Internet of Things 37 (2026) 101947

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

Quantum-resistant pairing evaluation for resource-constrained bluetooth classic

Pablo Gutierrez-Felix [a,*], Marc Manzano [b], Javier Lopez [a]

[a] Network, Information and Computer Security (NICS) lab, Universidad de Málaga, Málaga, Spain
[b] SandboxAQ, Bilbao, Spain

ARTICLE INFO

Keywords:
Post-quantum cryptography
Bluetooth classic
Quantum-resistant protocols
Embedded systems
Constrained devices

ABSTRACT

The security of state-of-the-art Bluetooth pairing relies on Elliptic Curve Diffie-Hellman (ECDH), which is vulnerable to quantum attacks. However, the feasibility of integrating standardized Post-Quantum Cryptographic (PQC) primitives into the controller-level Bluetooth Classic (BC) protocol stack remains an open question. This paper presents the first comprehensive evaluation of integrating a post-quantum secure key exchange for BC pairing. We analyze the impact of replacing the existing ECDH-based key establishment with NIST-standardized PQC Key Encapsulation Mechanisms (KEMs), focusing on protocol-level packetization, over-the-air transmission overhead, and execution performance in constrained hardware. Using BC's Link Manager Protocol (LMP) and DM1 packet structure, we quantify Public Parameter Transmission (PPT) latency for classical and post-quantum schemes. We further benchmark ECDH and ML-KEM implementations on an ARM Cortex-M4 microcontroller with characteristics comparable to those of constrained Bluetooth controllers. Our results show that ML-KEM is computationally feasible on such hardware, often matching or outperforming classical ECDH in execution time. However, PQC significantly increases over-the-air overhead. Under ML-KEM, PPT accounts for 60-75% of total key exchange latency, compared to less than 10% for ECDH. Code-based schemes such as HQC are shown to be impractical due to excessive memory and transmission requirements under provided hardware constraints. These findings indicate that the primary obstacle to post-quantum Bluetooth pairing is not computation but wireless packetization, reliability, and energy consumption; facilitating a safe and efficient quantum-secure migration of resource-constrained wireless communication settings.

1. Introduction

Bluetooth is one of the most widely adopted wireless communication technologies, enabling short-range connectivity across billions of devices, from consumer electronics to industrial sensors [1]. Its wide adoption makes its security infrastructure a critical target for emerging threats, including those posed by quantum computing. The key exchange algorithms currently employed in Bluetooth's latest and safest security mode, Secure Connections (SC), rely on Elliptic Curve Cryptography (ECC) [2, Vol. 2 Section H]. Consequently, these primitives are vulnerable to Shor's algorithm [3] once a Cryptographically Relevant Quantum Computer (CRQC) becomes available. As a result, ensuring the long-term security of Bluetooth networks requires migrating the key exchange involved during Bluetooth's pairing processes to a quantum-resistant scheme.

* Corresponding author.
E-mail addresses: pablogf@uma.es (P. Gutierrez-Felix), marc@sandboxaq.com (M. Manzano), javierlopez@uma.es (J. Lopez).

https://doi.org/10.1016/j.iot.2026.101947
Received 23 February 2026; Received in revised form 30 March 2026; Accepted 10 April 2026
Available online 16 April 2026

Despite significant advances in PQC standardization [4,5], the integration of quantum-safe algorithms into Bluetooth remains largely unexplored. Existing work either applies non-standard or experimental PQC primitives without practical implementation [6], or restricts post-quantum protections to application-layer communications after pairing, such as TLS handshakes [7,8]. To date, only a few studies have analyzed PQC in the context of Bluetooth Low Energy (BLE) key exchanges, typically focusing on packet-level feasibility rather than providing a fully operational protocol [9]. No prior work systematically examines the technical, protocol-level, and performance implications of migrating BC pairing to standardized PQC schemes. Furthermore, to the best of our knowledge, this is the first accurate comparison between ML-KEM and the Bluetooth-mandated P-192 and P-256 curves on ARM Cortex-M4 class hardware under realistic Bluetooth memory constraints, expanding the existing benchmarking literature regarding the comparison of PQC algorithms against their classical counterparts. Due to the limited research concerning the PQC migration of BC, our study focuses precisely on that, where pairing operations occur at the controller level and are constrained by strict over-the-air packetization and resource limitations. We quantify both the computational feasibility and the communication overhead introduced by standardized lattice and code-based KEMs, providing concrete performance benchmarks for ARM Cortex-M4 microcontrollers representative of resource-constrained Bluetooth controllers.

The remainder of this paper is organized as follows. Section 2 provides background knowledge on Bluetooth concepts, cryptographic mechanisms, and the PQC standards. Section 3 reviews related work on PQC integration in Bluetooth and IoT communication, as well as the contributions provided by this study. Section 4 outlines the process to migrate the internal cryptography of BC, focusing on public parameter transmission requirements. Section 5 presents the experiments, evaluates performance, packetization, and timing implications of PQC key exchanges on the proposed testing architecture. Section 6 contextualizes our results within the Internet of Things (IOT) ecosystem and analyzes the impact of our study on other IOT protocols. Finally, Section 7 summarizes the findings, discusses practical implications, and outlines directions for future research.

2. Background

2.1. Bluetooth types

Bluetooth is a widely adopted technology used for short-distance wireless communication, maintained by the Bluetooth Special Interest Group (SIG). As outlined in Bluetooth's specification document (version 6.2 as of today) [2], there exist two types of Bluetooth: BC and BLE. Both versions work on the 2.4GHz microwave spectrum, dividing it into a set of channels depending on the Bluetooth type. Through a pairing algorithm, a sequence of channels is outlined, and the devices hop on and off the different channels to communicate information. A channel is used for a very short period, followed by a hop to another channel designated by a pre-determined pseudo-random sequence; this process is repeated continuously in the frequency hopping sequence. Devices operating in the communication process are assigned roles depending on their engagement during the protocol and capabilities. The central scans for potential devices to communicate with and initiates the security procedure, whereas the peripheral advertises its availability to connect and accepts a connection request from a central. As the technology evolved, the two previously mentioned variants of Bluetooth emerged, each designed for a specific communication purpose.

2.1.1. Bluetooth classic

First released in 1999 with Bluetooth 1.0, this was the first version of Bluetooth ever designed, mainly used for audio-streaming and data transfer purposes [10] . It is also referred to as "Basic Rate" with an option for "Enhanced Data Rate" (BR/EDR) which improves data throughput to 2–3 Mb/s using more efficient modulation schemes for larger multi-slot packets [2, Vol. 2 Section A, Chapter 3]. However, due to the packets used for Bluetooth pairing, the EDR option does not provide any enhancement during the cryptographic key exchange, which is the main focus of the presented research. BC is mainly used for audio-streaming and data transfer, like wireless headsets [10]. It breaks down the 2.4 GHz band into 79 channels, 32 of which are inquiry channels. The central creates a hopping sequence generated from its static and unique Bluetooth address (48 bits) and its clock. This combination creates a pseudo-random hop pattern across the 79 channels that both devices will follow during communication. The procedure starts with the peripheral sending inquiry packets into the 32 inquiry channels in "inquiry scan mode" of the central. When the central receives an inquiry packet from the peripheral, it responds with its Bluetooth address and clock, initiating the connection request. With that information as input, the peripheral accepts the request and obtains the hopping path through the hop selection kernel deterministic algorithm outlined in the specification [2, Vol. 2 Part B Section 2.6].

2.1.2. Bluetooth low energy

BLE was introduced in 2009 (Bluetooth 4.0), although BC versions continued to be released. This type of Bluetooth is oriented towards low-power devices, such as fitness accessories. It breaks down the 2.4 GHz band into 40 channels. It reserves three special advertising channels (number 37, 38, and 39) to spread across the 2.4 GHz band. To start, the peripheral sends out advertising packets on those three channels. The central listens and sends the peripheral a connect request through that same channel. This request contains the channel map, the hop increment, the starting channel, and the timing information so that the peripheral can accurately follow the streamed information provided by the central. Then, they switch to channel 37 and start hopping based on the predefined route set by the central in the connect request.

Table 1
BC and BLE SC underlying security primitives.

Security Aspect	Classic	Low Energy
Bluetooth Versions	v4.2 onwards	v4.2 onwards
Key Exchange	P-192/P-256	P-256
Device Authentication	HMAC-SHA-256	AES-CMAC
Encryption	AES-CCM	AES-CCM
Association Models	JW, NC, PE, OOB	JW, NC, PE, OOB

2.2. Cryptography in bluetooth

When two devices communicate over Bluetooth, the cryptographic procedures differ depending on whether the relationship is temporary or long-term. For a one-time connection, the devices can establish a temporary encryption key that is discarded after disconnection. If the intention is to communicate during future connections, the devices go through the pairing procedure, where they agree on a common Link Key (LK) (in BC) or Long Term Key (LTK) (under BLE) after they have been synchronized through the channel-hopping process. During bonding, the LK or LTK is stored so that future connections between the same devices can be re-established without repeating the full pairing procedure, reusing the stored key material to derive encryption keys for subsequent sessions.

