Post-Quantum Authentication and Integrity in 3-Layer IoT Architectures

Juliet Samandari, Clémentine Gritti

Post-Quantum Authentication and Integrity in 3-Layer IoT Architectures

Juliet Samandari, University of Canterbury, New Zealand
Clémentine Gritti, INSA Lyon, France

November 5, 2024

Abstract

The Internet of Things (IoT) is a growing area of technology and has been identified as a key tool for enhancing industries' operation and performance. As IoT deployment rises worldwide, so do the threats; hence, security, especially authentication and integrity, is a critical consideration. One significant future threat is quantum attacks, which can only be defeated using Post-Quantum (PQ) cryptosystems. New Digital Signature (DS) standards for PQ security have been selected by the US National Institute of Standards and Technology (NIST). However, IoT comes with its own technical challenges from the constrained resources allocated to sensors and other similar devices. As a consequence, the use and suitability of these PQ schemes for IoT remains an open research area. In this paper, we identify an IoT architecture built from three distinct layers represented by a server, a gateway and an IoT device, respectively. We first test PQ DS scheme standards and compare them with current standards in order to assess their practicality for use in this architecture to provide authentication and integrity. Then, we select the most suitable PQ scheme at each layer according to the features of the corresponding device (server, gateway, IoT device) and the security property (authentication, integrity). We finally carry out experiments on our selection and provide an architectural model for making IoT communication and interaction PQ secure.

Keywords: Post-quantum Cryptography, Authentication, Integrity, Internet of Things.

1 Introduction

Internet of Things (IoT) refers to the use of connected devices for various purposes such as logistics, transport, industrial and domestic applications. IoT devices are already prevalent worldwide, and their number is forecast to increase to 22 billion by 2025[1]. As the use of IoT devices continues to grow,

[1] https://iot-analytics.com/number-connected-iot-devices/ (Accessed 22/03/24)

one major concern is to secure communication between these devices to provide assurance that exchanged information has not been altered or come from fraudulent entities.

The resources allocated to IoT devices are often very constrained. IoT devices often run at 8-24MHz, with 16-32KB of Flash and 8-16KB of RAM[2]. In comparison, an average laptop's CPU is around 100 times faster, the flash memory is around 10,000 times larger, and there is almost 1 million times more RAM. The above observation will be an important challenge in the future. Quantum computers, which have been planned to be available to a wide audience in a few decades, will break our current Internet security. To overcome those threats, new cryptographic schemes, which are quantum resistant, must be deployed. Nevertheless, those schemes are more cumbersome than the ones currently used, making their successful implementation in IoT a concern.

1.1 Problem Statement

Digital Signature (DS) schemes are used to address two main security considerations, *integrity* and *authentication*. Integrity is having assurance that data has not been modified in transit by an unauthorized device. Authentication is receiving assurance that the devices that a party is communicating with are who they claim to be.

As we look into the future, one major security concern is the threat of quantum computers, which are expected to break current DS schemes and other public-key cryptographic standards in the next 15-20 years [31], thus making Internet communication no longer secure [45]. Hence, new standards must be created to ensure secure communication in the future. These new schemes, called *Post-Quantum* (PQ) cryptosystems, need to provide security against quantum attacks for devices that rely on classical computing.

When assessing PQ cryptosystems, IoT devices require special consideration. Indeed, all proposed standards for security against quantum computing bring extra overhead in terms of computation, communication and storage [37]. On the other hand, the resources allocated to IoT devices are often very constrained. In addition, the management of underlying IoT networks, made of heterogeneous devices, is complex. Therefore, PQ schemes that might be suitable for scenarios such as communication between desktop computers and servers, can be impractical or even infeasible for use in an IoT context. In this paper, we will evaluate and analyze PQ DS schemes, and compare them with cryptosystems deployed nowadays, in order to select the most suitable ones, if any, for use in IoT.

1.2 Contributions

We propose a quantum resistant IoT architecture, made of three distinct layers represented by a server, a gateway and an IoT device, respectively. We aim to

[2]https://csrc.nist.gov/CSRC/media/Events/third-pqc-standardization-conference/documents/accepted-papers/atkins-requirements-pqc-iot-pqc2021.pdf (Accessed on 22/03/24)

integrate PQ authentication and integrity at all layers in our IoT framework. We first perform experiments evaluating the computation, communication, and storage costs of implementing PQ DS schemes selected for standardization, along with current DS schemes (serving as a baseline reference). From the above results, we identify the most suitable DS scheme for each party in our IoT architecture, according to the underlying security consideration (either authentication or integrity). Based on our scheme selection, we then outline the PQ architectural model of interaction by carrying out tests on each party.

1.3 Roadmap

In Section 2, we detail the necessary background for our paper. In Section 3, we present our IoT architecture. In Section 4, we test existing PQ DS schemes and compare them with current DS schemes. We also select the most appropriate PQ schemes based on the underlying party (server, gateway or IoT device) and security property (authentication or integrity). In Section 5, we test the selected schemes on their corresponding party. In Section 6, we discuss the state of the art related to PQ authentication and integrity in IoT. In Section 7, we conclude our paper and suggest possible areas for future work.

2 Background

2.1 Digital Signatures

A DS scheme consists of three probabilistic polynomial time algorithms, denoted as KeyGen, Sign and Verify, such that:

- $\mathsf{KeyGen}(\lambda) \rightarrow (pk, sk)$ is an algorithm that takes a security parameter λ, and outputs the pair of public key pk and private key sk.
- $\mathsf{Sign}(sk, m) \rightarrow \sigma$ is an algorithm taking the private key sk and the to-be-signed message m and outputs a signature σ.
- $\mathsf{Verify}(pk, m, \sigma) \rightarrow \{Accept, Reject\}$ is an algorithm taking the message m, its signature σ along with the supposed signing public key pk and either outputs $Accept$ if σ is a valid signature for m with pk or $Reject$ otherwise.

Correctness If $(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda)$ and $\sigma \leftarrow \mathsf{Sign}(m, sk)$ for a given message m, then $\mathsf{Verify}(m, \sigma, pk) \rightarrow Accept$.

Unforgeability We are interested in DS schemes that are existentially unforgeable under a chosen-message attack [15]. Informally, let an adversary get access to signatures for messages that she chooses. This adversary must not be able to create a valid signature for a new message. When unforgeability is assured, DS schemes can be used to provide assurance of authentication as well as data integrity [1].

2.2 Authentication

Authentication is achieved by establishing the identity of the communicating parties. This can be done one-way, such that only one party verifies the other, or mutually, that is the two communicating parties verify each other's identity. Authentication can be achieved using digital certificates, as on the Internet [9]. A certificate links a public key to its owner by having a trusted Certification Authority (CA) sign the contents of the certificate (including the public key and identifying information such as the name of the owner).

Let Alice want to authenticate Bob. She receives his digital certificate, and as long as the CA can be trusted, she can authenticate Bob by verifying the signature attached to the certificate. For mutual authentication, Bob will also receive Alice's digital certificate and check the embedded signature.

Nowadays, the management of these keys and certificates is carried out by a Public-Key Infrastructure (PKI). The most commonly used certificate is X.509, which includes information such as a unique ID for the certificate, the underlying DS scheme, the ID of the CA that issued the certificate, the ID of the certificate owner, and their public key [2].

2.3 Integrity

Integrity is achieved through the signing of the messages that are sent between two parties. Digital signatures are generated for specific messages: if the message is modified in transit, then the signature becomes invalid. The party that receives the signature and message can use the public key of the signer to verify the signature.

Let Alice receive a message-signature pair from Bob. She can use Bob's public verification key to check the validity of the signature, assuring that the received message is indeed the one that Bob sent.

