IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 September 2022, accepted 6 December 2022, date of publication 15 December 2022,
date of current version 21 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3229521

RESEARCH ARTICLE

The Feasibility of the CRYSTALS-Kyber Scheme for Smart Metering Systems

VINÍCIUS L. R. DA COSTA[1], ÂNDREI CAMPONOGARA[2], (Member, IEEE), JULIO LÓPEZ[3], AND MOISÉS V. RIBEIRO[1], (Senior Member, IEEE)

[1]Electrical Engineering Department, Federal University of Juiz de Fora (UFJF), Juiz de Fora 36036-900, Brazil
[2]Electrical Engineering Department, Federal University of Paraná, Curitiba 81530-000, Brazil
[3]Institute of Computing, Campinas State University (UNICAMP), Campinas 13083-970, Brazil

Corresponding author: Vinícius L. R. da Costa (vinicius.lagrota@engenharia.ufjf.br)

This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) under Grant 001, in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) under Grant 404068/2020-0 and Grant 314741/2020-8, in part by the Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) under Grant APQ-03609-17 and Grant TEC-PPM 00787-18, and in part by the Instituto Nacional de Energia Elétrica (INERGE).

ABSTRACT This paper investigates the feasibility of a quantum-secure Key Encapsulation Mechanism (KEM) in hardware constrained Smart Meter (SM) equipment. In this sense, the Cryptographic Suite for Algebraic Lattices (CRYSTALS)-Kyber scheme, the new standard for post-quantum Public-Key Encryption (PKE) systems chosen by the post-quantum cryptography (PQC) standardization process, is scrutinized and implemented in a System-on-a-Chip (SoC) Field Programmable Gate Array (FPGA) device. Experimental results show that the proposed hardware/software co-design implementation of the CRYSTALS-Kyber scheme in an SoC FPGA device reduces its execution time by around 70% compared to its software implementation. Moreover, the hardware resource analysis shows that the proposed hardware/software co-design implementation of the CRYSTALS-Kyber scheme in an SoC device is feasible for SM equipment.

INDEX TERMS Data security, post-quantum cryptography, system-on-a-chip, smart metering.

I. INTRODUCTION

Smart Grids (SGs) have been recognized as a necessary evolution of electric power systems for handling the increasing dependence on electricity. Certainly, SGs improve energy efficiency, reduce non-technical losses, and allow asset monitoring, observability, and reliability by integrating renewable energy sources, artificial intelligence, and information and communication technologies [1]. However, electric power systems are one of the most complex artificial systems. Consequently, academic institutions, R&D facilities, and industries have put research and development efforts into advancing technologies since SGs offer significant benefits to all stakeholders [2], [3], [4].

In this sense, it is worth bringing attention to SM equipment, which is one of the leading enablers of SGs, since they augment the capacity of monitoring, observing, and controlling assets of consumers and prosumers. However, data in SM contain sensitive information (e.g., users' energy consumption profiles). Consequently, security and privacy awareness in SMs have become a concern as existing security vulnerabilities can result in effective breaches and non-authorized access to SMs data. For instance, a successful attack might exploit flaws that allow tampering or manipulating SMs data without being detected [5]. In this regard, research focusing on security issues, countermeasures, and information privacy are critical in SM networks, see [6], [7] and references therein.

Regarding SMs, ensuring the security and privacy of data traveling through SM networks is of utmost importance. It is recognized that cryptographic schemes based on PKE and KEM, are one of the most employed methods to achieve this goal. These schemes rely on mathematical problems that demand high computing capacity to be solved in a short period. For instance, factoring integers or computing discrete logarithms (e.g., Rivest-Shamir-Adleman (RSA) and Elliptic Curve Cryptography (ECC)) [8]. However, such cryptographic schemes might be easily broken with the

The associate editor coordinating the review of this manuscript and approving it for publication was Shafiqul Islam.

advent of powerful and stable quantum computers, which are devices that relies on quantum physics to perform computations.

To advance data security in the era of quantum computing, the National Institute for Standards and Technology (NIST) started a process to standardize quantum-secure cryptographic schemes [9], the so-called PQC schemes. A PQC scheme is designed to be secure against a cryptanalytic attack by a quantum computer (as well as classical computers). In this sense, a few investigations have discussed the viability of PQC schemes, which have been presented in the NIST standardization process, in hardware-constrained equipment, such as Advanced RISC Machines (ARMs) and FPGAs. While [10] discussed a hardware implementation of the CRYSTALS-Kyber scheme in an ARM Cortex-M4 device, [11] detailed a hardware implementation of the CRYSTALS-Kyber scheme in an Artix-7 FPGA. Note that the ARM Cortex-M4-based implementation is easy to integrate into any application and demands a long execution time. On the contrary, the FPGA-based implementation is not easy to integrate into applications and demands a short execution time. In the sequel, [12] proposed a hardware/software co-design implementation of a generic instruction set for PQC schemes. Even though improvements are achieved using a hardware/software co-design implementation, the generic instruction set for PQC schemes lacks performance. Moreover, it demands excessive memory usage when a specific scheme is considered. Based on the existing literature, there is still a lack of investigations of hardware/software co-design implementation addressing specific PQC schemes when hardware-constrained equipment, such as SM, is considered.

In this sense, this paper investigates the feasibility of the CRYSTALS-Kyber scheme as the KEM for ensuring quantum-secure symmetric key exchange between nodes in SMs networks. The choice of such PQC scheme relies on the fact that it is one of the fastest and most lightweight PQC schemes in the NIST standardization process [13]. In this regard, the implementation of the CRYSTALS-Kyber scheme in a SoC FPGA device is detailed. The main contributions of this work are as follows:

- A presentation of an optimized hardware/software co-design implementation of the CRYSTALS-Kyber scheme on a hardware-constrained SM equipment that uses an SoC FPGA device.
- A discussion on the components of the CRYSTALS-Kyber scheme that allows us to come up with an advantageous trade-off between complexity and execution time when a hardware-constrained SM equipment based on an SoC FPGA device is considered.
- A performance comparison between the proposed hardware/software[1] co-design implementation against the software implementation (i.e., a benchmark implementation) of the CRYSTALS-Kyber scheme in terms of execution time. And, an analysis of the hardware resource usage demanded by the proposed hardware/software co-design implementation of the CRYSTALS-Kyber scheme in an SoC FPGA device.

Based on the attained results, our major findings are as follows:

- Evaluating the performance of all designed functions for the software implementation of the CRYSTALS-Kyber scheme, we recognize that a few functions must be hardware-implemented to come up with a low execution time. Furthermore, some of these functions share a few routines, which can be hardware implemented only once to provide additional hardware resource savings.
- The attained results show that the execution time of the proposed hardware/software co-design implementation of the CRYSTALS-Kyber scheme is around 70 % faster than the software implementation. Also, the proposal maintains compliance with the reference implementations.
- Since the CRYSTALS-Kyber scheme is a fast and lightweight PQC scheme based on module lattices, it can be efficiently embedded in SM equipment through a hardware-constrained implementation. Consequently, it is a good candidate for improving security in SM networks under classic and quantum computers.

The rest of this paper is organized as follows: Section II overviews cryptographic schemes applied to hardware-constrained devices; Section III formulates the investigated problem; Section IV details the CRYSTALS-Kyber scheme; Section V addresses the software implementation and Section VI the proposed hardware/software co-design implementation of this scheme; Section VII discusses numerical results of hardware resource usage, execution time, and performance comparison; Section VIII presents the limitations of the study; and, finally, Section IX asserts conclusive remarks.

A. NOTATION

The notation used in this work is the same one used in the supporting documentation of the CRYSTALS-Kyber scheme [14]. The input and output of functions of the CRYSTALS-Kyber scheme are byte arrays, and $\mathcal{B} = \{0, \ldots, 255\}$ is a set of unsigned integer of 8-bits or a byte. Also, we assume that $\mathcal{B}^K$ is the set of byte arrays of length K and $\mathcal{B}^*$ is the set of byte arrays of arbitrary length (i.e., a byte stream). The $||$ symbol denotes the concatenation of two-byte arrays. $\lceil x \rceil = \min\{m \in \mathbb{Z} | m \geq x\}$ is the ceiling function. Given a byte array a and a non-negative integer k, $a + k$ is the byte array starting at byte k of a (with indexing starting at zero). The rings R and R_q are denoted by $\mathbb{Z}[X]/(X^n + 1)$ and $\mathbb{Z}_q[X]/(X^n + 1)$, respectively, where $\mathbb{Z}$ is the set of all integers, $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ is the quotient ring of integers modulo q, and $n = 2^{n'-1}$ such that X^{n+1} is the $2^{n'}$-th cyclotomic polynomial. The values of n, n', and q are fixed

[1] In our study, hardware implementation refers to the use of an FPGA device while the software implementation is related to the use of a processor (softcore or hardcore) device.

throughout this document to $n = 256$, $n' = 9$, and $q = 3329$, respectively. Uppercase and lowercase bold letters are used for matrices and vectors, respectively, while $\mathbf{A}^T$ and $\hat{\mathbf{A}}$ are the transposed and the Number Theoretic Transform (NTT) of the matrix $\mathbf{A}$, respectively.

II. LITERATURE REVIEW

It is well-established that SM networks exchange sensitive information from consumers, prosumers, and utilities. Consequently, security and privacy awareness are important concerns that must be carefully addressed not only today but also in the future, which will be characterized by the presence of quantum computers. Regarding the security of SM data traveling through SM networks, we can rely on Physical Layer Security (PLS) and cryptography to guarantee information secrecy [15].

The PLS is a low-complexity technique that exploits characteristics of data communication channels to guarantee information secrecy [16], [17], [18]. Consequently, it can only be implemented in the physical layer, which is designed for connecting users sharing the same channel resources. PLS can make use of two approaches. The first approach is key-less because it seeks to degrade the Signal-to-Noise Ratio (SNR) of the eavesdropper for the intended receiver. The second approach aims at exploring the random characteristics of the data communication channel for extracting random keys and then providing secrecy. However, PLS has upsides such as low processing usage and no need for key management, and it requires that the eavesdropper channel be uncorrelated from the legitimate channel [19]. Also, due to the limited bandwidth used to transmit information in SM networks, it may not be possible to extract long keys, leading to a security breach [18].

