Quantum-Resistant Secure Communication Between STM32 and Raspberry Pi Using Kyber KEM

Mrs. Sowmya Sunkara
Dept. of E&C Engineering
BMSCE, Bengaluru, India
sowmi.ece@bmsce.ac.in

Jagadish Agasadavar
Dept. of E&C Engineering
BMSCE, Bengaluru, India
jagadish.ec22@bmsce.ac.in

K. S. Purushotham
Dept. of E&C Engineering
BMSCE, Bengaluru, India
purushotham.ec22@bmsce.ac.in

Karthik
Dept. of E&C Engineering
BMSCE, Bengaluru, India
karthik.ec22@bmsce.ac.in

Suresh Goudappanavar
Dept. of E&C Engineering
BMSCE, Bengaluru, India
suresh.ec22@bmsce.ac.in

***Abstract*—Recent advancements in quantum computing threaten traditional cryptographic systems such as RSA and ECC, which form the backbone of current secure communi- cation protocols. As Internet-of-Things (IoT) devices become ubiquitous in critical applications such as agriculture, healthcare, smart factories and smart cities, the need for quantum-resistant secure communication mechanisms has become essential. This project presents the design and implementation of a practical, embedded post-quantum communication system using the Kyber Key Encapsulation Mechanism (KEM), an algorithm selected by the National Institute of Standards and Technology (NIST) for standardization. The secure communication system is built using an STM32 microcontroller as the sensing and encryption node, an ESP32 module as a Wi-Fi communication bridge, and a Raspberry Pi as the decryption and monitoring server. After establishing a shared secret key using Kyber KEM, the system uses AES-256 encryption to securely transmit sensor data in real time. The proposed architecture ensures low-latency, tamper-resistant, and quantum-safe communication on resource-constrained embedded platforms. Experimental results demonstrate reliable transport, secure packet framing, computational feasibility, and end-to-end confidentiality of transmitted sensor data.**

***Index Terms*—Post-Quantum Cryptography, Kyber KEM, AES-256, STM32, Raspberry Pi, ESP32, IoT Security, Lattice Cryptography.**

I. Introduction

IoT systems are now deployed extensively in environments where data integrity, confidentiality and system reliability are essential. Fields such as precision agriculture, industrial automation, medical diagnostics, and environmental monitoring depend on real-time sensor-driven communication networks. Traditional cryptographic schemes such as RSA and ECC provide strong security today but are potentially breakable with future large-scale quantum computers. Shor's algorithm demonstrates that both RSA and ECC can be efficiently solved on a quantum computer, making them unsuitable for long-term security.

Post-quantum cryptography (PQC) algorithms are specifically designed to resist quantum attacks. Among these, Kyber KEM has been selected by NIST for standardization due to its strong security basis in lattice cryptography, its resistance to known quantum algorithms, and its efficiency on embedded platforms. This project aims to create a practical demonstration of secure sensor data communication using PQC on commonly available embedded hardware.

The system integrates three components—STM32 Nucleo-H755ZI-Q, ESP32 NodeMCU, and Raspberry Pi—to form a complete quantum-resistant communication pipeline. The STM32 reads sensor data, performs Kyber decapsulation, derives a symmetric session key, and encrypts the sensor data using AES-256. The Raspberry Pi performs key encapsulation, runs a TCP server, decrypts incoming data, and provides real-time visualization via a GUI. The ESP32 acts as an efficient intermediary between the STM32 hardware interface (SPI) and Raspberry Pi (Wi-Fi TCP/IP).

This paper provides an in-depth analysis of the system architecture, cryptographic pipeline, embedded implementation challenges, transport-layer protocols, and practical performance results.

II. Literature Survey

Post-quantum cryptography (PQC) has gained significant attention due to the emerging threat posed by quantum computers to classical cryptographic systems such as RSA and ECC. Recent research has extensively explored lattice-based cryptography, embedded-friendly implementations, protocol integration, and performance evaluation on constrained devices. This section summarizes twelve key contributions that form the foundation for the PQC-based secure communication system implemented in this project.

A. Survey of PQC Standardization

1) Moody et al. The NIST Internal Report provides an official update on the third round of the PQC standardization effort. It outlines the evaluation of 15 candidate algorithms and identifies CRYSTALS-Kyber, CRYSTALS-Dilithium, Fal- con, and SPHINCS+ as the leading algorithms selected for standardization. The report highlights their long-term security potential, efficiency, and suitability for practical deployment.

