Master thesis proposal: studies of the impacts of the PQC on embedded systems/Iot

1 Small introduction

With the advancement of quantum computers, it is becoming necessary to update the cryptography of classical computers to meet the new challenges posed by these more powerful technologies.

For this purpose, the National Institute of Standards and Technology (NIST) has issued three separate calls for new cryptographic standards. In the first call, three new standards were created (FIPS: 203, 204, 205). The second focus on public-key encryption and key establishment algorithms and third calls on additional digital signature algorithms.

While all of these algorithms are the subject of much study within the scientific community, an often overlooked area is the application of these algorithms in embedded systems and the Internet of Things (IoT).

2 Work proposal

(The proposal of this thesis is based on a recent study (see [**MQTT**]) in order to produce new results. In addition, the scope of this study will be enlarged to be adapted to a master's thesis from a content point of view.)

The main goal of this thesis is to study the 13 out of 14 algorithms[1] presented during the second round of the third NIST call (additional digital signature schemes [2]), on system used in embedded systems and in IoT project. In order to be able to quantify and evaluate their usefulness to know if they are applicable within its systems.

The choice to focus on the candidates of the third call for proposals is mainly based on the fact that they are little studied in the current literature, particu-

[1] Currently the algorithm Mirath to do not have a public git repository and the site page is empty [**mirath**]

[2] `https://csrc.nist.gov/projects/pqc-dig-sig`

larly in the context of systems with limited resources and use in IoT. This lack of study is due to the fact that the second round started recently on October 24, 2024.

More concretely, this thesis will focus on the study of the algorithms of the third call of NIST relating to additional digital signature schemes. On a raspberry Pi 4 system through a case study of a new MQTT protocol intended for post-quantum cryptographic algorithms (see article [**MQTT**]). A personal implementation of this MQTT protocol will be made on the top of Eclipse Mosquitto [3].The implementation focuses mainly on what the article presents as "Security level 1" (SL1) [4] (Present quickly in the next paragraph). Once implemented,multiple experimentations will be made on a real scenario in order to be able to study and compare the algorithms currently in competition based on metrics. The data processed by the protocol will be under the assumption that integrity and authenticity are mandatory for data to be sent.

Following new MQTT protocol, the scenarios will take the following form (see figure 1), Each new publisher will create a private and public key pair. The public key will be shared with the broker.Once the publisher produce data, it will sign its data with his private key and send the signature in addition to data to the broker. Each subscriber will be able to verify the integrity and authenticity of the data thanks to the keys in their possession, that they received during the subscription of the topic.

[3] `https://mosquitto.org/`

[4] To go into technical details, the article presents two phases before the SL1, the setup and the join. In the context of this thesis, it is mainly the SL1 phase and not the setup and join that will be studied because these two phases are based mainly on key exchange algorithms and not on signature algorithms. The simplification of using a simple Diffie–Hellman key exchange will be made to simplify these steps in order to study the SL1

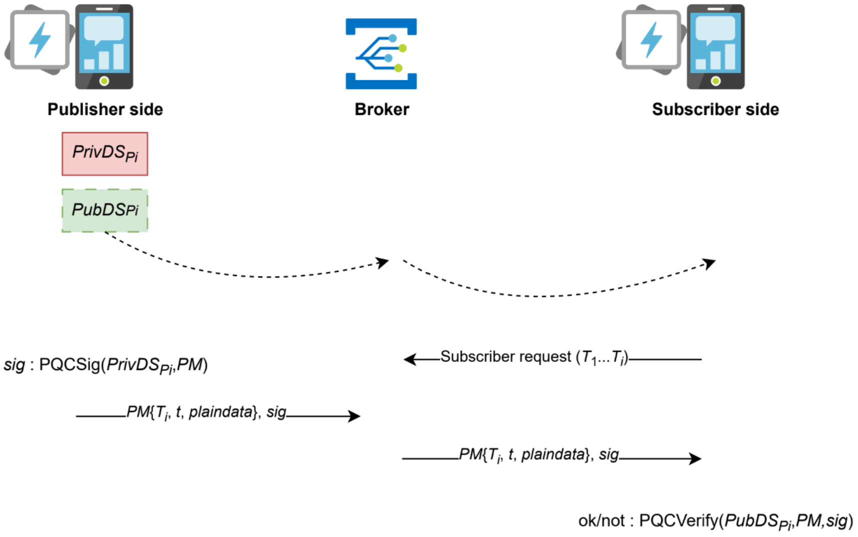
Publisher side
Broker
Subscriber side
PrivDS$_{Pi}$
PubDSPi
sig : PQCSig(PrivDS$_{Pi}$,PM)
Subscriber request (T_1...T_i)
PM{T_i, t, plaindata}, sig
PM{T_i, t, plaindata}, sig
ok/not : PQCVerify(PubDS$_{Pi}$,PM,sig)

Figure 1: New MQTT protocol : security level 1(source : [**MQTT**])

For each scenario, 3 sub-scenarios will be studied.

