

Implementation and Performance Evaluation of ML-KEM-512 for Resource-Constrained Internet of Medical Things Devices

Logeshwari Radhakrishnan
Post Doctoral Researcher
Lincoln University College
Malaysia 47301
SRM Institute of Science and
Technology
Kattangulathur, Chennai, India
pdf.logeshwari@lincoln.edu.my

Dr. L. Anand
Associate Professor
Department of Networking and
Communications
SRM Institute of Science and Technology
Kattangulathur, Chennai, India
anandl@srmist.edu.in

Abstract

With the growing use of resource constrained Internet of medical things (IoMT) devices has created a need for secure communication method that can withstand future attacks from quantum computer while still providing a practical performance. This paper presents the implementation and evaluation of ML-KEM-512, a post quantum key encapsulation mechanism, for secure key establishment in IoMT environments. The implementation was tested on a desktop based and an ARM Cortex-M4-base platform using different compiler optimization levels. The study evaluates the execution time of key generation, encapsulation, and decapsulation on the desktop platform, along with executable size and memory usage on both platforms. The findings indicate that compiler optimization has a significant impact on both the execution time and the size of the program code for ML-KEM-512. The implementation has been successfully verified on both platforms, the agreement of the shared secrets proving that the key establishment process was carried out correctly. These results thus form a practical foundation for further assessing ML-KEM-512 for use in secure communication on resource-limited IoMT devices.

Keywords

ML-KEM-512; Post-Quantum Cryptography; Internet of Medical Things; Key Encapsulation Mechanism; Resource-Constrained Devices

Introduction

The Internet of Medical Things (IoMT) is becoming an important part of modern healthcare. Connected medical devices can collect patient information, support remote monitoring, and exchange data with healthcare systems in real time. Although these capabilities improve healthcare services, they also create security concerns because the information handled by such devices is often sensitive. In addition, many IoMT devices have limited processing power, memory, storage and energy resources. As a result, security mechanisms must be carefully selected so that they provide adequate protection without placing excessive demands on the devices.

Most existing secure communication systems use conventional public-key cryptographic techniques to establish shared secret keys. These techniques are effective against currently known computational attacks, however advances in quantum computing may threaten the long-term security of these algorithms. A sufficiently powerful quantum computer could compromise widely used public-key algorithms [1], putting confidential medical information and communication systems at risk. This possibility highlights the needs for cryptographic solutions that can provide protection against both present-day and future quantum-based attacks.

Post-quantum cryptography (PQC) focuses on developing algorithms that remain secure even in the presence of quantum computers [1]. ML-KEM-512, a module-lattice -based key encapsulation mechanism [2], is one such approach intended for establishing shared secret keys between communicating parties. However, its practical use in IoMT environments requires detailed evaluation. The limited resources of embedded medical devices may affect execution time, memory requirements, and overall efficiency of cryptographic operations.

This paper investigates the implementation and performance evaluation of ML-KEM-512 for security establishment in resource constrained IoMT devices. The implementation is evaluated on a desktop and an ARM Cortex-M4-based platform using different compiler optimization levels. The study compares observed performance across these platforms to determine the practicality of deploying ML-KEM-512 in constrained environment. The findings are intended to support development of post quantum secure communication systems for connected healthcare devices and other embedded Internet of Things application.

Related Work

The development of secure communication systems for connected medical devices has traditionally relied on established public-key cryptographic techniques. These methods have been widely used for authentication, key exchange, and protecting data transmitted between devices. However, their long-term security is increasingly being questioned because sufficiently powerful quantum computers could compromise many of the mathematical problems on which conventional public-key algorithms depend.

To address this concern, researchers have explored post quantum cryptography (PQC), which aims to provide security against both classical and quantum- enabled attacks. Several cryptographic approaches have been investigated, including lattice-based, code-based, hash-based and multivariate cryptographic schemes.[6] Among these approaches, lattice-based cryptography has received considerable attention because it offers practical mechanism for key establishment and encryption while maintaining a balance between security and computational efficiency.

