A Prototype for evaluating Post-Quantum Cryptography on resource-constrained Hardware with real-world Smart City Sensor Data

Jan Seedorf [1], Darshana Rawal[1], Jonas Möwes[1], Omar Haj Abdulaziz[1], Ayham Alhasan[1], Thunyathep Santhanavanich[1]
[1] HFT Stuttgart, Schellingstraße 24, 70174 Stuttgart, Germany
jan.seedorf@hft-stuttgart.de, darshana.rawal@hft-stuttgart.de, 12mojo1bif@hft-stuttgart.de, 12haom1bif@hft-stuttgart.de, 22alay1bif@hft-stuttgart.de, thunyathep.santhanavanich@hft-stuttgart.de

Keywords: Smart City sensor data, Post-Quantum Cryptography

Abstract

As the threat of quantum computing to classical cryptography grows, the transition to post-quantum cryptographic (PQC) systems becomes essential—particularly for smart city infrastructures that rely heavily on secure, real-time sensor data. This paper investigates the performance of PQC algorithms currently in the final stages of standardization by the U.S. National Institute of Standards and Technology (NIST), focusing on their deployment in resource-constrained Internet of Things (IoT) devices. Leveraging real-world smart city sensor datasets, we develop a research prototype that simulates a realistic urban sensing scenario, using Raspberry Pi 3 and Pi Zero 2 W devices to cryptographically secure and transmit data to a central server. Experimental results demonstrate that the Kyber family is the most efficient for key encapsulation tasks, while Dilithium and Falcon offer strong performance for digital signatures. In contrast, PQC algorithms such as McEliece, HQC, and Rainbow exhibit substantial computational overhead—especially at higher security levels—limiting their suitability for time-sensitive or low-power environments. Our findings highlight the practical implications of PQC adoption in smart cities and provide evidence-based guidance for selecting efficient quantum-safe algorithms for real-world urban sensor networks.

1. Introduction and Motivation

In the context of smart cities, sensor data encryption and integrity protection are of importance for several reasons. Firstly, the vast array of sensors deployed throughout a smart city collect a diverse range of data types—from traffic patterns and environmental conditions to energy consumption and public infrastructure usage. This data can be sensitive and critical, making it a prime target for malicious actors. Unauthorized access to raw sensor data could lead to privacy breaches, such as revealing individual movements or behaviors, underscoring the importance of robust encryption to maintain confidentiality.

Moreover, ensuring data integrity is crucial for smart city safety. Spoofing or tampering with sensor data can disrupt systems, causing traffic issues or poor environmental responses. For example, false data injected into traffic management systems could cause congestion or accidents, while inaccuracies in environmental sensors could hinder effective responses to pollution or natural disasters. Cryptographic integrity mechanisms ensure data authenticity, maintaining trust in automated decision-making processes. In addition, the interconnected nature of smart city systems means that data compromise in one area can have widespread effects. Encrypting data and using integrity checks prevent unauthorized data alteration, safeguarding city infrastructure against routine and sophisticated attacks.

Quantum computing threatens current cryptographic methods, with the capability to break widely used algorithms like RSA and ECC, which secure today's digital infrastructure (Chen et al., 2025). As these threats grow, exploring Post-Quantum Cryptography (PQC), i.e. cryptographic algorithms designed to be secure against the potential threats posed by quantum computers, becomes urgent to secure smart city data and services. Transitioning to PQC ensures that sensitive information between city systems remains protected and trustworthy in the future, even against powerful quantum computers.

Implementing Post-Quantum Cryptography in smart city systems is vital to counter future quantum threats while also ensuring compatibility with existing technologies, especially resource-constrained IoT devices. Developing efficient PQC solutions helps maintain robust security and performance, positioning cities to be secure and adaptable for next-generation urban challenges.

In this work, we investigate the performance of currently unbroken post-quantum cryptographic (PQC) algorithms—particularly those in the final stages of NIST standardization—when applied to real-world smart city sensor data on resource-constrained IoT devices. We develop a research prototype simulating a realistic smart city scenario in which low-power sensors use PQC to secure data before transmitting it to a server. Experimental results compare the computational performance of several PQC key encapsulation and digital signature schemes, highlighting their suitability for constrained environments.

2. Post-Quantum Cryptography (PQC)

2.1 The Need for Post-Quantum Cryptography

Traditional public-key cryptosystems—such as RSA, DSA, and elliptic-curve cryptography (ECC)—rely on mathematical problems like integer factorization and discrete logarithms. These are computationally hard for classical computers, but vulnerable to quantum attacks:

- Shor's Algorithm (Shor, 1994) can solve both integer factorization and discrete logs in polynomial time on a quantum computer.
- A large enough quantum computer would break most of today's digital security, including secure web traffic, encrypted emails, and software updates.

Even though large-scale quantum computers are not yet a reality, "store now, decrypt later" attacks pose immediate risks: adversaries may store encrypted data now to decrypt it once quantum capabilities emerge.

2.2 State-of-the-Art of Post-Quantum Cryptography (PQC)

Post-Quantum Cryptography (PQC) is advancing rapidly to replace classical public-key cryptosystems vulnerable to quantum attacks. The state-of-the-art of post-quantum cryptographic (PQC) algorithms is centered around developing encryption and digital signature schemes resilient to the potential capabilities of quantum computers. Post-Quantum Cryptography (PQC) efforts, including NIST's standardization process, are focusing exclusively on a) KEMs (Key Encapsulation Mechanisms) and b) digital signatures because these two primitives are the core building blocks of almost all modern public-key cryptographic protocols:

- KEMs are a structured way to establish a shared secret between two parties over an insecure channel. This is the foundation of most secure communication protocols — for example:
    - In TLS (used for HTTPS), clients and servers negotiate a shared key to encrypt their session.
    - In VPNs (IPsec, WireGuard) and SSH, secure key exchange is vital to prevent eavesdropping.