Over time, several security mechanisms have been proposed to verify that the communication process carried by Bluetooth is cryptographically secure. These mechanisms vary depending on the type of Bluetooth, yet both BC and BLE cryptographic security are very similar except for some minor differences, as outlined by the official Bluetooth specification and emphasized by National Institute of Standards and Technology (NIST) [11]. Although each Bluetooth type had several previous mechanisms, they both share the latest released version, Secure Connections (SC), which enjoys the highest security features of all Bluetooth security mechanisms. Furthermore, SC proposes different association models that allow devices to authenticate engaging in the pairing process to establish trust before deriving cryptographic keys. These association models are enabled depending on the devices capabilities:

- **Numeric Comparison (NC)**: Designed for settings in which both devices have input/output capabilities to display 6-digit numbers and to select OK/Reject. The user is asked whether the numbers displayed on both machines match and should click "OK" if they do, a common example when pairing a mobile phone with a vehicle's Bluetooth system. It should be noted that the six-digit number is not related to the private key required to decrypt the communication, but it is rather a confirmation value derived from the shared secret.
- **Passkey Entry (PE)**: One device has a keyboard but no display, while the other has at least a display option. The device with the display outputs a number, which must be entered into the other device using the keyboard. This is the classic scenario of connecting a PC and a keyboard.
- **Just Works (JW)**: Designed for situations in which one of the devices does not have a display nor keyboard, such as wireless earphones. Because the user does not have any way to check the validity of the pairing process through a display, this association model is by design subject to Man-in-the-Middle (MITM) attacks. This is the case when pairing a mobile phone with an audio speaker.
- **Out-of-Band (OOB)**: Cryptographic values needed for Bluetooth pairing (public keys, nonces, or temporary keys) are exchanged over a separate communication channel such as NFC or QR code scanning, rather than using the Bluetooth radio itself.

Table 1 below summarizes the underlying security primitives of SC in BC and BLE [11]. As observed, the use of ECDH during key agreement opens the door to potential quantum attacks such as the one suggested by Shor in [3]. Consequently, the security mechanisms globally adopted since Bluetooth version 4.2 (for both BC and BLE) will become obsolete once quantum computers can execute Shor's algorithm to solve the underlying elliptic curve discrete logarithm problems.

2.3. Post-quantum cryptography

NIST has recently published drafts of the Federal Information Processing Standards (FIPS) that establish the next generation of post-quantum cryptographic algorithms. The standardization process comprehended a competition across a wide range of quantum-secure key exchange and digital signature algorithms. Specifically, FIPS-203 defines the *Module-Lattice Key Encapsulation Mechanism (ML-KEM)*, the current quantum-secure standard for key-exchanges. Nonetheless, recently, the NIST IR-8545 announced the selection for standardization of the *Hamming Quasi-Cyclic Key Encapsulation Mechanism (HQC)*, a code-based alternative relying on decoding hardness assumptions [4][5]. Consequently, the soon-to-be standardized HQC algorithm relies on a distinct mathematical foundation from that of ML-KEM, thereby ensuring that, in the event lattice-based schemes become vulnerable to future breakthroughs, a quantum-resistant alternative remains available for key exchange protocols.

ML-KEM originates from the *CRYSTALS-Kyber* lattice-based algorithm, which is founded on the Module Learning with Errors (MLWE) problem, and believed to be secure against quantum attacks. This problem consists of finding a secret vector $s \in R_q^k$ in the

polynomial ring $Z_q[x]/(x^n+1)$ given (A, b) such that $b = A \cdot s + e \bmod q$ where $A \in R_q^{k \times k}$ is a random matrix, $e \in R_q^k$ a random vector and q a prime number. ML-KEM converts a regular Public-Key Encryption (PKE) scheme based on the MLWE problem into a KEM using the Fujisaki-Okamoto (FO) transform, providing the resultant primitive with IND-CCA2 security [12][13]. Table 2 specifies the parameter sets for each security level of ML-KEM. The parameter k determines the dimension of the polynomial vectors and matrices, thereby expanding the effective dimension of the MLWE problem. Parameters η_1 and η_2 control the distribution for generating s, e, e_1, e_2 and y during key generation and encryption. Finally, parameters d_u and d_v are the parameters and inputs for several intermediate functions within the encryption and decryption process of ML-KEM. Furthermore, ML-KEM includes different features that accelerate the value computation process, such as Number-Theoretic Transform (NTT) accelerated multiplication, SHAKE-derived parameter derivation, and sampling from the centered binomial distribution. These refinements result in a fast and lightweight cryptographic system ideal for securing resource-constrained devices against quantum attacks.

HQC derives its security from the hardness of the *decisional Quasi-Cyclic Syndrome Decoding* (DQCSD) problem. The scheme relies on systematic double-circulant quasi-cyclic codes with parity-check matrix $H = (I_n \mid \mathrm{rot}(h))$ which in polynomial representation corresponds to $(1 \mid h) \in R^2$, where $R = \mathbb{F}_2[X]/(X^n - 1)$. During key generation, the secret key consists of low-weight vectors $x, y \in R_\omega$, while the public key contains $h \in R$ and the syndrome $s = x + h \cdot y$, which is equivalent to the syndrome $H(x, y)^\top$. While security relies on the hardness of quasi-cyclic syndrome decoding, decryption correctness is achieved using an auxiliary concatenated Reed-Muller/Reed-Solomon code. HQC also transforms the original IND-CPA PKE scheme into a IND-CCA2 KEM by using the salted FO transform with implicit rejection. Table 2 lists parameters for HQC, where n_1 represents the length of the Reed-Solomon code and n_2 the length of the Reed-Muller code, yielding concatenated code C of length $n = n_1 n_2$ and dimension k. Parameter n denotes the ambient space length, the smallest primitive prime exceeding $n_1 n_2$. Meanwhile, ω, ω_r, and ω_e specify Hamming weights for key vectors $(\mathbf{x}, \mathbf{y})$, encryption randomness $(\mathbf{r}_1, \mathbf{r}_2)$, and error vector $\mathbf{e}$, respectively [14].

Table 2
Parameter sets for ML-KEM and HQC-KEM.

Algorithm	n	k	q	η_1	η_2	(d_u, d_v)
ML-KEM-512	256	2	3329	3	2	(10, 4)
ML-KEM-768	256	3	3329	2	2	(10, 4)
ML-KEM-1024	256	4	3329	2	2	(11, 5)
Algorithm	n_1	n_2	n	k	ω	$\omega_r = \omega_e$
HQC-128	46	384	17,669	128	66	75
HQC-192	56	640	35,851	192	100	114
HQC-256	90	640	57,637	256	131	149

Table 3
Security level, variable sizes (bytes), and decapsulation failure rates for ML-KEM and HQC variants.

Algorithm	NIST Security Level	Public key	Secret key	Ciphertext	Failure rate
ML-KEM-512	1	800	1632	768	$2^{-138.8}$
ML-KEM-768	3	1184	2400	1088	$2^{-164.8}$
ML-KEM-1024	5	1568	3168	1568	$2^{-174.8}$
HQC-128	1	2249	2289	4481	2^{-128}
HQC-192	3	4522	4591	9026	2^{-192}
HQC-256	5	7245	7331	14,469	2^{-256}

ML-KEM and HQC represent complementary design families in post-quantum cryptography. Table 3 contains further details concerning their security level, variable sizes, and decapsulation failure rates. Although inherently larger in variable sizes than its lattice-based counterpart, HQC provides diversification across distinct post-quantum hardness assumptions, reason why it is in process of standardization as a backup algorithm for ML-KEM.

3. Related work

Some previous work has introduced PQC constructions into Bluetooth and the environment surrounding it. Although not many of them tackle the pairing process, significant progress has been made in combining Bluetooth constructions and PQC constructions for standardized use.