2.4 Post-Quantum Cryptography

PQ cryptography includes DS schemes and Key Encapsulation Mechanisms (KEMs). DS schemes have been described in Section 2.1. KEMs are used for achieving key agreement as follows: a key is generated and then encapsulated (encryption) with the other party's public key. The encapsulated key is then sent to the other party, which decapsulates the key (decryption) using their private key. This results in a shared secret key between the two parties [10].

There are five main types of PQ cryptosystems, based on the following: lattices, codes, multivariate polynomials, isogenies and hash functions. The two PQ constructions that have been used in the DS schemes we will test are lattices and hash functions. There have been other proposed PQ cryptographic schemes that have not fallen under these categories. However, there is not as much confidence in these cryptosystems due to a lack of research [13].

Lattice-based cryptosystems use a lattice: a set of points periodically structured in n-dimensional space. Lattice-based problems are hard and hence pro-

vide security. These problems are also comparatively simple, parallelizable and efficient to implement [30].

Hash-based schemes can only be used for PQ DS schemes. Security is provided based on the collision resistance of the hash function. However, the number of signatures that can be produced with these PQ schemes is limited, unless the size of the signatures is increased [12].

Standardization The U.S. National Institute of Standards and Technology (NIST) is responsible for providing standards and support for technological advancements all around the world. The NIST began a competition in 2016 for proposing PQ cryptographic standards.

NIST announced the selected PQ KEM and DS standards in 2022[3]. In this paper, we only focus on authentication and integrity, and hence, on DS schemes. For our experiments, we have tested all PQ DS schemes cryptosystems chosen by NIST.

3 Overview

3.1 IoT Architecture

IoT architectures are not standardized, so the design and implementation of architectures appropriate for IoT devices is challenging. Nonetheless, a 3-layer architecture is one of the most popular architectures [20] and was thus chosen as the architecture we follow. This architecture is like a triangle where there is one server (at the top) connected to many gateways (in the middle), and each gateway is connected to multiple IoT devices (at the bottom).

The bottom layer is the IoT device layer. It has the highest number of devices, from tens to hundreds of devices [25], and they all only communicate with a dedicated gateway. The devices in this layer are mainly sensors whose resources are noticeably limited, so they are focused on simple operations such as data collection.

The middle layer is a gateway that communicates with the server and multiple IoT devices. The gateway is expected to be less constrained than the IoT devices but more constrained than the server. The gateway will receive messages from the IoT devices that are connected to it. Then, it will collate and pass those messages to the upper layer for further data analysis.

The top layer is the central server where most processing and storage occur. The server can be seen as a powerful party with large resources. It will be in communication with the gateways in the network. The server makes decisions based on information received from gateways, and identifies if changes need to be made to the operations in the IoT system. The data processing and analysis done by the server is made available in order to let the network benefit from them.

[3]https://csrc.nist.gov/projects/post-quantum-cryptography (Accessed 15/02/24)

Use Case IoT technologies can be used in agricultural environments, in particular for monitoring nutrients, water, fertilizer and pesticide usage, and pH levels. Areas that rely on long-term data collection such as production and crop-cycle prediction are of interest for smart agriculture. Indeed, monitoring is facilitated by sensors and effective data collection improves the accuracy of predictions [4]. Another example is through supervising livestock, enabling enhanced productivity and better animal welfare [19, 43].

We aim to deploy our 3-layer architecture in such a smart agriculture context. At the bottom, multiple sensors are scattered across the agricultural field, for data collection and monitoring. In the middle, a gateway links the sensors to the server representing the local farm's network, by processing and broadcasting data collected by sensors. At the top, the server supervises the local network and connects it with the Internet. Further data processing and analysis are done by the server and made available to the network to accurately adjust to the best farming practices.

3.2 IoT Authentication

The cost and complexity of implementing certificates is a major constraint for IoT PKI [14]. This constraint has been addressed through moving computation for mutual authentication to more powerful devices [26] or replacing CAs to local actors [14]. Moreover, the size of the public key and signature has a large impact on the size of the certificates, and hence, on the related communication cost [38]. Therefore, we must take into account these different elements to identify an authentication protocol enabling an effective, yet local and distributed, PKI management.

Having one-way authentication leaves the IoT network open to attacks such as device forgery and impersonation [3, 29, 35]. Hence, mutual authentication is important in IoT networks to ensure that each device knows who they are communicating with. We benefit from the implicit hierarchy in our 3-layer architecture to generate and deliver digital certificates. At the top, an external CA, similar to the one found in the Internet, authenticates the server since the latter acts as a bridge between the local IoT network and the Internet. At the middle layer, the server plays the role of a Registration Authority (RA) for the network it represents, and thus for the gateways. It will first authenticate the gateway by creating and signing a local certificate. In return, the gateway can authenticate the server by checking the signature of the CA on the server's certificate. Then, the gateway generates and signs a certificate for each IoT device connected to it. On the other hand, those devices can authenticate the gateway by verifying the signature of the server on the gateway's certificate.

Within our hierarchical architecture, certificate generation begins at the top (i.e. server). In particular, the root CA generates the certificate for the server, which then generates the certificate for the gateway, which then generates the certificate for the IoT devices. This same hierarchy is used when updating certificates. We illustrate the mutual authentication process within our 3-layer architecture in Fig. 1 and 2.

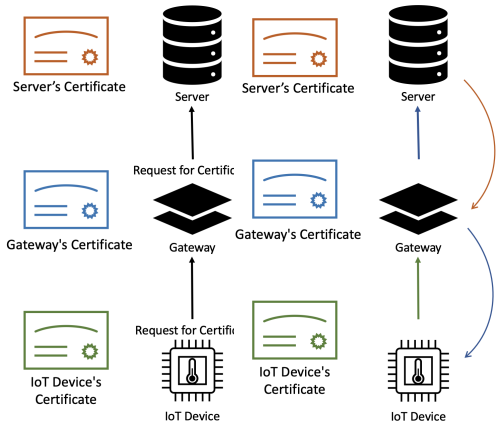

Figure 1: Certifi-Figure 2: Certificate Generation cate Verification

Use Case Following our smart agricultural example, the server, acting as a RA, leads the whole farm network. It registers every gateway in this network, by confirming their identity through certificate deployment. For instance, each gateway represents a paddock where multiple sensors are disseminated. Gateways also authenticate the server by requesting its certificate. Once mutual authentication has happened between the server and gateway, the latter, also acting as a RA, registers every sensor connected to it, by confirming its identity through certificate deployment. In return, sensors also authenticate their gateway by requesting its certificate.

3.3 IoT Integrity

Once authentication is established, the parties in our IoT model are required to sign and verify signatures to provide integrity at all layers. At the bottom layer, each IoT device senses and collects data, which is then forwarded to the gateway for further analysis. Hence, the main task of the IoT device is to sign messages containing collected data, and send message-signature pairs to the gateway. At the middle layer, the gateway often receives messages from IoT devices, along with sending messages to the server, at a lower frequency, for further data processing and analysis. The main tasks of the gateway are to verify devices' signatures and to sign and send messages to the server. At the upper layer, the server receives messages from the gateway. Hence, while the resources of the server are not a concern, our main focus for this remains signature verification.

Use Case In a smart agriculture context, the sensors on the field must sign their collected data, prior to sending them to the gateway. The gateway first collates the received information and if valid, it groups it. It then signs the

Figure 3: Performance of DS w.r.t. Number of Clock Cycles

grouped data and sends it to the server for further analysis and storage.