The development of standardised post quantum algorithm has further encouraged research into their use in embedded on Internet of Things environment. However, applying these algorithms to resource constrained devices remains challenging. Medical and IoT devices often have limited processing capacity, memory, and energy availability. Consequently, an algorithm that performs well on a desktop computer may introduce noticeable overhead when implemented on a microcontroller-based platform [5].

ML-KEM, formally associated with the Kyber family of algorithm is a lattice-based Key encapsulation mechanism designed for establishing a shared secret between two communicating parties.[4] The ML-KEM-512 Parameter set provides a potential foundation for post quantum secure communication in IoMT systems. Nevertheless, its suitable practical deployment depends not only on its security properties but also on execution time implementation requirements and performance on embedded hardware.

Previous studies have examined post quantum algorithms from the perspective of security, standardisation, and software performance. However, further practical evaluation is required for resource constrained medical and IoT devices particularly on ARM-based microcontroller platforms. [7],[8],[10] This work addresses that need by implementing ML-KEM-512 and evaluating its performance on both a desktop system and an ARM cortex dash M4 platform under different compiler optimization levels. The comparison provides a clear understanding of the computational requirements associated with using ML-KEM-512 for security establishment in constraint IoMT environments

Work or approach	Cryptographic category	Quantum-resistant design	Focus on constrained devices	Performance evaluation on ARM Cortex-M4
Conventional public-key cryptography	Classical public-key cryptography	No	Limited	Not considered
Lattice-based PQC approaches	Post quantum cryptography	Yes	Considered in several studies	Not necessarily evaluated
ML-KEM-related implementations	Lattice-based key encapsulation	Yes	Yes	Depends on the implementation
This work	ML-KEM-512	Yes	Yes	Yes

Key Contributions

The main contribution of this paper is the practical implementation and performance evaluation of ML-KEM-512 for secure key establishment in the resource-constrained Internet of Medical Things (IoMT) devices. Instead of focusing only on the theoretical security of post-quantum cryptography, this work evaluates how ML-KEM-512 performs across computing platforms with different resource limitations

The key contributions of this study are as follows:

1. Implementation of ML-KEM-512:
ML-KEM-512 is implemented as a post quantum key establishment mechanism for IoMT communication system.
2. Evaluation across different platforms:
The implementation is tested on a desktop system and an ARM-Cortex M4-based platform to understand its behaviour on both conventional and resource-constrained hardware.
3. Effect of compiler optimization:

The performance of the implementation is examined under different compiler optimization levels to observe their effect on execution efficiency.

4. Platform-based performance comparison:

The study compares the results obtained from the desktop and arm platforms providing, practical insight into computational requirements of ML-KEM-512 on embedded devices.

5. Assessment of IoMT applicability:

The findings contribute to the existing research on post-quantum security by examining the feasibility of using ML-KEM-512 for secure key establishment in connected medical devices and other constrained IoT systems.

Overall, this works adds practical performance observations to the existing body of knowledge and helps identify the suitability of ML-KEM-512 for future post-quantum-secure IoMT applications.

Method, Experiments and Results

Methodology

The study implements ML-KEM-512 for secure establishment in resource-constraint Internet of Medical Things devices using PQClean implementation [9]. The implementation was first tested on desktop system running Ubuntu x86_64 and subsequently cross compiled for an ARM Cortex-M4-based embedded platform.

The experiments focused on functional verification, compiler optimization, execution time, executable size, and memory utilization. On the desktop platform, the implementation was evaluated using GCC optimization levels -O0, -O1, -O2, and -Os. The ARM Cortex-M4 implementation was tested using -O0, -O1, -O2, and -Os. The results from both platforms were compared to assess the feasibility of deploying ML-KEM-512 in IoMT environments [10],[11].

