Invited Paper: Instruction Set Extensions for Post-Quantum Cryptography

Brohet, Marco; Valencia, Felipe; Regazzoni, Francesco

Publication date
2023
Document Version
Author accepted manuscript
Published in
2023 IEEE/ACM International Conference On Computer Aided Design (ICCAD)

Invited Paper: Instruction Set Extensions for Post-Quantum Cryptography

Marco Brohet*, Felipe Valencia† and Francesco Regazzoni*‡
* Informatics Institute, University of Amsterdam, The Netherlands, email: {m.j.a.brohet, f.regazzoni}@uva.nl
† Firmware and Security for Connected Devices, CSEM, Switzerland, email: andres.valencia@csem.ch
‡ Università della Svizzera italiana, Switzerland, email: francesco.regazzoni@usi.ch

***Abstract*—Quantum computing is one of the latest breakthroughs in the field of computer science, having the potential of breaking the underlying assumptions of public-key cryptography. With the National Institute of Standards and Technology (NIST) having announced that lattice-based KYBER as Key Encapsulation Mechanism (KEM) and DILITHIUM and FALCON as digital signatures are going to be standardized as the first Post-Quantum Cryptography (PQC) schemes, the scientific community needs to investigate how to efficiently implement these new primitives to ensure a smooth transition. We review in this work the state-of-the-art in Instruction Set Extensions (ISEs) for the lattice-based PQC schemes to be standardized. We categorize them into three groups. Firstly, tightly-integrated implementations that aim to be small and only accelerate the core functions, secondly more generic and bigger ISEs that target more lattice operations, and thirdly a special class that focuses on vectorized processing. While we observe promising results in improving on runtime and energy consumption, the memory footprint is often overlooked in the evaluation, even though this is a serious issue in PQC where keys, ciphertexts and signatures tend to be larger. Additionally, we envision that more generic lattice-based ISEs will surface, and that side-channel and fault attacks will become more important.**

I. INTRODUCTION

Quantum computing has emerged as the most recent breakthrough in the field of computing, with early versions already starting to be commercialized [1]. Quantum computers work with quantum bits (qubits), which contain the superposition of states '0' and '1' simultaneouly. However, once they are measured, they collapse to classical bits with a 50% probability. When evaluating a funtion in the quantum state, it is evaluated for all possible superposition states, giving an advantage to the quantum computer over a classical one. Still, quantum algorithms must increase the probability to collapse to the correct result when measuring the quantum state. This makes quantum computer especially useful for NP problems.

While the quantum era brings many opportunities, Shor [2] presented an algorithm capable of solving interger factorization problem and discrete logarithm problem in polynomial time, which destroys the hardness assumption used in public-key cryptography. This would effectively make protocols for securing network connections, device authentication and Secure Boot, to mention a few, renderless. Therefore, the National Institute of Standards and Technology (NIST) started the Post-Quantum Cryptography (PQC) competition to select the next-generation primitives that cannot be broken by quantum-computational power, but that can be executed by classical computers. Its third round came to the conclusion to standardize four candidates, out of which three are founded on lattice theory: KYBER, DILITHIUM and FALCON [3].

As it is now more clear which PQC schemes are going to be standardized, the Cybersecurity and Infrastructure Security Agency (CISA), the National Security Agency (NSA) and NIST released an advisory urging organizations to prepare for the adoption of PQC algorithms [4]. To facilitate a smooth migration, it is necessary to make sure that the software can run them as efficiently as possible under the constraints of their use case. For classical cryptography, we have seen that instruction set extensions were proposed as a way to accelerate their usability [5], [6]. This raises the question to which extent such hardware specialization, or more generally hardware/software co-design, is needed for a successful adoption of PQC. Recent work on software implementations [7]–[9] has suggested that especially the increased memory footprint due to larger PQC keys, ciphertexts and signatures is troublesome.