Regarding cryptography targeting SM networks, there are different approaches to providing data security, especially when hardware-constrained devices apply. Currently, traditional cryptographic schemes are widely recommended and used to protect sensitive data through some guidelines and security standards, such as AGA-12 [20] and IEC 62351[2] [21]. However, these schemes may not be efficient enough for embedding in hardware-constrained equipment [22]. Philips et al. [23] proposed an enhanced-RSA scheme to authenticate SMs equipment in SM networks, while [24] introduced a lightweight anonymous key distribution based on ECC to provide anonymity to SMs equipment. In [25], the authors presented a feasible implementation of a cryptographic scheme that certifies the SM's read of fine-grained electricity using a low-cost MSP430 microcontroller. Aiming to improve secrecy in SM networks, [26] discussed the use of homomorphic cryptography, in which data are stored in the cloud, and any calculation required is performed directly on encrypted data. Unfortunately, it requires powerful servers when it needs to handle a large amount of data. Other techniques and approaches to provide secrecy in SM networks can be found in [27] and [28].

Traditional cryptographic schemes can provide information security in SM networks. However, the advent of quantum computers imposes that information security can not be guaranteed anymore because a polynomial-time quantum algorithm, called Shor's algorithm [29], can quickly solve traditional problems (i.e., discrete logarithm problems and integer factorization problems) on which current cryptographic schemes rely. Even more, if an eavesdropper is collecting and storing data from SM networks, then, in the future, it might be capable of deciphering it and accessing sensitive information. Aiming to overcome this problem, in 2017, the NIST opened a call for the standardization of PKE/KEM and Digital Signature Algorithm (DSA) based on PQC schemes because they are unbreakable, as far as is known, by both quantum and classical computer attacks. In July 2022, the NIST announced the selected algorithms: CRYSTALS-Kyber in the PKE/KEM category and CRYSTALS-Dilithium, Falcon, and SPHINCS+ in the DSA category [9]. Notice that all of them are lattice-based schemes, except SPHINCS+, which is a hash-based scheme.

Considering the PQC, [30] presented a concise security comparison of key agreement protocols between PQC primitives and traditional problems. Also, [31] proposed a lattice-based PKE and KEM to provide mutual authentication between SMs and a neighborhood gateway. Regarding PQC implementations based on lattices, [32] presented a complete survey highlighting the most relevant implementations using software, hardware, and hardware/software co-design techniques.

In [33], the authors presented a software implementation of CRYSTALS-Kyber, FrodoKEM, and NewHope using Graphics Processing Unit (GPU) seeking higher performance, while [34] proposed pure hardware implementation of the CRYSTALS-Kyber scheme using FPGA focusing on high performance and seeking to reuse resources. Note that [33] and [34] may not be suitable for SM networks due to their high hardware resource usage and high-cost platforms. In contrast, [35] presented a lightweight PKE lattice-based scheme on small 8-bit ATXmega128 and 32-bit ARM Cortex-M0. To do so, the authors replaced the Gaussian noise distribution with a uniform binary error distribution, reducing key and ciphertext sizes at the cost of performance. Next, in [36], it is proposed a new compact Learning With Errors (LWE) scheme suitable for ultra-low-cost devices, such as the MSP430. Moreover, the authors in [37] proposed a lightweight implementation of the NTRU Prime using a high-cost FPGA.

Besides the hardware and software implementations previously discussed, the literature also reports hardware/software co-design implementations. For instance, [38] detailed a RISC-V co-processor for lattice-based cryptography using hardware to accelerate the NTT transform and hash generation using an SoC FPGA device. Also, [12] discussed an Instruction Set Architecture (ISA) for lattice-based

[2]IEC 62351 recommends RSA and ECC schemes to protect power data.

cryptography based on a so-called RISQ-V architecture implemented on an SoC FPGA device and Application-Specific Integrated Circuit (ASIC). Furthermore, [39] and [40] presented a crypto-processor for PQC schemes based on lattices, fabricated in TSMC 40nm and 28nm low-power CMOS process, respectively. Finally, as far as the authors know, the only implementation that adopts SoC FPGA devices focusing on only one specific scheme is addressed in [41]. This study details the implementation of the CRYSTALS-Dilithium using a softcore processor and hardcore processor. In both implementations, the NTT and inverse NTT are hardware accelerated. However, [41] lacks a comprehensive analysis regarding other bottlenecks in the CRYSTALS-Dilithium scheme, which could also have been hardware accelerated. Also, a relevant discussion about the overhead and bottleneck in the data communication between the FPGA and processor are missing.

The literature review shows a lack of studies that evaluate a specific PQC scheme along with its bottlenecks and optimization using a hardware/software co-design approach, in which hardware accelerates the main bottlenecks when a hardware-constrained SoC FPGA device suitable for SM networks is applied. In this sense, the following sections focus on a hardware/software co-design implementation of the CRYSTALS-Kyber scheme for SM networks.

III. PROBLEM FORMULATION

SMs are multitasking devices capable of monitoring and controlling the bidirectional energy flow, generating data about consumption and production beyond the unidirectional power billing functionality of traditional meters [42]. The quasi-real-time data collections performed by SM equipment are of utmost importance for stakeholders in the electricity sector since they enable effective energy planning and improve the reliability, efficiency, observability, and stability of electric power systems. These are only available due to the connectivity capability of SM equipment, which allows data to be exchanged among electricity stakeholders (i.e., utilities, regulators, consumers, and prosumers).

As data are regularly exchanged between SM equipment (i.e., consumers and prosumers) and Meter Data Management System (MDMS) (i.e., utilities), security and privacy awareness arise. Indeed, SM data can reveal consumers' or prosumers' behaviors (e.g., lifestyle, daily routine, and economic status). In other words, these data are sensitive information [43]. Consequently, utilities and consumers are increasingly concerned about security breaches in SM networks. Furthermore, as might be expected, more serious security risks are observed if the data communication between SMs and MDMSs relies on non-dedicated data networks (i.e., the Internet). In these scenarios, the introduction of security measures for preserving privacy and enabling SMs equipment to perform their tasks safely is necessary.

Several security threats are related directly or indirectly to the consumers, prosumers, and electric utilities in SM networks. Protection against threats in SM networks can be

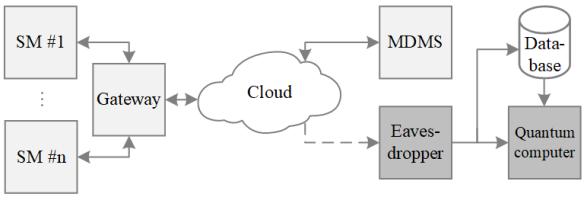


FIGURE 1. A typical scenario in which an eavesdropper performs a sniffing attack and makes use of a quantum computer to decipher the overheard SM data.

divided into three main areas: (i) privacy threats (e.g., misuse of private data); (ii) threats to system-level security (e.g., credential compromise and denial of service attacks); and (iii) threats to services (e.g., cloning, location, and migration of SMs equipment) [28]. Consequently, cryptography is one of the most applied techniques to protect SM sensitive data against attackers when data transmission over SM networks applies [17], [44], as already extensively discussed in Section II.

More specifically, regarding the long-term secure mechanism for SM networks, PQC schemes are more appropriate because of the imminent introduction of quantum computers. Currently, a powerful quantum computer has not been accomplished yet; however, the Sycamore quantum-processor, created by Google's Artificial Intelligence division, has achieved an experimental realization of quantum supremacy[3] for specific computation tasks [45]. Consequently, PQC schemes are tools to withstand security threats that a malicious attacker using a quantum computer represents.

In this regard, the scenario we are interested in is illustrated in Figure 1. It addresses the security of SM data exchanged among consumers/prosumers (i.e., SM equipment) and electric utilities (i.e., MDMSs) under the presence of eavesdroppers, which are supposed to be a classical or quantum computer and capable of performing sniffer attacks[4] in SM networks. In this scenario, a security breach arises if any node of an SM network could not use a PQC scheme. As SM equipment is designed with reduced processing power due to constraints in their commercialization price, a severe security problem arises if a PQC scheme can not be implemented. It is worth stating that PQC schemes are more complex than standard cryptography schemes, such as RSA and ECC, as they require more processing power and storage capacity. An SoC-based SM equipment may overcome not only the hardware constraint but also offer enough processing power to deal with time-consuming parts of the PQC scheme. These parts could be hardware-implemented while keeping the remaining less time-consuming ones in software, originating the proposed hardware/software co-design implementation. Hereafter, for simplicity, the proposed hardware/software

[3]Quantum supremacy is the experimental demonstration that a quantum computer can perform a task that a classic computer cannot in a reasonable amount of time.

[4]Sniffer attack means that the eavesdropper captures data packets in the network traffic.

implementation will refer to the proposed hardware/software co-design implementation.

In this sense, our study aims to answer the following research questions: *How to design a hardware/software implementation that is capable of accelerating the most time-consuming parts of the CRYSTAL-Kyber scheme for enabling PQC in SM networks? What kind of gains can a hardware/software implementation offer compared to software implementation?* The following sections aim to answer these research questions, which are critical for designing secure SM networks in the era of quantum computers.

IV. BACKGROUND OF THE CRYSTALS-KYBER SCHEME

Lattice-based cryptographic schemes rely on the worst-case hardness of lattice problems. If one succeeds in solving it, even by a slight chance, one may solve any instance of a particular lattice problem. As the average-case instance is at least as hard as the worst-case instance of a related lattice problem, these lattice-based cryptographic schemes bring a strong notion of security [46]. Recently, significant advances have been achieved concerning the lattice-based cryptography area. For instance, a new class of lattices, called ideal lattices, proposed by Lyubashevsky [47], [48], has emerged as one of the most relevant in the area. The idea behind ideal lattices is related to the notion of an ideal in a particular algebraic structure (e.g., polynomial rings).