  Classical key exchange methods like RSA and Elliptic-Curve Diffie-Hellman (ECDH) will be broken by quantum algorithms (e.g., Shor's algorithm). Thus, PQC research focuses on KEMs as a drop-in replacement to achieve quantum-resistant secure key exchange.
- Digital signatures ensure that data comes from a trusted source and has not been altered:
    - Digital signatures are used in software updates, electronic documents, secure boot, and digital certificates (e.g., X.509 in TLS).
    - Current schemes like RSA and ECDSA are quantum-vulnerable.

  Signatures are especially critical because:
    - Many signed documents (legal, medical, financial) must remain verifiable for decades.
    - They are the backbone of public key infrastructures (PKI) used to establish trust online.

  Thus, PQC research focuses on quantum-safe signature schemes like Dilithium, FALCON, and SPHINCS+ to secure long-term integrity and authenticity.

The algorithms in consideration encompass various mathematical foundations, including lattice-based cryptography, hash-based cryptography, code-based cryptography, multivariate polynomial cryptography, and others. Each of these approaches offers unique strengths in terms of security, efficiency, and resource requirements, which are crucial for diverse applications ranging from small IoT devices to large-scale data centers.

The current technical state of the art is led by lattice-based cryptography, particularly the NIST-standardized CRYSTALS-Kyber (Bos et al., 2018) for key encapsulation and CRYSTALS-Dilithium (Ducas et al., 2018) for digital signatures. These schemes are based on the Learning With Errors (LWE) and Short Integer Solution (SIS) problems, which are believed to be hard even for quantum computers. They offer strong performance, relatively small key and ciphertext sizes, and efficient implementation across platforms. FALCON (Fouque et al., 2018), another lattice-based signature scheme, offers more compact signatures but at the cost of more complex and delicate implementation due to its reliance on floating-point arithmetic.

Other PQC families play specialized roles. Code-based cryptography, like Classic McEliece (Bernstein et al., 2017), provides excellent security and decryption speed but suffers from very large public keys. Hash-based schemes, such as SPHINCS+ (Bernstein et al., 2019), are extremely conservative and rely only on hash functions for security, making them ideal for high-assurance use cases like long-term digital signatures despite their large signatures and slower performance. Multivariate and isogeny-based schemes once showed promise, but the most prominent candidates, e.g., Rainbow (Ding et al., 2017), have been broken or are no longer considered viable. As standardization progresses and protocols like TLS, SSH, and IPsec begin integrating PQC algorithms, the focus is shifting to implementation security, hardware acceleration, and hybrid deployment strategies to ease the transition.

2.3 NIST PQC Standardization

Recognizing the significant threat that quantum computing poses to classical public-key cryptographic systems, the U.S. National Institute of Standards and Technology (NIST) launched a Post-Quantum Cryptography (PQC) standardization initiative in 2016. The goal of this effort is to identify and establish cryptographic algorithms that can resist quantum attacks while remaining practical, efficient, and suitable for widespread deployment (Chen et al., 2025).

This multi-phase process involves open global collaboration, where researchers submit candidate algorithms for rigorous evaluation. These candidates undergo multiple rounds of public review, cryptanalysis, and performance testing to ensure both theoretical robustness and practical applicability across diverse systems and use cases.

As outlined previously (see section 2.2), NIST's standardization focuses primarily on Key Encapsulation Mechanisms (KEMs) and digital signature schemes, as these are the most widely used public-key primitives and the most directly threatened by quantum algorithms like Shor's. By prioritizing these two categories, NIST aims to secure the foundational components of the internet and critical infrastructure, enabling a smooth and secure transition to a post-quantum future.

NIST's standardized algorithms are expected to become the benchmarks for securing digital communications against quantum threats, ensuring the integrity, confidentiality, and authenticity of information in a post-quantum world. The ongoing process is closely watched by academia, industry, and governmental institutions as they prepare to transition to these new cryptographic standards.

In 2023, NIST selected a set of algorithms to be standardized based on their strong security, performance, and implementation properties (Chen et al., 2025). These were formally published as Federal Information Processing Standards (FIPS) in 2024. The standards include:

- FIPS 203, which specifies ML-KEM, a key encapsulation mechanism based on the lattice-based algorithm CRYSTALS-Kyber (NIST, 2024a).
- FIPS 204, defining ML-DSA, a lattice-based digital signature scheme based on CRYSTALS-Dilithium (NIST, 2024b).

- FIPS 205, which describes SLH-DSA, a stateless hash-based digital signature scheme built on SPHINCS+ (NIST, 2024c).

These algorithms are now the recommended options for federal agencies and are expected to serve as foundational primitives for public-key encryption and digital signatures in the post-quantum era.

Alongside the finalized standards, NIST continues to evaluate other promising candidates. One of the most prominent is FALCON, as it is a compact and efficient lattice-based digital signature scheme (see section 2.2). While not yet standardized, FALCON is considered a strong alternative for use cases that require smaller signature and key sizes than Dilithium. NIST has indicated that a separate standard for FALCON is under development and may be finalized in the near future.

NIST is also conducting a fourth round of evaluation, focusing on enhancing the diversity of algorithm families. This round includes ongoing analysis of Classic McEliece, a code-based KEM known for its strong security and resistance to quantum attacks, albeit with large public key sizes that make it less practical for many applications. The fourth round also revisits select multivariate and lattice-based candidates to assess their long-term viability and implementation trade-offs.