4 Preliminary Testing

To achieve effective authentication and integrity in our IoT hierarchical design, we need to assess PQ DS schemes and compare them with existing ones as a baseline reference. Therefore, we start by carrying out tests of several DS schemes, both classic and PQ. The U.S. NIST has two standardized classic DS schemes that are currently used for Internet communication, extending the original Digital Signature Algorithm (DSA) which is no longer considered secure [1]. NIST has also drafted standards for three PQ DS schemes[4]. The five tested DS schemes are the following:

- *ECDSA* extends DSA to groups built from elliptic curves, whose security relies on the discrete logarithm problem [1].
- *RSA DSA* extends DSA with the RSA algorithm, which is based on prime number factorization [1].
- *CRYSTALS-Dilithium* (Dilithium) is a lattice-based DS scheme, based on the Learning With Errors (LWE) and Short Integer Solution (SIS) problems [42].

[4]https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022 (Accessed on 17/04/24)

- *Falcon* is based on the SIS problem, over Nth degree Truncated Polynomial Ring Unit (NTRU) lattices [41].
- *SPHINCS+* is a stateless hash-based DS scheme that provides security based on the collision resistance of the underlying hash function [6].

NIST has used five security levels as benchmarks for the proposed PQ schemes[5]. Security levels 1, 3 and 5 correspond to being as difficult to break as a block cipher with 128-, 192- and 256-bit keys. Security levels 2 and 4 correspond to being as difficult to break as 256- and 384-bit hash functions. We compare the above PQ schemes with classic DS schemes as a baseline reference, w.r.t. their security levels.

Dilithium can be run at security level 2 (Dilithium2), 3 (Dilithium3) and 5 (Dilithium5). SPHINCs+ can be run at security level 1 (SPHINCS+128s), 3 (SPHINCS+192s) and 5 (SPHINCS+256s). Falcon can be run at security level 1 (Falcon-512) and 5 (Falcon-1024).

4.1 Implementation

The implementation of both classic and PQ schemes is done using the supercop benchmarking tool[6]. We use liboqs[7] for the implementation of the PQ DS schemes. We ran PQ DS schemes with all of their available constructions (i.e. at different security levels). Note that SPHINCS+ can be used with different hash functions. Therefore, we choose to run SPHINCS+ with the SHAKE hash function since it results in smaller signature sizes, which are the biggest limiting factor [6]. The number of calculations that occurred in 3 seconds was recorded for each step of the DS scheme, i.e. key generation, signing and verification, for messages of size 100B. Tests are run on an HP computer running Linux with an Intel Core i5 at 2.3GHz and 8GB of RAM.

We focus on computational, communication and storage costs, w.r.t. the completion of all the steps of the tested DS schemes. In particular, we consider the time taken and the memory usage to assess the computation overhead. Communication cost is the overhead caused by exchanging the DS-related data among participating parties. For instance, we evaluate the impact of the sizes of the components (e.g. signature) when broadcasting them. Storage cost is the overhead caused by the sizes of the components that must be stored by the parties to proceed with signing and verifying (e.g. public and private keys).

Scheme	Public Key	Private Key	Signature
ECDSA	132	198	132
RSA	256	2048	256
SPHINCS+	64	128	29,792
Falcon	1,793	2,305	1,280
Dilithium	2,592	4,864	4,595

Table 1: Component sizes (B) of DS at security level 5

4.2 Results

4.2.1 Component Size

The sizes of the components of the different DS schemes at security level 5 can be seen in Table 1. Overall, the classic ECDSA and RSA schemes have much smaller component sizes than their PQ counterparts. Falcon and Dilithium have components one order of magnitude larger than the classic schemes. SPHINCS+ has a comparable key size but a signature two orders of magnitude larger than the classic schemes. The storage costs incurred by signing and verifying, i.e. storing private and public keys, will be the least for SPHINCS+, followed by Falcon and Dilithium.

A digital certificate includes identifying information about the certificate owner, the owner's public key, and the CA's signature. While the amount of identifying information remains constant, the sizes of the signature and key depend on the underlying DS scheme. Using a PQ DS scheme noticeably increases the component size, and so the certificate size. Compared to ECDSA, the Falcon pair of public key and signature is 15 times larger, the Dilithium pair is 35 times larger, and the SPHINCS+ pair is 150 times larger. A current X.509 certificate using RSA is around 1.5KB [22], while it reaches 4KB using Falcon (2.6 times larger), 8KB using Dilithium (5 times larger) and up to 31KB using SPHINCS+ (20 times larger). The X.509 certificate sizes for classic and PQ DS schemes can be seen in Fig. 4.

4.2.2 Computational Cost

Comparison with Classic DS Preliminary results related to the clock cycles needed for signing and verification are shown in Fig. 3 in their log form. A clock cycle is one complete CPU operation. Therefore, the number of clock cycles indicates the amount of computation and time. We notice that SPHINCS+ requires a higher cycle count for both signing and verification. Dilithium has a smaller cycle count needed for both steps than any other scheme, including the current DSA-based standards; however, RSA still has slightly faster verification.

[5] https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization/Evaluation-Criteria/Security-(Evaluation-Criteria) (Accessed 11/02/24)

[6] https://bench.cr.yp.to/supercop.html (Accessed on 14/04/24)

[7] https://openquantumsafe.org/liboqs/ (Accessed 14/02/24)

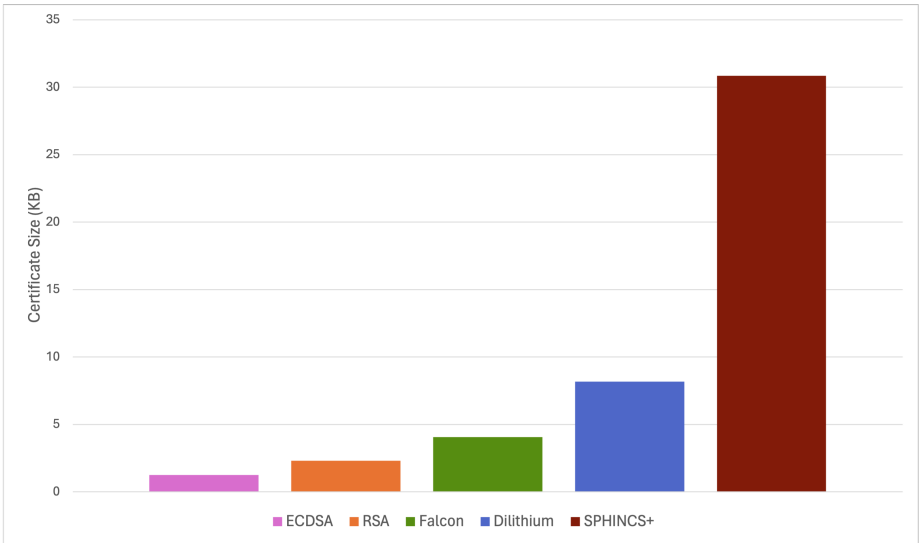

Figure 4: Sizes of One X.509 Certificate with Different DS Schemes

Falcon has a greater cycle count than Dilithium for signing but has a similar cycle count for verification. The only PQ DS scheme with a marked increase in clock cycles is thus SPHINCS+. Successfully, Dilithium and Falcon have either a smaller or comparable number of clock cycles at all security levels to the classic schemes currently used.

Time Taken for Key Generation The time taken for key generation is shown in Fig. 5. We see that Dilithium, at all security levels, is much faster than the other schemes. Timing results are 0.038ms, 0.063ms, and 0.111ms for security level 2, 3 and 5, respectively, compared to 9.5ms and 26.4ms for Falcon at security level 1 and 5, and 51.1ms, 75.7ms, and 49.5ms. It is interesting to see that level 5 is faster but matches what has been seen by other benchmarks[8]. This makes Dilithium at security level 2 250 times faster than Falcon and over 1,000 times faster than SPHINCS+, both at level 1. Dilithium is also 240 times faster than Falcon and 450 times faster than SPHINCS+ at the highest security level.