Regarding LWE, previous proposals of LWE-based cryptographic schemes were based either on LWE (e.g., Frodo [49]), which can be seen as the standard one, or Ring-Learning With Errors (R-LWE) (e.g., NewHope [50]), which is a very structured variant of LWE. The LWE-based cryptography is related to the lattice problems since a reduction exists from the LWE problem to certain hard lattice problem. However, the exact complexity of these approximate solutions to lattice problems is yet to be discovered in many cases. Since no efficient classical or quantum algorithm has been found yet to solve lattice problems, despite significant research effort, there is an optimism that the problem is quantum-secure.

Moreover, the LWE problem is easily scalable and does not require additional structure. Nonetheless, these advantages come at the cost of a significant decrease in performance. An alternative is the R-LWE problem, which has an additional structure based on polynomial rings. As a result, it offers improvement in speed and key/ciphertext sizes; however, it might facilitate malicious attacks and be difficult to scalable. Recent cryptographic schemes, which are based on the so-called Module-Learning With Errors (M-LWE), offer a trade-off between LWE and R-LWE. Moreover, the M-LWE problem is less structured than the R-LWE problem, which, according to the recent cryptanalytic progress, seems that practical attacks are less likely to succeed against PQC schemes relying on the M-LWE problem than on the R-LWE one [51]. The CRYSTALS-Kyber scheme [14], relying on the M-LWE problem, presents much better scalability and similar performance in comparison to the R-LWE problem.

On the NIST standardization process, the lattice-based cryptographic schemes correspond to more than 40% of the total submitted candidate schemes. Simplicity and capacity of performing operations in parallel (e.g., addition, multiplication, and modular reduction) heavily influenced such submissions. The CRYSTALS-Kyber scheme [14], [51], relying on the M-LWE problem, was designed seeking high-performance (e.g., NewHope) without losing flexibility (e.g., Frodo). As a result, three different versions of the CRYSTALS-Kyber scheme are available: CRYSTALS-Kyber-512, -768, and -1024. They aim to ensure a security level equivalent to AES-128, AES-192, and AES-256. Because of its characteristics, the CRYSTALS-Kyber scheme is the scheme standardized by the NIST standardization process in the PKE/KEM category. Software implementations of it in high-end Intel CPUs are found in [52].

The theoretical background of the M-LWE problem and how it applies to the CRYSTALS-Kyber scheme is briefly outlined in Subsection IV-A.

A. THE LEARNING WITH ERRORS PROBLEM

The security of the CRYSTALS-Kyber scheme relies on the hardness of the M-LWE problem. Although, the M-LWE problem is a variant of the R-LWE one, which in turn has its roots in the LWE problem. According to Regev [53], the LWE problem requests the recovery of a secret $s \in \mathbb{Z}_q^n$, relying on the knowledge of a sequence of ''approximate'' random linear equations on s. Formally: set a size parameter $n > 1$, a modulus $q \geqslant 2$, and an error probability distribution χ on $\mathbb{Z}_q$. Let $A_{s,\chi}$ on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ be the probability distribution obtained by choosing a vector $\mathbf{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $\epsilon \in \mathbb{Z}_q$ according to χ, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + \epsilon)$, where additions are performed in $\mathbb{Z}_q$. Therefore, an algorithm solves the LWE problem with modulus q and error distribution χ if, for any $s \in \mathbb{Z}_q^n$, given an arbitrary number of independent samples from $A_{s,\chi}$ it outputs $\mathbf{s}$ (with high probability).

The R-LWE problem is similar to the LWE problem; however, the former works with elements over polynomial rings rather than over the integers, as proposed by Lyubashevsky [48]. Note that [48] defined the R-LWE problem based on the proof that it is also hard to differentiate a variant of the LWE problem distribution from the uniform one over specific polynomial rings. For example, if someone sets the ring R_q to a polynomial ring with a dimension greater than 1, then the PQC scheme is based on the hardness of the M-LWE problem [46].

B. THE CRYSTALS-KYBER SCHEME AS A KEY ENCAPSULATION MECHANISM

The CRYSTALS-Kyber scheme [14] can be used as PKE and KEM. PKE is based on the PKE key pair generation, encryption, and decryption algorithms, described in Algorithms 1, 2, and 3, respectively. On the other hand, KEM, which is constituted by KEM key pair generation, encapsulation, and decapsulation, as described in

TABLE 1. Parameter values for the CRYSTALS-Kyber scheme.

CRYSTALS-Kyber	n	k	q	η_1	η_2	d_u	d_v
512	256	2	3329	3	2	10	4
768	256	3	3329	2	2	10	4
1024	256	4	3329	2	2	11	5

Algorithms 4, 5, and 6 respectively, is constructed via a slightly modified Fujisaki–Okamoto transform [54], which is based on PKE.

The values of the parameters used by Algorithms 1 to 6 are specified in Table 1. Note that the parameter $n = 256$ means that the objective is to encapsulate keys with 256 bits of entropy. In order to enable the fast NTT-based multiplication, a small prime q value is chosen. The parameter k is set to the lattice dimension as a multiple of n. As k increases, the security level also increases. Finally, the parameters η_1, η_2, d_u, and d_v are chosen to attain a balance between security, ciphertext size, and failure probability. More details about the choice of these parameters are shown in [14].

These key pair generation, encryption, and decryption algorithms call a few functions, detailed in [14]. Shortly, $\mathbf{PRF}(\cdot)$ is a pseudorandom function, $\mathbf{XOF}(\cdot)$ is an extendable output function, and $\mathbf{KDF}(\cdot)$ is a key derivative function. The CRYSTALS-Kyber scheme also makes use of two hash functions: $\mathbf{H}(\cdot)$ and $\mathbf{G}(\cdot)$. The $\mathbf{Parse}(\cdot)$ function uses deterministic approaches to sample elements in R_q while the $\mathbf{CBD}_\eta(\cdot)$ function samples noise from a centered binomial distribution. The $\mathbf{NTT}(\cdot)$ function performs multiplications in R_q in a very efficient way, and the function $\mathbf{NTT}^{-1}(\cdot)$ performs the inverse of the $\mathbf{NTT}(\cdot)$. The $\mathbf{Encode}(\cdot)$ function serializes a polynomial into a byte array, while the $\mathbf{Decode}(\cdot)$ function does the opposite. Moreover, the $\mathbf{Compress}_q(\cdot)$ function takes an element from $\mathbb{Z}_q$ and outputs an integer in the set $\{0, \ldots, 2^q - 1\}$, where $d < \lceil \log_2(q) \rceil$, and the $\mathbf{Decompress}_q(\cdot)$ function does the opposite.

The main part of the key pair generation of the KEM (Algorithm 4) is the generation of the public-key (pk) and the secret-key (sk), derived from the key pair generation (Algorithm 1) which performs the $\mathrm{pk} = As + e$ operation, using the public matrix $\mathbf{A}$ and the private vector $\mathbf{s}$.

The encapsulation (Algorithm 5) uses the public-key (pk) together with the encryption algorithm (Algorithm 2) to generate a shared-key K (i.e., a symmetric key) and a ciphertext c. More specific, the encryption algorithm regenerates the matrix $\mathbf{A}$ using the public-key, samples $\mathbf{r}$, $\mathbf{e}_1$, and e_2, and performs $\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$ and $v = \mathbf{t}^T\mathbf{r} + e_2 + \mathbf{Decompress}_q(m, 1)$, where m is a random message generated in the encapsulation algorithm (i.e. Algorithm 5).

Finally, the decapsulation (Algorithm 6) uses the secret-key sk, ciphertext c, encryption (Algorithm 2), and decryption (Algorithm 3) to obtain the same shared-key K previously generated in the encapsulation algorithm. In the decapsulation algorithm, given the secret-key sk and the ciphertext c, m' is computed by $m' = \mathbf{Decryption}(\mathrm{sk}, c)$, and c' is the encryption of m'. In possession of c and c', the shared-key K is computed.

The secret-key sk and the ciphertext c are decrypted, which mainly performs $m' = \mathbf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1)$ operation. Then m' is encrypted, making possible to recover c'. In possession of c and c', it is possible to generate the shared-secret K.

Algorithm 1: PKEKeyPairGeneration()

Input: None
Output:
public-key: pk$\in \mathcal{B}^{12kn/8+32}$
secret-key: sk$\in \mathcal{B}^{12kn/8}$
Procedure:
▷ Initialization:
$d \leftarrow \mathcal{B}^{32}$
$(\rho, \sigma) := G(d)$
$N := 0$
▷ Generate $\hat{\mathbf{A}} \in R_q^{k \times k}$ in the NTT domain:
for *i from* $0 : k - 1$ **do**
  for *j from* $0 : k - 1$ **do**
    $\hat{\mathbf{A}}[i][j] := \mathbf{Parse}(\mathbf{XOF}(\rho, j, i))$
  end
end
▷ Sample $\mathbf{s} \in R_q^k$ from B_{η_1}:
for *i from* $0 : k - 1$ **do**
  $\mathbf{s}[i] := \mathbf{CBD}_{\eta_1}(\mathbf{PRF}(\rho, N))$
  $N := N + 1$
end
▷ Sample $\mathbf{e} \in R_q^k$ from B_{η_1}:
for *i from* $0 : k - 1$ **do**
  $\mathbf{e}[i] := \mathbf{CBD}_{\eta_1}(\mathbf{PRF}(\rho, N))$
  $N := N + 1$
end
$\hat{\mathbf{s}} := \mathbf{NTT}(\mathbf{s})$
$\hat{\mathbf{e}} := \mathbf{NTT}(\mathbf{e})$
$\hat{\mathbf{t}} := \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$
Return:
public-key pk:= ($\mathbf{Encode}_{12}(\hat{\mathbf{t}} \bmod^+ q)) \parallel \rho$
secret-key sk:= $\mathbf{Encode}_{12}(\hat{\mathbf{s}} \bmod^+ q)$