	Publisher	**Subscriber**
sub-scenario 1	constrained device	Normal computer
sub-scenario 2	Normal computer	constrained devices
sub-scenario 3	constrained devices	constrained devices

The measures taken to evaluate and discuss the performance of different signature algorithms are grouped into three main families.

1. **temporal** :where the time (in seconds) and the number of processor cycles will be measured.
2. **the space used** (bytes): the size of ram used to store the program, the size of the input and output of each algorithm.
3. **Network** : the number packet send by a publisher to the broker and receive by a subscriber from the broker [5]

[5] This data comes from an analysis of the state of the art, which shows that the number of packets generated and sent depends on the output of the signature algorithm. When a packet size exceeds the device's Maximum Transmission Unit (MTU), it is fragmented, which increases the number of packets. This can be problematic if the device's buffer is limited. Additionally, this issue is often overlooked in similar studies, but it can be significant, especially with protocols like CoAP, which use UDP connections.

To be clearer, a scenario in the framework of this thesis, would be for example a raspberry which has a temperature sensor (Publisher) at periodic time it sends the collected data to the broker which send it to another raspberry (Subscriber). Each data sent by the Publisher will be signed by one algorithm of the NIST and could be verified by the Subscriber. The scenario will be repeated with the different candidates (algorithm)

There will be different scenarios with different sensors. The goal of testing different sensors is to also test the size of the data that will be provided to the signing algorithms and their effects. Indeed, a temperature sensor will not provide the same amount of information as for example an infrared sensor.

Once all the experimental data is collected and they will be analyzed in order to:

1. discuss the impact (in terms of resources and time) on resource-constrained systems.
2. Compare the algorithms of the NIST "Additional Digital Signature Schemes" to the Elliptic Curve Digital Signature Algorithm (ECDSA) in order to have a point of comparison with the current state of cryptography used in embedded systems[6]. (the data of ECDSA will also be collected during the scenarios.)
3. Comparing NIST "Additional Digital Signature Schemes" candidates with each other.

Regarding the content of the thesis, in addition to the practical section mentioned above, a theoretical part will also be developed. This section will mainly explore an analysis of the fundamental problems on which digital signature algorithms are based will also be presented. In addition, an explanation of the NIST organization and how its competition on quantum cryptography works will be included. Another part will be dedicated to embedded systems, focusing on the challenges these systems face, and a presentation of the MQTT protocol.

3 Extension of work

Because I do not have sufficient experience in the topic to know if the initial idea is sufficient for a master thesis, I propose two extensions which remain in the same topic allowing to enlarge the initial scope if the need arises.

3.1 CoAP

Creation of a new scenario with the CoAP (Constrained Application Protocol) protocol instead of MQTT. The choice to include it as an extension in this study

[6] signature function often presented in the state of the art as a reference in IoT

is explained by my prior theoretical knowledge of MQTT thanks to university courses, as well as by the fact that many state-of-the-art articles devoted to the study of this protocol are currently behind a paywall, making their access difficult, and are not available on Sci-Hub.

3.2 Scenario TLS

A specific scenario which will focus on the use of TLS certificate signature verification between client and broker. (This specific scenario is inspired by the work [**TLS**]). I propose this scenario as an extension due to the fact that it seems to require some work which seems to be considerable. It mainly consists of taking a specific library such as "mbed TLS"[7] or "WolfSSL"[8] and adapting their codes so that they can run algorithms not implemented within the library.

4 State of the art

As mentioned earlier, due to the novelty of the topic, little research has been done on to compare those algorithms between them. However, according to the NIST rules for the competition (rule 2.B.6 [**NistRules**]), in the first round, participants were required to submit a paper containing information related to the "Algorithm Specifications and Supporting Documentation." This rule requires a statement listing the advantages and limitations of the cryptosystem. It is also recommended to address the ability to implement these algorithms in various environments, including constrained systems (such as smartcards, satellites, low-power or memory-constrained systems, ...)[**NistRules**].

