A SoC Implementation of a PQC Scheme for Smart Meter

Vinícius L. R. da Costa, Julio López, and Moisés V. Ribeiro

***Abstract*— This paper focuses on the implementation of quantum-secure lattice-based cryptography as a key encapsulation mechanism for a smart meter device, which has hardware resource constraints. In this regard, the post-quantum cryptography scheme called FrodoKEM, a candidate to the NIST post-quantum standardization process, is implemented using a system-on-a-chip (SoC) device. Numerical results show that the processing time to run the FrodoKEM scheme implemented on an SoC device reduces in comparison to the benchmark implementation. In the end, we conclude that this SoC-based implementation of the FrodoKEM scheme may fit the hardware constraints related to smart meter devices.**

***Keywords*— post-quantum cryptography, smart grid, smart meter, field programmable gate array.**

I. Introduction

The development of Smart Grid (SG) technologies and solutions has become a priority in the electricity sector because SG is a powerful concept to ensure energy reliability and efficiency, to reduce technical and non-technical losses, to enable the integration of different types of renewable energy sources [1], and to fulfill current and future demands of all stakeholders in the electricity sector.

Moreover, the upgrade of the existing technologies in electric power systems is offering a new perspective for accelerating the implementation of the SG concepts [2]. In this scenario, Smart Meters (SMs) had emerged as a vital enabling technology to SGs because they are capable of monitoring and controlling the consumption of electricity and generating data that are useful for utilities, regulators, consumers and prosumers, beyond the traditional unidirectional power billing functionality. Indeed, data obtained from SMs are of great value for effective and dynamic energy planning that can enhance the stability, efficiency, and reliability of electric power systems.

Furthermore, serious security and privacy concerns had emerged because SM data may be exposed to different attacks that could be launched from different parties even from the customer side. For instance, key generation and distribution in cryptography schemes may result in a security breach when SM data are transmitted [3]. In this context, research efforts have been focused on privacy of information, security issues, and countermeasures, see [4] and reference therein. Regarding the reported research efforts, only classical computer attacks, in which the cryptographic scheme is based on the hardness of integer factoring assumption or the discrete logarithm problem (e.g., Rivest-Shamir-Adleman (RSA) and Elliptic Curve Cryptography (ECC)), are investigated to ensure the security of SM data traveling through data networks. However, such kind of cryptographic schemes will be easily broken when a sufficiently powerful and stable quantum computer runs Shor's algorithm, which is a polynomial-time quantum algorithm for integer factorization and discrete logarithms [5].

In this regard, this paper investigates the feasibility of using Post-Quantum Cryptography (PQC) as a key encapsulation mechanism (KEM) for securing data communication in SMs, which is a hardware constraint device. To do so, we choose the FrodoKEM scheme [6] to perform this analysis due to its well-studied and conservative approach, which relies on the Learning With Errors (LWE) problem [7]. Also, we evaluate the feasibility of its implementation in a hardware constraint device (i.e., SMs) and investigate if the presence of an Field Programmable Gate Array (FPGA) can bring significant performance improvement. Experimental results show the use of a System-on-a-Chip (SoC) devices can remarkably reduce total processing time in comparison to the benchmark (pure software) implementation.

II. Problem Formulation

Based on the previous discussion, security and privacy are among the main challenges faced by SM technologies as sensitive data are constantly produced and exchanged among consumers, prosumers, and utilities. Energy consumption and other data collected from SMs are sensitive because they are private information of consumers or prosumers from which their lifestyles and economic status can be leaked and, consequently, the concern about breaches of security in SM systems have been attracted huge attention from all involved stakeholders. The design of privacy-preserving solutions that enable SMs to perform billing, operations, and value-added services are of utmost importance because different requirements apply to these applications. Besides, SMs data are exchanged over Internet Protocol (IP)-based and dedicated data networks to the Meter Data Management System (MDMS) and will be over non-dedicated one (i.e., Internet) in the near future and, consequently, the risk of breach of security will be outweighed.

This study was supported in part by Universidade Federal de Juiz de Fora (UFJF), in part by Universidade Estadual de Campinas (UNICAMP), in part by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) under Grant 001, in part by the Ministério da Educação (MEC) under Grant #852.893/2017, in part by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), in part by Instituto Nacional de Energia Elétrica (INERGE), and in part by Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG).