An example is the paper introduced by [6], where the authors utilize the lattice-based algorithm NewHope, which has not progressed beyond the second round of the NIST competition. However, they provide a theoretical approach to constructing a Bluetooth pairing process around a key encapsulation mechanism. Several works have delved into the migration process of the TLS handshake performed once the Bluetooth key exchange has taken place. Providing security through post-quantum TLS to a Bluetooth Link is not inherently a wrong approach in the near-term, but they only cover Bluetooth use cases engaging in TLS connections. Furthermore, once quantum computers are widely available, the objective must be securing the entire protocol on its own, rather than relying on external protocols for Bluetooth security. In [7], the authors test the performance of several PQC KEMs and signatures when used in communications over the internet using BLE and Wi-Fi. Later, in [8], they migrated the TLS handshake between a Bluetooth IOT device and the gateway, providing experimental results tested on two Raspberry Pis. However, rather than redesigning the Bluetooth pairing process, these approaches apply PQC algorithms in the TLS handshake over different network types, notably BLE and Wi-Fi, after the devices have already paired. With respect to the PQC migration of the pairing process and the key exchange, the only existing work that was found is [9]. Throughout the study, the authors focus on a PQC-based key exchange for BLE using Kyber512. The experiments analyze the pairing time by altering the Attribute Protocol Maximum Transmission Unit (ATTMTU) size so that Kyber's 800-byte public key and 768-byte ciphertext are successfully transmitted given the 512-byte constraint posed by ATTMTU. The results show that larger ATTMTU sizes result in a faster PQC key exchange. Nonetheless, this paper does not present an operable quantum-resistant protocol, but rather an analysis of the sizing and timing of packets carrying the public-key and ciphertext exchange over BLE. Moreover, the study explicitly utilizes the Just-Works security mechanism, which does not prevent MITM attacks.

In addition to Bluetooth's migration to PQC, this paper focuses on its feasibility in constrained embedded environments, as is the case of widespread Bluetooth controllers [15]. Originally, many proposals approached the challenge of implementing PQC primitives in constrained devices, such as [16] and [17] for lattice-based primitives. The *pqm4* framework and accompanying studies encompassed these and other NIST PQC candidates into a single work, providing cycle-accurate measurements, as well as memory and stack usage on ARM Cortex-M4 devices [18][19]. Nonetheless, the results portrayed are limited to PQC schemes in isolation and do not include direct comparisons with classical elliptic curve primitives such as the SIG-mandated *secp192r1* or *secp256r1*. In [20], a performance evaluation of Kyber (ML-KEM's foundational primitive) is conducted on the Raspberry Pi 3 Model B, including a comparison with the binary curves *sect283r1*, *sect409r1*, and *sect571r1*. Unfortunately, these classical curves are irrelevant to Bluetooth pairing. The authors in [21] provide a broad performance evaluation of PQC in next-generation consumer electronics, measuring execution time, communication overhead, and throughput of PQC schemes relative to classical ECC/RSA across platforms. However, this study is not focused on cycle-accurate evaluation on microcontrollers and does not target Bluetooth-specific elliptic curves, leaving a gap that our work addresses. Other non-peer reviewed industry comparisons between classical and PQC exchanges have been conducted, such as the one posted by WolfSSL [22], illustrating the clear need for further benchmarking data in the PQC research ecosystem.

3.1. Contributions

Despite significant progress, the quantum-safe migration of Bluetooth's pairing processes remains largely unexplored. This study provides a comprehensive evaluation of the challenges and feasibility of integrating standardized PQC schemes into BC pairing. The main contributions of this work are:

- **Controller-level PQC analysis:** We present the first detailed study of migrating the key exchange in BC to NIST-standardized PQC KEMs at the controller level, explicitly considering the Bluetooth SIG protocol stack and LMP-layer mechanisms.
- **Packetization and over-the-air evaluation:** We quantify the transmission requirements for pre-quantum (P-192 and P-256) and post-quantum (ML-KEM and HQC) key exchanges, including DM1 packet counts, LMP encapsulated PDU usage, and corresponding pairing latency. This analysis identifies the significant communication overhead introduced by PQC and highlights its impact on reliability and energy consumption.
- **Empirical performance benchmarking:** We perform cycle-accurate measurements of ECDH and ML-KEM algorithms on an ARM Cortex-M4 platform representative of constrained Bluetooth controllers. The results demonstrate that PQC execution is computationally feasible for ML-KEM variants, with key generation and cryptographic operations completing within acceptable time frames. We also identify the infeasibility of implementing HQC in highly-constrained environments due to RAM insufficiency.
- **Identification of protocol-level challenges:** We show that, while computationally outperforming traditional algorithms in execution time, an ML-KEM-based migration significantly increases pairing payloads and airtime, introducing heightened vulnerability to packet loss, retransmissions, and potential DoS attacks. This finding underscores the need to consider wireless reliability and energy constraints alongside cryptographic strength.
- **Comparative contextualization within the IOT ecosystem:** We provide a comprehensive cross-protocol analysis of ten major wireless standards against the insights obtained from the experiments performed in this study, thus providing a foundational approach for the quantum-safe transition of diverse IOT wireless protocols.

Taken together, these contributions provide a rigorous technical evaluation into the efficient migration of BC pairing to quantum-safe cryptography.

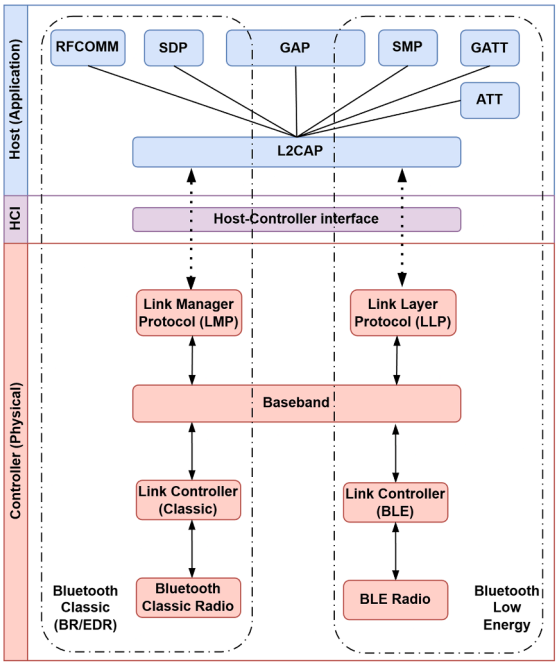

Fig. 1. Bluetooth system architecture.

4. Internal migration of BC

To establish a Bluetooth connection between several devices, Bluetooth's protocol stack is organized in different layers. Each of these layers is tasked with specific objectives that enable wireless communication using Bluetooth. Similar to the TCP/IP model, these layers can be progressively structured from lower-level hardware-related layers (also referred to as the controller) to higher-level software-related layers (host). The controller is implemented into a hardware module within the Bluetooth device, and the host is embedded through the operating system [23]. The Host Controller Interface (HCI) layer is tasked with integrating the Bluetooth controller and the Bluetooth host. It is essentially a protocol that physically sends data over USB/UART/SD through a physical bus and is present in the host as a driver and in the controller as firmware, which successfully connects both ends.

Each layer contains different elements depending on the Bluetooth type at hand. Whereas BC and BLE share many similarities in the controller architecture, they may execute radically different host protocols. Therefore, as depicted in Fig. 1, the Bluetooth protocol stack is complete from the hardware to the software level. Due to structural differences in the construction and implementation of each Bluetooth type, particularly in the cryptographic aspect, BC and BLE must be analyzed separately. This is mainly because BLE handles the cryptographic exchange at the host level, while BC at the controller level. As a result, the analysis provided in this study solely focuses on BC.

Under BC, most of the cryptographic work takes place in the LMP layer, within the controller. Each vendor utilizes the Bluetooth SIG specification to develop Bluetooth hardware and firmware. With respect to the cryptographic guidelines, all details regarding the key exchange in BC is thoroughly detailed in [2, Vol. 3 Section C Chapter 4]. Migrating BC to support post-quantum key establishment would therefore require modifications to the controller such that the LMP layer can execute the operations of a PQC-KEM. In the current ECDH design, the peers exchange public keys and derive a shared secret independently, as illustrated in Fig. 2.

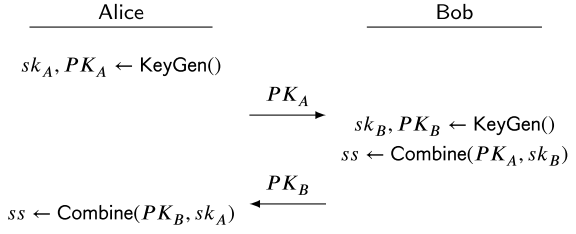

Fig. 2. ECDH protocol.

Unlike ECDH, KEMs operate differently. In a typical KEM protocol, Alice first transmits her public key to Bob. He then generates a random secret, encrypts it (usually referred to as *encapsulating*) using Alice's public key, and sends the resulting ciphertext back

to Alice. Upon receiving it, Alice decrypts (or *decapsulates*) the ciphertext with her private key to recover the shared secret. Fig. 3 outlines the steps of a KEM protocol. Noticeably, the number of message exchanges carrying public information is not altered, since both ECDH-based and KEM-based approaches rely on a single round trip. The variables exchanged do increase in size substantially in the case of PQC-KEMs, nonetheless, posing a significant challenge for the LMP layer handling the cryptographic key exchange.

