Quantum-Resistant and Secure MQTT Communication

Lukas Malina
malina@vut.cz
Brno University of Technology
Brno, Czech Republic

Patrik Dobias
xdobia13@vut.cz
Brno University of Technology
Brno, Czech Republic

Petr Dzurenda
dzurenda@vut.cz
Brno University of Technology
Brno, Czech Republic

Gautam Srivastava
srivastavag@brandonu.ca
Brandon University
Brandon, Canada

ABSTRACT

In this paper, we deal with the deployment of Post-Quantum Cryptography (PQC) in Internet of Things (IoT). Concretely, we focus on the MQTT (Message Queuing Telemetry Transport) protocol that is widely used in IoT services. The paper presents our novel quantum-resistant security proposal for the MQTT protocol that supports secure broadcast. Our solution omits using TLS with the handshake causing delay and is suitable for sending irregular short messages. Finally, we show how our solution can practically affect concrete use cases by the performance results of the proposed solution.

CCS CONCEPTS

• **Security and privacy** → **Public key (asymmetric) techniques**; **Security protocols**.

KEYWORDS

Applied Cryptography, Assessment, Performance, Post-Quantum Cryptography, IoT, MQTT

ACM Reference Format:
Lukas Malina, Patrik Dobias, Petr Dzurenda, and Gautam Srivastava. 2024. Quantum-Resistant and Secure MQTT Communication. In *The 19th International Conference on Availability, Reliability and Security (ARES 2024), July 30–August 02, 2024, Vienna, Austria.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3664476.3670463

1 INTRODUCTION

IoT-based services in intelligent infrastructure often demand strong security and privacy requirements. In previous years, the security solutions for IoT have mainly focused on efficiency and investigating if expensive cryptographic solutions such as asymmetric encryption and digital signature schemes are suitable for IoT networks equipped with constrained devices. Nowadays, incoming quantum computing and related quantum computing attacks will change the basic requirements and start recommending the cryptographic suites deploying Post-Quantum Cryptography (PQC) algorithms that should be resilient to those attacks. In 2022, NIST selected new PQC algorithms for asymmetric encryption and digital signatures for standardization[1]. This boosts up post-quantum transition in all ICT fields including IoT. In this work, we investigate the deployment of PQC in IoT, concretely, on the MQTT protocol. We focus on the direct integration of suitable post-quantum schemes into MQTT. Our solution can be suitable for various use cases with constrained devices.

1.1 Related Work

In the last years, the security in MQTT/IoT has been studied by numerous works such as [9, 14, 15, 21]. Hamad *et al.* [14] provided a comprehensive comparison of such works in 2023. Nevertheless, those proposals are often based on pre-quantum cryptography schemes. Further, the literature focused on the investigation of post-quantum cryptography schemes on resource-constrained devices used in IoT. For example, Roy and Kalita [19] dealt with the applicability of PQC on constrained devices, and compared schemes such as BLISS, NTRU, Lamport, Rainbow and McEliece. Chung *et al.* [8] evaluated PQC algorithms (BIKE, HQC, FrodoKEM, Kyber, NTRU, NTRU Prime, SABER, IKE.) on Raspberry Pi 3 and 4. They showed that PQC schemes cause affordable latency in TLS. The performance of PQC and ECDH/ ECDSA is similar but PQC schemes cause higher overheads in packets. There are more works that evaluate PQC schemes on popular single-board devices (e.g. Raspberry Pi), such as [3, 6, 20]. For instance, in 2023, Bozhko *et al.* [6] studied quantum-resistant TLS running on IoT devices, and compared handshake delays with various combinations of PQC signatures and Kyber versions via different communication settings such as BLE, Wifi, etc. between a laptop and Raspberry Pi 4. Nevertheless, those works usually focus on benchmarking the PQC but without the deployment in the MQTT protocol.

Only a few works also consider the feasibility of PQC in MQTT. In 2020, Agus *et al.* [1] compared the RSA encryption with PQC NTRU encryption on Raspberry Pi 3 using the MQTT-IoT protocol. NTRU provided more efficient times than RSA. These results indicate that PQC can be promising. Schoffel *et al.* [22] investigated the TLS handshake latency and energy with several NIST PQC finalists and alternate candidates on low-power IoT end nodes using MQTT protocol for communication. Rampazzo and Henriques

ARES 2024, July 30–August 02, 2024, Vienna, Austria

ACM ISBN 979-8-4007-1718-5/24/07
https://doi.org/10.1145/3664476.3670463

[1] https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022

[17] have recently evaluated post-quantum and pre-quantum cryptography and their hybrid combinations in MQTT. All the above works directly use TLS with various PQC settings. Nevertheless, we introduce a PQC-based solution without TLS that should be suitable for use cases that seek less latency in the MQTT protocol on very constrained devices.

1.2 Contributions and Paper Organization

The contributions (and organization) of this work are as follows:

- We firstly analyze the state of the art in PQC, the transition to PQC, and the support of PQC in practice and its suitability for IoT/MQTT (Section 2).
- We present our solution that directly employs PQC schemes in MQTT architecture and is suitable even for scenarios with constrained devices (Sections 3, 4).