Vinícius L. R. da Costa and Moisés V. Ribeiro, Electrical Engineering Department, Federal University of Juiz de Fora (UFJF), Juiz de Fora, MG, Brazil, e-mail: {vinicius.lagrota, mribeiro}@engenharia.ufjf.br)

Julio López, Institute of Computing, University of Campinas (UNICAMP), Campinas, SP, Brazil, e-mail: jlopez@ic.unicamp.br

The security of SM systems involves several aspects that are related to consumers, prosumers, and data networks [8]. In fact, it is well-established that data networks must present protections against threats to system-level security (e.g., credential compromise, denial of service attacks), threats to services (e.g., SMs cloning, location fraud), and threats to privacy (e.g., interception/eavesdroppers, misuse of private data). In this context, cryptography algorithms have emerged as one of the most powerful methods to protect SM data against eavesdroppers [9], [10] when they are transmitted through data networks.

The vast majority of cryptographic schemes rely on the hardness of large integer factoring assumption or the discrete logarithm problem due to its simplicity and lightweight implementation. Data networks relying on these schemes are secure nowadays because a classical computer can easily perform them; however, it is very difficult for classical computers to undo them, preventing unauthorized disclosure and data breaches. This scenario will significantly change with the advent of powerful quantum computers, which running Shor's factoring algorithm, can easily solve the aforementioned problems. Consequently, an eavesdropper, which makes use of a quantum computer, will be able to access sensitive data transmitted through data networks serving SM systems and to bring severe consequences for the stakeholders in the energy sector.

Fig. 1 illustrates the scenario that is addressed in this paper. Basically, the focus is on the security of SM data communication, which is transmitted in the Advanced Metering Infrastructure (AMI) using the Wide Area Network (WAN) to the MDMS, against eavesdroppers that are capable of performing sniffer attack. As already said, although there is still no feasible quantum computer, nowadays SM data communication might be deciphered in the future if stored, revealing customers' sensitive data. In this sense, we need to study PQC schemes for the current and future security of SM data communications. A PQC scheme is more complex than the current public-key systems, such as RSA and ECC, which requires more processing power and might be unfeasible for hardware constrained devices, such as SMs. One approach to overcome these hardware constraints is to use a SoC device, enabling a hardware/software implementation.

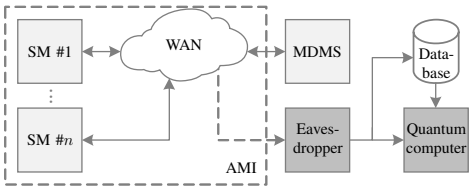

Fig. 1. A typical security breach scenario in which an attacker performs a sniff attack and uses a quantum computer to decipher the SM data overheard.

A few theoretical works focused on SG security using PQC are introduced in the literature [11]. Other works have also evaluated PQC schemes in software, hardware, and hardware/software [12]. However, in this work we focus on an implementation and evaluation of a PQC scheme on a constrained SM device. We present practical results, data security considerations (e.g.: billing, personal information, etc.), and integration with a complex system as SGs.

III. Benchmark of the FrodoKEM Scheme

In this section, it is summarized the reference implementation of the FrodoKEM scheme on an Advanced RISC Machine (ARM) Cortex-A9 device, presented in the Zynq-7000 SoC device of the MicroZed board. The FrodoKEM scheme can be basically divided into three algorithms: key pair generation, encapsulation, and decapsulation [6].

A. Analysis of the algorithm of FrodoKEM scheme

To identify the most time-consuming routines of the software implementation of FrodoKEM, it is executed the code from [13] on an ARM Cortex-A9 processor presented in the MicroZed board. After a detailed analysis, three operations stood out as the most time-consuming. For the key pair generation, it is required the matrix operation $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$, which is computational intensive because of the large size of the matrices. For the same reason, the operation $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$ is even more expensive, because it is used in both encryption and decryption algorithms. Finally, the most expensive operation is the computation of the Secure Hash Algorithm and KECCAK 128 (SHAKE128) hash function, member of the Secure Hash Algorithm 3 (SHA-3). It is used in all three algorithms and is also expensive due to its loop-based structure. All operations demand relevant computational resources because of the large size of FrodoKEM keys and the use of a public key or a private key, directly or indirectly. Table I summarizes the relative time consumption of FrodoKEM-640 in the aforementioned processor.

TABLE I
Relative time consumption of FrodoKEM-640, in percentage

Operation	(%)
$\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$	8.22
$\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$	30.94
SHAKE128	53.90
Others	6.94

IV. Proposed Implementation of the FrodoKEM Scheme

This section aims to present the proposed implementation of the FrodoKEM scheme using a hardware/software implementation, seeking to use the advantages of the combination of the ARM Cortex-A9 processor and the FPGA part of the Zynq-7000 SoC device. In this regard, the strategy is to implement the most time-consuming routines in hardware, already identified in the Subsection III-A (i.e., matrix-matrix multiplications and SHAKE128), and keep the software part (including the three operations) as close as possible to the original for ensuring fairness in further comparisons. The architecture can be divided into three main instances: Processing System (PS), Interconnect, and Programmable Logic (PL).