Similar to ECDH, PQC-KEMs are subject to MITM attacks, an issue that is tackled through the aforementioned association models. It is crucial to highlight that Bluetooth does not rely on any PKI to authenticate the ends engaging during the communication process. As a result, no PQC signatures are required for Bluetooth, reducing the migration task to the integration of a PQC-KEM instead of ECDH.

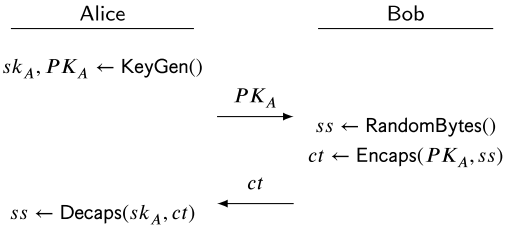

Fig. 3. KEM protocol.

The LMP layer transmits the key exchange public parameters using DM1 baseband packets, a specific type of Asynchronous Connectionless (ACL) packet comprised of 3 elements under the mandatory mode Basic Rate: Access code (68/72 bits), header (54 bits), and payload (18 bytes). The payload, where the cryptographic data will be stored, is formed by a payload header (1 byte), user payload (17 bytes) and 16 bits used for CRC [2, Vol. 2 Section B Chapter 6]. The LMP uses the `LMP_ENCAPSULATED_HEADER` (4 bytes) and `LMP_ENCAPSULATED_PAYLOAD` (17 bytes) PDUs to send the public parameters of the key exchange, where the first byte in both cases is reserved for the packet's opcode which helps identify the PDU. The first PDU mentioned is used to announce the start of an encapsulated PDU, while the second is used to relay the actual data [2, Vol. 2 Section C Chapters 2 and 4]. Fig. 4 depicts the composition of these LMP PDUs within DM1 packets. By using this 2-PDU approach, the LMP is able to transmit larger sets of data, such as ECDH public keys, which are significantly larger than the user payload allowed by a DM1 packet.

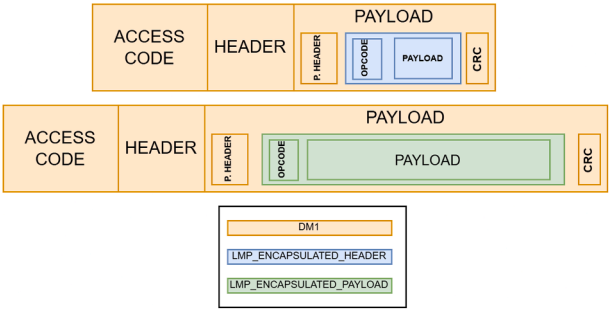

Fig. 4. BC PDUs used for cryptographic PPT.

Consequently, the cryptographic public parameters must be sent 16 bytes at a time using the `LMP_ENCAPSULATED_PAYLOAD` PDU. Table 4 presents the byte sizes of the public parameters used by the classical algorithms currently employed in BC for the ECDH key exchange. Note that, the Bluetooth documentation specifies that a valid public key is of the form $Q = (XQ, YQ)$, suggesting the use of uncompressed public keys for ECDH [2, Vol. 2 Section H Chapter 6]. Thus, ECDH public keys are formed by the X and Y coordinates of the curve point, where each coordinate has a size in bytes equivalent to the field size. The ML-KEM and HQC schemes are included as the post-quantum counterparts, since, as noted earlier, NIST selected them as the PQC KEMs standards. Besides, Table 4 also reports the number of DM1 packets required in order to exchange each instance of the public parameter. This metric includes 1 packet for the `LMP_ENCAPSULATED_HEADER` PDU and the rounded up number of `LMP_ENCAPSULATED_PAYLOAD` PDUs required to successfully transmit the entirety of the PPT. Observe that PQC schemes incur PPT overhead for one public key plus a ciphertext, whereas ECDH requires two public keys (one per party) but no ciphertext, owing to the inherent protocol design. In ECDH, each device independently generates its public/private key pair; the initiator sends its public key first, followed by the responder's reply only after receipt [2, Vol. 2 Section H Chapter 7.1]. Both parties compute the key generation process simultaneously during pairing (relevant later), but they engage in a sequential transmission of the public keys doubles the exchange time in PPT measurements.

Bluetooth devices utilize a native clock to manage various communication processes. This clock divides time into slots of 625 μs, which devices use to synchronize data packet transmissions. The central device transmits in even-numbered slots, while the peripheral transmits in odd-numbered slots [2, Vol. 2 Section B Chapter 2]. DM1 packets occupy a single transmission slot and are protected by ARQ, a baseband mechanism for packet acknowledgment and retransmission in case of corrupted content within DM1

Table 4
Packetization of public parameters for classical and PQC KEM algorithms in DM1.

Algorithm	Public Key	Ciphertext	DM1 Packet Count
P-192	48	–	4[a]
P-256	64	–	5[b]
ML-KEM-512	800	768	51/49
ML-KEM-768	1184	1088	75/69
ML-KEM-1024	1568	1568	98/98
HQC-128	2249	4433	142/279
HQC-192	4522	8978	284/563
HQC-256	7245	14,421	454/903

[a] The total roundtrip packets exchanged is 8, since each party must share their public key with the other party.
[b] Similarly, the total packets transferred is 10.

packets. After a sender transmits a DM1 packet, the receiver must respond in the subsequent slot with an ARQN acknowledgment bit. A negative acknowledgment (ARQN = 0) indicates an unsuccessful packet reception, typically due to payload CRC failure, packet loss, or header corruption. This prompts a retransmission of the same packet. Upon successful checks, the payload is accepted and positively acknowledged (ARQN = 1) in the following transmission slot. Because ARQ is mandatory for all DM packets, each DM1 transmission requires a return slot. Therefore, under ideal channel conditions, each DM1 packet consumes a minimum of two 625 μs slots, one for the corresponding DM1 packet containing the cryptographic material and one for the ARQ acknowledgment, totaling 1.25 ms per DM1 packet transmitted [2, Vol. 2 Section B Chapter 6].

Fig. 5 compares the total time required for PPT for each analyzed algorithm in the described scenario. Intuitively, larger parameter sizes naturally lead to longer transmission times. This is especially pronounced in high-security variants: ML-KEM-1024 executes it in 245 ms, HQC-256 in more than 1.5 s, while the classical P-256 needs only 6.2 ms. The time taken by ML-KEM variants is evenly distributed between the public key and ciphertext exchange, each consistently accounting for approximately 50% of the total latency. In contrast, HQC spends about $\frac{1}{3}$ of the total time sending the public key, with the remaining $\frac{2}{3}$ allocated to the ciphertext.

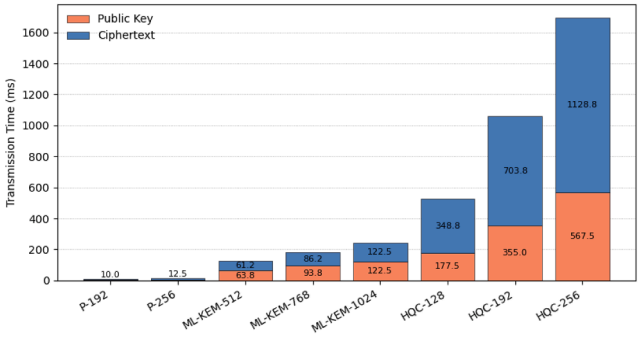

Fig. 5. Comparison of LMP packet transmission times during PPT for ECDH, ML-KEM, and HQC variants.

Thus, PQC algorithms may require more over-the-air overhead, raising several potential concerns. Increased DM1 packet usage during the exchange increases the pairing process's vulnerability to packet interference, linearly increasing the probability of corrupted packets with the total amount of DM1 packets exchanged. Corrupted DM1 content results in CRC failures, prompting the receiver to issue negative acknowledgments and request retransmissions. For PQC schemes involving large public keys and ciphertexts, this could occur multiple times during PPT, significantly increasing the pairing process latency. Another concern is the intensive use of radio-frequency signals during PPT, which may substantially increase radio power consumption during pairing. This is particularly problematic for low-power peripherals with limited battery life. Finally, longer PPT times under PQC schemes also provide more opportunities for DoS attacks, as attackers have a wider window to disrupt communication through selective jamming or by forcing retries and timeouts, potentially leading to a "pairing never completes" scenario.

5. Experiments

PPT is only one variable influencing the timing analysis of the protocol. The execution times of each algorithm's internal functions also significantly impact the total time required for the key exchange. Therefore, several experiments were conducted to accurately

Table 5
Cycle counts (cc) and execution time (ms) per algorithm function.