Looking ahead, further standards are expected to be finalized in the near future, including those based on FALCON and potentially Classic McEliece. In parallel, efforts are underway within other standards organizations such as the IETF and ISO to integrate these algorithms into protocols like TLS, X.509 certificates, SSH, and VPN systems.

3. Objectives and Contribution

The overall objective of our work is to investigate the performance of PQC algorithms that are unbroken at present[1] in the context of smart city sensors and smart city data. Specifically, our goal is to study the performance of PQC algorithms currently in the final stages of NIST-standardisation on resource-constrained IoT hardware when cryptographically protecting actual real-world smart city sensor data.

Our contributions are the following:

- We introduce a research prototype which simulates a realistic IoT scenario in which resource-contrained sensors protect real-world smart city sensor data using Post-Quantum Cryptography and then transmit this data to a server.
- We present experimental results obtained with this research prototype that compare the average computation times for several PQC key-establishment algorithms and several PQC digital signature algorithms (including the ones currently being standardised)

4. Prototype Design and Experimental Setup

4.1 Protoytype Design and Testbed Setup

We implemented a “PQC-Smart-City-Data” benchmarking system based on i) the PQClean library (Kannwischer et al., 2022), a C-based library that provides implementations of all PQC algorithms currently considered by NIST, and ii) on a Rust-based library that provides an additional implementation of the Kyber Post-Quantum key encapsulation mechanism (Argyle-Software, 2025).

Our prototype runs a simple Python Flask server (Pallets, 2024) and makes requests to it using Python’s requests library. To execute the PQClean library’s C-based algorithms from Python, the ctypes module was used. The C source files needed to be compiled beforehand. To automate this process, Python scripts were written to generate a Makefile for each algorithm and then execute make with it.

The Rust-based implementation (Argyle-Software, 2025) follows a different interface. To accommodate this, a wrapper was created in the Rust directory. A shell script was also developed to compile this wrapper and generate the necessary binaries.

Our testing system consists of two Raspberry Pi devices (Pi3 and Pi Zero 2W), which each can act as either a client or as a server in a particular experiment. Although Raspberry Pi devices are more capable than basic microcontrollers, they are increasingly used in real-world sensor networks, particularly in smart city and edge computing applications. Their low cost, small size, and relatively low power consumption make them suitable for roles such as local gateways, edge processors, or even as standalone smart sensing units. In many urban deployments, sensor nodes are expected to perform not just data collection, but also on-device processing, encryption, and secure communication—tasks that align well with the capabilities of Raspberry Pi hardware.

Evaluating post-quantum cryptographic (PQC) algorithms on Raspberry Pi platforms is therefore a realistic and meaningful approach. Given the added computational demands of PQC, it's essential to assess performance on devices that represent the upper bound of what might be deployed in edge environments. Raspberry Pi boards provide a practical balance between resource constraints and computational power, making them an appropriate and informative testbed for understanding the viability of PQC in next-generation sensor networks.

In our prototype, the client requests a public key from the server and uses this public key for the Key Encapsulation Mechanism (KEM) to wrap an AES key generated by the client, i.e. KEM is performed at the client side. The client encrypts the payload, i.e. smart city data, with the generated AES key and digitally signs a hash of the encrypted data, i.e. digital signatures are also performed on the client side. The server has the role of signature verification and key decapsulation in our prototype.

4.2 Communication Flow and Results Measured

In each experiment, the client first generates a symmetric AES key and encapsulates it with a PQC Key Encapsulation Mechanism (KEM); it then encrypts the data with a symmetric AES key and digitally signs the hash of the encrypted data with a PQC signature algorithm. It then sends the encrypted data, the digital signature, and the encapsulated AES key to the server. The server verifies the digital signature, decapsulates the AES key, and decrypts the data. The time needed for each of these individual operations is measured individually at the client and at the server.

The communication steps are hence as follows:

[1] Meaning PQC algorithms that are currently considered to be “quantum-computationally secure", i.e. algorithms that according to today’s knowledge cannot be feasibly broken within a reasonable time frame even by quantum computers.

1. The client requests a public key for a key encapsulation mechanism (KEM).
2. Using this public key, the client generates an AES key and encapsulates it.
3. The AES key is then used to encrypt the data.
4. The encrypted data is hashed and signed using a post-quantum signature mechanism.
5. Each process is timed, and the results are saved to a CSV file.
6. The client sends the following data to the server:
   a. Encrypted data
   b. Digital Signature
   c. Public key corresponding to the digital signature performed by client
   d. Algorithms used for digital signature and KEM
   e. Encapsulated AES key
   f. IV (Initialization Vector)
7. The server:
   a. Hashes the encrypted data and verifies the signature.
   b. Decapsulates the AES key and decrypts the data.
   c. Measures execution time and logs it in a CSV file.

In particular, the server logs in each experiment the following experiment paramters and measurements:

- KEM Algorithm: The key encapsulation mechanism used.
- Signature Algorithm: The digital signature scheme applied.
- Server Hash Time: The duration taken to hash the encrypted data.
- Verify Time: The time required to verify the signature.
- Decapsulation Time: The time needed to decapsulate the AES key.
- Decrypt Time: The time taken to decrypt the data.
- Device Name: The hardware device used for testing.