2 BACKGROUND

This section introduces the PQC schemes that have been selected by NIST for the standardization process and discusses some recent US/EU recommendations. Further, we analyze security libraries that support PQC and can be used for practical implementations in IoT.

2.1 NIST PQC Selected Algorithms

Currently, NIST PQC competition counts 3 digital signature schemes and 1 KEM scheme as candidates for standardization. Nevertheless, there is the fourth round with three more KEM schemes that will serve as alternatives. All schemes are briefly introduced below:

- **Dilithium** [10] is a quantum-resistant digital signature scheme based on Module Learning with Errors (M-LWE) and Module Short Integer Solution (M-SIS) problems. Dilithium provides high flexibility which allows for a variety of security and performance trade-offs. With its strong theoretical foundation and cryptoanalytic history, Dilithium is a suitable choice for a wide range of security applications. NIST has selected Dilithium as the primary signature algorithm for standardization due to its high efficiency, simple implementation, and strong security compared to other PQ signatures.
- **SPHINCS+** [4] is a stateless hash-based signature scheme. The scheme is based on the Merkle tree construction and provides high security against classical and quantum attacks. Unlike stateful hash-based signature schemes, which require keeping track of the state between signatures, SPHINCS+ does not need to maintain states but is slower and produces larger signatures than stateful schemes. SPHINCS+ has a relatively high time and memory complexity compared to other signatures.
- **Falcon** [13] is a lattice-based PQ digital signature scheme. Falcon is often considered as an efficient choice, especially for resource-constrained environments such as embedded systems and IoT devices. It provides strong security assurance and narrow bandwidth, which may be essential in certain applications typical for IoT.
- **Kyber** [5] is a lattice-based PQ encryption scheme designed for key encapsulation mechanisms (KEM). It is based on the M-LWE problem and offers high flexibility concerning security. It allows changes in parameters without affecting the underlying operations. Kyber has a solid performance in software, hardware, and hybrid settings and therefore was chosen as the first KEM candidate for standardization.
- **Classic McEliece** [7] is a KEM scheme based on error-correcting codes. As it was published in the 1970s it is considered one of the oldest KEM schemes. Although McEliece is relatively simple and offers a fast key encapsulation process, the disadvantage is larger public key sizes, making it less feasible for certain end-nodes with constrained memory. The public key size is over one million bytes and it's the largest compared to other candidates, as shown in Table 2.
- **BIKE** [2] is another code-based public key encryption scheme. BIKE has the most competitive performance among the non-lattice-based KEM schemes. The quasi-cyclic structure of BIKE enables public key and ciphertext sizes comparable to the structured lattice KEM schemes. BIKE could be a promising alternative to Kyber in IoT but requires more memory space for keys and ciphertexts.
- **HQC** [16] is also a code-based public key encryption scheme. HQC offers strong security assurances and a mature decryption failure rate analysis. Despite its quasi-cyclic structure allowing for reasonable sizes of public keys and ciphertexts, HQC's public keys and ciphertexts are larger than in other structured code-based and lattice-based KEM schemes.
- **SIKE** (https://sike.org/) was originally chosen by NIST for 4. round but the SIKE team acknowledged that SIKE and SIDH are insecure and should not be used[2].

2.2 US and EU Recommendations

National Security Agency (NSA) in US announced Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) to address the deployment of cryptanalytically relevant quantum computers. CNSA 2.0 lists quantum-resistant algorithms for key establishment, digital signatures, and encryption. Table 1 compares algorithms CNSA 1.0 and CNSA 2.0.

According to NSA, networking equipment with services should support and prefer CNSA 2.0 by 2026, and must use it by 2030. Nevertheless, other equipment, e.g., constrained devices including smart meters, IoT nodes with MQTT etc., should support and prefer CNSA 2.0 by 2030, and must use it by 2033. Custom applications and legacy equipment have to be updated or replaced by 2033.

European cybersecurity agencies such as French ANSSI [18] and German BSI [12] recommend a hybrid approach that combines PQC algorithms with classic cryptography for digital signatures and key establishment in the short and medium term. There is one exception which is the direct deployment of hash-based signatures (SPHINCS+, XMSS, LMS) for specific use cases such software updates.

2.3 Suitable Security Libraries with PQC for IoT/MQTT

We assume that typical IoT devices are small single-boards computing units constrained in memory and CPU and serve as the

[2]https://csrc.nist.gov/csrc/media/Projects/post-quantum-cryptography/documents/round-4/submissions/sike-team-note-insecure.pdf

Table 1: Listed Algorithms CNSA 1.0 and CNSA 2.0.

