Universidade Federal de Juiz de Fora
Programa de Pós-Graduação em Engenharia Eletrica
Doutorado em Engenharia Elétrica

Vinícius Lagrota Rodrigues da Costa

A Quantum-Resistant Advanced Metering Infrastructure

Juiz de Fora
2023

Ficha catalográfica elaborada através do Modelo Latex do CDC da UFJF com os dados fornecidos pelo(a) autor(a)

Costa, V. L. R. da.
A Quantum-Resistant Advanced Metering Infrastructure / Vinícius Lagrota Rodrigues da Costa. – 2023.
123 f. : il.

Orientador: Prof. Dr. Moisés Vidal Ribeiro
Coorientador: Prof. Dr. Julio López
Tese de Doutorado – Universidade Federal de Juiz de Fora, Doutorado em Engenharia Elétrica. Programa de Pós-Graduação em Engenharia Eletrica, 2023.

1. Advanced metering infrastructure. 2. Smart metering. 3. Post-quantum cryptography. 4. Security. 5. Privacy. 6. Field programmable gate array. 7. Processor. 8. Microcontroller. . I. Ribeiro, M. V., orient. II. López, J, coorient. III. Título.

Vinícius Lagrota Rodrigues da Costa

A Quantum-Resistant Advanced Metering Infrastructure

Tese de doutorado apresentada ao Programa de Pós-Graduação em Engenharia Eletrica da Universidade Federal de Juiz de Fora, na área de concentração em sistemas eletrônicos, como requisito parcial à obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Prof. Dr. Moisés Vidal Ribeiro
Coorientador: Prof. Dr. Julio López

Juiz de Fora
2023

FEDERAL UNIVERSITY OF JUIZ DE FORA

RESEARCH AND GRADUATE PROGRAMS OFFICE

Vinícius Lagrota Rodrigues da Costa

Quantum-Resistant Advanced Metering Infrastructure

Thesis submitted to the Graduate Program in Electrical Engineering of the Federal University of Juiz de Fora as a partial requirement for obtaining a Doctor's degree in Electrical Engineering.
Concentration area: Electronic Systems

Approved on 15 of June of 2023.

EXAMINING BOARD

Prof. Dr. Moisés Vidal Ribeiro – Academic Advisor
Federal University of Juiz de Fora

Prof. Dr. Julio César López Hernández – Academic co-supervisor
State University of Campinas

Prof. Dr. Luciano Manhães de Andrade Filho
Federal University of Juiz de Fora

Prof. Dr. Leandro Rodrigues Manso Silva
Federal University of Juiz de Fora

Prof. Dr. Fábio Borges de Oliveira
National Laboratory for Scientific Computing

Prof. Dr. Luis Antonio Brasil Kowada
Fluminense Federal University

Juiz de Fora, 06/15/2023.

Documento assinado eletronicamente por **Luciano Manhaes de Andrade Filho**, **Professor(a)**, em 15/06/2023, às 17:40, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Julio César López Hernández**, **Usuário Externo**, em 16/06/2023, às 09:32, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no Portal do SEI-Ufjf (www2.ufjf.br/SEI) através do ícone Conferência de Documentos, informando o código verificador **1309881** e o código CRC **65360BBB**.

To my wife Larissa
To my parents Luís Guilherme and Tereza Maria
To my brother Vítor

AGRADECIMENTOS

Em primeiro lugar, agradeço a toda minha família. Em especial, à minha esposa, Larissa, pelo amor incondicional, apoio e parceria durante toda essa jornada. Sem ela, a conclusão desta etapa não seria possível. Agradeço ao meu pai, Luís Guilherme, pelo incentivo e ajuda em toda minha caminha acadêmica. Agradeço a minha mãe, Tereza, por todo amor, carinho e suporte em todos os momentos. Agradeço ao meu irmão, Vítor, por estar sempre ao meu lado e por toda a paciência. Agradeço ainda as minhas avós, Ezir (*in memoriam*) e Maria Lúcia, que participaram ativamente de toda minha educação. Agradeço ainda a cada um dos meus familiares por me apoiarem durante todo o processo.

Agradeço a todos os meus amigos que me acompanharam na jornada do doutorado. Em especial, aos amigo Mateus Lima Filomeno, que agradeço pela constante parceria no doutorado. Agradeço ao amigo Ândrei Camponogara pela troca de experiências, sempre impulsionando o nosso crescimento pessoal. Ao amigo Leonardo Dib, eu agradeço por toda sua disponibilidade em me aconselhar neste trabalho. Ainda, agradeço a todos os colegas do Laboratório de Comunicações (LCom), alunos de iniciação científica, mestrandos, doutorandos e pós-doutorandos, que estiverem comigo ao longo dos últimos anos.

Agradeço ainda a todos os colegas do Centro de Pesquisa e Desenvolvimento para a Segurança das Comunicações (CEPESC), em especial à Coordenação-Geral de Pesquisa e Desenvolvimento (CGPD) e à Coordenação de Pesquisa Avançada (CPA). Gostaria de dedicar um agradecimento especial ao Rodrigo Pacheco, um grande amigo que me ajudou ativamente na conclusão deste trabalho e ao Thiago Araújo, por todos seus conselhos profissionais e pessoais durante esta jornada. Vocês são modelos de profissionais e pessoas que eu busco seguir.

Os meus agradecimentos se estendem ainda a todos os profissionais que direta ou indiretamente contribuíram para a minha formação profissional. Sou muito grato ao Professor Julio López que aceitou o desafio de me coorientar e contribuir com o meu trabalho, fazendo parte da minha formação. Saiba que você foi fundamental nesta jornada. Ao Professor Moisés Vidal Ribeiro, meu orientador, eu agradeço por me acolher no grupo de pesquisa ainda como aluno de iniciação científica, há mais de 10 anos, por me levantar nos momentos de desânimo, por enfrentar todos os desafios do doutorado ao meu lado e por mais tantas outras coisas. Muito obrigado por tudo, Professor Moisés.

De forma sincera, agradeço ainda a cada um dos professores que compuseram a banca de avaliação deste trabalho, tanto pela leitura e análise cuidadosa do meu trabalho como pelas valiosas contribuições.

Por fim, mas não menos importante, expresso minha gratidão ao povo brasileiro e mineiro que, direta ou indiretamente, apoiou financeiramente este trabalho por meio da Fapemig, Capes, CNPq e Inerge.

“Peace can only come as a natural consequence of universal enlightenment and merging of races,
and we are still far from this blissful realization.”
Nikola Tesla.

RESUMO

Esta tese de doutorado foca na discussão e implementação de uma Infraestrutura de Medição Avançada com Resistência Quântica (do inglês, Quantum-Resistant Advanced Metering Infrastructure - QR-AMI), que emprega esquemas criptográficos assimétricos e simétricos com resistência quântica para suportar ataques proveniente tanto de computadores quânticos, como clássicos. A solução proposta envolve a integração de um Módulo Criptográfico Dedicado com Resistência Quântica (do inglês, Quantum-Resistant Dedicated Cryptographic Modules - QR-DCMs) com Medidores Inteligentes (do inglês, Smart Meter - SM). Os QR-DCMs são projetados para embarcar esquemas criptográficos com resistência quântica adequados para aplicação em AMI. Nesse sentido, é investigado esquemas criptográficos assimétricos com resistência quântica baseado em fortes princípios criptográficos e abordagem com baixo uso de recursos para AMIs. Além disso, é analisado a implantação prática de um esquema com resistência quântica em QR-AMIs. Dois candidatos do processo de padronização da criptografia pós-quântica (do inglês, post-quantum cryptography - PQC) do Instituto Nacional de Padrões e Tecnologia (do inglês, National Institute of Standards and Technology - NIST), FrodoKEM e CRYSTALS-Kyber, são avaliados devido à adesão a fortes princípios criptográficos e abordagem com baixo uso de recursos. A viabilidade de embarcar esses esquemas em QR-DCMs em um contexto de AMI é avaliado por meio de implementação em software em hardwares de baixo custo, como um microcontrolador e processador, e implementações conjunta hardware/software usando um sistema em um chip (do inglês, System-on-a-Chip - SoC) com Arranjo de Porta Programável em Campo (do inglês, Field-Programmable Gate Array - FPGA). Resultados experimentais mostram que o tempo de execução para os esquemas FrodoKEM e CRYSTALS-Kyber em dispositivos SoC FPGA é, ao menos, um terço mais rápido que implementações em software. Além disso, os tempos de execuções atingidos e o uso de recursos demonstram a viabilidade desses esquemas para aplicações em AMI. O esquema CRYSTALS-Kyber parece ser uma escolha superior em todos os cenários, exceto quando fortes primitivas criptográficas são necessárias, ao menos teoricamente. Devido à falta de SMs no mercado que suportem esquemas criptográficos assimétricos com resistência quântica, um QR-DCM embarcando esquemas com resistência quântica é implementado e avaliado. Quanto à escolha do hardware para os QR-DCMs, microcontroladores são preferíveis em situações que requerem poder de processamento reduzido, enquanto dispositivos SoC FPGA são mais adequados para quando é demandado maior poder de processamento. O uso de recurso e o resultado do tempo de execução demonstram a viabilidade da implementação de AMI baseada em QR-DCMs, ou seja, uma QR-AMI, usando microcontroladores e dispositivos SoC FPGA.

Palavras-chave: Infraestrutura Avançada de Medição. Medição Inteligente. Criptografia Pós-Quântica. Segurança. Privacidade. Arranjo de Porta Programável em Campo. Processador. Microcontrolador.

ABSTRACT

This dissertation focuses on discussing and implementing a Quantum-Resistant Advanced Metering Infrastructure (QR-AMI) that employs quantum-resistant asymmetric and symmetric cryptographic schemes to withstand attacks from both quantum and classical computers. The proposed solution involves the integration of Quantum-Resistant Dedicated Cryptographic Modules (QR-DCMs) within Smart Meters (SMs). These QR-DCMs are designed to embed quantum-resistant cryptographic schemes suitable for AMI applications. In this sense, it investigates quantum-resistant asymmetric cryptographic schemes based on strong cryptographic principles and a lightweight approach for AMIs. In addition, it examines the practical deployment of quantum-resistant schemes in QR-AMIs. Two candidates from the National Institute of Standards and Technology (NIST) post-quantum cryptography (PQC) standardization process, FrodoKEM and CRYSTALS-Kyber, are assessed due to their adherence to strong cryptographic principles and lightweight approach. The feasibility of embedding these schemes within QR-DCMs in an AMI context is evaluated through software implementations on low-cost hardware, such as microcontroller and processor, and hardware/software co-design implementations using System-on-a-Chip (SoC) devices with Field-Programmable Gate Array (FPGA) components. Experimental results show that the execution time for FrodoKEM and CRYSTALS-Kyber schemes on SoC FPGA devices is at least one-third faster than software implementations. Furthermore, the achieved execution time and resource usage demonstrate the viability of these schemes for AMI applications. The CRYSTALS-Kyber scheme appears to be a superior choice in all scenarios, except when strong cryptographic primitives are necessitated, at least theoretically. Due to the lack of off-the-shelf SMs supporting quantum-resistant asymmetric cryptographic schemes, a QR-DCM embedding quantum-resistant scheme is implemented and evaluated. Regarding hardware selection for QR-DCMs, microcontrollers are preferable in situations requiring reduced processing power, while SoC FPGA devices are better suited for those demanding high processing power. The resource usage and execution time outcomes demonstrate the feasibility of implementing AMI based on QR-DCMs (i.e., QR-AMI) using microcontrollers or SoC FPGA devices.

Key-words: Advanced metering infrastructure. Smart metering. Post-quantum cryptography. Security. Privacy. Field programmable gate array. Processor. Microcontroller.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AAD	Additional Authenticated Data
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
AMI	Advanced Metering Infrastructure
AMR	Automatic Meter Reading
ANEEL	National Agency of Electric Energy
API	Application Programming Interface
ARM	Advanced RISC Machine
ASIC	Application-Specific Integrated Circuit
AXI-MM	Advanced eXtensible Interface Memory Mapped
BB	Broadband
BRAM	Block Random Access Memory
BSI	German Federal Office for Information Security
CEMIG	Minas Gerais Electric Power Company
CRYSTALS	Cryptographic Suite for Algebraic Lattices
COSEM	Companion Specification for Energy Metering
CTR	Counter
DCM	Dedicated Cryptographic Module
DCU	Data Concentrator Unit
DER	Distributed Energy Resources
DLMS	Device Language Message Specification
DH	Diffie-Hellman
DMA	Direct Memory Access
DoS	Denial of Service
DSP	Digital Signal Processing
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
FF	Flip Flop
FIPS	Federal Information Processing Standards
FPGA	Field Programmable Gate Array

GeMSS	Great Multivariate Short Signature
GCM	Galois Counter Mode
GPU	Graphics Processing Unit
HQC	Hamming Quasi-Cyclic
IND-CCA2	Adaptive Chosen Ciphertext Attack
IoT	Internet of Things
ISA	Instruction Set Architecture
JSON	JavaScript Object Notation
KEM	Key Encapsulation Mechanism
KEX	Key Exchange
LWE	Learning With Errors
LUT	Lookup Table
ms	milliseconds
MDMS	Meter Data Management System
M-LWE	Module-Learning With Errors
M-LWR	Module-Learning With Rounding
N-V	Nitrogen-Vacant
NAN	Neighborhood Area Network
NIST	National Institute for Standards and Technology
NTRU	N-th degree Truncated polynomial Ring Units
NTT	Number Theoretic Transform
PKE	Public-key encryption
PL	Programmable Logic
PLC	Power Line Communication
PLL	Phase-Locked Loop
PQC	Post-Quantum Cryptography
PS	Processing System
QKD	Quantum Key Distribution
QR-AMI	Quantum-Resistant AMI
QR-DCM	Quantum-Resistant - Dedicated Cryptographic Module
RF	Radio Frequency

R-LWE	Ring-Learning With Errors
RSA	Rivest-Shamir-Adleman
SHAKE	Secure Hash Algorithm and Keccak
SG	Smart Grid
SM	Smart Meter
SoC	System-on-a-Chip
SoM	System on Module
SVP	Shortest Vector Problem
TLS	Transport Layer Security
WAN	Wide Area Network

CONTENTS

1 INTRODUCTION

Smart Grid (SG) have recently emerged as a vital solution for long-standing challenges in the electricity sector [1–3], including handling numerous loads, integrating renewable energy sources, and ensuring reliability, efficiency, sustainability, and flexibility, among others. Electric power systems worldwide are rapidly transforming from manual operations and on-site control to remote, intelligent, and efficient control based on SGs. For instance, power generation plants are evolving from centralized large-scale facilities to smaller, distributed power plants and community power generation (i.e., prosumers). Although SG is a powerful concept, it is only effective when communication between parties occurs, and consumers, prosumers, and equipment are adequately sensed, controlled, and communicate with each other [4, 5], which raise the necessity of deploying Advanced Metering Infrastructures (AMIs). In this context, intelligent devices, such as Smart Meters (SMs), emerge as essential equipment for large-scale AMI deployments. However, SMs generate sensitive personal data containing users' confidential information (i.e., energy consumption profiles) [6].

Recently, interest in deploying AMIs has grown, with SM pilot projects launched primarily in European nations. By 2024, approximately 225 million SMs for electricity and 51 million for gas will be installed in Europe, with 44% of European consumers having smart gas meters and 77% having smart electric meters [7]. In North America, it is expected that over 85% of households in Canada and 70% in the US will be equipped with SMs. Conversely, the situation in Brazil is distinct. While European AMI implementation aligns with a broader policy framework related to climate change, Brazilian efforts stem primarily from National Agency of Electric Energy (ANEEL) and are limited to the electricity sector. Recently, Enel Brasil, a Brazilian electric utility, installed 150 thousand SMs in São Paulo, with a goal of reaching 300 thousand [8]. The budget for installing 300 thousand SMs is around US$ 45.4 million[1], with US$ 24.2 million financed by ANEEL for research and development, and the remaining US$ 25.2 million contributed by Enel Brasil. Additionally, Minas Gerais Electric Power Company (CEMIG), a publicly traded electric utility controlled by the state of Minas Gerais, is replacing its meters with smart ones, having already replaced around 100 thousand units in Belo Horizonte. The goal is to replace at least 500 thousand meters in the coming months, requiring an investment of around US$ 40 million [9]. Apart from these initiatives, no other wide-scale deployment projects for SMs are known.

In the coming years, an annual investment of US$ 600 billion is anticipated through 2030 in the energy sector, which is expected to boost the AMI market from US$ 19.24 billion in 2021 to US$ 77.37 billion by 2029. Consequently, research, development, and innovation efforts will involve public and private organizations working in partnership to achieve common energy goals. This vision is primarily implemented in Europe, where eight major electric utilities have partnered with large technology companies like Intel. Furthermore, equipment integrated into

[1] Conversion calculated based on an approximate exchange rate of US$ 1.00 equivalent to R$ 5.00.

the AMI must be as low-cost as possible, given that millions (or even billions) of such devices will be dispersed across countries and continents, forming part of the AMI.

Considering the widespread use and importance of AMI for utilities, it is essential to raise security and privacy awareness regarding potential vulnerabilities in AMIs to avoid and/or minimize security breaches and unauthorized access to SMs data [10–13]. For example, key generation and distribution in cryptographic schemes may result in security breaches when transmitting SM data [14, 15]. In this context, numerous research efforts have been devoted to the security of AMI, with the majority focusing on cryptography to ensure security and privacy for customers and utilities [16–20]. These studies, however, focused on cryptographic schemes mainly aimed at securing data against attacks from classical computers. When it comes to practical asymmetric cryptography – e.g., Public-key encryption (PKE) and Key Encapsulation Mechanism (KEM) – the security is typically based on the hardness of factoring integers or computing discrete logarithms – e.g., Rivest-Shamir-Adleman (RSA) and Elliptic Curve Cryptography (ECC) [19]. Such asymmetric cryptographic schemes, however, will no longer be secure when a sufficiently powerful and stable quantum computer runs Shor's algorithm, a polynomial-time quantum algorithm for integer factorization and discrete logarithm problem [21]. Regarding symmetric cryptography, it will not be significantly impacted as asymmetric will be with the advent of quantum computers. Grover proposed a quantum algorithm for database search [22] and, based on this algorithm, an attack, called Grover's attack, on symmetric cryptography was proposed [23], which reduced the security of the key to half its length. Although relevant, it does not threaten symmetric cryptography security as longer keys can be used to securely protect the information. Consequently, all secure and standardized symmetric schemes are potentially quantum-resistant.

Aiming to overcome the quantum threat, quantum-resistant asymmetrical and symmetric cryptographic schemes are mandatory for securing data [24]. The former performs a quantum-resistant key agreement, while the latter encrypts/decrypts and authenticates sensitive data. In this sense, investigations focusing on asymmetrical Post-Quantum Cryptography (PQC) schemes – i.e., schemes based on quantum-resistant primitives – targeting SG and SM are of utmost importance. In this respect, Cheng *et al.* [25] proposed a scheme for mutual authentication between SMs and gateway based on PQC. Borges *et al* [26] proposed a security comparison of key agreement protocols between PQC primitives and traditional problems. Moreover, Ahn *et al.* [27] evaluated quantum-resistant key distribution based on either Quantum Key Distribution (QKD) and PQC techniques in Distributed Energy Resources (DER).

Despite the recent efforts to address several matters on the AMI and PQC, to the extent of the authors' knowledge, the literature lacks investigations of practical implementation of a Quantum-Resistant AMI (QR-AMI) – i.e., an AMI based on quantum-resistant asymmetric and symmetric schemes. These timely investigations are of utmost importance to point out the benefits of an AMI capable of dealing with the threats of quantum computers in electric power systems. Moreover, no study compared different platforms – e.g., microcontrollers and

System-on-a-Chip (SoC) Field Programmable Gate Array (FPGA) devices – to enable a QR-AMI. This analysis must be carefully evaluated as large-scale implementations of AMIs present cost constraints, indirectly affecting the choice of platforms that embed quantum-resistant schemes. As a powerful quantum computer moves closer to being a reality, this type of analysis is of utmost importance as current sensitive data traveling through AMIs might already be at risk if they are being stored for future use in quantum computers.

1.1 OBJECTIVES

This dissertation addresses the feasibility of practical implementation of quantum-resistant schemes in AMIs. In this sense, this dissertation has the following objectives:

- To discuss the necessity of PQC schemes in a quantum era and identify promising candidates with interesting characteristics for AMIs. To this end, a brief review of PQC schemes and the National Institute for Standards and Technology (NIST) PQC standardization process is presented. Among all candidates in the competition, two PQC schemes (i.e., FrodoKEM and CRYSTALS-Kyber) are deeply assessed because they are promising schemes to be applied in AMIs as KEM.

- To investigate the use of a PQC scheme to perform a KEM, using a conservative approach (i.e., FrodoKEM) for ensuring quantum-resistant functionality in AMIs. In this regard, a thorough analysis is provided, approaching different hardware (i.e., microcontrollers and SoC FPGA devices) with different implementation techniques (i.e., software and hardware/software co-design implementations). Also, the most time-consuming routines are properly evaluated, and efficient hardware acceleration is provided.

- To assess the use of a lightweight quantum-resistant approach (i.e., CRYSTALS-Kyber) for performing KEM in AMIs. To do so, a meticulous analysis is provided based on different hardware (i.e., microcontrollers and SoC FPGA devices) with different implementation techniques (i.e., software and hardware/software co-design implementations). The most time-consuming routines are identified and evaluated, and optimized hardware acceleration provided.

- To propose a QR-AMI, which relies on Quantum-Resistant - Dedicated Cryptographic Modules (QR-DCMs), a dedicated cryptographic module embedded with quantum-resistant schemes. To do so, it is discussed the scenarios that best suit the QR-DCM be based on microcontrollers, low-cost processor, or SoC FPGA devices. Also, the asymmetric FrodoKEM and CRYSTALS-Kyber schemes are compared and their use in each scenario is discussed. The use of the symmetric Advanced Encryption Standard (AES)-256-Galois Counter Mode (GCM) scheme is also evaluated.

1.2 DISSERTATION OUTLINE

The remainder of this document is organized as follows:

- Chapter 2 presents a revision of the state-of-the-art and the problem formulation regarding AMI inserted in a quantum era. In this sense, this chapter makes a complete literature review of works in which security in AMI is discussed and points out gaps that must be filled. Also, a problem formulation is presented, which considers the gaps in the literature, and the research questions approach in this dissertation are presented.

- Chapter 3 analyzes the importance of AMIs for the success of SGs. The current global scenario of AMIs and its key elements and concepts are discussed. Also, an extensive discussion regarding the security and privacy issues in AMIs is provided.

- Charter 4 discusses the physical principles of a quantum computer and how its power can be explored. A brief review of cryptographic principles is provided and a deep discussion regarding PQC schemes is presented, highlighting the NIST PQC standardization process and its main candidates. Finally, suitable PQC schemes for AMIs are deeply discussed.

- Chapter 5 investigates the use of the FrodoKEM scheme as KEM for ensuring quantum-resistant key exchange between parties. The background of the FrodoKEM is discussed, and software analysis is provided. Based on this analysis, an optimized proposal of a hardware/software co-design implementation of the FrodoKEM scheme is introduced, accelerating the most time-consuming routines.

- Chapter 6 assesses the use of the CRYSTALS-Kyber scheme as KEM also for ensuring quantum-resistant key exchange between parties. Similarly, the background of the CRYSTALS-Kyber is presented and analyzed based on the software provided. Relying on this analysis, an optimized proposal of a hardware/software co-design implementation of the CRYSTALS-Kyber scheme, accelerating the most time-consuming routines.

- Chapter 7 proposes a practical implementation of a QR-AMI based on QR-DCMs. This proposal is based on the FrodoKEM and CRYSTALS-Kyber asymmetric cryptographic schemes and on the AES-256-GCM symmetric cryptographic scheme, all of them quantum-resistant. Implementations using different hardware are provided and a discussion regarding the best use-case scenario for each hardware and implementation technique is assessed.

- Chapter 8 ends this dissertation by stating concluding remarks.

2 PROBLEM STATEMENT

The discussion of quantum-resistant cryptographic schemes has gained relevance recently, but several issues must be mastered to advance their effective and widespread introduction in AMIs. While theoretical results have supported the introduction of new and effective PQC schemes, several gaps related to implementation aspects remain open. Aiming to advance this discussion, this chapter has the following objectives:

- Provide a literature review about security in AMIs and the state-of-the-art of PQC schemes for AMIs. Also, it pays attention to several issues that must be investigated for advancing the widespread use of PQC schemes in AMIs.
- Detail the problems raised by quantum computers on AMIs if only non-quantum (classical) cryptography is considered to ensure data security and privacy. Also, present the research questions investigated in the following chapters.

In this chapter, Section 2.1 presents a literature review, highlighting the state-of-the-art and future trends in AMI security, particularly those based on cryptography, and identifies a few gaps related to the implementation of PQC schemes requiring pressing attention. Section 2.2 formulates the investigated problem and concisely describes the research questions pursued in this dissertation.

2.1 LITERATURE REVIEW

It is well-known that AMIs transmit sensitive data among consumers, prosumers, and utilities. Therefore, ensuring security and privacy is a critical concern that must be carefully addressed today and in the future, particularly with the advent of quantum computers. To ensure the security of SM data transmitted through AMIs, techniques can be employed to ensure data confidentiality and integrity. Cryptography is one of the most used ones. With respect to cryptography targeting AMIs, there are different approaches to providing data security, especially when hardware-constrained equipment applies. Currently, traditional cryptographic schemes are widely recommended and used to protect sensitive data through some guidelines and security standards, such as AGA-12 [28] and IEC 62351[1] [29, 30].

Regarding asymmetric cryptography, the schemes recommended by AGA-12 and IEC 62351 may not be efficient enough for embedding in hardware-constrained equipment [31]. In this sense, Philips *et al.* [32] proposed an enhanced-RSA scheme to authenticate SMs equipment in AMI, while He *et al.* [33] introduced a lightweight anonymous key distribution based on ECC to provide anonymity to SMs equipment. Molina-Markham *et al.* [34] presented a feasible implementation of a cryptographic scheme that certifies the SM's read of

[1] IEC 62351 recommends RSA and ECC as asymmetric cryptographic schemes and AES as a symmetric cryptographic scheme to protect power data.

fine-grained electricity using a low-cost MSP430 microcontroller. Further research of asymmetric cryptography applied to AMI are discussed in [20, 35–37] and references therein.

On the other hand, symmetric cryptography requires less computation power than asymmetric cryptography [19]. Consequently, traditional schemes are usually efficient enough to be deployed in hardware-constrained equipment, as discussed in [38–41]. In AMI context, Saxena *et al.* [17] proposed dynamic secrets and shared secret schemes while Liu *et al.* [18] suggested a secure mechanism between SMs and Meter Data Management System (MDMS). Homomorphic cryptography is also evaluated in [42], 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 AMIs can be found in [43, 44].

Based on this discussion, traditional cryptographic schemes can provide data security in AMIs. However, as will be further discussed in Section 4.3, the advent of quantum computers imposes that data security cannot be guaranteed anymore because a polynomial-time quantum algorithm, called Shor's algorithm [21], can quickly solve traditional problems (i.e., discrete logarithm problems and integer factorization problems) on which current asymmetric cryptographic schemes rely on. Even more, if an eavesdropper is collecting and storing data that travels through AMIs, then, in the future, it might be capable of deciphering it and accessing sensitive information. Based on this discussion, two approaches stand out as possible options for quantum-resistant asymmetric cryptography: QKD and PQC.

QKD is a hardware-based solution that relies on quantum mechanics principles to protect key distribution between parties. In this sense, some researchers use these principles to guarantee privacy against eavesdroppers. A review of QKD protocol and their application in SGs are presented in [45]. Fabio *et al.* [46] described two privacy-enhancing protocols based on QKD focusing on customers' privacy. In [47], QKD is employed for enhancing the security of cloud-based AMIs. Although theoretically possible, a practical implementation of QKD faces challenges such as secret key rate, distance, size, cost, and practical security [48].

On the other hand, PQC stands out as a simpler solution since it is algorithm-based. Regarding asymmetric PQC, there are a considerable number of implementations focusing on specific targets. For instance, Nejatollahi *et al.* [49] presented a complete survey highlighting the most relevant implementations using software, hardware, and hardware/software co-design techniques. Gupta *et al.* [50] presented a software implementation of CRYSTALS-Kyber, FrodoKEM, and NewHope using Graphics Processing Unit (GPU) seeking higher performance, while Huang *et al.* [51] proposed pure hardware implementation of the CRYSTALS-Kyber scheme using FPGA focusing on high performance and seeking to reuse resources. Note that [50] and [51] may not be suitable for AMIs due to their high hardware resource usage and high-cost platforms. In contrast, Buchmann *et al.* [52] presented a lightweight PKE lattice-based scheme on small 8-bit ATXmega128 and 32-bit Advanced RISC Machine (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 [53], 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 [54] 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, Fritzmann *et al.* [55] detailed a RISC-V co-processor for lattice-based cryptography using hardware to accelerate the Number Theoretic Transform (NTT) and hash generation using an SoC FPGA device. Also, Fritzmann *et al.* [56] discussed an Instruction Set Architecture (ISA) for lattice-based cryptography based on a so-called RISQ-V architecture implemented on an SoC FPGA device and Application-Specific Integrated Circuit (ASIC). Furthermore, Banerjee *et al.* [57] and Xin *et al.* [58] presented a crypto-processor for PQC schemes based on lattices, fabricated in TSMC 40 nm and 28 nm 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 [59]. 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, Zhou *et al.* [59] 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.

Considering PQC in AMI, Borges *et al.* [26] presented a concise security comparison of key agreement protocols between PQC primitives and traditional problems. Ahn *et al.* [27] evaluated quantum-resistant key distribution based on either QKD and PQC techniques in DER. Also, Cheng *et al.* [25] proposed a scheme for mutual authentication between SMs and gateway based on PQC. In [26], a security comparison of key agreement protocols between PQC primitives and traditional problems was presented.

Symmetric cryptography will be considerably less impacted by the advent of a quantum computer than asymmetric cryptography. Grover proposed a quantum algorithm for database search [22], on which Grover's attack [23] is based. This attack can reduce the security of the key to half its length as it is a brute-force attack with complexity $O(\sqrt{N})$, where N is the key length in bits. Since the security level of 80-bit is considered secure against brute-force attacks [60], AES-192 and 256 will not be threatened by Grover's attack, at least in the near future, because it reduces the security of these schemes to 96 and 128-bits [61], respectively. Yet, Bogomolec *et al.* [62] proposed a new quantum-resistant symmetric scheme, based on the AES, called eAES, which offers an enhanced complexity regarding the quantum cryptanalysis Grover's attack compared to AES-256.

The awareness about the role and significance of PQC schemes in providing security and privacy in AMIs is growing because theoretical advancements and relevant initiatives, such as the NIST PQC standardization process, have resulted in the introduction of a few PQC schemes that can also be feasible for the electricity sector. However, the literature still presents several gaps regarding practical aspects of quantum-resistant schemes in AMIs. Among these gaps, the following ones deserve attention:

- A comprehensive evaluation of PQC schemes that can be effectively introduced in AMIs, considering their unique characteristics, requirements, and constraints. It requires delving into the strengths and weaknesses of different PQC schemes to determine the most suitable option for secure communication within AMIs.
- A thorough analysis of the performance and computational burden of PQC schemes. By identifying the primary bottlenecks, insights can be provided for advancing acceleration techniques for efficient and effective implementation of PQC schemes in cost-effective hardware in AMIs. This examination involves exploring trade-offs between performance, power consumption, and cost.
- A systematic investigation of how to deploy PQC schemes in AMIs, bearing in mind the processing limitations of SMs. This discussion also needs to include strategies for integrating PQC schemes with existing communication protocols, updating firmware, and interoperability issues.
- A detailed examination of integrating quantum-resistant asymmetric and symmetric cryptographic schemes within AMIs. As large-scale AMI deployments impose cost constraints that motivate the search for cost-effective hardware solutions.

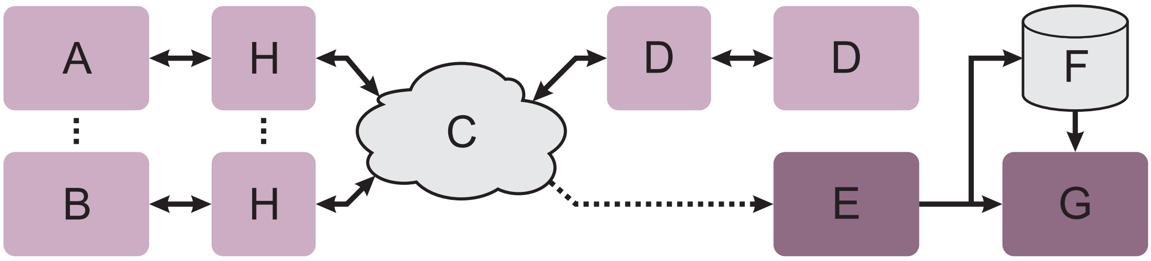
Drawing from the gaps observed in the literature, it is clear that there exists an urgent need to undertake practical investigations to facilitate the design of large-scale AMIs employing quantum-resistant schemes, especially for dealing with SMs, which are hardware-constrained devices, play a crucial role.

2.2 PROBLEM FORMULATION

An illustration of how an eavesdropper uses a classical computer to breach the security of an AMIs is shown in Figure 1. In this illustration, SMs connect to or embed a Dedicated Cryptographic Module (DCM), which implements cryptographic schemes for key agreement between parties and ciphering/deciphering and authentication functionalities. On the electric utility side (i.e., MDMS), another DCM is available to pair with the DCM located on the consumer/prosumer side. Currently, DCMs only embed classical cryptography schemes, which is sufficient for protecting sensitive data transmitted through AMIs given the current computing power of classic computers. In other words, even in the presence of an eavesdropper with a powerful classical computer, the communication between parties remains secure and private.

However, this sense of security may not be entirely justified as the development of quantum computers continues to progress rapidly. Currently, several companies and governments

Figure 1 – A classical scenario where SMs are connected to or embedded a DCM in the presence of an eavesdropper performing a sniffing attack and making use of a classical computer to decipher SM data.



Source: Personal collection.

are investing in the development of quantum computers, with a powerful quantum computer anticipated to be realized within the next two decades. Based on quantum mechanics, the underlying principles of quantum computers render it possible to break both asymmetric and symmetric cryptography, albeit in distinct ways. Although quantum computers are not yet available and may not be for several years, current data communication remains at risk. For example, an eavesdropper could store all communication between parties and decrypt it using a quantum computer in the future. Consequently, relying on classical ciphers in DCMs of AMIs to safeguard sensitive data is no longer secure, considering the looming threat that quantum computers represent, even if it is a future concern.

In light of the current scenario and the impending arrival of powerful quantum computers, it is crucial to employ quantum-resistant schemes to withstand attacks from both quantum and classical computers and, ultimately, to come up with truly secure communication within AMIs. Consequently, it is essential for DCMs to incorporate quantum-resistant schemes, leading to their evolution into the so-called QR-DCMs, which are DCM implementations utilizing quantum-resistant schemes. Figure 2 depicts a scenario where DCMs are replaced by QR-DCMs. Importantly, even in the presence of an eavesdropper with a powerful quantum computer or when sensitive data are stored, data communication between parties remains secure in the long-term using quantum-resistant schemes. In this context, the proposal of QR-AMIs that rely on QR-DCMs, which are equipped with quantum-resistant asymmetric and symmetric cryptographic schemes, is a timely and significant issue to be addressed, as the presence of QR-DCMs is essential for ensuring the security of sensitive data transmitted through the AMI.

