Post-Quantum Digital Signature Algorithms on IoT: Evaluating Performance on LoRa ESP32 Microcontroller

Mads Villum Nielsen[a], Magnus Raagaard Kjeldsen[b], Togu Turnip[c] and Birger Andersen[d]
Technical University of Denmark, Copenhagen, Denmark

Keywords: Post-Quantum Cryptography, Digital Signature, Resource-Constrained Devices, Internet of Things.

Abstract: As quantum computing continues to develop, traditional cryptographic schemes face increasing threats from quantum attacks, driving the need for post-quantum cryptographic (PQC) algorithms. This study evaluates the feasibility of PQC algorithms with higher NIST security level parameters on a resource-constrained IoT device, the LoRa ESP32 microcontroller. We benchmarked the performance of *CRYSTALS-Dilithium*, *Falcon*, *SPHINCS*$^+$ across multiple NIST levels, measuring latency, memory usage and discussing parameter sizes. Additionally, we examined the communication overhead introduced by transmitting the larger-than-usual digital signatures over a wireless network. Our findings reveal significant performance disparities between the tested algorithms, with *Dilithium* demonstrating the fastest execution and *Falcon* balancing speed and memory efficiency – even at higher NIST security levels. In contrast, *SPHINCS*$^+$ proved impractically slow for IoT applications. This research investigates practical considerations and challenges of deploying PQC digital signature algorithms on IoT devices.

1 INTRODUCTION

As quantum computing technology matures, it will pose a greater threat to today's cryptographic schemes. Quantum computers have the potential to break widely used public-key encryption systems, such as RSA, which play a crucial role in secure communication on the internet (Bhatia and Ramkumar, 2020). These cryptographic algorithms rely on mathematical problems, such as integer factorization, that are computationally challenging for classical computers, but it has been shown that some of the algorithms can be solved efficiently by quantum computers by using algorithms like Shor's algorithm (Shor, 1997).

In addressing this threat, the National Institute of Standardization and Technology (NIST) has been leading to find candidates for quantum-resistant cryptographic schemes in 2017. In 2022, a selection of several promising algorithms was made (NIST, 2022). Among these, NIST selected three digital signature schemes: *CRYSTALS-Dilithium*, *Falcon* and *SPHINCS*$^+$ (Shajahan, et al., 2024).

While the security of these algorithms has been analysed, some of their performance metrics on resource-constrained devices such as Internet of Things (IoT) microcontrollers remains underexplored (Liu et al., 2024). IoT devices often have limited computational power, memory or might run on batteries with limited energy capacity, which can present a challenge for the implementation of post-quantum cryptographic (PQC) algorithms (Tan et al., 2022). The IoT is rapidly expanding, with billions of devices deployed for various applications, such as smart homes, healthcare, agriculture and even critical infrastructure (Mansoor et al., 2025). A key property of IoT devices is their ability to handle data and communicate over insecure networks. Many of these applications require handling and communication of sensitive data, and failing to implement quantum-resistant cryptographic schemes leaves them vulnerable to attackers. IoT devices are often deployed in environments where they are expected to

[a] https://orcid.org/0009-0001-2240-3578
[b] https://orcid.org/0009-0006-6971-6885
[c] https://orcid.org/0000-0002-8269-0572
[d] https://orcid.org/0000-0003-1402-0355

Nielsen, M. V., Kjeldsen, M. R., Turnip, T. and Andersen, B.
Post-Quantum Digital Signature Algorithms on IoT: Evaluating Performance on LoRa ESP32 Microcontroller.
DOI: 10.5220/0013508400003979
In *Proceedings of the 22nd International Conference on Security and Cryptography (SECRYPT 2025)*, pages 592-600
ISBN: 978-989-758-760-3; ISSN: 2184-7711

operate for prolonged periods of time without frequent updates or replacement. This makes these devices particularly susceptible to future quantum threats (Cheng et al., 2017). Ensuring that the PQC algorithms can be implemented on IoT devices is crucial to their future-proofing, especially as society gets increasingly more dependent on such devices.