An example for data being measured at the server in an experiment is the following:

```
KEM Algorithm,Signature Algorithm,Server Hash Time,Verify Time,Decapsulation
Time,Decrypt Time,Device Name
mceliece348864,rainbowIclassic,1294269,10848104,100581416,12927995,raspi3
mceliece460896,dilithium3,1313435,1491403,208070206,2117392,raspi3
mceliece460896,falcon1024,2474058,1258018,251924750,2177079,raspi3
```

The client logs in each experiment the following experiment paramters and measurements:

- KEM Algorithm: The key encapsulation algorithm used.
- Signature Algorithm: The digital signature scheme applied.
- Client Device: The name of the testing device.
- Encapsulation Time: Time taken to encapsulate the AES key.
- Encryption Time: Time required to encrypt the data.
- Client Hash Time: Time taken to hash the encrypted data.
- Sign Time: The time needed to generate the digital signature.
- Data Size: The size (in bytes) of the encrypted data.

An example for data being measured at the client in an experiment is the following:

```
KEM Algorithm,Signature Algorithm,Client Device,Encapsulation Time,Encryption
Time,Client Hash Time,Sign Time,Data Size
mceliece460896,rainbowVclassic,raspi3,1925776,21083535,2156661,234415071,58943
mceliece6688128,dilithium3,raspi3,2922648,3377282,2059942,10801897,58943
mceliece6688128,falcon512,raspi3,1730672,2041088,1266871,27710338,58943
```

4.3 Smart City Data Sets

The actual payload data being sent cryptographically protected from client to server consists of 100 real-world temperature measurements from actual IoT sensors (located in Fukuoka, Japan), encoded in JSON format[2]. The following JSON snippet shows one example of these 100 measurements:

```
{"@iot.id":15648,"phenomenonTime":"2022-08-
27T05:55:00.000Z","result":39.0,"resultTime":"2022-08-
27T05:55:00.000Z",
"@iot.selfLink":"https://ogcapi.hft-
stuttgart.de/sta/udigit4icity/v1.1/Observations(15648)"
,"Datastream@iot.navigationLink":"https://ogcapi.hft-
stuttgart.de/sta/udigit4icity/v1.1/Observations(15648)/
Datastream","FeatureOfInterest@iot.navigationLink":"htt
ps://ogcapi.hft-
stuttgart.de/sta/udigit4icity/v1.1/Observations(15648)/
FeatureOfInterest","MultiDatastream@iot.navigationLink"
:"https://ogcapi.hft-
stuttgart.de/sta/udigit4icity/v1.1/Observations(15648)/
MultiDatastream"}
```

This record represents a sensor observation from a smart city data platform, captured through the OGC SensorThings API. The key details are:

- Observation ID: 15648
- Measurement Value: 39.0 (temperature)
- Time of Measurement: 2022-08-27T05:55:00.000Z (both as phenomenonTime and resultTime)
- API Source: OGC SensorThings API – HFT Stuttgart
- Data Links: Associated Datastream, FeatureOfInterest, and MultiDatastream resources are linked, suggesting that this observation is part of a structured sensor data stream (e.g., a time series).

This type of dataset is typical in smart city deployments and could represent readings from environmental, mobility, infrastructure, or energy sensors. It is important to note that we encrypt and protect the entire JSON object—not just the measurement value—to ensure the integrity and authenticity of both the data and its contextual metadata. In smart city and IoT contexts, sensor data is typically transmitted in structured formats like JSON, which include not only the measurement (result) but also metadata such as timestamp, location, device ID, and links to data streams. While the actual sensor reading (e.g., temperature, air quality) is central to the application, the surrounding metadata is equally critical to ensuring data integrity, context, and trustworthiness.

Protecting only the measurement value leaves the rest of the message—such as phenomenonTime, FeatureOfInterest, and Datastream—vulnerable to tampering. An attacker could alter the timestamp to make old data appear current, change the location to misrepresent where the measurement was taken, or swap links to mislead downstream systems. This could have serious implications in smart city applications, from faulty traffic management to false pollution alerts.

By applying digital signatures or encryption to the entire JSON structure, we ensure that the full semantic meaning of the observation is preserved and verifiable. This guarantees that the data's origin, context, and value are all protected as a single,

[2] The dataset being used for our experiments is available at: https://ogcapi.hft-stuttgart.de/sta/udigit4icity/v1.1/Observations

inseparable unit—crucial for audits, traceability, and public trust in smart city systems.

5. Results

In this section, we present the results of our experimental evaluation of post-quantum cryptographic (PQC) algorithms on resource-constrained IoT hardware using real-world smart city sensor data. Our analysis focuses on both key encapsulation mechanisms (KEMs) and digital signature algorithms that are currently in the final stages of standardization by NIST. Using our custom-built research prototype—which encrypts and digitally signs typical smart city data—we benchmarked the average computation times for key generation, encapsulation, decapsulation, signing, and verification for two different Raspberry Pi models.

For all graphs shown, the results for the Rust implementation of Kyber are denoted with *rust, e.g. kyber512rust for the Kyber512 results based on the Rust implementation (Argyle-Software, 2025). All other results are based on the C implementation of the PQClean library (Kannwischer et al., 2022).

5.1 Key Encapsulation (KEM) Performance

Figure 1 shows results we obtained for the time needed for key encapsulation on a Raspberry Pi 3B for various PQC key encapsulation algorithms/parametrisations. Similarly, Figure 2 displays the time needed for key encapsulation on a Raspberry Pi Zero 2W for various PQC key encapsulation algorithms/parametrisations.