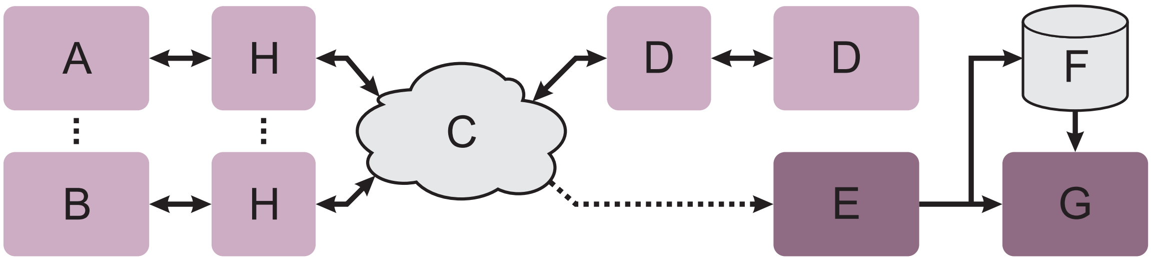
An existing problem is that off-the-shelf SMs do not have enough processing power to run quantum-resistant schemes in their processing unit because these devices are produced at the lowest cost. Moreover, the manufacturers of SMs follow the standards; however, the established standards do not contemplate a quantum-resistant scheme, as this is a recent issue in the electricity sector. As the development of QR-DCMs is vital for securing sensitive data transmitted through AMIs in the quantum era, it is necessary to advance the implementations of quantum-resistant schemes in SMs. For a feasible implementation of QR-DCM, the following approaches can be considered for including quantum-resistant asymmetric and symmetric cryptographic schemes into SMs:

- **Approach #1**: This approach entails increasing the processing capacity of the processor used by a SM, as the QR-DCM would be implemented on it. However, for current SMs, this call for significant hardware redesign, as the processor must be replaced, requiring new certifications. Moreover, this approach does not isolate keys since only one processor is available.
- **Approach #2**: This approach involves adding a dedicated cryptographic processor, where the QR-DCM would be implemented, to the SM that already contains a general-purpose processor. This method also demands considerable hardware redesign for existing SMs, and new certifications would be necessary. Different from **Approach #1**, it provides an additional layer of security, as the keys would be stored in an isolated processor. In this sense, an attacker would need to break the communication layer security to access the general-purpose processor and then break the another layer of security provided by the dedicated cryptographic module. In this sense, compared with **Approach #1**, this approach is safer.
- **Approach #3**: This approach proposes designing a specific module containing quantum-resistant cryptographic primitives connected to the interfacing ports of SMs and the communication infrastructure. It is the easiest and most straightforward way to enable existing SMs to access quantum-resistant cryptographic primitives and it is the convenient choice for avoiding new hardware redesigns and certification processes. However, since the QR-DCM is external to the SM, it may present potential security issues, such as facilitating physical access to the QR-DCM.

Regarding the electric utility side (i.e., MDMS), cryptographic schemes must be implemented within the MDMS, which can have its processing capacity effortlessly upgraded to run quantum-resistant schemes. In this case, **Approach #1** or **#2** applies. The MDMS can also utilize a specific quantum-resistant module, referring to the use of **Approach #3**.

Furthermore, quantum-resistant cryptographic primitives must be implemented in both SMs and MDMS to be effective. However, no standards or protocols exist for quantum-resistant

Figure 2 – A quantum-resistant scenario where SMs are connected to or embedded a QR-DCM in the presence of an eavesdropper performing a sniffing attack and making use of a quantum computer to decipher SM data.



Source: Personal collection.

communications in AMI, and numerous SMs are already installed in the field. In this scenario, a viable direction for promptly protecting AMIs against security attacks executed by classical and quantum computers is to foster **Approach #3**. Consequently, a QR-DCM can be designed and implemented to provide a QR-AMI, which is an AMI based on quantum-resistant schemes. Moreover, QR-DCMs can be easily connected to the existing interfacing ports of SMs and MDMS. In this context, a QR-DCM can overcome the hardware constraints in existing SMs for performing quantum-resistant communications. Note that **Approach #3** can be deemed a short-term solution because it can be easily attached to the current AMIs to provide quantum resistance. Long-term solutions, on the other hand, should be based on **Approaches #1** or **#2**, as they integrate the QR-DCM within the SM, eliminating potential security flaws that an external QR-DCM might present. Among **Approach #1** and **#2**, based on a security perspective, the latter is a superior option, as it adds a security layer by isolating keys in a dedicated cryptographic processor but it may be more expensive. For advancing **Approach #3**, the following research questions are systematically investigated in this dissertation:

- What kind of quantum-resistant schemes can be feasible and useful for SMs belonging to QR-AMI? To answer this question, Chapters 4 details two quantum-resistant asymmetric schemes that offer different trade-offs between security levels and processing costs.
- How feasible is it to implement a quantum-resistant asymmetric cryptographic scheme that, in theory, prioritizes strong cryptographic principles? The answer to this research question should consider analyzing resource usage and timing performance on various hardware using different implementation techniques. Chapter 5 thoroughly discusses this research question.
- How feasible is it to implement a quantum-resistant asymmetric cryptographic scheme that is categorized as a lightweight cryptographic scheme? Like the previous research question, the answer must be based on analyzing resource usage and timing performance on various hardware using different implementation techniques. This research question is extensively debated in Chapter 6.
- How does a QR-AMI perform? And, what kind of hardware resources are required to implement a asymmetric cryptographic scheme based on strong or lightweight cryptographic principles? To answer this research question, Chapter 7, drawing on the results of Chapters 5 and 6, determines the best combination of hardware, implementation techniques, and quantum-resistant schemes for possible scenarios.

2.3 SUMMARY

This chapter has presented a thorough literature review focusing mainly on AMI security. In this sense, the review approach works which uses cryptography to protect sensitive data exchange between parties in AMI using asymmetric and symmetric schemes. Works focused on quantum computer algorithms are also presented, which threats the security of classical cryptography. Aiming to approach this issue, works based on QKD and PQC are cited and discussed. Regarding the latter, works based on this approach are discussed, highlighting the hardware and implementation techniques already studied by the academic community and their suitability for AMIs. Based on the literature review, the gaps in the literature are pointed out. This chapter also assess the problem formulation approached in this dissertation. Briefly, AMIs nowadays are not prepared to secure sensitive data in a quantum era as nowadays AMIs are only equipped with classic cryptography. Considering the rapid advances in the quantum computer development, it is vital to come up with solutions aiming to protect the sensitive data traveling through AMIs from the threat of quantum computers. In this regard, the research questions of this dissertation are presented.

3 ADVANCED METERING INFRASTRUCTURE

In the electricity sector, AMIs play a crucial role in the successful implementation of SG [44]. It is a concept that extends Automatic Meter Reading (AMR). Recently, AMR arose as a step forward to conventional energy metering, where data collection and billing were done manually and on-site. In AMRs, the device (e.g., SMs) automatically sends reports periodically to the utility, enabling one-way communication with consumers. Note that in AMRs, utilities cannot send commands to the metering devices, only the other way around. Aiming to improve this communication infrastructure, the AMI was introduced to allow two-way communication, enabling the utility to send commands to meters, favoring both parts.

In this sense, as sensitive data, such as personnel information, travels through the communication infrastructure, security and privacy are vital for a trustworthy AMIs. Therefore, this chapter seeks to discuss this concerns, although, before approaching the security issues in AMI, it is important to have a big picture of its architecture, discussing its key elements, their responsibilities, and roles. With the AMI architecture in mind, the security issues in AMI must be discussed. As sensitive data travel through the AMI, vulnerabilities and threats faced by AMIs must be mapped, identified, and discussed, aiming to determine the source of these issues and alternatives to avoid them. Furthermore, concepts regarding security and privacy in AMI must be defined and discussed. These concepts are the pillar for a secure AMI and must be considered in the initial phase of it, seeking to avoid future vulnerabilities and threats. The main contributions of this chapter are as follows:

- An introduction of the architecture of an AMI highlighting its main key elements: consumers/prosumers, communication infrastructure, and electric utility. Their importance in the AMI is discussed along with their interaction with each other. Also, their main responsibilities are pointed out and discussed.

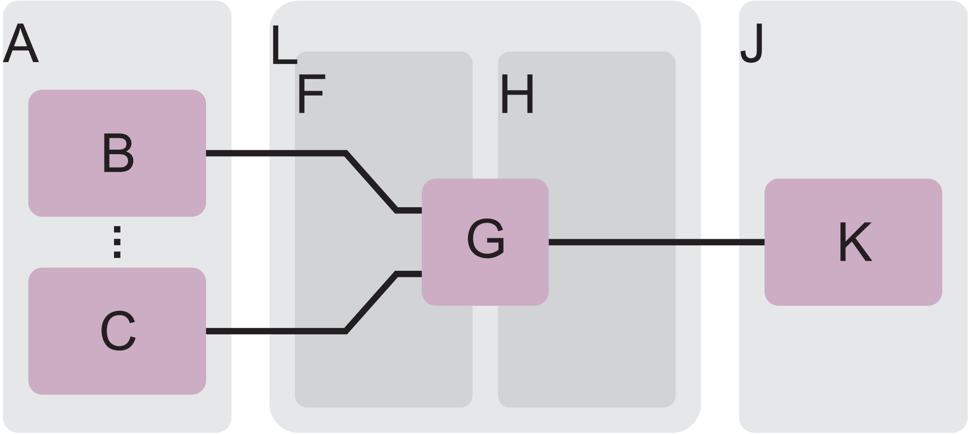
- A thorough discussion regarding security issues in AMIs. In this sense, vulnerabilities, threats, security, and privacy issues in AMIs are presented and classified. The former two present ways that an attacker might explore vulnerabilities and how it threats AMIs security, while the latter two present security and privacy principles that must be followed by all AMIs.

This chapter is organized as follows: In Section 3.1 it is presented the key elements of AMIs and its architecture and Section 3.2 will discuss the vulnerabilities, security, privacy, and threats to provide a better understanding of the security concerns related to the AMI.

3.1 KEY ELEMENTS

The main task of AMIs is to enable two-way communication between customers and utilities benefiting both parties. Customers can benefit from remote consumption control, dynamic

Figure 3 – Key elements of an AMI.



Source: Personal collection.

electricity pricing, consumption forecast, and other functionalities. Electric utilities, on the other hand, can benefit from fault and outage detection, loss measurement, asset monitoring, remote connection and disconnection of customer loads, and non-technical loss detection. Moreover, AMI enables the collection of a vast amount of data to understand load patterns and design automated systems to optimize energy distribution. An AMI with all the aforementioned features contributes to enhancing power quality and energy delivery reliability, benefiting customers and electric utilities alike.

The AMI mainly comprises three key elements: consumers/prosumers, communication infrastructure, and electric utilities, as shown in Figure 3. To comprehensively understand each of these elements, the following subsections will detail their features and requirements. Subsection **3.1.1** focuses on consumers/prosumers, highlighting their functionalities and needs. Subsection **3.1.2** discusses the communication infrastructure, including the technologies involved in two-way communication between consumers/prosumers and the electric utility. Finally, Subsection **3.1.3** explores the electric utility, including the necessary infrastructure and equipment for data collection, processing, and management.

3.1.1 Consumer/prosumer

The term consumer/prosumer in the context of SGs refers to entities such as individuals, households, or organizations that consume energy from the grid while also having the ability to generate and sell energy back to it through distributed energy resources like solar panels, wind turbines, or energy storage systems. This ability to consume and produce energy breaks down the traditional boundaries between energy producers and consumers and plays a vital role in creating more efficient, reliable, and sustainable electric power systems.

The consumer/prosumer side primarily comprises meters, such as SMs, responsible for collecting data and sending periodic readings to the utility. They can also receive requests from the utility, such as connecting/disconnecting loads or reading grid parameters, and reply accordingly.

The SMs are a critical component of the AMI due to their ability to perform tasks beyond metering electricity consumption. They facilitate bidirectional communication with utilities, allowing for dynamic electricity pricing, power outage alerts, power quality monitoring, and real-time access to data for consumers/prosumers. The data obtained from these SMs are valuable for effective and dynamic energy planning, leading to improved electric power system stability, efficiency, and reliability.

3.1.2 Communication Infrastructure

The communication infrastructure is critical in the AMI as it enables two-way communication between customers/prosumers and utilities for data transmission. Its main goal is to ensure secure and reliable data exchanges between SMs and the MDMS. The communication network comprises three key elements: Neighborhood Area Network (NAN), Data Concentrator Unit (DCU), and Wide Area Network (WAN) [63]:

- **NAN**. This element provides the infrastructure to connect SMs with the DCU. Typically, short-range communication technologies such as Power Line Communication (PLC)-Prime [64, 65], Radio Frequency (RF)/PLC [66–70], G3-PLC [64, 65], Wi-SUN [71], Wi-Fi [72], RF [66,73–75], Ethernet [76], among others, are used. The NAN is responsible for collecting energy data from SMs and transmitting them to the DCU. It is essential for enabling real-time data collection from SMs to utilities.

- **DCU**. This element is responsible for receiving and sending data to and from SMs through the NAN, as well as for receiving and sending data to and from the utility through the WAN. The DCU acts as a mediator between SMs and the utility, aggregating data from multiple SMs and sending them to the utility. It is equipped with hardware and software to store and manage the collected data and handle communication protocols between the NAN and WAN. If the NAN or WAN services are temporarily unavailable, the DCU stores the data in its database and forwards it to its destination when the services are restored.

- **WAN**. This element provides the medium for connecting the DCU to the utility. Typically, long-range communication technologies such as GPRS/3G/4G/5G [77,78], optic fibers [79], Broadband (BB)-PLC [80–83], and the Internet [84], among others, are used. The WAN is responsible for transmitting data from the DCU to the utility's server for further analysis and processing. It enables utilities to manage the collected data, perform real-time analysis, and use the information to optimize energy distribution.

The communication infrastructure is a crucial element in the AMI and must be carefully specified and designed to ensure reliable and secure information exchanges between SMs, the DCU, and the electric utilities. Using standardized communication protocols and security mechanisms such as encryption and authentication is mandatory to preclude unauthorized access to data and safeguard the privacy of consumers/prosumers.

3.1.3 Electric Utility

The electric utility side of the AMI is responsible for managing and processing data received from the consumer/prosumer side through the communication infrastructure. The MDMS is the core element at the electric utility side, and it plays a crucial role in data collection, storage, and analysis.

One of the primary functions of the MDMS is to store and manage large amounts of data from SMs and other devices in the AMI. This data is analyzed for various purposes such as billing, peak load management, grid parameters monitoring, and energy management. The MDMS also has the ability to remotely control and manage SMs by ordering their connecting/disconnecting and requesting power status verification. Moreover, it can acquire energy consumption data on-demand, enabling better energy management by the utility.

In addition to its real-time monitoring and control capabilities, the MDMS has long-term data storage, which enables it to provide valuable insights and perform demand forecasts. The data can also be used to identify patterns and trends in energy consumption, which can be used for energy planning and management. Furthermore, an MDMS typically offers an Application Programming Interface (API), which allows third-party applications and services to access and control SMs and other devices in the AMI. This API enables the development of new services and applications that can improve energy efficiency, reduce costs, and enhance the overall functionality of the AMI.

3.2 SECURITY ISSUES IN AMI

Ensuring the security, privacy, and reliability of AMIs is crucial for maintaining the stability of electric power systems. Vulnerabilities can exist in different components of the AMI, including the communication infrastructure, SMs, and data management systems. In Subsection **3.2.1**, we will discuss some of the most common vulnerabilities of AMIs, such as insufficient access control, lack of encryption, and vulnerabilities in the communication protocols. Later, Subsection **3.2.2** presents the main threats that AMIs face, including cyber attacks, physical attacks, natural disasters, and human errors. By understanding these threats, appropriate measures can be taken to mitigate them and ensure the security and reliability of electric power systems.

To address these vulnerabilities and faced threats, Subsection **3.2.3** delves into security considerations that should be considered in the design and implementation of AMIs. This includes strong authentication mechanisms, encryption, and secure communication protocols. Privacy is also a key concern in AMIs, as energy consumption data can reveal sensitive information about customers/prosumers. In the sequel, Subsection **3.2.4** discusses privacy aspects that can enable secure data exchange, such as data anonymization, data minimization, and consent-based data sharing.

3.2.1 Vulnerabilities

An AMI can be seen as a collection of interdependent systems (e.g., SM, sensors, gateways, among others). Each of these systems can present a vulnerability that an attacker can exploit. Moreover, an adversary can use multiple vulnerabilities to orchestrate a sophisticated attack on the AMI [85]. The following are some of the main vulnerable elements in the AMI:

- **Applications.** They manage metering data and value-added services, such as billing. Common vulnerabilities in software, such as flaws and misconfigurations, can be exploited by adversaries to deliver altered data to customers or utilities, resulting in cyber attacks [86].
- **Embedded hardware.** It is a significant source of vulnerabilities in the AMI. Hardware design flaws, firmware flaws, or outdated firmware can all serve as potential weak points for attackers, especially if they have physical access to the device [11, 12, 87]. An attacker who exploits these vulnerabilities can cause a blackout in a house or perform other malicious actions.
- **Data networks.** They provide the largest attack surface for the AMI. An attacker can exploit protocol flaws through devices like SMs, routers, and gateways to execute various attacks [13]. Without strong key management, such as symmetric or asymmetric keys, data networks lack authentication, confidentiality, and integrity, leaving them vulnerable to successful attacks.

3.2.2 Threats

The security threats faced by the AMI can be categorized into three main classes: threats to the system-level security, threats (or theft) to services, and threats to privacy. To address these threats, countermeasures have been identified as follows::

- **Threats to the system-level security.** These threats can compromise numerous customers simultaneously. For example, an adversary or a group of adversaries (e.g., a colluding group) may exploit embedded hardware flaws or flaws in the data network to access and alter the program instructions of SMs, changing their alarm threshold. The adversary may then send a command to all SMs (e.g., a disconnection command) impersonating the utility. If successful, this attack can cause all the affected SMs to shut off the power, leading to a blackout. The two main types of attacks in this threat are radio subversion (or takeover) and denial of service (DoS). They are explained below:
  - *Radio subversion or takeover.* In this type of attack, the adversary attempts to take control of the communication channel used by SMs. Once the attacker has control,

they can inject malicious firmware into the SM or send commands, potentially taking over buildings, houses, or facilities and causing a loss of availability and integrity. To mitigate this threat, it is necessary to use encrypted communication, such as the OpenSSL protocol, which establishes secure TLS-based communication between the SM and the electric utility [88]. Another option, proposed in [89], is a secure scheme based on ECC that authenticates control commands in the AMI. Although secure, this scheme requires high computational cost and data rate link due to the number of messages exchanged by the entities.

  - *Denial of Service (DoS).* This type of attack aims to weaken the data networks' performance or even interrupt data communication. The success of this attack can severely impact the services provided by the AMI, making them unavailable or with poor performance. DoS attacks can be carried out in several ways, including RF spectrum jamming, where a radio emitter device is used to interfere with the communication between SMs and the DCU, interrupting the data communication [90, 91]. Routing attacks are another efficient technique to execute a DoS attack. A compromised node in a multihop data communication can pose as the shortest path and mislead the SM data from its intended destination (e.g., a compromised node posed as the shortest path receives SM data to be retransmitted, but does not forward it) [91]. Jabbering is another DoS attack that uses a compromised SM to transmit so much traffic (i.e., flooding) that other SMs cannot communicate [13]. To mitigate the jamming attack, Premarathne and Atiquzzaman [92] designed a solution using a cognitive radio sensor network in AMI. Additionally, Lee *et al.* [93] developed a new mechanism to detect the misbehavior of neighboring nodes in a SM mesh data network. On the other hand, Hu and Gharavi [94] proposed a novel security approach based on authentication protocols to prevent DoS attacks.

- **Threats or theft via smart metering services.** This threat refers to an adversary compromising a SM to prevent utilities from collecting revenues. One technique involves intentionally sending incorrect energy consumption data, reducing the amount to be paid in the monthly bill [95]. The main theft techniques are cloning, compromise/intrusion, and location fraud, which are briefly discussed below:

  - *Cloning.* Usually, with the help of insiders (i.e., electric utility employees), an adversary can produce a copy of a SM, including its keys and IDs. In possession of the cloned SM, the attacker can impersonate the original SM to lower their own energy consumption report by changing the data message or even sending a zero usage report. Consequently, electric utilities might have significant revenue loss if many SMs are cloned [96]. A possible countermeasure is to perform firmware encryption to ensure that it is impossible for the attacker to tamper with it. Another countermeasure, although expensive, is based on RFID tags for ensuring authentication, as proposed in [96].

  - *Compromise/intrusion.* One possible way to perform energy theft is by disconnecting the communication module from the SM, making sending energy consumption reports to the electric utility impossible. This method can be easily detected by electric utilities due to the lack of reports, although report tampering might be much harder to identify. For instance, a perpetrator can send a zero usage report, or a mischievous customer can replay the last energy consumption usage report to the electric utility. Moreover, the customer can pose as a green energy producer, altering the report by increasing the units of fake production. A possible countermeasure is based on a cryptographic approach using a signcryption algorithm with the help of a trusted authority [97]. Other countermeasures are based on behavioral analysis of the compromised SM and adversaries [98, 99].
  - *Location fraud.* With the advent of dynamic pricing, an adversary can exploit flaws to change the SM location from a site that charges high energy prices to another site where it charges low energy prices [100]. A countermeasure for this kind of theft is a location-aware key management system [101], where the secret key is associated with the SM location coordinates. If a SM changes its location, it is possible to detect it by analyzing the secret key.

- **Threats to privacy in AMI.** As extensively discussed, privacy is paramount for AMIs. A message alteration or leakage can have fatal outcomes for the target customer, possibly revealing their identity and personal information to unauthorized entities. The main techniques that compromise customers' privacy are interception/eavesdropping and forwarding point compromise.

  - *Interception/eavesdropping.* Packet collection by an unauthorized person (i.e., an eavesdropper) could invade customers' privacy if the attacker manages to read the messages. For instance, energy consumption usage reports can be collected and analyzed to determine the customer's routine and even other personal information [102]. To mitigate eavesdropping, encryption and authentication approaches are adopted. Due to the importance of this topic, Chapter 4 will bring more details about it.
  - *Forwarding point compromise.* If a forward entity (e.g., a data concentrator) is compromised, it can forward SM messages toward an adversary. The adversaries can extract a significant amount of privacy information if they manage to read the message. Countermeasure is proposed in [103], in which a novel broadcast authentication protocol was described.

3.2.3 Security

Security is a critical aspect of AMI systems, given the sensitive information that flows through the heterogeneous data networks. Therefore, it is essential to consider the following security aspects from the earliest stages of AMI design [16, 104, 105]:

- **Availability.** The deployment of SM and AMI has enabled customers to generate and sell their own energy to utilities, providing greater control and flexibility. This new paradigm places high importance on the availability of SMs as they play a key role in determining dynamic energy pricing. The ability to remotely connect and disconnect customers and accurately bill them depends on the availability of SMs. However, it is important to note that availability is subjective and can vary depending on the application's purpose. For instance, alerts of anomalies detected in the grid require immediate reporting while pricing does not, but both rely on the availability of SMs. Therefore, ensuring the availability of SMs is one of the main aspects of a reliable and effective AMI.

- **Integrity.** In AMI, integrity is a critical security aspect, as it guarantees that data, commands, and alerts transmitted through data networks are generated by authorized entities and have not been modified during transmission. Lack of integrity can result in incorrect decisions by the systems' operators, such as mistaken alerts or wrong billing, which can have severe consequences for customers and electric utilities. Therefore, ensuring the integrity of AMI is essential for properly functioning electric power systems.

- **Confidentiality.** It is another vital security aspect of AMI. It is linked to customers' privacy, and the disclosure of consumption usage data can reveal their way of life and even indicate if someone is at home. This information can be used for nefarious purposes such as burglary, espionage, or blackmailing. Hence, it is crucial to ensure that consumption usage data is confidential and only accessible to authorized entities. The confidentiality aspect in AMI is ensured by encrypting the data while transmitting through data networks and ensuring that only authorized entities can access it.

- **Authentication.** In the context of AMI, authentication is crucial to prevent unauthorized access to the system and protect sensitive information. Authentication methods can include passwords, digital certificates, biometric identifiers, or other forms of identity verification. Note that the strength of the authentication mechanism depends upon its security level. A weak authentication mechanism can be easily bypassed by attackers, compromising the entire system's security. Therefore, designing and implementing a robust authentication mechanism is mandatory for ensuring the security of AMIs. Moreover, continuous monitoring of authentication mechanisms and updates to strengthen them helps to prevent potential attacks.

- **Non-repudiation.** It is an essential aspect of AMI security since it provides evidence of the origin and authenticity of data, commands, and alerts exchanged within the system.

Non-repudiation ensures that the parties involved in information exchanges cannot deny their participation or the authenticity of their messages. This means that once a message is sent, it cannot be denied by the sender or the receiver, providing accountability in case of disputes or legal issues. Non-repudiation is often implemented using digital signatures, which provide a unique and verifiable signature for each message. Therefore, non-repudiation is a critical security aspect of AMI that ensures the authenticity and accountability of the exchanged information.

- **Access control.** It is critical to AMI security, managing access to sensitive data with procedures and rules for granting or denying access. It can monitor and audit user activities to detect unauthorized or suspicious access. However, balancing security and utility is challenging since utilities require access to some data for efficient grid management. Access control policies must be carefully designed, regularly audited for compliance, and supplemented with encryption to protect sensitive data during transit and at rest. Careful management of encryption is necessary to avoid a negative impact on system performance or usability.

- **Auditing.** Periodic auditing is crucial in AMI to detect potential security breaches, assess security controls, and ensure regulatory compliance. Regular audits of system logs, access controls, and other security measures provide a comprehensive view of the security posture. It also helps identify areas for improvement, patterns of suspicious behavior, and unauthorized access. Moreover, auditing is essential in establishing trust with customers and other stakeholders.

3.2.4 Privacy

A significant amount of data generated by SMs is transmitted through the AMI, and the volume of data is increasing steadily. Electric utilities previously collected readings on a monthly basis per meter, but with the implementation of AMI, the frequency of data collection has increased. For instance, collecting a reading every 15 minutes generates over $8,000$ readings monthly per consumer/prosumer. This increase in data volume can help electric utilities operate more dynamically and efficiently. However, collecting personal information through SM data raises concerns about privacy. This data can reveal detailed information about individual behaviors, activities, and time patterns, potentially exposing sensitive information about customers. For example, data can reveal the number of occupants in a building, allowing one to determine whether the property is empty or not.

To address these privacy concerns, several approaches have been proposed in the literature [106–109]. These approaches enable the collection of energy consumption usage data while maintaining customer privacy and usefulness for practical applications such as billing and demand-supply management. Short descriptions of them are as follows:

- **Anonymity.** It refers to the inability to identify which entity sent a given data in an AMI. This is a crucial privacy target, as it prevents adversaries from determining the source of sensitive information even if they can collect and interpret the data.
- **Unlinkability.** It is the characteristic that prevents an adversary from establishing a relationship between two entities in the system. This privacy feature ensures that observing the system does not reveal any additional information about the entities if they are unlinked.
- **Undetectability.** In an AMI with many entities, undetectability is an essential feature that prevents adversaries from determining whether an item (e.g., a SM or a message) exists or not.
- **Unobservability.** It refers to a system's inability to be observed by an outside observer. In an AMI, this feature prevents an adversary from detecting whether an entity is transmitting or not, such as whether a SM is sending an energy consumption message or in an idle state.
- **Pseudonymity.** To enhance privacy, each SM in AMI should have several identifiers or pseudonyms to avoid easy recognition of energy consumption data by unauthorized entities. This feature ensures that only the entities communicating with each other can determine the other party's identity, enhancing the privacy of customer data.

3.3 SUMMARY

This chapter has presented an overview regarding AMIs focusing mainly on its security issues. To do so, firstly, the key elements of the AMI – i.e, consumers/prosumers, where the SMs are placed; communication infrastructure, composed by NAN, DCU, and WAN; and electric utility, where the MDMS is deployed – are presented and detailed. Regarding AMI security issues, it can be divided into four main areas: vulnerabilities, security considerations, privacy aspects, and threats faced by AMIs. Vulnerabilities in AMI can be exploited when applications are flaw or misconfigured, hardware are not securely deployed, and/or data networks present breaches. Security in AMI is another concept that must be considered in early stages of AMIs where availability, integrity, confidentiality, authentication, non-repudiation, access control, and auditing are its vital aspects. Privacy aspect is another essential feature in AMI aiming to unable an eavesdropper to extract any sensitive information. To do so, a system which guarantee privacy must offer anonymity, unlinkability, undetectability, unobservability, and pseudonymity. Finally, threats faced by AMIs are further divided into threats to the system-level security, where a numerous customers are simultaneously compromised; threats of theft via smart metering services, where an adversary compromises SMs preventing revenue collection by electric utilities; and threats to privacy in AMI, where customer's privacy is compromised.

4 POST-QUANTUM CRYPTOGRAPHY

Nowadays, cryptography is present in many aspects of our daily life, often unnoticed. For example, wire transfers and credit card payments rely on cryptography to ensure that no attacker corrupts the transaction, providing users with security and reliability. Safe data storage is another useful application, as logins, passwords, and personal information are protected using cryptography when stored in databases. Another recent application is a cryptocurrency, which uses asymmetric cryptography, symmetric cryptography, and hashing. Asymmetric cryptography is used as a digital signature to allow secure, anonymous, and non-repudiable transactions, meaning that there is no longer a need for a central player (e.g., a bank) and it is not necessary to know anything about anyone in the chain. Furthermore, asymmetric cryptography is also required to perform key exchange between parties for further use as a symmetric key. In turn, symmetric cryptography is used to encrypt/decrypt sensitive messages. Finally, hashing is used as proof of work to verify the accuracy of new transactions that are added to the blockchain as it is a decentralized system.

In addition to the aforementioned applications, the most common use of cryptography is in communication systems where parties require secure communication protected against eavesdroppers or interference by a third party. In this scenario, asymmetric and symmetric cryptography play an important part, enabling secure key exchange between parties, authentication, and encryption/decryption of messages. Although applications secured by cryptography might be at low risk, the significant advances in the accomplishment of quantum computers with the sponsorship of major technology companies put classical cryptography at risk, as already developed quantum algorithms can easily break it. This concern has increased even more recently as the quantum computer achieved its first quantum supremacy in a particular application, meaning that a quantum computer solved a problem that a classical computer cannot solve in a reasonable timeframe.

In this sense, with the advent of a quantum computer in the near future, all systems that rely on classical cryptography are potentially insecure. This means that wire transfers and credit cards will no longer be secure, stored logins and passwords might be compromised, and encrypted communications might not be private anymore, among other applications, leading to complete global chaos and instability. To anticipate this issue, in 2016, NIST initiated a contest to evaluate and standardize one or more quantum-resistant asymmetric cryptographic schemes, called NIST PQC standardization process. In 2022, the first four schemes were standardized, one for KEM and three for digital signature. The standardization process is not yet concluded, as the NIST intends to standardize more quantum-resistant asymmetric cryptographic schemes. Aiming to advance in this discussion, this chapter has the following contributions:

- A concise review regarding the background and future trends of quantum computation and how it threatens classical cryptographic schemes. To do so, the basics of quantum physics are presented; a comparison between classical and quantum computers is discussed;

and, finally, what challenges must be overcome to turn quantum computers scalable and powerful.

- A discussion of cryptographic fundamentals which will be used throughout this dissertation. In this sense, asymmetric cryptography is discussed highlighting the KEM. Symmetric cryptography is also presented by approaching block and stream ciphers. A brief discussion on how asymmetric and symmetric cryptography interact with each other is also provided.
- A complete scrutiny of the NIST PQC standardization process, presenting the classes presented in the contest and in what principle they rely on. A deep analysis regarding lattice-based cryptography is provided, taking a deeper look into FrodoKEM and CRYSTALS-Kyber schemes.

Based on this discussion, this chapter aims to present the background and future trends of quantum computation in Section 4.1, discuss the cryptographic fundamentals which allow data security nowadays in Section 4.2, and finally, approach the NIST candidates, their classes schemes, and their differences aiming for security in a quantum era in Section 4.3.

4.1 QUANTUM COMPUTER

In 1927, Werner Heisenberg introduced the uncertainty principle, which asserts that it is impossible to know everything about a quantum particle simultaneously. The more precisely its momentum is determined, the less precisely its position can be predicted, and vice versa. Later, Earle Hesse Kennard and Hermann Weyl derived the inequality that relates the standard deviation of position σ_x and the standard deviation of momentum σ_p:

$$\sigma_x \sigma_p \geqslant \frac{\hbar}{2}, \tag{4.1}$$

where $\hbar$ is the reduced Plank constant.

However, it was only in 1962 that the phrase "quantum information theory" was coined [110], and the suggestion that entanglement might be used as a communication resource was introduced. In 1981, the theoretical physicist Richard Feynman challenged a group of computer scientists to develop a new breed of computers based on quantum physics. This challenge has yet to be fully realized, but significant progress has been made in recent years.

According to the literature, it was not until 1994 that a large number of interesting theoretical and experimental results appeared in the field of quantum computing. That year, Peter Shor demonstrated that it is possible to efficiently factor large numbers into their primes and solve the discrete logarithm problem using a quantum computer running a specific algorithm (later known as Shor's algorithm) [21]. This result significantly impacted the cryptography community, as many cryptographic algorithms rely on the hardness of integer factorization or discrete logarithm problem. Due to its importance, Shor's algorithm will be discussed in Section 4.3.

Three years later, in 1998, Isaac Chuang and his team built the first quantum computer with only 2-qubits at the IBM Almaden Research Center [111]. Later, in 2007, the startup D-Wave revealed its 28-qubit quantum annealer processor[1] [112], and in 2011, this company presented the D-Wave One, the first commercially available quantum annealer processor [113]. Only four years later, in 2015, D-Wave announced that the 1000-qubit barrier was broken [114] and in 2017 the D-Wave 2000Q, a 2000-qubits quantum annealer processor was presented [115].

Yet in 2017, IBM unveils a 17-qubit quantum processor [116] and, in 2018, Google and Intel presented a 72-qubit and 49-qubit quantum processor [117, 118], respectively. In 2021, IBM presented the IBM Eagle [119], a 127-qubit quantum processor and, in 2022, the successor of IBM Eagle, called Osprey [120], the most powerful quantum processor known with 433-qubits. Note that the quantum processors developed by IBM, Intel, and Google aims universal applications, therefore they are capable of execute the Shor's algorithm. In the same year, D-Wave presented its end-to-end quantum platform based on the Advantage quantum annealer processor with more than 5000-qubits [121].

However, despite these advances, quantum computers are not yet as reliable as state-of-the-art classical computers. It is essential to emphasize that even though the principles of quantum computing have been proven to work, quantum computers are not yet powerful enough to break 2048-bit RSA keys, and it is unlikely that they will be in last than 10 years.

4.1.1 Quantum physics principles

To facilitate the understanding of the basic principles of a quantum computer, it is essential to address a few important principles of quantum physics, namely superposition and entanglement. Each of them is briefly detailed as follows:

- **Superposition.** Much like waves in classical physics, quantum states can be added together and superposed to yield a new valid quantum state. Conversely, every quantum state can be seen as a linear combination of other distinct quantum states. The double-slit experiment visually demonstrates the superposition phenomenon by shooting quantum particles at two narrow slits. A classical particle will always pass through the top or bottom slit, but a quantum particle can be put in a superposition of two paths: one through the top slit and the other through the bottom slit. This experiment leaves a pattern that shows that the quantum particle also behaves as a wave, a concept called wave-particle duality. The superposition state is the characteristic that allows the so-called quantum parallelism.