V. SOFTWARE IMPLEMENTATION

This section presents the software implementation of the CRYSTALS-Kyber scheme on the ARM Cortex-A9 processor of an SoC-based development board. To detail the software implementation, this section is organized as follows: Subsection V-A describes the SoC FPGA development board used to implement the software version of the CRYSTALS-Kyber scheme and its hardware/software implementation, which is detailed in Section VI; Subsection V-B discusses a preliminary analysis of the CRYSTALS-Kyber scheme implemented on the ARM Cortex-A9 processor, highlighting the most time-consuming functions that ought to be hardware implemented; and, finally, Subsection V-C provides additional details about the software implementation

Algorithm 2: Encryption()

public-key: pk$\in \mathcal{B}^{12kn/8+32}$
Message: $m \in \mathcal{B}^{32}$
Random coins: $r \in \mathcal{B}^{32}$
Output:
Ciphertext: $c \in \mathcal{B}^{d_u kn/8+d_v n/8}$
Procedure:
▷ Initialization:
$N := 0$
$\hat{\mathbf{t}} := \mathbf{Decode}_{12}(\text{pk})$
$\rho :=$pk$+12kn/8$
▷ Generate $\hat{\mathbf{A}} \in R_q^{k\times k}$ in the NTT domain:
for *i from* $0 : k-1$ **do**
  for *j from* $0 : k-1$ **do**
    $\hat{\mathbf{A}}^T[i][j] := \mathbf{Parse}(\mathbf{XOF}(\rho, i, j))$
  end
end
▷ Sample $\mathbf{r} \in R_q^k$ from B_{η_1}:
for *i from* $0 : k-1$ **do**
  $\mathbf{r}[i] := \mathbf{CBD}_{\eta_1}(\mathbf{PRF}(r, N))$
  $N := N+1$
end
▷ Sample $\mathbf{e}_1 \in R_q^k$ from B_{η_2}:
for *i from* $0 : k-1$ **do**
  $\mathbf{e}_1[i] := \mathbf{CBD}_{\eta_2}(\mathbf{PRF}(r, N))$
  $N := N+1$
end
▷ Sample $e_2 \in R_q$ from B_{η_2}:
$e_2 := \mathbf{CBD}_{\eta_2}(\mathbf{PRF}(r, N))$
$\hat{\mathbf{r}} := \mathbf{NTT}(\mathbf{r})$
$\mathbf{u} := \mathbf{NTT}^{-1}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_1$
$v := \mathbf{NTT}^{-1}(\hat{\mathbf{t}} \circ \hat{\mathbf{r}}) + e_2 +$
  $\mathbf{Decompress}_q(\mathbf{Decode}_1(m), 1)$
$c_1 := \mathbf{Encode}_{d_u}(\mathbf{Compress}_q(\mathbf{u}, d_u))$
$c_2 := \mathbf{Encode}_{d_v}(\mathbf{Compress}_q(v, d_v))$
Return:
Ciphertext $c := (c_1 \parallel c_2)$

Algorithm 3: Decryption()

Input:
secret-key: sk$\in \mathcal{B}^{12kn/8}$
Ciphertext: $c \in \mathcal{B}^{d_u kn/8+d_v n/8}$
Output:
$\mathbf{u} := \mathbf{Decompress}_q(\mathbf{Decode}_{d_u}(c), d_u)$
$v := \mathbf{Decompress}_q(\mathbf{Decode}_{d_v}(c + d_u kn/8), d_v)$
$\hat{\mathbf{s}} := \mathbf{Decode}_{12}(\text{sk})$
Return:
Message $m := \mathbf{Encode}_1(\mathbf{Compress}_q(v - \mathbf{NTT}^{-1}(\hat{\mathbf{s}}^T \circ$
$\mathbf{NTT}(\mathbf{u}), 1))$

Algorithm 4: KEMKeyPairGeneration()

Input: None
Output:
public-key: pk$\in \mathcal{B}^{12kn/8+32}$
secret-key: sk$\in \mathcal{B}^{24kn/8+96}$
Procedure:
Initialization:
$z \leftarrow \mathcal{B}^{32}$
(pk, sk'):= PKEKeyPairGeneration()
$\text{sk}' := \mathbf{Encode}_{12}(\hat{\mathbf{s}} \bmod^+ q)$
Return:
public-key pk
secret-key sk:=(sk'$\parallel$ pk $\parallel \mathbf{H}$(pk)$\parallel z$)

Algorithm 5: Encapsulation()

Input:
public-key: pk$\in \mathcal{B}^{12kn/8+32}$
Output:
Ciphertext: $c \in \mathcal{B}^{d_u kn/8+d_v n/8}$
shared-key: $K \in \mathcal{B}^*$
Procedure:
Initialization:
$m \leftarrow \mathcal{B}^{32}$
$m \leftarrow \mathbf{H}(m)$
$(\bar{K}, r) := \mathbf{G}(m \parallel \mathbf{H}(\text{pk}))$
$c := \mathbf{Encryption}(\text{pk}, m, r)$
Return:
Ciphertext $c := (c_1 \parallel c_2)$
shared-key $K := \mathbf{KDF}(\bar{K} \parallel \text{H}(c))$

of high-level functions that must be considered to come up with a software/hardware implementation.

A. THE SYSTEM-ON-CHIP DEVELOPMENT BOARD

The MicroZed 7010 development board [55] was chosen for implementing the CRYSTALS-Kyber scheme because it enables software, hardware, and hardware/software implementations. It is a low-cost System on Module (SoM) based on the Xilinx Zyqn-7000 SoC FPGA. In addition to the Xilinx Zyqn-7000 SoC FPGA, the development board contains 1 GB of DDR3 SDRAM, 128 Mb of quad serial peripheral interface (QSPI) Flash, 33.33 MHz oscillator, well-established interfaces, and other accessories.

The Xilinx Zyqn-7010 SoC FPGA used is a XC7Z010-1CLG400C, which is composed of a Processing System (PS) and Programmable Logic (PL). While PS is an ARM Cortex-A9 processor with the ARMv7-architecture, the PL is a 28 nm Artix-7 FPGA with 28k Programmable Logic Cells, 17.6k Lookup Tables (LUTs), 35.2k Registers, 60 Block Random Access Memorys (BRAMs) with 36kb each, and 80 Digital Signal Processing (DSP) blocks [55]. Using a Phase-Locked Loop (PLL), a 666.66 MHz and a 100 MHz clocks are derived from the 33.33 MHz built-in oscillator to feed PS and PL, respectively. For the sake of simplicity, from now on, FPGA and ARM will refer respectively to the 28 nm

Algorithm 6: Decapsulation()

Input:
secret-key: sk$\in \mathcal{B}^{24kn/8+96}$
Ciphertext: $c \in \mathcal{B}^{d_u kn/8+d_v n/8}$
Output:
shared-key: $K \in \mathcal{B}^*$
Procedure:
pk:= sk $+ 12kn/8$
$h :=$ sk $+ 24kn/8 + 32 \in \mathcal{B}^{32}$
$z :=$ sk $+ 24kn/8 + 64$
$m' =$ **Decryption**(sk, c)
$(\bar{K}', r') :=$ **G**($m' \parallel h$)
$c' =$ **Encryption**(pk, m', r')
Return:
if $c = c'$ **then**
  Shared-key $K :=$ **KDF**($\bar{K}' \parallel$ **H**(c))
else
  Shared-key $K :=$ **KDF**($z \parallel$ **H**(c))
end

Artix-7 FPGA and the ARM Cortex-A9 processor with the ARMv7-architecture.

It is important to notice that the Zyqn-7010 SoC FPGA device allows high flexibility, enabling the balance of the workload between the FPGA and ARM processor. Besides, if this specific device (i.e., Zynq-7010) could not deal with the amount of workload required, it is almost straightforward to upgrade to a more powerful device from the same family, such as the Zynq-7020, which brings a strong notion of scalability. Although, when there is a necessity to change from one family device to another due to workload requirements, especially when the SM equipment is based on microcontrollers, this upgrade might not be so simple as is the case of most SM equipment reported by the existing literature.

B. PRELIMINARY ANALYSIS

A preliminary analysis of the software implementation of the CRYSTALS-Kyber scheme allows us to identify the most time-consuming functions. The source code available in [9] was executed using the ARM processor (i.e., the software implementation). Table 2 summarizes the relative time-consuming of functions used by the CRYSTALS-Kyber-512 scheme in the software implementation. Similar results were achieved for the CRYSTALS-Kyber-768 and CRYSTALS-Kyber-1024 schemes; however, for simplicity, they are omitted.

Table 2 shows that **NTT**(·), **NTT**$^{-1}$(·), **Muliply and Accumular**(·), **Keccak-f**(·), **Tomont**(·), and **Reduce**(·) are the main time-consuming functions. For the sake of simplicity, from now on, these functions will be categorized as high-level functions. Other functions are also required to perform the key pair generation, encapsulation, and decapsulation; however, none demand a significant amount of time compared to the functions listed in Table 2. Apart from the **Keccak-f(1600)**(·), the high-level functions are mainly used to perform matrix-vector multiplication, **NTT**(·), and **NTT**$^{-1}$(·), (i.e., functions found in Algorithms 1 to 6), which result in a high computational burden since they are based in long nested loops, covering all elements of polynomials or array of polynomials. Moreover, **Keccak-f(1600)**(·) is the core of hash functions used in the CRYSTALS-Kyber scheme, then this high-level function is called every time that pseudo-random values must be generate (i.e., when **H**(·), **G**(·), **PRF**(·), **XOF**(·), **KDF**(·) functions are called).

TABLE 2. The relative time execution, in percentage, of the high-level functions for the CRYSTALS-Kyber-512.