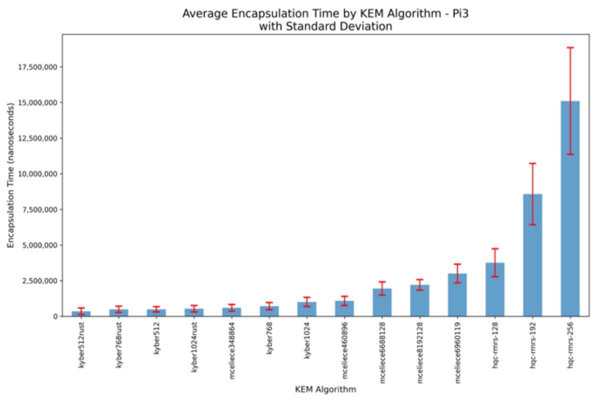

Figure 1: Encapsulation time on a Raspberry Pi 3B for various PQC Key Encapsulation Algorithms/Parametrisations

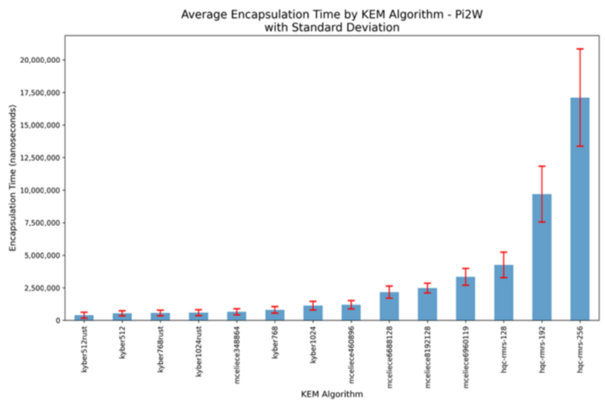

Figure 2: Encapsulation time on a Raspberry Pi Zero 2W for various PQC Key Encapsulation Algorithms/Parametrisations

The Kyber family (Kyber512, Kyber768, Kyber1024) consistently achieves the lowest encapsulation times across both Raspberry Pi devices, making it the most efficient choice for time-sensitive applications. In contrast, the McEliece and HQC families exhibit significantly higher encapsulation times, particularly at higher security levels, with HQC being the most computationally intensive. As expected, encapsulation operations are slower on the Pi2W than on the Pi3, but performance trends remain consistent across devices.

5.2 Digital Signature Performance

Figure 3 shows results we obtained for the needed signature time on a Raspberry Pi Zero 2W for various PQC signature algorithms/parametrisations. Figure 4 shows similar results obtained on a Raspberry Pi Zero 2W.

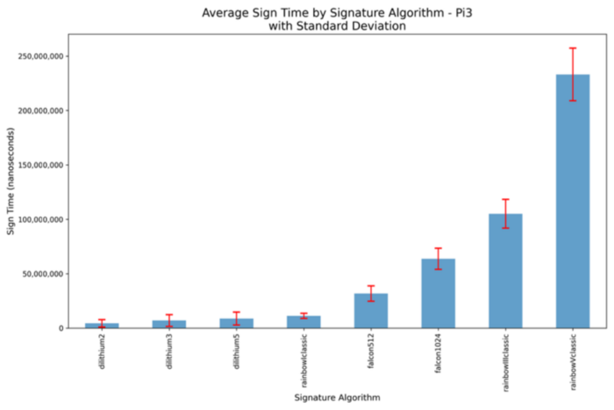

Figure 3: Signature time on a Raspberry Pi 3B for various PQC Signature Algorithms/Parametrisations

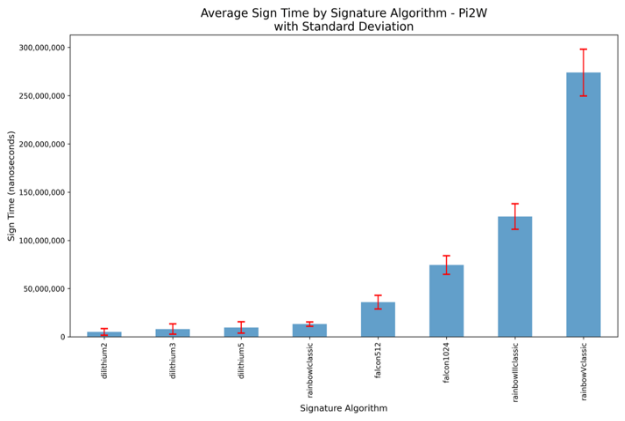

Figure 4: Signature time on a Raspberry Pi Zero 2W for various PQC Signature Algorithms/Parametrisations

Dilithium variants (2, 3, 5) show the fastest signing times, making them highly suitable for constrained devices, while Falcon offers slightly higher but still efficient performance. In contrast, the Rainbow family—especially RainbowVclassic—exhibits extremely high signing times, rendering it impractical for low-power environments. Performance differences are consistent across both Raspberry Pi models (the Pi2W generally slower). The graphs confirm the efficiency of Dilithium and Falcon compared to the heavy computational load of Rainbow.

5.3 Decapsulation Performance

Figure 5 and Figure 6 show measured decapsulation times for Kyber variants on a Raspberry Pi 3B and a Raspberry Pi Zero

2W, respectively. Figure 7 and Figure 8 show similar results for all the other (i.e. non-Kyber) PQC KEM algorithms.

The server-side evaluation highlights clear differences in performance among post-quantum KEM algorithms, with the Kyber family demonstrating the highest efficiency, particularly Kyber512 in Rust, which had the lowest decapsulation times. In contrast, non-Kyber algorithms such as McEliece and HQC showed scalability and performance challenges, with McEliece8192128 exhibiting the highest computational overhead.