Time Taken for Signature Generation The time taken for signing, as shown in Fig. 6, is relatively similar for Dilithium and Falcon, with Falcon taking slightly longer and signing with SPHINCS+ taking the longest. Signing takes 0.102ms, 0.166ms and 0.211ms for Dilithium at level 2, 3 and 5, respectively. Falcon takes 0.338ms at level 1 and 0.6668ms at level 5. SPHINCS+ takes 388ms at level 1 and 672ms at both levels 3 and 5. Interestingly, in the SPHINCS+

[8]https://bench.cr.yp.to/primitives-sign.html (Accessed on 29/04/24)

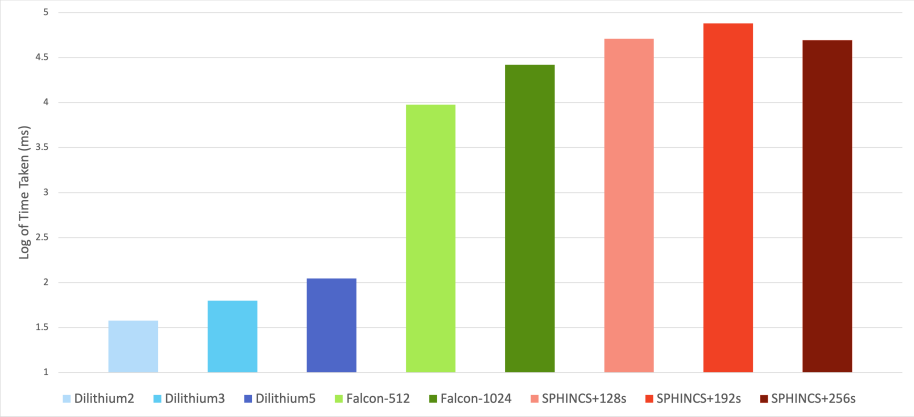

Figure 5: Time Taken for Key Generation

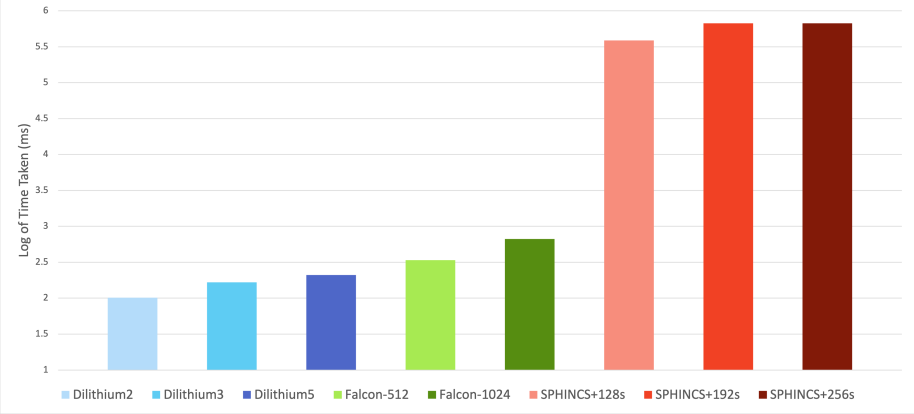

Figure 6: Time Taken for Signature Generation

submission package [6], the authors showed that signing is slightly faster with SPHINCS+256 than with SPHINCS+192, whereas we found the values to be basically the same for the two versions. Falcon is generally 3 times slower than Dilithium but 1,000 times faster than SPHINCS+ at all security levels.

Time Taken for Signature Verification Signature verification time, as seen in Fig. 7, shows that Dilithium is generally the fastest algorithm. However, Falcon at level 1 is faster than Dilithium at level 2. Again, SPHINCS+ is the slowest scheme. Verification with Dilithium at level 2 takes 0.036ms, at level 3 0.064ms, and at level 5 0.097ms. Time taken for verifying with Falcon takes 0.058ms and 0.115ms for level 1 and 5, respectively, and with SPHINCS+ is 0.564ms, 0.739ms and 1.10ms for security level 1, 3 and 5, respectively. Falcon is generally 10 times faster than SPHINCS+, but between 1.2 and 1.5 times

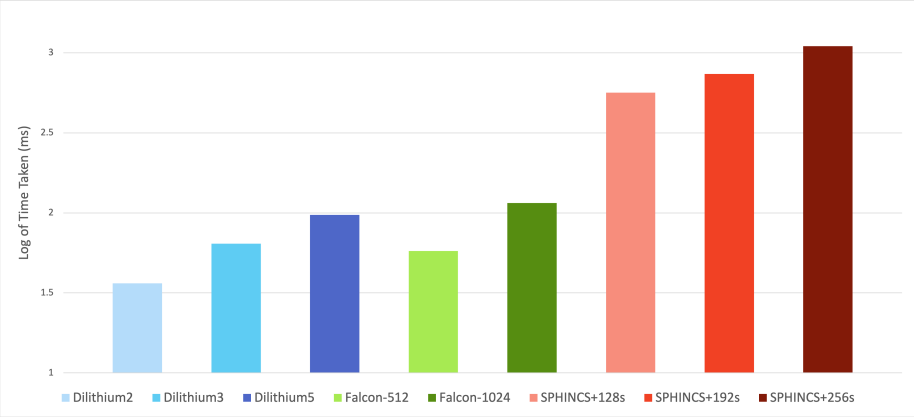

Figure 7: Time Taken for Signature Verification

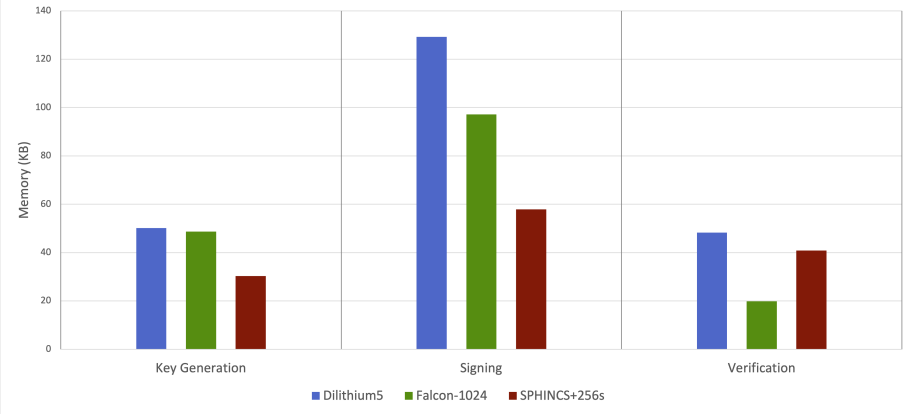

Figure 8: Memory Used at Level 5

slower than Dilithium.

Memory Usage We report the memory used at all steps (key generation, signing and verification) at security level 5 in Fig. 8. The memory required for key generation is the least for SPHINCS+. Dilithium requires nearly 20KB of extra memory and Falcon needs only slightly less memory for that step. For signing, SPHINCS+ also requires the least memory, followed by Falcon and Dilithium. Finally, for verification, Falcon requires the least memory. SPHINCS+ requires around twice as much memory and Dilithium slightly more than that.