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 PS and PL using the AXI-MM protocol. Finally, the PL, based on an FPGA, is where the implementation of 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 presented in Table I, it is implemented the most time-consuming operations in hardware using an FPGA to accelerate the execution of the FrodoKEM scheme as follows:

A. The implementation of operation $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$

The operation $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$, which consumes more than 8% of the total time, has its matrix-matrix multiplication implemented in hardware. 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 matrix $\mathbf{E}$ from PS to PL is taken into account.

Fig. 2 shows the hardware schematic of the operation $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$. The schematic consists of four main instances: Block Random Access Memorys (BRAMs) of matrix $\mathbf{A}$, BRAMs of matrix $\mathbf{S}$, a multiplier instance, and BRAMs of matrix $\mathbf{AS}$. Note that BRAMs are Block Random Access Memory where larger amount of data are stored.

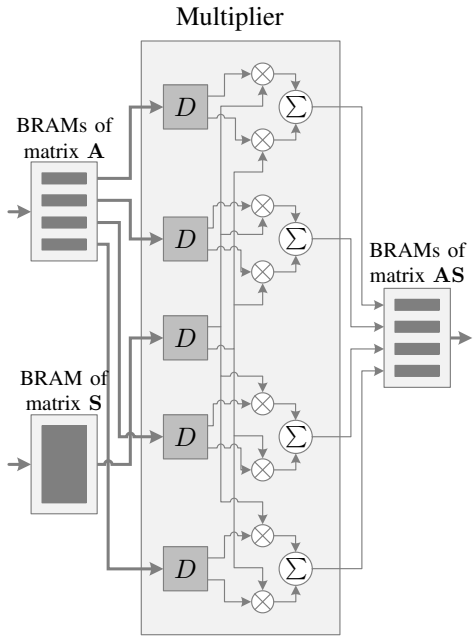

Fig. 2. Schematic representation of $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$. Thick arrows represent 32-bits buses and thin arrows 16-bits buses. Control signals have been omitted.

In order to save resources, the matrix $\mathbf{A}$ is generated on-the-fly. Four rows are generated and transmitted from the PS to the BRAMs of 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. Aiming to speed up the transfer process, 32-bits are transferred at a time, which means that two subsequent values are concatenated and stored in the BRAMs. These values are further separated using the block D. The matrix $\mathbf{S}$ is transmitted from PS to PL. At this time, the whole matrix is transferred since it has already been fully generated in PS. The transmission process follows the same 32-bits transmit principle as the matrix $\mathbf{A}$, although, for the matrix $\mathbf{S}$ a 2560×32-bits BRAM is used. When both matrices are completely stored in their BRAMs, the hardware implemented reads the BRAMs and processes the data, saving the results in the BRAMs of the matrix $\mathbf{AS}$. New data in the BRAMs of matrix $\mathbf{A}$ are received until all data are processed.

B. The implementation of operation $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$

Responsible for more than 30% of the time consumption, the operation $\mathbf{B'} \leftarrow \mathbf{S'A} + \mathbf{E'}$ also has its matrix-matrix multiplication implemented in hardware. 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 Fig. 3, it is shown the schematic representation of the operation $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$. It also has four instances: BRAMs of matrix $\mathbf{S'}$, BRAMs of matrix $\mathbf{A}$, a multiplier instance, and BRAM of matrix $\mathbf{S'A}$.

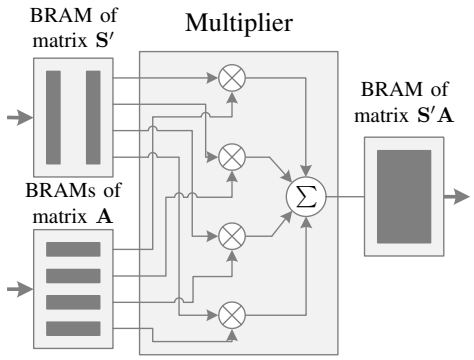

Fig. 3. Schematic representation of $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$. Thick arrows represent 32-bits buses and thin arrows 16-bits buses. Control signals have been omitted.

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 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 operation $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$: four 320×32-bits BRAMs and 32-bits bus for transfer. After the matrices are received, the hardware implemented reads the BRAMs and processes the data, saving the results in the BRAMs of matrix $\mathbf{S'A}$. New data in the BRAMs of matrices $\mathbf{A}$ and $\mathbf{S'}$ are received until all data are processed. Note that in this hardware implementation, only half of matrix $\mathbf{S'}$ is stored at a time, showing that there are alternatives to save resources at the expense of performance.