Purpose	Algorithms CNSA 1.0	Algorithms CNSA 2.0
Symmetric encryption	Advanced Encryption Standard (AES) with 256-bit keys	Advanced Encryption Standard (AES) with 256-bit keys
Key establishment	RSA with min. 3072-bit modulus, ECDH P-384, DH 3072-bit modulus	CRYSTALS-Kyber with sec. level V
General digital signatures	RSA with min. 3072-bit modulus, ECDSA P-384	CRYSTALS-Dilithium with sec. level V
Digital signatures of FW/SW	Not specified, could be RSA with 3072-bit modulus, ECDSA P-384	Leighton-Micali Signature (LMS) with SHA256, Xtended Merkle Signature Scheme (XMSS)
Data hashing	SHA-384	SHA-384 or SHA512

Table 2: Memory Requirements Comparison of NIST PQC digital signature and KEM schemes for Security level 5 (equivalent to AES-256).

Scheme	NIST Level	Private Key $[B]$	Public Key $[B]$	Signature/ Ciphertext $[B]$
Dilithium	5	4864	2592	4595
SPHINCS+	5	128	64	49856/29792
Falcon	5	2305	1793	**1**330
Kyber	5	3168	1568	1568
McEliece	5	13892	1044992	240
BIKE	5	16494	5122	5144
HQC	5	7285	7245	14469

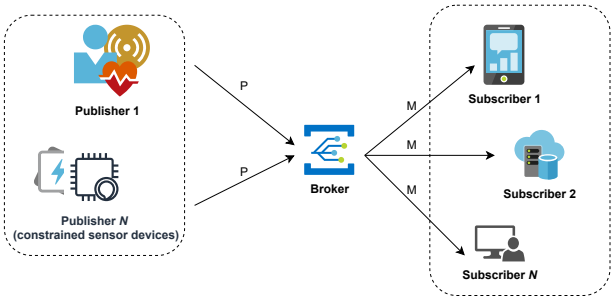

Figure 1: Scenario A: MQTT with thin publisher.

simple sensing units at the end nodes. Suitable libraries for such devices should be compact. Such libraries are usually programmed on C/C++ and eventually JAVA programming languages. There are various MQTT implementations that can be downloaded at the website https://mqtt.org/software/. For instance, the Eclipse Mosquitto is an open-source broker written in C/C++, which implements the MQTT broker (MQTT versions 3.1, 3.1.1 and 5.0.) and is a light and easy-to-use.

Table 3 lists well-known libraries providing PQC algorithms and can be deployed in IoT. The Open Quantum Safe project[3] presented a core experimental library called liboqs that serves as the base for many other libraries. The provided overview of libraries lists mainly those ones that can be adapted for IoT devices, i.e., multiplatform support, compact sizes, etc. Besides those libraries in Table 3, there are more libraries and implementations provided PQC schemes, e.g., libssh, boringssl, etc.

3 PROPOSAL OF QUANTUM RESISTANT MQTT PROTOCOL

In this section, we propose a novel quantum-resistant security framework for IoT/MQTT services with constrained devices. In the following text, we describe the high-level overview and architecture and all phases of the solution in detail.

3.1 High-level Overview and Architecture

The general communication model consists of three parties: Publisher (P), Broker (B), and Subscriber (S). We assume that a broker is a semi-trusted party that arranges cryptographic settings. Publishers sense data and create messages that are sent to the broker. The messages sent to brokers are then delivered to end users (subscribers) based on assigned topics. The broker works as the gateway and sorts messages inserted in topic fields.

[3]https://github.com/open-quantum-safe

The system architecture can be divided into three scenarios based on used devices. The first scenario depicted in Figure 1 assumes constrained devices on the publisher's side.

The second scenario depicted in Figure 2 assumes constrained devices at the subscriber's side.

The third scenario assumes constrained devices on both sides (constrained publishers and subscribers). The typical use case is device-to-device communication in sensor networks with constrained devices.

Our proposed protocol is designed into two security-level settings. The first security level (SL1) provides data integrity, data authenticity, and non-repudiation (only a valid publisher can create original data with a concrete signature). In this security setting, we deploy the quantum-resistant digital signature that offers short signatures, i.e., Falcon [13].

The second security level (SL2) adds one-time quantum-resistant asymmetric encryption. We choose Kyber [5] as the recommended algorithm by CNSA 2.0[4] and also it provides a good trade-off in key and ciphertext lengths in comparison with other 4. round NIST algorithms (i.e., McEliece, BIKE, HQC).

The proposed solution contains these phases: Setup, Join, and Communication Phase: Security level 1 (or Communication Phase: Security level 2) and Revocation.

3.2 Setup Phase

During the setup phase, the broker as the semi-trusted party established their own cryptography parameters, i.e., broker's Kyber public and private keys ($PrivKEM_B$, $PubKEM_B$) for quantum-resistant asymmetric encryption.

The public key is openly released for all parties who will be joining MQTT services in the Join phase.

[4]https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF

Table 3: Software Libraries of PQC (not exhaustive) and its suitability for small platforms used in IoT/MQTT.