Private Key / Public Key Pair Specification	Private Key	Public Key	Total
SPHINCS+ / SPHINCS+	128	64	192
SPHINCS+ / Falcon	128	1793	1921
SPHINCS+ / Dilithium	128	2592	2720
Falcon / SPHINCS+	2305	64	2369
Falcon / Falcon	2305	1793	4098
Falcon / Dilithium	2305	2592	4897
Dilithium / SPHINCS+	4864	64	4928
Dilithium / Falcon	4864	1793	6657
Dilithium / Dilithium	4864	2592	7456

Table 2: Storage Costs (B) at Security Level 5

4.2.3 Storage Cost

To securely and successfully communicate with each other, parties must store their own private key, along with the public keys of all other parties in the IoT network. Therefore, it is important to evaluate and analyse the storage cost implied by the private and public key sizes. In Table 2, the size (in Bytes) of combining private and public keys is given for all PQ DS schemes. We highlight in red the worst cost and in green the best cost. SPHINCS+ is the best scheme since the key pairs where at least one of them is from SPHINCS+ have smaller storage costs than any other pairs. Falcon key pairs are one order of magnitude larger than SPHINCS+ pairs but about half the size of Dilithium pairs.

4.2.4 Communication Cost

In Table 3, we evaluate the information (in Bytes) that must be communicated for each PQ DS scheme. Specifically, the communication overhead is made up of the signature and the public key that must be exchanged. We highlight in red the worst cost and in green the best cost. Falcon has the smallest amount of information, followed by Dilithium and SPHINCS+. Dilithium's communication overhead is just over twice as large as Falcon's overhead, and SPHINCS+ has an overhead that is about four times larger than Dilithium.

4.3 Scheme Selection

Here, we discuss the above results in order to make the best scheme choices for our PQ implementation of our IoT 3-layer model of interaction.

4.3.1 Authentication

We start by focusing on the incurred communication costs since they depend on the size of the certificate (which embeds a public key and a signature) that needs to be exchanged, and thus on the underlying DS scheme that is used to generate

Scheme	Public Key	Signature	Total
Dilithium2	1312	2420	3732
Dilithium3	1952	3293	5245
Dilithium5	2592	4595	7187
Falcon-512	897	666	1563
Falcon-1024	1793	1280	3073
SPHINCS+128s	32	7856	7888
SPHINCS+192s	48	16224	16272
SPHINCS+256s	64	29792	29856

Table 3: Communication Costs (B)

the certificate signature. As SPHINCS+ has significantly larger signatures than any other PQ DS scheme, the resulting communication overhead makes it the least interesting choice for our IoT use case.

Falcon uses the smallest combination of PQ public key and signature. Therefore, we would use Falcon for certificate generation for the server and gateway. The slight increase in time would be compensated by the decrease in parameter size and memory usage required. A certificate using Falcon needs 1.6KB and 3.1KB of space for the public key and signature, for security level 1 and 5, respectively.

However, due to the need for double precision floating-point values, most IoT devices cannot accept Falcon. Therefore, the best option for IoT devices is to choose Dilithium. This means that the certificate will include a Dilithium signature and public key which will be 3.7KB, 5.2KB and 7.2KB depending on the security level 2, 3 and 5, respectively.

4.3.2 Integrity

We start by focusing on the communication overhead which involves the addition of a signature to each message sent. Regarding computing costs at the bottom IoT layer, the focus is on minimizing the time taken and memory usage when signing. We discard SPHINCS+ since this scheme has the longest timings at all steps. On the other hand, the shortest times are found with Dilithium. Falcon shows better memory usage results than Dilithium. Nevertheless, as already mentioned previously, floating-points are used in Falcon, making it impractical for use by IoT devices. Hence, Dilithium is the best choice for that layer. Dilithium also requires the least amount of clock cycles to complete signature generation, which is an important consideration when trying to minimize the power consumption of running such a scheme on these IoT devices. Moreover, at the gateway, the focus is on minimizing time and memory usage when verifying signature and message pairs. Dilithium-based verification is well suited for the middle layer.

Both the gateway and server are more powerful parties than IoT devices, and can thus handle calculations with floating-point values. Falcon shows a good

Security	Party	Scheme	Consideration
Authentication	Server	Falcon	Communication
	Gateway	Falcon	Communication
	IoT Device	Dilithium	Computation
Integrity	Gateway-Server	Falcon	Communication
	IoT Device-Gateway	Dilithium	Computation

Table 4: Scheme Selection

trade-off between timing and memory costs, even if Falcon is slightly slower than Dilithium and requires more memory than SPHINCS+. Consequently, Falcon is a good choice for securing interactions between the gateway and server.

4.3.3 Summary

In Table 4, we summarise our PQ scheme selection. Depending on the context (e.g. authentication and integrity in IoT), various factors such as communication and computation costs need to be considered. In our testing, the server and gateway are mainly impacted by communication overheads. Thus, we have chosen Falcon due to smaller component sizes. The IoT device is mostly impacted by computation costs. Therefore, we have chosen Dilithium since it requires the least amount of computation. We have not selected SPHINCS+ due to the large signature sizes and time required for computation. However, if keeping the memory footprint as low as possible is the top priority, then SPHINCS+ would be the most suitable choice.

To conclude, we propose to use a mixed-certificate chain for PQ authentication [34]. In doing so, we cleverly mix the PQ schemes at different layers to reduce the size of the certificates. Similarly, for integrity, we combine different PQ DS schemes to benefit as much as possible of their pros while minimizing their cons according to parties' specific features.

5 Main Testing

In this section, we implement and evaluate the selected PQ schemes following our IoT 3-layer architecture.

5.1 Implementation

To represent our IoT device in our hierarchical architecture, we run the implementation on an Arduino Due with an AT91SAM3X8E processor, running at 84MHz with 96KB of RAM and 512KB of flash memory. To represent our gateway, we choose the Raspberry Pi (RPi) 4 Model B with a Quad Core ARM64 Cortex-A72 processor, running at 1.5GHz with 1GB of RAM. Finally, to represent our server, we do the testing on a Linux-based HP computer, with an Intel Core i5 Processor running at 2.3GHz with 8GB of RAM.

We use liboqs[9] for the implementation of the DS schemes on our computer and RPi. We use the Cortex-M implementation that was outlined in [16] for our Arduino Due. Note that an optimization is necessary to feasibly run Dilithium on the Arduino Due. This optimization has been done specifically for the instruction set used by the Arduino Due.

5.2 Results

5.2.1 Authentication

Server and gateway Generating keys on the computer using Falcon takes 9.5ms and 26.4ms at security level 1 and 5, respectively. Generating keys on the RPi using Falcon takes 19ms with 30KB of memory at level 1, and 53ms with 49KB of memory at level 5. To generate a Falcon signature for the certificate, it takes 0.338ms at level 1 and 0.667ms at level 5 on the computer, and 0.668ms at level 1 and 1.330ms at level 5 on the RPi. Note that the above steps only happen when a new certificate is generated, i.e. every 1-10 years [34], hence these overheads are acceptable. Verifying the Falcon signature of a received certificate takes 0.058ms and 0.115ms on the computer, and 0.088ms and 0.173ms on the RPi, for level 1 and 5, respectively.

Moreover, the storage capacity and memory usage on the computer are minimal in comparison to the available resources. For the RPi, the storage of around 3KB is required for the highest security level, that is manageable given the 1GB of RAM.

IoT Device Key generation using Dilithium on the Arduino Due takes 0.3ms and 0.4ms for security level 2 and 3, respectively. Note that the time taken is less than the one for the other parties (computer and RPi) running Falcon. The IoT device does not need to generate any certificates, so no signing is needed (which is the most intensive step). Verification takes 0.3ms at security level 2, and 0.4ms at security level 3. Hence, this optimization of Dilithium made for the Arduino Due is very suitable.

The IoT device stores a key pair, whose size ranges from 4KB to 7.5KB, from the lowest to the highest security level. The Arduino Due has 512KB of flash memory, thus such a storage cost should be feasible in practice.