Algorithm	Build	PPT (ms)	keygen (cc)	keygen (ms)	combine (cc)	combine (ms)	encaps (cc)	encaps (ms)	decaps (cc)	decaps (ms)	TOTAL (ms)	PPT%
P-192	default	10.0	3,742,977	104.0	3,741,528	103.9	–	–	–	–	217.9	5%
P-192	optimized	10.0	1,974,939	54.9	1,973,691	54.8	–	–	–	–	119.7	8%
P-256	default	12.5	9,933,674	275.9	9,931,275	275.9	–	–	–	–	564.0	2%
P-256	optimized	12.5	5,763,536	160.1	5,761,502	160.0	–	–	–	–	332.5	4%
ML-KEM-512	clean	125.0	642,211	17.8	–	–	761,987	21.2	962,941	26.7	190.8	66%
ML-KEM-512	m4fspeed	125.0	430,087	11.9	–	–	426,644	11.9	464,116	12.9	161.7	77%
ML-KEM-512	m4fstack	125.0	430,468	12.0	–	–	429,188	11.9	467,040	13.0	161.9	77%
ML-KEM-768	clean	180.0	1,070,422	29.7	–	–	1,248,926	34.7	1,521,015	42.3	286.7	63%
ML-KEM-768	m4fspeed	180.0	701,045	19.5	–	–	716,417	19.9	765,580	21.3	240.6	75%
ML-KEM-768	m4fstack	180.0	702,992	19.5	–	–	722,778	20.1	772,746	21.5	241.1	75%
ML-KEM-1024	clean	245	1,667,847	46.3	–	–	1,879,878	52.2	2,218,146	61.6	405.2	60%
ML-KEM-1024	m4fspeed	245	1,108,155	30.8	–	–	1,117,974	31.1	1,181,828	32.8	339.7	72%
ML-KEM-1024	m4fstack	245	1,117,490	31.0	–	–	1,133,514	31.5	1,198,272	33.3	340.8	72%

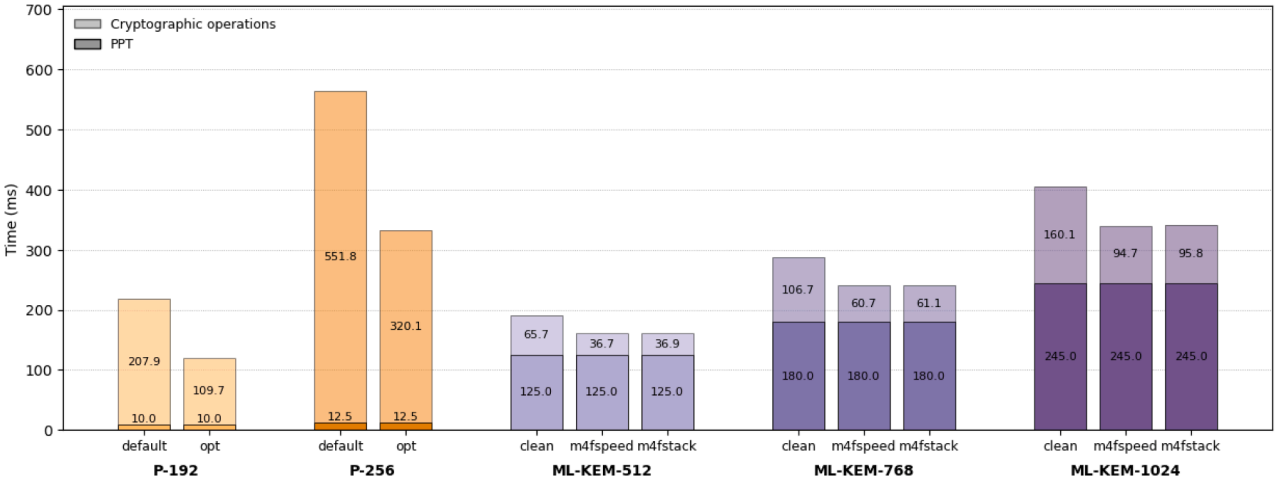

Fig. 6. Per-algorithm comparison of the time spent on PPT and operations execution during a BC key exchange.

portray the execution details under both classical and post-quantum settings. The ARM Cortex-M4 processor was chosen for testing due to its widespread adoption in Bluetooth modules across various applications, including home automation, medical and fitness devices, and industrial systems using both BC and BLE [15]. The algorithms were tested using the STM32F3 Microcontroller Unit (MCU), which features an ARM Cortex-M4 CPU, 40KB of RAM, and 256KB of flash memory [24]. This setup enabled a comprehensive performance analysis of the algorithms under constraints similar to those of Bluetooth chips, such as the BT122 BC and BLE controller from Silicon Labs, a common choice for several different Bluetooth applications [25]. This module utilizes the Texas Instruments CC2564C dual-mode controller [26]. In these legacy architectures, cryptographic operations like ECDH are typically executed as black-box functions within the controller's internal firmware. In this case, the firmware runs on an integrated ARM7TDMI core and utilizes fixed-function microcoded hardware accelerators. While this design is efficient for established standards like P-256, it lacks the flexibility to incorporate lattice-based PQC algorithms. Consequently, achieving quantum resistance in such systems necessitates offloading the pairing procedures from the fixed-function controller to the programmable ARM Cortex-M4 host or directly replacing them with ARM Cortex-M4 processors. By benchmarking on the proposed processor, this study captures the performance impact of this necessary architectural transition. While Bluetooth controllers with more RAM and flash memory exist, our focus was on highly constrained Bluetooth controllers supporting both BC and BLE, allowing for a complete analysis within the Bluetooth ecosystem.

The analysis employed two open-source libraries targeting the ARM Cortex-M4 family: *micro-ecc* for ECDH schemes and *pqm4* for PQC KEM primitives. The former provides side-channel-resistant elliptic-curve-based schemes for MCUs in C and inline assembly. Among the supported primitives are the P-192 and P-256 curves (*secp192r1* and *secp256r1*, respectively) used by the Bluetooth specification [27]. The latter, *pqm4*, is a well-known repository containing ARM Cortex-M4 implementations for several PQC algorithms [19], including ML-KEM and HQC. The experiments presented can be reproduced using the publicly-available GitHub repository for

Table 6
Experimental scenarios for statistical analysis.

Scenario	Operation	Sec. Level	Algorithms	Implementations
1	Keygen	Low	P-192 ML-KEM-512	Default Clean
2	Keygen	High	P-256 ML-KEM-1024	Optimized m4fspeed
3	Secret Deriv.	Low	P-192 ML-KEM-512	Default Clean
4	Secret Deriv.	High	P-256 ML-KEM-1024	Optimized m4fspeed

this project[1], given that the user has access to the same target board used during the experimentation process outlined in this work [24].

The selected MCU lacks the RAM capacity to execute HQC tests, since this algorithm requires up to 66,560 bytes in its low-security variant and 205,824 bytes in its high-security variant, while the STM32F3 only has 40KB of RAM. Consequently, HQC was excluded from the tests due to its impracticality given the resource limitations of the targeted Bluetooth controller family. As a result, implementing HQC in constrained environments can be a challenge, making ML-KEM the only standardized PQC KEM available for such applications. Nevertheless, researchers are actively exploring strategies to reduce peak RAM usage and stage memory (particularly for machine learning) between flash and RAM [28]. For instance, RAM-optimized deployment techniques can enable the execution of large models with minimal RAM usage by storing most state in non-volatile memory and carefully structuring execution.

Table 5 presents the average CPU cycles and corresponding milliseconds required to execute the LMP key exchange round-trip for each analyzed algorithm. The tests involved 100 runs per operation, from which average CPU cycles were calculated. A CPU frequency of 36.864MHz was set during testing, achieved by multiplying the HSE source frequency (7.37MHz) by 5 using PLL integer multipliers, including one wait state during execution. This configuration closely replicates the 40MHz CPU frequency found in common Bluetooth modules such as the BT122 Silicon Labs module mentioned earlier. The total time in milliseconds was derived using the formula $t_{ms} = \frac{cycles}{CPU\,frequency}$. Finally, the last column indicates the percentage of the total key exchange time attributed to the PPT process. The same compilation parameters were used for both the ECDH and PQC KEM tests, employing the *arm-none-eabi-gcc* compiler with the `-O3` and `-g3` flags, which are the default settings for speed tests in the *pqm4* framework [29]. Both repositories provided clean and optimized implementations, all of which were included in the tests. The ECDH tests compare a baseline generic C implementation against an optimized version with curve-specific modular reduction and ARM assembly for addition/subtraction operations, as well as single and multi-curve multiplication. The ML-KEM implementations included the *clean* implementation from PQClean [30], along with optimized implementations *m4fspeed* and *m4fstack*, incorporating ARM Cortex-M4F specific optimizations using assembly floating-point registers to improve speed and stack usage, respectively.