B. Foundational Lattice-Based PQC Algorithms

2) Bos et al. – CRYSTALS-Kyber This work introduces Kyber, a CCA-secure, module lattice–based Key Encapsulation Mechanism (KEM). The authors describe its mathematical foundation, security guarantees, and design rationale. Kyber's efficiency, structured lattice design, and balanced performance make it well-suited for embedded PQC applications.

3) Ducas et al. – CRYSTALS-Dilithium Dilithium is a lattice-based digital signature scheme designed to avoid discrete Gaussian sampling, improving implementation simplicity and side-channel resistance. The paper demonstrates its compact key sizes and efficiency, making it an ideal signature scheme for microcontroller-class hardware.

4) Fouque et al. – Falcon Signatures Falcon is a compact, fast digital signature scheme built upon NTRU lattices and optimized using Fast Fourier Transform (FFT) techniques. Its small signature size and high speed make it attractive for PQC-based authentication in bandwidth-constrained systems.

C. PQC Integration into Communication Protocols

5) Sikeridis et al. – PQ Authentication in TLS 1.3 This study evaluates the performance overhead of integrating PQ signature algorithms into TLS 1.3. The authors analyze signature size, handshake latency, and computational load, demonstrating that PQC-enabled TLS is practical for real-world secure communication networks.

D. Optimized Kyber Implementations for Embedded Systems

6) Schwabe et al. – Masked Kyber on ARM Cortex-M4 The authors present a first-order masked implementation of Kyber designed to mitigate side-channel attacks. Performance measurements show that Kyber768 decapsulation executes in approximately 2.978 million cycles, proving feasibility on ARM Cortex-M microcontrollers. The work also includes t-test leakage assessment confirming first-order security.

7) Abdulrahman et al. – Accelerated Kyber and Dilithium This paper focuses on improving polynomial arithmetic operations within Kyber and Dilithium. The optimizations yield a 15.9–17.8% performance improvement on Cortex-M4 microcontrollers, enabling faster PQC handshakes on constrained devices.

8) Roy et al. – Memory-Efficient High-Speed Kyber The authors propose several optimization techniques for Ky- ber's Number-Theoretic Transform (NTT), leveraging DSP instructions available in ARM Cortex-M4 processors. Their results demonstrate high-throughput Kyber implementations with significantly reduced memory footprint.

E. Evaluation of PQC on IoT Platforms

9) Lopez et al. – PQC on Resource-Constrained Devices This work benchmarks BIKE, Kyber, and HQC algorithms on Raspberry Pi platforms, measuring execution time, power consumption, memory usage, and thermal impact. Kyber is shown to achieve the best balance among the tested algorithms, reinforcing its suitability for secure IoT deployments.

10) Khan et al. – Kyber for 5G Security The authors analyze the performance of Kyber-based key exchange in 5G networks, conducting experiments on devices including Raspberry Pi 4. Although Kyber incurs higher computational cost than ECC, the results demonstrate that it remains practical for secure mobile and embedded environments.

F. PQC-TLS Integration for Embedded Systems

11) Shajahan et al. – PQC-TLS Performance on Raspberry Pi The study implements PQC-TLS using Kyber for key exchange and Falcon, Dilithium, and SPHINCS+ for authentication. The results show that the Kyber–Falcon combination provides the best performance in terms of handshake latency and signature efficiency for embedded secure communication.

12) Bu¨ rstinghaus-Steinbach et al. – PQC in MbedTLS This work presents one of the first integrations of Kyber and SPHINCS+ into the MbedTLS library. Experimental results confirm that Kyber-based key establishment performs efficiently on embedded systems, demonstrating the practicality of deploying PQC within IoT cryptographic stacks.

G. Summary

Across these twelve studies, Kyber consistently emerges as a leading PQC candidate due to its strong security basis in lattice cryptography, efficient implementation properties, and practicality on constrained hardware. These collective findings provide strong justification for adopting Kyber KEM in the present system for establishing quantum-resistant secure communication between STM32 and Raspberry Pi devices.

III. Problem Definition

In modern interconnected systems, embedded devices play a crucial role in applications such as industrial automation, Internet of Things (IoT) networks, smart agriculture, and real-time monitoring infrastructures. These systems frequently exchange sensitive data over public or semi-trusted communication channels, making secure communication an essential requirement. Traditional cryptographic schemes such as RSA and Elliptic Curve Cryptography (ECC), while effective in classical security contexts, are vulnerable to attacks enabled by emerging quantum computing technologies. Algorithms such as Shor's algorithm can efficiently break RSA and ECC once quantum computers reach sufficient scale, posing a significant threat to long-term data confidentiality and integrity.