- **Entanglement.** Entanglement is a special connection between particles. A pair or group of particles is said to be entangled when it is not possible to describe the quantum state of each particle independently from the quantum states of the other particle(s). The whole quantum

[1] Quantum annealer is a branch of quantum computing that approach a restricted set of problems, commonly optimization problems. Therefore, they are not universal quantum computers and are not capable of run the Shor's algorithm, for instance.

state of the system can be described, but each part of the system cannot. Entanglement can be created by bringing two particles close together, performing an operation to entangle them, and then moving them apart again. If two particles are entangled, they will remain entangled even if they are separated by large distances. The entanglement will manifest in the outcome of measurements on these particles, which will be correlated regardless of the distance between them.

Understanding the aforementioned principles, the qubit, which is the foundation of a quantum computer, can be explained. It is well-known that an electron has a spin, either up or down (i.e., $|1\rangle$ or $|0\rangle$, respectively, using Dirac notation), with both choices being equally probable. Until measured, the electron is in a superposition of the two states $|1\rangle$ and $|0\rangle$, known as a single qubit. A magnetic field can be used to initialize a qubit to either spin up or down, which will maintain for a short period of time until outside influences degrade it. This short period of time is called coherence time.

For example, some properties of a hydrogen molecule with two electrons can be represented by a system with two qubits, with the possibility of measuring four different states ($|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$). If the electrons are not entangled, before the measurement, the system is in a superposition of all four states equally likely. If two qubits are entangled, for instance, in state $(|00\rangle + |11\rangle)/\sqrt{2}$, if the measurement of the first qubit is $|0\rangle$, then the probability of measurement of the second qubit is $|0\rangle$. This manipulation enables quantum logic, where input qubits control the behavior of output qubits.

4.1.2 Classical vs quantum computers

Classical and quantum computers differ greatly in how they are built, what they rely on, how they approach a problem, and how they are programmed. Classical computers have been a part of our everyday lives for a long time, and technology continues to evolve, with smaller, more powerful, and more efficient architectures. However, the quantum computer is still in its infancy and presents many challenges that have yet to be solved, as described in Subsection **4.1.3**.

Classical computers are based on bits, which can only assume two values: 0 or 1. In contrast, quantum computers are based on qubits, which can assume either $|0\rangle$ or $|1\rangle$. Furthermore, a qubit can be in a coherent superposition of both states simultaneously. Additionally, while measuring a bit would not disturb its state, measuring a qubit would destroy its coherence and break the superposition state.

Another significant difference between classical and quantum computers is how they approach a problem. For instance, consider an optimization problem. A classical computer using a brute force algorithm would calculate every possible outcome one at a time, eventually finding the best solution. On the other hand, a quantum computer with enough qubits can solve the problem at once because of superposition. A classical 2-bit computer has four possible states (00, 01, 10, and 11), while a quantum 2-qubit computer has four possible states ($|00\rangle$, $|01\rangle$,

$|10\rangle$, and $|11\rangle$). However, due to superposition, it can represent all four states simultaneously. Mathematically, an n-qubit quantum computer can simultaneously represent 2^n states. This explains why quantum computers are deemed to be astonishingly more efficient than classical computers in solving certain classes of problems. For instance, Grover's algorithm [22] explores this characteristic to perform efficient searches.

The programming of classical and quantum computers is also distinct. Classical computers rely on logic gates such as AND, OR, and NOT, which output deterministic values (true or false). In contrast, quantum computers are based on reversible quantum gates. Each applied constraint makes some states more likely than others. The states in a quantum computer are unknown until a measurement is performed, breaking the superposition state.

4.1.3 The challenges of quantum computer

An operable and reliable quantum computer must fulfill the following constraints:

- **Stability.** The quantum state must have a long coherence time to perform calculations.
- **Scalability.** It must be possible to create a large number of identical qubits, enabling interaction over distances while preserving the superposition of the quantum state.
- **Manipulation.** It must be possible to set an initial state of the qubits at the beginning of the computation and to measure the result at the end.

The requirements for qubits, particularly stability and scalability, are challenging and tend to be in conflict with each other. For example, maintaining stability in large systems can be difficult. An isolated system is essential for enhancing qubit stability by shielding it as much as possible from disturbances. Thermal vibrations are detrimental, and quantum computers must operate at very low temperatures, near zero Kelvin, to provide a feasible coherence time.

A few chemical structures provide the required characteristics to be a qubit candidate. One such structure of interest is the nitrogen-vacancy center in diamonds. In the diamond structure, each carbon atom is bound to its four neighbors with covalent bonds. Replacing one carbon atom with a nitrogen atom creates a new covalent bond that is slightly weaker than the surrounding carbon-carbon bonds. If a second carbon is removed, a vacancy is created, resulting in an Nitrogen-Vacant (N-V) center with an extra electron trapped in a rigid carbon lattice. More details about the N-V center are beyond the scope of this dissertation, but it is important to mention that this structure has complex optical and electrical behavior. For instance, the appropriate use of a certain wavelength of light can be used to initialize and measure the spin of the associated electron.

All the aforementioned characteristics make N-V centers a promising technique as a qubit technology. A roadmap focused on diamond-based quantum computers is gaining momentum among researchers; however, there are challenges to overcome. One of them is the ability to construct an array of identical qubits, as only a small fraction of implanted nitrogen atoms lead to the creation of an N-V center, which impacts the scalability of the quantum computer.

Furthermore, light propagation between qubits requires waveguides and other photonic structures. Diamond is one of the most inert structures known, and this characteristic makes instrumentation difficult. Despite all the challenges that the N-V center-based qubit presents, the literature shows that it remains one of the best choices. Overall, the implementation of quantum computing is still complex and challenging.

4.2 CRYPTOGRAPHIC FUNDAMENTALS

Cryptographic schemes are the most used techniques to protect sensitive data against adversaries. In an AMI, an infrastructure with multiple entities which require constant messages exchange, the attack surface is vast. The main countermeasures against these adversaries are strong encryption and authentication schemes. In this regard, this section seeks to present classical cryptographic fundamentals useful for AMIs, such as symmetric and asymmetric cryptographic schemes.

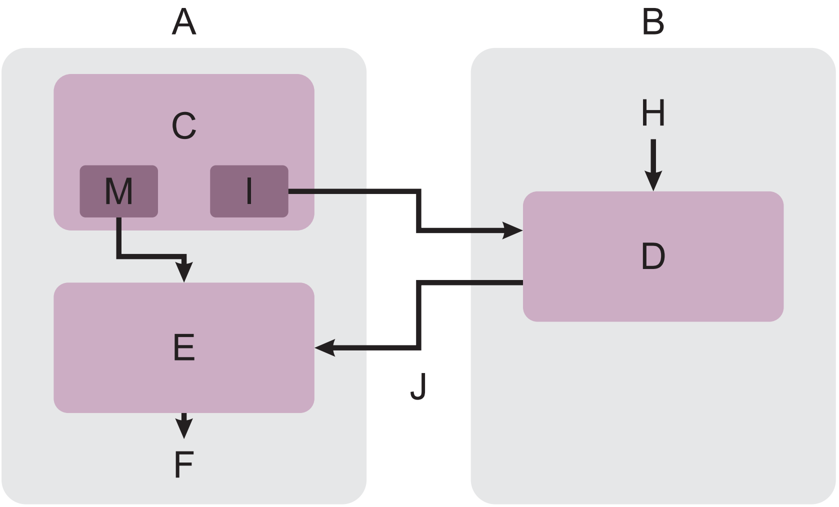
4.2.1 Asymmetric cryptography

Asymmetric cryptography, also known as public-key cryptography, relies on a pair of keys – i.e., public and private key. A public key is a key that can be distributed to interested parties, while private keys (also known as secret keys) can never be shared and must be stored securely by each part. In this research field, there are numerous techniques to provide message security and key agreement. Three of them are PKE, KEM, and Key Exchange (KEX).

- **PKE.** It aims to provide encrypted communication between parties. The PKE can be divided into three steps: (i) key pair generation, where Alice generates a public key pk_a and private key sk_a and shares the latter with Bob. This step is required to be performed only once; (ii) encryption, in which Bob uses Alice's public key pk_a to encrypt a message; and (iii) decryption, where Alice received the encrypted message and uses its own private key sk_a to decrypt it. The PKE block diagram is presented in Figure 4 where Bob sends a message to Alice.

- **KEM.** It aims to share between parties (e.g., Alice and Bob) a shared secret key *ss* – also known as an ephemeral key – usually a key for symmetric cryptography. The KEM is divided into three-step: (i) key pair generation, where Alice generates public key pk_a and private key sk_a and shares the former with Bob; (ii) encapsulation, in which Bob uses Alice's public key pk_a to generate a shared secret key *ss* and returns an encapsulated key to be sent do Alice as a ciphertext; and (iii) decapsulation, where Alice uses its private key sk_a to decapsulate the shared secret key *ss*, achieving the same one as Bob. This process

Figure 4 – Block diagram illustrating the PKE. Note that pk_a, and sk_a stand for a Alice's public key and Alice's private key, respectively.

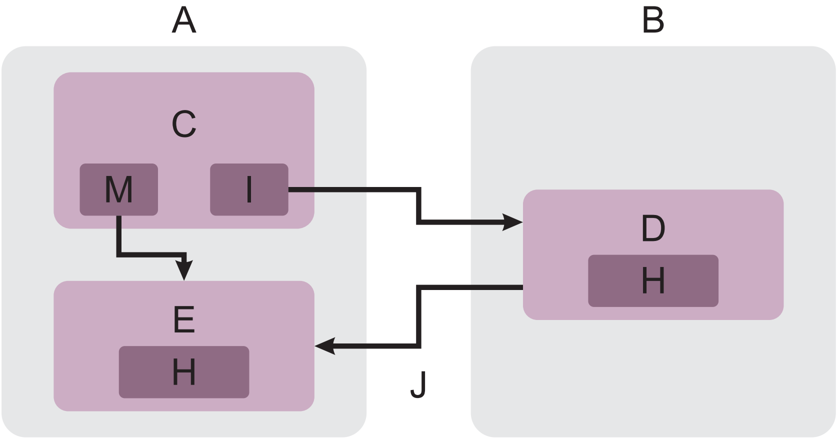


Source: Personal collection.

must be repeated occasionally to renew the shared secret key ss, raising security. Figure 5 depicts the KEM.

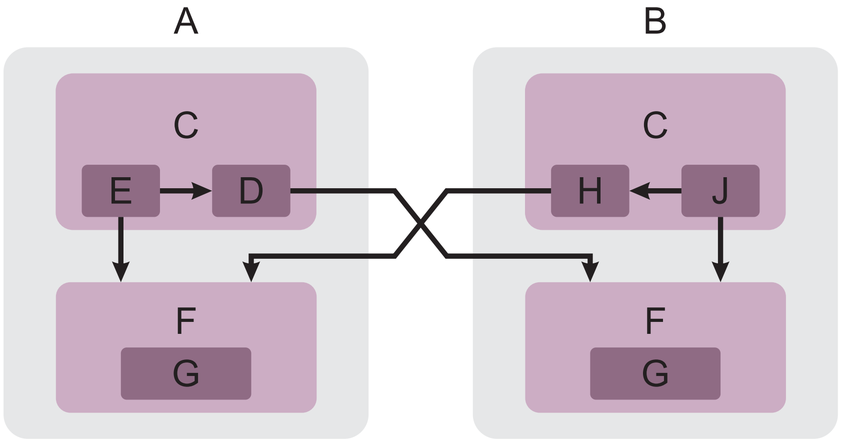
- **KEX**. As the KEM, it is also used for key agreement, but the KEX requires both Alice and Bob to generate their own private key sk_a and sk_b, respectively. Alice executes a mathematical operation using its private key sk_a to generate its public key pk_a, while Bob does the same generating its public key pk_b using its private key sk_b. Both public keys pk_a and pk_b are exchanged and a common shared secret key ss can be achieved by performing another mathematical operation using the other's public key. The KEX block diagram is shown in Figure 6.

Figure 5 – Block diagram illustrating the KEM. Note that pk_a, sk_a, and ss stand for a Alice's public key, Alice's private key, and shared secret, respectively.



Source: Personal collection.

Figure 6 – Block diagram illustrating the KEX. Note that pk_a, sk_a, and *ss* stand for a Alice's public key, Alice's private key, and shared secret, respectively.



Source: Personal collection.

Note that to perform PKE, KEM, and KEX, it is required to rely on a cryptographic scheme, which performs mathematical operations to generate key pair, perform encryption/decryption or encapsulation/decapsulation. Two popular examples that are usually used for these purposes are RSA and ECC. The RSA relies on the hardness of factoring the product of two large prime numbers. On the other hand, ECC relies on the assumption that finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point is infeasible. Both algorithms are relatively slow compared to symmetric cryptography, so they are not often used for encryption/decryption (i.e., as PKE). Instead, they are used to exchange a shared secret key (i.e., as KEM or KEX), which is then used for encryption/decryption.

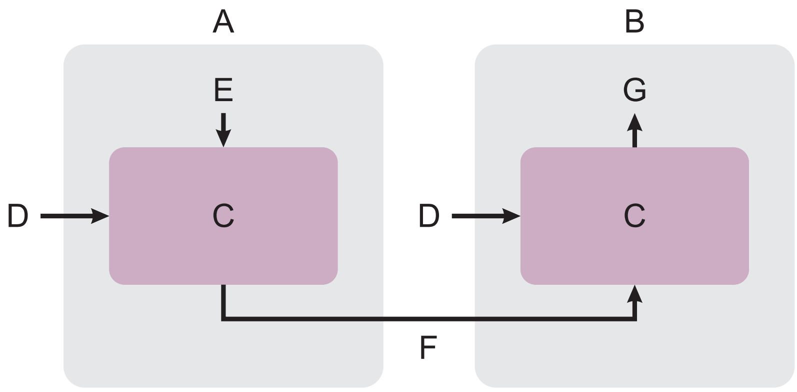
Regarding KEM and KEX, it can be seen in Figure 6 that the KEX requires both parties to generate public and private keys. On the other hand, KEM demands that only one party performs this step, as illustrated in Figure 5. In this sense, KEM is usually preferable, especially when one party of the key agreement relies on hardware-constrained equipment, leaving the key pair generation to the pair that has more processing power.

4.2.2 Symmetric cryptography

Symmetric cryptography relies on the same key (i.e., a shared secret key) for encryption and decryption. The shared secret key can be obtained using asymmetric cryptography, as described in Subsection **4.2.1**. In this sense, Alice and Bob can use symmetric cryptography to encrypt/decrypt sensitive data using the same shared secret agreed previously (i.e, performing KEM or KEX). This class of cryptography consists of two main categories: block ciphers and stream ciphers.

- **Block ciphers.** It encrypts/decrypts data by taking blocks of n-bits at a time. For encryption, the block cipher receives the shared secret key *ss* and n-bits of the plaintext and outputs n-bits of the ciphertext. In the decryption, the process is similar but the shared secret key *ss*

Figure 7 – Simplified block diagram of block ciphers. Note that *ss* stand for shared secret.



Source: Personal collection.

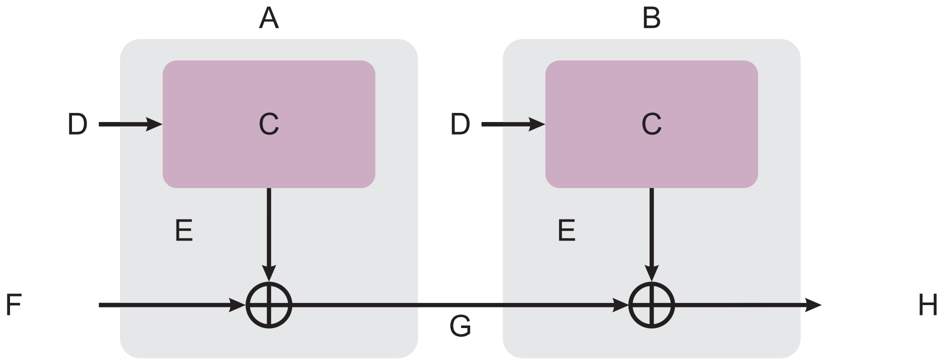
and *n*-bits of the ciphertext are the inputs, while the output is *n*-bits of the recovered plaintext. Block ciphers offer high diffusion since a single plaintext block can be used in multiple encryption iterations, although it can cause a high error propagation rate. Block ciphers have two common operation modes: (i) confidentiality-only, which focused on keeping the communication private (e.g., Counter (CTR) mode); and (ii) Authenticated Encryption with Associated Data (AEAD), which ensures data authentication and confidentiality (e.g., GCM mode). AES, TwoFish, and Camellia are very popular block ciphers. Figure 7 depicts a simplified block diagram for the block cipher.

- **Stream ciphers.** It encrypts/decrypts data by taking one byte at a time. More specifically, using the shared secret key *ss*, the stream cipher generates a keystream, which is *xored* with the plaintext, generating the ciphertext. To reverse, the same process is repeated but *xoring* the keystream with the ciphertext, recovering the plaintext. This process is repeated until all plaintext is encrypted and all the ciphertext is decrypted. Stream ciphers are simple to implement and present good performance. Although, they lack diffusion because each plaintext digit is mapped to one ciphertext output. Salsa20 and ChaCha20 are relevant examples of stream ciphers. They can also be combined with AEAD for data authentication and confidentiality. Figure 8 shows a simplified block diagram for the stream cipher.

Nowadays, the most popular symmetric cryptographic schemes are AES and Chacha20. The AES is a block cipher of 128-bits, with three possible key sizes: 128, 192, and 256-bits. It can be combined with the GCM (AES-GCM), which is a stream cipher mode for AES[2]. Also, the AES is specified in NIST Special Publication 800-38D [122] for providing authenticated encryption (confidentiality and authentication). The Chacha20, specified in RFC 7539 [123], on the other hand, is a stream cipher that supports only a 256-bits key. Based also on the RFC 7539, the Chacha20 can be combined with the Poly1305 (Chacha20-Poly1305) for also providing

[2] Besides the AES is a block cipher, when combined with the GCM it is transformed in a stream cipher mode.

Figure 8 – Simplified block diagram of stream ciphers. Note that *ss* stands for the shared secret.



Source: Personal collection.

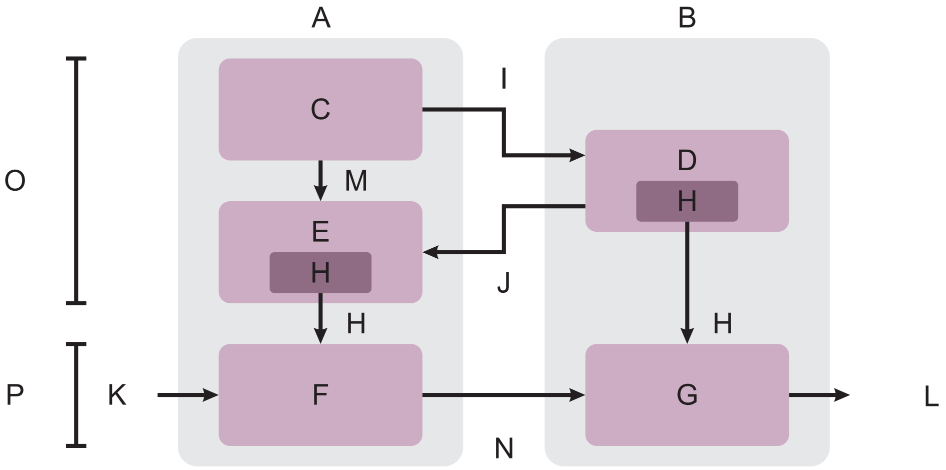
authenticated encryption. The Chacha20 is by its construction time constant. Comparing AES and Chacha20, many processors make intrinsic implementations of the AES available, resulting in better performance than Chacha20. However, Chacha20 is faster than AES in pure-software implementation – i.e., when intrinsic implementation of AES is not available. Overall, both schemes are secure, as far as it is known, lightweight, and perform well.

4.2.3 Asymmetric and symmetric schemes working together

This section shows the relationship of two important cryptographic concepts: (i) the secure key agreement between parties (i.e., KEM) and (ii) the encryption/decryption of sensitive data. They are performed by asymmetric cryptography and symmetric cryptography, respectively. Figure 9 illustrates these two concepts to provide secure communication between parties, which are explained in the following.

The upper part of Figure 9 illustrates the KEM, composed of key pair generation,

Figure 9 – Block diagram illustrating the key agreement between parties using KEM and the encryption/decryption process. Note that pk_a, sk_a, and *ss* stand for a Alice's public key, Alice's private key, and shared secret, respectively.



Source: Personal collection.

encapsulation, and decapsulation. One of the parties, in this case, Alice, starts the process of generating a key pair composed of a public key pk_a and a private key sk_a. Alice safely stores the sk_a and sends a public key certificate of pk_a to Bob, whom randomly generates a shared secret key *ss* and encapsulates it using Alice's pk_a producing a ciphertext. The ciphertext is sent to Alice, which uses its stored sk_a to decapsulate the ciphertext, revealing the *ss* generated previously by Bob. After these three steps, both parties hold the same *ss*, which can be used as a symmetric key.

After the key agreement between parties, the symmetric cryptography, depicted in the lower part of Figure 9, can be used to securely transfer data from Alice to Bob and vice-versa. When data is required to be transmitted by one party, it uses the *ss* to encrypt data and authenticate it using a symmetric cryptographic scheme with AEAD. The ciphered data is sent to the other party, which uses the same *ss* to decrypt the data and verify it, guaranteeing that the other party actually sent the data. Besides the *ss*, other parameters are required by the symmetric cryptography, but they are public – e.g., nonce, Additional Authenticated Data (AAD), tag – and due to that were omitted in Figure 9.

4.3 POST-QUANTUM CRYPTOGRAPHIC SCHEMES

A quantum computer can potentially outperform a classical computer in some tasks, which will consequently be exploited to remarkably enhance some areas that heavily depend upon intensive computation such as artificial intelligence, molecular modeling, financial modeling, weather forecasting, particle physics, and cryptography, among others. Concerning the cryptography area, Shor's Algorithm [21] is going to be a severe threat because it has the capacity to find prime factors of an integer in polynomial-time $O((\log N)^3)$, where N is the number to be factorized[3]. Therefore, if one cannot rely anymore on the hardness of integer factoring assumption or computing discrete logarithm, the asymmetric cryptography based on large primes will no longer be secure because an adversary will be allowed to easily factor large numbers or solve the discrete logarithm problem, which severely compromises popular schemes such as RSA and ECC. Consequently, the PKE, KEM, and KEX based on these schemes will no longer be secure, compromising the security of communication.

Note that Shor's algorithm will only be feasible for large numbers when a quantum computer with a sufficient number of qubits could operate with very small error – i.e., without succumbing to quantum noise and other quantum decoherence phenomena. It is estimated that to break 2048-bits RSA would take at least one million qubits demanding a time interval longer than 8 hours. Nowadays, there are very few quantum computers with more than a hundred qubits, such as the IBM's Osprey chip with only 433 qubits. Researchers estimate that it will take more than one decade, maybe two, to achieve the capability to break a 2048-bits RSA.

[3] The most efficient known classical factoring algorithm runs in a sub-exponential time $O(e^{1.9(\log N)^{1/3}(\log\log N)^{2/3}})$. Shor's algorithm is almost exponentially faster than the fastest classical algorithm.

Since introducing Shor's algorithm, the scientific community has sought solutions to secure communications. Two main approaches against powerful quantum computers are known as QKD and PQC. QKD is a hardware-based approach that uses the principle of quantum mechanics to detect eavesdroppers. As mentioned before, in quantum mechanics, the act of observing states has an effect on the observed state. Consequently, Alice and Bob know that the key is secure because an interception will disturb the key, leading to transmission errors, which can be detected by a legitimate user. This approach needs only a few algorithm replacements or modifications, although it requires an entirely specialized hardware infrastructure (i.e., quantum enable devices and quantum repeaters) because QKD only works with optical communication (i.e., optical fiber or free space optical). Consequently, it is a very expensive solution due to its costly infrastructure. QKD also struggles to communicate over long distances, requiring the use of quantum repeaters. As an alternative, the use of satellites is being investigated using line-of-sight to overcome the distance issue.

On the other hand, the PQC is a more feasible approach because it is an algorithm-based solution. In this sense, it can use most of the hardware and data communication infrastructure that exists today without any distance issues. PQC algorithms will act as current PKE, KEM, KEX, or digital signature, although instead of being based on the hardness of factoring large numbers or computing discrete logarithms, which is easily solved by Shor's algorithm, it will be based on algorithms that are theoretically hard for quantum (and classical) computers to calculate.

4.3.1 NIST algorithms classes

With all the aforementioned concerns, the NIST, since 2016, has been holding a PQC process to standardize a quantum-resistant algorithm. For the first round, NIST PQC standardization process received 69 submissions divided into two applications: PKE/KEM and digital signature. These 69 submitted algorithms can also be divided into six classes: lattice-based, multivariate, hash-based, code-based, supersingular elliptic curve isogeny, and symmetric key quantum resistance [124]. A short discussion about them is as follows:

- **Lattice-based.** It is the general term for primitives that use lattices in their construction or in security proof. Generally, they are based on problems such as LWE, Ring-Learning With Errors (R-LWE), Module-Learning With Errors (M-LWE), Module-Learning With Rounding (M-LWR), N-th degree Truncated polynomial Ring Units (NTRU), among others. These schemes have been extensively studied and no successful feasible attacks have been found so far. Studies suggest that NTRU algorithms have more secure properties than other lattice-based algorithms. Lattice-based NIST candidates are the majority with more than 40% of the submissions. FrodoKEM (LWE-based), NTRU Prime (NTRU-based), CRYSTALS-Kyber (M-LWE-based), NTRU (NTRU-based), and Saber (M-LWR-based) are PKE/KEM candidates for Round #3 in the NIST PQC standardization process in the class of lattice-based cryptography. The first two remained as alternate candidates, while the last three were finalists with the CRYSTALS-Kyber being the winners of the contest. In digital signature application, CRYSTALS-Dilithium (M-LWE-based) and Falcon (NTRU-based) are lattice-based standardized schemes in this class.

- **Multivariate.** It is a cryptographic system based on the difficulty of solving systems of multivariate equations. Previous attempts to build a secure system based on multivariate have failed, although, Rainbow is a finalist of the NIST PQC standardization process and Great Multivariate Short Signature (GeMSS) remained as an alternate candidate, both in the digital signature application.

- **Hash-based.** A Hash signature-based cryptographic system is one of the oldest signature methods and has been mostly treated as one alternative for a number-theoretic digital signature like RSA. Due to the limitation that a hash-based public key can only sign a limited number of signatures using the corresponding set of private keys, there was reduced interest in the hash-based signature schemes. Recently it has been revived due to its resistance to the attack of quantum computers. In the NIST PQC standardization process, the SPHINCS+, a hash-based scheme for digital signature application, remains an alternate candidate.

- **Code-based.** This cryptographic system relies on error-correcting codes, such as the McEliece encryption algorithm, which has been used for a long date for signatures using random Goppa codes without being broken. Variants of the McEliece encryption algorithm, which added structures to reduce the size of keys, have been shown to be insecure. The Classic McEliece, Hamming Quasi-Cyclic (HQC), and HQC encryption algorithms were in Round #4 of the NIST PQC standardization process in the PKE/KEM application group.

- **Supersingular elliptic curve isogeny.** This cryptographic system relies on supersingular elliptic curves and supersingular isogeny graphs. The combinations of them result in a Diffie-Hellman (DH) replacement with forwarding secrecy, which can serve as a quantum-resistant replacement for DH and Elliptic Curve Diffie-Hellman (ECDH) key exchange, a widely used method nowadays. In Round #4 of NIST PQC standardization process, SIKE was evaluated in the PKE/KEM application group. Supersingular elliptic curve isogeny cryptographic system has no finalist in digital signature applications. Recently, Castryck and Decru [125] present an isogeny cryptanalysis that breaks the SIKE in a very short time interval.

- **Zero-knowledge proof.** It is a method by which one party (the prover) must prove to another party (the verifier) that they possess the knowledge of a value, but without revealing it. The challenge relies on how to prove such possession without revealing the information itself. Regarding the NIST PQC standardization process, the Picnic scheme is an alternate candidate in the digital signature application. However, a zero-knowledge proof cryptographic scheme has no finalists either in PKE/KEM or in digital signature applications.

4.3.2 Lattice-based cryptography

Lattice-based cryptography is a very promising construction for providing data security and privacy in the quantum era. This trend was confirmed in the NIST PQC standardization process, in which the great majority of finalist candidates were lattice-based and the final process resulted in three out of four standardized schemes being Lattice-based. As this dissertation focused on lattice-based asymmetric cryptography to provide security and privacy against attacks from adversaries in possession of quantum computers, this section provides a discussion about it. In this sense, Subsection *4.3.2.1* presents the notation used throughout this dissertation, while Subsection *4.3.2.2* introduces the LWE problem and its variants, which are problems that some lattice-based schemes rely on. In the following, Subsection *4.3.2.3* introduces two promising PQC schemes (i.e., FrodoKEM and CRYSTALS-Kyber), which besides being lattice-based, use different approaches and, finally, Subsection *4.3.2.4* compare both of them.

4.3.2.1 Notation

The notation used in this dissertation is the same one used in the supporting documentation of the FrodoKEM [126] and CRYSTALS-Kyber scheme [127]. The input and output of functions of the schemes are byte arrays, and $\mathcal{B} = \{0, ..., 255\}$ is a set of unsigned integers of 8-bits or a byte. Also, it is assumed 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 inner product of two n-dimensional vectors $\mathbf{a}, \mathbf{b}$ is represented by $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=0}^{n-1} a_i b_i$. Uppercase and lowercase bold letters are used for matrices and vectors, respectively, while $\mathbf{A}^T$ denotes the transpose of $\hat{\mathbf{A}}$ and the NTT of the matrix $\mathbf{A}$, respectively.

4.3.2.2 The LWE problem and its variants

This subsection aims to present an overview of the LWE problem and its variants, presenting its mathematical basis and comparing them. The LWE problem, proposed by Regev in [128], requests to recover a secret $s \in \mathbb{Z}_q^n$ knowing 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).

Later, Lyubashevsky in [129] proposed the R-LWE, a variant of the LWE problem based on ideal lattices. A lattice is considered ideal when it corresponds to an ideal in a particular algebraic structure, such as polynomial rings. They are very similar, but while the LWE works with integer elements, the R-LWE works with elements over polynomial rings. Based on the R-LWE ideas, Albrecht in [130] introduced the M-LWE. The M-LWE problem can be seen as an R-LWE problem but with the dimension of the polynomial ring greater than 1 – i.e., the R-LWE is an M-LWE problem but with a single dimension.

Comparing the LWE and R-LWE problems, they are distinct in many aspects. The LWE is a mature and well-studied cryptographic primitive that relies on the hardness of the worst case of Shortest Vector Problem (SVP) in a standard lattice. It is easily scalable, facilitating the adaptation for different security levels. Moreover, it is based on very conservative mathematical principles due to the lack of any additional structure, but it also degrades performance and requires long keys and ciphertext. On the other hand, R-LWE has an additional algebraic structure and relies on the worst case of an ideal lattice. Due to such additional algebraic, ideal lattices are more efficient than standard lattices because they need a small memory and perform better, but the addition of a structure that is difficult to scale might facilitate malicious attacks. Furthermore, while standard lattices are based mainly on matrix-by-vector (or matrix-by-matrix) multiplications, ideal lattices are based on polynomial multiplication, which considerably reduces the complexity and increases efficiency [131].

The M-LWE tries to get the best of the LWE and R-LWE. It indeed has an additional structure but it is less structured than R-LWE. According to recent cryptanalytic progress, an attack against schemes relying on the M-LWE problem seems to be less likely than against schemes relying on the R-LWE [132]. It also presents a similar performance when compared with the R-LWE and similar keys and ciphertext sizes. Finally, the M-LWE has better scalability than the R-LWE problem.

Based on this discussion, it can be said that the LWE problem is the most conservative one, privileging consolidated principles by the academic community, while the R-LWE is focused on performance. The M-LWE evaluates a trade-off between consolidated principles and performance.

4.3.2.3 Promising PQC schemes

This subsection aimed to describe in more detail the quantum-resistant asymmetric cryptographic schemes investigated in this dissertation (i.e., FrodoKEM and CRYSTALS-Kyber schemes). Both schemes belong to the same class, lattice-based cryptography; however, each of them relies on different mathematical principles.

Regarding the FrodoKEM scheme, it is important to emphasize that it is based on a conservative approach, relying on the LWE problem, which reduces its performance and requires a significant amount of memory usage. Although, the LWE problem, and as a consequence, the FrodoKEM scheme, does not present extra structures (such as used in R-LWE to increase performance and memory usage), which are not yet well studied and might present weaknesses in the future. On the other hand, CRYSTALS-Kyber scheme uses a lightweight approach, relying on the M-LWE problem, which is focused on performance and reduced memory usage. Besides it uses a few extra structures in the M-LWE problem, this scheme does not jeopardize security because it is still similar to unstructured lattices used in the LWE problem.

Note that there is an evident contrast between both schemes. The former is very conservative in secure premises and, as a consequence, compromises performance, while the latter performs very well but can present security flaws in the future. Due to that contrast, these two schemes are suitable to be compared and evaluated as possible choices to provide security for AMI. In the following, it is highlighted the main characteristics of these schemes and Subsection *4.3.2.4* compares both of them. Deeper analysis regarding FrodoKEM and CRYSTALS-Kyber schemes are approached in Chapters 5 and 6, respectively.

- **The FrodoKEM scheme**. It is an alternate candidate in the NIST PQC standardization process in the class of lattice-based cryptography and PKE/KEM category. This scheme is designed to be conservative yet practical post-quantum constructions whose security derives from parameterizations of the well-studied LWE problem, not being exposed to the possible weakness of the added structures in variants of this problem (e.g, R-LWE, M-LWE). As requested in the NIST PQC standardization process, FrodoKEM is designed for Adaptive Chosen Ciphertext Attack (IND-CCA2) security at three levels:

  - Level 1 in the NIST PQC standardization process (matching or exceeding the brute-force security of AES-128): FrodoKEM-640.
  - Level 3 in the NIST PQC standardization process (matching or exceeding the brute-force security of AES-192): FrodoKEM-976.
  - Level 5 in the NIST PQC standardization process (matching or exceeding the brute-force security of AES-256): FrodoKEM-1344.

  Each of the above schemes is provided with AES-128 or Secure Hash Algorithm and Keccak (SHAKE)128 algorithms to pseudorandomly generate a large public matrix. The use of the AES-128 variant is suitable for devices having intrinsic AES hardware acceleration, while SHAKE128 provides a competitive or better performance in comparison with the AES variant in the absence of hardware acceleration. For all security levels, the only operations required are additions and multiplications. FrodoKEM-976 and FrodoKEM-1344 are post-quantum schemes recommended by the German Federal Office for Information Security (BSI) as cryptographically suitable for long-term confidentiality [133].

- **The CRYSTALS-Kyber scheme.** It is the winner in the NIST PQC standardization process in PKE/KEM category. CRYSTALS-Dilithium, a CRYSTALS-Kyber variant for digital signature, is also one of the winners in the digital signature category. This scheme derives its security from the M-LWE problem, which uses modules lattices. This type of lattice can be thought of as a lattice that lies between the ones used in the LWE problem and those used in the R-LWE problem. They inherit the efficiency of the R-LWE but the only operations required are additions and multiplications in the $\mathbb{Z}_q$ field and the NTT. Moreover, the lattices used in this scheme have less algebraic structure than those used in R-LWE problems, being closer to the unstructured lattices used in the LWE problem. Consequently, if an algebraic attack appears in the future against R-LWE, it may be less effective against M-LWE problems due to its similarity with unstructured lattices. As requested in the NIST PQC standardization process, CRYSTALS-Kyber is designed for IND-CCA2 security at three levels:

  - Level 1 in the NIST PQC standardization process (matching or exceeding the brute-force security of AES-128): CRYSTALS-Kyber-512.
  - Level 3 in the NIST PQC standardization process (matching or exceeding the brute-force security of AES-192): CRYSTALS-Kyber-768.
  - Level 5 in the NIST PQC standardization process (matching or exceeding the brute-force security of AES-256): CRYSTALS-Kyber-1024.