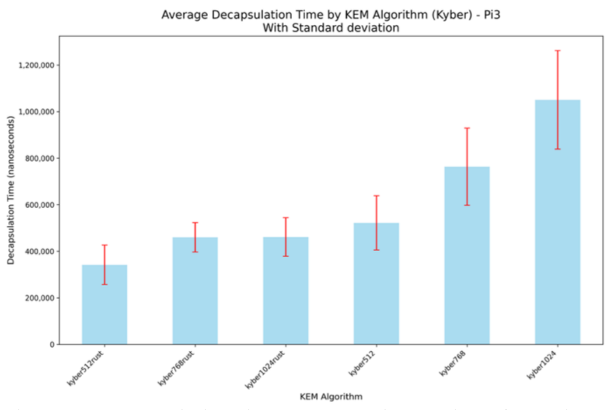

Figure 5: Decapsulation time on a Raspberry Pi 3B for various Kyber Key Encapsulation Implementations/Parametrisations

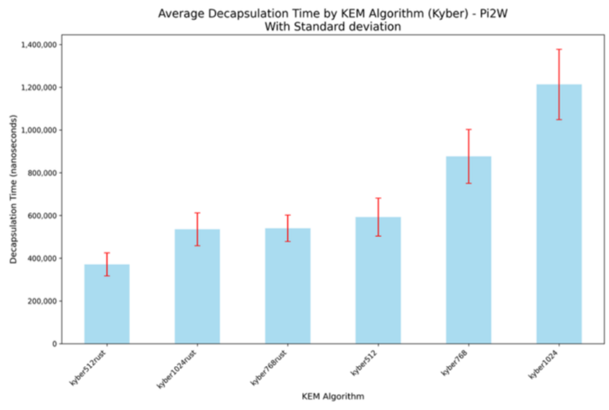

Figure 6: Decapsulation time on a Raspberry Pi Zero 2W for various Kyber Key Encapsulation Implementations /Parametrisations

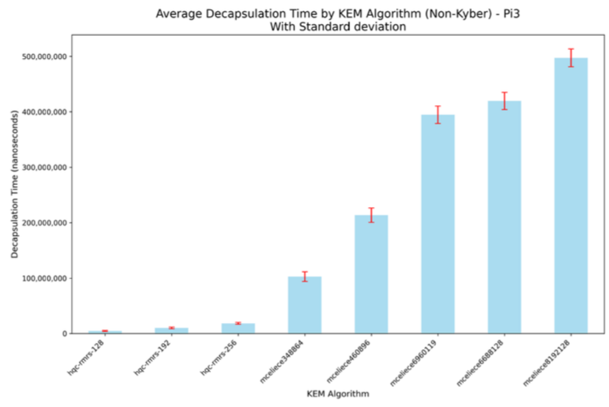

Figure 7: Decapsulation time on a Raspberry Pi 3B for various Key Encapsulation Algorithms/Parametrisations

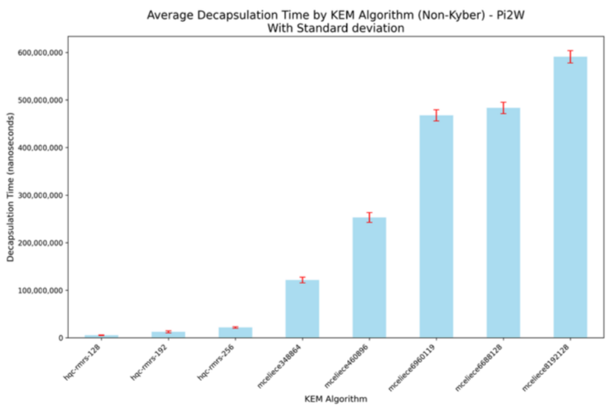

Figure 8: Decapsulation time on a Raspberry Pi Zero 2W for various Key Encapsulation Algorithms/Parametrisations

Decapsulation was slightly faster on the Raspberry Pi 3 compared to the Pi 2W, reflecting their hardware differences. Overall, Kyber consistently outperformed other algorithms on both platforms, making it the most suitable choice for decapsulation in resource-constrained or time-sensitive environments.

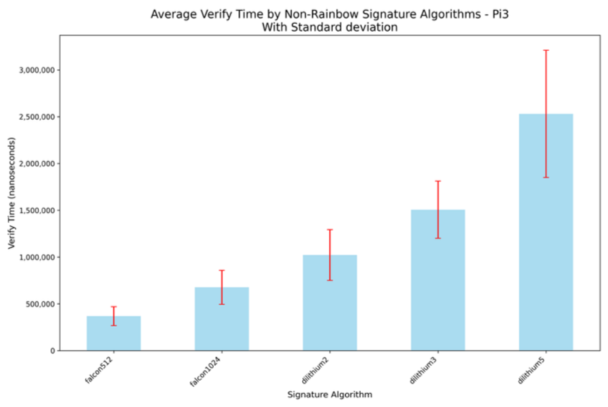

Figure 9: Signature Verification time for selected PQC Signature Algorithms on a Raspberry Pi 3B

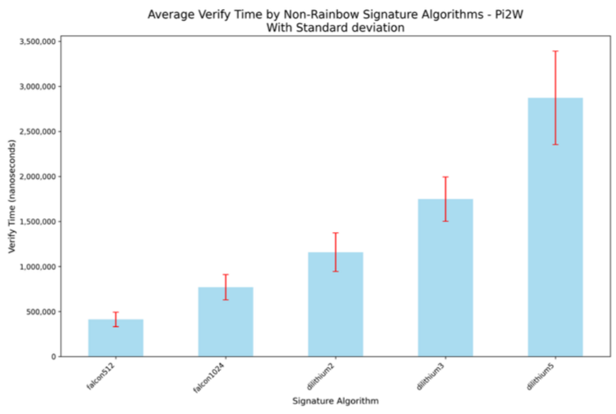

Figure 10: Signature Verification time for selected PQC Signature Algorithms on a Raspberry Pi Zero 2W