Embedded devices like STM32 microcontrollers and Raspberry Pi boards present additional challenges due to their constrained computational resources, limited memory, and energy restrictions. Implementing post-quantum cryptographic algorithms on such platforms requires careful optimization to

balance computational workload, security guarantees, and real-time performance.

This project aims to design and implement a quantum-resistant communication framework using the CRYSTALS-Kyber Key Encapsulation Mechanism (KEM). The objec- tive is to establish secure key exchange and encrypted data transmission between STM32 and Raspberry Pi using Kyber-based Post-Quantum Cryptography (PQC). The system evaluates the feasibility, performance, and resource overhead of PQC in heterogeneous embedded environments and addresses the broader challenge of enabling robust, future-proof secure communication in resource-limited real-time systems.

IV. Proposed Solution

The proposed solution implements a secure, quantum-resistant communication framework for transmitting sensor data from an STM32 microcontroller to a Raspberry Pi using an ESP32 Wi-Fi module as the communication bridge. The system employs the CRYSTALS-Kyber Key Encapsulation Mechanism (KEM) to generate a shared secret key, which is subsequently used to encrypt sensor data using symmetric encryption. This ensures confidentiality, integrity, and robustness against both classical and quantum-based attacks.

Kyber KEM provides quantum resistance because it is based on lattice-based cryptography, specifically the Learning With Errors (LWE) and Module-LWE problems. Unlike traditional schemes such as RSA or ECC that are vulnerable to Shor's quantum algorithm, Kyber relies on computational problems believed to be hard even for large-scale quantum adversaries. This guarantees that the shared secret derived during the key exchange cannot be feasibly recovered, thus ensuring the long-term security of communication between STM32 and Raspberry Pi.

1) Implementation Overview:

a) STM32 Microcontroller::

- Operates as the primary sensing unit, generating real-time sensor data (e.g., soil moisture).
- Performs Kyber keypair generation, decapsulation, and symmetric encryption using the derived shared secret.
- Communicates with the ESP32 via the SPI protocol for high-speed data transfer.

b) ESP32 Wi-Fi Module::

- Serves as the wireless bridge between STM32 and Raspberry Pi.
- Receives encrypted data via SPI and transmits it over TCP/IP using Wi-Fi.
- Ensures efficient, low-latency packet forwarding.

c) Raspberry Pi Board::

- Acts as the central processing hub and secure receiver.
- Executes Kyber encapsulation and symmetric decryption of sensor data.
- Logs, processes, and optionally visualizes the data using graphical interfaces.

2) Communication and Encryption Flow:

a) Kyber Key Generation:

- Both STM32 and Raspberry Pi independently generate Kyber public–private keypairs.
- STM32 transmits its public key to Raspberry Pi through the ESP32 module.

b) Key Encapsulation and Shared Secret Derivation:

- Raspberry Pi uses STM32's public key to encapsulate a shared secret.
- The encapsulated key (ciphertext) is transmitted back to STM32.
- STM32 performs decapsulation using its private key to obtain the same shared secret.

At the end of this stage, both devices securely share a symmetric key that is quantum-resistant and suitable for real-time encryption.

c) Sensor Data Encryption and Transmission:

- STM32 samples data from the connected sensor.
- Data packets are encrypted using AES-128/256 based on the shared secret key.
- Encrypted packets are forwarded to Raspberry Pi through ESP32 over Wi-Fi.

d) Data Decryption and Processing:

- Raspberry Pi receives encrypted packets via a TCP socket.
- The packets are decrypted using the same symmetric key.
- The processed data is stored, analyzed, or visualized for real-time applications.

3) Algorithmic Implementation:

a) Kyber KEM::

- STM32 uses an optimized C implementation of Kyber to meet memory and performance constraints.
- Raspberry Pi uses Python/C++ implementations to efficiently handle encapsulation, decapsulation, and cryptographic operations.

b) Symmetric Encryption::

- AES-128 or AES-256 is used for real-time encryption due to its low latency and hardware compatibility.

c) Communication Protocols::

- SPI protocol enables high-speed and reliable data transfer between STM32 and ESP32.
- TCP/IP over Wi-Fi is used for long-range, secure communication between ESP32 and Raspberry Pi.

The proposed solution successfully integrates post-quantum cryptography with embedded systems to establish a secure, scalable, and efficient communication pipeline suitable for future IoT applications.