While PQC algorithms are being standardized and tested extensively, the feasibility of implementing algorithms with parameters corresponding to higher NIST security levels on resource-constrained IoT devices remains less explored. The more secure PQC algorithms can pose a challenge by being more computationally intensive and requiring more memory to operate. The main contributions of this paper are summarized as follows:

1. Evaluation of PQC Digital Signature at Higher NIST security level on IoT hardware: We evaluate the computational performance and memory requirements of CRYSTALS-Dilithium, Falcon, and SPHINCS+ algorithms at high NIST security levels on the ESP32 IoT microcontroller.
2. Assessment of practical communication overhead: We empirically analyze the latency overhead introduced by wirelessly transmitting large PQC digital signatures, providing insights into the realistic feasibility and impact of deploying PQC digital signature algorithms in bandwidth-limited IoT scenarios.

2 RELATED WORK

Recent studies have investigated the performance of PQC on embedded and resource-constrained devices, with a particular focus on digital signature algorithms. Halak et al. (2024) conducted an evaluation of PQC schemes on constrained hardware, employing a testbed to assess latency, memory consumption, and energy dissipation. Their findings highlighted *Dilithium2* as a computationally efficient scheme while deeming *SPHINCS+* impractical for IoT applications due to its high resource demands. However, their study primarily focused on TLS handshakes and did not include an analysis of *Falcon* or higher NIST security level schemes such as *Dilithium3*, *Dilithium5*, or *SPHINCS+256-f*, leaving gaps in the understanding of PQC feasibility at increased security levels.

Similarly, Vidaković and Miličević (2023) examined the performance of *Dilithium*, *Falcon*, and *SPHINCS+* on an ARM Cortex M4 microcontroller, considering their applicability in resource-constrained environments. While this study provided valuable insights, it did not extend the analysis to higher NIST security levels for *Dilithium* or *SPHINCS+*, thus leaving the question of what scheme is best suitable for resource-constrained devices on higher NIST security levels. In addition to these investigations, other works, Fitzgibbon and Ottaviani (2024), have assessed PQC performance in constrained settings. While these studies contribute to understanding PQC implementation challenges, they predominantly focus on lower security levels and do not comprehensively address the feasibility of higher NIST security level digital signature schemes in resource-constrained environments.

Fournaris et al. (2023) and Kannwischer et al. (2024) examined memory usage, runtime efficiency, and energy performance of NIST PQC candidates on resource-constrained microcontrollers. Both studies identified Dilithium as advantageous due to its balanced performance compared to Falcon and SPHINCS+, which exhibited higher complexity and resource demands. However, neither study comprehensively investigated higher NIST security levels nor explicitly evaluated widely-used IoT platforms such as ESP32 microcontrollers, highlighting important gaps for future research.

As part of its standardization efforts, NIST has introduced two distinct standards for post-quantum digital signature algorithms. The Module-Lattice-Based Digital Signature Standard (NIST, 2024a) defines a set of lattice-based digital signature schemes, including *Falcon* and *Dilithium*. *Dilithium* is based on the Module Learning with Errors (MLWE) problem, a structured variant of the Learning with Errors (LWE) problem, whereas *Falcon* relies on the Short Integer Solution (SIS) problem over *NTRU* lattices. Additionally, NIST has standardized a Stateless Hash-Based Digital Signature Standard (NIST, 2024b), which includes the *SPHINCS+* algorithm. This scheme relies on the computational hardness of finding collisions in hash functions and is distinguished by its stateless nature, meaning that it does not require maintaining an internal state during the signing. While *SPHINCS+* offers strong security assurances, its computational overhead remains a challenge for resource-constrained environments (Opiłka, et al., 2024).

Despite these ongoing standardization efforts and recent studies, there remains a research gap regarding the performance and applicability of higher security level PQC digital signature algorithms in constrained environments. Additionally, evaluations on ESP32 microcontrollers remain limited. Thus, further research addressing these gaps is essential to clarify

the practicality of higher-security PQC algorithms in constrained environments.

3 EXPERIMENTAL SETUP

To benchmark the performance of the selected post-quantum cryptographic signature algorithms, we implemented the algorithms on the *ESP32* microcontroller. The *SPHINCS*$^+$ scheme has two variants: a fast variant with a large signature size, and a slower variant but with a smaller signature size. Our attention was directed towards the fast variant, given its greater computational practicality for the ESP32 microcontroller. Table 1 summarizes PQC digital signature algorithms and their NIST security levels, ranging from Level I for basic security to Level V for the highest security.