In this work, we review the state-of-the-art in ISEs for the lattice-based PQC schemes that are going to be standardized. Our work does not focus on accelerating the building blocks of those primitives, but rather at how such acceleration can be exposed to the software. Here, we use "acceleration" not only to refer to speedups, but generally to improve on a certain metric or a group thereof. We envision that our review aids processor designers to understand how quantum-resistant cryptography effectively can effectively be made available. In summary, our work contributes the following:

1) We review state-of-the-art implementations of ISEs for the lattice-based PQC cryptosystems that are going to be standardized, providing an overview and summarizing them in Table I.
2) We show that while runtime and energy consumption of those ISEs have seen improvements, the memory footprint is most often not part of the evaluation and therefore unknown.
3) Besides evaluating the memory usage, we recommend to look for ways to share resources or combine acceleration while remaining "crypto-agile", and to also consider classical threat models, such as side-channel and fault attacks.

II. Lattice-based Cryptography

Lattice-based cryptography is a family of cryptographic algorithms that rely on the hardness of lattice problems. A lattice $\mathcal{L}$ is a discrete set of points in the space $\mathbb{R}^n$ with a periodic structure. $\mathcal{L}$ can be expressed [10] as the linear combination of basis vectors $\beta = \{b_1, b_2, ..., b_N\}$ as:

$$\mathcal{L} = \sum x_i b_i, \forall\ x_i \in Z, 1 \leqslant i \leqslant N$$

Ideal lattices are lattices defined in a quotient ring, where:

- $\mathbb{Z}[x]$ and $\mathbb{R}[x]$: set of polynomials with integer and real coefficients, respectively
- $\mathbb{Z}^n[x]$: set of polynomials in $\mathbb{Z}[x]$ with degree smaller than n. Same notation applies for $\mathbb{R}^n[x]$.
- $\mathbb{Z}_q^n[x]$: set of polynomials in $\mathbb{Z}^n[x]$ which coefficients are in the residue field $\mathbb{Z}/q\mathbb{Z}$, for instance, coefficients in $[0, q-1]$
- $\mathcal{R}$: quotient ring $\mathbb{Z}^n[x]/ < f >$ where f is a monic (leadign coeffcient is 1) and irreducible polynomial. Similarly $\mathcal{R}_q = \mathbb{Z}_q^n[x]/ < f >$

In practice ideal lattices reduce memory footprint and allow to speed up the polynomial multiplication using Number Theoretic Transform (NTT). NTT is the fast Fourier transform defined in a ring. NTT reduces the polynomial multiplication from $\mathcal{O}(n^2)$ to $\mathcal{O}(n \log n)$.

Two problems, and their variants, are used to build cryptographic primitives: Closest Vector Problem (CVP) and Shortest Vector Problem (SVP). Generally, cryptosystems are defined based on variants of CVP and SVP, for instance, Learning With Errors (LWE) problem (variant of CVP) and Short Integer Solution (SIS) problem (equivalent to SVP). LWE problem consists in solving a system of linear equations that have been disturbed by an error vector. Given $(A, b = A * x + e \bmod q)$ is necessary to find $x \in \mathbb{Z}^m$ with $A \in \mathbb{Z}_q^{m \times n}$. m, n, q are fixed parameters of the problem and e is an error vector taken from a random probability distribution χ in $\mathbb{Z}_q$ [11]. SIS consists in finding $x \in \mathbb{Z}^m$ such that $A * x = 0 \bmod q$. Schemes based on LWE and SIS tend to consume large amount of memory, therefore it is possible to move to ideal lattices, which are lattices defined in a ring [12].

Additionally to arithmetic operations, hash functions and random samplers are very important modules in LBC. Hash functions are used to give resistance to Adaptive Chosen-Ciphertext Attacks using the Fujisaki-Okamoto transform [13]. Random samplers generate the error vectors in cryptographic schemes and they can be discrete Gaussian, uniform and binomial samplers.