Name	PQC supported	Licenses/languages	Platforms - IoT Suitability	Website
liboqs	Kyber, Classic McEliece, BIKE, HQC, FrodoKEM, NTRU-Prime, Dilithium, Falcon, SPHINCS+	Open source C library; bindings/wrappers in Python, C#, C++, JAVA, Go, Rust	Linux, macOS, and Windows, supports x86_64 and ARM architectures.	https://openquantumsafe.org/liboqs/
libpqcrypto	19 PQC schemes (BIG QUAKE, Classic McEliece, DILITHIUM, KYBER, DAGS, FrodoKEM, Gui, KINDI, LUOV, MQDSS, NewHope, NTRU-HRSS-KEM, NTRU Prime, Picnic, qTESLA, Rainbow, Ramstake, SABER, SPHINCS+)	Experimental C library for Debian/Ubuntu systems, Python/C API	Can be compiled in 32-bit mode and run on various singleboards.	https://libpqcrypto.org/
CIRCL[11]	Dilithium, Kyber, FrodoKEM, SIDH/SIKE	Cloudflare Interoperable, Reusable Cryptographic Library written in Go	Focused on x86-64 but ARM platform is supported.	https://github.com/cloudflare/circl
ISARA Radiate™ Quantum-safe Library Version 3.1	Classic McEliece, DILITHIUM, KYBER, HSS, XMSS, SPHINCS+	Licensed C-based library via gcc or clang compilers	Multiplatform (Windows, Linux, macOS, iOS, Android).	https://www.isara.com/toolkit/3/doc/library/index.html
OpenSSH	NTRU (hybrid mode with EC method), XMSS (only for experimental use)	BSD-style licensed SSH library with the support of the LibreSSL library for some cryptography methods	Used in various OS and network devices.	https://www.openssh.com/
OQS-OpenSSH	alone or in hybrid mode with ECDH: BIKE, ClassicMcEliece, FrodoKEM, HQC, Kyber; Dilithium, Falcon, SPHINCS-Haraka/SHA256/SHAKE256	C-based library. A fork based on OpenSSH version 8.9 with liboqs.	Tested on Ubuntu 20.04.1, less suitable for IoT.	https://github.com/open-quantum-safe/openssh
OQS-OpenSSL	BIKE, Kyber, HQC, FrodoKEM, Dilithium, Falcon, SPHINCS-Haraka/SHA256/SHAKE256	A fork of multi-platform robust OpenSSL library (1.1.1) written in C; PQC is provided via oqsprovider, depended on liboqs	General UNIX and Windows platforms. Less suitable for IoT.	https://github.com/open-quantum-safe/oqs-provider
Wolf SSL	Dilithium, FALCON, SPHINCS+, Kyber KEM	Lightweight SSL/TLS library written in ANSI C	Targeted at IoT, embedded, and RTOS environments.	https://www.wolfssl.com/
Bouncy Castle	Kyber, Dilithium, Falcon, SPHINCS+, BIKE, HQC, NTRU, NTRU Prime, Picnic, FrodoKEM, GeMSS, LMS, Newhope, Rainbow, Saber, XMSS, SIKE	MIT licensed Java library with cryptographic algorithms and security protocols	Robust JAVA (C#) library for general platforms. Less suitable for IoT.	https://www.bouncycastle.org/releasenotes.html

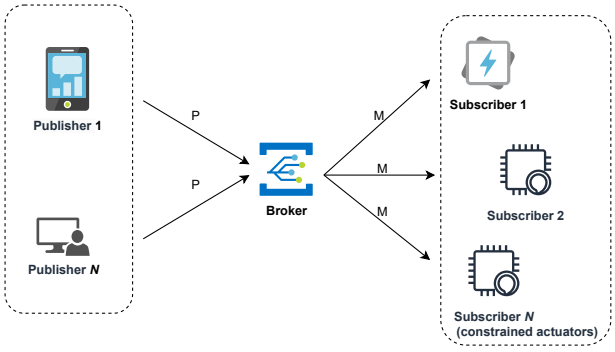

Figure 2: Scenario B: MQTT with thin subscriber.

3.3 Join Phase

Firstly, each i publisher who joins the system downloads the broker's public key ($PubKEM_B$), stores it, and generates the Falcon's public and private keys ($PrivDS_{P_i}$, $PubDS_{P_i}$). The private key is securely stored on the publisher side, e.g., in a secure element. The public key is directly sent to the broker.

Secondly, each i subscriber who registers on the broker, receives from B the set of public Falcon keys from all publishers based on chosen topics that the subscriber wants to receive. Then, the thick subscribers generate the Kyber public and private keys ($PrivKEM_{S_i}$, $PubKEM_{S_i}$) for decrypting messages in the SSL2 mode. These subscribers have to deliver their Kyber public keys to the broker during the communication phase. If the subscribers are thin (constrained), we assume that these subscribers will use only the setting security level 1.

3.4 Communication Phase: Security level 1

The communication phase in the SL1 protects published messages against their modification and ensures message authenticity with non-repudiation. To be noted that SL1 does not provide data confidentiality and is suitable for non-vital data that can be published but must be not modified or forged. Also, this option is suitable for thin subscribers who do not need to decrypt the messages.