5.2.2 Integrity

Server-Gateway The communication between the server and gateway is secured using Falcon. In particular, the main gateway's task is to sign messages while the main server's task is to verify message-signature pairs. Timings for these steps, taken by the gateway (RPi) and the server (computer) respectively, can be seen in Fig. 9. The time taken on the RPi for signing at security level 2 is nearly twice as long as signing at security level 1 (1.33ms and 0.668ms resp.). Signing requires a maximum of 4KB of storage (keys) and 97KB of memory for

[9] https://openquantumsafe.org/liboqs/ (Accessed 14/02/24)

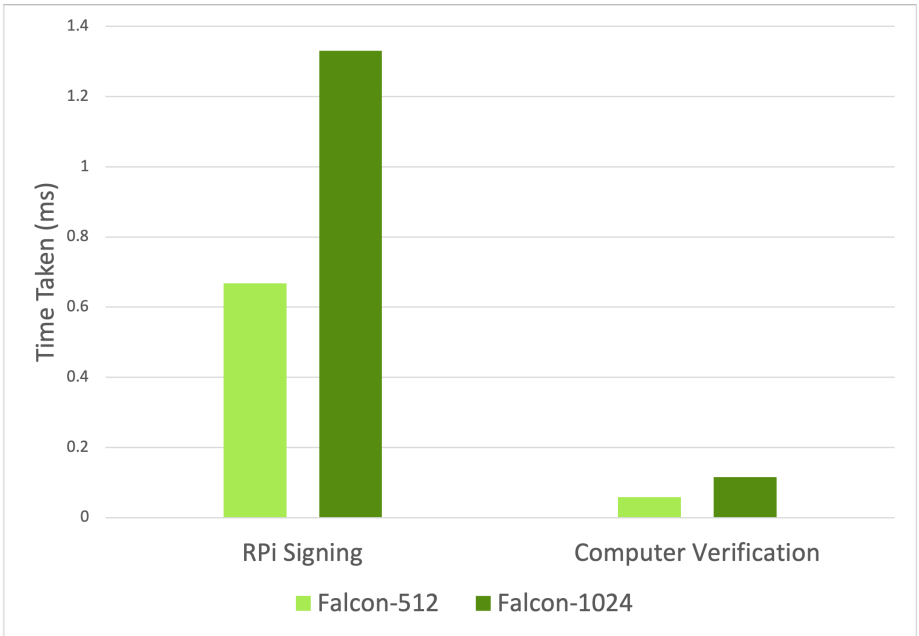

Figure 9: Server-Gateway Communication

processing. Given the 1GB of available RAM on the RPi, if signing happens moderately, say once every 30 seconds, then choosing Falcon for the gateway is reasonable. As seen in Fig. 9, Falcon's signature verification on the computer takes 0.058ms at security level 1 and 0.115ms at security level 5. The memory required for processing is just less than 20KB which is negligible for a computer.

Gateway-IoT Device Dilithium is used to secure the interactions between the gateway and IoT device. In particular, the latter mostly signs messages from collected data and the former verifies the corresponding signatures for further data processing. The time taken to sign messages on the Arduino Due is around 4ms at security levels 2 and 3, as seen in Fig. 10. Note that the optimization used for the Arduino Due (IoT device) is only available for security levels 2 and 3 due to the technical constraints of the device. For the RPi (gateway), the time taken to verify these signatures is 0.15ms and 0.26ms for levels 2 and 3, respectively. This means that the overall processing has an overhead of less than 5ms in total for each message. If the frequency of signing is less than once every second, then the observed time taken should be acceptable. Moreover, we found that the size of the message did not affect the timing to generate the signature.

The memory required for signing on the Arduino Due optimization takes 10.5KB for security level 2 and 11.5KB for security level 3. This places the stack size required to run Dilithium at just over 10% and 12% of the available RAM

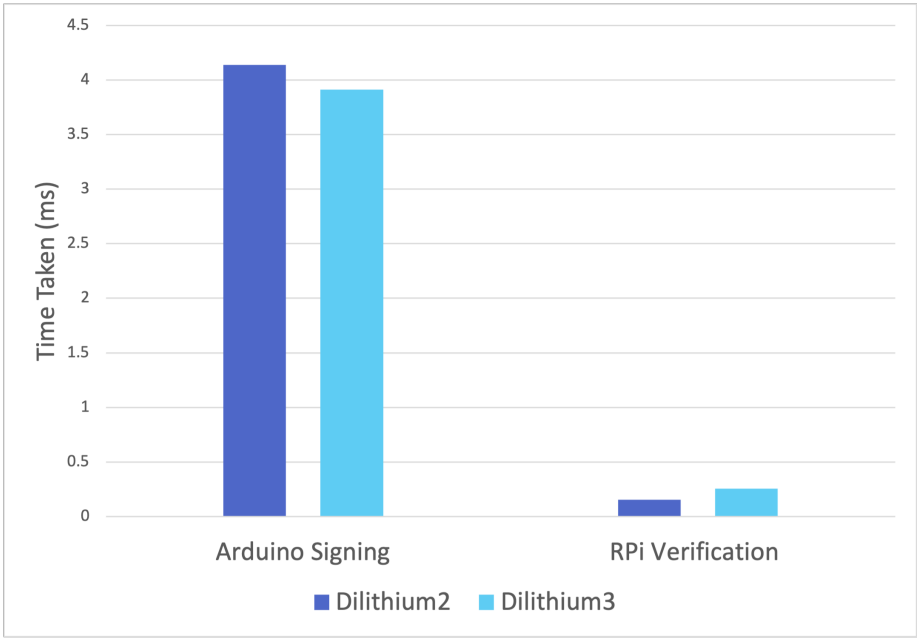

Figure 10: Gateway-IoT Device Communicationn

on the Arduino Due, respectively. The Arduino Due also needs to store a copy of the IoT device's private key in order to generate signatures, which requires 2.5KB and 4.0KB of storage for levels 2 and 3, respectively. The memory required for verification on the RPi is 32KB for security level 2 and 39KB for security level 3. Given the available 1GB of RAM, such a cost should not cause problems.

5.3 Discussion

Authentication We recall that we choose Dilithium for the IoT device since its execution is the fastest among all schemes. Our experiments demonstrate that its optimization is computationally fast on the Arduino Due, even more than the competing Falcon running its original version on the more powerful RPi and computer.

We chose to deploy a mixed-certificate chain for authentication. The size of the parameters (keys, signatures), and thus of the certificates, decreases thanks to the combination of Falcon and Dilithium (rather than using Dilithium only). One may suggest to use Dilithium alone since this scheme is the best one regarding computational costs. However, certificate generation only happens once while certificate storage remains throughout the lifespan of all involved parties. Therefore, minimizing certificate storage costs is of greater concern, motivating our choice of a mixed-certificate chain.

Integrity Regarding the IoT device-gateway interaction (Dilithium), communication and storage costs remain substantial while computational costs are moderate. Regarding the gateway-server interaction (Falcon), the computation costs are noticeable but the communication and storage costs stay reasonable. Nevertheless, to provide robust security against quantum attacks, additional overheads must be accepted. Indeed, both Falcon and Dilithium result in parameter sizes that are an order of magnitude larger than the ones from current standards. Note that those extra costs may have an apparent impact on the IoT device (Arduino Due) but less on the gateway and server since they are more powerful.

To guarantee integrity, we propose to embed two schemes, Dilithium and Falcon, in our 3-layer architecture. Consequently, the gateway stores two pairs of public and private keys. At the highest security level, a total of 11.5KB is needed, which is negligible compared to the 1GB storage available on the RPi. On the other side, the server and IoT device store one pair of keys only. The server requires 4KB from the 8GB of available storage, which is negligible. The IoT device requires around 1.5% of its available storage (i.e. 7.5KB out of 512KB), which is acceptable.