A. Kyber

Kyber is a Key Exchange Mecanism (KEM) based on the LWE problem in module lattices [14] with IND-CCA2 security. KEM is a technique to encapsulate a symmetric key using an asymmetric cryptographic algorithm. Kyber is built using a Public Key Scheme (PKE) secured with the Fujisaki and Okamoto transform, which is implemented using functions from SHA3 family. Kyber speeds up the polynomial multiplication with the NTT and by definition the algorithm assumes the constant matrix in NTT domain.

Kyber uses SHA3-256, SHA3-512, SHAKE-128, and SHAKE-256, for hashing the data in different places. All of these functions rely on the Keccak primitive making easy their implementation. The random samples are taken from a centered binomial distribution and also, more noise is inserted in a compression phase [15].

B. Dilithium

Dilithium is a signature algorithm based on *Fiat-Shamir with Aborts approach* [16], and together with Kyber forms a Cryptographic Suite for Algebraic Lattices (CRYSTALS). A digital signature is a scheme that guarantees the integrity and authenticity of a message. Dilithium was designed under the paradigms of being easy to implement securely, small signature and keys, conservative parameters and modular structure [17].

Dilithium operates in the ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$, where for all security levels $q = 2^{23} - 2^{13} + 1$ and $n = 256$. Operations are performed in modular lattices, as in Kyber. Dilithium submission to NIST contest presents three security levels. The difference between the three parameter sets lies on the public matrix size, the maximum possible value for a key component and for the multiplication ciphertext times key.

Similar to Kyber, the polynomial multiplication is accelerated in dilithium using NTT and SHAKE128 and SHAKE256 are used as eXtendable Output Functions (XOF). Dilithium uses uniform samples at the cost of doubling key and ciphertext size.

C. Falcon

Falcon is digital signature algorithm design to reduce the size of the combination between the public key and digital signature. It is based on NTRU lattices, in *hash-and-sign* paradigm and in Fast Fourier sampling [18].

Main modules of Falcon are the FFT (Fast Fourier Transform) for computing the private key, NTT for public key operations, bimodal Gaussing sampler, ChaCha20 as the PRNG and SHAKE-256 as XOF for all the security levels.

III. Instruction Set Extensions

Tailoring the processor's design to the needs of a specific use-case is a topic that has already been studied before [28], [29]. This practice is also sometimes referred to as software/hardware co-design, to emphasize that the additions to the processor's architecture determines how the program code needs to be organized and vice versa. Exposure of the hardware functionality to the code is usually governed through Instruction Set Extensions (ISEs). In this section, we explore the different types of ISEs that we encountered in our review of the literature.

Table I

COMPARISON OF THE STATE-OF-THE-ART ISES FOR KYBER, DILITHIUM, AND FALCON. RELATIVE IMPROVEMENT OR OVERHEAD WITH RESPECT TO THE UNMODIFIED CORE IS PRESENTED. — MEANS NOT REPORTED, ∼ MEANS THAT NO BASELINE WAS GIVEN, SO NO SPEEDUP CAN BE CALCULATED.