Desktop Experiments and Results

The desktop experiment measured the execution time of three ML-KEM-512 operations: key generation, encapsulation and decapsulation. The results for the different compiler optimizations level are presented in Table 1

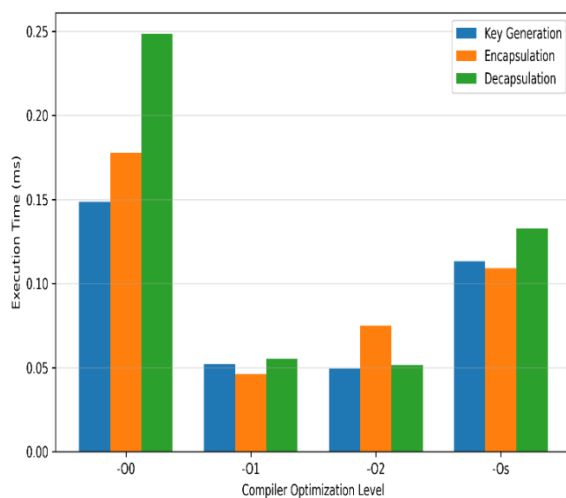


Figure 1. Execution time of ML-KEM-512 operations on Ubuntu x86_64 under different compiler optimization level

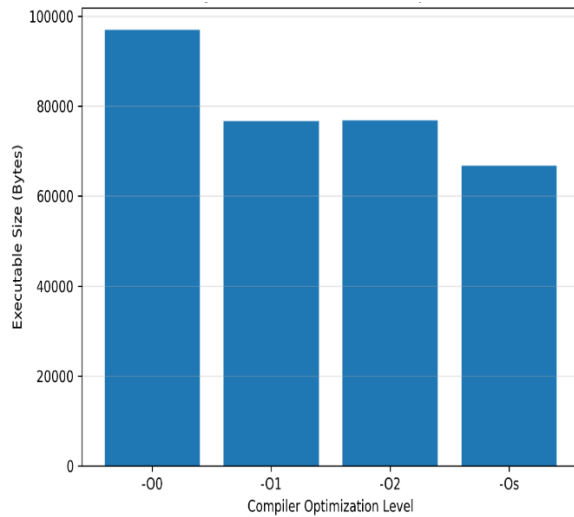


Figure 2. Execution size of MLKEM 512 operations on Ubuntu_x86_64 under different compiler optimization level

Figure 3. Memory Size of MLKEM 512 operations on Ubuntu x86_64 under different compiler optimization level

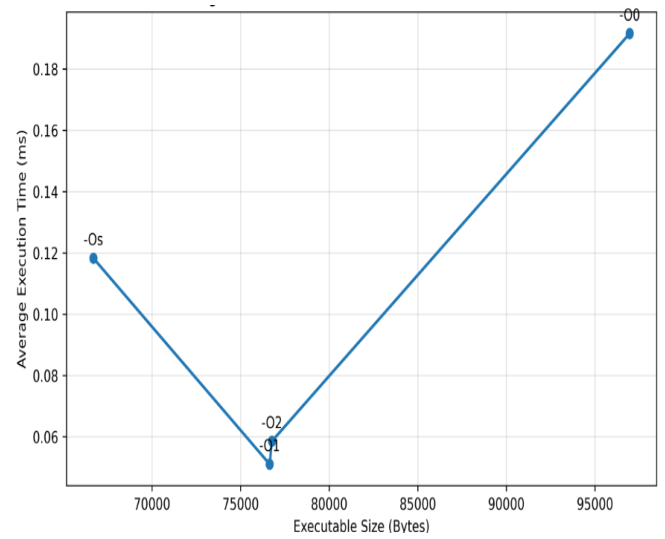
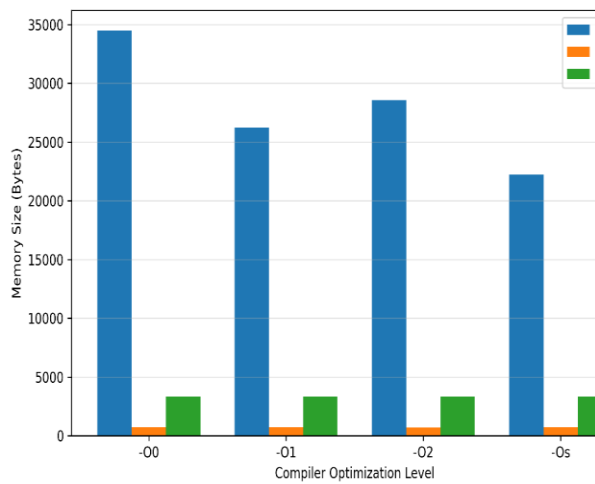