Function	Key Pair Generation	Encap-sulation	Decap-sulation
NTT(·)	24.80%	11.11%	19.54%
NTT^{-1}(·)	-	25.88%	30.23%
Multiply and Accumulate(·)	18.37%	24.47%	28.19%
Keccak-f(1600)(·)	8.90%	9.18%	6.82%
Tomont(·)	2.18%	-	-
Reduce(·)	1.59%	1.70%	1.35%
Others	44.16%	27.66%	13.87%

C. IMPLEMENTATION

Relying on the preliminary analysis in Subsection V-B and without paying attention to the **Keccak-f(1600)**(·) function, we see that high-level functions are mainly processed based on three other functions: (i) **Fqmul**(·), (ii) **Montgomery Reduction**(·), and (iii) **Barrett Reduction**(·), which from now on will be categorized as low-level functions. Further analyses of low-level and **Keccak-f(1600)**(·) functions allow us to understand the necessity of their hardware implementation.

Starting with the **Fqmul**(·) function, we can say that it receives two values, multiplies them, and sends the result to be reduced by the **Montgomery Reduction**(·) function. Finally, the reduced value is the output of the **Fqmul**(·) function. As can be seen, the **Fqmul**(·) function is simple, requiring only one multiplication and another function call. However, the multiplication process in software is not well optimized, requiring some instructions. Consequently, when multiplications are called in nested loops, which is the case of the **Fqmul**(·) function, it may take a significant amount of time to execute all required operations.

The **Montgomery Reduction**(·) accelerates reductions by transforming the inputs into a special form called the Montgomery form. It efficiently reduces ar^{-1} mod q, where a is the number to be reduced. As in the CRYSTALS-Kyber scheme, the module q is constant; thus, a routine variable can be turned into a constant, sparing processing time. Another improvement is a wise choice of the constant r to replace division by shift operation, a less time expensive operation. So, if r is defined as the constant integer 16, then the integer q^{-1} mod r is the constant integer -3327. As the

Algorithm 7: Montgomery Reduction(·)

Input:
a: 32-bits integer
Output:
t: 16-bits integer
Constants:
$q := 3329$
$r := 16$
$q^{-1} : -3327$ ▷ $q^{-1} \bmod 2^r$
Procedure:
$t := aq^{-1}$
Return:
$t := (a - tq) \gg r$

Algorithm 8: Barrett Reduction(·)

Input:
a: 16-bits integer
Output:
t: 16-bits integer
Constants:
$q := 3329$
$v := ((1 \ll 26) + q/2)/q$
Procedure:
$t := (va + (1 \ll 25)) \gg 26$
$t := tq$
Return:
$t := a - t$

Montgomery Reduction(·) function is required many times in the CRYSTALS-Kyber scheme, a wise choice of r and the use of a precalculated constant considerably reduce the time consumed by this function. Algorithm 7 illustrates the **Montgomery Reduction**(·) function.

The **Barrett Reduction**(·) function aims to optimize the operation $c = a \bmod q$ by pre-calculating a constant and assuming that q does not change. This way, it can avoid the slowness of long divisions, which multiplications can replace. Algorithm 8 shows the **Barrett Reduction**(·) function.

The **Keccak-f(1600)**(·) function is the core of Keccak, which is a family of sponge functions standardized as SHA3-224 to SHA3-512 hash functions, and SHAKE128 and SHAKE256 extendable output functions in Federal Information Processing Standards (FIPS) 202 [56]. In the CRYSTALS-Kyber scheme, the function **H**(·) actually performs a SHA3-256 hash function and **G**(·) performs a SHA3-512. On the other hand, **XOF**(·) performs SHAKE128, and **PRF**(·) and **KDF**(·) perform SHAKE256. The functions mentioned above make use of the **Keccak-f(1600)**(·) function, which is detailed in the following paragraphs.

The **Keccak-f(1600)**(·) function is performed in five steps, which are called θ, ρ, π, χ, and ι. The algorithm for each step takes a state array as input, called $\mathcal{A}$, and outputs an updated array, called $\mathcal{A}'$. The state $\mathcal{A}$ can be seen as a three-dimensional array where each piece is one bit of the state. The size of $\mathcal{A}$ depends on the specification of the Keccak function; however, it has to follow the dimensions of x-by-y-by-z, in which x and y are equal to 5 for the **Keccak-f(1600)**(·) function. Consequently, 1600 in the **Keccak-f(1600)**(·) function means that there are 1600 bits in the state array, consequently, z must be equals to 64. Figure 2 illustrates the three-dimensional state array. In this sense, let $0 \leq i < x$, $0 \leq j < y$, and $0 \leq w < z$, variables that are associated with x, y, and z axes, respectively. A *lane* is defined by $lane(i, j) = \mathcal{A}[i, j, 0] \parallel \ldots \parallel \mathcal{A}[i, j, z-1]$, while a column is defined by $column(i, w) = \mathcal{A}[i, 0, w] \parallel \ldots \parallel \mathcal{A}[i, y-1, w]$.

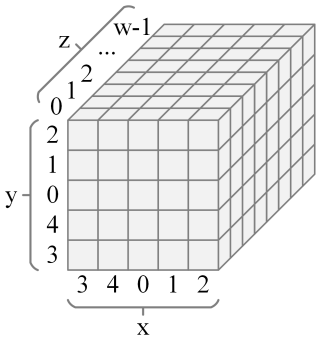


FIGURE 2. Three-dimensional state array of the Keccak-f(1600)(·) function.

It is important to emphasize that each step seeks to scramble the previous state as follows: in the first step θ, each bit in $\mathcal{A}$ is XORed with the parity of the two columns in the array computed by a predefined function; step ρ rotates the bits of each lane by a predefined offset, which depends on the x and y coordinates of the current lane; step π aims to rearrange the position of the lanes. In step χ, each state bit is XORed with a non-linear function of two other bits in its row; finally, step ι modifies some of the bits of $lane(0, 0)$. More details can be found in [56]. Due to the absorb and squeeze characteristics of sponge functions, the **Keccak-f(1600)**(·) function may be executed a few times until all data is processed.

VI. HARDWARE/SOFTWARE IMPLEMENTATION

This section details the proposed hardware/software implementation of the CRYSTALS-Kyber scheme. Relying on the MicroZed development board [55], Figure 3 shows the block diagram of the proposed hardware/software implementation of the CRYSTALS-Kyber scheme. As we can see, this implementation can be divided into three main instances: PS, Interconnect, and PL. The PS is responsible for hosting the ARM processor, and it is where the software implementation is placed. The interconnect controls the data exchange between the PS and PL using the Advanced eXtensible Interface memory mapped (AXI-MM) [57]. The PL, based on an FPGA device, is where the most time-consuming functions (i.e., functions listed in Table 2) are hardware implemented.

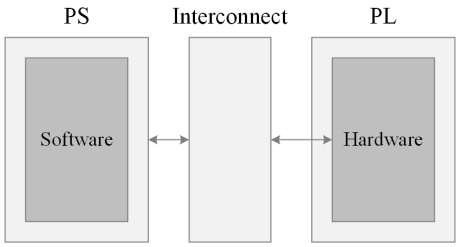


FIGURE 3. The Block diagram for the proposed hardware/software implementation. The arrows represent 32-bit buses. Note that the control signals are omitted.

Considering the results in Table 2, we see that the execution time reduction can be attained by implementing the most time-consuming functions (i.e., the high-level functions) in isolated hardware blocks using the FPGA device, which accelerates the execution of the CRYSTALS-Kyber scheme. Note that, as elucidated in Subsection V-C, the high-level functions share the low-level functions, which can be explored for saving hardware resource usage. Consequently, the high-level functions are implemented in separated hardware blocks, although each of them can access the low-level functions, which are also implemented in independent hardware blocks. From now on, the functions implemented in hardware will be called blocks and not functions to differentiate the software implementation from the hardware one.

Furthermore, the hardware implementation also reduces data communication burden, which is accomplished by using the Direct Memory Access (DMA), and saving memory due to using the dual BRAMs with dual-port each. Both DMA and dual BRAMs are used by all high-level blocks. Also, they work as interfaces between the ARM processor and FPGA device. Figure 4 shows the block diagram for the hardware implementation, which illustrates the high-level and low-level blocks in the PL, omitting control signals and simplifying the block connections. MR and BR are acronyms for Montgomery Reduction and Barrett Reduction in this block diagram.

Note that the DMA is responsible for transferring data to/from the Interconnect, consequently, to/from the FPGA device. The dual BRAM block is constituted of two BRAMs with dual-port each, enabling double write/read operations at once per BRAM. When data is received from the PS, it is stored in one of the BRAMs and becomes available to be processed. To accelerate the data transmission between the PS and PL, we concatenate two 16-bit words to use a 32-bit bus, transmitting two words at once, which are stored in the BRAM. Consequently, each position loads two 16-bit words at once. Moreover, as a BRAM owns a dual-port, two addresses can be read at once, meaning four words of 16-bit each can be loaded at a time.

After the data reception, one of the high-level blocks reads the data in the BRAM (i.e., four 16-bit words) and processes it using the low-level blocks when required (i.e., Fqmul, Montgomery Reduction, and Barrett Reduction). It is important to note that low-level blocks are shared between high-level blocks to reduce hardware consumption, saving DSP blocks and LUTs. The implemented blocks are briefly detailed as follows:

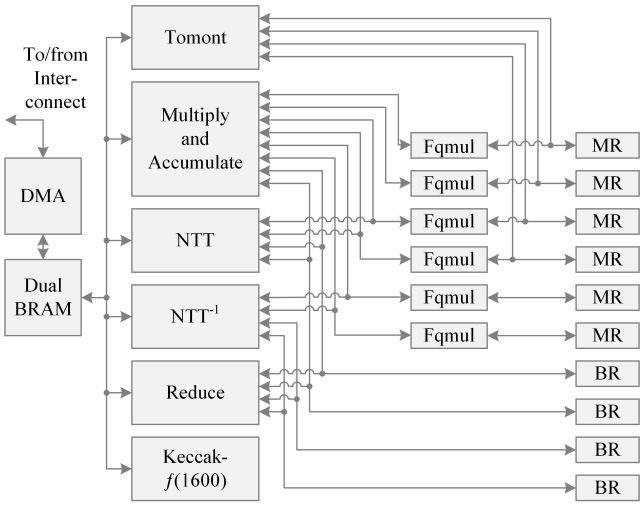


FIGURE 4. Block diagram of the proposed hardware implementation for the CRYSTALS-Kyber scheme. Control signals have been omitted and connections simplified.