  Similar to FrodoKEM, each of the above schemes are provided with SHAKE variants or AES-256 to pseudorandomly generate public keys. As previously mentioned, AES-256 variants are suitable for devices having intrinsic AES hardware acceleration. The design and implementation of CRYSTALS-Kyber and Dilithium have been supported by IBM, NXP, and other organizations and are the base of the quantum-resistant cryptography Transport Layer Security (TLS) for IBM Key Protect (the IBM Cloud key management solution) [134].

4.3.2.4 Schemes Overview

Table 1 summarizes the main characteristics of FrodoKEM and CRYSTALS-Kyber schemes. As already mentioned, both schemes are lattice-based, but FrodoKEM relies on the LWE problem, while the CRYSTALS-Kyber in the M-LWE problem. As a consequence, it reflects directly in the public key, private key, and ciphertext sizes, which are more than ten times larger in FrodoKEM than in CRYSTALS-Kyber. Considering these size differences and the problem each scheme is based on, it can be said that the FrodoKEM scheme has a conservative approach, while the CRYSTALS-Kyber scheme is a lightweight approach, see Section *4.3.2.2*. In

Table 1 – Overview of FrodoKEM and CRYSTALS-Kyber schemes.

	FrodoKEM 640/976/1344	CRYSTALS-Kyber 512/768/1024
Class	Lattice	Lattice
Mathematical Problem	LWE	M-LWE
Public Key Size (bytes)	9616/15632/21520	800/1184/1568
Private Key Size (bytes)	19888/31296/43088	1632/2400/3168
Ciphertext Size (bytes)	9720/15744/21632	768/1088/1568
Approach	Conservative	Lightweight

this sense, the CRYSTALS-Kyber scheme tends to consume fewer memory resources and can be significantly faster than the FrodoKEM scheme.

4.4 SUMMARY

This chapter aimed to contextualize the urgent necessity of works focused on PQC. To do so, firstly, a historic regarding quantum computer is provided, starting from quantum mechanics principles introduced by Heisenberg until the quantum processors already available nowadays. A brief discussion regarding quantum mechanics is provided along with the comparison between classical and quantum computers. Later, important cryptographic concepts are discussed, such as asymmetric and symmetric cryptography. The former mainly used for key agreement between parties and the latter for ciphering/deciphering and authentication. Based on the discussion regarding quantum computer and cryptography fundamentals, the importance of PQC is highlighted. Briefly, under the threat of quantum computer running Shor's algorithm and the necessity of the modern digitized society to exchange key securely, the PQC arises as a solution. In this sense, the NIST PQC standardization process was proposed and 69 submission was received under 6 different classes. Under these classes, due to its characteristics, the lattice-based one stood out from the others. Among the lattice-based schemes, two of them presented interesting characteristics for AMIs and are interesting for deeper evaluation: FrodoKEM and CRYSTALS-Kyber. The former with strong cryptographic principles, at least in theory, while the latter based on a lightweight approach.

5 THE FRODOKEM SCHEME

The asymmetric cryptographic scheme called FrodoKEM [126] is an alternate candidate in the NIST PQC standardization process and relies on a well-studied principle, called LWE. Mainly due to that, this scheme is considered cryptographically strong, at least in theory, by the academic community, and no major vulnerability was reported as far as is known. Although, it comes at a cost of timing performance and resource usage. In this sense, there are studies that discuss software and hardware implementations of FrodoKEM [50, 135], which mainly aims to overcome those constraints. Besides these implementations, the literature also reports hardware/software co-design implementations.

The FrodoKEM scheme is an option to implement QR-AMI due to its strong cryptographic principles; however, some aspects must be investigated because of the large-scale implementation of AMIs impose constraints on costs, which directly reflects on using hardware-constrained devices. In this regard, it must be verified if it is feasible to implement the FrodoKEM in these devices. For instance, Fritzmann *et al.* [55] 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. However, the literature lacks evaluations of the FrodoKEM scheme using a hardware/software co-design implementation on hardware-constrained SoC FPGA, in which hardware accelerates the main bottlenecks of the scheme. Such evaluations are valuable analysis for AMIs.

This chapter investigates the usefulness of implementing the FrodoKEM scheme as the KEM for ensuring quantum-resistant symmetric key exchange between nodes in the AMI relying on the **Approach #3**; however its findings are also valid for **Approaches #1** and **#2**. The choice of the FrodoKEM scheme relies mainly on its strong cryptographic principles, a conservative choice since the cryptanalysis shows that attacks on a non-structured lattice (i.e., LWE problem) are less likely than in a structured lattice (i.e., R-LWE and M-LWE problem). In this regard, the software implementation of the FrodoKEM scheme using a microcontroller and a low-cost processor and the hardware/software co-design implementation using an SoC device are detailed. The main contributions of this chapter are as follows:

- An analysis of functions of the FrodoKEM scheme that are most time-consuming which are suitable candidates to be hardware accelerated when a hardware-constrained QR-DCM based on an SoC FPGA device is available. Moreover, a presentation of a hardware/software co-design implementation of the FrodoKEM scheme on a hardware-constrained QR-DCM that uses an SoC FPGA device.

- A performance comparison between software implementation of the FrodoKEM scheme using a microcontroller and a low-cost processor with the hardware/software co-design implementation using an SoC device in terms of execution time. Also, an analysis of hardware resource usage demanded by the hardware/software co-design implementation of the FrodoKEM scheme in an SoC FPGA device.

The rest of this chapter is organized as follows: Section 5.1 provides a concise presentation of the FrodoKEM scheme; In Section 5.2 a detailed description of the hardware used in this dissertation is provided; Section 5.3 addresses the software implementation of FrodoKEM and Section 5.4 the hardware/software co-design implementation of the scheme; Section 5.5 discusses numerical results of hardware resource usage, execution time, and performance comparison between implementations; finally, Section 5.6 outlines concluding remarks.

5.1 BACKGROUND OF THE FRODOKEM SCHEME

The FrodoCCS key exchange scheme [136] was designed by exchanging a little efficiency for high-security trust in the post-quantum era. Its simplicity is confirmed by applying only basic operations, such as addition and multiplication. Furthermore, its parameter adjustments are more flexible and easier to scale than ideal lattice-based schemes, such as the NewHope [137]. The latter has more restrictions as it uses the NTT algorithm for polynomial multiplication. Consequently, FrodoCCS can achieve different security levels with linear resource expenditure. Based on FrodoCCS, the FrodoKEM scheme [126], which is a KEM, was submitted to the NIST PQC standardization process. It was selected for the third round of the competition as one of eight alternate candidates. The values of the parameters used by Algorithms #1, #2, and #3 are specified in Table 2. Note that D is the exponent which defines the scheme modulus $q = 2^D$; n, $\overline{n}$, and $\overline{m}$ are integer matrix dimensions with $n \equiv 0 \ (mod\ 8)$. Observing Table 2, we can see that the size of public key, private key, and ciphertext are considered large, especially for hardware-constrained equipment. More details about the choice of these parameters are shown in [126].

5.1.1 The Frodo Key Encapsulation Mechanism Scheme

The FrodoKEM scheme can be basically divided into three algorithms: key pair generation, encapsulation, and decapsulation [126] as described in Algorithms #1, #2, and #3, respectively. A few subroutines are called by these algorithms, see [126] for more details. Briefly, the **Gen**(.) function receives as input a seed and outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times n}$ which was generated using a hash function. Similarly, the **SampleMatrix**(.) function outputs a matrix sampled from the χ error probability distribution. The **Pack**(.) function transforms the received matrix into a bit string, while **Unpack**(.) function does the opposite. Finally, the **Encode**(.) function encodes bit

Table 2 – Parameter values for the FrodoKEM scheme.

FrodoKEM	D	q	n	$\overline{n}$	$\overline{m}$	Public key size (bytes)	Private key size (bytes)	Ciphertext size (bytes)
640	15	2^{15}	640	8	1	9616	19888	9720
976	16	2^{16}	976	8	1	15632	31296	15744
1344	16	2^{16}	1344	8	1	21520	43088	21632

Algorithm #1: Key pair generation

```
Input: None
Output:
Public key: $pk \in \{0,1\}^{\mathrm{len}_{\mathbf{seed_A}}+D\cdot n\bar{n}}$,
Private key: $sk' \in \{0,1\}^{\mathrm{len}_{\mathbf{s}}+\mathrm{len}_{\mathbf{seed_A}}+D\cdot n\bar{n}} \times \mathbb{Z}_q^{n\times\bar{n}} \times \{0,1\}^{\mathrm{len}_{\mathbf{pkh}}}$
Procedure:
Choose uniformly random seeds: $\mathbf{s}||\mathbf{seed_{SE}}||\mathbf{z} \leftarrow_{\$} U(\{0,1\}^{\mathrm{len}_{\mathbf{s}}+\mathrm{len}_{\mathbf{seed_{SE}}}+\mathrm{len}_{\mathbf{z}}})$
Generate a pseudo-random seed: $\mathbf{seed_A} \leftarrow \mathbf{SHAKE}(\mathbf{z}, \mathrm{len}_{\mathbf{seed_A}})$
Generate Matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$ via $\mathbf{A} \leftarrow \mathbf{Gen}(\mathbf{seed_A})$
Generate pseudo-random bit string:
 $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, ..., \mathbf{r}^{(2n\bar{n}-1)}) \leftarrow \mathbf{SHAKE}(0\text{x}5\text{F}||\mathbf{seed_{SE}}, 2n\bar{n}\cdot \mathrm{len}_{\chi})$
Sample error matrix $\mathbf{S^T} \leftarrow \mathbf{SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, ..., \mathbf{r}^{(n\bar{n}-1)}), \bar{n}, n, T_{\chi})$
Sample error matrix $\mathbf{E} \leftarrow \mathbf{SampleMatrix}((\mathbf{r}^{(n\bar{n})}, \mathbf{r}^{(n\bar{n}+1)}, ..., \mathbf{r}^{(2n\bar{n}-1)}), n, \bar{n}, T_{\chi})$
Compute $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$
Compute $\mathbf{b} \leftarrow \mathbf{Pack}(\mathbf{B})$
Compute $\mathbf{phk} \leftarrow \mathbf{SHAKE}(\mathbf{seed_A}||\mathbf{b}, \mathrm{len}_{\mathbf{pkh}})$
Return:
Public key $pk \leftarrow \mathbf{seed_A}||\mathbf{b}$
Private key $sk' \leftarrow (\mathbf{s}||\mathbf{seed_A}||\mathbf{b}, \mathbf{S^T}, \mathbf{pkh})$
```

strings as mod-q integer matrices. On the other hand, the **Decode**(.) function does the inverse operation. Finally, all bit string lengths ($\mathrm{len}_{\mathbf{seed_A}}$, $\mathrm{len}_{\mathbf{seed_{SE}}}$, len_{μ}, $\mathrm{len}_{\mathbf{s}}$, $\mathrm{len}_{\mathbf{k}}$, $\mathrm{len}_{\mathbf{pkh}}$, $\mathrm{len}_{\mathbf{ss}}$, $\mathrm{len}_{\mathbf{z}}$, len_{χ}) are previously known constants; and T_{χ} is the distribution table for sampling.

The main part of the key pair generation (Algorithm #1) is the calculation of the LWE sample operation $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$. The matrix $\mathbf{A}$ is generated by a pseudo-random seed while $\mathbf{seed_A}$ is created from a uniformly random seed hashed by a function. The FrodoKEM scheme can use two hash functions: a hash based on the AES cipher and the SHAKE128 hash algorithm. The matrices $\mathbf{E}$ and $\mathbf{S}$ are sampled according to the distribution χ. Later, matrix $\mathbf{B}$ is packed into bit string $\mathbf{b}$, and the bit string $\mathbf{seed_A}$ and $\mathbf{b}$ are hashed to get a hash value $\mathbf{phk}$. Finally, the public key pk is composed of $\mathbf{seed_A}$ and $\mathbf{b}$, while the private key sk' is composed of $\mathbf{s}$ (previously uniformly random generated), $\mathbf{seed_A}$, $\mathbf{b}$, $\mathbf{S^T}$, and $\mathbf{pkh}$.

In the encapsulation (Algorithm #2), three noise matrices are generated: $\mathbf{S'}$, $\mathbf{E'}$, and $\mathbf{E''}$. To create these matrices, a pseudo-random bit string is sampled according to χ. The input of the algorithm, bit strings $\mathbf{seed_A}$ and $\mathbf{b}$, are used to retrieve matrices $\mathbf{A}$ and $\mathbf{B}$. Later, they are used to calculate $\mathbf{B'} \leftarrow \mathbf{S'A} + \mathbf{E'}$ and $\mathbf{V} \leftarrow \mathbf{S'B} + \mathbf{E''}$. Using the matrix $\mathbf{V}$ added by the encoded μ (previously uniformly random generated), the matrix $\mathbf{C}$ is created. Then, matrices $\mathbf{B'}$ and $\mathbf{C}$ are packed, generating bit strings $\mathbf{c}_1$ and $\mathbf{c}_2$, which concatenated form the ciphertext. Finally, bit strings $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{k}$ (pseudo-randomly generated using the hash function) are hashed, creating the shared secret key $\mathbf{ss}$.

The decapsulation (Algorithm #3) aims to check if the ciphertext ($\mathbf{c}_1 \,||\, \mathbf{c}_2$) is valid. To keep it short, bit strings $\mathbf{c}_1$ and $\mathbf{c}_2$ are unpacked, retrieving matrices $\mathbf{B'}$ and $\mathbf{C}$. Then, $\mathbf{M} \leftarrow \mathbf{C}+\mathbf{B'S}$ is calculated and then decoded, getting μ'. The encapsulation steps are redone, although this

Algorithm #2: Encapsulation

```
Input:
Public key: $pk = \mathbf{seed_A}||\mathbf{b} \in \{0,1\}^{\mathrm{len}_{\mathbf{seed_A}}+D\cdot n\overline{n}}$
Output:
Ciphertext: $\mathbf{c}_1||\mathbf{c}_2 \in \{0,1\}^{(\overline{m}n+\overline{mn})D}$
Shared secret key: $\mathbf{ss} \in \{0,1\}^{\mathrm{len}_{\mathbf{ss}}}$
Procedure:
Choose a uniformly random key: $\mu \leftarrow_{\$} U(\{0,1\}^{\mathrm{len}_{\mu}})$
Compute $\mathbf{pkh} \leftarrow \mathbf{SHAKE}(pk, \mathrm{len}_{\mathbf{phk}})$
Generate pseudo-random values $\mathbf{seed_{SE}}||\mathbf{k} \leftarrow \mathbf{SHAKE}(\mathbf{phk}||\mu, \mathrm{len}_{\mathbf{seed_{SE}}} + \mathrm{len}_{\mathbf{k}})$
Generate pseudo-random bit string:
 $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, ..., \mathbf{r}^{(2\overline{m}n+\overline{mn}-1)}) \leftarrow \mathbf{SHAKE}(0\mathrm{x}96||\mathbf{seed_{SE}}, (2\overline{m}n + \overline{mn}) \cdot \mathrm{len}_{\chi})$
Sample error matrix $\mathbf{S}' \leftarrow \mathbf{SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, ..., \mathbf{r}^{(\overline{m}n-1)}), \overline{m}, n, T_{\chi})$
Sample error matrix $\mathbf{E}' \leftarrow \mathbf{SampleMatrix}((\mathbf{r}^{(\overline{m}n)}, \mathbf{r}^{(\overline{m}n+1)}, ..., \mathbf{r}^{(2\overline{m}n-1)}), \overline{m}, n, T_{\chi})$
Generate $\mathbf{A} \leftarrow \mathbf{Gen}(\mathbf{seed_A})$
Compute $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$
Compute $\mathbf{c}_1 \leftarrow \mathbf{Pack}(\mathbf{B}')$
Sample error matrix $\mathbf{E}''$:
 $\mathbf{E}'' \leftarrow \mathbf{SampleMatrix}((\mathbf{r}^{(2\overline{m}n)}, \mathbf{r}^{(2\overline{m}n+1)}, ..., \mathbf{r}^{(2\overline{m}n+\overline{mn}-1)}), \overline{m}, \overline{n}, T_{\chi})$
Compute $\mathbf{B} \leftarrow \mathbf{Unpack}(\mathbf{b}, n, \overline{n})$
Compute $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
Compute $\mathbf{C} \leftarrow \mathbf{V} + \mathbf{Encode}(\mu)$
Compute $\mathbf{c}_2 \leftarrow \mathbf{Pack}(\mathbf{C})$
Compute $\mathbf{ss} \leftarrow \mathbf{SHAKE}(\mathbf{c}_1||\mathbf{c}_2||\mathbf{k}, \mathrm{len}_{\mathbf{ss}})$
Return:
Ciphertext $\mathbf{c}_1||\mathbf{c}_2$
Shared secret key $\mathbf{ss}$
```

time generating matrices $\mathbf{B}''$ and $\mathbf{C}'$. If matrices $\mathbf{B}''$ and $\mathbf{C}'$ matches with matrices $\mathbf{B}'$ and $\mathbf{C}$, the shared secret key returned is the hash of $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{k}'$ (pseudo-randomly generated using hash function based on μ'). Otherwise, the shared secret key returned is the hash of $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{s}$ (part of the secret key sk').

More information and details about the parameters, the error sampling procedure, the lattice structure, and security proofs can be consulted in the official specification of FrodoKEM [126, 138].

5.2 HARDWARE DESCRIPTION

This section aims to describe and justify the hardware and implementation choices that will be used to evaluate the FrodoKEM scheme. Depending on application constraints and relying on **Approach #3**, the QR-DCM implementation can be accomplished using a microcontroller- and SoC-based hardware because both of them offer distinct trade-offs for implementing QR-DCMs. For instance, adopting a microcontroller results in a low-cost solution that demands a long execution time, while a SoC-based solution is more expensive but offers hardware acceleration of

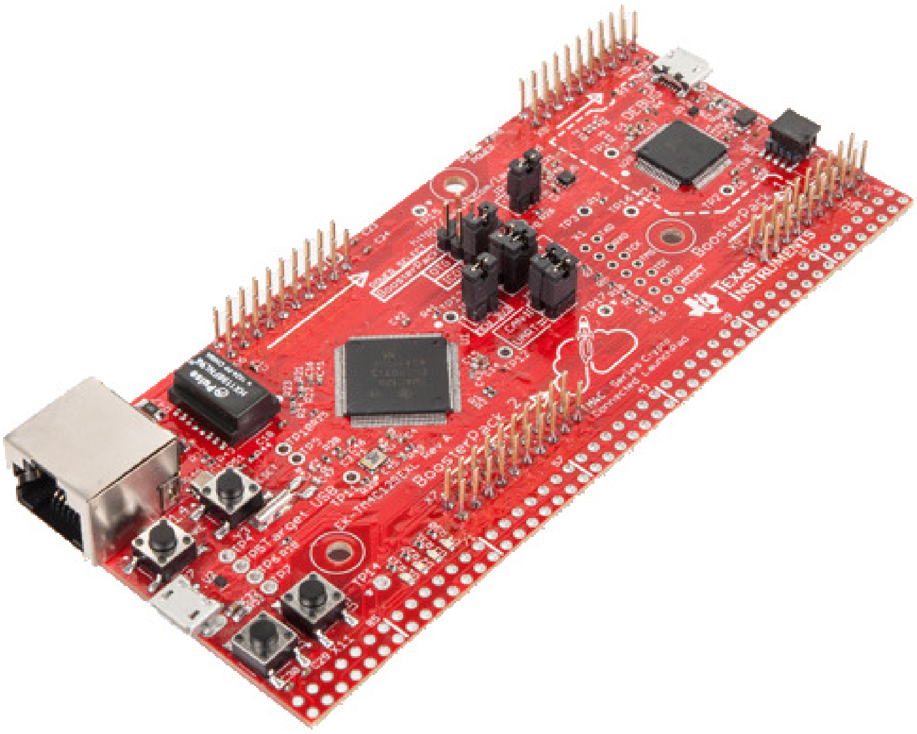
```
Algorithm #3: Decapsulation
  Input:
  Ciphertext: $\mathbf{c}_1||\mathbf{c}_2 \in \{0,1\}^{(\overline{m}n+\overline{mn})D}$
  Private key: $sk' = (\mathbf{s}||\mathbf{seed_A}||\mathbf{b}, \mathbf{S^T}, \mathbf{pkh}) \in \{0,1\}^{\mathrm{len}_\mathbf{s}+\mathrm{len}_{\mathbf{seed_A}}+D\cdot n\overline{n}} \times \mathbb{Z}_q^{n\times\overline{n}} \times \{0,1\}^{\mathrm{len}_\mathbf{pkh}}$
  Output:
  Shared secret key: $\mathbf{ss} \in \{0,1\}^{\mathrm{len}_\mathbf{ss}}$
  Procedure:
  Compute $\mathbf{B}' \leftarrow \mathbf{Unpack}(\mathbf{c}_1, \overline{m}, n)$
  Compute $\mathbf{C} \leftarrow \mathbf{Unpack}(\mathbf{c}_2, \overline{m}, \overline{n})$
  Compute $\mathbf{M} \leftarrow \mathbf{C} + \mathbf{B}'\mathbf{S}$
  Compute $\mu' \leftarrow \mathbf{Decode}(\mathbf{M})$
  Parse $pk \leftarrow \mathbf{seed_A}||\mathbf{b}$
  Generate pseudo-random values:
   $\mathbf{seed}'_\mathbf{SE}||\mathbf{k}' \leftarrow \mathbf{SHAKE}(\mathbf{phk}||\mu', \mathrm{len}_{\mathbf{seed_{SE}}} + \mathrm{len}_\mathbf{k})$
  Generate pseudo-random bit string:
   $(\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, ..., \mathbf{r}^{(2\overline{m}n+\overline{mn}-1)}) \leftarrow \mathbf{SHAKE}(0\mathrm{x}96||\mathbf{seed}'_\mathbf{SE}, 2\overline{m}n + \overline{mn} \cdot \mathrm{len}_\chi)$
  Sample error matrix $\mathbf{S}' \leftarrow \mathbf{SampleMatrix}((\mathbf{r}^{(0)}, \mathbf{r}^{(1)}, ..., \mathbf{r}^{(\overline{m}n-1)}), \overline{m}, n, T_\chi)$
  Sample error matrix $\mathbf{E}' \leftarrow \mathbf{SampleMatrix}((\mathbf{r}^{(\overline{m}n)}, \mathbf{r}^{(\overline{m}n+1)}, ..., \mathbf{r}^{(2\overline{m}n-1)}), \overline{m}, n, T_\chi)$
  Generate $\mathbf{A} \leftarrow \mathbf{Gen}(\mathbf{seed_A})$
  Compute $\mathbf{B}'' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$
  Sample error matrix $\mathbf{E}'' \leftarrow \mathbf{SampleMatrix}((\mathbf{r}^{(2\overline{m}n)}, \mathbf{r}^{(2\overline{m}n+1)}, ..., \mathbf{r}^{(2\overline{m}n+\overline{mn}-1)}), \overline{m}, \overline{n}, T_\chi)$
  Compute $\mathbf{B} \leftarrow \mathbf{Unpack}(\mathbf{b}, n, \overline{n})$
  Compute $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
  Compute $\mathbf{C}' \leftarrow \mathbf{V} + \mathbf{Encode}(\mu')$
  if $\mathbf{B}'||\mathbf{C} = \mathbf{B}''||\mathbf{C}'$ then
    $\overline{\mathbf{k}} \leftarrow \mathbf{k}'$
  else
    $\overline{\mathbf{k}} \leftarrow \mathbf{s}$
  end
  Return:
  Shared secret key $\mathbf{ss} \leftarrow \mathbf{SHAKE}(\mathbf{c}_1||\mathbf{c}_2||\overline{\mathbf{k}}, \mathrm{len}_\mathbf{ss})$
```

time-consuming routines and, consequently, a short execution time. Brief descriptions of these hardware are detailed as follows:

- **Microcontroller**. This hardware has low-energy consumption and, consequently, a not-so-fast processing unit for performing its tasks. It fits well for low processing power (e.g., processing a small amount of data). The chosen microcontroller is in the EK-TM4C129EXL evaluation kit [139], which relies on an ARM Cortex-M4F with 120 MHz operation, 1024 kB flash memory, 256 kB single-cycle System SRAM, 6 kB of EEPROM, and other features and well-established interfaces. Figure 10 shows the development board based on this microcontroller, which only allows software implementation.

- **SoC FPGA device**. The chosen SoC FPGA device is in the MicroZed 7010 Board [140]. Consequently, the development board is a System on Module (SoM) based on the Xilinx

Figure 10 – EK-TM4C129EXL evaluation kit



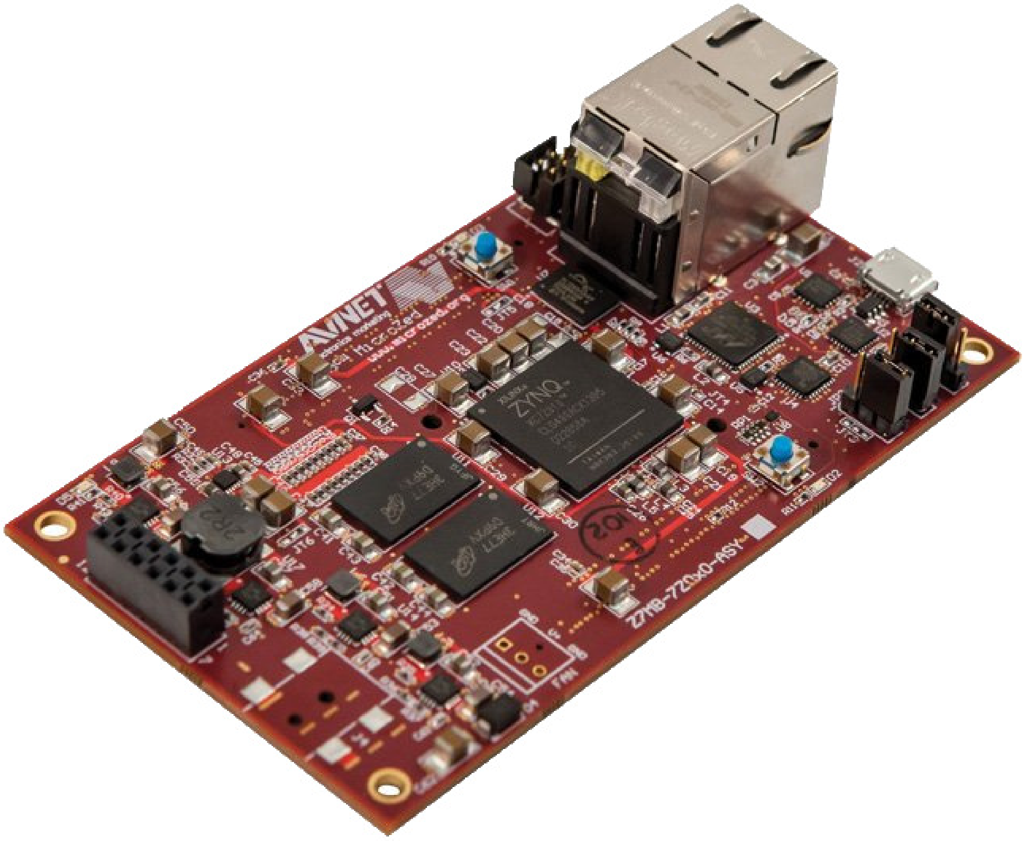
Source: Texas Instruments.

Zyqn-7000 SoC with 1 GB of DDR3 SDRAM, 128 Mb of QSPI Flash, 33.33 MHz oscillator, and other features and well-established interfaces. The Xilinx Zyqn-7000 SoC used is an XC7Z010-1CLG400C, which is composed of a Processing System (PS) and Programmable Logic (PL). The PS is based on an ARM Cortex-A9 processor with ARMv7 architecture. On the other hand, the PL is based on a FPGA with 28 k programmable logic cells, 17.6 k Lookup Tables (LUTs), 35.2 k registers, 60 Block Random Access Memory (BRAM) with 36 kb each, and 80 Digital Signal Processing (DSP) blocks [140]. Using a Phase-Locked Loop (PLL), 666.66 MHz and 100 MHz clocks are derived from the 33.33 MHz built-in oscillator to feed the PS and PL, respectively. This development board is flexible and allows different implementations, such as software implementation (i.e, using only the ARM Cortex-A9 processor) or hardware/software co-design implementation (i.e., using ARM Cortex-A9 and FPGA together). Figure 11 shows the development board.

Based on these two development boards, the following implementations are discussed:

- **Implementation #1**. A bare-metal software implementation using the ARM Cortex-M4F in the TM4C129EXL-based development board. This implementation is based on a low-cost microcontroller with low-power processing unit.

Figure 11 – MicroZed 7010 Board



Source: Texas Instruments.

- **Implementation #2**. A bare-metal software implementation using the ARM Cortex-A9 in the MicroZed 7010 board. When compared with **Implementation #1**, this implementation has more processing capacity because it is based on a high-power processing unit, which reflects in a more expensive device. However, among the processors available in the market, it can yet be considered a low-cost processor.

- **Implementation #3**. A hardware/software co-design implementation using the ARM Cortex-A9 and FPGA also in the MicroZed 7010 board. This implementation, when compared with **Implementations #1** and **#2**, is the one which has the most processing capacity. However, this reflects on costs, but even so it is considered a low-cost SoC device when compared with other SoCs in the market.

Implementations #1 and **#2** aim to show the performance difference between low- and high-power processing units based on a microcontroller (ARM Cortex-M4F) and a low-power processor (ARM Cortex-A9) only, while **Implementation #3** seeks to evaluate the improvement when hardware acceleration is available. Table **??** summarize this section, systematically showing the implementation technique, hardware, processing unit, and approach of **Implementations #1**, **#2**, and **#3**.

Table 3 – Summarize of the implementations.

	Implementation #1	**Implementation #2**	**Implementation #3**
Implementation technique	Software	Software	Hardware/software co-design
Hardware	Microcontroller	SoC FPGA device	SoC FPGA device
Processing Unit	ARM Cortex-M4F	ARM Cortex-A9	ARM Cortex-A9 + XC7Z010-1CLG400C
Aproach	Low-cost	Trade-off between cost and performance	High performance

5.3 SOFTWARE IMPLEMENTATION

This section details the software implementation of the FrodoKEM scheme on ARM Cortex-M4F microcontroller (i.e., **Implementation** #**1**) and ARM Cortex-A9 processor (i.e., **Implementation** #**2**). To do so, this section is organized as follows: In Subsection **5.3.1**, it is discussed a preliminary analysis of the CRYSTALS-Kyber scheme implemented on the ARM Cortex-M4F microcontroller and ARM Cortex-A9 processor, showing the most time-consuming functions that must be considered to be hardware implemented; and Subsection **5.3.2** provides additional details about the software implementation of the matrix-by-matrix multiplications and the SHAKE128 hash function.

5.3.1 Preliminary Analysis

A preliminary analysis of the FrodoKEM scheme is necessary to identify the most time-consuming functions. As already discussed in Section 5.1, the FrodoKEM scheme relies on conservative primitives that are more complex to be performed. Due to that, only the FrodoKEM-640 is approached in this chapter, as the two versions of FrodoKEM with higher security levels (i.e., FrodoKEM-976 and FrodoKEM-1344) are even more penalized in terms of timing performance, being prohibitive their deployment in data communication systems which requires rapid responses and are cost-effective, such as AMIs. For the sake of simplicity, FrodoKEM-640 will be called only FrodoKEM henceforth.

To identify the main bottlenecks associated with the **Implementations** #**1** and #**2** of FrodoKEM, the code from [124] was executed on an ARM Cortex-M4F microcontroller and Cortex-A9 ARM processor. After a detailed analysis of the code execution, the following three operations stood out as the most time-consuming:

- $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$. It is used in the key pair generation process, has a high computational burden because of its large matrix-by-matrix multiplication, which requires long loops with addition and multiplication operations.

Table 4 – The relative execution time, in percentage, of the FrodoKEM-640 scheme based on **Implementation #1**.

Operation	Key Pair Generation	Encapsulation	Decapsulation
$\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$	28.84%	-	-
$\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$	-	46.79%	45.75
SHAKE128	57.95%	42.53%	43.76%
Others	13.21%	10.68%	10.49%

- $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$. It is even more expensive than $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$ because it is used in both encryption and decryption algorithms.

- SHAKE128 hash function. The most expensive operation is the SHAKE128 hash function, a specific version of the SHAKE hash function. It is time-consuming in all three algorithms due to its loop-based structure.

These operations demand relevant computational burdens because of the large size of FrodoKEM keys and the use of a public or private key, directly or indirectly. Table 4 summarizes the relative time-consuming of the functions used by the FrodoKEM-640 scheme in the ARM Cortex-M4F microcontroller. Similar results were achieved for the ARM Cortex-A9 processor; however, for simplicity, they are omitted.

5.3.2 Implementation

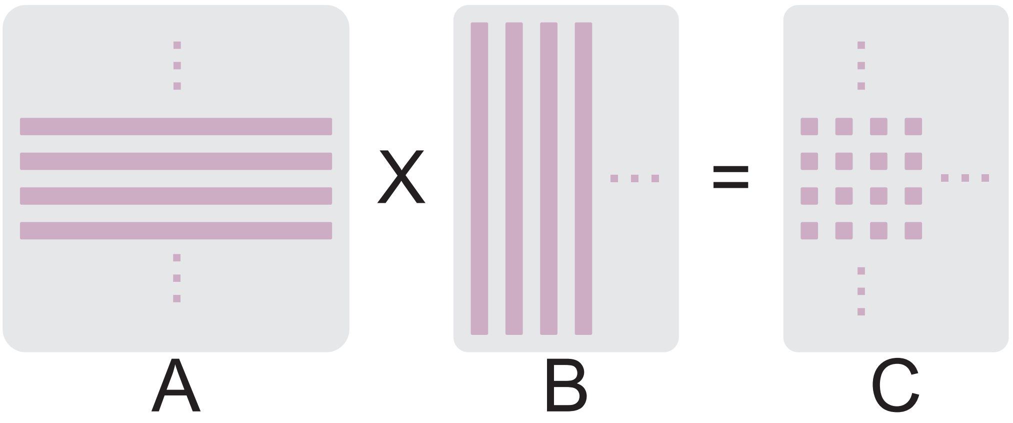
Relying on the preliminary analysis presented in Subsection **5.3.1**, three functions must be discussed in more detail to understand why they are the most time-consuming routines: $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$ and $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$, and SHAKE128 hash function.