5.4 Signature Verification Performance

Figure 9 and Figure 10 display the signature verification times we measured on a Raspberry Pi 3B and a Raspberry Pi Zero 2W, respectively. Note that we did not further include Rainbow results in our analysis because the algorithm was withdrawn from the NIST post-quantum standardization process due to the discovery of practical cryptographic attacks. As a result, it is no longer a viable or secure option and offers little relevance to current or future post-quantum applications.

The verification performance analysis shows that Falcon512 offers the fastest verification times, followed closely by Falcon1024. The Dilithium family also performs efficiently, albeit slower than Falcon, with verification times increasing moderately alongside security levels but remaining suitable for most applications.

5.5 Key Generation Times

Figure 11 and Figure 12 show the overall key generation times we measured for the various KEM algorithms and the various digital signature algorithms, resprectively.

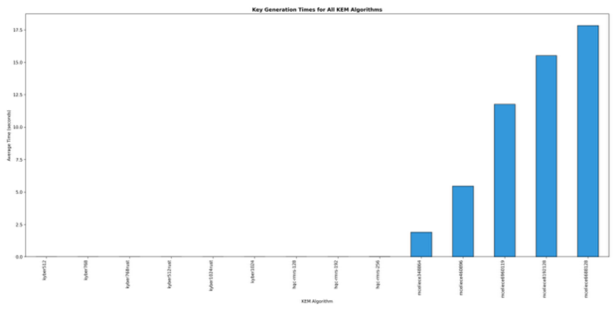

Figure 11: Key Generation Times for various Key Encapsulation algorithms

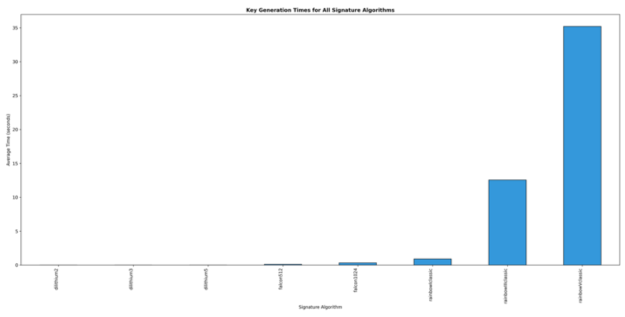

Figure 12: Key Generation Times for various Digital Signature algorithms

The analysis of key generation times for Key Encapsulation Mechanism (KEM) algorithms highlights the Kyber family as the most efficient, with consistently low key generation times under one second and minimal performance overhead. This makes Kyber highly suitable for real-time or resource-constrained environments. In contrast, the McEliece family, particularly at higher security levels, shows significantly longer key generation times—exceeding 17 seconds—which reflects its strong security guarantees but also reveals its substantial computational demands.

For signature algorithms, the Falcon family stands out with the shortest key generation times across all security levels. Falcon maintains rapid performance even at higher configurations, offering a strong balance between efficiency and cryptographic strength. This makes Falcon a practical choice for systems that require frequent key regeneration or with strict time constraints.

5.6 Summary and Discussion of Findings

Our evaluation of post-quantum cryptographic (PQC) algorithms with real-world smart city data sets on Raspberry Pi devices—specifically the Raspberry Pi 3 and Raspberry Pi Zero 2 W—has yielded valuable insights into their performance in resource-constrained environments. These findings highlight the practical implications of deploying quantum-resistant algorithms on low-power edge devices. Overall, Kyber and Dilithium are the most suitable choices for encapsulation and signing in resource-constrained smart city deployments.

In the realm of Key Encapsulation Mechanisms (KEMs), the Kyber family consistently stands out as the most efficient across all key operations, including key generation, encapsulation, and decapsulation. Its low computational overhead makes it particularly well-suited for constrained devices. In contrast, while McEliece-based algorithms offer strong security guarantees, they impose substantial computational costs, especially at higher security levels. This makes McEliece less practical for time-sensitive or interactive applications.

For digital signatures, Falcon demonstrates the best performance in terms of key generation and signature verification, offering a strong balance between speed and security. However, the Dilithium family also performs efficiently, and is notably faster in signing, which may be key in smart sensor networks, as the key generation needs to be performed less frequently and the verification is usually done at (powerful) servers, while signing sensor data happens at the resource-contrained sensor node and with high frequency.

When comparing devices, the Raspberry Pi Zero 2 W shows consistently slower performance than the Raspberry Pi 3, attributed to its more limited hardware resources. On average, operations on the Pi Zero 2 W are approximately 20% slower. In terms of specific cryptographic roles, server-side operations such as signature verification and decapsulation highlight the efficiency of Kyber and Falcon. On the client side, encapsulation and signing tasks further reinforce Kyber's suitability for KEM and the utility of Falcon and Dilithium for digital signatures.

In summary, the Kyber family is the most practical choice for applications requiring efficient and lightweight KEM operations. Falcon and Dilithium offer strong candidates for signature-based use cases, combining security and performance in a balanced way. McEliece and Rainbow, while offering theoretical strengths in certain areas, are best reserved for applications where performance is not a critical constraint and maximum security is prioritized.