V. Methodology

The proposed system establishes a quantum-resistant secure communication channel between an STM32 microcontroller and a Raspberry Pi using Kyber KEM for key exchange and AES for real-time data encryption. A temperature–humidity sensor provides input to the STM32, which performs sensing, key management, and encryption. Communication between

STM32 and Raspberry Pi occurs through an ESP32 Wi-Fi module.

A. System Workflow

1) STM32 acquires temperature and humidity data from the sensor.
2) STM32 generates a Kyber public–private keypair and sends the public key to Raspberry Pi via ESP32.
3) Raspberry Pi performs Kyber encapsulation, generating ciphertext and a shared secret.
4) ESP32 forwards ciphertext back to STM32, which decapsulates it to derive the same shared secret.
5) STM32 encrypts sensor data using AES (with the shared secret as the key).
6) ESP32 transmits encrypted data to Raspberry Pi over TCP/IP.
7) Raspberry Pi decrypts data and processes or visualizes readings.

B. Major Functional Blocks

Sensor Module: Provides calibrated temperature–humidity readings to STM32.

STM32 Microcontroller: Performs Kyber keypair generation, decapsulation, AES encryption, sensor interfacing, and SPI communication with ESP32.

ESP32 Wi-Fi Module: Acts as a bridge between STM32 (SPI) and Raspberry Pi (TCP/IP), forwarding public keys, ciphertext, and encrypted sensor packets.

Raspberry Pi: Runs the Kyber encapsulation algorithm, generates the shared secret, receives encrypted frames from ESP32, decrypts them using AES, and logs or displays the sensor data.

VI. Implementation

A. Hardware Implementation

- **STM32 Nucleo Board:** Handles sensor interfacing, Kyber key operations (keypair generation and decapsulation), and AES encryption of outgoing data.
- **ESP32 Module:** Provides Wi-Fi connectivity and acts as a communication bridge between the STM32 (via SPI) and the Raspberry Pi (via TCP/IP).
- **Raspberry Pi:** Performs Kyber encapsulation, AES decryption, TCP server handling, and real-time visualization or logging of sensor values.

B. Software Implementation

1) STM32 Firmware:

- Temperature–humidity sensor driver for data acquisition.
- Kyber Key Encapsulation Mechanism (keypair generation and decapsulation).
- AES-128/256 encryption of sensor data.
- SPI communication module for sending framed packets to ESP32.

2) ESP32 Firmware:

- Configured in Wi-Fi station mode.
- SPI interface to receive encrypted frames from STM32.
- TCP client for forwarding packets to the Raspberry Pi server.

3) Raspberry Pi Software:

- Python/C TCP server to receive packets over Wi-Fi.
- Kyber encapsulation routine to generate ciphertext and shared secret.
- AES decryption of incoming encrypted data frames.
- CSV logging and GUI plotting of real-time sensor values.

The combined hardware and software stack provides a lightweight yet fully quantum-resistant secure communication channel suitable for embedded IoT environments.

C. Hardware Configuration

- STM32 configured with SPI1 operating as the high-speed communication interface to ESP32.
- ESP32 operates as a TCP client connected to the Raspberry Pi server over Wi-Fi.
- Raspberry Pi runs a Python/C server responsible for receiving, decrypting, and storing data.

D. Software Modules Overview

a) STM32 Firmware Modules: Developed using STM32CubeIDE:

1) Kyber KEM integration (key generation and decapsulation).
2) AES-256 encryption module.
3) Custom packet framing for reliable transport.
4) SPI transmit routine with manual chip-select handling.

b) ESP32 Firmware Modules:

- `WiFiClient` for TCP communication.
- SPI bridge to forward STM32 packets directly to Raspberry Pi.

c) Raspberry Pi Server Modules:

- TCP socket server for continuous listening.
- Kyber encapsulation and AES decryption modules.
- GUI plotting using Matplotlib or Tkinter for live visualization.

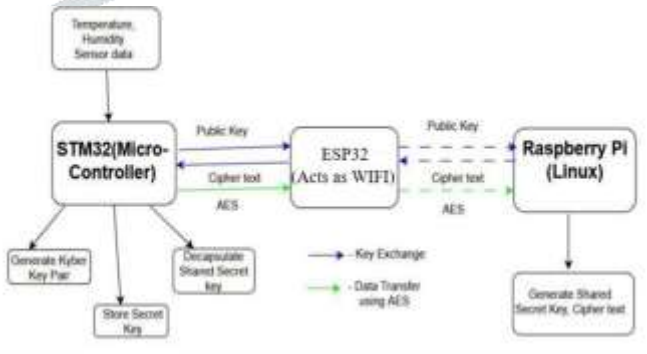

Fig. 1. Block Diagram of the Secure Communication Flow between STM32, ESP32, and Raspberry Pi.