Starting with the $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$ operation, due to the size of matrix **A**, it is unfeasible to generate it all at once in an embedded device due to its memory constraints. Therefore, matrix **A** is generated slightly differently as presented in Algorithm #4. To reduce the necessity of large memory, generating parts of the matrix **A** on the fly and overwritten by a new part after use is proposed. Due to this technique, the FrodoKEM scheme can be embedded in hardware-constrained equipment. Therefore, the matrix-by-matrix multiplication **AS** has three main loops. Four rows of the matrix **A** are generated on the fly using the SHAKE128 hash function in the outer loop. The middle loop selects one column from the previously fully generated matrix **S**. In the inner loop, four elements, one element of each of four generated rows of the matrix **A**, are multiplied by one element from the selected matrix column **S**, and the four results are accumulated. Note that there is a trade-off between the number of rows of matrix **A** generated at once versus memory resources used. If more rows are generate at once, more memory is required and vice-versa. Algorithm #4 shows the process in detail, as illustrated in Figure 12.

```
Algorithm #4: $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$
  Input:
  Matrix S: $\mathbf{S} \in \mathbb{Z}_q^{n\times\overline{n}}$
  Matrix E: $\mathbf{E} \in \mathbb{Z}_q^{n\times\overline{n}}$
  Seed: $\mathbf{seed_A} \in \{0,1\}^{\mathrm{len}_{\mathbf{seed_A}}}$
  Output:
  Matrix B: $\mathbf{B} \in \mathbb{Z}_q^{n\times\overline{n}}$
  Procedure:
  $\mathbf{A} \in \mathbb{Z}_q^{4\times n} \leftarrow \{0\}^{4\times n}$
  $\mathbf{B} \leftarrow \mathbf{E}$
  for $i \leftarrow 0, 4, 8, ..., n$ do
   $\mathbf{A}[0..3] \leftarrow shake128(\mathbf{seed_A})$
   for $k \leftarrow 0, ..., \overline{n}$ do
    $sum \in \mathbb{Z}_q^4 \leftarrow \{0\}^4$
    for $j \leftarrow 0, ..., n$ do
     $sum[0..3] \leftarrow sum[0..3] + \mathbf{A}[0..3, j] \cdot \mathbf{S}[k, j]$
    end
    $\mathbf{B}[i + 0..3, k] \leftarrow \mathbf{B}[i + 0..3, k] + sum[0..3]$
   end
  end
```

The operation $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A}+\mathbf{E}'$ occurs differently from the previous one. It also (re)generates the matrix **A** on the fly, although the multiplication process must be adapted. It occurs because the regenerated matrix **A**, used in the encapsulation and decapsulation processes, must be the same as used in the key pair generation. Therefore, each SHAKE128 hash function will generate a row of the matrix **A**, which will slightly complicate the logic of the multiplication process, considering that the matrix **A** in this operation is on the right-hand side and it is no longer possible to multiple an entire row of the matrix **A** by an entire column of the matrix **S**, in this case, the matrix **S′**. The Algorithm #5 presents this matrix-by-matrix multiplication process, which is also pictured in Figure 13.

Figure 12 – Operation $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$. Multiplication of four rows of the matrix **A** by matrix **S**.

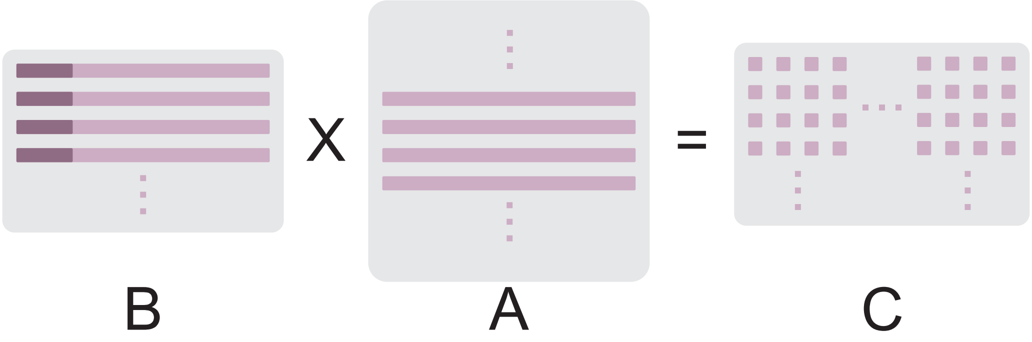


Source: Personal collection.

```
Algorithm #5: $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$
Input:
Matrix $\mathbf{S}'$: $\mathbf{S}' \in \mathbb{Z}_q^{n \times \overline{n}}$
Matrix $\mathbf{E}'$: $\mathbf{E}' \in \mathbb{Z}_q^{n \times \overline{n}}$
Seed: $\mathbf{seed_A} \in \{0, 1\}^{\mathrm{len}_{\mathbf{seed_A}}}$
Output:
Matrix $\mathbf{B}'$: $\mathbf{B}' \in \mathbb{Z}_q^{n \times \overline{n}}$
Procedure:
$\mathbf{A} \in \mathbb{Z}_q^{4 \times n} \leftarrow \{0\}^{4 \times n}$
$\mathbf{B}' \leftarrow \mathbf{E}'$
for $i \leftarrow 0, 4, 8, ..., n$ do
 $\mathbf{A}[0..3] \leftarrow shake128(\mathbf{seed_A})$
 for $k \leftarrow 0, ..., \overline{n}$ do
  $sum \in \mathbb{Z}_q^n \leftarrow \{0\}^n$
  for $j \leftarrow 0, ..., 4$ do
   for $p \leftarrow 0, ..., n$ do
    $sum[p] \leftarrow sum[p] + \mathbf{S}'[k, i + j] \cdot \mathbf{A}[j, p]$
   end
  end
  for $j \leftarrow 0, ..., n$ do
   $\mathbf{B}[k, j] \leftarrow \mathbf{B}[k, j] + sum[j]$
  end
 end
end
```

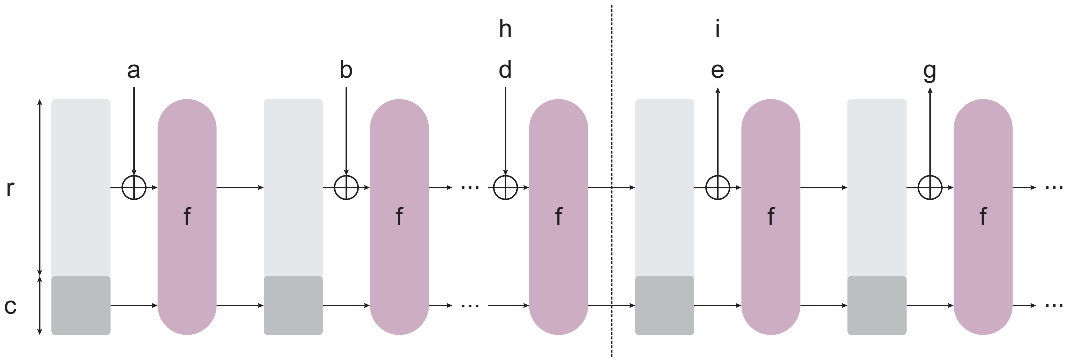
Finally, SHAKE128 [141] is a hash function with an output length of 256-bits and a security level of 128-bits. SHAKE128 is an instance of SHA-3, the latest member of the Secure Hash Algorithm family of standards, released by NIST. The SHA-3 family is based on the sponge construction [142], which is shown in Figure 14. The SHAKE128 hash function uses the Keccak-f[1600] function as the transform function composed of five permutation steps, whose parameters are the block size r equal to 1344-bits and its capacity c equal to 256-bits, resulting in the internal state with 1600-bits. The SHAKE128 hash function can be divided into two main

Figure 13 – Operation $\mathbf{S}'\mathbf{A} \leftarrow \mathbf{S}' \times \mathbf{A}$. Multiplication of first four elements of the first four rows of the matrix $\mathbf{S}'$, highlighted in dark purple, by four rows of the matrix $\mathbf{A}$.



Source: Personal collection.

Figure 14 – Sponge construction.



Source: Personal collection.

parts: the absorbing and squeezing phases. In the absorbing phase, the input blocks (message) are *xored* into the bit string r of the internal state. Then, the internal state is inputted in the Keccak-f[1600] function. When the entire input message is absorbed, the squeeze phase begins. In this case, the outputted blocks are read from the bit string r of the state, alternating with the Keccak-f[1600] function until the desired output size is reached.

5.4 HARDWARE/SOFTWARE CO-DESIGN IMPLEMENTATION

This section details the proposed hardware/software co-design implementation using the ARM Cortex-A9 together with the FPGA (i.e., **Implementation #3**) of the FrodoKEM scheme. Considering the results in Table 4, it can be seen that the execution time reduction can be attained by implementing the most time-consuming functions

Figure 15 shows the block diagram of the proposed hardware/software co-design implementation of the FrodoKEM scheme. The PS, based on an ARM Cortex-A9, is where the software implementation is placed. The interconnect instance is responsible for providing an interface between the PL using the Advanced eXtensible Interface Memory Mapped (AXI-MM) protocol [143]. Finally, the PL, based on an FPGA, is where the operations $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$ and $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$ together with the SHAKE128 hash function are hardware implemented, based on the results of Table 4. From now on, these operations and functions implemented in hardware will be called blocks, allowing to distinguish operations and functions from the software and hardware implementations. The blocks $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$, $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$, and SHAKE128 are detailed in Subsections **5.4.1**, **5.4.2**, and **5.4.3**, respectively.

It is important to mention that it is not used Direct Memory Access (DMA) technique in this hardware/software co-design implementation of the FrodoKEM because the data exchanged between the software and hardware are in small fractions; consequently, the reduction of time in data exchange when this technique is deployed is relatively small and still requires a considerable amount of hardware resources. Furthermore, it is also important to mention that the implementation of the blocks $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$, $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$, and SHAKE128 are timing constant in order to be secure against timing side-channel attacks.

5.4.1 The Block $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$

The operation $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$, which consumes almost 30% of the total time of the key pair generation, has its matrix-by-matrix multiplication implemented in hardware as the block $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$. The operation of adding matrix $\mathbf{E}$ is performed in software, as it is not an expensive operation and would not bring significant time saving, under the assumption that the transferring time of the matrix $\mathbf{E}$ from PS to PL is taken into account.

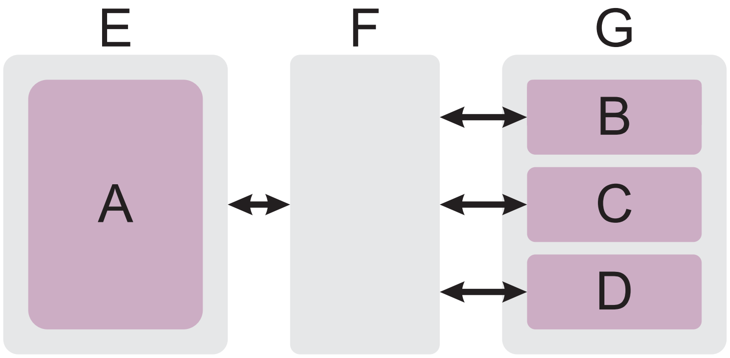
Figure 16 shows the hardware schematic of the block $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$. The schematic consists of four main instances: BRAMs of the matrix $\mathbf{A}$, BRAMs of the matrix $\mathbf{S}$, a multiplier instance, and BRAMs of the matrix $\mathbf{AS}$. Note that BRAMs are where more data are stored.

As mentioned earlier, matrix $\mathbf{A}$ is generated on the fly to save memory resources. Four rows are generated and transmitted from the PS to the BRAMs of the matrix $\mathbf{A}$, located in PL, to be stored. Four 320 × 32-bits BRAMs are ready to receive these rows. Each generated row is composed of 640 × 16-bits. To speed up the data transfer process, 32-bits are transferred at a time, which means that two subsequent values are concatenated and stored in the BRAMs. The matrix $\mathbf{S}$ is transmitted from PS to PL. The whole matrix is transferred since it was fully generated in PS. The transmission process follows the same 32-bits transmit principle as with matrix $\mathbf{A}$, although, for the matrix $\mathbf{S}$ a 2560 × 32-bits BRAM is used.

The multiplication process begins when both matrices are completely stored in their BRAMs. From BRAMs of the matrix $\mathbf{A}$ the n-element of each BRAM is read. As 2 × 16-bits words are stored concatenated in a 32-bits BRAM position, eight values are loaded from memory and sent to the multiplier instance. In parallel, the n-element of the matrix $\mathbf{S}$ is also read. For the same reason as the concatenated storage, 2 × 16-bits values are loaded and transferred to the multiplier instance.

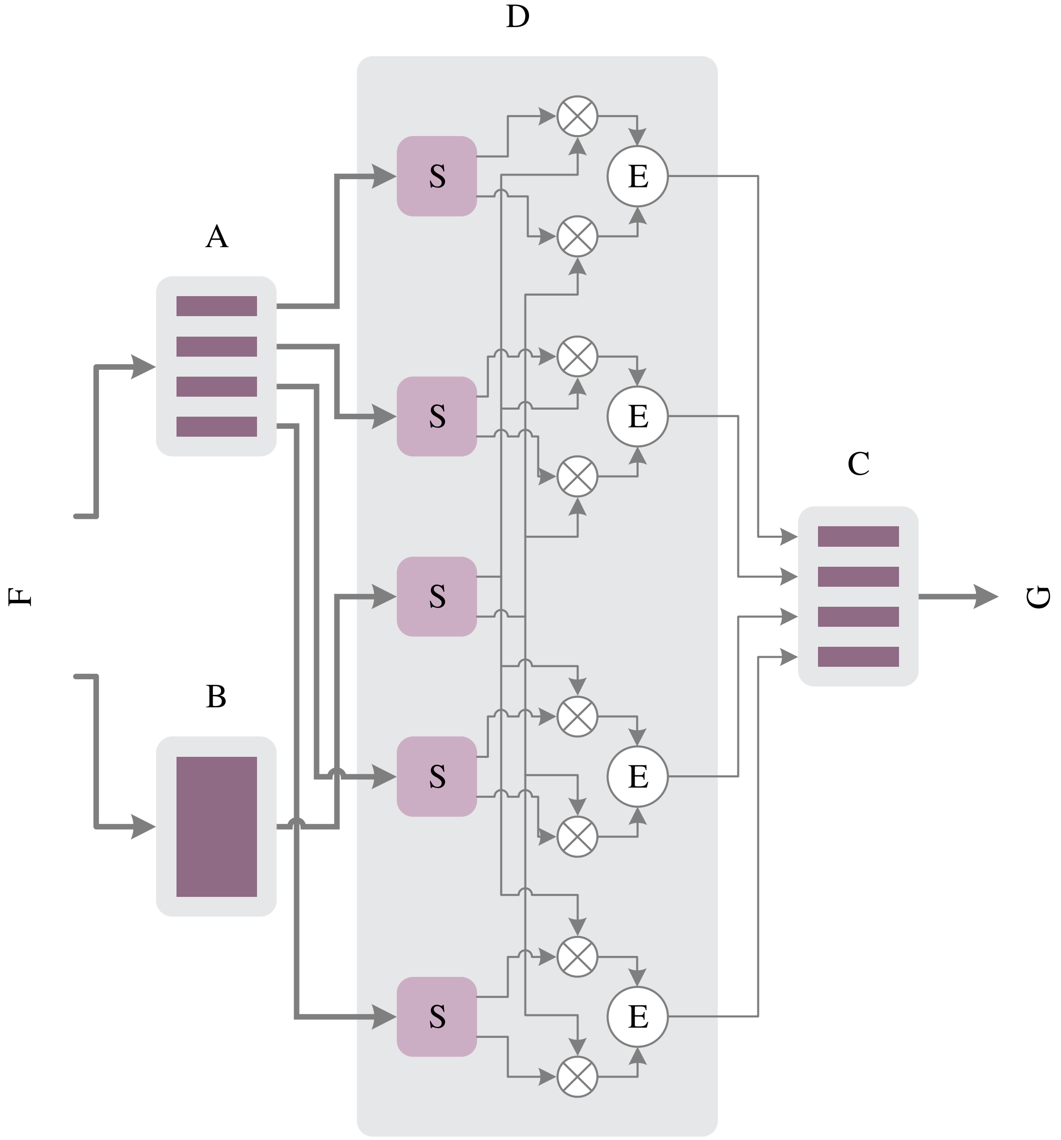
In the multiplication instance, each 32-bits element is split into 2 × 16-bits words using the function S. Then, the multiplications are properly performed, and their results are added to the previous iteration results. Any concern about overflow issues is needed, which is an advantage of the FrodoKEM scheme in saving hardware resources. While the multiplication

Figure 15 – Schematic representation of the structure. The arrows represent 32-bit buses. Control signals have been omitted.



Source: Personal collection.

Figure 16 – Schematic representation of block **AS** ← **A** × **S**. Thick arrows represent 32-bits buses and thin arrows 16-bits buses. Control signals have been omitted.



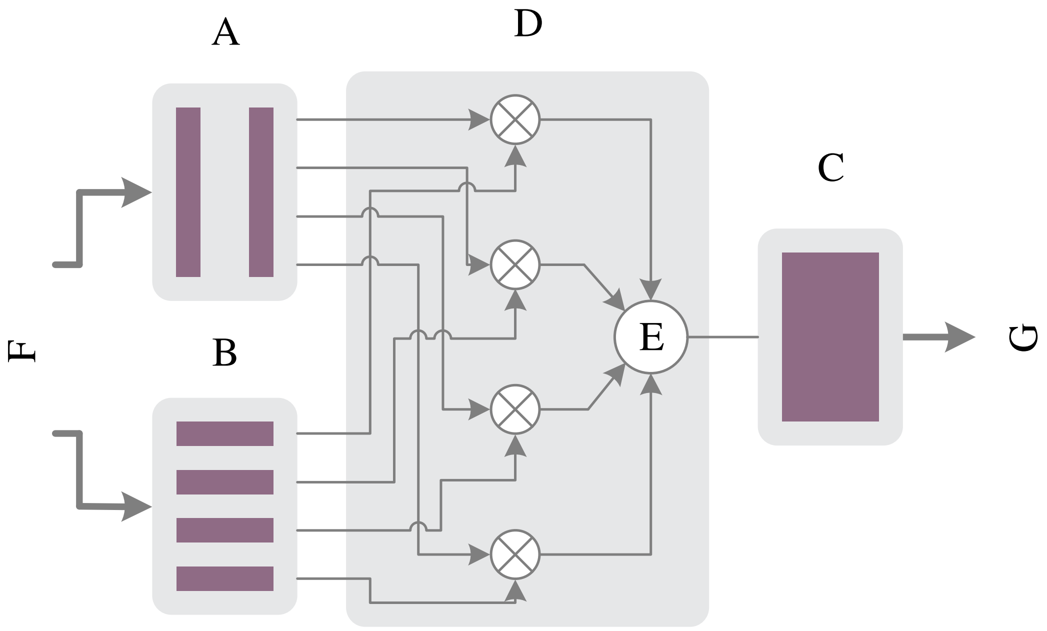
Source: Personal collection.

is performed, the following values from the BRAMs are loaded, keeping the pipeline full to achieve maximum performance. When all elements of the BRAMs are loaded and processed, the 4 × 16-bits accumulated values are transferred to the BRAMs of the matrix **AS**, which has four 1280 × 16-bits BRAMs.

The next four matrix rows **A** must be generated and transferred to PL. The matrix **S** has already been completely transferred in the first iteration and should not be sent again. The next iteration begins, and the process mentioned above is performed again. When all iterations are complete, the BRAMs of the matrix **AS** store the **AS** ← **A** × **S** result and transfers it to PS using a 32-bits bus.

Note that in this case four rows of matrix **A** are generated at once, but it is flexible. However, there is a compromise between memory usage and performance. If more rows are

Figure 17 – Schematic representation of $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$. Thick arrows represent 32-bits buses and thin arrows 16-bits buses. Control signals have been omitted.



Source: Personal collection.

generated at once, more memory would be required, but the processing time would be reduced. On the other hand, if less rows are generated at once, less memory would be required, but the processing time would increase.

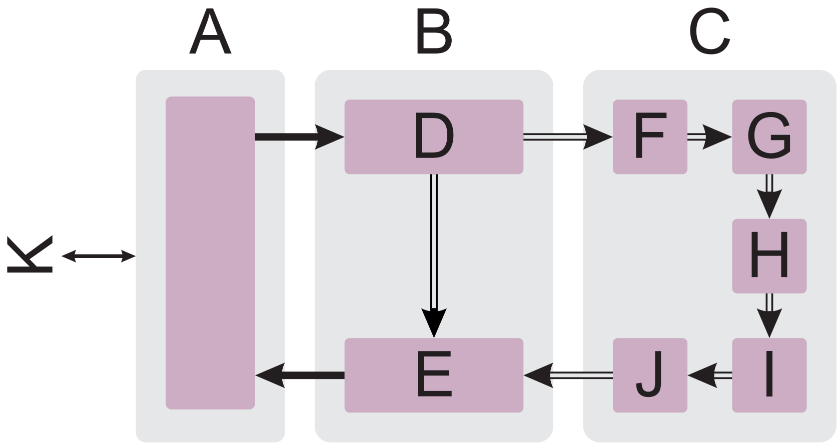
5.4.2 The Block $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$

This operation is responsible for more than 45% of the execution time of encapsulation and decapsulation. The operation $\mathbf{B'} \leftarrow \mathbf{S'A} + \mathbf{E'}$ also has its matrix-by-matrix multiplication implemented in hardware as block $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$. Similarly to the operation $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$, the addition operation of the matrix $\mathbf{E'}$ is carried out in software due to the same reasons. In Figure 17 it can be seen that the schematic representation of the block $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$. It also has four instances: BRAMs of the matrix $\mathbf{S'}$, BRAMs of the matrix $\mathbf{A}$, a multiplier instance, and BRAM of the matrix $\mathbf{S'A}$.

Two 640×32-bits BRAMs are used to store half of the matrix $\mathbf{S'}$. As mentioned earlier, concatenated values are transferred from PS to PL using 32-bits bus; therefore, each BRAM can be split in half (upper 16-bits and lower 16-bits), storing two concatenated word by word columns of the matrix $\mathbf{S'}$, giving a total of four columns. On the other hand, the matrix $\mathbf{A}$ and its transfer process have exactly the same configuration as used in the block $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$: four 320×32-bits BRAMs and 32-bits bus for transfer.

After the matrices are received, multiplication is performed. The n-element of each matrix $\mathbf{S'}$ BRAM are loaded. Since 2×16-bits words are stored concatenated in a 32-bits BRAM position, four values are read and sent to the multiplier instance in 16-bits buses. The p-element

Figure 18 – Block diagram representation of the block SHAKE128. Thin lines represent 32-bits buses, thick lines represent 64-bits buses, and double lines represent 1600-bits buses. Control signals have been omitted.



Source: Personal collection.

of each BRAM of the matrix **A** are read in parallel. Only the upper or lower 16-bits of the four loaded values are sent to the multiplier instance, depending on the p parity.

Therefore, the multiplier instance receives 8 × 16-bits values, four from each matrix. Finally, the values are correctly multiplied, and the results are added, resulting in a 16-bits value. Immediately, the multiplication and addition process results are sent to matrix $\mathbf{S'A}$, stored in one 5120 × 16-bits BRAM, to be added to a previous value stored in a particular position. Simultaneously, new values from matrices $\mathbf{S'}$ and **A** are loaded to keep the pipeline full, restarting the process.

When all iterations are completed, the other half of the matrix $\mathbf{S'}$ and new rows of the matrix **A** must be transferred to PL. When the process ends, the BRAMs of the matrix $\mathbf{S'A}$ will store the result of operation $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$ and it is transmitted to PS using a 32-bits bus.

5.4.3 Block SHAKE128

FrodoKEM's most time-consuming operation is the SHAKE128 hash function, which is responsible for almost 58% of the total execution time of the key pair generation and around 43% of the total execution time of encapsulation and decapsulation. The proposed scheme can be organized into three main instances: a BRAM, the sponge construction, and a Keccak-f[1600], as illustrated in Figure 18.

The BRAM is a 2583 × 64-bits memory. This size was chosen based on the maximum size that FrodoKEM needs. This BRAM is responsible for storing the input values (message) received by the 32-bits bus. Each 32-bits word is a concatenation of 4 × 8-bits characters. 2 × 32-bits words or 8 × 8-bits characters are concatenated and stored in the BRAM.

When all values are received and stored in the BRAM, 168-bytes are sent to the block sponge construction instance via a 64-bits bus, starting the absorb phase. These bytes built

Table 5 – Hardware resource usage.

Block name	Slice LUTs	Slice Registers	Block RAM	DSP
$\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$	357	272	10	8
$\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$	264	301	6	4
SHAKE128	5387	3648	7.5	0
Interconnect	1452	1105	0	0
Others	1097	3295	0	0
Total	8557	8621	23.5	12

the internal state. If necessary in the absorb phase, the Keccak-f[1600] instance is called, and the entire internal state is sent to the Keccak-f[1600] instance, which performs its five steps (θ, ρ, π, χ, and ι) in a single clock. Then, the new scrambled internal state returns to the sponge construction instance. Next, another 168-bytes are loaded from BRAM and the process is repeated until the entire inputted message is read, finalizing the absorb phase.

When the absorb phase ends, the squeeze phase starts using the Keccak-f[1600] instance. To save resources, the first 168-bytes of the internal state of each step in the squeeze phase are stored in the same BRAM, which previously stored the inputted message and now stores the output values (cipher). When the desired output length is reached, the process is complete, and the BRAM uses the 32-bits bus to send the stored output values back to PS.

5.5 PERFORMANCE EVALUATION

The hardware resource and timing analyses of the FrodoKEM scheme are presented in this section. The focus is comparative analyses between **Implementations #1** and **#2** (i.e., software implementations), briefly described in Section 5.3, and the detailed **Implementation #3** (i.e., hardware/software co-design implementation), presented in Section 5.4. In this sense, this section is organized as follows: Subsection **5.5.1** details the hardware resources of the functions implemented in the PL and, in the sequel, Subsection **5.5.2** focuses on the timing analysis of the blocks $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$ and $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$, and the SHAKE128 hash function with and without hardware acceleration.

5.5.1 Hardware Resource Analysis

The hardware resource usage of blocks $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$, $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$, and SHAKE128 are listed in Table 5. It shows that the blocks $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$ and $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$ use very few Slice LUTs and Slice Registers in comparison to the block SHAKE128. It relies on the logic circuit complexity necessary to implement the matrix-by-matrix multiplications, which is much simpler than the one required to implement the block SHAKE128.

The block $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$ uses 10 BRAMs, while the block $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$ uses only 6 BRAMs. This difference occurs because the former stores the entire matrix $\mathbf{S}$ in the PL, while

Table 6 – The average execution time of FrodoKEM for different levels of hardware implementations with 5000 simulations for each implementation, in ms.

Scheme	Algorithm	**Implementation #1**	**Implementation #2**	**Implementation #3**	Relative Time Improvement ($\alpha^f_{\mathrm{TI}_1}$)	Relative Time Improvement ($\alpha^f_{\mathrm{TI}_2}$)
FrodoKEM-640	Key Pair Generation	1387.73	467.25	146.35	86.80%	68.15%
	Encapsulation	1432.16	644.74	205.11		
	Decapsulation	1423.46	645.35	208.26		
	Total	4243.35	1757.33	559.72		

the latter stores only half of it. The block SHAKE128 uses 7.5 BRAMs, enough memory space to store the larger cipher required by FrodoKEM. Finally, the block $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$ uses 8 DSP blocks, which is in accordance with Figure 16. Contrastingly, the block $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$ uses only 4 DSP blocks, as presented in Figure 17. The block SHAKE128 does not use any DSP block.

5.5.2 Timing Analysis

To compare the performance of the FrodoKEM scheme with and without hardware acceleration, the following analyses are considered: in Subsection *5.5.2.1* a time comparison between **Implementations #1**, **#2**, and **#3** is presented, evaluating the time required to process the key pair generation, encapsulation, and decapsulation. Subsection *5.5.2.2* compares **Implementations #1** and **#2** with **Implementation #3** of the functions listed in Table 4 in terms of performance. Finally, Subsection *5.5.2.3* provides a detailed analysis of the aforementioned functions listed in Table 4, comparing the time required to process data against the time that is spent transferring data between PS and PL. Note that each run time measurement was obtained using a high-resolution timer in order to provide good accuracy.

5.5.2.1 Comparison between the implementations

This subsection compares **Implementations #1**, **#2**, and **#3** of the FrodoKEM 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 of each implementation.

We can see in Table 6 that, as expected, **Implementation #3** considerably outperforms **Implementations #1** and **#2**. The former required 559.72 ms to perform the key pair generation, encapsulation, and decapsulation, while **Implementation #1** required 4243.32 ms and **Implementation #2**, 1757.33 ms. The relative time improvement ($\alpha^f_{\mathrm{TI}_1}$) between the **Implementation #1** and **Implementation #3** and relative time improvement ($\alpha^f_{\mathrm{TI}_2}$) between the **Implementation #2** and **Implementation #3** are given by

$$\alpha^f_{\mathrm{TI}_1} = 1 - \left(\frac{T^f_{\mathrm{TET}_3}}{T^f_{\mathrm{TET}_1}} \right), \tag{5.1}$$

and

$$\alpha^{f}_{\mathrm{TI}_2} = 1 - \left(\frac{T^{f}_{\mathrm{TET}_3}}{T^{f}_{\mathrm{TET}_2}}\right), \tag{5.2}$$

where $T^{f}_{\mathrm{TET}_1}$, $T^{f}_{\mathrm{TET}_2}$, and $T^{f}_{\mathrm{TET}_3}$ are the total execution time of **Implementations #1**, **#2**, and **#3**, respectively. The $\alpha^{f}_{\mathrm{TI}_1}$ achieved are 86.90%, while the $\alpha^{f}_{\mathrm{TI}_2}$ obtained are 68.15% for the FrodoKEM-640 scheme.

5.5.2.2 Execution Time Analysis

This subsection intends to individually analyze the blocks $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$, $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$, and SHAKE128 (i.e., without considering its impact on the whole implementation). To carry out this analysis, it is assumed that all routines necessary to generate matrices **B** and **B′** are considered, not just the matrix-by-matrix multiplication. Table 7 lists the attained results related to these three blocks, presenting their average execution time using **Implementations #1**, **#2**, and **#3** as $T^{f}_{\mathrm{AET}_1}$, $T^{f}_{\mathrm{AET}_2}$, and $T^{f}_{\mathrm{AET}_3}$, respectively, which are expressed as

$$T^{f}_{\mathrm{AET}_1} = \sum_{1}^{N_I} \frac{T^{f}_{\mathrm{TET}_1}, i}{N_I}, \tag{5.3}$$

$$T^{f}_{\mathrm{AET}_2} = \sum_{1}^{N_I} \frac{T^{f}_{\mathrm{TET}_2}, i}{N_I}, \tag{5.4}$$

and

$$T^{f}_{\mathrm{AET}_3} = \sum_{1}^{N_I} \frac{T^{f}_{\mathrm{TET}_3}, i}{N_I}, \tag{5.5}$$

where $(T^{f}_{\mathrm{TET}_1}, i)$, $(T^{f}_{\mathrm{TET}_2}, i)$, and $(T^{f}_{\mathrm{TET}_3}, i)$ are the total execution time using **Implementations #1**, **#2**, and **#3**, 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 $T^{f}_{\mathrm{AET}_3}$. Finally, the relative time improvement $\beta^{f}_{\mathrm{TI}_1}$ between the **Implementations #1** and **#3**, and relative time improvement $\beta^{f}_{\mathrm{TI}_2}$ between the **Implementations #2** and **#3**, presented in Table 7, are given by

$$\beta^{f}_{\mathrm{TI}_1} = 1 - \left(\frac{T^{f}_{\mathrm{AET}_3}}{T^{f}_{\mathrm{AET}_1}}\right). \tag{5.6}$$

and

$$\beta^{f}_{\mathrm{TI}_2} = 1 - \left(\frac{T^{f}_{\mathrm{AET}_3}}{T^{f}_{\mathrm{AET}_2}}\right). \tag{5.7}$$

Table 7 shows that blocks $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$ and $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$ achieved similar average execution time (i.e., $T^{f}_{\mathrm{AET}_1}$, $T^{f}_{\mathrm{AET}_2}$, and $T^{f}_{\mathrm{AET}_3}$), independently from the implementation used. Consequently, the obtained relative time improvement $\beta^{f}_{\mathrm{TI}_1}$ and $\beta^{f}_{\mathrm{TI}_2}$ are also similar. $\beta^{f}_{\mathrm{TI}_1}$ and

Table 7 – Execution time analysis, in ms.

Algorithm	Average Execution Time			Relative Time Improvement $(\beta^f_{\mathrm{TI}_1})$	Relative Time Improvement $(\beta^f_{\mathrm{TI}_2})$
	Implementation #1 $(T^f_{\mathrm{AET}_1})$	**Implementation #2** $(T^f_{\mathrm{AET}_2})$	**Implementation #3** $(T^f_{\mathrm{AET}_3})$		
$\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$	1266.289	447.356	136.424	89.22%	69.50%
$\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$	1362.034	616.859	189.652	86.07%	69.25%
SHAKE128	13.800	3.834	0.641	95.35%	83.26%

$\beta^f_{\mathrm{TI}_2}$ obtained by block $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$ are 89.22% and 69.50%, respectively, while for block $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$, 86.07% and 69.25%, also respectively.

Regarding the SHAKE128 hash function, the average execution time is considerably lower, which is expected as this function requires much less processing than blocks $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$ and $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$. Furthermore, it can be seen that the relative time improvement $\beta^f_{\mathrm{TI}_1}$ and $\beta^f_{\mathrm{TI}_2}$ are higher when compared with the ones obtained by the matrix-by-matrix operations. The $\beta^f_{\mathrm{TI}_1}$ and $\beta^f_{\mathrm{TI}_2}$ obtained by the SHAKE128 hash function are 95.35% and 83.26%, respectively. This greater improvement can be explained by the looped-based implementation in software (i.e., **Implementations #1** and **#2**) in contrast with an efficient hardware implementation (i.e., **Implementation #3**).

5.5.2.3 Hardware processing time analysis

Another meaningful analysis is the hardware-only time analysis. This analysis focused on comparing each routine's execution time and processing time, both in hardware. To carry out this analysis, in the blocks $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$ and $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$, only the matrix-by-matrix multiplication is considered because it is the part performed in hardware. Due to that, the $T^f_{\mathrm{AET}_3}$ presented in Table 7 is not equal to the T^f_{AET} presented in Table 8 as the former takes into account the summation of matrices $\mathbf{E}$ and $\mathbf{E}'$. The T^f_{AET} can be separated into two components, and it is given by

$$T^f_{\mathrm{AET}} = T^f_{\mathrm{APT}} + T^f_{\mathrm{DTT}}, \tag{5.8}$$

where the average processing time in hardware (T^{kf}_{APT}) is the time required to process the data, and the data transfer time (T^f_{DTT}) is the time required to transfer data to/from the PS. This analysis is numerically evaluated by using the so-called relative time (γ^f_T) parameter, which is given by

$$\gamma^f_{\mathrm{T}} = \frac{T^f_{\mathrm{APT}}}{T^f_{\mathrm{AET}}}, \tag{5.9}$$

where T^f_{APT} is the average processing time in hardware per iteration and T^f_{AET} is the average execution time in hardware per iteration of the aforementioned blocks. The attained results are presented in Table 8.

Table 8 – Hardware-only execution and processing time, in ms.

Operation	Average execution time in hardware (T^f_{AET})	Average processing time in hardware (T^f_{APT})	Relative time (γ^f_T)
$\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$	58.61	4.094	6.98%
$\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$	62.71	8.190	13.06%
SHAKE128	0.129	0.005	4.07%

As can be seen, the processing time in the hardware of blocks $\mathbf{AS} \leftarrow \mathbf{A}\times\mathbf{S}$, $\mathbf{S'A} \leftarrow \mathbf{S'}\times\mathbf{A}$, and SHAKE128 are much shorter than the execution time, representing 6.98%, 13.06%, and 4.07%, respectively. These results mean that most of the time is spent receiving data from PS or sending data back to PS while only a short time is dedicated to performing the operations in hardware.