C. The implementation of SHAKE128 hash function

FrodoKEM's most time-consuming operation is the SHAKE128 hash function, which is responsible for almost

54% of the total time consumption. The proposed implementation can be divided into three instances: a system BRAM, the SHAKE128 instance, and a Keccak-f[1600], as illustrated in Fig. 4. The BRAM system 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 and output values received/transmitted 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 BRAM.

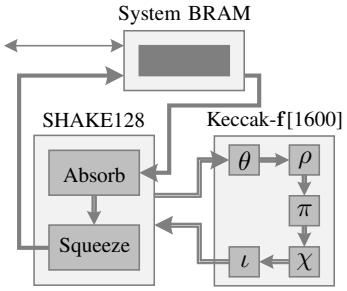

Fig. 4. Schematic representation of SHAKE128. Thick arrows represent 64-bits buses, thin arrows represent 32-bits buses, and double arrows 1600-bits buses. Control signals have been omitted.

When all values are received and stored in BRAM, 168-bytes are sent to the SHAKE128 instance via a 64-bits bus, starting the absorb phase. These bytes are used to construct the internal state. If necessary in the absorb phase, the Keccak-f[1600] function is called and the entire internal state is sent to the Keccak-f[1600] function, which performs its five steps (θ, ρ, π, χ, and ι) in a single clock. Then, the new scrambled internal state returns to the SHAKE128 instance. 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] function. In order to save resources, the firsts 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.

V. Performance Evaluation

The FPGA timing comparison between implementations of FrodoKEM are discussed in this section. The focus is a comparative analysis of the operations $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$ and $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$, and SHAKE128 hash function, in terms of software (i.e., benchmark) and hardware/software (i.e., proposed) implementation. The processing time results presented in Table II were acquired using a dedicated timer implemented in the PL and compare the benchmark implementation timing with the proposed implementation.

As we can see in Table II, when $\mathbf{AS} \leftarrow \mathbf{A} \times \mathbf{S}$, $\mathbf{S'A} \leftarrow \mathbf{S'} \times \mathbf{A}$, and SHAKE128 are implemented in hardware, the relative time improvement achieved is 68.15%. Therefore, when hardware-acceleration is used, the three operations can be performed with less than one-third of the time required by the fully-software implementation, dropping from more than 1700 ms to less than 560 ms of processing time.

TABLE II
Total Processing Time and Performance

	Benchmark	Proposed
Key pair generation (ms)	467.25	146.35
Encapsulation (ms)	644.74	205.11
Decapsulation (ms)	645.35	208.26
Total (ms)	1757.33	559.72
Relative time improvement	-	68.15%

VI. Conclusion

This work has investigated the feasibility of implementing a PQC scheme in a hardware-constrained SM device for ensuring the security of SM data traveling through data networks. Numerical results showed that the proposed implementation of the most time-consuming and complex routines of FrodoKEM in hardware offers a one-third reduction of the processing time in comparison to the benchmark implementation of it. The timing results show that it is possible to implement the FrodoKEM scheme in a hardware-constrained device, such as SMs. When an FPGA device is available to run the most time-consuming routines, a significant improvement can be achieved. Last but not least, the attained results constitute a baseline for the adoption of PQC schemes in SM systems.

References

[1] 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., 2007.

[2] 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, Apr. 2018.

[3] 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.

[4] Y. Liu, W. Guo, C. Fan, L. Chang, and C. Cheng, "A practical privacy-preserving data aggregation (3PDA) scheme for smart grid," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 3, pp. 1767–1774, Feb. 2018.

[5] 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.

[6] 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., 2020.

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

[8] R. Rashed Mohassel, A. Fung, F. Mohammadi, and K. Raahemifar, "A survey on advanced metering infrastructure," *International Journal of Electrical Power & Energy Systems*, vol. 63, pp. 473–484, Dec. 2014.

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

[10] ——, "Physical layer security of in-home PLC systems: Analysis based on a measurement campaign," *IEEE Systems Journal*, pp. 1–12, June 2020.

[11] 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, 2019.

[12] H. Nejatollahi, R. Cammarota, and N. Dutt, "Flexible ntt accelerators for rlwe lattice-based cryptography," Abu Dhabi, United arab emirates, 2019, pp. 329 – 332. [Online]. Available: http://dx.doi.org/10.1109/ICCD46524.2019.00052

[13] 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, 2016.