- **Tomont block**: it is responsible for converting all polynomial coefficients from the usual domain to the Montgomery domain. In this sense, this block multiplies each coefficient by a constant and then applies the Montgomery Reduction. As the Tomont block processes four coefficients in parallel, it requires four DSP blocks and uses four Montgomery Reduction blocks.
- **Multiply and Accumulate block:** it aims to multiply two arrays of polynomials using the Fqmul block and, consequently, the Montgomery Reduction block, in the NTT domain. Each one of the polynomials of the array is multiplied, coefficient by coefficient, by its corresponding polynomial in the other array. In the end, the results of the multiplications are summed up and presented at the block output. Moreover, the Multiply and Accumulate block does not use any DSP block since the multiplication takes place in the Fqmul block; however, it uses six Fqmul blocks and two Barrett Reduce blocks.
- **NTT and NTT^{-1} blocks:** they apply the forward and inverse NTT to all coefficients of an array of polynomials. In this sense, two Fqmul blocks and two Barrett Reduction blocks are used.
- **Reduce block:** it applies a reduction to each coefficient in an array of polynomials, requiring four Barrett Reduction blocks because four 16-bit words are loaded from the BRAM at once.
- **Keccak-f(1600) block:** it implements the five steps aforementioned in Subsection V-C. It also gets the initial state from one of the BRAMs and stores the updated state using the other one. See Subsection V-C for more details.

- **Fqmul block:** it receives two coefficients from a high-level block and multiplies them, forwarding the result to a Montgomery Reduction block. Six Fqmul blocks are available; consequently, six Montgomery Reduction blocks are required. As expected, each Fqmul block requires one DSP block.
- **Montgomery Reduction block:** it receives the processed value from the Fqmul block and reduces it, requiring two DSP blocks for performing it, see Algorithm 7 in Subsection V-C.
- **Barrett Reduction block:** it receives a coefficient to be reduced, which requires two DSP blocks as shown in Algorithm 8, see details in Subsection V-C.

A. VALIDATION METHOD

In order to validate the proposed hardware/software implementation, extensive testing was performed using random seeds. The first step consists of generating a key pair (i.e., asymmetric keys). Then the encapsulation and decapsulation procedures are performed, comparing the shared-key (i.e., a symmetric key) obtained by each one. If they match, the same steps aforementioned are reproduced using the software implementation (i.e., based on the source code provided in [9]) using the same random seeds. Again, another comparison between the obtained shared-keys are performed. The test will only succeed if all shared-keys match.

VII. PERFORMANCE EVALUATION

This section discusses hardware resource and timing analyses of the implementation of the CRYSTALS-Kyber scheme. The focus is comparative analyses between the software implementation, briefly described in Section V, and the detailed hardware/software implementation, presented in Section VI. In this sense, this section is organized as follows: Subsection VII-A presents the hardware resource usage of the blocks implemented in the PL and, in Subsection VII-B, the timing analysis comparing the implementations with and without hardware acceleration.

A. HARDWARE RESOURCE ANALYSIS

This subsection evaluates the hardware resource demanded by DMA, dual BRAM, and high-level and low-level blocks, which were discussed in Section VI. For this analysis, it does not matter which security level of the CRYSTALS-Kyber scheme we are using because the structure of the blocks does not change with the security level. The only difference is that for higher security levels, more loops these blocks are required to run but using the same resources. The attained results are reported in Table 3.

Table 3 shows that Keccak-$f(1600)$ and DMA are the blocks which require, as expected, more slice LUTs and slice registers. The former demand more hardware resource to implement the five steps, as explained in Subsection V-C. The latter uses many control signals to coordinate the data transfer between the PL and PS. The dual BRAM is the block demanding more BRAMs, as expected because it aims to store the received and processed data. Finally, the Tomont, Fqmul, Montgomery Reduction, and Barrett Reduction are the operations requiring DSP blocks, agreeing with the discussion in Section VI.

TABLE 3. Hardware resource usage.

Algortithm	Number of blocks	Slice LUTs	Slice Register	BRAM	DSP
Tomont	1	68	111	0	4
Multiply and Accumulate	1	535	676	0	0
Reduce	1	99	248	0	0
NTT	1	287	475	0	0
NTT^{-1}	1	284	463	0.5	0
Keccak-$f(1600)$	1	4206	3458	0	0
Fqmul	6	0 (0)	1 (6)	0 (0)	1 (6)
Montgomery Reduction	6	1 (6)	80 (480)	0 (0)	2 (12)
Barrett Reduction	4	1 (4)	33 (132)	0 (0)	2 (8)
Dual BRAM	1	120	107	4	0
DMA	1	1353	1955	2	0
Others	-	3290	5186	0	0
Total	-	10252	13300	6.5	30

B. TIMING ANALYSIS

To compare the performance of the CRYSTALS-Kyber scheme with and without hardware acceleration, the following analyses are considered: in Subsection VII-B1 a time comparison between the software and hardware/software implementations is presented, evaluating the time required to process the key pair generation, encapsulation, and decapsulation. Subsection VII-B2 compares the high-level functions and blocks in terms of time performance. Finally, Subsection VII-B3 provides a thorough analysis of each high-level block, comparing the time that a block is processing data with the time that this block spends transferring data. Note that each run time measurement was obtained using a dedicated timer, which is implemented in the PL (i.e., hardware-implemented).

1) COMPARISON BETWEEN THE IMPLEMENTATIONS

This subsection focuses on comparing the software implementation [14] with the proposed hardware/software implementation for each security level of the CRYSTALS-Kyber scheme. To do so, it shows the time required to execute the key pair generation, encapsulation, and decapsulation. Also, it discusses the total execution time.

Table 4 shows that when higher security levels are used, the total execution time increases. This fact occurs due to the use of more polynomials in order to increase security, which requires more execution time. It is also clear that using the software implementation, the total execution time to perform the CRYSTALS-Kyber-512, -768, and -1024 schemes are 12.815, 20.398, and 30.595 ms, respectively. The

TABLE 4. Total execution time and performance in ms.

Scheme	Software ARM-A9 [14]				Hardware/software				Relative Time Improvement (α_{TI})
	Key Pair Generation	Encap-sulation	Decap-sulation	Total (T_{TETS})	Key Pair Generation	Encap-sulation	Decap-sulation	Total (T_{TETH})	
CRYSTALS-Kyber 512	3.286	4.465	5.062	12.815	0.869	1.393	1.792	4.054	68.36%
CRYSTALS-Kyber 768	5.436	7.103	7.858	20.398	1.407	2.014	2.436	5.857	71.28%
CRYSTALS-Kyber 1024	8.539	10.575	11.480	30.595	2.183	2.845	3.308	8.337	72.75%

total execution time for the same security levels but using the proposed hardware/software implementation are 4.054, 5.857, and 8.337 ms, respectively. The relative time improvement (α_{TI}) between the software and hardware/software implementations is given by

$$\alpha_{TI} = 1 - (T_{TETH}/T_{TETS}), \quad (1)$$

where T_{TETH} is the total execution time in hardware and T_{TETS} is the total execution time in software. The α_{TI} achieved are 68.36%, 71.28%, and 72.75% for the CRYSTALS-Kyber-512, -768, and -1024 schemes, respectively.

It is important to note that α_{TI} increases as the security level rises. In fact, at higher security levels, more data must be processed and transferred to/from the PS and PL. Also, α_{TI} increases because the processing in the PL is faster than in the PS, and this fact compensates for the increase of the communication burden. In the end, we see higher relative time improvement for higher security levels.

2) EXECUTION TIME ANALYSIS

This subsection seeks to analyze only the high-level functions and blocks individually, ignoring their impact on the whole implementation. In this sense, only the software and hardware/software implementations of the CRYSTALS-Kyber-512 scheme are compared in Table 5. The average execution time of software (T_{AETS}) and hardware (T_{AETH}) are expressed as

$$T_{AETS} = \sum_{1}^{N_I} T_{TETS,i}/N_I \quad (2)$$

and

$$T_{AETH} = \sum_{1}^{N_I} T_{TETH,i}/N_I, \quad (3)$$

where $T_{TETS,i}$ and $T_{TETH,i}$ are the total execution time in software and hardware, respectively, during the ith simulation while N_I is the number of simulations of each function or block, which was arbitrarily set to 5000. It is important to highlight that the communication burden between the PL and PS is included in the T_{AETH}. Finally, the relative time improvement (β_{TI}) between the software and hardware/software implementation, see its values in Table 5, is given by

$$\beta_{TI} = 1 - (T_{AETH}/T_{AETS}). \quad (4)$$

TABLE 5. Average execution time analysis in μs.

Algorithm	Average Execution Time		Relative Time Improvement (β_{TI})
	Software (T_{AETS})	Hardware (T_{AETH})	
Tomont	26.989	11.825	56.18%
Reduce	48.896	17.723	63.75%
Multiply and Accumulate	212.740	22.249	89.54%
NTT	378.480	37.179	90.18%
NTT^{-1}	550.380	37.214	93.24%
Keccak-f(1600)	56.846	9.434	83.40%

Since the average execution times and relative time improvements of both security levels of 768 and 1024 of the CRYSTALS-Kyber are almost the same as in the CRYSTALS-Kyber-512 scheme, they were omitted.