Multiple participants explicitly discussed the capabilities of their algorithms within constraint systems or give useful information:

1. Cross claim that the random parity-check matrix they use in their algorithm can be reconstructed from a small seed using Cryptographically secure pseudorandom number generator (CSPRNG). This allows the public key to be compressed below 0.1 kB. In addition to the size of the public key and the size of the signature, their algorithm is good in the design of X.509 certificates (e.g. TLS) [**cross**].

2. Less explains that there are two limitations to constrained systems. The first is the size of their public keys, which could cause some problems. The second is that in terms of computation, the bottleneck of their algorithm is the "gaussian elimination algorithm" [**less**]

3. SQIsign offers several advantages, a fast verification, small private keys for example up to 128 bytes for high security, a small signature size up

[7] `https://www.trustedfirmware.org/projects/mbed-tls/`

[8] `https://www.wolfssl.com/`

to 335 bytes for this same security. However, a limitation is the size of the private keys which can go up to 1509 bytes. However, no information related to the capacity of use this algorithm within systems with limited resources was found in the specification document [**sqisign1**, **sqisign2**].In the scientific literature, several research projects are being developed in relation to Isogeny, we can cite article [**isogeny3**] which shows that research on SQIsign is underway to optimize it in 32-bit systems in order to improve it.

4. Hawk presents itself as "well suited for various hardware", its advantage that it highlights is that its algorithm is "free of floating-point arithmetic" rather important element because floating point arithmetic requires a Floating Point Unit (FPU) for calculation acceleration. Element that several constrained devices do not have [**hawk**].

5. MQOM does not really describe its capabilities in relation to systems with low resources. However, it does have some advantages such as low memory usage for its keys and signatures. However, it seems to be affected by several limitations including performance issues [**MQOM**].

6. Perk from the point of view of the subject treated for this thesis has only one advantage, the only positive point is the use of "little" memory (in terms of bytes) for its parameters. [**PERK**]

7. RYDE has a majors flaws, as described in its documentation, it is complex to implement in embedded systems, has slow performance for signing. However the size of the signature and the public key is small. [**RYDE**]

8. Sdith proposes an algorithm allowing small key size, a size that varies between 120 to 240 bytes favourable element in the use of X.509 certificate. In addition they explain that they have a "small code-based signatures". [**Sdith**]

9. MAYO presents itself as advantageous in terms of signature size and key used. However these advantages are quickly lost when higher security levels are reached [**MAYO**].

10. QR-UOV explains that it is difficult to implement their algorithms on constrained systems such as smartcards. Indeed, as explained, due to their large size of the public key, it is difficult to hold them in the memory of these systems. [**QR-UOV**]

11. SNOVA offers interesting advantages, small public key (4112 bytes), small signature size (376 bytes), small secret key (48 bytes) and light computing power. In addition, their algorithm propose simple arithmetic which may be useful for systems with constrained capacity of calculus. However, a notable element from the security point of view is that certain parameters of the algorithm must have certain properties in order not to generate weak keys. [**SNOVA**]

12. UOV highlights the simplicity of its algorithm which is based on linear algebra on small finite field. This element would allow an efficient implementation on ”low cost devices”[**UOV**].

13. FAEST is mainly presented as an evolution of SPHINCS+, it has a faster signature and a smaller signature in terms of byte. However in terms of signature verification it would be slower than SPHINCS+. In terms of key size it offers public keys between 32 and 64 bytes and for private keys it offers sizes ranging from 16 to 32 bytes. Its signature size for the smallest security level is 5038 bytes [**faest**].

Mirath which is the combination of MIRA and MiRitH from the first round is a special case in the context of the state of the art, it has little information about him. Currently to my knowledge no public git repository has been opened and the site page is empty [**mirath**]. So testing it may be complicated in the context of this research because the end of the second round is scheduled for January 17, 2025, so the group has no obligation to release information until that date. [**nistSecondCall**]

5 Current situation

5.1 Hardware

As for the hardware needed for the experiment, I already have all the essential components to get started. This includes several sensors, two Raspberry Pi 4s and a router to establish a private network (which allows to have a controlled environment for the experimentation).

5.2 Software

I am currently working on several libraries, including PAPI [9], to accurately retrieve the number of cycles of a process. The Mosquitto library[10] is also in the testing phase for managing MQTT communication.

[9] `https://icl.utk.edu/papi/`

[10] `https://mosquitto.org/`