6. Conclusion

6.1 Summary of Contribution and Results

This study has provided a practical assessment of post-quantum cryptographic (PQC) algorithms on resource-constrained platforms, specifically the Raspberry Pi 3 and Raspberry Pi Zero 2 W, using real-world smart city datasets to simulate realistic operational conditions. Through detailed benchmarking, we identified Kyber as the most efficient key encapsulation mechanism, demonstrating strong performance across key generation, encapsulation, and decapsulation. For digital signatures, Falcon and Dilithium emerged as the most suitable candidates, balancing low computational overheads with robust security even at higher security levels. In contrast, McEliece and Rainbow, despite their theoretical resilience, exhibited significant performance bottlenecks that make them likely impractical for many time-sensitive or embedded deployments.

By incorporating authentic data representative of smart city scenarios, our findings underscore the feasibility of integrating quantum-safe cryptographic primitives into real-world IoT and edge computing environments. Kyber, Falcon, and Dilithium provide viable, efficient alternatives to classical public-key systems and are particularly well-suited for applications in connected urban infrastructure, sensor networks, and lightweight client devices. As the cryptographic community moves toward post-quantum readiness, this evaluation offers actionable insights for guiding secure and efficient adoption on constrained platforms operating in data-rich environments.

6.2 Future Work

While this study provides an initial exploration into the practical deployment of post-quantum cryptographic algorithms on constrained devices using real-world smart city data, it represents only a first step toward comprehensive integration. Our evaluation focused on core cryptographic operations in isolation, and further research is needed to understand the broader system-level implications of PQC adoption in real-world environments.

Future work should involve integrating PQC algorithms into full communication stacks and security protocols (e.g., TLS 1.3, MQTT, and CoAP) to assess their impact on end-to-end system performance. This includes examining latency, throughput, and interoperability with existing infrastructures. Additionally, detailed analysis of side-channel resistance, energy consumption, and memory footprint will be essential, particularly for battery-powered or ultra-low-power deployments. Including hybrid cryptographic approaches and emerging PQC candidates will also help address transitional needs and optimize for various application contexts across smart cities and industrial IoT ecosystems.

Acknowledgment

This work is an outcome of the project "Datasecurity4iCity", a subproject of "iCity: Intelligent City" (https://www.hft-stuttgart.com/research/projects/i-city). We extend our gratitude for the funding received through the FHImpuls program under the number 13FH9E04IA by the German Federal Ministry of Education and Research (BMBF).

References

Argyle-Software, 2025. Argyle-Software/kyber: A rust implementation of the Kyber post-quantum KEM, https://github.com/Argyle-Software/kyber, (last visited: April 25, 2025)

Bernstein, D.J., Chou, T., Hülsing, A., Lange, T., Niederhagen, R., van Vredendaal, C., 2017. Classic McEliece: conservative code-based cryptography. Submission to the NIST Post-Quantum Cryptography Standardization Project, National Institute of Standards and Technology. https://classic.mceliece.org/nist.html

Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P., Wilms, F., 2019. SPHINCS+: Submission to the NIST Post-Quantum Cryptography Standardization Project. National Institute of Standards and Technology (NIST). https://sphincs.org/data/sphincs+-round3-specification.pdf (last visited: July 5, 2025)

Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., 2018. CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. 2018 IEEE European Symposium on Security and Privacy, London, UK, 353–367. https://doi.org/10.1109/EuroSP.2018.00032

Chen, L., Moody, D., Liu, Y.-K., 2025. NIST (National Institute for Standards and Technology): Post-Quantum Cryptography PQC, https://csrc.nist.gov/projects/post-quantum-cryptography, (last visited: July 1, 2025)

Ding, J., Petzoldt, A., Schmidt, D., 2017. Rainbow: A multivariable public key signature scheme. Submission to the NIST Post-Quantum Cryptography Standardization Project, National Institute of Standards and Technology (NIST). https://pq-crystals.org/rainbow/

Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D., 2018. CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems, 2018(1), 238–268. https://doi.org/10.13154/tches.v2018.i1.238-268

Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z., 2018. FALCON: Fast-Fourier lattice-based compact signatures over NTRU. Submission to the NIST Post-Quantum Cryptography Standardization Project, 36(5), 1–75. https://falcon-sign.info/ (last visited: July 1, 2025)

Kannwischer, M.J.., Schwabe, P., Stebila, D., Wiggers, T., 2022. Improving Software Quality in Cryptography Standardization Projects, European Symposium on Security and Privacy (Workshops), Genoa, Italy, June 6-10, 2022, IEEE Computer Society, Los Alamitos, CA, USA doi.org/10.1109/EuroSPW55150.2022.00010.

NIST, 2024a. FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM). Federal Information Processing Standards Publication 203, National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.FIPS.203

NIST, 2024b. FIPS 204: Module-Lattice-Based Digital Signature Algorithm (ML-DSA). Federal Information Processing Standards Publication 204, National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.FIPS.204

NIST, 2024c. FIPS 205: Stateless Hash-Based Digital Signature Algorithm (SLH-DSA). Federal Information Processing Standards Publication 205, National Institute of Standards and Technology, Gaithersburg, MD. https://doi.org/10.6028/NIST.FIPS.205

Pallets, 2024. Flask (Version 3.1.0) [Computer software]. Python Package Index (PyPI). https://palletsprojects.com/ (last visited: July 1, 2025)

Shor, P.W., 1994. Algorithms for quantum computation: Discrete logarithms and factoring. Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS), Santa Fe, NM, 20–22 November 1994, 124–134. https://doi.org/10.1109/SFCS.1994.365700