Table 1: Algorithms and their NIST Security levels

Algorithms	NIST Security Level
Falcon512	I
SPHINCS$^+$128-f	I
SPHINCS$^+$192-f	III
Dilithium3	III
Dilithium5	V
Falcon1024	V
SPHINCS$^+$256-f	V

First, the algorithms were executed locally on the microcontroller. Subsequently, the algorithms that seemed practical were tested over Wi-Fi. Here our aim was to test whether the large signature sizes created by the algorithms would introduce additional latency during communication, that may make the algorithms infeasible. This approach simulates real-life scenarios.

The figure 1 illustrates a testbed setup where a LoRa ESP32 IoT device communicates with an unconstrained device via a router, enabling wireless interactions for performance benchmarking or data exchange. When the algorithms were tested locally on the microcontroller, the unconstrained device would log the output of the program, using the USB connection. When the wireless communication was tested, the unconstrained device measured the round-trip time of signing requests made to the microcontroller using a socket connection.

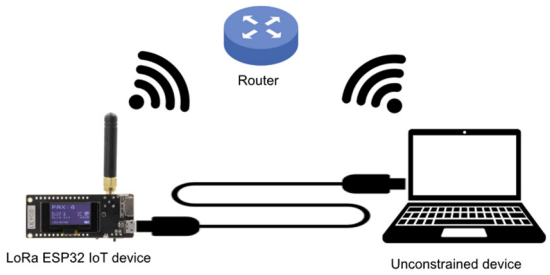

Figure 1: Test bed measuring latency and memory usage of PQC algorithms when communicating with a resource-constrained device.

3.1 Implementation

The implementation of each algorithm was obtained from the *PQClean*[1] *library* (Kannwischer et al., 2022), a collection of reference implementations for NIST-standardized PQC algorithms. The complete source code used in this study is available on GitHub[2]. All algorithms were implemented on an ESP32 microcontroller, with the specifications detailed in Table 2.

Table 2: Specifications of the ESP32 microcontroller.

Parameter	Description
Model	LoRa ESP32
Processor	Xtensa dual-core 32-bit LX7
Clock Speed	240MHz
SRAM	512KB
PSRAM	2MB
Flash	4MB

When benchmarking PQC digital signatures on the ESP32, SRAM handles real-time computations and temporary data, PSRAM stores large keys and signatures when SRAM is insufficient, and Flash Memory holds the algorithms but is too slow for execution. These three memories work together to run and evaluate PQC performance, though resource limits may influence execution.

3.2 Measurements

Multiple performance metrics were captured for each algorithm. The time-related metrics were measured using the high-resolution microsecond timer available on the *ESP32* platform. For measuring the memory usage, the minimum free heap was measured

[1] https://github.com/PQClean/PQClean.

[2] https://github.com/madsvnielsen/ESP32-PQSignature-Test

and subtracted from the allocated heap to measure how much dynamically allocated memory was required. The stack size was kept constant across all implementations to ensure constant measurement conditions. In total, the following data points were captured for all algorithms as shown in Table 3.

Table 3: Metrics measured for each algorithm.

Latency			Memory	Param. sizes		
Key	Sign	Verify	Heap usage	pk	sk	sig

The parameters and heap usage are constant and do not vary between different executions. However, the execution times would vary so each operation was repeated multiple times, and the mean values were calculated. This approach ensured that any anomalies or outliers were minimized, providing a more reliable comparison of the algorithms' performance. Afterwards, we aimed to measure the communication overhead for each algorithm. For this setup, an unconstrained device and the *ESP32* microcontroller was connected to the same local internet over Wi-Fi. The microcontroller listens on a port, and once a packet is received it signs the packet body and replies with the signature. This experiment is meant to investigate the communication overhead introduced by the algorithms.

Following this methodology, we aimed to evaluate the computational feasibility of running the PQC algorithms on resource-constrained IoT devices and provide insights into their efficiency and resource demands.

4 RESULTS AND DISCUSSION

In this section, a discussion of the measured metrics of the *SPHINCS*$^+$, *Dilithium* and *Falcon* algorithms with different parameters will be compared and discussed. The comparisons include computation time, heap usage and the practical implications of the public key sizes, secret key sizes and signature sizes. Furthermore, the communication overhead of the digital signatures is investigated.

4.1 Parameter Analysis

To evaluate the feasibility of the post-quantum algorithms, we analysed the cryptographic parameters of each algorithm. These parameters are the public key (*pk*) size, secret key (*sk*) size and signature (s*ig*) size.