Ref.	Cipher	RISC-V core	Code size	Runtime			FPGA			ASIC		
				Key-Gen	Enc/ Sign	Dec/ Verify	Model	#LUTs	Energy (mJ)	Techn.	Area (kGE)	Energy (mJ)
[19]	KYBER	CVA6	—	36%	47%	63%	Xilinx ZCU106	5%	40–60%	—	—	—
[19]	DILITHIUM	CVA6	—	15%	28%	19%	Xilinx ZCU106	6%	20–30%	—	—	—
[20]	KYBER	VexRiscv	+25%	20%	22%	31%	Xilinx Artix-35T	—	—	—	—	—
[21]	KYBER	PicoRV32	—	73%	73%	73%	Xilinx Virtex-7	∼	—	—	—	—
[21]	DILITHIUM	PicoRV32	—	64%	64%	64%	Xilinx Virtex-7	∼	—	—	—	—
[21]	FALCON	PicoRV32	—	83%	83%	83%	Xilinx Virtex-7	∼	—	—	—	—
[22]	KYBER	CV32E40P	26%	91%	91%	90%	Xilinx Zynq-7000	59%	—	UMC 65 nm	59%	84%
[23][1]	KYBER	RV32	—	—	—	—	—	—	—	TSMC 40 nm	—	98%
[23][1]	DILITHIUM	RV32	—	90%	87%	86%	—	—	—	TSMC 40 nm	—	∼
[24]	DILITHIUM	CV32E40P	25%	83%	83%	83%	Xilinx UltraScale+	48%	—	GF 22 nm	50%	92%
[24]	FALCON	CV32E40P	37%	—	—	62%	Xilinx UltraScale+	48%	—	GF 22 nm	50%	57%
[25][1]	KYBER	SCR1		96%	92%	86%	—	—	—	TSMC 28 nm	2546%	99%
[26][1]	KYBER	Rocket		98%	97%	92%	Xilinx Artix-7	—	—	TSMC 28 nm	501%	∼
[26][1]	DILITHIUM	Rocket		98%	97%	95%	Xilinx Artix-7	—	—	TSMC 28 nm	501%	—
[27]	KYBER	Ibex	—	—	—	—	Xilinx UltraScale+	1896%	—	—	—	—

A. *Tightly-coupled accelerators*

In the case of tightly-coupled accelerators, the functionality is deeply integrated with the pipeline of the computer architecture. Functions implemented this way are usually small. Otherwise, it could increase the length of the pipeline and in the end slow down the processor in general. On the other hand, keeping the functionality could mean that they are less specialized, but perhaps generalizable for different use-cases or algorithms.

B. *Loosely-coupled accelerators*

The opposite approach is too implement the accelerators in a loosely-coupled fashion. Instead of being part of the pipeline, the functionality is implemented as a separate co-processor that is connected via a bus to the pipeline. A downside is that this introduces a larger communication overhead. To overcome, the functionality needs to be larger than for tightly-coupled accelerators. Besides, in this setting, an increase in the pipeline's length is also less likely. This approach fits better when the entire algorithm should be implemented in hardware.

[1]The reported baseline was Arm Cortex-M4 instead of the unmodified processor.

C. *Vectorized accelerators*

While the previous types were distinguishable through their connection to the core, the following category is actually more based on the type of data. While scalar data can only process one data item at a time, with vectorization allows for multiple data items to be processed concurrently. However, doubling the processing power usually means doubling the amount of functional units, which translates to a larger chip area. The latter could even lead to a higher energy consumption, unless the speedup is sufficiently significant.

IV. STATE-OF-THE-ART ISES

This section provides an overview of the various types of ISEs for lattice-based PQC that we encountered in literature. We do not aim to directly compare one work to another, but rather observe from a high level which design choices can be made and their benefits and downsides. To remain concise,

we only report for the smallest parameter sets, i.e. KYBER-512, DILITHIUM-II and FALCON-512. Several implementations were developed before the third round concluded and therefore also included candidates that were not accepted in the third round. Information on those candidates is here excluded. Detailed information about the experiments and the results, which we omit in the text, can be found in Table I.

A. Small extensions

Three works have attempted to keep the number of added instructions as small as possible. Nannipieri *et al.* [19] developed two sets of ISEs, one for KYBER and one for DILITHIUM with five instructions each. The authors profiled the reference implementation to identify the parts mostly responsible for the runtime overhead. Based on their analysis, ISEs were developed to perform multiplication, reduction, twiddle factor selection and (inverse) NTT. Although the Keccak permutation also had a significant impact, it was deemed too costly to implement this in hardware.

Alkim *et al.* [20] tried a more generic approach. Driven by their observation that many PQC proposals depend on small finite fields, the authors developed ISEs for finite-field addition, subtraction multiplication with NTT and Barrett reduction. A speedup of 20 to 35% could be measured on 32-bit RISC-V. The FPGA logic overhead of 6% and a 0.4 MHz decrease of the maximum clock frequency, but most interestingly the code size had to increase with 25% due to loop unrolling.