5.6 SUMMARY

This chapter has dedicated attention to implementing the FrodoKEM scheme in hardware. In this context, the FrodoKEM, a post-quantum lattice-based scheme, was detailed and, in the sequel, implemented using **Implementations #1**, **#2**, and **#3**. Evaluating the performance of all functions for **Implementations #1** and **#2** of the FrodoKEM scheme, it has been verified that three functions, listed in Table 4, must be hardware implemented (i.e., **Implementation #3**) to achieve a lower execution time. Furthermore, as two of these functions are mainly based on multiplications, the use of DSP presented in the hardware significantly accelerates the process. In this sense, numerical results have shown that the hardware/software co-design implementation (i.e., **Implementation #3**) of the most time-consuming and complex routines of the FrodoKEM in hardware results in a one-sixth and one-third reduction of the execution time compared to the **Implementations #1** and **#2**, respectively. According to the execution time analysis, three operations consume almost all execution time. The **Implementation #3** of these three operations in blocks reached an improvement of 86.80% and 68.15% (i.e., the execution time reduces from 4.2 seconds and 1.7 seconds to less than 0.6 seconds) when compared to **Implementations #1** and **#2**, respectively. Since the FrodoKEM scheme is a conservative quantum-resistant scheme based on standard lattices with a not-so-good timing performance, hardware acceleration is an alternative to meet timing constraints that might be required in AMIs. On the other hand, due to its strong cryptographic primitives, it is a good candidate for improving security in AMI under classic and quantum computers. Overall, this chapter has shown that the discussed implementations of the FrodoKEM scheme are suitable for QR-DCMs.

6 THE CRYSTALS-KYBER SCHEME

The CRYSTALS-Kyber scheme is the winner of the NIST PQC standardization process and relies on the M-LWE problem. As already deeply discussed in Subsection *4.3.2.2*, the M-LWE problem tries to be a trade-off between security and performance because it is less structured than R-LWE and scales better. In this sense, cryptanalytic progress has shown that an attack against the M-LWE problem is less likely than on the R-LWE problem [132]. Therefore, schemes based on M-LWE (e.g., CRYSTALS-Kyber) present much better timing performance than schemes based on LWE (e.g., FrodoKEM) and are also, at least in theory, less likely to be broken than schemes based on R-LWE. In a QR-AMI context, where hardware-constrained devices are deployed, such as QR-DCM, schemes that perform well and are based on lightweight approaches are of utmost importance.

In this sense, this chapter investigates the feasibility of the CRYSTALS-Kyber scheme as the KEM for ensuring quantum-resistant symmetric key exchange between nodes in the AMI based on **Approach #3**[1]. 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 PQC standardization process [144] and also the winner of the contest. In this regard, implementations of the CRYSTALS-Kyber scheme using **Implementations #1**, **#2**, and **#3**[2] are detailed. The main contributions of this chapter are as follows:

- 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 considering a hardware-constrained QR-DCM based on an SoC FPGA device. Moreover, a presentation of an optimized hardware/software co-design implementation (i.e., **Implementation #3**) of the CRYSTALS-Kyber scheme on a hardware-constrained QR-DCM that uses an SoC FPGA device.
- A performance comparison between **Implementation #3** and **Implementations #1** and **#2** of the CRYSTALS-Kyber scheme in terms of execution time. Also, an analysis of the hardware resource usage demanded by the proposed hardware/software co-design implementation (i.e., **Implementation #3**) of the CRYSTALS-Kyber scheme on an SoC FPGA device.

The rest of this chapter is organized as follows: in Section 6.1 it is detailed the CRYSTALS-Kyber scheme; in Section 6.2 it is addressed the software implementation (i.e., **Implementations #1** and **#2**) and in Section 6.3 the hardware/software co-design implementation (i.e., **Implementation #3**) of this scheme; in Section 6.4 it is discussed numerical results of

[1] The investigation provided in this chapter focused on **Approach #3**, but the results are also valid for **Approaches #1** and **#2**.

[2] Descriptions of these three type of implementations are in Section 5.2.

Table 9 – Parameter values for the CRYSTALS-Kyber scheme.

CRYSTALS-Kyber	n	k	q	η_1	η_2	d_u	d_v	Public key size (bytes)	Private key size (bytes)	Ciphertext size (bytes)
512	256	2	3329	3	2	10	4	800	1632	768
768	256	3	3329	2	2	10	4	1184	2400	1088
1024	256	4	3329	2	2	11	5	1568	3168	1568

hardware resource usage, execution time, and performance comparison between implementations; and, finally, in Section 6.5 it is asserted conclusive remarks.

6.1 BACKGROUND OF THE CRYSTALS-KYBER SCHEME

The CRYSTALS-Kyber scheme [127, 132], relying on the M-LWE problem (already discussed in Chapter 4), was designed seeking high-performance (e.g., as presented by NewHope) without losing flexibility (e.g., as FrodoKEM). In this regard, 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 PQC standardization process in the PKE/KEM category.

6.1.1 The CRYSTALS-Kyber scheme as a Key Encapsulation Mechanism

The CRYSTALS-Kyber scheme [127] can be used as PKE and KEM. PKE is based on the PKE key pair generation, encryption, and decryption algorithms, described in Algorithms #6, #7, and #8, respectively. On the other hand, KEM, which is constituted by KEM key pair generation, encapsulation, and decapsulation, as described in Algorithms #9, #10, and #11 respectively, is constructed via a slightly modified Fujisaki-Okamoto transform [145], which is based on the PKE. The values of the parameters used by Algorithms #6 to #11 are specified in Table 9.

Note that the parameter $n = 256$ means 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. Compared with the parameters of FrodoKEM presented in Table 2, the size of the public key, private key, and ciphertext are considerably smaller. More details about the choice of these parameters are shown in [127].

These key pair generation, encryption, and decryption algorithms call a few functions, detailed in [127]. 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

Algorithm #6: PKEKeyPairGeneration()

```
Input: None
Output:
Public key: $pk \in \mathcal{B}^{12kn/8+32}$
Private 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
        $\hat{\mathbf{A}}[i][j] :=$ Parse(XOF($\rho, j, i$))
    end
end
▹ Sample $\mathbf{s} \in R_q^k$ from $B_{\eta_1}$:
for i from $0 : k - 1$ do
    $\mathbf{s}[i] :=$ CBD$_{\eta_1}$(PRF($\rho, N$))
    $N := N + 1$
end
▹ Sample $\mathbf{e} \in R_q^k$ from $B_{\eta_1}$:
for i from $0 : k - 1$ do
    $\mathbf{e}[i] :=$ CBD$_{\eta_1}$(PRF($\rho, N$))
    $N := N + 1$
end
$\hat{\mathbf{s}} :=$ NTT($\mathbf{s}$)
$\hat{\mathbf{e}} :=$ NTT($\mathbf{e}$)
$\hat{\mathbf{t}} := \hat{\mathbf{A}} \circ \hat{\mathbf{s}} + \hat{\mathbf{e}}$
Return:
Public key $pk :=$ ( Encode$_{12}$($\hat{\mathbf{t}}$ mod$^+q$)) $\|$ $\rho$
Private key $sk :=$ Encode$_{12}$($\hat{\mathbf{s}}$ mod$^+q$)
```

distribution. The **NTT**$(\cdot)$ function performs multiplications in R_q in a very efficient way, and the function **NTT**$^{-1}(\cdot)$ performs the inverse of the **NTT**$(\cdot)$. The **Encode**$(\cdot)$ function serializes a polynomial into a byte array, while the **Decode**$(\cdot)$ function does the opposite. Moreover, the **Compress**$_q(\cdot)$ function takes an element from $\mathbb{Z}_q$ and outputs an integer in the set $\{0, ..., 2^q - 1\}$, where $d < \lceil \log_2(q) \rceil$, and the **Decompress**$_q(\cdot)$ function does the opposite.

The main part of the key pair generation of the KEM (Algorithm #9) is the generation of the public key *pk* and the private key *sk*, derived from the key pair generation (Algorithm #6) which performs the $pk = \mathbf{As} + \mathbf{e}$ operation, using the public matrix **A** and the private vector **s**.

The encapsulation (Algorithm #10) uses the public key *pk* together with the encryption algorithm (Algorithm #7) to generate a shared secret key K (i.e., a symmetric key) and a ciphertext c. More specifically, the encryption algorithm regenerates the matrix **A** using the public key *pk*, samples **r**, $\mathbf{e}_1$, and e_2, and performs $\mathbf{u} = \mathbf{A}^T\mathbf{r} + \mathbf{e}_1$ and $v = \mathbf{t}^T\mathbf{r}+e_2 + \textbf{Decompress}_q(m, 1)$, where

Algorithm #7: 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:
 $N := 0$
 $\hat{\mathbf{t}} := \mathbf{Decode}_{12}(pk)$
 $\rho := pk + 12kn/8$
▹ 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 $B_{\eta_1}$:
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 $B_{\eta_2}$:
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 $B_{\eta_2}$:
$e_2 := \mathbf{CBD}_{\eta_2}(\mathbf{PRF}(r, N))$
$\hat{\mathbf{r}} := \mathbf{NTT}(\mathbf{r})$
$\mathbf{u} := \mathbf{NTT}^{-1}(\hat{\mathbf{A}}^T \circ \hat{\mathbf{r}}) + \mathbf{e}_1$
$v := \mathbf{NTT}^{-1}(\hat{\mathbf{t}} \circ \hat{\mathbf{r}}) + e_2 +$
      $\mathbf{Decompress}_q(\mathbf{Decode}_1(m), 1)$
$c_1 := \mathbf{Encode}_{d_u}(\mathbf{Compress}_q(\mathbf{u}, d_u))$
$c_2 := \mathbf{Encode}_{d_v}(\mathbf{Compress}_q(v, d_v))$
Return:
Ciphertext $c := (c_1 \| c_2)$
```

m is a random message generated in the encapsulation algorithm (i.e. Algorithm #10).

Finally, the decapsulation (Algorithm #11) uses the private key *sk*, ciphertext c, encryption (Algorithm #7), and decryption (Algorithm #8) to obtain the same shared secret key K previously generated in the encapsulation algorithm. In the decapsulation algorithm, given the private key *sk* and the ciphertext c, m' is computed by m' = **Decryption**(*sk*,*c*), and c' is the encryption of m'. In possession of c and c', the shared secret key K is computed.

The private 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 it possible to recover c'. In

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

```
Algorithm #9: KEMKeyPairGeneration()
    Input: None
    Output:
    Public key: $pk \in \mathcal{B}^{12kn/8+32}$
    Private key: $sk \in \mathcal{B}^{24kn/8+96}$
    Procedure:
    Initialization:
    $z \leftarrow \mathcal{B}^{32}$
    $(pk, sk') :=$ PKEKeyPairGeneration()
    $sk' :=$ Encode$_{12}$($\hat{\mathbf{s}}$ mod$^+ q$)
    Return:
    Public key $pk$
    Private key $sk := (sk' \| pk \| \mathbf{H}(pk) \| z)$
```

possession of c and c', generating the shared secret key K is possible. The security proofs of the CRYSTALS-Kyber scheme can be found in [127].

6.2 SOFTWARE IMPLEMENTATION

This section presents the software implementation of the CRYSTALS-Kyber scheme on the ARM Cortex-M4F microcontroller (i.e., **Implementation #1**) and ARM Cortex-A9 processor (**Implementation #2**). To detail these implementations, this section is organized as follows: Subsection **6.2.1** discusses a preliminary analysis of the CRYSTALS-Kyber scheme implemented on the ARM Cortex-M4F microcontroller and ARM Cortex-A9 processor, highlighting the most time-consuming functions that ought to be hardware implemented; and Subsection **6.2.2** provides additional details about the software implementation of high-level functions that must be considered to come up with a hardware/software co-design implementation.

6.2.1 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 [124] was executed using the ARM Cortex-M4F microcontroller and ARM Cortex-A9 processor

Algorithm #10: 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 secret key: $K \in \mathcal{B}^*$
Procedure:
Initialization:
$m \leftarrow \mathcal{B}^{32}$
$m \leftarrow \mathbf{H}(m)$
$(\bar{K}, r) := \mathbf{G}(m \parallel \mathbf{H}(pk))$
$c := \mathbf{Encryption}(pk, m, r)$
Return:
Ciphertext $c := (c_1 \parallel c_2)$
Shared secret key $K := \mathbf{KDF}(\bar{K} \parallel \mathrm{H}(c))$
```

(i.e., **Implementations #1** and **#2**, respectively). Table 10 summarizes the relative time-consuming of the functions used by the CRYSTALS-Kyber-512 scheme in the ARM Cortex-M4F microcontroller. Similar results were achieved for the CRYSTALS-Kyber-768 and CRYSTALS-Kyber-1024 schemes and also using the ARM Cortex-A9 processor; however, for simplicity, they are omitted.

Table 10 shows that **NTT**$(\cdot)$, **NTT**$^{-1}(\cdot)$, **Muliply and Accumular**$(\cdot)$, **Keccak-**$f(\cdot)$, **Tomont**$(\cdot)$, and **Reduce**$(\cdot)$ 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

Algorithm #11: Decapsulation()

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

Table 10 – The relative time execution, in percentage, of the high-level functions for the CRYSTALS-Kyber-512 based on **Implementation #1**.

Function	Key Pair Generation	Decapsulation	Encapsulation
NTT$(\cdot)$	24.80%	11.11%	19.54%
NTT$^{-1}(\cdot)$	-	25.88%	30.23%
Multiply and Accumulate$(\cdot)$	18.37%	24.47%	28.19%
Keccak-f**[1600]**$(\cdot)$	8.90%	9.18%	6.82%
Tomont$(\cdot)$	2.18%	-	-
Reduce$(\cdot)$	1.59%	1.70%	1.35%
Others	44.16%	27.66%	13.87%

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 10. Apart from the **Keccak-**f**[1600]**$(\cdot)$, the high-level functions are mainly used to perform matrix-by-vector multiplication, **NTT**$(\cdot)$, and **NTT**$^{-1}(\cdot)$, (i.e., functions found in Algorithms #6 to #11), 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]**$(\cdot)$ 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 generated (i.e., when **H**$(\cdot)$, **G**$(\cdot)$, **PRF**$(\cdot)$, **XOF**$(\cdot)$, **KDF**$(\cdot)$ functions are called).

6.2.2 Implementation

Relying on the preliminary analysis in Subsection **6.2.1** and without paying attention to the **Keccak-**f**[1600]**$(\cdot)$ function, it can be seen that high-level functions are mainly processed based on three other functions: (i) **Fqmul**$(\cdot)$, (ii) **Montgomery Reduction**$(\cdot)$, and (iii) **Barrett Reduction**$(\cdot)$, which from now on will be categorized as low-level functions. Further analyses of low-level and **Keccak-**f**[1600]**$(\cdot)$ functions allow us to understand the necessity of their hardware implementation.

Starting with the **Fqmul**$(\cdot)$ function, it can be said that it receives two values, multiplies them, and sends the result to be reduced by the **Montgomery Reduction**$(\cdot)$ function. Finally, the reduced value is the output of the **Fqmul**$(\cdot)$ function. As can be seen, the **Fqmul**$(\cdot)$ 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**$(\cdot)$ function, it may take a significant amount of time to execute all required operations.

The **Montgomery Reduction**$(\cdot)$ 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;

Algorithm #12: Montgomery Reduction$(\cdot)$

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

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 **Montgomery Reduction**$(\cdot)$ 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 #12 illustrates the **Montgomery Reduction**$(\cdot)$ function.

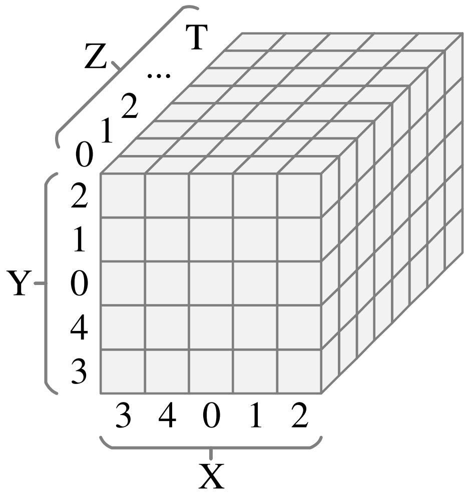
The **Barrett Reduction**$(\cdot)$ function aims to optimize the operation $c = a$ mod q by pre-calculating a constant and assuming that q does not change. This way, it can avoid the slowness of long divisions, which multiplications can replace. Algorithm #13 shows the **Barrett Reduction**$(\cdot)$ function.

The **Keccak-**f[1600]$(\cdot)$ 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 [141]. In the

Algorithm #13: Barrett Reduction$(\cdot)$

```
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:
$t := a - t$
```

Figure 19 – Three-dimensional state array of the **Keccak-**f**[1600](**$\cdot$**)** function.



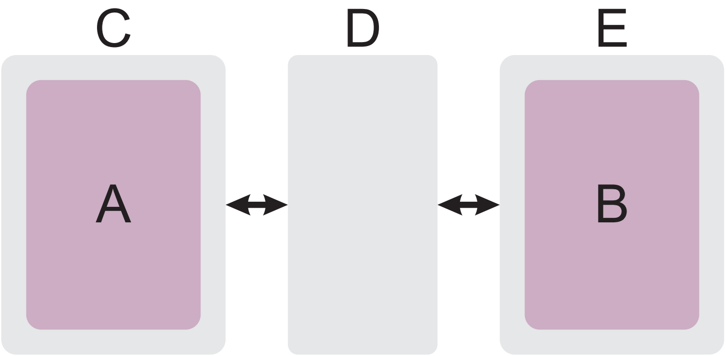
Source: Personal collection.

CRYSTALS-Kyber scheme, the function **H**($\cdot$) actually performs a SHA3-256 hash function and **G**($\cdot$) performs a SHA3-512. On the other hand, **XOF**($\cdot$) performs SHAKE128, and **PRF**($\cdot$) and **KDF**($\cdot$) perform SHAKE256. The functions mentioned above make use of the **Keccak-**f**[1600](**$\cdot$**)** function, which is detailed in the following paragraphs.

The **Keccak-**f**[1600](**$\cdot$**)** 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](**$\cdot$**)** function. Consequently, 1600 in the **Keccak-**f**[1600](**$\cdot$**)** function means that there are 1600 bits in the state array, consequently, z must be equals to 64. Figure 19 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 $lane(i, j) = \mathcal{A}[i, j, 0] \parallel ... \parallel \mathcal{A}[i, j, z-1]$, while a column is defined by $column(i, w) = \mathcal{A}[i, 0, w] \parallel ... \parallel \mathcal{A}[i, y-1, w]$.

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 [141]. Due to the absorb and squeezing characteristics of sponge functions, the **Keccak-**f**[1600](**$\cdot$**)** function may be executed a few times until all data is processed.

Figure 20 – The Block diagram for the proposed hardware/software implementation. The arrows represent 32-bit buses. Note that the control signals are omitted.



Source: Personal collection.

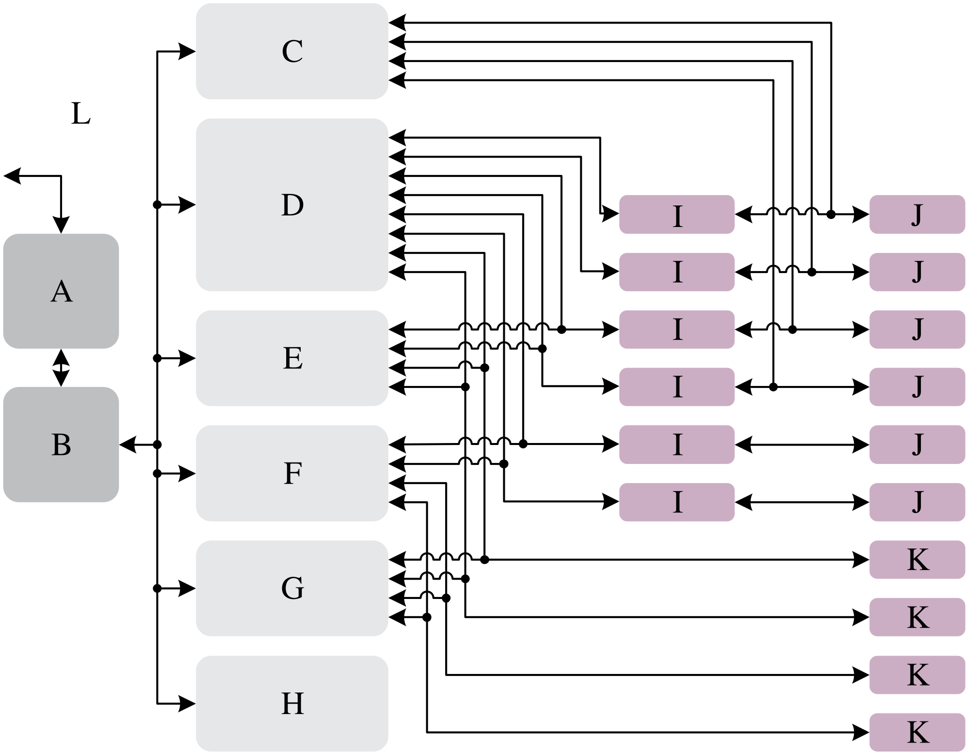
6.3 HARDWARE/SOFTWARE CO-DESIGN IMPLEMENTATION

This section details the proposed hardware/software co-design implementation using the ARM Cortex-A9 together with the FPGA (i.e., **Implementation #3**) of the CRYSTALS-Kyber scheme. Figure 20 shows the block diagram of the proposed hardware/software implementation of the CRYSTALS-Kyber scheme. As the FrodoKEM hardware/software co-design implementation presented in Chapter 5, this implementation can also be divided into three main instances: PS, Interconnect, and PL. The PS is responsible for hosting the ARM Cortex-A9 processor, where the software implementation is placed. The interconnect controls the data exchange between the PS and PL using the AXI-MM [143]. The PL, based on an FPGA device, is where the most time-consuming functions (i.e., functions listed in Table 10) are hardware implemented.

Considering the results in Table 10, it can be seen 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 **6.2.2**, 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 separate hardware blocks, and although all of them can access the low-level functions, they can also be 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 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 21 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.

Figure 21 – Block diagram of the proposed hardware implementation for the CRYSTALS-Kyber scheme. Interface blocks are highlighted in dark gray, high-level blocks in light gray, and low-level blocks in purple. Control signals have been omitted and connections simplified.



Source: Personal collection.

Note that the DMA transfers data to/from the Interconnect 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 can be processed. To accelerate the data transmission between the PS and PL, it is concatenated 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 simultaneously, 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:

- **Tomont block.** It converts 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 **6.2.2**. It also gets the initial state from one of the BRAMs and stores the updated state using the other one. See Subsection **6.2.2** 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; 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 #12 in Subsection **6.2.2**.
- **Barrett Reduction block.** It receives a coefficient to be reduced, which requires two DSP blocks as shown in Algorithm #13, see details in Subsection **6.2.2**.

It is important to mentioned that all blocks implemented in hardware are timing constant in order to avoid any timing side-channel attack.

6.4 PERFORMANCE EVALUATION

This section discusses hardware resource and timing analyses of the implementation of the CRYSTALS-Kyber scheme. The focus is comparative analyses between **Implementations #1** and **#2** (i.e., software implementations), briefly described in Section 6.2, and the detailed **Implementation #3** (i.e., hardware/software implementation), presented in Section 6.3. In this sense, this section is organized as follows: Subsection **6.4.1** presents the hardware resource usage of the blocks implemented in the PL and, in Subsection **6.4.2**, the timing analysis comparing the implementations with and without hardware acceleration.

Table 11 – Hardware resource usage.

Block name	Number of blocks	Slice LUTs	Slice Register	BRAM	DSP
Tomont	1	67	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	122	107	4	0
DMA	1	1354	1955	2	0
Others	-	3312	5190	0	0
Total	-	10276	13301	6.5	30

6.4.1 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 6.3. For this analysis, it does not matter which security level of the CRYSTALS-Kyber scheme is being used 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 11.

Table 11 shows that Keccak-f[1600] and DMA are the blocks that require, as expected, more slice LUTs and slice registers. The former demand more hardware resources to implement the five steps, as explained in Subsection **6.2.2**. 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 6.3.

6.4.2 Timing Analysis

To compare the performance of the CRYSTALS-Kyber scheme with and without hardware acceleration, the following analyses are considered: in Subsection *6.4.2.1* a time comparison between **Implementations #1**, **#2**, and **#3** is presented, evaluating the time required to process the key pair generation, encapsulation, and decapsulation. Subsection *6.4.2.2* compares the high-level functions (using **Implementations #1** and **#2**) and blocks (using **Implementations #3**) in terms of time performance. Finally, Subsection *6.4.2.3* thoroughly analyzes 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 high-resolution timer in order to provide good accuracy.

6.4.2.1 Comparison between the implementations

This subsection focuses on comparing **Implementations #1** and **#2** with the **Implementation #3** 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 of each implementation.

Table 12 shows that when higher security levels are used, the total execution time increases independent of the implementation used. 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 **Implementation #1**, the total execution time to perform the CRYSTALS-Kyber-512, -768, and -1024 schemes are 37.435, 62.558, and 101.858 ms, respectively, while using the **Implementation #2**, 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 total execution time for the same security levels but using the **Implementation #3** are 4.054, 5.857, and 8.337 ms, respectively. The relative time improvement (α_{TI_1}) between the **Implementation #1** and **Implementation #3** and relative time improvement (α_{TI_1}) between the **Implementation #2** and **Implementation #3** are given by

$$\alpha^k_{\text{TI}_1} = 1 - \left(\frac{T^k_{\text{TET}_3}}{T^k_{\text{TET}_1}} \right), \tag{6.1}$$

and

$$\alpha^k_{\text{TI}_2} = 1 - \left(\frac{T^k_{\text{TET}_3}}{T^k_{\text{TET}_2}} \right), \tag{6.2}$$

where $T^k_{\text{TET}_1}$, $T^k_{\text{TET}_2}$, and $T^k_{\text{TET}_3}$ are the total execution time of **Implementations #1**, **#2**, and **#3**, respectively. The $\alpha^k_{\text{TI}_1}$ achieved are 89.17%, 90.63%, and 91.81%, while the $\alpha^k_{\text{TI}_2}$ 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 $\alpha^k_{\text{TI}_1}$ and $\alpha^k_{\text{TI}_2}$ increase 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, $\alpha^k_{\text{TI}_1}$ and $\alpha^k_{\text{TI}_2}$ increase because the processing in the hardware is faster than in software, and this fact compensates for the increase of the communication burden. In the end, higher relative time improvement for higher security levels can be seen.

Table 12 – The average execution time of CRYSTALS-Kyber for different levels of hardware implementations with 5000 simulations for each implementation, in ms.

Scheme	Algorithm	**Implementation #1**	**Implementation #2**	**Implementation #3**	Relative Time Improvement ($\alpha^k_{\text{TI}_1}$)	Relative Time Improvement ($\alpha^k_{\text{TI}_2}$)
CRYSTALS-Kyber-512	Key Pair Generation	9.776	3.286	0.869	89.17%	68.36%
	Encapsulation	13.446	4.465	1.393		
	Decapsulation	14.213	5.062	1.792		
	Total	37.435	12.815	4.054		
CRYSTALS-Kyber-768	Key Pair Generation	16.824	5.436	1.407	90.63%	71.28%
	Encapsulation	22.328	7.103	2.014		
	Decapsulation	23.406	7.858	2.436		
	Total	62.558	20.398	5.857		
CRYSTALS-Kyber-1024	Key Pair Generation	28.630	8.539	2.183	91.81%	72.75%
	Encapsulation	35.653	10.575	2.845		
	Decapsulation	37.575	11.480	3.308		
	Total	101.858	30.595	8.337		

6.4.2.2 Execution Time Analysis

This subsection seeks to analyze only the high-level functions (implemented using the ARM Cortex-M4F and ARM Cortex-A9) and blocks (using the FPGA) individually, ignoring their impact on the whole implementation. Table 13 presents the comparison for the CRYSTALS-Kyber-512. The other security levels of this scheme were omitted since similar results were obtained. The average execution time $T^k_{\text{AET}_1}$, $T^k_{\text{AET}_2}$, and $T^k_{\text{AET}_3}$ of **Implementations #1**, **#2**, and **#3**, respectively, are expressed as

$$T^k_{\text{AET}_1} = \sum_{1}^{N_I} \frac{T^k_{\text{TET}_1}, i}{N_I}, \tag{6.3}$$

$$T^k_{\text{AET}_2} = \sum_{1}^{N_I} \frac{T^k_{\text{TET}_2}, i}{N_I}, \tag{6.4}$$

and

$$T^k_{\text{AET}_3} = \sum_{1}^{N_I} \frac{T^k_{\text{TET}_3}, i}{N_I}, \tag{6.5}$$

where $(T^k_{\text{TET}_1}, i)$, $(T^k_{\text{TET}_2}, i)$, and $(T^k_{\text{TET}_3}, i)$ are the total execution time using **Implementations #1**, **#2**, and **#3**, 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 $T^k_{\text{AET}_3}$. Finally, the relative time improvement $\beta^k_{\text{TI}_1}$ between the **Implementations #1** and **#3**, and relative time improvement $\beta^k_{\text{TI}_2}$ between the **Implementations #2** and **#3**, see their values in Table 13, are given by

$$\beta^k_{\text{TI}_1} = 1 - \left(\frac{T^k_{\text{AET}_3}}{T^k_{\text{AET}_1}} \right). \tag{6.6}$$

Table 13 – Average execution time analysis in μs.

Algorithm	Average Execution Time **Implementation #1** ($T^k_{\text{AET}_1}$)	 **Implementation #2** ($T^k_{\text{AET}_2}$)	 **Implementation #3** ($T^k_{\text{AET}_3}$)	Relative Time Improvement ($\beta^k_{\text{TI}_1}$)	Relative Time Improvement ($\beta^k_{\text{TI}_2}$)
Tomont	38.983	26.989	11.825	69.66%	56.18%
Reduce	77.958	48.896	17.723	77.26%	63.75%
Multiply and Accumulate	672.425	212.740	22.249	96.69%	89.54%
NTT	1038.058	378.480	37.179	96.41%	90.18%
NTT^{-1}	1609.625	550.380	37.214	97.68%	93.24%
Keccak-f[1600]	176.808	56.846	9.434	94.66%	83.40%

and

$$\beta^k_{\text{TI}_2} = 1 - \left(\frac{T^k_{\text{AET}_3}}{T^k_{\text{AET}_2}}\right). \tag{6.7}$$

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 13 shows that the algorithm for the Multiply and Accumulate, NTT, and NTT^{-1} spend a significant amount of time on software (i.e., **Implementations #1** and **#2**). 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 (i.e., **Implementation #3**). Consequently, these blocks show the highest $\beta^k_{\text{TI}_1}$ and $\beta^k_{\text{TI}_2}$. The Keccak-f[1600] is another block with a high $\beta^k_{\text{TI}_1}$ and $\beta^k_{\text{TI}_2}$ 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 $\beta^k_{\text{TI}_1}$ and $\beta^k_{\text{TI}_2}$. The reason is that their implementations require a low number of multiplications, making the hardware performance improvement of these blocks less significant compared to software implementations. Nevertheless, they attain improvement greater than 69% and 77% when compared with **Implementation #1**, and 56% and 63% when compared with **Implementation #2**, respectively.

6.4.2.3 Hardware processing time analysis

The hardware processing time analysis is another evaluation parameter that deserves attention. We focus only on **Implementation #3** to conduct this analysis. We compare the average execution time in hardware ($T^k_{\text{AET}_3}$), already presented in the fourth column of Table 13, versus its processing time only. The $T^k_{\text{AET}_3}$ can be separated into two components, and it is given

Table 14 – Hardware processing time analysis in μs.

Algorithm	Average Execution Time in Hardware (T^k_{AET})	Average Processing Time in Hardware (T^k_{APT})	Relative Time (γ^k_{T})
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%

by

$$T^k_{\text{AET}_3} = T^k_{\text{APT}} + T^k_{\text{DTT}}, \tag{6.8}$$

where the average processing time in hardware (T^k_{APT}) is the time required to process the data, and the data transfer time (T^k_{DTT}) is the time required to transfer data to/from the PS. Table 14 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^k_{TD}), which is given by

$$\gamma^k_{\text{T}} = \frac{T^k_{\text{APT}}}{T^k_{\text{AET}_3}}, \tag{6.9}$$

it can be seen that the Tomont, Reduce, Multiply and Accumulate, and Keccak-f[1600] blocks attain a very low γ^k_{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^k_{TD} higher than 50%, which means that these two blocks spend more time processing data than transferring them. The reason for a higher γ^k_{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.

6.5 SUMMARY

This chapter has discussed the feasibility of using the CRYSTALS-Kyber scheme, a winner in the NIST PQC standardization process, in hardware-constrained equipment, such as QR-DCMs, for securing sensitive data traveling through AMIs. In this sense, it has presented a description of software (**Implementations #1** and **#2**) and hardware/software co-design implementation (**Implementations #3**) of the CRYSTALS-Kyber scheme and a comparison between them. Evaluating the performance of all designed functions for **Implementations #1** and **#2** of the CRYSTALS-Kyber scheme, it can be recognized that the functions listed in Table 10 must be hardware-implemented (i.e., **Implementation #3**) 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. Experimental results have shown that **Implementation #3**, accelerating the most time-consuming functions of the CRYSTALS-Kyber scheme, can reduce the execution time by around 90% and 70% compared to **Implementations #1** and **#2**. **Implementation #3** reduces the execution time from 37.435 ms (**Implementation #1**) and 12.815 ms (**Implementation #2**) to 4.054 ms at the lowest security level and from 101.858 ms (**Implementation #1**) and 30.595 ms (**Implementation #2**) to 8.337 ms at the highest security level. 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, QR-DCMs, in which an FPGA runs the most-time consuming component of the PQC scheme. Since the CRYSTALS-Kyber scheme is a fast and lightweight PQC scheme based on module lattices, it can be efficiently embedded in QR-DCM through a hardware-constrained implementation. Consequently, it is a good candidate for improving security in AMI under classic and quantum computers. Overall, the detailed analyses of this chapter have shown that the implementations of the CRYSTALS-Kyber scheme are suitable for QR-DCMs.

7 AN IMPLEMENTATION OF A QUANTUM-RESISTANT AMI

As discussed in Section 2.2, there are different approaches to achieve a QR-AMI. **Approaches #1** and **#2** modify SMs to enable then to embed PQC schemes, although it would require new compliance tests and might also require new hardware designs, enabling the current SMs already deployed on the field to embed such PQC schemes. On the other hand, **Approach #3** only requires a specific module (i.e., a QR-DCM) to be installed connected to SMs and MDMSs to quickly arms AMIs against quantum threats.

Based on this discussion, large-scale implementations of AMIs can only be accomplished with low-cost solutions, which means that QR-DCMs must be based on hardware-constrained devices; however, no study compared different hardware – e.g., low-power microcontroller and SoC FPGA devices – to enable a QR-AMI. In this regard, based on the results achieved in Chapters 5 and 6, this chapter proposes an AMI based on quantum-resistant schemes (i.e., QR-AMI) to protect key agreements and sensitive data exchange between consumers/prosumers and electric utilities against attacks from quantum and classical computers when implementation issues are considered. The main contributions of this chapter are as follows:

- The proposal of a QR-AMI that relies on FrodoKEM or CRYSTALS-Kyber (already discussed in Chapter 5 and 6, respectively) as asymmetric cryptographic schemes, and the AES-256-GCM performing symmetric cryptography. Also, the introduction of the QR-DCM for including and enabling quantum-resistant key agreements between parties and quantum-resistant encryption and authentication of sensitive data traveling through the AMI.

- The implementation of the QR-AMI based on different hardwares (i.e., low-power microcontroller and SoC FPGA device) using different implementations (i.e., **Implementations #1**, **#2**, and **#3**) that highlights how the QR-DCM interacts and communicates with the well-known components of an AMI.

- A compilation of the performance comparison between the implementations based on low-power microcontrollers, low-cost processor, and SoC FPGA device in terms of resource usage and timing performance presented in Chapters 5 and Chapter 6, and a discussion of scenarios in which the implementations are more suitable.

The rest of this chapter is organized as follows: Section 7.1 proposes and implements a QR-AMI based on QR-DCMs and Section 7.2 evaluates the proposed QR-AMI. Finally, Section 7.3 states some concluding remarks.