The communication phase of SL1 (depicted in figure 3) contains the following steps:

(1) Subscribers, who want to subscribe to the messages of chosen topics T, send to the broker a subscriber request message with the requested security level 1 and a topic name. B collects all requests from subscribers to decide on the next phases where the messages should be distributed.
(2) The publisher senses data and prepares the publisher message PM for the MQTT protocol. In SL1, PM contains topic T, actual timestamp t, plain data, and the Falcon's signature of PM message sig : PQCSig($PrivDS_{P_i}$, PM) created by its private key and those data are sent to the broker.
(3) The broker receives the published message with the signature and checks the content syntax, timestamp, and topic name. If the timestamp is fresh and the content is OK, B forwards this message to all subscribers who subscribe to this topic. In special cases (within scenario B) with highly constrained subscribers, B could verify the signature of the published message from P and just send the message without signature to these subscribers.
(4) The subscribers receive the messages from the broker and verify the signature of the published message by the publisher's public key (the Falcon signature).

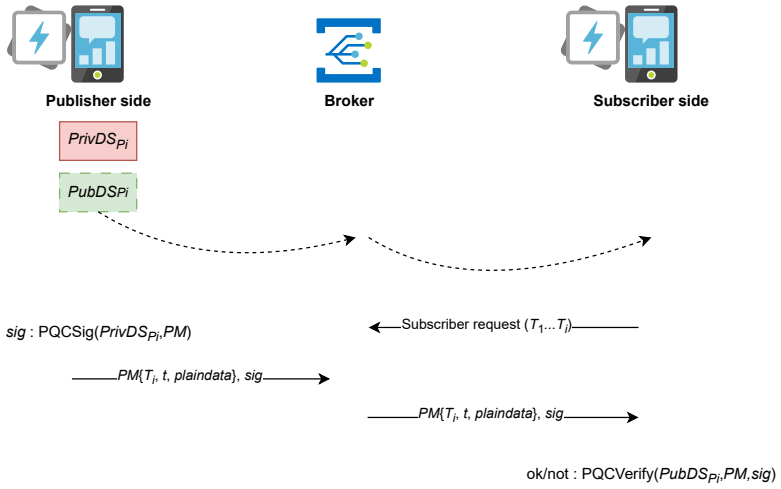

Figure 3: Communication Phase: SL1

3.5 Communication Phase: Security level 2

At this level, the messages sent from publishers to subscribers are protected against modification and also eavesdropping. This level is suitable for protecting vital user data on the whole communication chain but both sides need to be ready to perform both PQC signature and encryption algorithms. Therefore, we assume that both sides will be not-constrained nodes.

The communication phase of SL2 (depicted in Figure 4) contains the following steps:

(1) Subscribers, who want to confidentially subscribe to the messages of chosen topics T, send to the broker subscriber request messages with the requested security level 2 (SL2), their own public keys, and requested topic names. B collects all those requests and stores public keys from subscribers.
(2) The publisher senses data and prepares the message for the MQTT protocol stack. In SL2, P firstly signs the plain data, topic, and actual timestamp t by its private key. Secondly, plain data are encrypted and the ciphertext, signature, topic name, and timestamp are sent to the broker.
(3) The broker receives the published message with the signature and ciphertext and firstly checks if the timestamp is OK. If the timestamp is fresh, B decrypts the ciphertext to plain data. Then, the broker encrypts the data, topic name, and timestamp by the public key of the subscriber, and forwards this encrypted message to the subscriber together with the signature.
(4) The subscribers receive the messages from the broker and decrypt the ciphertext by their own public key. Then, S checks the freshness by timestamp and also verifies the signature of the published message by the publisher's public key (the Falcon signature).

3.6 Proposed Solution for Secure Group Broadcast

At this level, the messages sent from publishers are efficiently distributed via broadcast to all subscribers. These messages are protected against modification and eavesdropping. This level is suitable for protecting vital user data on the whole communication chain but both sides need to be ready to perform both PQC signature and encryption algorithms. The advantage is that the broker can efficiently encrypt the message from the publisher only once by the established group key and send it to all subscribers instantly.

The communication phase of secure group broadcast (depicted in Figure 5) contains the following steps:

(1) Subscribers, who want to confidentially subscribe to the messages of chosen topics T within groups, send to the broker subscriber request messages with the requested security level 2 (SL2), their own public keys, and requested topic names. B collects all those requests and stores public keys from subscribers. Based on the same topics and groups, the broker pre-computes the group key $KGroup_T$ and creates the groups of subscribers for related topics with the relevant keys. The group key $KGroup_T$ has to be securely sent to each subscriber, i.e., encrypted by the subscriber's Kyber public key. To avoid false group key sending, the ciphertext is signed by the broker (by Falcon private key). The subscriber verifies this signature by the broker's public key. If the signature is ok, the subscriber stores this group key for the following encrypted messages.
(2) The publisher senses data and prepares the message for the MQTT protocol stack. The security procedure is equal to SL2, i.e., P firstly signs the plain data, topic, and actual timestamp t by its private key. Secondly, plain data are encrypted (by the broker's public key) and the ciphertext, signature, topic name and timestamp are sent to the broker.
(3) The broker receives the published message with the signature and ciphertext and firstly checks if the timestamp is OK. If the timestamp is fresh, B decrypts the ciphertext to plain