An even more minimal approach was taken by Karabulut and Aysu [21]. Instead of extending the instruction set, the authors proposed an architecture—named RANTTT—which almost transparently performs the acceleration on the software running. When RANTT's tracker recognizes that an NTT software implementation is running, the controller enables several optimizations that are specific to NTT. As it is very generic, the implementations of all three proposals can be accelerated, showing on 32-bit RISC-V speedups in the order of 60% to 80%. 417 LUTs more were required, although no baseline was given.

B. Larger extensions

The following three designs go beyond accelerating NTT to achieve higher speedups, thereby specializing into lattice-based. Fritzmann *et al.* [22] proposed a total of 29 instructions, ranging from butterfly operations to binomial sampling, and integrated tightly into the pipeline. While the runtime was improved significantly, resource usage on both FPGA and ASIC went up with 59%, although the energy consumption still went down by 84%.

Instead of tightly integrating the extra logic into the pipeline, Banerjee *et al.* [23] presented Sapphire, a loosely-coupled co-processor for lattice-based cryptography. The implemented functionality includes polynomial arithmetic and a Keccak core to support the sampling. In addition, an efficient memory architecture for the NTT was developed, which makes use of single-port instead of dual-port SRAMs to reduce the area by an equivalent of 124 kGE. An ASIC was fabricated at TSMC's 40-nm technology and its energy and power consumption was extensively evaluated. However, instead of using the original 32-bit RISC-V core as a baseline, the Arm Cortex-M4 was taken, which is not in line with the other works discussed here. Finally, the authors note that Sapphire can resist timing-based and simple power-based side-channel attacks.

Opting for a hybrid solution, Karl et al. [24] implemented SHAKE128 tightly-coupled in the core, and NTT and polynomial as a loosely-coupled module. The authors used these ISEs to accelerate the full scheme of DILITHIUM, and for FALCON only verification. The idea is that FALCON verification is more lightweight, and thus it would make sense to accelerate it for verify-only use-cases, such as Secure Boot. Runtime overhead and energy consumption were improved significantly, for DILITHIUM in the order of 80% to 90% and for FALCON 60%. The memory usage was also measured, which was reduced for DILITHIUM by 2% and for FALCON by 39%.

C. Vectorized extensions

While the following three designs are also targeting lattice-based cryptography in general, they additionally have the interesting property of being vectorized. In other words, the data is processed in parallel, thereby reducing the runtime at the cost of more occupied area. Xin *et al.* [25] proposed VPQC, a vectorized co-processor to accelerate sampling and NTT for cryptosystems based on Ring-LWE and Module-LWE. The 12 added instructions mainly drive the vector arithmetic units, vectorized samplers and a vector Keccak core. Also here, the Cortex-M4 is taken as the baseline instead of the original processor. Nevertheless, the fact that the co-processor is 942 kGE and as such 25 times as large as the original 32-bit RISC-V core already shows that its area overhead is very high.

Zhao *et al.* [26] aimed to optimize more by specifically targeting cryptosystems based on Module-LWE, exploiting its mathematical properties to enable better instruction- and data-level parallelism. Therefore, this co-processor, which can be controlled by 14 custom instructions, is used to accelerate KYBER and DILITHIUM. The Cortex-M4 was used as the baseline instead of the original core. The area occupation is better than VPQC's, going from 942 to 581 kGE.

Instead of making a separate co-processor, Li et al. [27] started from a RISC-V processor that already supported the official RISC-V vector specification, and then extended the vector implementation to support NTT, inverse NTT and coefficient-wise multiplication for KYBER. 12 instructions were specifically for KYBER, while 4 were generically for finite field arithmetic. However, the speedup was only measured for the individual modules and not for the full KYBER operations, so a direct comparison is not possible. In terms of FPGA resource occupation, the simplest 4-lane configuration uses 19 times more LUTs than original Ibex.

