand, December 6-10, 2021*. IEEE, 2021, pp. 1–10. DOI: 10.1109/ICFPT52863.2021.9609917. URL: https://doi.org/10.1109/ICFPT52863.2021.9609917.

[13] Luke Beckwith, Duc Tri Nguyen, and Kris Gaj. "High-Performance Hardware Implementation of Lattice-Based Digital Signatures". In: *IACR Cryptol. ePrint Arch.* (2022), p. 217. URL: https://eprint.iacr.org/2022/217.

[14] Aikata et al. "A Unified Cryptoprocessor for Lattice-based Signature and Key-exchange". In: *IACR Cryptol. ePrint Arch.* (2021), p. 1461. URL: https://eprint.iacr.org/2021/1461.

[15] Ahmed Ferozpuri and Kris Gaj. "High-speed FPGA Implementation of the NIST Round 1 Rainbow Signature Scheme". In: *2018 International Conference on ReConFigurable Computing and FPGAs, ReConFig 2018, Cancun, Mexico, December 3-5, 2018*. Ed. by David Andrews et al. IEEE, 2018, pp. 1–8. DOI: 10.1109/RECONFIG.2018.8641734. URL: https://doi.org/10.1109/RECONFIG.2018.8641734.

[16] Jan Richter-Brockmann, Johannes Mono, and Tim Güneysu. "Folding BIKE: Scalable Hardware Implementation for Reconfigurable Devices". In: *IEEE Trans. Computers* 71.5 (2022), pp. 1204–1215. DOI: 10.1109/TC.2021.3078294. URL: https://doi.org/10.1109/TC.2021.3078294.

[17] Jan Richter-Brockmann et al. "Racing BIKE: Improved Polynomial Multiplication and Inversion in Hardware". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022.1 (2022), pp. 557–588. DOI: 10.46586/tches.v2022.i1.557-588. URL: https://doi.org/10.46586/tches.v2022.i1.557-588.

References III

[18] Adrian Marotzke. "A Constant Time Full Hardware Implementation of Streamlined NTRU Prime". In: *Smart Card Research and Advanced Applications - 19th International Conference, CARDIS 2020, Virtual Event, November 18-19, 2020, Revised Selected Papers*. Ed. by Pierre-Yvan Liardet and Nele Mentens. Vol. 12609. Lecture Notes in Computer Science. Springer, 2020, pp. 3–17. DOI: 10.1007/978-3-030-68487-7_1. URL: https://doi.org/10.1007/978-3-030-68487-7%5C_1.

[19] Bo-Yuan Peng et al. "Streamlined NTRU Prime on FPGA". In: *IACR Cryptol. ePrint Arch.* (2021), p. 1444. URL: https://eprint.iacr.org/2021/1444.

[20] Mojtaba Bisheh-Niasar, Reza Azarderakhsh, and Mehran Mozaffari Kermani. "A Monolithic Hardware Implementation of Kyber: Comparing Apples to Apples in PQC Candidates". In: *Progress in Cryptology - LATINCRYPT 2021 - 7th International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, October 6-8, 2021, Proceedings*. Ed. by Patrick Longa and Carla Ràfols. Vol. 12912. Lecture Notes in Computer Science. Springer, 2021, pp. 108–126. DOI: 10.1007/978-3-030-88238-9_6. URL: https://doi.org/10.1007/978-3-030-88238-9%5C_6.

[21] Archisman Ghosh et al. "A 334uW 0.158mm2 Saber Learning with Rounding based Post-Quantum Crypto Accelerator". In: *IEEE Custom Integrated Circuits Conference, CICC 2022, Newport Beach, CA, USA, April 24-27, 2022*. IEEE, 2022, pp. 1–2. DOI: 10.1109/CICC53496.2022.9772859. URL: https://doi.org/10.1109/CICC53496.2022.9772859.

[22] Erdem Alkim et al. "ISA Extensions for Finite Field Arithmetic Accelerating Kyber and NewHope on RISC-V". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.3 (2020), pp. 219–242. DOI: 10.13154/tches.v2020.i3.219-242. URL: https://doi.org/10.13154/tches.v2020.i3.219-242.