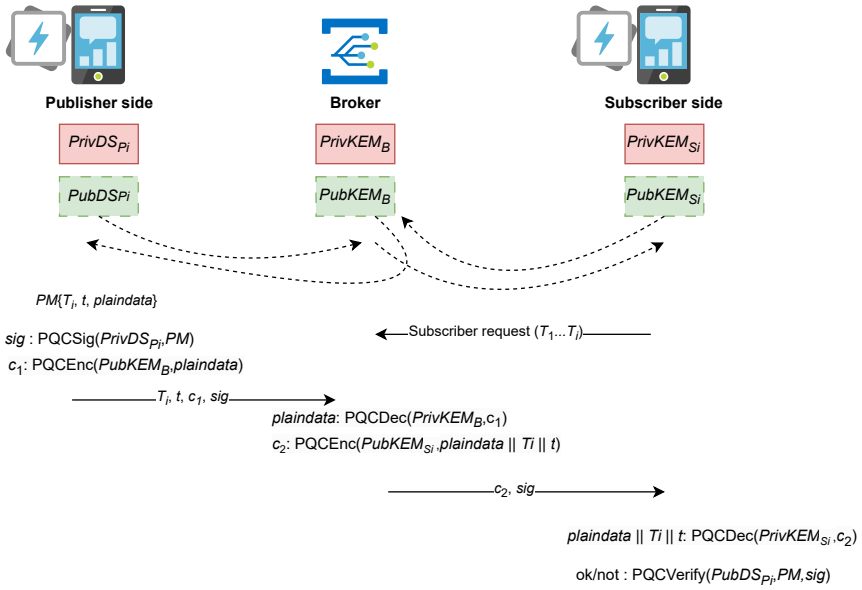

Figure 4: Communication Phase: SL2

data by his private key. Then, the broker encrypts the data, topic name, and timestamp by the established group key for subscribers' group (by AES-256 algorithm), and forwards this encrypted message to all subscribers together with the signature from the publisher.

(4) The subscribers receive the messages from the broker and decrypt the ciphertext by the established group key (by the AES-256 algorithm). Then, S checks the freshness by timestamp and also verifies the signature of the published message by the publisher's public key (the Falcon signature).

4 EXPERIMENTAL RESULTS AND PERFORMANCE EVALUATION

This section contains the performance assessments and experimental results of PQC schemes used in our solution.

4.1 Experimental Setup

In our measurement, we use these devices:

- Singleboard device - Raspberry PI Zero with 1 core, max 1GHz, and 512 MB RAM
- Smartphone Android device - CPU Samsung Exynos 1280, 8 cores, 2 - 2.4 GHz, 6 GB RAM
- Smartphone iOS device - CPU Apple A14 Bionic, 6 cores, max 3.1 GHz, 4 GB RAM

The chosen single-board ARM Cortex-A53 platform (Raspberry PI Zero with 1 core, max 1GHz, and 512 MB RAM) represents low-cost and less powerful devices. Smartphones represent handheld devices without computing constraints.

4.2 Performance Results

Table 4 represents the results of atomic operations used in our proposal, i.e., Kyber and Falcon. Besides the runtimes of basic operations, the sizes of keys and outputs are presented.

Table 4: Results of Falcon and Kyber for NIST security level 5 for Raspberry/Android/iOS devices.

Metric	Falcon	Kyber
Private Key [B]	2,305	3,168
Public Key [B]	1,793	1,568
Signature/Ciphertext [B]	1,330	1,568
Key generation op. [ms]	1,239/55.08/36.80	14.11/0.231/0.114
Signing/Encapsulation op. [ms]	145/10.98/11.00	17.17/0.175/0.069
Verification/Decapsulation op. [ms]	3.96/0.128/0.021	14.99/0.156/0.093

For the Falcon digital signature, the chosen size of input messages is 50 B, and for Kyber we use the standard 32 B inputs. The experimented results are average values from numerous repetitions. On Raspberry, 5 repetitions were performed, on smartphones, we performed 20 repetitions for Falcon and 100 repetitions for Kyber. The most constrained device Raspberry PI Zero requires 145 ms for signing a message by Falcon and only ca. 4 ms for its verification. On the same device, Kyber encapsulation and decapsulation operations require ca. 17 ms, 15 ms respectively. Besides key generation/setting which is usually the most demanding phase, the Falcon signing operation is the most expensive operation and takes around 11 ms on smartphones. When the publisher is a constrained device, Falcon's singing would be a significant bottleneck, that is why we also compare the impact of changing the signature scheme on the implemented communication phases SL1 and SL2. The results are presented in Figure 6. We can see that for the first scenario with constrained publishers, it might be more beneficial to use the Dilithium signature, as it offers a simpler signing phase. However,