The public key is distributed to verify digital signatures. A smaller *pk* size minimizes communication overhead, making the algorithm more suitable for IoT devices with limited bandwidth. The secret key is stored solely on the signing device and never shared. Therefore, the *sk* size only impacts the memory requirements for the device. The signature size is the resulting output of the signing process. This is transmitted to the recipient for verification. A smaller signature size decreases communication cost and latency. On devices with limited bandwidth, this is also a crucial consideration. The algorithms and their respective parameter values are summarized in Table 4.

Table 4: Parameter sizes for the different algorithms.

Algorithm	***pk size (bytes)***	***sk size (bytes)***	***sig size (bytes)***
Falcon512	897	1281	752
Falcon1024	1793	2305	1462
Dilithium2	1312	2560	2420
Dilithium3	1952	4032	3309
Dilithium5	2592	4896	4627
SPHINCS+-128-f	32	64	17088
SPHINCS+-192-f	48	96	35664
SPHINCS+-256-f	64	128	49856

There is a noticeable difference between the parameter sizes for the different algorithms. The *SPHINCS*$^+$ scheme has an exceptionally small *pk* and *sk* size, but a large signature size. The large signature could pose a significant challenge for IoT devices with limited bandwidth and memory. The *Falcon* scheme offers relatively small *pk* and *sig* sizes, making it more ideal for real-time communication based on the parameter sizes. Noticeably, the *Falcon1024* has surprisingly small parameter sizes, considering that it is a NIST security level 5 algorithm. The *Dilithium* parameters are all larger than the *Falcon512* scheme, but still with a small signature size. A comparison of the parameter sizes can be seen in Figure 2.

Analysis of parameter sizes demonstrates that the *Dilithium* and *Falcon* schemes reveal advantages over the *SPHINCS+* scheme, especially at higher NIST security levels, with *Falcon1024* being a significant highlight. *Dilithium* and *Falcon*, having to their comparatively smaller key and signature sizes, are promising candidates for PQC implementations on resource-constrained IoT devices, provided their computational feasibility and memory efficiency are suitable.

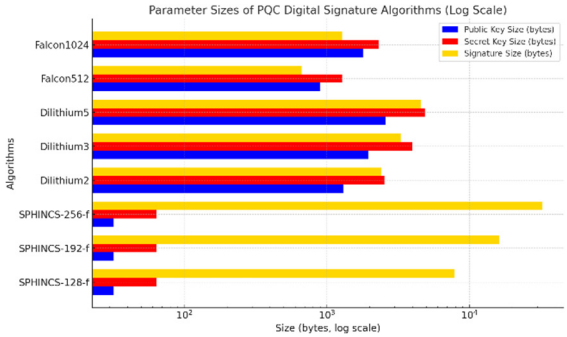

Figure 2: Comparison of parameter sizes showing that *SPHINCS*⁺ scheme has the smallest public and secret key size but with a very large signature size.

4.2 Latency Analysis

This part summarizes the findings from the time required for the three fundamental cryptographic operations of digital signatures: key generation, signing, and verification.

4.2.1 Key Generation

Key generation is the initial phase of the cryptographic protocols, where the public and secret keys are generated. Generating the random seeds required for the key generation, we used the platform-specific method *esp_fill_random*. The timing was measured in microseconds using the high-resolution timer offered by the ESP-platform. Specifically, the method *esp_timer_get_time* was used. The algorithms used for key generations vary significantly between the algorithms.

To show the different latencies based on key generation, signing and verification we had to use a logarithmic scale, as there was a large difference in latency between the PQC algorithms. The mean of all time measurements was calculated to accurately measure the latency. It is worth noting that the sample sizes for the *SPHINCS*⁺ algorithms are smaller, as doing many operations was time-consuming due to the latency of the algorithm. The latencies are plotted and compared in Figure 3.

During key generation, we see *Dilithium* being by far the fastest scheme, whilst *Falcon* is significantly slower and *SPHINCS*⁺ being in between. As expected, all algorithms get slower, the higher the security level it has. More accurately, the typical key generation for *Falcon1024* is 80 times slower than *Dilithium5* even though they have the same NIST security level. To generate one key pair, it takes *Falcon1024* nearly 4 seconds. Whilst *SPHINCS*⁺ is faster than *Falcon* it still takes nearly a second to generate a key pair and this is only on the security level 2.