Figure 4. Performance vs Code size Trade-off



Optimization	Key Generation (ms)	Encapsulation (ms)	Decapsulation (ms)	Executable Size (Bytes)	.text	.data	.bss
-O0	0.148612	0.177627	0.248512	96,959	34,493	728	3,328
-O1	0.052159	0.046058	0.055094	76,641	26,233	712	3,328
-O2	0.049443	0.074972	0.051397	76,769	28,585	704	3,328
-Os	0.113145	0.109037	0.132762	66,691	22,217	712	3,328

Table 1. Desktop performance and memory results

The result shows that The unoptimized implementation, -O0, recorded the highest execution times for all the three operations. The -O1 configuration produced the lowest encapsulation time, while -O2 the lowest key-generation and decapsulation times. In the terms of memory usage -Os generated the smallest executable and the smallest .text section. The .bss section remained constant across all the tested optimization levels. These results indicate that compiler optimization has a clear effect on both execution performance and executable size.

ARM Cortex-M4 Experiments and Results

The implementation was then cross compiled for the ARM Cortex-M4 platform to the to evaluate its suitability for resource-constrained embedded deployment. Since the ARM experiment focused on the resource utilization, the main parameters considered were the .text, .data and .bss sections, total image size, ELF size and BIN size.

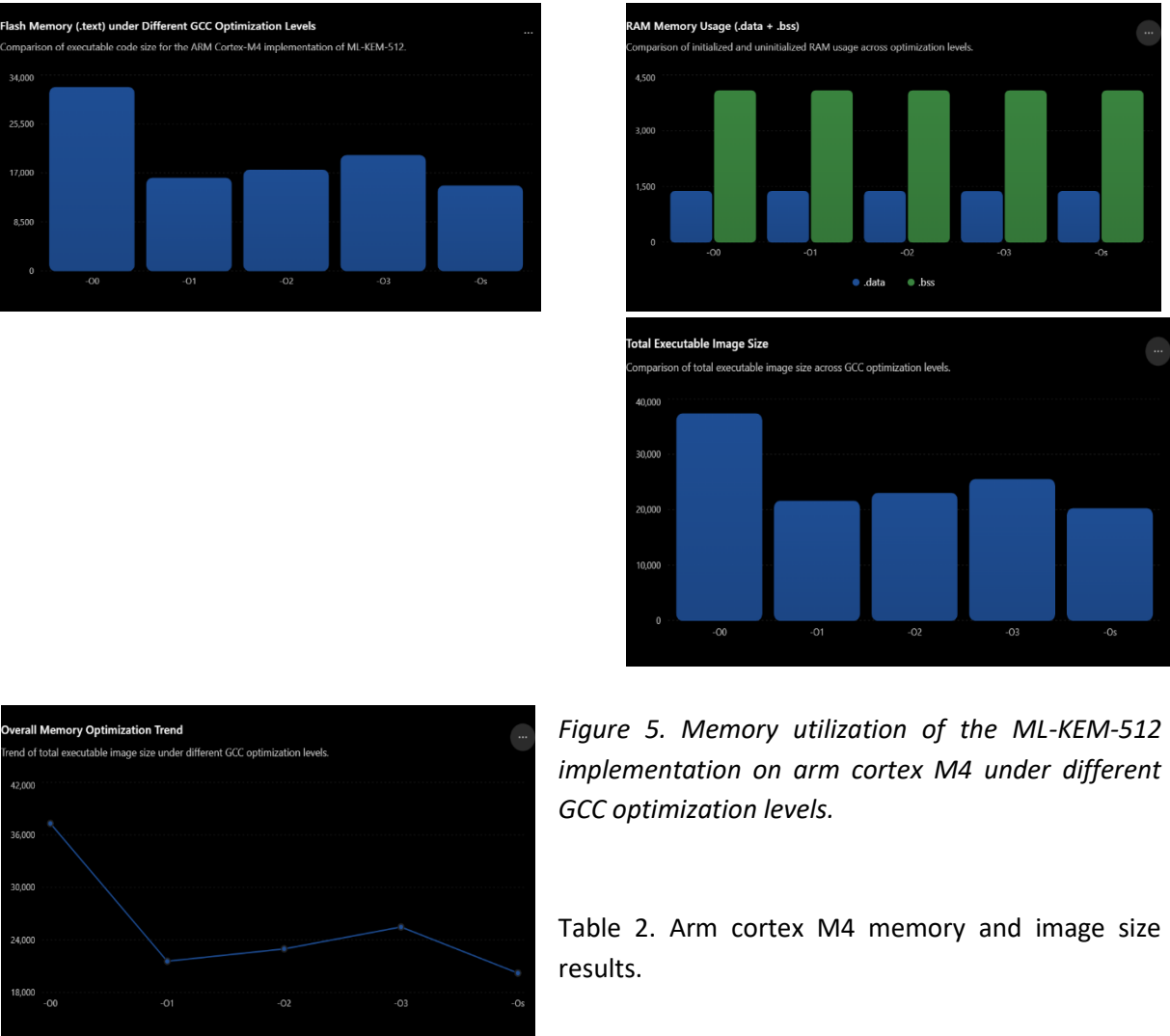


Figure 5. Memory utilization of the ML-KEM-512 implementation on arm cortex M4 under different GCC optimization levels.

Table 2. Arm cortex M4 memory and image size results.

GCC Optimization	.text (Bytes)	.data (Bytes)	.bss (Bytes)	Total Image (Bytes)	ELF Size	BIN Size
------------------	---------------	---------------	--------------	---------------------	----------	----------

-O0	31,864	1,372	4,080	37,316	160 KB	83 KB
-O1	16,100	1,372	4,080	21,552	145 KB	83 KB