7.1 THE QUANTUM-RESISTANT AMI

This section presents the proposed QR-AMI and its key elements. In this sense, Subsection **7.1.1** discusses suitable cryptographic schemes for a quantum-resistant AMI, while

Subsection **7.1.2** discusses the QR-AMI and the main requirements for it to be considered quantum-resistant. Later, Section **7.1.3**, presents a practical implementation, highlighting its main components.

7.1.1 Suitable Cryptographic Schemes For a Quantum-Resistant AMI

This subsection discusses the most appropriate cryptographic schemes to be evaluated in a QR-AMI. As extensively discussed in Chapters 5 and 6, FrodoKEM and CRYSTALS-Kyber are promising choices as asymmetric cryptographic schemes for providing a quantum-resistant KEM. The FrodoKEM scheme is considered cryptographically stronger, at least in theory, compared to the CRYSTALS-Kyber. Although, it comes at the cost of worse timing performance. On the other hand, the CRYSTALS-Kyber scheme relies on a lightweight approach, presenting a better timing performance. Due to that contrast between these schemes, it is interesting their evaluation performing KEM in a QR-AMI in order to identify which scenario each one fits better.

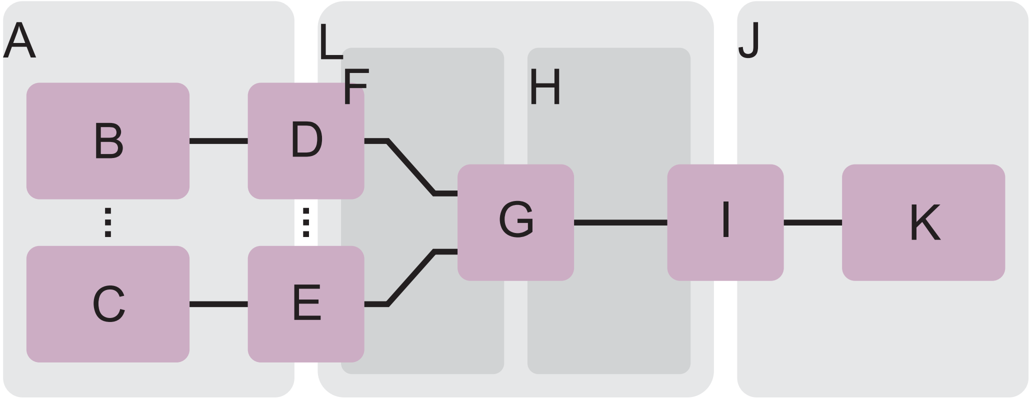
Regarding the symmetric cryptographic scheme, a suitable one for AMIs must be reliable, fast, and lightweight. It is also desirable that this scheme can be combined to provide AEAD. In this sense, two options stand out from the others: (i) AES-256-GCM and (ii) Chacha20-Poly1305. As presented in Section **4.2.2**, both options are secure and perform well, although the former presents a significant performance improvement when the intrinsic implementation of the AES is available (i.e., AES instruction set). As QR-AMIs can be based on different platforms and deployed in various scenarios, it is interesting to choose a symmetric cryptographic scheme that can be accelerated to handle big data loads. Therefore, the AES-256-GCM is considered and evaluated as the symmetric cryptographic scheme for the proposed QR-AMIs. For the sake of simplicity, it will be called AES-GCM henceforth.

7.1.2 A Description of a Quantum-Resistant AMI

A QR-AMI is an infrastructure resistant to attacks in scenarios where adversaries are in possession of quantum (and classical) computers. Consequently, it must rely on quantum-resistant asymmetric and symmetric cryptographic schemes, as described in Section 4.2. In this sense, this chapter proposes the QR-AMI whose block diagram is depicted in Figure 22. It consists of the three parts already known: i) the consumer/prosumer side; ii) communication infrastructure; and iii) the electric utility side. Nonetheless, a key element is also introduced between the consumer/prosumer side and communication infrastructure and between the electric utility side and communication infrastructure, providing quantum resistance: the QR-DCM.

Currently, no off-the-shelf SM embeds quantum-resistant asymmetric cryptographic schemes. In this sense, a QR-DCM is required to interface with the SM to collect data and encrypt it using a quantum-resistant symmetric cryptographic scheme with AEAD (i.e., AES-256-GCM), and send it through the AMI using the communication infrastructure. The proposed QR-DCM performs a key agreement through a KEM using either FrodoKEM or CRYSTALS-Kyber before encrypting and authenticating data, see Subsection **4.2.3** for more details. The introduced

Figure 22 – Block diagram of the proposed QR-AMI divided into i) consumer/prosumer side, ii) communication infrastructure, and iii) electric utility side.



Source: Personal collection.

QR-DCM also receives encrypted data from the communication infrastructure (e.g., control messages), which needs to be authenticated using AEAD, decrypted using the shared secret, and, finally, forwarded to the SM. In our proposed model shown in Figure 22, there is one QR-DCM for each SM, although multiple SMs can be connected to one QR-DCM. The maximum number of SMs for each QR-DCM will depend on the project constraints, such as hardware capabilities and distance between SMs, among others.

On the other hand, the QR-DCM on the electric utility side interfaces with the communication infrastructure receiving/sending encrypted and authenticated data. As the QR-DCM presented on the consumer/prosumer side, it checks the authenticity of received data through the AEAD and decrypts it using a quantum-resistant symmetric cryptographic scheme (i.e., AES-256-GCM). Next, data correctly authenticated is forwarded to the MDMS. Naturally, a key agreement is previously executed using a quantum-resistant asymmetric cryptographic scheme (i.e., FrodoKEM or CRYSTALS-Kyber). It is important to note that the QR-DCM on the electric utility side receives/sends a considerable amount of data depending on the size of the AMI (i.e., depending on how many SMs are presented on it). Therefore, the hardware resource of this QR-DCM must be carefully dimensioned.

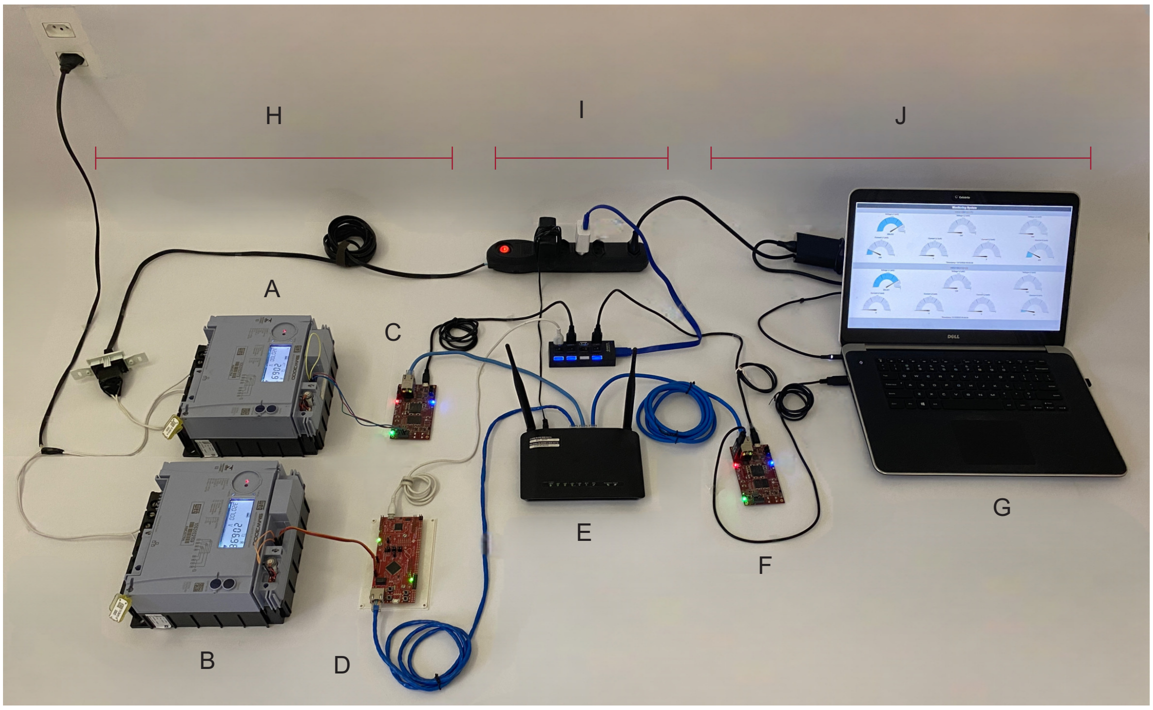
7.1.3 Implementation of the Quantum-Resistant AMI

This subsection details an implementation of the QR-AMI when **Approach #3** is considered. In this sense, Figure 23 shows the implementation carried out to emulate a simplified AMI in which QR-DCMs are implemented in specific microcontroller- and SoC-based development boards. Details of this implementation are presented as follows.

7.1.3.1 Smart Meter

The SMs, tagged as #1A and #1B in Figure 23, are the SMW3000 manufactured by WEG [146]. It is an ANSI bottom-connected type three-phase meter for direct measurement

Figure 23 – An implementation of the proposed QR-AMI.



Source: Personal collection.

up to 120 A. The SMW3000 supports the communication protocol Device Language Message Specification (DLMS)/Companion Specification for Energy Metering (COSEM) through different interfaces (e.g., local port, Ethernet, optical port, serial communication), enabling integration with different market technologies. The DLMS/COSEM protocol is the global standard protocol for energy smart management, advanced control, and innovative metering [147]. In this particular implementation, the SM communicates with the QR-DCM using DLMS/COSEM protocol through serial communication. Notice that in a future off-the-shelf product, the QR-DCM, discussed in Subsection **7.1.2**, may be the processing unit of the SM or part of it in order to reduce costs, see the description of **Approaches #1** and **#2**.

7.1.3.2 Quantum-Resistant Dedicated Cryptographic Module

The QR-DCMs, tagged as #2A, #2B, and #2C in Figure 23, are central pieces because they allow the proposed AMI to be quantum-resistant. Besides embedding quantum-resistant asymmetric and symmetric cryptography, the QR-DCM also embeds the DLMS/COSEM protocol to communicate with the SM properly. Depending on the scale of the AMI, the QR-DCM may demand specific processing power, mainly for the QR-DCM used by the electric utility, because it will need to communicate with all other QR-DCMs of the QR-AMI. Relying on **Approach #3**, the QR-DCM implementation can be accomplished using a microcontroller- and SoC-based technologies because both of them offer distinct tradeoffs for implementing QR-DCMs.

The hardware tagged as #2A in Figure 23 is a low-energy consumption microcontroller and, consequently, a limited processing unit for performing its tasks, which fits well for processing a small amount of data. For instance, on the consumer/prosumer side, connected to only one or a few SMs, or even on the electric utility side when an AMI is dedicated to a small number of SMs. This hardware is based on **Implementation #1**, consequently, symmetric (i.e., AES-256-GCM) and asymmetric (i.e., FrodoKEM and CRYSTALS-Kyber) cryptography schemes are implemented in software. As the processing power of a microcontroller is low, the execution times demanded to run these cryptography schemes are long.

On the other hand, the hardware tagged as #2*B* and #2*C* are based on SoC FPGA device which fits better in scenarios demanding high processing power. For example, when one QR-DCM is connected to many SMs on the consumer/prosumer side, or on the electric utility side when a large number of SMs constitute the AMI. Due to the board flexibility, it is allowed different implementation techniques, such as **Implementation #2** (i.e, a software implementation using only the ARM Cortex-A9) or **Implementation #3** (i.e., hardware/software co-design implementation using ARM Cortex-A9 and FPGA together). In the **Implementation #3**, the FPGA accelerates the most time-consuming routines of the FrodoKEM and CRYSTALS-Kyber schemes, significantly reducing their processing time, see Chapter 5 and 6 for more details. The acceleration brings significant impacts when a QR-DCM at the electric utility side needs to communicate simultaneously with several others QR-DCMs located on the consumer/prosumer side. Note that symmetric cryptography (i.e., AES-256-GCM) is not processing intensive when short data lengths are required to be encrypted, as occurs in AMIs compared to asymmetric cryptography (i.e., FrodoKEM and CRYSTALS-Kyber). Consequently, the former cryptography does not demand hardware acceleration. Section 2.2 shows more details about the hardware used.

Regardless of the hardware adoption, the QR-DCM communicates with the SM using the DLMS/COSEM protocol through serial communication. Also, it communicates with the MDMS through serial communication but using JavaScript Object Notation (JSON), a lightweight data formatting standard, which is much more efficient. An Ethernet connection also connects to the DCU. The communication between the QR-DCM and the MDMS is carried out using serial communication using JSON. Nonetheless, the communication between the QR-DCM and MDMS can rely on any communication protocol, which will depend on physical constraints, the distance between them, expected performance, and the required security level, among others.

7.1.3.3 Communication Infrastructure

The communication infrastructure only comprises a commercial D-Link DSL-2740E router, tagged as **#3** in Figure 23, which forwards packets from one node to another (i.e., allows two-way communication) using the Ethernet network. Note that all the QR-DCMs are connected in the same data network, so they can easily communicate with each other. As it is aimed to focus on the implementation aspects of the quantum-resistant functionality in a QR-AMI, the type of communication infrastructure is not a relevant issue; therefore, no further efforts are necessary to embrace this matter.

In a real QR-AMI implementation, the Ethernet network can be replaced by a short-range data network for providing data communication between SMs/QR-DCMs and the DCU. On the other hand, a long-range data network can be implemented for performing the data communication between DCU and QR-DCMs at the electric utility side.

7.1.3.4 MDMS

Regarding the MDMS, tagged as #4 in Figure 23, it is emulated on a Python application with Flask, a micro-framework suitable for small applications with simple requirements, communicating with the QR-DCM. Mainly, it receives decrypted and authenticated data from the QR-DCMs, which contains information collected by SMs, and presents it on a screen using Flask. As for the simulated communication infrastructure, no further efforts were spent trying to emulate an MDMS better because it would not bring any relevant contribution to the scope of this work.

7.2 PERFORMANCE EVALUATION

This section seeks to evaluate the proposed QR-AMI. The analysis focuses on the QR-DCM since it is the central piece that allows the quantum-resistant communication within AMI. As already discussed in Section 5.2, **Implementations #1** and **#2** aim to show the performance difference between low- and high-power processing units based on a microcontroller (ARM Cortex-M4F) and a low-power processor (ARM Cortex-A9) only, while **Implementation #3** seeks to evaluate the improvement when hardware acceleration is available. Also, hardware acceleration of symmetric cryptography is not considered as it does not bring significant benefits, as will be, henceforth, explained and justified.

For a fair comparison, the source codes used for all implementations are based on those submitted to the NIST PQC standardization process [124]. Furthermore, regarding the FrodoKEM, only the FrodoKEM-640 hardware acceleration is contemplated because its more secure versions (i.e., FrodoKEM-976 and FrodoKEM-1344) would consume a considerable amount of resources, especially BRAMs, as extensively discussed in Chapter 5. Regarding the CRYSTALS-Kyber, all security levels present similar resource usage, as shown in Chapter 6. This section, however, is focused on the CRYSTALS-Kyber-512 to fairly compare both asymmetric cryptographic schemes in terms of security. The implementation of the asymmetric and symmetric cryptographic schemes is evaluated in terms of resource usage and timing performance in Subsections **7.2.1** and **7.2.2**, respectively. Furthermore, Subsection **7.2.3** discusses the most advisable choices for the QR-AMI based on the results presented in the aforementioned subsections.

7.2.1 Resource Usage

This subsection evaluates the resource usage of **Implementations #1**, **#2**, and **#3**. In this sense, Subsection *7.2.1.1* analyses the resource usage of the software part, whereas Subsection *7.2.1.2* addresses the hardware resources of the hardware/software co-design implementation only.

7.2.1.1 Memory Resources

In this analysis, the memory resource usage of **Implementations #1** and **#2** based on software, and the software part of **Implementation #3** is evaluated. Three main segments are evaluated for each implementation: (i) the text segment, which contains the executable instructions; (ii) the read-only data segment, which contains static constants (e.g., global constants) and cannot be altered at run time; and (iii) the data segment, which contains initialized static variables (e.g., global variables and static local variables) and can be modified at run time. The first two segments are usually stored in flash memory as they are constant during the entire run time. On the other hand, variables presented in the data segment might be altered at run time, therefore it is stored in RAM.

Figure 24 depicts the memory usage of the three aforementioned segments. Note that implementations based on the ARM Cortex-M4F (i.e., **Implementation #1**) use fewer memory resources than ARM Cortex-A9-based implementations (i.e., **Implementation #2**). Although the source codes are similar in both implementations, their compilations differ. More powerful processors such as ARM Cortex-A9 tend to be more complex, especially the management part, running in the background, resulting in more resource usage. Naturally, such processors also have more resources available to handle this higher complexity, as pointed out in the description of the evaluation kit in Section *7.1.3.2*.

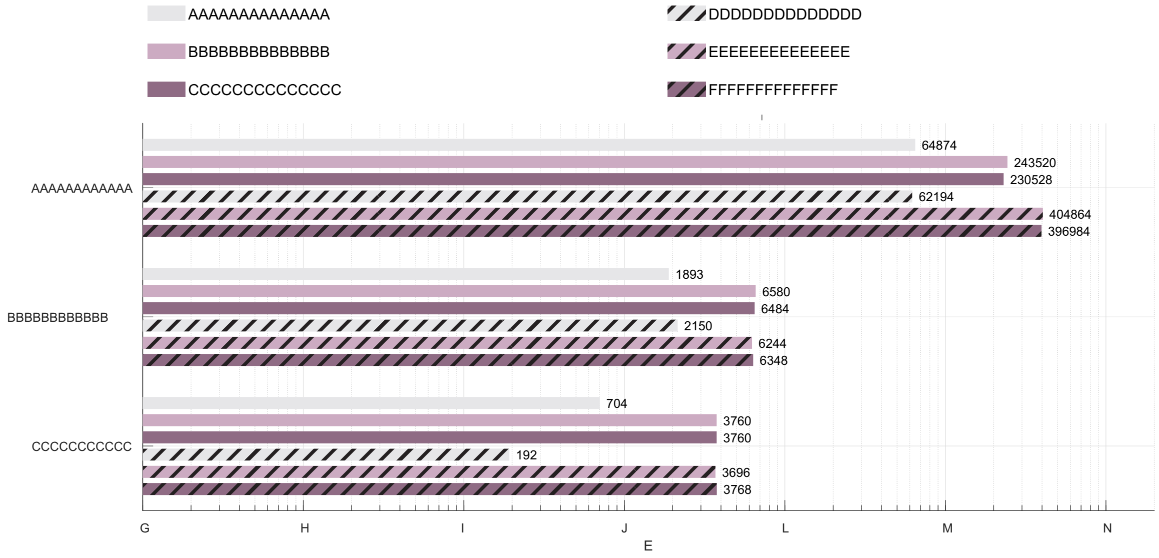
Comparing **Implementation #2** and **Implementation #3** using the same scheme, it can be seen that the hardware/software co-design implementation (i.e., **Implementation #3**) uses fewer memory resources than the software implementation (i.e., **Implementation #2**). It occurs because the functions that are hardware implemented in the **Implementation #3** are not compiled, saving memory resources. However, the difference in size between each segment is insignificant when it exists because only some functions are hardware accelerated, see Chapters 5 and 6.

Under the same implementation, the FrodoKEM uses fewer memory resources than the CRYSTALS-Kyber scheme in **Implementations #2** and **#3**. On the other hand, this scenario is reversed in **Implementation #1** so that the CRYSTALS-Kyber is more economical (except for the read-only data segments, which are very similar in size). Besides using similar source codes, this scenario inversion occurs because they use different compilers. Therefore, a scenario presented by one compilation does not necessarily reflect on the others.

7.2.1.2 Hardware resources

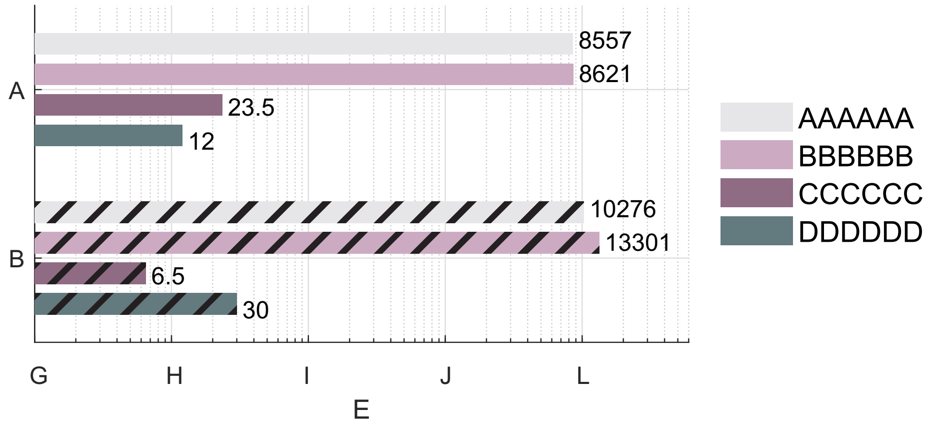
This analysis compares the hardware resource usage of the FrodoKEM and CRYSTALS-Kyber schemes and therefore it considers only the **Implementation #3**. Four main resources are evaluated: (i) slice LUTs, which are small asynchronous RAMs manipulated to implement combinational logic; (ii) slice registers, based on Flip Flop (FF) and used to hold states,

Figure 24 – Memory usage for text, read-only data, and data segments in bytes of the FrodoKEM and CRYSTALS-Kyber schemes considering **Implementations #1**, **#2**, and **#3**.



Source: Personal collection.

Figure 25 – Hardware resource usage of the FrodoKEM and CRYSTALS-Kyber implementations in terms of Slice LUTs, Slice Register, BRAM, and DSP blocks.



Source: Personal collection.

(iii) BRAM, based on synchronous RAM and used to store large volumes of data; and (iv) DSP blocks, which are dedicated blocks for performing multiplications, increasing performance, and reducing logic use (e.g., slice LUTs and registers).

Figure 25 shows the resource usage of the FrodoKEM and CRYSTALS-Kyber schemes. Observe that the CRYSTALS-Kyber scheme consumes slightly more slice LUTs, slice registers, and DSP blocks than the FrodoKEM scheme. It can be explained by the fact that FrodoKEM implements fewer functions in hardware than the CRYSTALS-Kyber scheme, see Chapters 5 and 6. On the other hand, FrodoKEM consumes more BRAMs, because it has longer keys when compared with CRYSTALS-Kyber, see Table 1. Consequently, it uses more memory to store and operate on them temporarily.

7.2.2 Timing Analysis

This subsection presents a timing analysis of the QR-DCM. In this regard, Subsection *7.2.2.1* compares the execution time of the asymmetric cryptographic schemes, while Subsection *7.2.2.2* presents the timing analysis of the symmetric cryptography.

7.2.2.1 Asymmetric cryptography

Now, it is shown the execution times of key pair generation, encapsulation, and decapsulation (i.e., the three steps to perform a key agreement between parties). FrodoKEM and CRYSTALS-Kyber, which are software and hardware/software implemented (i.e., **Implementations #1**, **#2**, and **#3**), are then compared based on the total execution time.

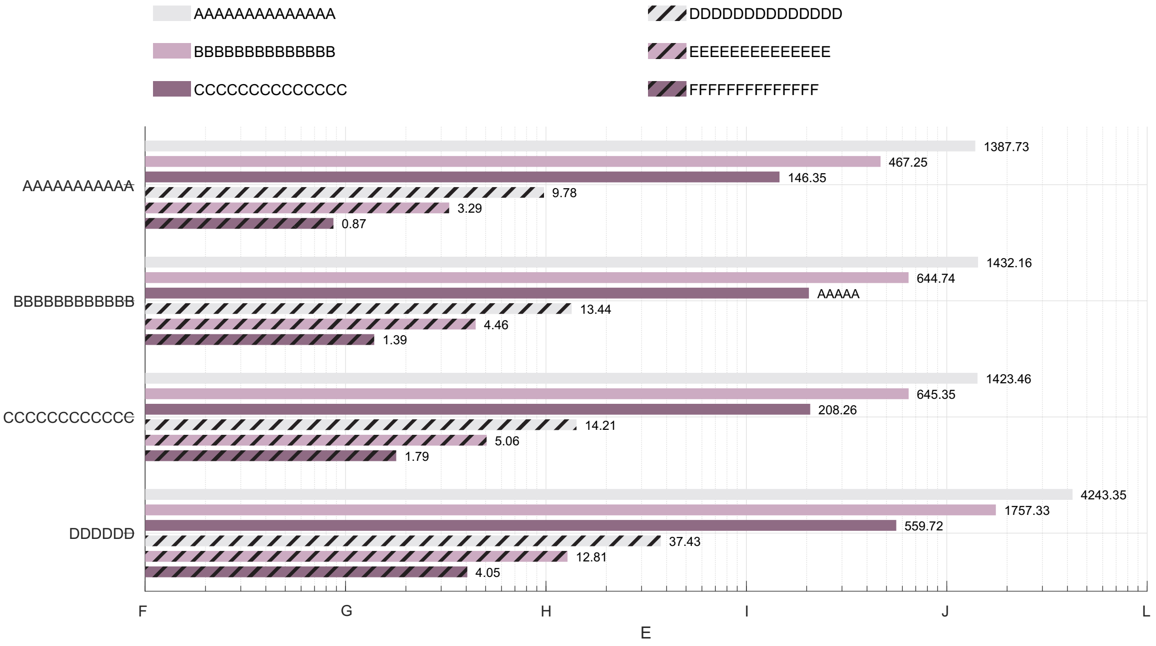
Figure 26 shows the execution time results for the aforementioned cases. Notice the significant difference in the total execution time of the FrodoKEM and CRYSTALS-Kyber schemes. Comparing the execution time of each implementation technique for the FrodoKEM with its counterpart for the CRYSTALS-Kyber, the latter is at least a hundred times faster than the former, regardless of the implementation used in the comparison. This outcome validates the comparative discussion in Subsection *4.3.2.4*.

Regarding the FrodoKEM scheme, its **Implementation #3** is around 7.5 and 3.1 times faster than its **Implementations #1** and **#2**, respectively, reducing the total execution time from 4243.35 ms and 1757.33 ms to 559.72 ms if hardware acceleration is taken into account. The CRYSTALS-Kyber achieves similar results such that its **Implementation #3** is around 9.9 and 3.2 times faster than its **Implementations #1** and **#2**, respectively. As a result, it reduces the total execution time of **Implementations #1** and **#2** from 40.02 ms and 12.81 ms, also respectively, to 4.05 ms when hardware acceleration applies (i.e., **Implementations #3**).

7.2.2.2 Symmetric cryptography

Regarding symmetric cryptography, the time required to encrypt/decrypt and authenticate data is directly correlated with the data length of the AES-256-GCM scheme. To encrypt/decrypt and authenticate 50 bytes, the AES-256-GCM scheme takes 0.262 ms and 0.062 ms using **Implementations #1** and **#2**, respectively. Based on the obtained results, it can be seen that the execution time of symmetric cryptography is very small and irrelevant when compared with the execution time of asymmetric cryptography. This fact confirms that a hardware implementation of the AES-256-GCM scheme would not bring relevant advantages in terms of execution time when small packets of data are required, which is the case of an AMI or a QR-AMI, besides still consuming hardware resources. For this reason, symmetric cryptography is only implemented in the software version.

Figure 26 – Timing performance in ms of the FrodoKEM and CRYSTALS-Kyber schemes using **Implementations #1**, **#2**, and **#3**.



Source: Personal collection.

7.2.3 Quantum-Resistant AMI Analysis

The FrodoKEM scheme demands a resource usage similar to the CRYSTALS-Kyber and is considerably slower. Also, the only situation that would be advisable to use the FrodoKEM would be when robust cryptographic principles, at least in theory, are of utmost importance, as it relies on mathematical primitives well-established and studied by the academic community, see Subsection *4.3.2.2*. However, FrodoKEM might not be feasible on large QR-AMIs because it might not meet timing constraints. In all other situations, the CRYSTALS-Kyber scheme would be a better option. When it comes to symmetric cryptographic schemes, as discussed in Subsection **4.2.2**, AES-256-GCM is suitable in any scenario. It is a lightweight block cipher with AEAD applicable to be embedded in any hardware.

Regarding QR-DCM, the conditions suitable for using microcontrollers (i.e., **Implementation #1**), low-power processor (i.e., **Implementation #2**), and SoC FPGA devices (i.e., **Implementation #3**) are very distinct. For consumers/prosumers requiring low processing power (e.g., QR-DCM is only connected to one SM), the hardware based on a microcontroller with an ARM Cortex-M4F (i.e., **Implementation #1**) would be more advised aiming to reduce costs. Nonetheless, if the QR-DCM is connected with multiple SMs, timing constraints will apply and a processing unit with an ARM Cortex-A9 may suit well (i.e., **Implementation #2**). In extreme situations where high processing power is needed, the QR-DCM should be based on **Implementation #3**.

On the electric utility side, the size of the AMI must be considered because the QR-DCM of a MDMS will receive data from all SMs. In this sense, using **Implementation #1** would only be possible in a very small AMIs because it is possible to comply with timing constraints. As the size of the AMI grows, **Implementation #2** would be a better choice. In large-size AMIs, where high loads of data are exchanged, **Implementation #3** will be required.

7.3 SUMMARY

This chapter discussed the QR-AMI for securing sensitive data traveling through the data communication networks against attacks from quantum (and classical) computers. The QR-AMI mainly relies on implementing cryptography schemes on QR-DCMs, which is based on two asymmetric cryptographic schemes, one with more robust cryptographic principles (i.e., FrodoKEM) and another which performs well (i.e., CRYSTALS-Kyber). As no off-the-shelf SM supports quantum-resistant asymmetric cryptographic schemes, QR-DCM is a simple solution for including and enabling quantum resistance in AMI. In addition, QR-DCMs rely on symmetric cryptographic schemes which efficiently encrypt/decrypt and authenticate data (i.e., AES-256-GCM). The numerical results show that FrodoKEM and CRYSTALS-Kyber asymmetric cryptographic schemes present similar resource usage in software (Implementations #**1** and #**2**) and hardware/software co-design implementation (Implementation #**3**). Regarding timing performance, the CRYSTALS-Kyber performs considerably better in all implementations. Although, these schemes present a trade-off between timing performance and security. Theoretically, FrodoKEM is cryptographically stronger but at the cost of worst timing performance, while CRYSTALS-Kyber is the exact opposite. Regarding the execution time of symmetric cryptography, it is negligible compared to the execution time of asymmetric cryptography. The implementation of the QR-AMI shows that QR-DCM implemented on a microcontroller or SoC FPGA device and the use of the FrodoKEM or CRYSTALS-Kyber schemes depends on the applied constraints. The implementation using a microcontroller suits well in AMI demanding small data volume. On the other hand, SoC-based implementation is necessary for AMIs that exchange a large volume of data. Furthermore, the use of the FrodoKEM scheme only makes sense when stronger cryptographic principles are required. Otherwise, the CRYSTALS-Kyber scheme is a better choice. Overall, the attained results show that the proposed QR-AMI can be carried out using QR-DCMs based on low-cost hardware to secure sensitive data exchange against attacks from quantum (and classical) computers.

8 CONCLUSIONS

This dissertation has investigated a few aspects related to the implementation of QR-AMI, which is an AMI relying on quantum-resistant schemes for providing security and privacy. The feasibility of implementing asymmetric cryptographic schemes performing key agreements between parties to secure AMIs facing the threat of quantum computers has been analyzed based on QR-DCM. Also, the feasibility of implementing a symmetric cryptographic scheme has been also investigated to perform ciphering/deciphering and authentication of sensitive data traveling through the AMI from a practical perspective. Overall, it has investigated different directions for implementing QR-AMI based on a feasible and cost-effective QR-DCMs. The investigated QR-DCMs can be used AMIs already deployed or be integrated into the new ones.

In this sense, Chapter 2 has presented the state-of-art of AMI security and pointed out the necessity of advancing the investigation of quantum-resistant schemes for AMIs. Based on this, a problem formulation regarding the security of AMI in a quantum era has been covered, where QR-AMI and QR-DCM were detailed. The formulation statement introduced the research questions investigated in this dissertation.

Chapter 3 has provided a concise description of AMIs, emphasizing its key elements (consumers/prosumers, communication infrastructure, and electric utility). The communication infrastructure of AMIs comprises different communication media and relies on various equipment; this chapter discussed the vulnerabilities, threats, security, and privacy issues.

Chapter 4 has highlighted quantum computers and has discussed which principles they rely on, why they potentially will break classical cryptography schemes, and their recent advances and achievements. Cryptographic fundamentals were also discussed, where asymmetric and symmetric cryptography were reviewed, and how they interact with each other was explained. In the sequel, the PQC schemes were discussed, which are quantum-resistant schemes (i.e., resistant against attacks from quantum computers as far as is known). This chapter has also reported the NIST contest to standardize quantum-resistant schemes. In this contest, the lattice-based schemes, FrodoKEM and CRYSTALS-Kyber were recognized due to their characteristics. While the former is based on strong cryptographic principles, at least in theory, the latter is based on a lightweight ones. In the end, this chapter presented the justification for pursuing FrodoKEM and CRYSTALS-Kyber schemes for implementing QR-DCM for QR-AMIs.

Chapter 5 has investigated the FrodoKEM as a KEM embedded in QR-DCMs for QR-AMIs. In this sense, the backgrounds of this scheme were presented, highlighting its main characteristics. The FrodoKEM relies on the LWE problem, which has a conservative approach based on strong principles, at least in theory. Although, it comes at the cost of timing performance. To overcome this issue, this chapter has presented and analyzed a software implementation of this scheme using two different hardware (i.e., a microcontroller and a low-cost processor). The analysis aimed to identify the most time-consuming routines, which are good candidates to be hardware accelerated and, consequently, to improve timing performance. The analysis showed that two matrix-by-matrix multiplications and a hash function are the most time-consuming routines in the FrodoKEM scheme. Therefore, these routines were implemented in hardware while the remaining scheme was implemented in software, originating a hardware/software co-design implementation of the FrodoKEM scheme. Regarding timing performance, the hardware/software co-design implementation improves by 86.80% and 68.15% compared with the software implementation in the microcontroller and low-cost processor, respectively. As the FrodoKEM is a conservative scheme with a not-so-good timing performance, hardware acceleration is recommended for meeting the timing constraints of AMIs, which impose the necessity of using hardware with more computational power; however, the strong cryptographic primitives of the FrodoKEM schemes makes them very interesting choices under the requirement of more sense of security.

Chapter 6, in contrast with Chapter 5, has investigated the CRYSTALS-Kyber as a KEM embedded in QR-DCMs for QR-AMIs. The backgrounds of this scheme were discussed, showing its main characteristics. The CRYSTALS-Kyber relies on the M-LWE problem, which has a lightweight approach based on polynomial rings (in contrast with matrix-by-matrix multiplications of FrodoKEM). Although this additional structure (e.g., polynomial rings) might present weaknesses in a yet-to-be-discovered attack, it is less likely to be applied in M-LWE problem than in R-LWE problem. On the other hand, its structure significantly enhances timing performance compared with the LWE problem. Software implementations were evaluated using two different hardware (i.e., a microcontroller and a low-cost processor), seeking to identify the most time-consuming routines, which are candidates to be hardware accelerated for improving, even more, the timing performance. Based on this analysis, the most time-consuming routines were hardware implemented, aiming to achieve an architecture where commonly required resources between blocks were shared. The remaining routines were implemented in software, originating a hardware/software co-design implementation of the CRYSTALS-Kyber scheme. The attained results showed that the hardware/software co-design implementation improved $\approx 90\%$ and $\approx 70\%$ compared to the software implementation of the microcontroller and low-cost processor, respectively. Overall, the CRYSTALS-Kyber, the winner of the NIST PQC standardization process, shows an excellent timing performance; however, hardware acceleration stands as an interesting alternative when extremely high performance (very short timing constraint) is required.