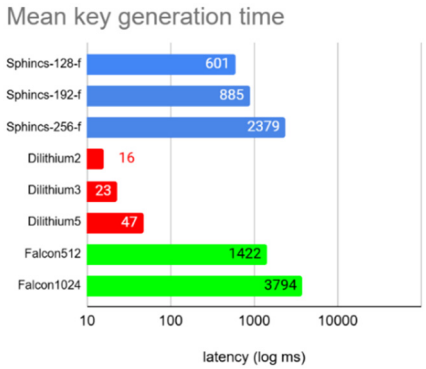

Figure 3: Mean key generation times in ms using a logarithmic scale showing *Dilithium* as the fastest scheme with *SPHINCS*⁺ being an order of magnitude slower and *Falcon* nearly two orders of magnitude slower.

4.2.2 Signing

In this section, the signing latency of each algorithm will be evaluated. The signing latency is the measured time it takes to produce a digital signature for a fixed-length message of 14 bytes. A logarithmic scale is used due to significant latency differences. A comparison of the measurements can be seen in Figure 4.

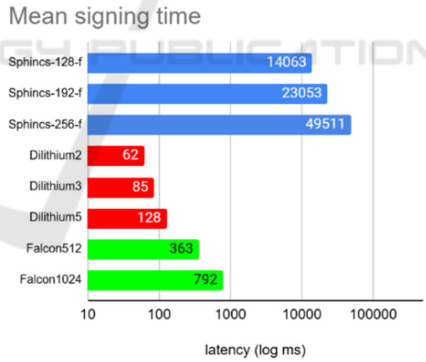

Figure 4: Mean signing times in ms using a logarithmic scale showing *Dilithium* as the fastest scheme and *SPHINCS*⁺ being more than 200 times slower.

For the signing times, we can see that *SPHINCS*⁺ is now the slowest algorithm. With it taking as much as 49,5 seconds for *SPHINCS*⁺*-256-f* to sign a message, the measurements indicates that *SPHINCS*⁺ is not usable on IoT devices, as most use cases would require a much smaller signing time. *Falcon* and *Dilithium* perform reasonably well. When comparing NIST level 5 schemes, *Falcon1024* and *Dilithium5*, *Dilithium5* clearly outperforms in signing latency.

4.2.3 Verification

The verification process is responsible for verifying the authenticity of a given signature. As well as in the signing tests, the verification step was timed. The verification times are important, as verifications are done relatively often. Depending on the use case, either signing or verification may have a greater impact on the overall performance. The logarithmic scale was used for comparison. The results are seen in Figure 5.

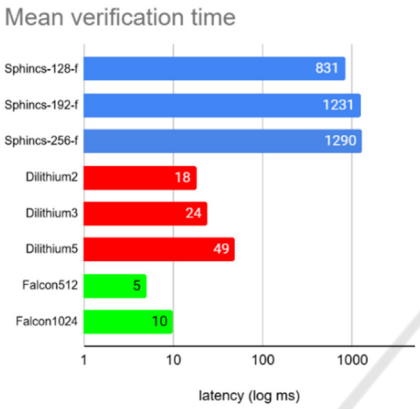

Figure 5: Mean verification times in ms using a logarithmic scale showing *Falcon* as the fastest scheme with *Dilithium* taking twice as long and *SPHINCS*$^+$ being more than 100 times slower.

For the verification times, *Falcon* is now the fastest. *SPHINCS*$^+$ is still the slowest but for verification, it is much less pronounced. Unlike the other parts of the process, all these latencies are arguably fast enough for many use cases. The verification latency is important, as signatures are required to be checked frequently when receiving packets.

4.4 Memory Analysis

We measured the heap usage by logging the allocated heap and the minimum free heap available. Using these numbers, we inferred the heap memory usage.

We measured *Dilithium5* to be the most memory-intensive of all the algorithms. There is a trade-off with *Dilithium* being the fastest algorithm but also requiring the most amount of memory. Despite the large signature size for the *SPHINCS*$^+$ scheme, it was the most memory-efficient scheme.

The figure 6 reveals that *SPHINCS+* is the most memory-efficient PQC digital signature algorithm, with *SPHINCS-128-f* requiring just 60 kB of heap space, and even its highest-security variant, *SPHINCS-256-f*, using only 96 kB, making it ideal for resource-constrained environments. In contrast, *Dilithium* exhibits a sharp rise in memory usage at higher security levels, with *Dilithium5* demanding 305 kB, which is significantly higher than other algorithms. *Falcon* strikes a balance, with *Falcon512* and *Falcon1024* requiring 86 kB and 130 kB, respectively, offering a more alternative compared to high-security *Dilithium* variants.