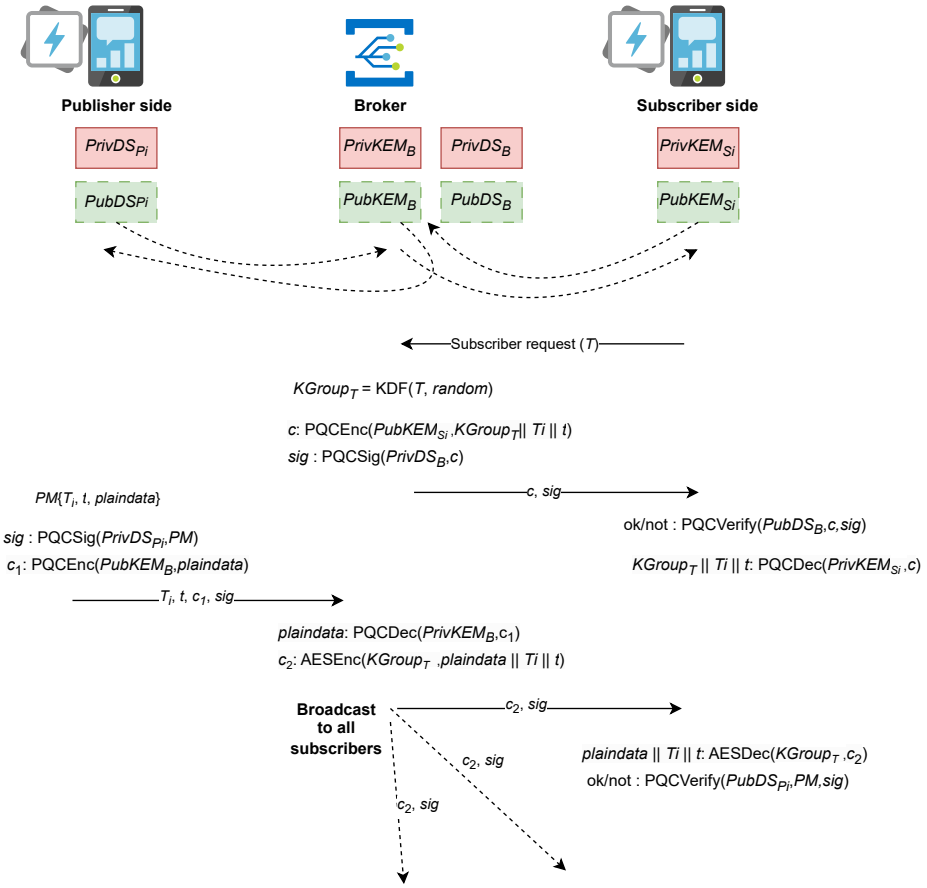

Figure 5: Communication Phase: secure group broadcast

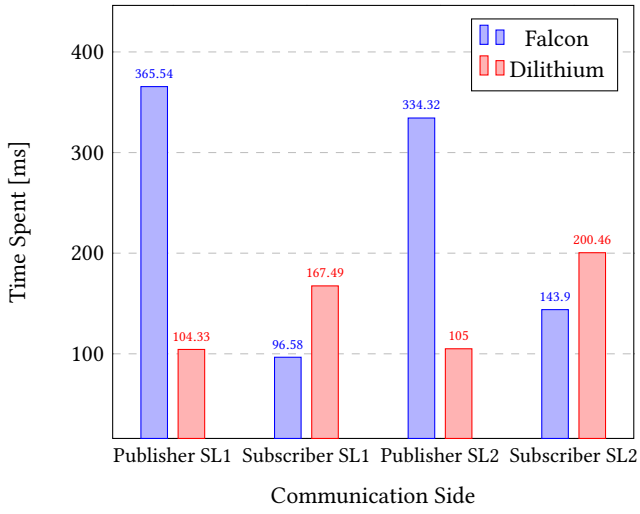

Figure 6: Comparison of Falcon and Dilithium with same security level for communication phases (Publisher - RPi Zero with 1 GHz CPU, Subscriber - RPi 1 with 700 MHz).

the choice may also depend on the available data bandwidth, as Dilithium's signature is more than three times larger than Falcon's.

5 CONCLUSION

In this paper, we focus on post-quantum safe IoT/MQTT communication. We analyze and map the current state of the security of MQTT and the approaches leading to its quantum resistance. Our designed proposal employs suitable post-quantum cryptographic schemes into an MQTT architecture that could have numerous scenarios. Three designed security levels provide various security properties that are based on the post-quantum cryptography algorithms. The experimental results show that PQC algorithms can be performed in practical times (typically tens milliseconds) even on constrained small devices. The solution is suitable for real-time applications using MQTT that require small delays (less than 300 ms) and require post-quantum security. Our future work focuses on getting more results on constrained platforms and adding more privacy properties for both parties subscribers and publishers.

ACKNOWLEDGMENTS

This work is supported by Ministry of the Interior of the Czech Republic under Grant VJ03030014.