-O2	17,520	1,372	4,080	22,972	146 KB	83 KB
-O3	20,032	1,368	4,080	25,480	155 KB	83 KB
-Os	14,752	1,372	4,080	20,204	143 KB	83 KB

The -O0 configuration produced the largest .text section and the highest total image size. The -Os configuration provided the most compact implementation, with the smallest .text section and total image size. The .bss value remained unchanged at 4,080 bytes, while the .data section varied only slightly. The ELF size has lowest with -Os, whereas the BIN size remained constant across all the optimization levels. These results show that size-oriented optimization is particularly useful when deploying ML-KEM-512 on memory- constrained embedded platforms.

Comparative Analysis

The desktop and ARM experiments serve different but complementary purposes. The Ubuntu x86_64 platform was used mainly to measure the execution time of key generation, encapsulation and decapsulation. In contrast, the ARM Cortex-M4 platform was used to examine memory requirements and confirm the generation of embedded plat deployment files.

Functional verification was successful on both platforms, and the generated shared secret matched, confirming that key establishment operations were executed correctly. The successful generation of ARM ELF and BIN files also demonstrated that the implementation could be prepared for embedded deployment. The comparative results are summarized in Table 3.

Table 3. Comparison of desktop and arm cortex M4 evaluations.

Evaluation Parameter	Desktop (Ubuntu x86_64)	ARM Cortex-M4	Outcome
Implementation Objective	Functional validation and performance benchmarking	Embedded deployment feasibility	Complementary evaluation environments
Functional Verification	Successful	Successful	ML-KEM-512 operations executed correctly
Shared Secret Verification	Match	Match	Correct key establishment achieved