Use Case We now explore the way to deploy our PQ secure 3-layer model of interaction in the context of smart agriculture. We assume that certificates are valid for two years. Consequently, we choose to run Falcon and Dilithium at their highest security level, namely 5 and 3 respectively. We recall that the optimization of Dilithium for the Arduino Due, and thus the IoT device, is not available at the security level 5.

Let us illustrate our use case with a specific example. A small farm is composed of five fields used to grow crops. Each field is managed by a gateway, and the whole farm by an unique server. Each field, thus gateway, has eight sensors These devices are used to measure, among others, soil, temperature, humidity and atmospheric data. We assume that sensors send updates to the gateway regularly, say every 15 minutes. Then, the gateway processes and aggregates the updates, which results in a message which is sent to the server.

We consider that the data collected and broadcast to the gateway is low-risk. Thus, we select Dilithium at the security level 2 for the IoT device-gateway communication and Falcon at the security level 1 for the gateway-server communication. Moreover, each sensor must store 2.8KB for its private key and the gateway's public key. Each gateway needs around 15KB to store both its Dilithium and Falcon private keys, the eight IoT devices and server's public keys. Finally, the server keeps around 5.8KB for its private key and the five gateways' public keys. From the above observations, one concern is the storage costs since keys must be stored long term for each interaction. The gateway has higher storage costs compared to the other two parties, because of the number of connections (eight devices per gateway). However, considering the storage available on the RPi, the costs represent less than 0.0002%.

We also found that the time needed to sign messages (around 4ms for the

IoT device and less than 1.5ms for the gateway) was reasonable. In our smart agriculture scenario, we assume that communication occurs periodically once every 15 minutes, hence the above signing time has a low impact on the overall system.

Therefore, our experiments demonstrate that our proposed 3-layer architecture with PQ integrity and authentication is suitable in the case of smart agriculture.

6 Related Work

Communication Protocol Existing work has been done to achieve PQ authentication and integrity in a specific IoT communication protocol, namely the Datagram TLS (DTLS) protocol. Both [39] and [33] focus on developing a lightweight PQ communication interaction over DTLS. However, the authors consider the schemes NTRU and qTESLA, which have not been selected for standardization by NIST. In [28], the authors present a full PQ DTLS handshake protocol. They identify the best NIST finalist PQ schemes to be used by IoT devices and evaluate the associated costs. However, none of these papers consider an IoT 3-layer model of interaction between distinct parties with different resources.

Authentication There have been a few existing works focusing on transitioning authentication to the PQ era. In [21], the authors identify that PQ schemes have a significant impact on the costs associated with deploying X.509 certificates, but conclude that it is still feasible. Though, a similar deployment in time-sensitive or resource constrained environments would be difficult. In [36], the authors look into deploying X.509 certificates using the selected NIST finalists and find that careful considerations must be made to manage the overheads.

Combining several (PQ and non-PQ) schemes in a certificate for the transition to the PQ era was first proposed in [8] as a "hybrid" certificate. However, the authors do not consider the possible inefficiencies with the use of PQ schemes. However, this "hybrid" design was tested in [21]. While the time taken was noticeable, the authors deemed 200ms negligible. However, such a value could have a severe impact in specific contexts such as IoT.

In [34], the authors outline a protocol for "mixed-certificate chains" combining different DS schemes at the various levels of the certificate chain. They use SPHINCS+ at the root level with either ECDSA, Dilithium or Falcon used at the lower levels. This proposed protocol improves the signing time but still has substantial certificate sizes. Consequently, the storage overhead might have a negative effect in an IoT environment. It also requires the end entity to carry out multiple verifications, using the different schemes in the certificate chain. In [27], the authors propose a lightweight option for a PQ certificate chain by using a combination of XMSS and Dilithium. However, XMSS is not a NIST finalist.

Integrity In [40], the authors assess the suitability of PQ DS schemes, including those selected for standardization by NIST, in IoT by running them on different processors. However, those experiments are collated from external sources which used different experimental setups, making the comparison difficult. In [44], the authors evaluate the feasibility of PQ DS schemes in resource-constrained environments such as IoT. However, the evaluation remains too generic.

There have been several proposed optimized implementations of the NIST PQ DS schemes finalists for processors commonly found in IoT: Dilithium [5, 11, 16, 18, 32], Falcon [5, 23, 24], and SPHINCS+ [7, 17]. These optimizations generally improve compared to more generic implementations. However, these are specific to the given processor, thus cannot be easily deployed.

7 Conclusion and Future Work

In this paper, we developed a PQ 3-layer IoT architecture. First, we have tested all to-be-standardized PQ DS schemes to provide authentication and integrity within our model of interaction. In particular, we have considered their computation, communication and storage costs to better select them in function of the features of the party at each level (server, gateway, IoT device). Based on our selection, we conducted further tests on specific devices (computer, RPi, Arduino Due) within our IoT architecture. Our assessment revealed that Dilithium is a suitable choice for IoT devices, while Falcon performs better for more powerful entities like the gateway and server. When using a mixed-DS approach with Falcon and Dilithium, we observed a reduction in the communication overhead by about 45% compared to using a Dilithium-only approach.

Future work includes running similar tests on other resource-constrained devices, such as ARM cortex-M4 and -M3, to confirm our findings regarding Dilithium's suitability for IoT devices. Additionally, we would test Falcon on IoT devices equipped with processors that can perform floating-point calculations, such as STM32. Another avenue for future research involves utilizing a connected IoT testbed to observe the communication overhead in practice within our proposed mixed-DS 3-layer architecture.

References

[1] Digital signature standard (dss). Technical Report Federal Inf. Processing Standards Publications (FIPS PUBS) 186-5, U.S. Department of Commerce, Washington, D.C., 2023.

[2] Carlisle Adams and Steve Lloyd. *Understanding PKI: concepts, standards, and deployment considerations.* 2003.

[3] Muhammad Naveed Aman, Kee Chaing Chua, and Biplab Sikdar. Mutual authentication in iot systems using physical unclonable functions. *IEEE Internet of Things Journal*, 4(5):1327–1340, 2017.

[4] P. Anand Prabu and L. S. Jayashree. A smart agricultural model using iot, mobile, and cloud-based predictive data analytics. In *Proc. of Int. Conf. on Artificial Intelligence, Smart Grid and Smart City Applications*, pages 383–387, Cham, 2020.

[5] Luke Beckwith, Duc Tri Nguyen, and Kris Gaj. Hardware accelerators for digital signature algorithms dilithium and falcon. *IEEE Design & Test*, pages 1–1, 2023.

[6] Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, and Peter Schwabe. Sphincs+ - submission to the nist post-quantum cryptography project, 2017.

[7] Quentin Berthet, Andres Upegui, Laurent Gantel, Alexandre Duc, and Giulia Traverso. An area-efficient sphincs+ post-quantum signature coprocessor. In *2021 IEEE IPDPSW*, pages 180–187, 2021.

[8] Nina Bindel, Udyani Herath, Matthew McKague, and Douglas Stebila. Transitioning to a quantum-resistant public key infrastructure. In *Post-Quantum Cryptography: 8th Int. Workshop, PQCrypto 2017, Proc. 8*, pages 384–405, 2017.

[9] Sharon Boeyen, Stefan Santesson, Tim Polk, Russ Housley, Stephen Farrell, and David Cooper. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, 2008.

[10] Joppe Bos, Leo Ducas, Eike Kiltz, T Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Secur. and Privacy*, pages 353–367, 2018.

[11] Joppe W Bos, Joost Renes, and Daan Sprenkels. Dilithium for memory constrained devices. Cryptology ePrint Archive, Paper 2022/323, 2022.