REFERENCES

[1] Y. M. Agus, M. A. Murti, F. Kurniawan, N.D.W. Cahyani, and G.B. Satrya. 2020. An Efficient Implementation of NTRU Encryption in Post-Quantum Internet of Things. In *2020 27th International Conference on Telecommunications (ICT)*. 1–5. https://doi.org/10.1109/ICT49546.2020.9239560

[2] Nicolas Aragon, Paulo SLM Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, et al. 2017. BIKE: bit flipping key encapsulation. (2017).

[3] Jon Barton, Nikolaos Pitropakis, William Buchanan, Sarwar Sayeed, and Will Abramson. 2022. Post quantum cryptography analysis of TLS tunneling on a constrained device. In *In Proceedings of the 8th International Conference on Information Systems Security and Privacy - ICISSP*. 551–561.

[4] Daniel J Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. 2015. SPHINCS: practical stateless hash-based signatures. In *Advances in Cryptology–EUROCRYPT 2015: 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I 34*. Springer, 368–397.

[5] Joppe Bos, Leo Ducas, Eike Kiltz, T Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. 2018. CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. 353–367.

[6] Jessica Bozhko, Yacoub Hanna, Ricardo Harrilal-Parchment, Samet Tonyali, and Kemal Akkaya. 2023. Performance Evaluation of Quantum-Resistant TLS for Consumer IoT Devices. In *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*. IEEE, 230–235.

[7] Tung Chou, Carlos Cid, S UiB, J Gilcher, T Lange, V Maram, R Misoczki, R Niederhagen, KG Paterson, and E Persichetti. 2020. Classic McEliece: conservative code-based cryptography, 10 October 2020.

[8] Chia-Chin Chung, Chu-Chi Pai, Fu-Shiang Ching, Chao Wang, and Ling-Jyh Chen. 2022. When post-quantum cryptography meets the Internet of Things: An empirical study. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*. 525–526.

[9] Markus Dahlmanns, Jan Pennekamp, Ina Berenice Fink, Bernd Schoolmann, Klaus Wehrle, and Martin Henze. 2021. Transparent end-to-end security for publish/subscribe communication in cyber-physical systems. In *Proceedings of the 2021 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems*. 78–87.

[10] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. 2018. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018, 1 (Feb. 2018), 238–268. https://doi.org/10.13154/tches.v2018.i1.238-268

[11] Armando Faz-Hernández and Kris Kwiatkowski. 2019. *Introducing CIRCL: An Advanced Cryptographic Library*. Cloudflare. Available at https://github.com/cloudflare/circl. v1.3.3 Accessed May, 2023.

[12] Federal Office for Information Security (BSI). 2021. Quantum-safe cryptography – fundamentals, current developments and recommendations.

[13] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, Zhenfei Zhang, et al. 2018. Falcon: Fast-Fourier lattice-based compact signatures over NTRU. *Submission to the NIST's post-quantum cryptography standardization process* 36, 5 (2018).

[14] Mohammad Hamad, Andreas Finkenzeller, Hangmao Liu, Jan Lauinger, Vassilis Prevelakis, and Sebastian Steinhorst. 2023. SEEMQTT: Secure End-to-End MQTT-Based Communication for Mobile IoT Systems Using Secret Sharing and Trust Delegation. *IEEE Internet of Things Journal* 10, 4 (2023), 3384–3406. https://doi.org/10.1109/JIOT.2022.3221857

[15] Lukas Malina, Gautam Srivastava, Petr Dzurenda, Jan Hajny, and Radek Fujdiak. 2019. A secure publish/subscribe protocol for internet of things. In *Proceedings of the 14th international conference on availability, reliability and security*. 1–10.

[16] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loıc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and IC Bourges. 2018. Hamming quasi-cyclic (HQC). *NIST PQC Round* 2, 4 (2018), 13.

[17] Felipe José Aguiar Rampazzo and Marco Aurélio Amaral Henriques. 2023. Assessment of the Impact of Hybrid Post-Quantum Cryptography on the Performance of the MQTT Communication Protocol. In *2023 Symposium on Internet of Things (SIoT)*. IEEE, 1–5.

[18] Mélissa Rossi. 2023. PQC TRANSITION IN FRANCE ANSSI VIEWS. In *Real World Post-Quantum Crypto*.

[19] Kumar Sekhar Roy and Hemanta Kumar Kalita. 2019. A survey on post-quantum cryptography for constrained devices. *International Journal of Applied Engineering Research* 14, 11 (2019), 2608–2615.

[20] PC Sajimon, Kurunandan Jain, and Prabhakar Krishnan. 2022. Analysis of Post-Quantum Cryptography for Internet of Things. In *2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS)*. IEEE, 387–394.

[21] Eduardo Buetas Sanjuan, Ismael Abad Cardiel, Jose A Cerrada, and Carlos Cerrada. 2020. Message queuing telemetry transport (MQTT) security: A cryptographic smart card approach. *IEEE Access* 8 (2020), 115051–115062.

[22] Maximilian Schöffel, Frederik Lauer, Carl C Rheinländer, and Norbert Wehn. 2021. On the energy costs of post-quantum kems in tls-based low-power secure iot. In *Proceedings of the International Conference on Internet-of-Things Design and Implementation*. 158–168.