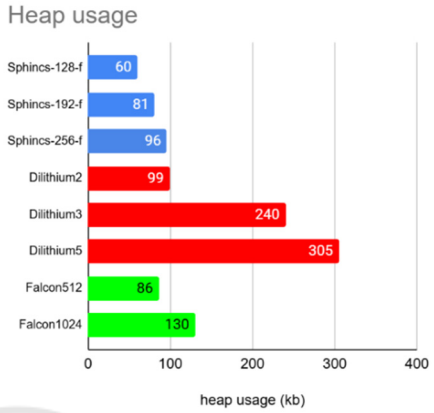

Figure 6: Heap usage measured in KB showing *Dilithium* as using the most memory whilst *SPHINCS*$^+$ and *Falcon* being roughly equal and using half the amount of KB.

4.5 Round-Trip Latency for Signing via Wi-Fi

In the second stage of testing, we want to test the real-time performance of the promising PQC algorithms. This was done to test whether the large signature sizes could introduce additional latency during communication. We decided to test all versions of *Dilithium* and *Falcon* over Wi-Fi. *SPHINCS*$^+$ latency was too high, so we decided that it was unnecessary to do any further testing.

The microcontroller would listen on port 8000, and upon receiving a message, it would sign the message with a pre-computed key and reply with the signature. As in the previous test, we evaluated the latency using the different parameters corresponding to the different NIST security levels. An unconstrained device measures the round-trip time to investigate the latency and the introduced communication overhead. While sending a message to a microcontroller to get it signed may not be a common use case, it reflects other real-world scenarios reasonably well. In such cases, the request would be sent to the controller, the requested data signed, and both the digital signature and data returned to the user. This experiment evaluates the difference in overhead of the signing process for the

different algorithms. A linear scale was used for comparison. The results are seen in Figure 7.

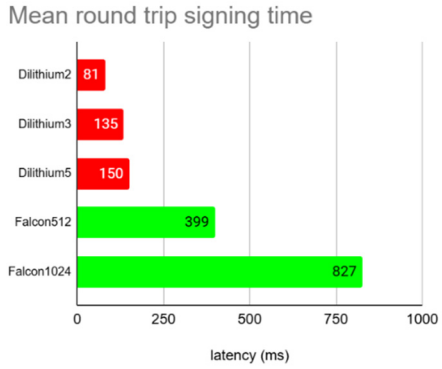

Figure 7: Mean round trip signing times in ms using a linear scale showing *Dilithium* as the fastest scheme and *Falcon* being five times slower.

There is not a significant difference in latency compared to the local test. For *Dilithium* the latency is only increased by around 19 to 50 millisecond and a 35% average increase. For *Falcon* we see about the same increase in latency around 35 ms. This test demonstrates that the additional latency caused by transmitting large signature sizes over Wi-Fi is both constant and minimal, allowing the speed of the algorithms to remain consistent.

4.6 Comparison of the Implemented Algorithms

Based on Table 5, the security level of *Dilithium5* is the highest possible whilst also being faster than any version of *SPHINCS*$^+$ and *Falcon*. This makes *Dilithium* more desirable as it maintains speed whilst not compromising security. However, if your device has less memory than an ESP32 microcontroller, *Falcon* may be the more suitable algorithm. *Falcon* is slower, but in turn, it has both smaller parameter sizes and heap usage.

Adding all the timings together from the tests done locally over a USB connection, we can clearly see that *SPHINCS*$^+$ is significantly the lowest, followed by *Falcon,* which remains considerably slower than the fastest algorithm, *Dilithium*. Whilst *SPHINCS*$^+$ shows minimal memory consumption, it demonstrates low overall performance. Because of the latency of *SPHINCS*$^+$, it does not seem suitable for use on a resource-limited IoT device. Its security level is lower, and the key generation and verification latency is so significant, that the delay would be unacceptable for any use case.

Table 5: Summary of all metrics.