[23] Tim Fritzmann, Georg Sigl, and Johanna Sepúlveda. "RISQ-V: Tightly Coupled RISC-V Accelerators for Post-Quantum Cryptography". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2020.4 (2020), pp. 239–280. DOI: 10.13154/tches.v2020.i4.239-280. URL: https://doi.org/10.13154/tches.v2020.i4.239-280.

[24] Guozhu Xin et al. "VPQC: A Domain-Specific Vector Processor for Post-Quantum Cryptography Based on RISC-V Architecture". In: *IEEE Trans. Circuits Syst. I Regul. Pap.* 67-I.8 (2020), pp. 2672–2684. DOI: 10.1109/TCSI.2020.2983185. URL: https://doi.org/10.1109/TCSI.2020.2983185.

[25] Daniel Heinz et al. "First-Order Masked Kyber on ARM Cortex-M4". In: *IACR Cryptol. ePrint Arch.* (2022), p. 58. URL: https://eprint.iacr.org/2022/058.

References IV

[26] Joppe W. Bos et al. "Masking Kyber: First- and Higher-Order Implementations". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.4 (2021), pp. 173–214. DOI: 10.46586/tches.v2021.i4.173-214. URL: https://doi.org/10.46586/tches.v2021.i4.173-214.

[27] Michiel Van Beirendonck et al. "A Side-Channel-Resistant Implementation of SABER". In: *ACM J. Emerg. Technol. Comput. Syst.* 17.2 (2021), 10:1–10:26. DOI: 10.1145/3429983. URL: https://doi.org/10.1145/3429983.

[28] Suparna Kundu et al. "Higher-order masked Saber". In: *IACR Cryptol. ePrint Arch.* (2022), p. 389. URL: https://eprint.iacr.org/2022/389.

[29] Mike Hamburg et al. "Chosen Ciphertext k-Trace Attacks on Masked CCA2 Secure Kyber". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.4 (2021), pp. 88–113. DOI: 10.46586/tches.v2021.i4.88-113. URL: https://doi.org/10.46586/tches.v2021.i4.88-113.

[30] Kalle Ngo et al. "A Side-Channel Attack on a Masked IND-CCA Secure Saber KEM Implementation". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.4 (2021), pp. 676–707. DOI: 10.46586/tches.v2021.i4.676-707. URL: https://doi.org/10.46586/tches.v2021.i4.676-707.

[31] Peter Pessl and Lukas Prokop. "Fault Attacks on CCA-secure Lattice KEMs". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.2 (2021), pp. 37–60. DOI: 10.46586/tches.v2021.i2.37-60. URL: https://doi.org/10.46586/tches.v2021.i2.37-60.

[32] Julius Hermelink, Peter Pessl, and Thomas Pöppelmann. "Fault-Enabled Chosen-Ciphertext Attacks on Kyber". In: *Progress in Cryptology - INDOCRYPT 2021 - 22nd International Conference on Cryptology in India, Jaipur, India, December 12-15, 2021, Proceedings*. Ed. by Avishek Adhikari, Ralf Küsters, and Bart Preneel. Vol. 13143. Lecture Notes in Computer Science. Springer, 2021, pp. 311–334. DOI: 10.1007/978-3-030-92518-5_15. URL: https://doi.org/10.1007/978-3-030-92518-5%5C_15.

[33] Jeroen Delvaux and Santos Merino Del Pozo. "Roulette: Breaking Kyber with Diverse Fault Injection Setups". In: *IACR Cryptol. ePrint Arch.* (2021), p. 1622. URL: https://eprint.iacr.org/2021/1622.

[34] Prasanna Ravi, Anupam Chattopadhyay, and Anubhab Baksi. "Side-channel and Fault-injection attacks over Lattice-based Post-quantum Schemes (Kyber, Dilithium): Survey and New Results". In: *IACR Cryptol. ePrint Arch.* (2022), p. 737. URL: https://eprint.iacr.org/2022/737.

[35] Markus Krausz et al. "Efficiently Masking Polynomial Inversion at Arbitrary Order". In: *IACR Cryptol. ePrint Arch.* (2022), p. 707. URL: https://eprint.iacr.org/2022/707.