The results obtained during testing were analyzed to show their statistical significance with respect to the hypotheses proposed during experimentation. We focused on total handshake costs measured in clock cycles, accounting for ECDH as key generation plus two shared secret computations, and ML-KEM as key generation plus encapsulation and decapsulation. This reflects that under ECDH, as discussed in Section 4, both parties generate their key pairs simultaneously but compute shared secrets independently, whereas ML-KEM performs the three operations sequentially. Due to the high amount of data gathered, we focused on four specific scenarios to compare the statistical relevance of the results obtained. These are detailed in Table 6.

To rigorously evaluate the results in Table 5, we first performed Kolmogorov-Smirnov (KS) and Levene tests to characterize the underlying data distributions. As shown in Table 7, the ECDH scenarios exhibit a normal distribution ($p > .05$), whereas ML-KEM introduces a non-normal, bimodal distribution ($p < .001$). This behavior, visually depicted in Fig. 7, indicates that while the majority of cryptographic operations execute within a consistent cycle range, a minor portion of ML-KEM operations require additional cycles. This variance is attributed to rejection sampling, which occasionally necessitates additional iterations to complete the operation. Furthermore, the Levene test results ($p < 0.001$) indicate a significant lack of homogeneity in variance between the two groups. These findings strictly justify the use of the non-parametric Mann-Whitney U test, providing a robust measure of significance given the probabilistic nature of the gathered data. The resulting p-values ($p \approx 2.56 \times 10^{-34}$) confirm that the performance gap is not an artifact of system noise or measurement error. Beyond the significance of the p-value, a Cliff's Delta of $\delta = 1.0$ across all scenarios demonstrates the fundamental superiority of ML-KEM over ECDH. Even accounting for rejection sampling overhead, ML-KEM handshake operations are approximately 5× faster than those of ECDH, proving that the post-quantum transition offers a significant computational dividend for the ARM Cortex-M4 architecture.

[1] https://github.com/pablo-gf/pqbt-eval

Table 7
Statistical validation of the proposed scenarios: normality, variance, and significance analysis.

Scenario	KS ECDH (p)	KS ML-KEM (p)	Levene (p)	MW (p)	Cliff's δ	Speedup
1	0.982	1.47×10^{-19}	2.64×10^{-17}	2.56×10^{-34}	1.0	5.83×
2	0.858	5.26×10^{-15}	2.92×10^{-10}	2.56×10^{-34}	1.0	5.20×
3	0.760	1.16×10^{-7}	3.23×10^{-09}	2.56×10^{-34}	1.0	4.34×
4	0.457	3.5×10^{-8}	2.30×10^{-13}	2.56×10^{-34}	1.0	5.01×

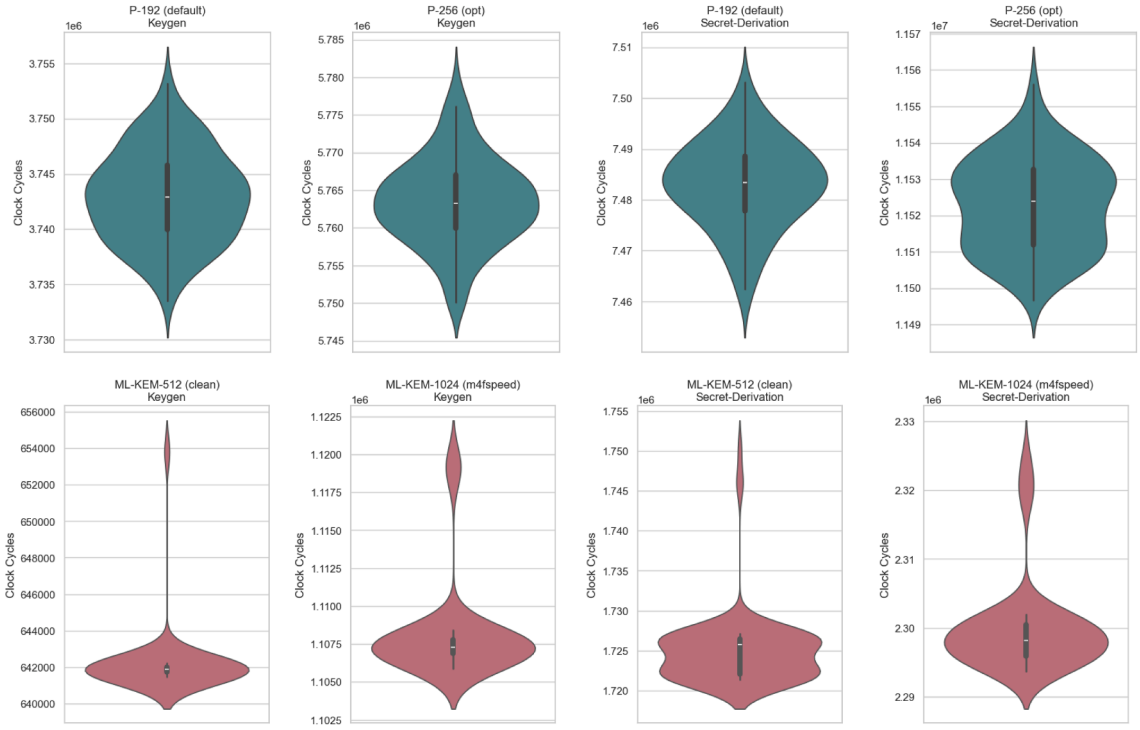

Fig. 7. CPU cycles distribution over the compared scenarios.

Overall, these results indicate that migrating the internal cryptography of BC to PQC would significantly increase the time for the PPT process, but not the overall key exchange time. This is primarily due to the time required for the ECDH *key generation* and *combine* functions. Comparing the lower-security variants, ML-KEM-512 notably outperforms P-192, reducing the time by 50 ms for the clean/default implementations. It also remains below the 100 ms for the optimized implementations, whereas the P-192 curve slightly exceeds the 200 ms and 100 ms thresholds for the default and optimized implementations, respectively. At a higher NIST security level, the ML-KEM-768 key exchange takes approximately 100 ms more execution time than the P-192 curve. For the high-security variants, the clean implementation of ML-KEM-1024 reduces the time required by 150 ms compared to the P-256 algorithm. However, their optimized implementations perform similarly, with P-256 performing marginally better by 8 ms. Fig. 6 illustrates that 60–75% of the time taken by the PQC algorithms to perform the key exchange is spent on PPT, a stark contrast to the ECDH variants, where it is always well below 10% of the total time. These results align with [31], which suggest that Kyber (the precursor of ML-KEM) is the most efficient lattice-based scheme for secure communications under constrained-resource environments. They also follow the trends depicted in [20], where Kyber clearly outperforms different set of binary ECDH curves. Besides, works like [32] and [33] also claim similar behavior under TLS, where Kyber and ML-KEM clearly outperform ECDH algorithms in terms of the total throughput of completed key exchanges at the protocol level.

6. Cross-protocol implications

The results obtained throughout this exploration provide critical insights into the PQC migration of other IOT protocols operating under resource-constrained environments. This transition is particularly vital as the industry shifts toward utilizing edge devices for AI and data-collection purposes. While traditional IOT devices function primarily as gateways for sharing data with remote servers, edge computing decentralizes this architecture by processing and storing data locally. By bringing computational resources to the information source, edge devices enhance both the latency and quality of data insights [34]. Whether an edge device collects data

Table 8
Comparison of widely-adopted wireless IoT protocols analyzing their use cases, key exchange, range, PPT fragmentation, and ARM Cortex-M4 availability in mainstream implementations.

Protocol	Use Cases	Key Exchange	Range	PPT Frag.	ARM Cortex-M4
BC [2]	High-fidelity Audio, Medical Telemetry, Wearables	ECDH (P-192/P-256)	10 – 150m	High	Common
Wi-Fi [35]	High-bandwidth Multimedia, Video Streaming, Enterprise WLAN	ECDH (WPA3-SAE)	30 – 100 m	Low	Sometimes
Z-Wave [36]	Residential Lighting, HVAC, Security Sensors	ECDH(S2 Framework)	30 – 100 m	High	Sometimes
Thread [37]	Low-latency Mesh Home Automation, IP-based IoT	ECDH (J-PAKE)	10 – 30 m	High	Common
Matter [38]	Interoperable Smart Home Ecosystems, Cross-vendor Device Control	ECDH (SPAKE2 +)	Depends onTransport	Depends on Transport	Common
WI-SUN [39]	Smart Utility Grids, Smart Cities, Mesh Sensing	ECDH (EAP-TLS)	200 m – 1 km	High	Common
ZigBee [40]	Smart Lighting, Industrial Monitoring	Pre-Shared Keys	10 – 100 m	Not Required	Common
LoRaWAN [41]	Agriculture, Long-range Asset Tracking	Pre-Shared Keys	2 – 15 km	Not Required	Common
NB-IoT [42,43]	Smart Metering, Deep-indoor Industrial Sensing	SIM Contains Keys	1 – 10 km	Not Required	Not MCU-dependant
LTE-A [42,43]	Edge Computing, Vehicle-to-Everything	SIM Contains Keys	1 – 5 km	Not Required	Not MCU-dependant

directly or is placed next to a sensing IOT node, the entire ecosystem must support quantum-secure cryptographic protocols to safeguard the transmission and handling of sensitive information.