Metrics	Sphincs-128-f	Sphincs-192-f	Sphincs-256-f	Dilithium 2	Dilithium 3	Dilithium 5	Falcon512	Falcon1024
pk size (B)	32	48	64	1312	1952	2592	897	1793
sk size (B)	64	96	128	2560	4032	4896	1281	2305
sig size (B)	17088	35664	49856	2420	3309	4627	752	1462
key gen (ms)	601	885	2379	16	23	47	1422	3794
sign (ms)	14063	23053	49511	62	85	128	363	792
verify (ms)	831	1231	1290	18	24	49	5	10
total (ms)	15495	25169	53180	96	132	224	1790	4596
heap (KB)	60	81	96	99	240	305	86	130
Sign RTT (ms)	-	-	-	81	135	150	399	827

These findings align with another study that found a very significant difference in latency between the algorithms. As with our experiments, *Dilithium* was faster in key generation and signing, *Falcon* fastest at verification and *SPHINCS*$^+$ orders of significantly slower (Vidaković & Miličević, 2023). However, another study found that during a TLS handshake, *SPHINCS*$^+$ had a significantly larger heap usage compared to the *Dilithium* scheme (Halak et al., 2024). But this study measured that *SPHINCS*$^+$ has very low memory usage. Our methodology when dealing with the issue of limited stack size and the difference in scenarios may have contributed to the observed discrepancies.

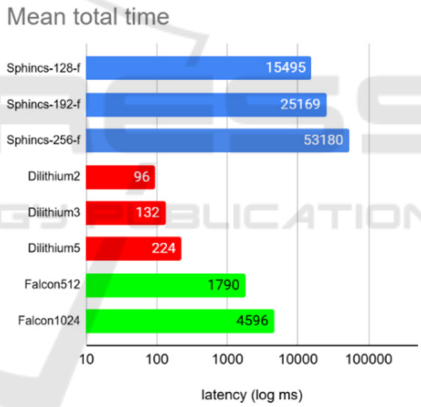

Figure 8: Total mean times in ms using a logarithmic scale showing *Dilithium* being the overall fastest scheme *Falcon* as the second and *SPHINCS*$^+$ as the slowest with it taking nearly 200 times longer than *Dilithium*.

From our experiment's results, using *Dilithium* or *Falcon* is desirable, as they both are fast enough to be run on an IoT device. *Dilithium* outperforms *Falcon* due to its structured lattice approach, reducing computational overhead. *Falcon*, while efficient, requires floating-point operations, making it slower on ESP32's integer-based architecture. Furthermore, our experiment indicates that running high NIST security level PQC algorithms is feasible even on resource-constrained IoT devices.

4.7 Challenges and Limitations

During our experiments, we encountered some constraints that influenced our results. Firstly, we were unable to run the algorithms without dynamically allocating some data on the heap. Our attempts to increase the stack size were unsuccessful, which meant that certain memory allocations could not be properly standardized across all algorithms. For all the tests, the stacks were constant at 8.192 words (32.768 bytes). We would consistently allocate the keys and signatures on the heap, but there would still be insufficient stack memory. To solve that, we inspected the key generation, verification and signing methods to identify variables with significant memory consumption. These variables were modified to utilize dynamic memory allocation. This introduced a minor inconsistency in the memory measurements, making the comparisons less accurate, as the different schemes may have been allocated slightly more heap memory than necessary. To improve this metric, an accurate measure of the used stack memory would have to be conducted. These inconsistencies mainly affected the memory usage metrics rather than the timing results and still provide insight into the memory requirements of the algorithms.

For the timing metrics, most measurements were reasonably accurate. The *SPHINCS*$^+$ algorithm timings are slightly less so due to fewer measurements being made because of the longer runtime. Despite this, the observed variance for the *SPHINCS*$^+$ timings was small enough to confidently conclude that it is significantly slower than the other alternatives. The limited flash memory of the device made it difficult to easily set up a test environment for signing larger files, constraining our ability to assess performance of varying input sizes.

5 CONCLUSION

In this study, we evaluated the NIST-selected PQC algorithms on a resource-constrained IoT device, the LoRa ESP32. Our tests covered algorithms with different NIST security levels to identify if high NIST security levels for such devices were feasible. We measured the latency and memory usage for each algorithm. The algorithms that could be run locally without excessive latency or memory usage was then subsequently tested over Wi-Fi. Our finding indicates that all version of the *SPHINCS*$^+$ scheme was too slow to be practical for any use case on resource constrained devices. Consequently, it was decided to exclude it from tests over Wi-Fi and using it on a resource constrained IoT devices does not seem practical. From the second test it was shown that the latency introduced from communicating over Wi-Fi is insignificant, regardless of the choice of algorithm. Conclusively, running PQC schemes on resource-constrained IoT devices seem feasible, even when using larger parameter sizes for increased security. Specifically, *Dilithium5* and *Falcon1024* are good candidates for schemes with high security even on resource-constrained devices.