Chapter 7 has paid attention to the AMI relying on asymmetric and symmetric quantum-resistant schemes (i.e., a QR-AMI). In this sense, suitable quantum-resistant cryptographic schemes for embedding in the QR-AMI were discussed. Besides the asymmetric cryptographic schemes FrodoKEM and CRYSTALS-Kyber scheme already discussed, the symmetric cryptographic scheme AES-256-GCM is an interesting option for ciphering/deciphering and authenticating sensitive data. With these schemes embedded in a QR-DCM, an implementation of a QR-AMI was introduced, highlighting where the QR-DCM should be placed to guarantee a quantum-resistant AMI. Also, a practical implementation of a QR-AMI was described, paying attention to the details of its elements. The attained results showed that the resource usage of FrodoKEM and CRYSTALS-Kyber schemes are very similar, but the timing performance between them is not. The CRYSTALS-Kyber scheme is, at least, a hundred times faster than the FrodoKEM. For instance, the FrodoKEM requires 559.72 ms to perform key pair generation, encapsulation, and decapsulation based on the hardware/software co-design implementation while the CRYSTALS-Kyber requires only 4.05 ms using the same implementation technique. Based on this, the only situation that would be advisable to adopt the FrodoKEM instead of the CRYSTALS-Kyber scheme in a QR-AMI would be when robust cryptographic principles, at least in theory, are of utmost importance. Furthermore, regarding the QR-DCM, the conditions suitable for using microcontrollers, low-cost processors, and SoC FPGA devices are very distinct. The use of microcontrollers would be more advised when the QR-DCM is connected to only one or a few SMs on the consumer/prosumer side. Also, when very small QR-AMI applies, a microcontroller would be a good choice to be deployed on the electric utility side to reduce costs. Although, when it is demanded that the QR-DCM connects with multiple SMs simultaneously, a low-cost processor would be more advisable as more processing power would be required. Furthermore, when the QR-DCM is connected with the MDMS (i.e., connected with the electric utility) and it communicates with large-size QR-AMIs, a QR-DCM based on a low-cost processor would also be advisable. Finally, in extreme scenarios in which a large QR-AMIs must be deployed, the SoC FPGA device better suits for the MDMS.

As QR-AMI is a new and complex topic to be addressed, the following topics deserve attention in future works:

- To study optimization techniques that would accelerate the matrix-by-matrix multiplication of FrodoKEM scheme. Furthermore, optimize data transfer between PS and PL aiming to reduce the average execution time of the schemes.

- Analysis of optimized software implementations of the FrodoKEM and CRYSTALS-Kyber scheme focused on characteristics of specific targets, instead of the reference implementation which does not present any optimization. This investigation would seek to improve timing performance by taking advantage of the architecture and optimizing resources of processors.

- Evaluation of other symmetric cryptographic schemes, especially the ASCON-80pq scheme [148], which is the quantum-resistant version of the winner of the NIST lightweight standardization process, a competition aiming to select a symmetric scheme for Internet of Things (IoT) devices.

REFERENCES

[1] A. Ghasempour and J. Lou, “Advanced metering infrastructure in smart grid: Requirements, challenges, architectures, technologies, and optimizations,” in *Smart Grids: Emerging Technologies, Challenges and Future Directions*. Nova Science Publishers, Oct. 2017.

[2] A. Caillé, M. Al-Moneef, F. B. de Castro, A. Bundgaard-Jensen, A. Fall, N. F. de Medeiros, C. Jain, Y. D. Kim, M.-J. Nadeau, C. Testa *et al.*, “Deciding the future: Energy policy scenarios to 2050,” World Energy Council, London, UK, Tech. Rep., Apr. 2007.

[3] L. d. 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 of Things Journal*, vol. 5, no. 2, pp. 655–667, Oct. 2017.

[4] 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 Communications Surveys Tutorials*, vol. 21, no. 1, pp. 920–939, Aug. 2018.

[5] J. Anatory, M. V. Ribeiro, A. M. Tonello, and A. Zeddam, “Power-line communications: smart grid, transmission, and propagation,” *Journal of Electrical and Computer Engineering*, vol. 2013, pp. 4–4, Apr. 2013.

[6] M. R. Asghar, G. Dán, D. Miorandi, and I. Chlamtac, “Smart meter data privacy: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2820–2835, June 2017.

[7] “Advanced metering infrastructure market-growth,” https://www.mordorintelligence.com /industry-reports/advanced-metering-infrastructure-market, accessed: 2023-04-13.

[8] “Enel begins installation of smart meters in são paulo,” https://www.enel.com/media/explore/search-press-releases/press/2021/01/enel-begins-installation-of-smart-meters-in-so-saulo, accessed: 2023-04-13.

[9] “Cemig investe mais de r$ 200 milhões para modernizar parque de medidores de energia,” https://www.cemig.com.br/noticia/cemig-investe-mais-de-200-milhoes-de-reais-para-modernizar-parque-de-medidores-de-energia/, accessed: 2023-04-13.

[10] M. Shokry, A. I. Awad, M. K. Abd-Ellah, and A. A. Khalaf, “Systematic survey of advanced metering infrastructure security: Vulnerabilities, attacks, countermeasures, and future vision,” *Future Generation Computer Systems*, vol. 136, pp. 358–377, Nov. 2022.

[11] S. Ali, T. Al Balushi, Z. Nadir, and O. K. Hussain, *Embedded Systems Security for Cyber-Physical Systems*. Cham: Springer International Publishing, Mar. 2018, pp. 115–140.

[12] D. Chen, Z. Li, T. Li, M. Teng, and F. Xie, “Big data based intrusion detection of smart meters,” in *Proc. 2nd International Conference on Computer, Network Security and Communication Engineering*, May 2017, pp. 108–112.

[13] P. Yi, T. Zhu, Q. Zhang, Y. Wu, and L. Pan, “Puppet attack: A denial of service attack in advanced metering infrastructure network,” *Journal of Network and Computer Applications*, vol. 59, pp. 325 – 332, Jan. 2016.

[14] J. Tsai and N. Lo, "Secure anonymous key distribution scheme for smart grid," *IEEE Transactions on Smart Grid*, vol. 7, no. 2, pp. 906–914, June 2015.

[15] V. Odelu, A. K. Das, M. Wazid, and M. Conti, "Provably secure authenticated key agreement scheme for smart grid," *IEEE Transactions on Smart Grid*, vol. 9, no. 3, pp. 1900–1910, Aug. 2016.

[16] N. Komninos, E. Philippou, and A. Pitsillides, "Survey in smart grid and smart home security: Issues, challenges and countermeasures," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 1933–1954, Apr. 2014.

[17] N. Saxena and S. Grijalva, "Dynamic secrets and secret keys based scheme for securing last mile smart grid wireless communication," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1482–1491, Sept. 2016.

[18] T. Liu, Y. Liu, Y. Mao, Y. Sun, X. Guan, W. Gong, and S. Xiao, "A dynamic secret-based encryption scheme for smart grid wireless communication," *IEEE Transactions on Smart Grid*, vol. 5, no. 3, pp. 1175–1182, Aug. 2014.

[19] W. Wang and Z. Lu, "Cyber security in the smart grid: Survey and challenges," *Computer Networks*, vol. 57, no. 5, pp. 1344 – 1371, Apr. 2013.

[20] C. Fan, S. Huang, and Y. Lai, "Privacy-enhanced data aggregation scheme against internal attackers in smart grid," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 666–675, Aug. 2014.

[21] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th Annual Symposium on Foundations of Computer Science*, Nov. 1994, pp. 124–134.

[22] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proc. 28th annual ACM symposium on Theory of computing*, July 1996, pp. 212–219.

[23] M. Grassl, B. Langenberg, M. Roetteler, and R. Steinwandt, "Applying Grover's algorithm to AES: Quantum resource estimates," in *Post-Quantum Cryptography*, T. Takagi, Ed. Cham: Springer International Publishing, Feb. 2016, pp. 29–43.

[24] D. Joseph, R. Misoczki, M. Manzano, J. Tricot, F. D. Pinuaga, O. Lacombe, S. Leichenauer, J. Hidary, P. Venables, and R. Hansen, "Transitioning organizations to post-quantum cryptography," *Nature*, vol. 605, no. 7909, pp. 237–243, May 2022.

[25] 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 Transactions on Smart Grid*, vol. 10, no. 6, pp. 6386–6395, Mar. 2019.

[26] 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. 142 413–142 422, July 2020.

[27] J. Ahn, H.-Y. Kwon, B. Ahn, K. Park, T. Kim, M.-K. Lee, J. Kim, and J. Chung, "Toward quantum secured distributed energy resources: Adoption of post-quantum cryptography (PQC) and quantum key distribution (QKD)," *Energies*, vol. 15, no. 3, Jan. 2022.

[28] M. Hadley, K. Huston, and T. Edgar, "Aga-12, part 2 performance test results," *Pacific Northwest National Laboratories*, Aug. 2007.

[29] P. CODE, "Technical IEC specification ts 62351-1," *IEC*, May 2018.

[30] S. M. S. Hussain, T. S. Ustun, and A. Kalam, "A review of IEC 62351 security mechanisms for IEC 61850 message exchanges," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 5643–5654, Nov. 2019.

[31] Y. Li, P. Zhang, and R. Huang, "Lightweight quantum encryption for secure transmission of power data in smart grid," *IEEE Access*, vol. 7, pp. 36 285–36 293, Jan. 2019.

[32] A. Philips, J. Jayaraj, F. Josh, and P. Venkateshkumar, "Enhanced RSA key encryption application for metering data in smart grid," *International Journal of Pervasive Computing and Communications*, Nov. 2021.

[33] D. He, H. Wang, M. K. Khan, and L. Wang, "Lightweight anonymous key distribution scheme for smart grid using elliptic curve cryptography," *IET Communications*, vol. 10, no. 14, pp. 1795–1802, Sept. 2016.

[34] A. Molina-Markham, G. Danezis, K. Fu, P. Shenoy, and D. Irwin, "Designing privacy-preserving smart meters with low-cost microcontrollers," in *Proc. International Conference on Financial Cryptography and Data Security*. Springer, Jan. 2012, pp. 239–253.

[35] J. Seo, J. Jin, J. Y. Kim, and J.-J. Lee, "Automated residential demand response based on advanced metering infrastructure network," *International Journal of Distributed Sensor Networks*, vol. 12, no. 2, pp. 1–4, Feb. 2016.

[36] J. Ni, K. Zhang, K. Alharbi, X. Lin, N. Zhang, and X. S. Shen, "Differentially private smart metering with fault tolerance and range-based filtering," *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2483–2493, Feb. 2017.

[37] J. Won, C. Y. T. Ma, D. K. Y. Yau, and N. S. V. Rao, "Privacy-assured aggregation protocol for smart metering: A proactive fault-tolerant approach," *IEEE/ACM Transactions on Networking*, vol. 24, no. 3, pp. 1661–1674, May 2015.

[38] M. Cazorla, K. Marquet, and M. Minier, "Survey and benchmark of lightweight block ciphers for wireless sensor networks," in *2013 International Conference on Security and Cryptography (SECRYPT)*, July 2013, pp. 1–6.

[39] J. Lee, K. Kapitanova, and S. H. Son, "The price of security in wireless sensor networks," *Computer Networks*, vol. 54, no. 17, pp. 2967 – 2978, Dec. 2010.

[40] A. Trad, A. A. Bahattab, and S. Ben Othman, "Performance trade-offs of encryption algorithms for wireless sensor networks," in *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, Jan. 2014, pp. 1–6.

[41] J. H. Kong, L.-M. Ang, and K. P. Seng, "A comprehensive survey of modern symmetric cryptographic solutions for resource constrained environments," *Journal of Network and Computer Applications*, vol. 49, pp. 15 – 50, Mar. 2015.

[42] 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 Generation Computer Systems*, vol. 72, pp. 327–338, July 2017.

[43] S. Finster and I. Baumgart, "Privacy-aware smart metering: A survey," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 1088–1101, Apr. 2015.

[44] 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 Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2886–2927, Feb. 2019.

[45] P.-Y. Kong, "A review of quantum key distribution protocols in the perspective of smart grid communication security," *IEEE Systems Journal*, vol. 16, no. 1, pp. 41–54, Oct. 2022.

[46] F. Borges, R. A. Santos, and F. L. Marquezino, "Preserving privacy in a smart grid scenario using quantum mechanics," *Security and communication networks*, vol. 8, no. 12, pp. 2061–2069, Aug. 2015.

[47] R. Diovu and J. Agee, "Enhancing the security of a cloud-based smart grid ami network by leveraging on the features of quantum key distribution," *Transactions on Emerging Telecommunications Technologies*, vol. 30, no. 6, p. e3587, Mar. 2019.

[48] E. Diamanti, H.-K. Lo, B. Qi, and Z. Yuan, "Practical challenges in quantum key distribution," *npj Quantum Information*, vol. 2, no. 1, pp. 1–12, Nov. 2016.

[49] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-quantum lattice-based cryptography implementations: A survey," *ACM Computing Surveys*, vol. 51, no. 6, Jan. 2019, art. no. 129.

[50] N. Gupta, A. Jati, A. K. Chauhan, and A. Chattopadhyay, "PQC acceleration using GPUs: Frodokem, newhope, and kyber," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 575–586, Sept. 2020.

[51] Y. Huang, M. Huang, Z. Lei, and J. Wu, "A pure hardware implementation of CRYSTALS-kyber PQC algorithm through resource reuse," *IEICE Electronics Express*, vol. 17, no. 17, Aug. 2020.

[52] J. Buchmann, F. Göpfert, T. Güneysu, T. Oder, and T. Pöppelmann, "High-performance and lightweight lattice-based public-key encryption," in *Proc. of the 2nd ACM international workshop on IoT privacy, trust, and security*, May 2016, pp. 2–9.

[53] D. Liu, N. Li, J. Kim, and S. Nepal, "Compact-LWE: Enabling practically lightweight public key encryption for leveled iot device authentication," *Cryptology ePrint Archive*, Jan. 2017.

[54] A. Marotzke, "A constant time full hardware implementation of streamlined NTRU prime," in *Proc. International Conference on Smart Card Research and Advanced Applications*. Springer, Nov. 2020, pp. 3–17.

[55] 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, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2019, pp. 1148–1153.

[56] 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*, pp. 239–280, Aug. 2020.

[57] 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, Aug. 2019.

[58] 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, Apr. 2020.

[59] 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, June 2021.

[60] V. Mavroeidis, K. Vishi, M. D., and A. Jøsang, “The impact of quantum computing on present cryptography,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 3, Mar. 2018.

[61] D. J. Bernstein and T. Lange, “Post-quantum cryptography,” *Nature*, vol. 549, no. 7671, pp. 188–194, Sept. 2017.

[62] X. Bogomolec, J. G. Underhill, and S. A. Kovac, “Towards post-quantum secure symmetric cryptography: A mathematical perspective,” Oct. 2019.

[63] Z. Wan, G. Wang, Y. Yang, and S. Shi, “SKM: Scalable key management for advanced metering infrastructure in smart grids,” *IEEE Transactions on Industrial Electronics*, vol. 61, no. 12, pp. 7055–7066, June 2014.

[64] Z. Sadowski, “Comparison of PLC-PRIME and PLC-G3 protocols,” in *2015 International School on Nonsinusoidal Currents and Compensation (ISNCC)*, June 2015, pp. 1–6.

[65] V. L. R. da Costa, V. Fernandes, and M. V. Ribeiro, “Narrowband hybrid PLC/wireless: Transceiver prototype, hardware resource usage and energy consumption,” *Ad Hoc Networks*, vol. 94, Nov. 2019, art. no. 101945.

[66] H. K. Sahu, A. K. Padhan, S. G. Dontamsetti, and P. R. Sahu, “Performance analysis of smart grid network with energy harvesting over mixed rf/plc channel,” in *2023 National Conference on Communications (NCC)*, Feb. 2023, pp. 1–6.

[67] Â. Camponogara, H. V. Poor, and M. V. Ribeiro, “PLC systems under the presence of a malicious wireless communication device: Physical layer security analyses,” *IEEE Systems Journal*, vol. 14, no. 4, pp. 4901–4910, Jan. 2020.

[68] T. R. Oliveira, A. A. Picorone, C. B. Zeller, S. L. Netto, and M. V. Ribeiro, “On the statistical characterization of hybrid plc-wireless channels,” *Electric Power Systems Research*, vol. 163, pp. 329–337, Oct. 2018.

[69] V. Fernandes, H. V. Poor, and M. V. Ribeiro, “A hybrid power line/wireless dual-hop system with energy harvesting relay,” *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 4201–4211, July 2018.

[70] T. Oliveira, F. Andrade, A. Picorone, H. Latchman, S. Netto, and M. Ribeiro, “Characterization of hybrid communication channel in indoor scenario,” *Journal of Communication and Information Systems*, vol. 31, no. 1, Sep. 2016.

[71] H. Harada, K. Mizutani, J. Fujiwara, K. Mochizuki, K. Obata, and R. Okumura, "IEEE 802.15. 4g based Wi-SUN communication systems," *IEICE Transactions on Communications*, vol. 100, no. 7, pp. 1032–1043, July 2017.

[72] S. Banerji, "On IEEE 802.11: Wireless LAN technology," *CoRR*, vol. abs/1307.2661, Mar. 2014.

[73] V. Fernandes, T. F. A. Nogueira, H. V. Poor, and M. V. Ribeiro, "Statistical modeling of energy harvesting in hybrid PLC-WLC channels," *Sustainability*, vol. 14, no. 1, Dec. 2021.

[74] M. de L. Filomeno, M. L. R. de Campos, H. Vincent Poor, and M. V. Ribeiro, "Hybrid power line/wireless systems: Power allocation for minimizing the average bit error probability," *IEEE Transactions on Communications*, vol. 70, no. 2, pp. 810–821, Dec. 2021.

[75] V. Fernandes, T. Oliveira, and M. Ribeiro, "The usefulness of the energy harvested from additive noises in power line and wireless media," *Journal of Communication and Information Systems*, vol. 35, no. 1, pp. 61–65, Mar. 2020.

[76] K. Christensen, P. Reviriego, B. Nordman, M. Bennett, M. Mostowfi, and J. A. Maestro, "IEEE 802.3az: the road to energy efficient ethernet," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 50–56, Nov. 2010.

[77] M. G. Jaatun, I. A. Tøndel, and G. M. Køien, "GPRS security for smart meters," in *1st Cross-Domain Conference and Workshop on Availability, Reliability, and Security in Information Systems (CD-ARES)*. Springer, Sept. 2013, pp. 195–207.

[78] E. Ezhilarasan and M. Dinakaran, "A review on mobile technologies: 3G, 4G and 5G," in *2017 Second International Conference on Recent Trends and Challenges in Computational Models (ICRTCCM)*. IEEE, Oct. 2017, pp. 369–373.

[79] D. Bian, M. Kuzlu, M. Pipattanasomporn, and S. Rahman, "Analysis of communication schemes for advanced metering infrastructure (AMI)," in *2014 IEEE PES General Meeting | Conference & Exposition*, July 2014, pp. 1–5.

[80] S. Galli, A. Scaglione, and Z. Wang, "Power line communications and the smart grid," in *2010 First IEEE International Conference on Smart Grid Communications*, Oct. 2010, pp. 303–308.

[81] F. P. V. de Campos, L. M. Sirimarco, M. L. R. de Campos, and M. V. Ribeiro, "The implementation of the clustered-OFDM-based transceiver on an FPGA device: A comprehensive comparison," *IET Communications*, vol. 15, no. 6, pp. 824–839, Feb. 2021.

[82] H. V. Poor, "Clustered-orthogonal frequency division multiplexing for power line communication: when is it beneficial?" *IET Communications*, vol. 8, pp. 2336–2347(11), September 2014. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-com.2014.0056

[83] V. L. R. da Costa, H. V. Schettino, A. Camponogara, F. P. de Campos, and M. V. Ribeiro, "Digital filters for clustered-ofdm-based plc systems: Design and implementation," *Digital Signal Processing*, vol. 70, pp. 166–177, Nov. 2017.

[84] M. Kuzlu, M. Pipattanasomporn, and S. Rahman, “Communication network requirements for major smart grid applications in han, nan and wan,” *Computer Networks*, vol. 67, pp. 74–88, July 2014.

[85] X. Liu and Z. Li, “False data attacks against AC state estimation with incomplete network information,” *IEEE Transactions on Smart Grid*, vol. 8, no. 5, pp. 2239–2248, Feb. 2016.

[86] S. Tweneboah-Koduah, A. K. Tsetse, J. Azasoo, and B. Endicott-Popovsky, “Evaluation of cybersecurity threats on smart metering system,” in *Information Technology-New Generations*. Springer, July 2017, pp. 199–207.

[87] K. Fu, A. Drobnis, G. Morrisett, E. Mynatt, S. Patel, R. Poovendran, and B. Zorn, “Safety and security for intelligent infrastructure,” *arXiv preprint arXiv:1705.02002*, May 2017.

[88] R. de T. Caropreso, R. A. S. Fernandes, D. P. M. Osorio, and I. N. Silva, “An open-source framework for smart meters: Data communication and security traffic analysis,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 2, pp. 1638–1647, Feb. 2018.

[89] B. Vaidya, D. Makrakis, and H. Mouftah, “Secure communication mechanism for ubiquitous smart grid infrastructure,” *The Journal of Supercomputing*, vol. 64, no. 2, pp. 435–455, May 2013.

[90] T. Nghia Le, W. Chin, and H. Chen, “Standardization and security for smart grid communications based on cognitive radio technologies - a comprehensive survey,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 1, pp. 423–445, Sept. 2016.

[91] A. A. Khan, M. H. Rehmani, and M. Reisslein, “Cognitive radio for smart grids: Survey of architectures, spectrum sensing mechanisms, and networking protocols,” *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 860–898, Sept. 2015.

[92] U. S. Premarathne, I. Khalil, and M. Atiquzzaman, “Secure and reliable surveillance over cognitive radio sensor networks in smart grid,” *Pervasive and Mobile Computing*, vol. 22, pp. 3 – 15, Sept. 2015, special Issue on Recent Developments in Cognitive Radio Sensor Networks.

[93] G. Lee, Y.-S. Kim, and J. Kang, “An adaptive dos attack mitigation measure for field networks in smart grids,” in *Advances on Broad-Band Wireless Computing, Communication and Applications*, L. Barolli, F. Xhafa, and K. Yim, Eds. Cham: Springer International Publishing, Oct. 2016, pp. 419–428.

[94] B. Hu and H. Gharavi, “Smart grid mesh network security using dynamic key distribution with merkle tree 4-way handshaking,” *IEEE Transactions on Smart Grid*, vol. 5, no. 2, pp. 550–558, Sept. 2013.

[95] S. McLaughlin, D. Podkuiko, and P. McDaniel, “Energy theft in the advanced metering infrastructure,” in *Critical Information Infrastructures Security*, E. Rome and R. Bloomfield, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, Jan. 2009, pp. 176–187.

[96] Z. Zhou, “Design and implementation about secure smart electricity meter sealing based on rf tag,” *International Journal of Security and Its Applications*, vol. 9, no. 6, pp. 79–88, June 2015.

[97] H. J. Jo, I. S. Kim, and D. H. Lee, “Efficient and privacy-preserving metering protocols for smart grid systems,” *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1732–1742, July 2015.

[98] W. Han and Y. Xiao, “NFD: Non-technical loss fraud detection in smart grid,” *Computers & Security*, vol. 65, pp. 187 – 201, Mar. 2017.

[99] S.-C. Yip, K. Wong, W.-P. Hew, M.-T. Gan, R. C.-W. Phan, and S.-W. Tan, “Detection of energy theft and defective smart meters in smart grids using linear regression,” *International Journal of Electrical Power & Energy Systems*, vol. 91, pp. 230 – 240, Oct. 2017.

[100] “Smart grid security myths vs. reality,” SilverSpring Networks, Tech. Rep., Nov. 2016.

[101] I. Parvez, F. Abdul, and A. I. Sarwat, “A location based key management system for advanced metering infrastructure of smart grid,” in *2016 IEEE Green Technologies Conference (GreenTech)*, Apr. 2016, pp. 62–67.

[102] Y. Gong, Y. Cai, Y. Guo, and Y. Fang, “A privacy-preserving scheme for incentive-based demand response in the smart grid,” *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1304–1313, Mar. 2015.

[103] D. He, S. Chan, and M. Guizani, “Cyber security analysis and protection of wireless sensor networks for smart grid monitoring,” *IEEE Wireless Communications*, vol. 24, no. 6, pp. 98–103, Apr. 2017.

[104] A. Bari, J. Jiang, W. Saad, and A. Jaekel, “Challenges in the smart grid applications: An overview,” *International Journal of Distributed Sensor Networks*, vol. 10, no. 2, p. 974682, Feb. 2014.

[105] “Guidelines for smart grid cybersecurity volume 1 - smart grid cybersecurity strategy, architecture, and high-level requirements,” NIST, Gaithersburg, MD, Tech. Rep. 7628, Sept. 2014.

[106] J. Hajny, P. Dzurenda, and L. Malina, “Privacy-enhanced data collection scheme for smart-metering,” in *Information Security and Cryptology*, D. Lin, X. Wang, and M. Yung, Eds. Cham: Springer International Publishing, May 2016, pp. 413–429.

[107] H. Nicanfar, P. Talebifard, A. Alasaad, and V. C. M. Leung, “Enhanced network coding to maintain privacy in smart grid communication,” *IEEE Transactions on Emerging Topics in Computing*, vol. 1, no. 2, pp. 286–296, Nov. 2013.

[108] V. Tudor, M. Almgren, and M. Papatriantafilou, “Analysis of the impact of data granularity on privacy for the smart grid,” in *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, Nov. 2013, pp. 61–70.

[109] S. Goel, “Anonymity vs. security: The right balance for the smart grid,” *Communications of the Association for Information Systems*, vol. 36, no. 1, p. 2, Jan. 2015.

[110] J. P. Gordon, “Quantum effects in communications systems,” *Proceedings of the IRE*, vol. 50, pp. 1898–1908, 1962.

[111] I. L. Chuang, N. A. Gershenfeld, and M. Kubinec, “Experimental implementation of fast quantum searching,” *Physical Review Letters*, vol. 80, pp. 3408–3411, Apr. 1998.

[112] C. S. Calude and E. Calude, "The road to quantum computational supremacy," in *From Analysis to Visualization*, D. H. Bailey, N. S. Borwein, R. P. Brent, R. S. Burachik, J.-a. H. Osborn, B. Sims, and Q. J. Zhu, Eds. Cham: Springer International Publishing, Mar. 2020, pp. 349–367.

[113] "Quantum-computing firm opens the box," https://web.archive.org/web/20110515083848/http://physicsworld.com/cws/article/news/45960, accessed: 2023-04-13.

[114] "D-wave systems breaks the 1000 qubit quantum computing barrier," https://futurism.com/d-wave-systems-breaks-the-1000-qubit-quantum-computing-barrier, accessed: 2023-04-13.

[115] "D-wave announces d-wave 2000q quantum computer and first system order," https://web.archive.org/web/20170127044404/http://www.dwavesys.com/press-releases/d-wave%C2%A0announces%C2%A0d-wave-2000q-quantum-computer-and-first-system-order, accessed: 2023-04-13.

[116] "Ibm just made a 17 qubit quantum processor, its most powerful one yet," https://www.vice.com/en/article/wnwk5w/ibm-17-qubit-quantum-processor-computer-google, accessed: 2023-04-13.

[117] "Google moves toward quantum supremacy with 72-qubit computer," https://www.sciencenews.org/article/google-moves-toward-quantum-supremacy-72-qubit-computer, accessed: 2023-04-13.

[118] "Ces 2018: Intel's 49-qubit chip shoots for quantum supremacy," https://spectrum.ieee.org/intels-49qubit-chip-aims-for-quantum-supremacy, accessed: 2023-04-13.

[119] "Ibm unveils breakthrough 127-qubit quantum processor," https://spectrum.ieee.org/intels-49qubit-chip-aims-for-quantum-supremacy, accessed: 2023-04-13.

[120] "Ibm unveils 400 qubit-plus quantum processor and next-generation ibm quantum system two," https://spectrum.ieee.org/intels-49qubit-chip-aims-for-quantum-supremacy, accessed: 2023-04-13.

[121] "Europe's first quantum computer with more than 5k qubits launched at jölich," https://www.hpcwire.com/off-the-wire/europes-first-quantum-computer-with-more-than-5k-qubits-launched-at-julich/, accessed: 2023-04-13.

[122] M. J. Dworkin, *Sp 800-38d. recommendation for block cipher modes of operation: Galois/counter mode (GCM) and GMAC*. National Institute of Standards & Technology, Nov. 2007.

[123] Y. Nir and A. Langley, "RFC 8439: ChaCha20 and Poly1305 for IETF Protocols," June 2018.

[124] 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, Tech. Rep. 8105, Apr. 2016.

[125] W. Castryck and T. Decru, "An efficient key recovery attack on SIDH," July 2022, https://eprint.iacr.org/2022/975.

[126] E. Alkim, J. W. Bos, L. Ducas, K. Easterbrook, B. LaMacchia, P. Longa, I. Mironov, M. Naehrig, V. Nikolaenko, C. Peikert, A. Raghunathan, and D. Stebila, "FrodoKEM: Learning with errors key encapsulation," NIST, Gaithersburg, MD, Tech. Rep., Sept. 2020.

[127] R. Avanzi, J. W. Bos, L. Ducas, T. L. Eike Kiltz, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber algorithm specifications and supporting documentation (version 3.0)," NIST, Gaithersburg, MD, Tech. Rep., Jan. 2021.

[128] O. Regev, "The learning with errors problem (invited survey)," in *Proc. IEEE 25th Annual Conference on Computational Complexity*, July 2010, pp. 191–204.

[129] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," *Journal of the ACM*, vol. 60, no. 6, Nov. 2013, art. no. 43.

[130] M. R. Albrecht and A. Deo, "Large modulus ring-LWE ≥ module-LWE," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, Nov. 2017, pp. 267–296.

[131] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Designs, Codes and Cryptography*, vol. 75, no. 3, pp. 565–599, June 2015.

[132] 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 *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, Apr. 2018, pp. 353–367.

[133] "Cryptographic mechanisms: Recommendations and key lengths," Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, Germany, Tech. Rep. BSI TR-02102-1, Jan. 2023.

[134] "Introducing quantum-safe crypto TLS for IBM key protect," https://www.ibm.com/cloud/blog/introducing-quantum-safe-crypto-tls-for-ibm-key-protect, accessed: 2021-02-22.

[135] J. Howe, M. Martinoli, E. Oswald, and F. Regazzoni, "Exploring parallelism to improve the performance of FrodoKEM in hardware," *Journal of Cryptographic Engineering*, vol. 11, no. 4, pp. 317–327, Feb. 2021.

[136] 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 Conference on Computer and Communications Security*, Oct. 2016, pp. 1006–1018.

[137] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange: A new hope," in *Proc. 25th USENIX Security Symposium*, Aug. 2016, pp. 327–343.

[138] "FrodoKEM practical quantum-secure key encapsulation from generic lattices," https://frodokem.org/, accessed: 2022-08-22.

[139] T. Instruments, "Tiva TM4C129ENCPDT Microcontroller," June 2014. [Online]. Available: https://www.ti.com/lit/pdf/spms441

[140] Xilinx, "Zynq-7000 SoC Data Sheet: Overview," Jul. 2018. [Online]. Available: https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf

[141] M. J. Dworkin, “SHA-3 standard: Permutation-based hash and extendable-output functions,” NIST, Gaithersburg, MD, Tech. Rep. 202, Aug. 2015.

[142] G. Bertoni, J. Daemen, M. Peeters, and G. Assche, “Sponge functions,” in *Proc. ECRYPT Workshop on Cryptographic Hash Functions*, May 2007, pp. 1–22.

[143] Xilinx, “AXI-MM memory-mapped interface,” Aug. 2022. [Online]. Available: https://docs.xilinx.com/r/en-US/pg231-v-proc-ss/AXI-MM-Memory-Mapped-Interface

[144] 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 International Workshop on Advances in Sensors and Interfaces*, June 2019, pp. 194–199.

[145] E. Fujisaki and T. Okamoto, “Secure integration of asymmetric and symmetric encryption schemes,” in *Annual international cryptology conference*. Springer, June 1999, pp. 537–554.

[146] WEG. (2022) Intelligent power meters – SMW series. [Online]. Available: https://www.weg.net/catalog/weg/BR/en/Generation%2CTransmission-and-Distribution/Intelligent-Power-Meters/SWM-Series/Intelligent-Power-Meters---SMW-Series/p/MKT_WTD_SMART_METERS_SMW

[147] P. Matoušek, “Analysis of DLMS protocol,” *Brno University of Technology*, Dec. 2017.

[148] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer, “Ascon,” *Submission to the CAESAR competition: https://ascon.iaik.tugraz.at/*, Jan. 2014.

APPENDIX A – List of Publications

The list of papers related to the dissertation published or submitted during the doctoral period is as follows:

- **V. L. R. da Costa**, J. López, and M. V. Ribeiro, "A SoC Implementation of a PQC Scheme for Smart Meter," in *XXXIX Brazilian Symposium on Telecommunications and Signal Processing*, Sept. 2021, pp. 1-5.

- **V. L. R. da Costa**, J. López, and M. V. Ribeiro, "A System-on-a-Chip Implementation of a Post-Quantum Cryptography Scheme for Smart Meter Data Communications," *Sensors*, vol. 22, pp. 7214-7235, Jun. 2022.

- **V. L. R. da Costa**, Â. Camponogara, J. López, and M. V. Ribeiro, "The Feasibility of the CRYSTALS-Kyber Scheme for Smart Metering Systems," *IEEE Access*, vol. 10, pp. 131303-131317, Dec. 2022.

- **V. L. R. da Costa**, Leonardo de M. B. A. Dib, Mateus de L. Filomeno, J. López, and M. V. Ribeiro, "A Quantum-Secure Advanced Metering Infrastructure," (to be submitted).

The list of papers non-related to the dissertation published or submitted during the doctoral period is as follows:

- L. G. da. S. Costa, G. R. Colen, A. C. M de Queiroz, **V. L. R. da Costa**, U. R. C. Vítor, F. V. dos Santos, and M. V. Ribeiro, "Access impedance in brazilian in-home, broadband and low-voltage electric power grids," *Electric power systems research*, vol. 171, no. 19, pp. 141-149, Sept. 2019.

- **V. L. R. da Costa**, V. Fernandes, and M. V. Ribeiro, "Narrowband hybrid PLC/wireless: Transceiver prototype, hardware resource usage and energy consumption," *Ad Hoc Networks*, vol. 94, pp. 101945, Nov. 2019.

- L. G. da. S. Costa, A. C. M de Queiroz, **V. L. R. da Costa**, and M. V. Ribeiro, "An Analog Filter Bank-based Circuit for Performing the Adaptive Impedance Matching in PLC Systems," *Journal of Communication and Information Systems*, vol. 36, no. 1, pp. 133-150, Aug. 2021.

- L. G. da. S. Costa, **V. L. R. da Costa**,Â. Camponogara, and M. V. Ribeiro, “An Initial Discussion of an Adaptive Impedance Matching Circuit for PLC Systems,” in *XXXIX Brazilian Symposium on Telecommunications and Signal Processing*, Sept. 2021, pp. 1-4.

- R. Pacheco, D. Braga, I. Passos, T. Araújo, **V. L. R. da Costa**, and M. Coutinho, “Libharpia: a New Cryptographic Library for Brazilian Elections,” in *XXII Brazilian Symposium on Information and Computational Systems Security*, Sept. 2022, pp. 250-263.