A diverse array of wireless protocols exists to serve varying communication requirements. While our work primarily addresses short-range Bluetooth communication, numerous other standards offer distinct specifications that directly influence their security architectures. For this study, the most prominent protocols were selected based on their widespread adoption across industrial and consumer sectors. Table 8 summarizes these protocols, highlighting the technical characteristics most relevant to the variables examined in our PQC analysis.

As detailed in Table 8, protocols such as ZigBee, LoRaWAN, and cellular-based standards (NB-IoT, LTE-A) often rely on pre-shared keys or SIM-based credentials provisioned during manufacturing or through physical out-of-band exchange. Because these do not utilize traditional Diffie-Hellman-based handshakes for session establishment, they are inherently shielded from the "Harvest Now, Decrypt Later" threat targeting asymmetric key exchanges. By contrast, protocols that rely on ECDH, such as WIFI, Z-Wave, Thread, and WI-SUN, must follow a migration path similar to the one proposed for Bluetooth in this study. For these standards, the transition to ML-KEM is not merely a trivial update, but a fundamental shift in how the pairing phase handles increased cryptographic overhead. Protocols with a high fragmentation label, such as Thread or WI-SUN, will require a significant increase in packet volume to transmit cryptographic material during PPT. This increased airtime has a cascading effect:

- **RF Interference**: In long-range scenarios (e.g., WI-SUN's 1 km range), an increase in packet count significantly raises the probability of collision and packet loss, potentially stalling the pairing process.
- **Energy Constraints**: Since radio-frequency operations are typically the most power-consuming component of a constrained system, the increased reliance on the wireless medium for PPT will lead to a proportional decrease in battery life.

The computational analysis using the ARM Cortex-M4 in this study is highly representative of the mid-tier IOT market, where this processor is widely used in different IOT deployments, as suggested by Table 8. For high-performance modules, the ARM Cortex-M4 often acts as a controller, offloading heavy mathematical operations to dedicated hardware cryptographic sub-engines. However, devices utilizing smaller processors or different architectures will require a distinct migration approach. Many current systems may be forced to shift toward updated system-on-chip modules with hardware acceleration for lattice-based cryptography, or adopt a rearranged execution scheme similar to the one demonstrated for Bluetooth in this exploration.

Though the results of this study establish a critical baseline, the diversity of the IOT landscape requires further research into the specific structural, packetization and cryptographic design of each individual protocol. As summarized in Table 8, many of these standards utilize distinct authentication mechanisms that will interact uniquely with PQC primitives. Matter, for example, operates by encapsulating security at the application layer, typically running over a transport layer such as Thread, Wi-Fi, or Bluetooth. Therefore, it executes a double handshake: one to secure the underlying transport and a subsequent exchange to establish the Matter fabric. If both layers migrate to PQC, the PPT effort is effectively doubled. Furthermore, protocols that utilize Password-Authenticated Key Exchanges (PAKEs), such as Wi-Fi, Thread, and Matter present a unique migration challenge. These protocols do not merely perform a key exchange; they bind the process to a user-provided password to prevent offline dictionary attacks. Transitioning these to a quantum-safe equivalent requires a thorough analysis of post-quantum PAKE algorithms and their feasibility under the provided scenarios. Consequently, while this exploration provides the foundational groundwork, future work must conduct a deep-dive into the internal functioning of each individual scheme to ensure a seamless and reliable PQC transition of IOT systems.

7. Conclusion

This study offers the first comprehensive analysis of post-quantum migration in BC pairing, combining protocol-level modeling, over-the-air timing analysis, and practical implementation benchmarks. Our findings yield three key takeaways. First, standardized PQC schemes are computationally feasible on constrained Bluetooth controllers. The experiments carried out in this study show that ML-KEM implementations complete key generation and encapsulation/decapsulation operations within practical timing budgets, often matching or exceeding the efficiency of classical ECDH schemes. This demonstrates that CPU limitations are not the primary obstacle to PQC adoption in BC. Second, the dominant bottleneck in PQC migration arises from communication overhead. The larger public keys and ciphertexts inherent to PQC substantially increase DM1 packet usage, over-the-air transmission times, which may increase the need for ARQ retransmissions. The time spent on Public Parameter Transmission (PPT) for PQC algorithms accounts for 60-75% of total key exchange latency, compared to less than 10% for classical ECDH, highlighting that migration challenges are primarily a wireless protocol issue rather than a computational one. Third, not all standardized PQC KEMs are suitable for constrained BC devices. While ML-KEM can be deployed effectively, HQC exceeds RAM capacities of resource-constrained Bluetooth controllers and imposes prohibitive transmission delays, rendering it impractical without architectural modifications. This distinction has significant implications for Bluetooth implementers and standards organizations considering post-quantum migration strategies. As a result, this work demonstrates that the safe integration of post-quantum key exchange in BC is achievable with carefully selected schemes. However, successful adoption requires careful consideration of over-the-air packetization, energy consumption, and protocol reliability; issues native not only to Bluetooth, but to general wireless communication protocols. This analysis lays groundwork for future research on less constrained Bluetooth controllers, which may support other KEM-based schemes, as well as hybrid, staged, or application-layer PQC deployment for wireless systems.

CRediT authorship contribution statement

Pablo Gutierrez-Felix: Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Marc Manzano:** Writing – review & editing, Validation, Supervision, Project administration, Funding acquisition; **Javier Lopez:** Writing – review & editing, Validation, Supervision, Project administration, Funding acquisition.

Data availability

The data that supports the findings of this study is openly available at https://github.com/pablo-gf/pqbt-eval.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] B. SIG, Bluetooth market update 2025, 2025, Accessed: 2026, https://www.bluetooth.com/2025-market-update/.
[2] Bluetooth Core Specification, Bluetooth SIG, 2025.
[3] P.W. Shor, Algorithms for quantum computation: discrete logarithms and factoring, in: Proceedings 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 124–134. https://doi.org/10.1109/SFCS.1994.365700
[4] National Institute of Standards and Technology, Module-lattice-based key-encapsulation mechanism (ML-KEM) standard, Federal Information Processing Standards Publication FIPS 203, U.S. Department of Commerce, Washington, D.C., 2024. NIST FIPS 203, https://doi.org/10.6028/NIST.FIPS.203
[5] G. Alagic, M. Bros, P. Ciadoux, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, D. Moody, R. Peralta, R. Perlner, A. Robinson, H. Silberg, D. Smith-Tone, N. Waller, Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process, NIST Internal Report (IR) NIST IR 8545, National Institute of Standards and Technology, Gaithersburg, MD, 2025. NIST IR 8545, https://doi.org/10.6028/NIST.IR.8545
[6] Y.-J. Yang, K.-b. Jang, G.-j. Song, H.-J. Kim, Y.-J. Oh, H.-J. Seo, Proposal of bluetooth model with post-Quantum cryptography, in: Annual Conference of KIPS, 2021, pp. 236–239. https://doi.org/10.3745/PKIPS.Y2021M11A.236
[7] J. Bozhko, Y. Hanna, R. Harrilal-Parchment, S. Tonyali, K. Akkaya, Performance evaluation of quantum-resistant TLS for consumer IoT devices, in: 2023} IEEE 20th Consumer Communications & Networking Conference (CCNC), 2023, pp. 230–235. https://doi.org/10.1109/CCNC51644.2023.10060762

[8] Y. Hanna, J. Bozhko, S. Tonyali, R. Harrilal-Parchment, M. Cebe, K. Akkaya, A comprehensive and realistic performance evaluation of post-quantum security for consumer IoT devices, Internet Things 33 (2025) 101650. https://www.sciencedirect.com/science/article/pii/S2542660525001647. https://doi.org/10.1016/j.iot.2025.101650

[9] T. Liu, G. Ramachandran, R. Jurdak, Towards quantum resilient IoT: a backward-compatible approach to secure BLE key exchange against quantum threats, in: 2024 IEEE/ACM Ninth International Conference on Internet-of-Things Design and Implementation (IoTDI), 2024, pp. 170–180. https://doi.org/10.1109/IoTDI61053.2024.00019

[10] V. Zubkov, T. Sacchetti, D. Antonioli, M. Strohmeier, Bluetooth security testing with BlueToolkit: a large-scale automotive case study, in: Proceedings of the 19th USENIX Conference on Offensive Technologies, WOOT '25, USENIX Association, USA, 2025.

