Poster: On the use of hardware accelerators in QC-MDPC code-based cryptography

Andrea Galimberti*
Politecnico di Milano
Milano, Italy
andrea.galimberti@polimi.it

Davide Galli
Politecnico di Milano
Milano, Italy
davide11.galli@mail.polimi.it

Gabriele Montanaro
Politecnico di Milano
Milano, Italy
gabriele.montanaro@mail.polimi.it

William Fornaciari
Politecnico di Milano
Milano, Italy
william.fornaciari@polimi.it

Davide Zoni
Politecnico di Milano
Milano, Italy
davide.zoni@polimi.it

CCS CONCEPTS

• **Hardware → Hardware accelerators**; • **Security and privacy → Hardware security implementation**; *Public key encryption.*

KEYWORDS

Post-quantum cryptography, Code-based cryptography, Hardware accelerators, FPGA

ACM Reference Format:
Andrea Galimberti, Davide Galli, Gabriele Montanaro, William Fornaciari, and Davide Zoni. 2022. Poster: On the use of hardware accelerators in QC-MDPC code-based cryptography. In *19th ACM International Conference on Computing Frontiers (CF'22), May 17–19, 2022, Torino, Italy.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3528416.3530243

ACKNOWLEDGMENTS

This work was supported by the EU Horizon 2020 "TEXTAROSSA" project (Grant No. 956831).

1 INTRODUCTION

Public-key cryptography (PKC) allows exchanging keys over an insecure channel without sharing a secret key. However, quantum computers threaten to break traditional PKC, thus, to mitigate such risk, post-quantum cryptography (PQC) aims to develop cryptosystems that are secure against attacks from quantum and classical computers. BIKE [1] is a key encapsulation mechanism (KEM) based on quasi-cyclic moderate-density parity-check (QC-MDPC) codes that is a candidate within the NIST standardization process to identify a set of PQC algorithms [4]. Figure 1 depicts the key exchange between two client and server nodes, which requires the sequential execution of the key generation, encapsulation, and decapsulation KEM primitives. Key generation and decapsulation are performed on the client side, while encapsulation is carried out by the server. Despite the vast literature targeting efficient hardware support for BIKE, each proposal delivered computing platforms meant either to maximize performance or minimize resource utilization.

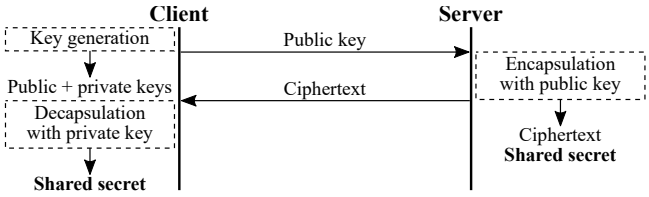

Figure 1: Key exchange implemented as a KEM.

2 CONTRIBUTIONS

This paper presents a complete hardware implementation of BIKE that targets the Xilinx Artix-7 family of FPGAs and supports client- and server-side KEM operations. The proposed architecture leverages a set of state-of-the-art configurable accelerators [2, 3, 6, 7] that implement the key operations of the KEM primitives. Our architecture is evaluated against the FPGA-based hardware [5] implementations of BIKE.

3 BIKE KEY ENCAPSULATION MECHANISM

Figure 2 details the key generation, encapsulation, and decapsulation primitives of BIKE. The *key generation* module receives as an input a random seed and outputs the public h and private H keys, as well as a random value σ. The key generation algorithm of BIKE requires performing sequentially pseudorandom generation, binary polynomial inversion, and binary polynomial multiplication.

The *decapsulation* module receives as inputs the private key H, σ, and the ciphertext c and it outputs the shared secret K. The decapsulation primitive of BIKE requires executing in sequence binary polynomial multiplication, QC-MDPC bit-flipping decoding, computation of SHA-3 hash digest, and pseudorandom generation.

The *encapsulation* module takes as inputs a random message m and the public key h and outputs the shared secret K and the ciphertext c. The encapsulation primitive of BIKE requires subsequently performing pseudorandom generation, binary polynomial multiplication, and computation of the SHA-3 hash function.

*Corresponding author

CF'22, May 17–19, 2022, Torino, Italy

ACM ISBN 978-1-4503-9338-6/22/05.
https://doi.org/10.1145/3528416.3530243

Algorithm 1 Key generation.

1: **function** $[H, \sigma, h]$ KeyGen $(seed)$
2: $H = \text{PRNG}(\text{SHAKE256}(seed))$;
3: $h_{0_{inv}} = \text{Invert}(h_0)$;
4: $h = h_1 \odot h_{0_{inv}}$;
5: $\sigma = \text{TRNG}\ ()$;

Algorithm 2 Encapsulation.

1: **function** $[K, c]$ Encaps (h, m)
2: $e = \text{PRNG}(\text{SHAKE256}(m))$;
3: $s = e_0 \oplus (e_1 \odot h)$;
4: $m' = m \oplus \text{Trunc}_{256}(\text{SHA3-384}(e))$;
5: $c = \{s, m'\}$;
6: $K = \text{Trunc}_{256}(\text{SHA3-384}(\{m, c\}))$;

Algorithm 3 Decapsulation.