There are several areas for future work, that would be significant and interesting to explore. Firstly, evaluating PQC signature transmission over LoRaWAN on the *ESP32* microcontroller would focus on latency and power consumption. Furthermore, testing a specific scenario and measuring more metrics, such as the stack memory and energy dissipation, would create a more complete picture of real-world use cases.

REFERENCES

Bhatia, V., Ramkumar, K. R. (2020). An efficient quantum computing technique for cracking RSA using Shor's algorithm. 2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 89–94. https://doi.org/10.1109/ICCCA49541.2020.9250806

Cheng, C., Lu, R., Petzoldt, A., and Takagi, T. (2017). Securing the Internet of Things in a quantum world. IEEE Communications Magazine, 55(2), 116–120. https://doi.org/10.1109/MCOM.2017.1600522CM

Fitzgibbon, G. Ottaviani, C. (2024). Constrained device performance benchmarking with the implementation of post-quantum cryptography. Cryptography, 8(2), 21. https://doi.org/10.3390/cryptography8020021

Fournaris, A. P., Tasopoulos, G., Brohet, M., and Regazzoni, F. (2023). Running Longer To Slim Down: Post-Quantum Cryptography on Memory-Constrained Devices. *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, Berlin, Germany, pp. 1-6. https://doi.org/10.1109/COINS57856.2023.10189268

Halak, B., Gibson, T., Henley, M., Botea, C., Heath, B., and Khan, S. (2024). Evaluation of performance, energy, and computation costs of quantum-attack resilient encryption algorithms for embedded devices. *Applied Sciences*.https://doi.org/10.1109/ACCESS.2024.3350775

Kannwischer, M. J., Schwabe, P., Stebila, D., and Wiggers, T. (2022). Improving software quality in cryptography standardization projects. In Security Standardization Research – EuroS&P Workshops 2022.

Kannwischer, M. J., Krausz, M., Petri R., and Yang S.Y. (2024). pqm4: Benchmarking NIST Additional Post-

Quantum Signature Schemes on Microcontrollers. Cryptology ePrint Archive, Paper 2024/112. https://eprint.iacr.org/2024/112

Liu, T., Ramachandran, G., and Jurdak, R. (2024). Post-quantum cryptography for Internet of Things: A survey on performance and optimization. arXiv. https://arxiv.org/abs/2401.17538

Mansoor, K., Afzal, M., Iqbal, W. et al. (2025). Securing the future: exploring post-quantum cryptography for authentication and user privacy in IoT devices. Cluster Comput 28, 93. https://doi.org/10.1007/s10586-024-04799-4

National Institute of Standards and Technology (NIST). (2022). Selected algorithms. Retrieved January 19, 2025, from https://csrc.nist.gov/projects/post-quantum-cryptography/selected-algorithms-2022

National Institute of Standards and Technology (NIST). (2024). Module-Lattice-Based Digital Signature Standard (FIPS PUB 204). Retrieved January 23, 2025, from https://doi.org/10.6028/NIST.FIPS.204

National Institute of Standards and Technology (NIST). (2024). Stateless Hash-Based Digital Signature Standard (FIPS PUB 205). Retrieved January 23, 2025, from https://doi.org/10.6028/NIST.FIPS.205

Opiłka, F., Niemiec, M., Gagliardi, M., and Kourtis, M. A. (2024). Performance analysis of post-quantum cryptography algorithms for digital signature. Applied Sciences, 14(12), 4994. https://doi.org/10.3390/app14124994

Shajahan, R., Jain, K., and Krishnan, P. (2024). A Survey on NIST 3rd Round Post Quantum Digital Signature Algorithms," *2024 5th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, Lalitpur, Nepal, pp. 132-140. https://doi.org/10.1109/ICMCSI61536.2024.00027

Shor, P. W. (1997). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Journal on Computing, 26(5):1484–1509.

Tan, T.G., Szalachowski, P. and Zhou, J. (2022). Challenges of post-quantum digital signing in real-world applications: a survey. Int. J. Inf. Secur. 21, 937–952. https://doi.org/10.1007/s10207-022-00587-6

Vidaković, M., Miličević, K. (2023). Performance and applicability of post-quantum digital signature algorithms in resource-constrained environments. Algorithms, 16(11), 518. https://doi.org/10.3390/a16110518