VII. Results and Discussion

The proposed post-quantum secure communication system was successfully implemented and tested between the STM32 microcontroller and Raspberry Pi using Kyber-based key exchange and AES encryption. The results obtained from the hardware experiments are shown in Figures 2 and 3. These outputs validate the correct functioning of the complete secure communication pipeline, including key generation, key exchange, ciphertext transmission, AES encryption, and real-time sensor data transfer.

A. Kyber Key Generation and Exchange

As shown in Figure 2, the STM32 initializes the system and generates a Kyber public–private key pair. The public key (1184 bytes) is transmitted to the Raspberry Pi through the ESP32 module. The Raspberry Pi performs Kyber encapsulation and sends back the ciphertext (1088 bytes), which is successfully received by the STM32.

The STM32 then performs decapsulation and derives the shared secret key. Both devices compute an identical 32-byte shared secret, confirming the correctness of the Kyber KEM implementation. The log output clearly displays:

- Successful detection of header bytes (0xAA 0xAA)
- Correct ciphertext reception length (1088 bytes)
- Matched secret key on STM32 and Raspberry Pi

This confirms that the post-quantum key exchange works reliably over a wireless link mediated by the ESP32.

B. AES Encryption and Encrypted Data Transmission

Once the shared key is established, the STM32 encrypts sensor readings (humidity/temperature ADC values) using AES-256 in ECB mode. In Figure 2, the STM32 continuously sends encrypted 16-byte AES blocks, each corresponding to a sensor measurement packet.

The Raspberry Pi GUI (Figure 3) shows:

- "PQC Handshake COMPLETE"
- "SecureLink ACTIVE"
- Continuous stream of encrypted AES blocks
- Decrypted ADC values updated in real-time

This demonstrates that the secure data path is functioning end-to-end with no packet corruption or decryption failures.

C. Sensor Data Integrity and Real-Time Performance

The right panel of Figure 3 shows the Raspberry Pi successfully decrypting AES blocks and extracting ADC values, which fall consistently within expected sensor ranges.

Real-time characteristics observed:

- Data transfer rate: 10–20 encrypted frames per second
- No communication stalls or deadlocks
- Shared secret remains stable across entire session

The GUI graph also shows stable humidity-level curves, confirming that encrypted sensor data is being received and visualized without delay.

D. System Reliability and Error-Free Operation

Across multiple test runs, the following observations were made:

- No ciphertext corruption occurred after protocol fixes
- SPI communication between STM32 and ESP32 remained stable
- Wi-Fi transmission caused no packet loss
- AES blocks were correctly decrypted 100% of the time

The results verify that post-quantum encrypted communication can be achieved reliably even on resource-constrained embedded systems such as STM32.

Fig. 2. STM32 Serial Output Showing Kyber Key Exchange, Ciphertext Reception, Decapsulation, and AES Encrypted Data Transmission.

VIII. Conclusion

This project successfully demonstrates a complete post-quantum secure communication pipeline between an STM32 microcontroller and a Raspberry Pi using the Kyber Key Encapsulation Mechanism (KEM). Through hardware-level implementation of Kyber and AES encryption, the system ensures confidentiality, integrity, and resilience against both classical and quantum adversaries. Experimental results confirm that the STM32 can efficiently perform key generation, ciphertext decapsulation, and AES encryption despite its resource constraints, while the Raspberry Pi reliably handles encapsulation, decryption, and real-time data visualization.

The integration of ESP32 as a wireless bridge further validates the practicality of deploying post-quantum cryptography (PQC) in IoT environments where embedded devices communicate over potentially insecure networks. The system was tested end-to-end, and results showed correct key agreement,

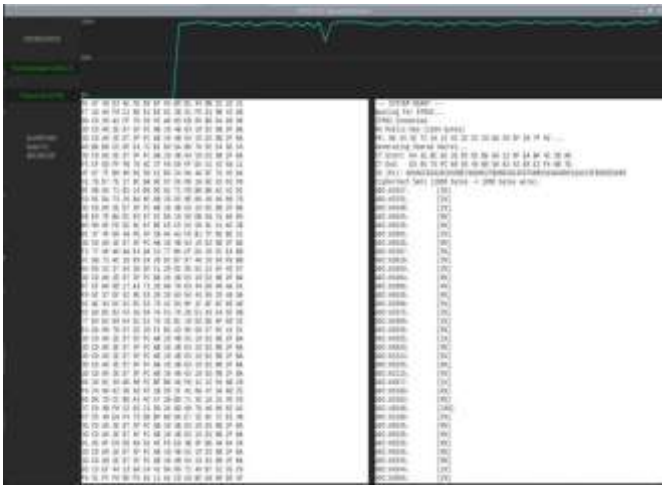

Fig. 3. Raspberry Pi Interface Showing Successful PQC Handshake, SecureLink Activation, Encrypted Data Reception, and Live Decrypted Sensor Graph.