Table 5 shows that the algorithm for the Multiply and Accumulate, NTT, and NTT^{-1} spend a significant amount of time on software. It occurs since both of them require a lot of multiplications, which are expensive software operations. On the other hand, these operations are performed in parallel, and dedicated DSP blocks are used to perform this operation in hardware. Consequently, these blocks show the highest β_{TI}. The Keccak-f(1600) is another block with a high β_{TI} because it is looped-based. Consequently, it requires significant time to be executed in software; however, in hardware, its execution time is reduced.

Last but not least, the Tomont and Reduce blocks attain the lowest β_{TI}. The reason is that their implementations require a low number of multiplications, making the hardware performance improvement of these blocks less significant compared to the software implementation. Nevertheless, they attain improvement greater than 54% and 64%, respectively.

3) HARDWARE PROCESSING TIME ANALYSIS

The hardware processing time analysis is another evaluation parameter that deserves attention. We focus only on the proposed hardware/software implementation to conduct this analysis. We compare the average execution time in hardware (T_{AETH}), already presented in the third column of Table 5,

TABLE 6. Hardware processing time analysis in μs.

Algorithm	Average Execution Time in Hardware (T_{AETH})	Average Processing Time in Hardware (T_{APTH})	Relative Time (γ_{TD})
Tomont	11.825	0.750	6.34%
Reduce	17.723	1.380	7.79%
Multiply and Accumulate	22.249	2.940	13.21%
NTT	37.179	20.720	55.73%
NTT^{-1}	37.214	20.750	55.76%
Keccak-f(1600)	9.434	1.300	13.78%

versus its processing time only. The T_{AETH} can be separated into two components, and it is given by

$$T_{\text{AETH}} = T_{\text{APTH}} + T_{\text{DTT}}, \tag{5}$$

where the average processing time in hardware (T_{APTH}) is the time required to process the data, and the data transfer time (T_{DTT}) is the time required to transfer data to/from the PS. Table 6 shows the attained results in the hardware implementation. Note that, for simplicity, this subsection only contemplates the analysis for the CRYSTALS-Kyber-512 since very similar results are obtained using the CRYSTALS-Kyber-768 and -1024 schemes.

Observing the relative time (γ_{TD}), which is given by

$$\gamma_{\text{TD}} = T_{\text{APTH}}/T_{\text{AETH}}, \tag{6}$$

we can see that the Tomont, Reduce, Multiply and Accumulate, and Keccak-f(1600) blocks attain a very low γ_{TD}, which means that these blocks spend much more time transmitting data between the PL and PS than actually processing them. It occurs for two main reasons: (i) the processing is simple, and (ii) a significant amount of data needs to be transmitted.

On the other hand, the NTT and NTT^{-1} blocks have a γ_{TD} higher than 50%, which means that these two blocks spend more time processing data than transferring them. The reason for a higher γ_{TD} is that the NTT and NTT^{-1} blocks (i) have a few data to transfer to/from the PL and (ii) require a considerable amount of computation. Note that the worst results would be obtained without the use of DMA because it accelerates the data transfer between the PS and PL significantly.

VIII. LIMITATIONS OF THE STUDY

After the key exchange post-quantum protocol is completed (i.e., both SM and MDMS hold a symmetric key), an encryption algorithm (e.g., the Advanced Encryption Standard (AES) combined with the Galois Counter Mode (GCM)) applies for accomplishing a secure communication. The security of this solution is currently well established in the state of the art of cryptography and, as far as it is known, quantum secure. In other words, we are not seeking to present a new authentication scheme for SM networks.

Moreover, we understand that our findings can be extended for a dedicated cryptographic module in the MDMS server, for instance. These modules are typically found in the market as isolated solutions that hold the cryptographic part (mainly private- and shared-keys), separating from the management part and adding a layer of security. Furthermore, due to their flexibility and hardware acceleration potential, these modules should rely on SoC FPGA devices.

Finally, this study aims at supporting the deployment and quantum-security of SGs, focusing on SMs. However, the proposed implementation of the CRYSTALS-Kyber scheme could be applied to other applications that consider similar data communication infrastructure based on hardware-constraint devices, such as sensors, other metering, and IoT-related devices, among others. Note that these applications share the same requirements (i.e., low-cost hardware implying a hardware-constrained device, exchange of sensitive information, high availability), making the implementation proposed in this study straightforward.

IX. CONCLUSION

This paper has discussed the feasibility of using the CRYSTALS-Kyber scheme, a winner in the NIST standardization process for PQC, in hardware-constrained equipment, such as SM, for securing sensitive data traveling through SM networks. In this sense, important issues related to its optimized software/hardware implementation in an SoC FPGA device were pointed out. Also, it has presented a description of hardware/software co-design implementation of the CRYSTALS-Kyber scheme in an SoC FPGA device.

Experimental results have shown that the proposed hardware/software implementation, accelerating the most time-consuming functions of the CRYSTALS-Kyber scheme, can reduce the execution time by around 70% compared to its software implementation. Furthermore, it reduces the execution time from 12.815 ms to 4.054 ms at the lowest security level and from 30.595 ms to 8.337 ms at the highest security level compared to the software implementation. Moreover, the hardware resource analysis has shown that the CRYSTALS-Kyber scheme could be embedded in hardware-constrained equipment that makes use of SoC FPGA device. For instance, SMs, in which an FPGA runs the most-time consuming component of the PQC scheme.

Overall, the detailed analyses show that the proposed hardware/software co-design implementation of the CRYSTALS-Kyber scheme is feasible for SM equipment.

REFERENCES

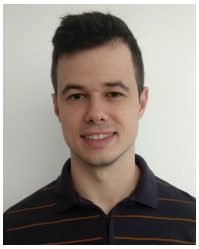
[1] A. Caillé, M. Al-Moneef, F. Barnés de Castro, A. Bundgaard-Jensen, A. Fall, N. Franco de Medeiros, and C. P. Jain, "Deciding the future: Energy policy scenarios to 2050," World Energy Council, London, U.K., Tech. Rep., 2007.

[2] L. De M. B. A. Dib, V. Fernandes, M. de L. Filomeno, and M. V. Ribeiro, "Hybrid PLC/wireless communication for smart grids and Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 655–667, Apr. 2018.