Execution Time Analysis	Measured (KeyGen, Encapsulation, Decapsulation)	Not measured	Desktop used for runtime performance evaluation
Compiler Optimization	-O0, -O1, -O2, -Os	-O0, -O1, -O2, -O3, -Os	Optimization impact evaluated on both platforms
Executable / Memory Analysis	Executable size and memory sections analyzed	Flash (.text), RAM (.data + .bss), ELF size analyzed	Resource utilization evaluated
Deployment Validation	Native Linux executable	ARM ELF/BIN generated successfully	Confirms embedded deployment feasibility
Overall Outcome	Performance validation	Resource-constrained implementation validation	Proposed framework is suitable for IoMT deployment

Overall the experiments show that ML-KEM-512 can be functionally implemented across both desktop and ARM Cortex-M4 environments. The desktop results highlight the effect of compiler optimization on execution time, while the ARM results demonstrate the importance of code size optimization for embedded systems.

Discussion

The experimental results show that performance of ML-KEM-512 is strongly influenced by the compiler optimization level and the targeted platform. On the Ubuntu x86_64 system, the unoptimized configuration -O0 produced the highest execution time of for key generation, encapsulation and decapsulation. This indicates that the compiler optimization is important for improving the runtime performance of ML-KEM-512.

The results also show that no single optimization level provided the best outcome for every parameter [11]. The -O1 configuration is achieved the lowest encapsulation time while the -O2 provide the most the lowest key generation and decapsulation times in contrast -Os produce the smallest executable and smallest .text section the difference is important for IoMT applications because a configuration that improves execution speed may not always provide the lowest memory requirement. Therefore, choice of optimization level should depend on whether the target device prioritises processing speed program size or a balance of both.

The ARM Cortex-M4 results further highlight the importance of both size optimization embedded environments [10]. The -O0 configuration generated the largest .text section and the highest total image size. On the other hand -Os produced the smallest tech section and the lowest total limit size. Since the embedded IoT and medical devices often have limited flash and memory resources reduce, reducing the size of complicated implementation can make deployment more practical. The relatively

stable .data and .bss value across the optimization levels also suggest that main reduction that came from the program code rather than from large changes in static memory requirements.

The comparison between the two platform provides a broader view implementation suitability for IoT systems [7],[8]; the desktop environment was useful for measuring execution time of the ML-KEM-512 operation whereas the ARM Cortex-M4 for environment was more relevant for evaluating an embedded memory requirements and deployment feasibility. Successful functional verification and matching shared secrets on both platforms indicate that the key-establishment operations were performed correctly. In addition, the generation of a ARM ELF and BIN files confirm that the implementation could be prepared for an embedded target.

Overall, the findings suggest that ML-KEM-512 can be implemented across desktop ARM Cortex-M4 environments, but its deployment and resource-constrained IoT devices required careful selection of compiler settings. The desktop results emphasize the need to consider execution time, while the ARM results demonstrate the importance of minimizing code size. These findings support practical feasibility of ML-KEM-512 for IoT key establishment, although the most suitable configuration will depend on the memory and processing limitations of device.

Conclusion

1. Problem addressed and motivation

This work addresses the need for secure practical establishment in resource constrained Internet of medical things devices. Since medical devices handle sensitive information [3] and may have limited processing and memory resources, the study explored the suitability of ML-KEM-512 as a post quantum cryptographic solution.

2. Method used

ML-KEM-512 was implemented using PQClean and tested in two environments : Ubuntu x86_64 and ARM cortex-M4 based platform. The implementation was functionally verified, and different compiler optimization levels were evaluated. The experiment concerned execution time, executable size, code-section size, memory utilization, and embedded image size.

3. Key findings

The result showed that compiler optimization had noticeable effect on the performance and size of the implementation. And the on the desktop platform, -O1 produced the lowest encapsulation time and the lowest average across the three operations, while -O2 produced the lowest key-generation and decapsulation times. The -Os produce smallest executable and .text section. On the ARM Cortex-M4 platform, -Os generated the smallest .text section and the lowest total image size. Successful functional verification and matching shared secrets on both platforms confirm the key that key establishment operation worked correctly. These findings indicate that ML-KEM-512 can be implemented on both desktop and embedded platform, although the choice of optimization level depends on the device's resources constraint

4. Limitations and future work

The study focused mainly on functional verification, execution time on the desktop platform, and memory related measurements on the ARM Cortex-M4 platform. Execution time was not measured directly on ARM device, and experiments did not include energy consumption, end to end IOMT communication or integration with a complete medical device security system. Future work should therefore evaluate ML-KEM-512 directly on physical IoMT hardware, major energy use is communication latency and investigate its integration with symmetric encryption scheme such as AES 256-GCM [13] or ASCON-128a [12]. These extensions would provide a more complete assessment of its practical use in real world IoMT deployments.