V. Discussion

As can be seen from Table I, the primary targets of the ISEs have been to reduce the runtime and to a lesser extent the energy consumption. NTT has been identified as one of the most important part to accelerate, which we see in all ISEs and especially the smaller ones. All ISEs have also been developed on RISC-V. Although the aim of this review was not to make a direct comparison among the ISEs, it is interesting that several works [23], [25], [26] have not considered their original processor without modifications as their baseline, but rather the Cortex-M4. This makes a comparison more difficult.

A. Memory footprint

Recent work on software implementations [7]–[9] has shown the memory footprint to be mostly the bottleneck, especially when run on resource-constrained devices. What can be concluded from Table I, is that the memory footprint is generally not considered when evaluating the performance of an ISE. Impact on the code size is measured more often, and Alkim *et al.* [20] already showed that ISEs might not always be beneficial for this figure of merit. The first step would be to include this in the analysis of PQC to see whether current ISEs are already capable of reducing the memory usage significantly. If not, then Fournaris *et al.* [30] suggested possible strategies to employ for reducing the memory footprint, also specifically for lattice-based PQC implementations.

B. Crypto-agility and sharing resources

Several works have attempted to create accelerators or hardware modules supporting multiple algorithms or use-cases. As the conclusion of the third round was only made last year, we expect that in the future more accelerators supporting all three lattice-based algorithms will be developed. Especially the smaller, tightly-integrated extensions should be more easily adaptable and in that sense more *agile*. An interesting direction to explore is to efficiently combine the PQC acceleration with acceleration for other cryptosystems, such as homomorphic encryption [31] and identity-based encryption [32]. On the other hand, NIST is already looking for other digital signature candidates, including non-lattice-based. Thus, it would also be interesting to explore the combined acceleration of algorithm that are less related to each other in a mathematical sense.

C. Other threat models

Designers should not forget that PQC is meant to be executed on classical computers. Therefore, also more classical attacks need to be considered. Banerjee *et al.* [23] already hinted that their ISE is executed in constant time and can resist simple power side-channel attacks. Fritzmann *et al.* [33] already proposes, but also the impact of fault attacks should be analyzed.

VI. Related Work

Bernstein and Lange [34] presented an introductory review of the PQC field. As their work predated NIST's first competition deadline, it did not discuss the candidates themselves. Rather, the focus was on the mathematical families of quantum-resistant problems that could spawn proposals. More recently, Fernández-Caramés [35] also discussed other PQC proposals, but primarily focused on the second-round candidates of the NIST competition. The performance results of the software and FPGA implementations indicated that especially for IoT the energy consumption needs to be reduced.

Zooming in on lattice-based, Nejatollahi *et al.* [36] comprehensively discussed the theory, associated quantum-hard problems and PQC proposals based on them. The survey concluded with an overview of current software and hardware implementations. Asif [37] narrowed the focus further down to lattice-based PQC for IoT. Interestingly, the latter survey suggests to perform NTT in a memory-efficient manner to reduce the memory consumption.

Lastly, instead of focusing on PQC primitives and their implementations, Kampanakis and Lepoint [38] discussed the issues surrounding their use in secure network protocols. For instance, current mitigations against amplification attacks do not take into account the increased size of PQC handshakes.

VII. Conclusions

The quantum era, when arriving, will not only bring opportunities, but also challenges. The PQC field was born because of Shor's quantum algorithm [2] that breaks the mathematical problems contemporary public-key cryptography relies on. Lattice-based schemes have shown to be promising for being quantum-resistant, and now that Kyber, Dilithium and Falcon will be standardized by NIST, we need to prepare for transitioning to these new PQC schemes. ISEs have been shown to be effective in reducing the runtime overhead and energy consumption, and also reduce the pressure on memory. Our review has shown that the acceleration of NTT, Keccak and other operations are effective, and we look forward to more memory-reducing, crypto-agile or fault-resisisting software/hardware co-designs.