[11] J. Padgette, J. Bahr, M. Batra, R. Smithbey, L. Chen, K. Scarfone, Guide to bluetooth security, 2022, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=934038. https://doi.org/10.6028/NIST.SP.800-121r2-upd1

[12] E. Fujisaki, T. Okamoto, Secure integration of asymmetric and symmetric encryption schemes, J. Cryptol. 26 (1) (2013) 80–101. https://doi.org/10.1007/s00145-011-9114-1

[13] J.B. Almeida, S.A. Olmos, M. Barbosa, G. Barthe, F. Dupressoir, B. Grégoire, V. Laporte, J.-C. Léchenet, C. Low, T. Oliveira, H. Pacheco, M. Quaresma, P. Schwabe, P.-Y. Strub, Formally verifying kyber episode v: machine-checked IND-CCA security and correctness of ML-KEM in EasyCrypt, 2024, (Cryptology ePrint Archive, Paper 2024/843). https://eprint.iacr.org/2024/843.

[14] H.S. Team, Hamming Quasi-cyclic (HQC) KEM specification, 2025, https://pqc-hqc.org/doc/hqc_specifications_2025_08_22.pdf.

[15] DigiKey, The industry's broadest IoT bluetooth® module portfolio, 2025, Accessed: 2025-1-16, https://www.digikey.com/en/product-highlight/c/cypress/iot-bluetooth-module-portfolio.

[16] J. Howe, T. Oder, M. Krausz, T. Güneysu, et al., Standard lattice-based key encapsulation on embedded devices, IACR Trans. Cryptograph. Hardware Embedded Syst. 2018 (3) (2018) 372-393. https://tches.iacr.org/index.php/TCHES/article/view/7279. https://doi.org/10.13154/tches.v2018.i3.372-393

[17] L. Botros, M.J. Kannwischer, P. Schwabe, Memory-efficient high-speed implementation of kyber on cortex-M4, in: J. Buchmann, A. Nitaj, T. Rachidi (Eds.), Progress in Cryptology – AFRICACRYPT 2019, Springer International Publishing, Cham, 2019, pp. 209–228.

[18] M.J. Kannwischer, J. Rijneveld, P. Schwabe, K. Stoffelen, pqm4: testing and benchmarking NIST PQC on ARM Cortex-M4, 2019, (Cryptology ePrint Archive, Paper 2019/844). https://eprint.iacr.org/2019/844.

[19] Post-quantum crypto library for the ARM Cortex-M4, 2025, Accessed: 2025-1-16, https://github.com/mupq/pqm4.

[20] M.A. Mighri, A. Benfarah, A. Meddeb, Performance evaluation and benchmarking of PQC CRYSTALS-Kyber on embedded devices, in: 2024 IEEE/ACS 21st International Conference on Computer Systems and Applications (AICCSA), 2024, pp. 1–7. https://doi.org/10.1109/AICCSA63423.2024.10912602

[21] M.A. Khan, F. Noor, S. Javaid, J. Żywiołek, Implementation and performance of post-quantum cryptography for resource constrained consumer electronics, Discover Internet Things 5 (1) (2025) 139. https://doi.org/10.1007/s43926-025-00238-x

[22] Post-quantum kyber benchmarks (ARM Cortex-M4), 2024, Accessed: 2025-1-16, https://www.wolfssl.com/post-quantum-kyber-benchmarks-arm-cortex-m4/.

[23] C. Bala Kumar, P.J. Kline, T.J. Thompson, in: C. Bala Kumar, P.J. Kline, T.J. Thompson (Eds.), Bluetooth Application Programming with the Java APIs, Morgan Kaufmann, Burlington, 2004, pp. 1–22. https://www.sciencedirect.com/science/chapter/monograph/pii/B9781558609341500044?via%3Dihub. https://doi.org/10.1016/B978-155860934-1/50004-4

[24] L. NewAE Technology, ChipWhisperer CW308T: STM32F Target, 2023, Accessed: 2026-02-02, https://chipwhisperer.readthedocs.io/en/latest/chipwhisperer-target-cw308t/CW308T_STM32F/README.html.

[25] S. Labs, BT122 data sheet, 2021, Accessed: 2026-02-02, https://www.silabs.com/documents/public/data-sheets/bt122-datasheet.pdf.

[26] Texas Instruments, CC2564C dual-mode bluetooth controller datasheet, 2020. Revision B, https://www.ti.com/lit/ds/symlink/cc2564c.pdf.

[27] K. MacKay, ECDH and ECDSA for 8-bit, 32-bit, and 64-bit processors, 2022, Accessed: 2025-1-16, https://github.com/kmackay/micro-ecc.

[28] B. Sudharsan, P. Yadav, J.G. Breslin, M. Intizar Ali, An SRAM optimized approach for constant memory consumption and ultra-fast execution of ML classifiers on tinyML hardware, in: 2021 IEEE International Conference on Services Computing (SCC), 2021, pp. 319–328. https://doi.org/10.1109/SCC53864.2021.00045

[29] ARM, ARM GNU toolchain version 15.2.Rel1, 2025, Accessed: 2025-1-16, https://developer.arm.com/downloads/-/arm-gnu-toolchain-downloads.

[30] PQClean, Clean, portable, tested implementations of post-quantum cryptography, 2025, Accessed: 2025-1-16, https://github.com/PQClean/PQClean.

[31] A.K. Kwala, S. Kant, A. Mishra, Comparative analysis of lattice-based cryptographic schemes for secure IoT communications, Discover Internet Things 4 (1) (2024) 13. https://doi.org/10.1007/s43926-024-00069-2

[32] J.A. Montenegro, R. Rios, J. Lopez-Cerezo, A performance evaluation framework for post-quantum TLS, Future Gener. Comput. Syst. 175 (2026) 108062. https://www.sciencedirect.com/science/article/pii/S0167739X25003577. https://doi.org/10.1016/j.future.2025.108062

[33] B. Dong, Q. Wang, EPQUIC: Efficient post-quantum cryptography for QUIC-Enabled secure communication, in: Proceedings of the Great Lakes Symposium on VLSI 2025, GLSVLSI '25, Association for Computing Machinery, New York, NY, USA, 2025, p. 141-146. https://doi.org/10.1145/3716368.3735199

[34] L. Kong, J. Tan, J. Huang, G. Chen, S. Wang, X. Jin, P. Zeng, M. Khan, S.K. Das, Edge-computing-driven internet of things: a survey, ACM Comput. Surv. 55 (8) (2022). https://doi.org/10.1145/3555308

[35] IEEE Standard for information technology–Telecommunications and information exchange between systems local and metropolitan area networks–Specific requirements part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications, IEEE Std 802.11–2024 (Revision of IEEE Std 802.11–2020) (2025) 1–5956. https://doi.org/10.1109/IEEESTD.2025.10979691

[36] Z-Wave Alliance, Z-wave specification, Technical Report, Z-Wave Alliance, 2025. RElease 2025B, https://z-wavealliance.org/development-resources-overview/specification-for-developers/.

[37] Thread Group, Thread specification version 1.4.0, Technical Report, Thread Group, 2024. https://threadgroup.org/ThreadSpec.

[38] Connectivity Standards Alliance, Matter specification version 1.0, Technical Report, Connectivity Standards Alliance, 2022. https://csa-iot.org/wp-content/uploads/2022/11/22-27349-001_Matter-1.0-Core-Specification.pdf.

[39] Wi-SUN Alliance, Wi-SUN field area Network (FAN) specification, Technical Report, Wi-SUN Alliance, 2019. https://wi-sun.org/wp-content/uploads/Wi-SUN-FAN-Tech-Overview-r0.pdf.

[40] ZigBee Alliance, ZigBee specification, Technical Report 05-3474-23, ZigBee Alliance, 2015. https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf.

[41] LoRa Alliance, LoRaWAN specification version 1.1, Technical Report, LoRa Alliance, 2017. https://resources.lora-alliance.org/technical-specifications/lorawan-specification-v1-1.

[42] 3rd Generation Partnership Project (3GPP), LTE; evolved universal terrestrial radio access (E-UTRA), radio resource control (rrc); protocol specification, Technical Report TS 36.331, 3GPP, 2016. https://www.3gpp.org/dynareport?code=36-series.htm.

[43] 3rd Generation Partnership Project (3GPP), 3GPP system architecture evolution (SAE); security architecture, Technical Report TS 33.401, 3GPP, 2019. https://www.3gpp.org/dynareport?code=36-series.htm.