[3] R. M. de Oliveira, A. B. Vieira, H. A. Latchman, and M. V. Ribeiro, "Medium access control protocols for power line communication: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 920–939, Aug. 2019.
[4] V. L. R. D. Costa, V. Fernandes, and M. V. Ribeiro, "Narrowband hybrid PLC/wireless: Transceiver prototype, hardware resource usage and energy consumption," *Ad Hoc Netw.*, vol. 94, Nov. 2019, Art. no. 101945.
[5] X. Liu and Z. Li, "False data attacks against AC state estimation with incomplete network information," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2239–2248, Sep. 2016.
[6] H. Khurana, M. Hadley, N. Lu, and D. A. Frincke, "Smart-grid security issues," *IEEE Security Privacy*, vol. 8, no. 1, pp. 81–85, Feb. 2010.
[7] N. Komninos, E. Philippou, and A. Pitsillides, "Survey in smart grid and smart home security: Issues, challenges and countermeasures," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1933–1954, 4th Quart., 2014.
[8] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Rev.*, vol. 41, no. 2, pp. 303–332, 1999.
[9] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, "Report on post-quantum cryptography," NIST, Gaithersburg, MD, USA, Tech. Rep. 8105, 2016.
[10] L. Botros, M. J. Kannwischer, and P. Schwabe, "Memory-efficient high-speed implementation of kyber on cortex-M4," in *Proc. Int. Conf. Cryptol. Afr.* Cham, Switzerland: Springer, 2019, pp. 209–228.
[11] Y. Xing and S. Li, "A compact hardware implementation of CCA-secure key exchange mechanism CRYSTALS-KYBER on FPGA," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2021, no. 2, pp. 328–356, Feb. 2021.
[12] T. Fritzmann, G. Sigl, and J. Sepúlveda, "RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography," *IACR Trans. Cryptograph. Hardw. Embedded Syst.*, vol. 2020, no. 4, pp. 239–280, Aug. 2020.
[13] A. Khalid, S. McCarthy, M. O'Neill, and W. Liu, "Lattice-based cryptography for IoT in a quantum world: Are we ready?" in *Proc. IEEE 8th Int. Workshop Adv. Sensors Interfaces (IWASI)*, Jun. 2019, pp. 194–199.
[14] 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," NIST, Tech. Rep., 2020, vol. 2, no. 4, pp. 1–43.
[15] S. N. Islam, Z. Baig, and S. Zeadally, "Physical layer security for the smart grid: Vulnerabilities, threats, and countermeasures," *IEEE Trans. Ind. Informat.*, vol. 15, no. 12, pp. 6522–6530, Dec. 2019.
[16] A. D. Wyner, "The wire-tap channel," *Bell Syst. Tech. J.*, vol. 54, no. 8, pp. 1355–1387, Jan. 1975.
[17] A. Camponogara, H. V. Poor, and M. V. Ribeiro, "PLC systems under the presence of a malicious wireless communication device: Physical layer security analyses," *IEEE Syst. J.*, vol. 14, no. 4, pp. 4901–4910, Dec. 2020.
[18] J. M. Hamamreh, H. M. Furqan, and H. Arslan, "Classifications and applications of physical layer security techniques for confidentiality: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1773–1828, 2nd Quart., 2019.
[19] I. Csiszár and J. Korner, "Broadcast channels with confidential messages," *IEEE Trans. Inf. Theory*, vol. IT-24, no. 3, pp. 339–348, May 1978.
[20] M. Hadley, K. Huston, and T. Edgar, "AGA-12, Part 2 performance test results," Pacific Northwest Nat. Laboratories, Richland, WA, USA, Tech. Rep., 2007.
[21] *P. CODE*, document Tech. IEC specification TS 62351-1, 2018.
[22] Y. Li, P. Zhang, and R. Huang, "Lightweight quantum encryption for secure transmission of power data in smart grid," *IEEE Access*, vol. 7, pp. 36285–36293, 2019.
[23] A. Philips, J. Jayaraj, F. Josh, and P. Venkateshkumar, "Enhanced RSA key encryption application for metering data in smart grid," *Int. J. Pervasive Comput. Commun.*, vol. 17, no. 5, pp. 596–610, Dec. 2021.
[24] D. He, H. Wang, M. K. Khan, and L. Wang, "Lightweight anonymous key distribution scheme for smart grid using elliptic curve cryptography," *IET Commun.*, vol. 10, no. 14, pp. 1795–1802, Sep. 2016.
[25] A. Molina-Markham, G. Danezis, K. Fu, P. Shenoy, and D. Irwin, "Designing privacy-preserving smart meters with low-cost microcontrollers," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Cham, Switzerland: Springer, 2012, pp. 239–253.
[26] V. Mai and I. Khalil, "Design and implementation of a secure cloud-based billing model for smart meters as an Internet of Things using homomorphic cryptography," *Future Gener. Comput. Syst.*, vol. 72, pp. 327–338, Jul. 2017.
[27] S. Finster and I. Baumgart, "Privacy-aware smart metering: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 1088–1101, 2nd Quart., 2015.
[28] P. Kumar, Y. Lin, G. Bai, A. Paverd, J. S. Dong, and A. Martin, "Smart grid metering networks: A survey on security, privacy and open research issues," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2886–2927, 3rd Quart., 2019.
[29] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annu. Symp. Found. Comput. Sci.*, Nov. 1994, pp. 124–134.
[30] F. Borges, P. R. Reis, and D. Pereira, "A comparison of security and its performance for key agreements in post-quantum cryptography," *IEEE Access*, vol. 8, pp. 142413–142422, 2020.
[31] C. Cheng, Y. Qin, R. Lu, T. Jiang, and T. Takagi, "Batten down the hatches: Securing neighborhood area networks of smart grid in the quantum era," *IEEE Trans. Smart Grid*, vol. 10, no. 6, pp. 6386–6395, Nov. 2019.
[32] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-quantum lattice-based cryptography implementations: A survey," *ACM Comput. Surveys*, vol. 51, no. 6, pp. 1–41, Feb. 2019.
[33] N. Gupta, A. Jati, A. K. Chauhan, and A. Chattopadhyay, "PQC acceleration using GPUs: FrodoKEM, NewHope, and Kyber," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 3, pp. 575–586, Mar. 2021.
[34] Y. Huang, M. Huang, Z. Lei, and J. Wu, "A pure hardware implementation of CRYSTALS-KYBER PQC algorithm through resource reuse," *IEICE Electron. Exp.*, vol. 17, no. 17, 2020, Art. no. 20200234.
[35] J. Buchmann, F. Göpfert, T. Güneysu, T. Oder, and T. Pöppelmann, "High-performance and lightweight lattice-based public-key encryption," in *Proc. 2nd ACM Int. Workshop IoT Privacy, Trust, Secur.*, May 2016, pp. 2–9.
[36] D. Liu, N. Li, J. Kim, and S. Nepal, "Compact-LWE: Enabling practically lightweight public key encryption for leveled IoT device authentication," *Cryptol. ePrint Arch.*, vol. 2017, pp. 1–28, Dec. 2017.
[37] A. Marotzke, "A constant time full hardware implementation of streamlined NTRU prime," in *Proc. Int. Conf. Smart Card Res. Adv. Appl.* Cham, Switzerland: Springer, 2020, pp. 3–17.
[38] T. Fritzmann, U. Sharif, D. Müller-Gritschneder, C. Reinbrecht, U. Schlichtmann, and J. Sepulveda, "Towards reliable and secure post-quantum co-processors based on RISC-V," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2019, pp. 1148–1153.
[39] U. Banerjee, T. S. Ukyab, and A. P. Chandrakasan, "Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols," *CoRR*, vol. abs/1910.07557, pp. 1–40, Jun. 2019.
[40] 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 Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 8, pp. 2672–2684, Aug. 2020.
[41] Z. Zhou, D. He, Z. Liu, M. Luo, and K.-K.-R. Choo, "A software/hardware co-design of crystals-dilithium signature scheme," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 14, no. 2, pp. 1–21, Jun. 2021.
[42] L. A. Kumar, V. Indragandhi, R. Selvamathi, V. Vijayakumar, L. Ravi, and V. Subramaniyaswamy, "Design, power quality analysis, and implementation of smart energy meter using Internet of Things," *Comput. Electr. Eng.*, vol. 93, Jul. 2021, Art. no. 107203.
[43] C.-C. Sun, A. Hahn, and C.-C. Liu, "Cyber security of a power grid: State-of-the-art," *Int. J. Elect. Power Energy Syst.*, vol. 99, pp. 45–56, Jul. 2018.
[44] A. Camponogara, H. V. Poor, and M. V. Ribeiro, "Physical layer security of in-home PLC systems: Analysis based on a measurement campaign," *IEEE Syst. J.*, vol. 15, no. 1, pp. 617–628, Mar. 2020.
[45] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, and B. Burkett, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.
[46] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Des., Codes Cryptogr.*, vol. 75, no. 3, pp. 565–599, 2015.
[47] V. Lyubashevsky, "Lattice-based identification schemes secure under active attacks," in *Proc. Public Key Cryptogr.*, vol. 4939, R. Cramer, Ed. Berlin, Germany: Springer, Mar. 2008, pp. 162–179.
[48] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," *J. ACM*, vol. 60, no. 6, pp. 1–35, Nov. 2013.
[49] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila, "Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2016, pp. 1006–1018.

[50] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange: A new hope," in *Proc. 25th USENIX Secur. Symp.*, Aug. 2016, pp. 327–343.
[51] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS—Kyber: A CCA-secure module-lattice-based KEM," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS P)*, Apr. 2018, pp. 353–367.
[52] *CRYSTALS Cryptographic Suite for Algebraic Lattices*. Accessed: Aug. 8, 2022. [Online]. Available: https://pq-crystals.org/kyber/index.shtml
[53] O. Regev, "The learning with errors problem (invited survey)," in *Proc. IEEE 25th Annu. Conf. Comput. Complex.*, Jun. 2010, pp. 191–204.
[54] E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," in *Proc. Annu. Int. Cryptol. Conf.* Cham, Switzerland: Springer, 1999, pp. 537–554.
[55] Xilinx. *Zynq-7000 SoC Data Sheet: Overview*. Accessed: Jul. 2018. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf
[56] M. J. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions," NIST, Gaithersburg, MD, Tech. Rep. 202, 2015.
[57] Xilinx. *AXI-MM Memory-Mapped Interface*. Accessed: Aug. 2022. [Online]. Available: https://docs.xilinx.com/r/en-U.S./pg231-v-proc-ss/AXI-MM-Memory-Mapped-Interface

VINÍCIUS L. R. DA COSTA received the B.Sc. and M.Sc. degrees in electrical engineering from the Federal University of Juiz de Fora (UFJF), Brazil, in 2015 and 2017, respectively, where he is currently pursuing the D.Sc. degree. His major research interests include signal processing, firmware, hardware development, smart grids, and post-quantum cryptography.



ÂNDREI CAMPONOGARA (Member, IEEE) received the dual B.Sc. degree in industrial design and in computer engineering from the Federal University of Santa Maria (UFSM), Rio Grande do Sul, Brazil, in 2010 and 2014, respectively, and the M.Sc. and D.Sc. degrees in electrical engineering from the Federal University of Juiz de Fora (UFJF), State of Minas Gerais, Brazil, in 2016 and 2020, respectively. He is currently an Adjunct Professor with the Federal University of Paraná (UFPR). His research interests include digital signal processing, power line communication, wireless communications, hybrid communication, physical layer security, and the Internet of Everything.

JULIO LÓPEZ received the B.Sc. and M.Sc. degrees in mathematics from the University of Valle, Colombia, in 1982 and 1988, respectively, the M.A. degree in mathematics from the University of Texas at Austin, in 1991, and the D.Sc. degree in computer science from the University of Campinas, Brazil, in 2000. He has been an Associated Professor with the Institute of Computing, University of Campinas, since 2004. His primary research interests include cryptographic engineering and post-quantum cryptography.

MOISÉS V. RIBEIRO (Senior Member, IEEE) received the B.S. degree in electrical engineering from the Federal University of Juiz de Fora (UFJF), State of Minas Gerais, Brazil, in 1999, and the M.Sc. and D.Sc. degrees in electrical engineering from the University of Campinas, São Paulo, Brazil, in 2001, and 2005.

He was a Visiting Scholar at the University of California in Santa Barbara, CA, USA, in 2004, a Visiting Professor (2005–2007), and also an Assistant Professor (2007–2015) at UFJF. Since 2015, he has been an Associate Professor with UFJF. He is also the Director of the Brazilian National Institute of Science and Technology for Electricity (INERGE) and the INERGE Embrapii Unit. He has Co-Founded Smarti9 Ltd., and Wari Ltd., in 2012 and 2015, respectively. He had served as the Secretary of the IEEE ComSoc TC-PLC. He has advised 44 graduate students in these fields and authored over 240 peer-reviewed papers and nine book chapters. He holds 13 issued/pending patents. His research interests include signal processing, power line communication, wireless communication, artificial, and the Internet of Things.

Dr. Ribeiro was a recipient of Fulbright Visiting Professorship at Stanford University, Stanford, CA, USA, in 2011, and Princeton University, Princeton, NJ, USA, in 2012. He was awarded Student Awards from 2001 IEEE IECON and 2003 IEEE ISIE, Winner of 2014 I2P Global Competition, Honorable Mention in 2014 Global Venture Labs Investment Competition, 3rd Place Prêmio Mineiro de Inovação 2014, Engie Brazil Innovation Award 2016, and Unicamp Inventor Award in 2017 and 2018. He was the General Chair of the 2010 IEEE ISPLC, 2013 IWSGC, and SBrT 2015, and a Guest Co-Editor for Special Issues in the EURASIP Journal on Advances in Signal Processing and EURASIP *Journal of Electrical and Computer Engineering*.