References

- [1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM journal on computing*, Vol. 26, no. 5, pp. 1484-1509, 1997.
- [2] National Institute of Standards and Technology, "Module-lattice-based Key-Encapsulation Mechanism Standard", NIST, Gaithersburg, MD, USA, FIPS PUB 203, Aug. 2024. doi: 10.6028/NIST.FIPS.203.
- [3] R. U. Rasool, H.F. Ahmad, W. Rafique, A. Qayyum, and J. Qadir, "Security and privacy of Internet of medical things: A contemporary review in the age of surveillance, botnets and adversarial ML," *Journal of Network and Computer Applications*, vol. 201, art. no. 103332, 2022. doi: 10.1016/j.jnca.2022.103332
- [4] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J.M. Schanck, P. Schwabe, G. Seiler, and D. Stehle, "CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM," in *Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, London, U.K., 2018, pp. 353-367. doi:10.1109/EuroSP.2018.00032.
- [5] T. Liu, G. Ramachandran, and R. Jurdak, "Post-Quantum Cryptography for Internet of Things: A survey on performance and optimization," *arXIV: 2401.17538*, 2024.
- [6] T.M Fernandez-Carames, "From pre-quantum to post-quantum IoT security: A survey on quantum-resistant cryptosystems for Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6457-6480, Jul. 2020. doi:10.1109/JIOT.2019.2958788
- [7] J.-A. Septien-Hernandez, M. Arellano-Vazquez, M. A. Contreras-Cruz, and J.-P. Ramirez-Paredes, "A comparative study of post-quantum cryptosystems for Internet-of-Things applications," *Sensors*, volume 22, number 2, article no. 489, 2022. doi: 10.3390/s22020489.
- [8] K. Basu, D. Soni, M. Nabeel, and R. Karri, "NIST post-quantum cryptography—A hardware evaluation study", *IACR Cryptology ePrint Archive*, document number 2019/047, 2019. [Online]. Accessed at: <https://eprint.iacr.org/2019/047>
- [9] M. J. Kannwischer, P. Schwabe, D. Stebila, and T. Wiggers, "Improving software quality in cryptography standardization projects," in *Proceedings of the 2022 IEEE European Symposium*

on Security and Privacy Workshops (EuroS&PW), 2022. [Online] Available at <https://eprint.iacr.org/2022/337>

[10] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "pqm4: Testing and benchmarking NIST PQC on ARM Cortex-M4", IACR Cryptology ePrint Archive, Paper 2019/844, 2019. [Online]. Accessed: <https://eprint.iacr.org/2019/844>

[11] L. Botros, M. J. Kannwischer, and P. Schwabe, "Memory-efficient high-speed implementation of Kyber on Cortex-M4", in Progress in Cryptology—AFRICACRYPT 2019, Lecture Notes in Computer Science, vol. 11627, Cham, Switzerland: Springer, 2019, pp. 209–228. doi: 10.1007/978-3-030-23696-0_11.

[12] M. Sönmez Turan, K. A. McKay, D. Chang, J. Kang, and J. Kelsey, 'Ascon-based lightweight cryptography standards for constrained devices: Authenticated encryption, hash, and extendable output functions', NIST, Gaithersburg, MD, USA, NIST SP 800-232, 2025. doi: 10.6028/NIST.SP.800-232.

[13] M. J. Dworkin, "Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC", NIST, Gaithersburg, MD, USA, in NIST SP 800-38D, 2007. doi: 10.6028/NIST.SP.800-38D.