1: **function** $[K]$ Decaps (H, σ, c)
2: $s' = h_0 \odot s$;
3: $e' = \text{Decode}\ (s', H)$;
4: $m'' = m' \oplus \text{Trunc}_{256}(\text{SHA3-384}(e'))$;
5: $a = (e' = \text{PRNG}(\text{SHAKE256}(m''))) \ ?\ m'' : \sigma$;
6: $K = \text{Trunc}_{256}(\text{SHA3-384}(\{a, c\}))$;

Figure 2: Algorithms for the key generation, encapsulation, and decapsulation primitives of BIKE.

Table 1: Area results, expressed in terms of look-up tables (LUT), flip-flops (FF), and block RAM (BRAM).

Module	Code	Our						BIKE [5]					
		Artix-7 35			Artix-7 200			Lightweight (LW)			High-speed (HS)		
		LUT	FF	BRAM	LUT	FF	BRAM	LUT	FF	BRAM	LUT	FF	BRAM
Client	*B1*	20663	15128	42	121238	48889	358	11454	4602	14	43084	610	39
	B3	18420	16464	49	118513	50270	358	-	-	-	-	-	-
Server	*B1*	19531	11997	41	89011	45091	277.5	6730	3298	3	14829	3471	10
	B3	19605	13490	42	68265	36944	236.5	-	-	-	-	-	-
Available		20800	41600	50	134600	269200	365	20800	41600	50	134600	269200	365

Table 2: Execution times, expressed in milliseconds.

Module	Code	Our		BIKE [5]	
		Artix-7 35	Artix-7 200	Lightweight (LW)	High-speed (HS)
Client	*B1*	9.03	0.51	35.25	4.66
	B3	39.55	1.51	-	-
Server	*B1*	0.04	0.02	1.25	0.13
	B3	0.09	0.04	-	-

4 EXPERIMENTAL EVALUATION

Experimental setup - The proposed architectures, as well as the reference BIKE hardware [5] implementations, target the security levels 1 (*B1*) and 3 (*B3*) of BIKE. The proposed architectures were described in SystemVerilog and implemented in Xilinx Vivado 2020.2, targeting Artix-7 FPGAs and a clock frequency of 91 MHz. All the identified instances satisfed the area constraints given by the available resources on the target FPGAs and the timing requirements.
Area results - The proposed architectures make wide use of BRAM, allowing them to scale from Artix-7 35 to Artix-7 200 FPGAs, as shown in Table 1. The results show that BRAM memories are the most used resources, except in the Artix-7 35 *B1* client and the *B1* and *B3* servers, while the high-speed reference [5] uses the resources available on larger FPGAs unefficiently, employing only 32%, 2%, and 11% of the LUT, FF, and BRAM of the Artix-7 200 chip.
Performance results - As reported in Table 2, our Artix-7 35 $B1$ client is around three times faster than the lightweight reference [5], while the Artix-7 200 client is around nine times faster than the high-speed one [5]. Both the Artix-7 35 and 200 servers outperform the high-speed $B1$ instance of [5].

5 CONCLUSIONS

This paper presented an effective hardware support for BIKE on FPGA targets. Compared to the reference hardware, our Artix-7 35 and Artix-7 200 clients were 3 and 9 times faster than the lightweight and high-speed instances of [5], considering the $B1$ use case. Moreover, the proposed server instance outperformed the high-speed reference by at least three times.

REFERENCES

[1] Nicolas Aragon, Paulo S. L. M. Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Güneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, Valentin Vasseur, and Gilles Zémor. 2021. BIKE: Bit Flipping Key Encapsulation - Round 3 Submission. https://bikesuite.org/files/v4.2/BIKE_Spec.2021.09.29.1.pdf.

[2] Alessandro Barenghi, William Fornaciari, Andrea Galimberti, Gerardo Pelosi, and Davide Zoni. 2019. Evaluating the Trade-offs in the Hardware Design of the LEDAcrypt Encryption Functions. In *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. 739–742. https://doi.org/10.1109/ICECS46596.2019.8964882

[3] Andrea Galimberti, Gabriele Montanaro, and Davide Zoni. 2022. Efficient and scalable FPGA design of GF(2m) inversion for post-quantum cryptosystems. *IEEE Trans. Comput.* (2022), 1–1. https://doi.org/10.1109/TC.2022.3149422

[4] National Institute of Standards and Technology (NIST) - U.S. Department of Commerce. 2020. NISTIR 8309, Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process. https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf.

[5] Jan Richter-Brockmann, Johannes Mono, and Tim Guneysu. 2021. Folding BIKE: Scalable Hardware Implementation for Reconfigurable Devices. *IEEE Trans. Comput.* (2021). https://doi.org/10.1109/TC.2021.3078294

[6] Davide Zoni, Andrea Galimberti, and William Fornaciari. 2020. Efficient and Scalable FPGA-Oriented Design of QC-LDPC Bit-Flipping Decoders for Post-Quantum Cryptography. *IEEE Access* 8 (2020), 163419–163433. https://doi.org/10.1109/ACCESS.2020.3020262

[7] Davide Zoni, Andrea Galimberti, and William Fornaciari. 2020. Flexible and Scalable FPGA-Oriented Design of Multipliers for Large Binary Polynomials. *IEEE Access* 8 (2020), 75809–75821. https://doi.org/10.1109/ACCESS.2020.2989423