References

[1] C. Q. Choi, "An IBM quantum computer will soon pass the 1,000-qubit mark," *IEEE Spectrum*, Dec. 2022. [Online]. Available: https://spectrum.ieee.org/ibm-condor

[2] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.

[3] G. Alagic, D. Apon, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, and D. Smith-Tone, "Status report on the third round of the NIST post-quantum cryptography standardization process," National Institute of Standards and Technology (U.S.), Tech. Rep., Sep. 2022.

[4] Cybersecurity and Infrastructure Security Agency (CISA), National Security Agency (NSA), and National Institute of Standards and Technology (NIST), "Quantum-readiness: Migration to post-quantum cryptography," U.S. Cybersecurity and Infrastructure Security Agency, Tech. Rep., Aug. 2023, version 1.0. [Online]. Available: https://media.defense.gov/2023/Aug/21/2003284212/-1/-1/0/CSI-QUANTUM-READINESS.PDF

[5] S. Gueron, "Intel® Advanced Encryption Standard (AES) New Instructions set," Intel Corporation, Tech. Rep., Sep. 2012, rev. 3.01. [Online]. Available: https://www.intel.com/content/www/us/en/developer/articles/tool/intel-advanced-encryption-standard-aes-instructions-set.html

[6] RISC-V, "RISC-V Cryptography Extensions Volume I," RISC-V, Tech. Rep., Feb. 2022, version v1.0.1. [Online]. Available: https://github.com/riscv/riscv-crypto/releases/tag/v1.0.1-scalar

[7] M. Schöffel, F. Lauer, C. C. Rheinländer, and N. Wehn, "Secure IoT in the era of quantum computers—where are the bottlenecks?" *Sensors*, vol. 22, no. 7, p. 2484, Mar. 2022.

[8] G. Tasopoulos, J. Li, A. P. Fournaris, R. K. Zhao, A. Sakzad, and R. Steinfeld, "Performance evaluation of post-quantum TLS 1.3 on resource-constrained embedded systems," in *Information Security Practice and Experience*. Springer International Publishing, 2022, pp. 432–451.

[9] G. Tasopoulos, C. Dimopoulos, A. P. Fournaris, R. K. Zhao, A. Sakzad, and R. Steinfeld, "Energy consumption evaluation of post-quantum TLS 1.3 for resource-constrained embedded devices," in *Proceedings of the 20th ACM International Conference on Computing Frontiers (CF)*, May 2023.

[10] D. P. Chi, J. W. Choi, J. S. Kim, and T. Kim, "Lattice based cryptography for beginners," Cryptology ePrint Archive, Paper 2015/938, 2015. [Online]. Available: https://eprint.iacr.org/2015/938

[11] O. Regev, "The learning with errors problem," in *Proc. of 25th IEEE Annual Conference on Computational Complexity (CCC)*, 2010.

[12] V. Lyubashevsky, "Lattice-based identification schemes secure under active attacks," in *Public Key Cryptography –PKC 2008*. Springer Science and Business Media, 2008, pp. 162–179.

[13] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," *Journal of Cryptology*, vol. 26, no. 1, pp. 80–101, Jan. 2013.

[14] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle, "CRYSTALS-kyber: A CCA-secure module-lattice-based KEM," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, Apr. 2018.

[15] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-KYBER: Algorithm specifications and supporting documentation," CRYSTALS, Tech. Rep., Aug. 2021, version 3.02. [Online]. Available: https://pq-crystals.org/kyber/resources.shtml

[16] V. Lyubashevsky, "Fiat-shamir with aborts: Applications to lattice and factoring-based signatures," in *ASIACRYPT '09*, 2009.

[17] S. Bai, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-DILITHIUM: Algorithm specifications and supporting documentation," CRYSTALS, Tech. Rep., Feb. 2021, version 3.1. [Online]. Available: https://pq-crystals.org/dilithium/resources.shtml

[18] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "FALCON: Fast-fourier lattice-based compact signatures over NTRU," FALCON, Tech. Rep., Oct. 2020, version 1.2. [Online]. Available: https://falcon-sign.info/falcon.pdf

[19] P. Nannipieri, S. D. Matteo, L. Zulberti, F. Albicocchi, S. Saponara, and L. Fanucci, "A RISC-V post quantum cryptography instruction set extension for number theoretic transform to speed-up CRYSTALS algorithms," *IEEE Access*, vol. 9, pp. 150 798–150 808, 2021.