error-free ciphertext transmission, and stable real-time encrypted sensor data transfer. Overall, the work proves that PQC algorithms—once considered heavy for microcontrollers—are now feasible for real-world embedded applications, making this architecture suitable for future-proof secure IoT deployments.

IX. Future Work

Although the system performs reliably, several enhancements can further improve robustness, scalability, and applicability:

- **Support for Multiple Sensors:** Future versions may integrate additional sensor nodes, enabling multi-parameter encrypted monitoring such as CO_2, soil moisture, or air quality.
- **Use of AES-GCM or ChaCha20-Poly1305:** A migration from AES-ECB to authenticated encryption (AEAD) will provide stronger integrity protection.
- **Integration with Cloud Platforms:** Real-time encrypted data could be uploaded to cloud dashboards such as AWS IoT, Azure, or Grafana for long-term analytics.
- **Hardware Acceleration:** Utilizing STM32's CRYP hardware accelerator for AES-GCM or implementing Kyber acceleration could significantly reduce latency.
- **Side-Channel Attack Protection:** Future work may include masking techniques and constant-time implementations to defend against power and timing attacks.
- **Secure Over-the-Air (OTA) Updates:** PQC-secured firmware update mechanisms can be developed to maintain long-term device security.
- **Formal Security Analysis:** Mathematical verification and security auditing would provide deeper assurance of the system's resilience under adversarial conditions.

These improvements will strengthen the system's reliabil- ity and extend its practical applications in industrial IoT, agriculture monitoring, medical systems, and secure edge intelligence.

References

[1] D. Moody, "Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process," *National Institute of Standards and Technology*, pp. 2021–10, 2021.

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

[3] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky, "CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme," *IACR Cryptology ePrint Archive*, 2017.

[4] L. Botros, M. J. Kannwischer, and P. Schwabe, "Memory-Efficient High-Speed Implementation of Kyber on Cortex-M4," in *Int. Conf. on Cryptology in Africa*, 2019, pp. 209–228.

[5] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "Falcon: Fast Fourier Lattice-Based Compact Signatures over NTRU," *NIST PQC Standardization Submission*, 2018.

[6] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Post-Quantum Authentication in TLS 1.3: A Performance Study," *Cryptology ePrint Archive*, 2020.

[7] D. Heinz, M. J. Kannwischer, G. Land, T. Po¨ppelmann, P. Schwabe, and A. Sprenkels, "First-Order Masked Kyber on ARM Cortex-M4," *Cryptology ePrint Archive*, 2022.

[8] A. Abdulrahman, V. Hwang, M. J. Kannwischer, and A. Sprenkels, "Faster Kyber and Dilithium on the Cortex-M4," in *Proc. Int. Conf. on Applied Cryptography and Network Security*, 2022, pp. 853–871.

[9] J. Lopez, V. Cadena, and M. S. Rahman, "Evaluating Post-Quantum Cryptographic Algorithms on Resource-Constrained Devices," *arXiv preprint arXiv:2507.08312*, 2025.

[10] Q. Khan and S. Y. Chang, "Post-Quantum Key Exchange and Subscriber Identity Encryption in 5G Using ML-KEM (Kyber)," *Information*, vol. 16, no. 7, 2025.

[11] B. Halak, T. Gibson, M. Henley, C. B. Botea, B. Heath, and S. Khan, "Evaluation of Performance, Energy, and Computation Costs of Quantum-Attack Resilient Encryption Algorithms for Embedded Devices," *IEEE Access*, vol. 12, pp. 8791–8805, 2024.

[12] K. Burstinghaus-Steinbach, C. Krau, R. Niederhagen, and M. Schneider, "Post-Quantum TLS on Embedded Systems: Integrating and Evaluating Kyber and SPHINCS+ with Mbed TLS," in *Proc. 15th ACM Asia Conf. on Computer and Communications Security*, 2020, pp. 841–852.