[12] Johannes Buchmann, Erik Dahmen, and Michael Szydlo. Hash-based digital signature schemes. In *Post-Quantum Cryptography*, pages 35–93. 2009.

[13] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. Report on post-quantum cryptography. Technical Report NISTIR 8105, NIST, 2016.

[14] Jean-Guillaume Dumas, Pascal Lafourcade, Francis Melemedjian, Jean-Baptiste Orfila, and Pascal Thoniel. Localpki: An interoperable and iot friendly pki. In *E-Business and Telecommunications*, pages 224–252, 2019.

[15] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Comp.*, 17(2):281–308, 1988.

[16] Denisa O. C. Greconici, Matthias J. Kannwischer, and Amber Sprenkels. Compact dilithium implementations on cortex-m3 and cortex-m4. *IACR Trans. on Cryptographic Hardware and Embedded Systems*, 2021, Issue 1:1–24, 2020.

[17] Thomas Hanson, Qian Wang, Santosh Ghosh, Fernando Virdia, Anne Reinders, and Manoj R. Sastry. Optimization for sphincs+ using intel secure hash algorithm extensions. Cryptology ePrint Archive, Paper 2022/1726, 2022.

[18] Junhao Huang, Alexandre Adomnicăi, Jipeng Zhang, Wangchen Dai, Yao Liu, Ray CC Cheung, Çetin Kaya Koç, and Donglong Chen. Revisiting keccak and dilithium implementations on armv7-m. *IACR Trans. on Cryptographic Hardware and Embedded Systems*, 2024(2):1–24, 2024.

[19] Anitha Ilapakurti and Chandrasekar Vuppalapati. Building an iot framework for connected dairy. In *2015 IEEE 1st Int. Conf. on Big Data Comp. Service and Applications*, pages 275–285, 2015.

[20] Mohammad Ali Jabraeil Jamali, Bahareh Bahrami, Arash Heidari, Parisa Allahverdizadeh, and Farhad Norouzi. *IoT Architecture*, pages 9–31. 2020.

[21] Panos Kampanakis, Peter Panburana, Ellie Daw, and Daniel Van Geest. The viability of post-quantum X.509 certificates. Cryptology ePrint Archive, Paper 2018/063, 2018.

[22] Panos Kampanakis, Peter Panburana, Ellie Daw, and Daniel Van Geest. The viability of post-quantum x. 509 certificates. *Cryptology ePrint Archive*, 2018.

[23] Emre Karabulut and Aydin Aysu. A hardware-software co-design for the discrete gaussian sampling of falcon digital signature. Cryptology ePrint Archive, Paper 2023/908, 2023.

[24] Youngbeom Kim, Jingyo Song, and Seog Chung Seo. Accelerating falcon on armv8. *IEEE Access*, 10:44446–44460, 2022.

[25] Mr Kalpataru B Patil1 Mr Mahesh, B Girase2 Mr Vijay A Mali, Mr Sachin S Koli, and Jayvant R Saindane. Agriculture environment monitoring system using android wi-fi. *IJSRD*, 6:39–44, 2018.

[26] Francesco Marino, Corrado Moiso, and Matteo Petracca. Pkiot: A public key infrastructure for the internet of things. *Trans. on Emerging Telecommunications Technologies*, 30(10):e3681, 2019.

[27] Soundes Marzougui and Jean-Pierre Seifert. Xmss-based chain of trust. In *Proc. of 10th Int. Workshop*, volume 87, pages 66–82, 2022.

[28] Callum McLoughlin, Clémentine Gritti, and Juliet Samandari. Full post-quantum datagram tls handshake in the internet of things. In *Codes, Cryptology and Inf. Secur.*, pages 57–76, 2023.

[29] Reem Melki, Hassan N Noura, and Ali Chehab. Lightweight multi-factor mutual authentication protocol for iot devices. *Int. J. of Inf. Secur.*, 19(6):679–694, 2020.

[30] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-quantum cryptography*, pages 147–191. 2009.

[31] Michele Mosca and Marco Piani. Quantum Threat Timeline Report 2023, 2023.

[32] K. Denisse Ortega L. and Luis J. Dominguez Perez. Implementing crystal-dilithium on frdm-k64. In *2021 Ubiquitous Comp., Electronics & Mobile Commun. Conf. - UEMCON'21*, pages 178–183, 2021.

[33] Simpy Parveen, Reihaneh Safavi-Naini, and Marc Kneppers. Dtls with post-quantum secure source authentication and message integrity. In *2021 IEEE GC Wkshps*, pages 1–6, 2021.

[34] Sebastian Paul, Yulia Kuzovkova, Norman Lahr, and Ruben Niederhagen. Mixed certificate chains for the transition to post-quantum authentication in tls 1.3. Cryptology ePrint Archive, Paper 2021/1447, 2021.

[35] Sebastian Paul, Felix Schick, and Jan Seedorf. Tpm-based post-quantum cryptography: A case study on quantum-resistant and mutually authenticated tls for iot environments. In *Proc. of the 16th Int. Conf. on Availability, Re. and Secur.*, pages 1–10, 2021.

[36] Manohar Raavi, Pranav Chandramouli, Simeon Wuthier, Xiaobo Zhou, and Sang-Yoon Chang. Performance characterization of post-quantum digital certificates. In *2021 Int. Conf. on Comp. Commun. and Netw. (ICCCN)*, pages 1–9, 2021.

[37] Manohar Raavi, Simeon Wuthier, Pranav Chandramouli, Yaroslav Balytskyi, Xiaobo Zhou, and Sang-Yoon Chang. Security comparisons and performance analyses of post-quantum signature algorithms. In Kazue Sako and Nils Ole Tippenhauer, editors, *Applied Cryptography and Network Security*, pages 424–447, 2021.

[38] Javier Sanchez Guerrero, Robert Vaca Alban, Marco Guachimboza, Cristina Páez Quinde, Margarita Narváez Ríos, and Luis Alfredo Jimenez Ruiz. Cryptography applied to the internet of things. In *Teaching and Learning in a Digital World: Proc. of the 20th Int. Conf. on Interactive Collaborative Learning–Volume 1*, pages 390–398, 2018.

[39] Johanna Sepúlveda, Shiyang Liu, and Jose M Bermudo Mera. Post-quantum enabled cyber physical systems. *IEEE Embedded Systems Letters*, 11(4):106–110, 2019.

[40] Kyung-Ah Shim. On the suitability of post-quantum signature schemes for internet of things. *IEEE Internet of Things Journal*, 11(6):10648–10665, 2024.

[41] Deepraj Soni, Kanad Basu, Mohammed Nabeel, Najwa Aaraj, Marc Manzano, Ramesh Karri, Deepraj Soni, Kanad Basu, Mohammed Nabeel, Najwa Aaraj, et al. Falcon. *Hardware Architectures for Post-Quantum Digital Signature Schemes*, pages 31–41, 2021.

[42] Deepraj Soni, Kanad Basu, Mohammed Nabeel, Najwa Aaraj, Marcos Manzano, and Ramesh Karri. CRYSTALS-Dilithium. *Hardware Architectures for Post-Quantum Digital Signature Schemes*, pages 13–30, 2021.

[43] Kerry Taylor, Colin Griffith, Laurent Lefort, Raj Gaire, Michael Compton, Tim Wark, David Lamb, Greg Falzon, and Mark Trotter. Farming the web of things. *IEEE Intelligent Systems*, 28(6):12–19, 2013.

[44] Marin Vidaković and Kruno Miličević. Performance and applicability of post-quantum digital signature algorithms in resource-constrained environments. *Algorithms*, 16(11), 2023.

[45] Petros Wallden and Elham Kashefi. Cyber security in the quantum era. *Commun. of the ACM*, 62(4):120–120, 2019.