[20] E. Alkim, H. Evkan, N. Lahr, R. Niederhagen, and R. Petri, "ISA extensions for finite field arithmetic," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 219–242, Jun. 2020.

[21] E. Karabulut and A. Aysu, "RANTT: A RISC-V architecture extension for the number theoretic transform," in *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, Aug. 2020.

[22] T. Fritzmann, G. Sigl, and J. Sepúlveda, "RISQ-V: tightly coupled RISC-V accelerators for post-quantum cryptography," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 4, pp. 239–280, Aug. 2020.

[23] U. Banerjee, T. S. Ukyab, and A. P. Chandrakasan, "Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 17–61, Aug. 2019.

[24] P. Karl, J. Schupp, T. Fritzmann, and G. Sigl, "Post-quantum signatures on RISC-V with hardware acceleration," *ACM Transactions on Embedded Computing Systems*, Jan. 2023.

[25] G. Xin, J. Han, T. Yin, Y. Zhou, J. Yang, X. Cheng, and X. Zeng, "VPQC: A domain-specific vector processor for post-quantum cryptography based on RISC-V architecture," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 8, pp. 2672–2684, Aug. 2020.

[26] Y. Zhao, R. Xie, G. Xin, and J. Han, "A high-performance domain-specific processor with matrix extension of RISC-V for module-LWE applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 7, pp. 2871–2884, Jul. 2022.

[27] H. Li, N. Mentens, and S. Picek, "A scalable SIMD RISC-V based processor with customized vector extensions for CRYSTALS-Kyber," in *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC '22)*, 2022, p. 733–738.

[28] C. Galuzzi and K. Bertels, "The instruction-set extension problem," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 4, no. 2, pp. 1–28, May 2011.

[29] N. Paulino, J. C. Ferreira, and J. M. P. Cardoso, "Improving performance and energy consumption in embedded systems via binary acceleration: A survey," *ACM Computing Surveys*, vol. 53, no. 1, pp. 1–36, Feb. 2020.

[30] A. P. Fournaris, G. Tasopoulos, M. Brohet, and F. Regazzoni, "Running longer to slim down: Post-quantum cryptography on memory-constrained devices," in *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, Jul. 2023.

[31] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *Advances in Cryptology – ASIACRYPT 2017*. Springer International Publishing, 2017, pp. 409–437.

[32] C. P. Rentería-Mejia and J. Velasco-Medina, "Lattice-based cryptoprocessor for CCA-secure identity-based encryption," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 7, pp. 2331–2344, Jul. 2020.

[33] T. Fritzmann, M. V. Beirendonck, D. B. Roy, P. Karl, T. Schamberger, I. Verbauwhede, and G. Sigl, "Masked accelerators and instruction set extensions for post-quantum cryptography," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 414–460, Nov. 2021.

[34] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, Sep. 2017.

[35] T. M. Fernández-Caramés, "From pre-quantum to post-quantum IoT security: A survey on quantum-resistant cryptosystems for the internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6457–6480, Jul. 2020.

[36] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-quantum lattice-based cryptography implementations," *ACM Computing Surveys*, vol. 51, no. 6, pp. 1–41, Jan. 2019.

[37] R. Asif, "Post-quantum cryptosystems for Internet-of-Things: A survey on lattice-based algorithms," *IoT*, vol. 2, no. 1, pp. 71–91, Feb. 2021.

[38] P. Kampanakis and T. Lepoint, "Vision paper: Do we need to change some things?" in *Security Standardisation Research*. Springer Nature Switzerland, 2023, pp. 78–102.