Post-Quantum Cryptography Efficiency

Samuel Jacquier

School of Computer and Communication Sciences

Semester Project

January 2021

Responsible
Prof. Serge Vaudenay
EPFL / LASEC

Supervisor
Loïs
Huguenin-Dumittan
EPFL / LASEC

Contents

Abstract

In this report, we present our research on the field of the Post Quantum Cryptography. This subject is and will be a main topic of research for the Computer Science searchers as the quantum computers could appear during the next decades. In this report, we present the issue by introducing the reader to the modern cryptography techniques and protocols. Then, we explain the need of research upon the future cryptography in response of the Shor Algorithm and all its implications, presenting the mechanisms and practical projects that have been conducted in the world. Finally, we present our project and our contribution to the research : Quantumnet, a benchmarking software that allow the user to test and benchmark post quantum key exchange and signature algorithms, by simulating a network and all its characteristics. Then, we use this program to construct our own tests, that we analyzed to draw conclusions about the behavior and performance of these algorithms.

Acknowledgments

For this project, we address our thanks to the project responsible teacher, Prof. Serge Vaudenay, and to the project supervisor Loïs Huguenin-Dumittan, which has been helping us during the whole semester, proposing ideas and tools to conduct this project. The LASEC laboratory also helped a lot by renting servers to compute all the simulations of Quantumnet. It lasted for days, their funds were necessary to draw all our conclusions. Finally, we want to acknowledge all the contributors cited in the bibliography, and especially the ones of the Open Quantum Safe project which has been the angular stone of all this project. Without their wonderful tools, nothing would be possible to search easily on this topic.

Chapter 1

Modern Cryptography

1.1 Public-key cryptography

To protect our communications, we commonly use public standards and do not rely on the secrecy of the cryptosystem used. This has been a shift since the times when, to protect some piece of information, the engineers just created secret cryptosystems, hoping for it to stay secret. Of course, we estimated that this is not a solution anymore. In 1883 did Kerckhoffs stated, in the well known Kerckhoffs' principle, that the cryptosystem should still be secure if everything is public an known from the attackers, except the key. This approach can be endangered by any leak of data, and hackers from the entire world showed their capabilities in the field of reverse engineering. This is what the multiple stories of Slot Machine Hackers tells us: even when something is said as totally secure or random, if you only counts on the fact that nobody should learn how it works, then someone will find a solution anyway.

This is in this spirit that the cryptography experts are designing such cryptosystems that we could call **IND-CPA** or **IND-CCA**: with all the cryptographic material in a black box that we call an Oracle, and if you can use this Oracle to encrypt chosen data (in the CPA model) or event to decrypt chosen data (in the CCA model), then you will never be able to even distinguish encrypted data from totally random data. More formally : an oracle could never let an attacker deduce even 1 bit of information on the plaintext given the ciphertext.

Public-key cryptography is commonly used to establish the following properties :

- **Confidentiality**: protects the information from being accessed by unauthorized parties.
- **Authenticity**: allows the authorized parties to claim their identity during a secure communication.

- **Non-repudiation**: disallows any authorized parties to claim their messages have been forged or modify by someone, without having their private key stolen before.
- When adding other primitives such as MAC or symmetric crypto, **Integrity**: protects the information from being modified by unauthorized parties.

Public-Key Cryptography is often used in pair with symmetric cryptography, which costs less ressources to compute. In this case, the symmetric key is shared via asymmetric cryptography.

Today, we use very complex problems to generate the key-pairs that classical computers cannot break in reasonable polynomial time. The RSA algorithm is based on prime numbers problems, such as the RSA problem, which is describing the task of getting a private key from its associated public key, by factoring numbers in prime factors, would take ages to decrypt a bunch of keys.

1.2 TLS and SSH

To protect our communications, we all use two major public-key cryptography protocols. They are used to provide a shared secret key to a server and a client, but to achieve this goal, we use **RSA keys** and **Elliptic Curve Keys**.

1.2.1 SSH

The **SSH protocol** (Secure Shell) is used to access a remote machine and execute commands via a terminal or proceed to file transfers via the SSH file transfer protocol. Its goal is to provide the user **a secure channel** between the machine and the remote controller **over an insecure network** such as the Internet. Here is how it works:

- 1 - The client initiates a connection with the server.
- 2 - The server sends a public key, for example using RSA encryption.
- 3 - There is then a negotiation between the server and the client to define security parameters: Diffie-Hellman parameters, curves, symmetric encryption...
- 4 - The client can, on the secure channel, authenticate himself.

The Asymmetric Cryptography is used to create a secure channel. It can also be used to authenticate the client, as it is possible to casually log in by entering a username and a password, **SSH does also allow public-key authentication** if the client and the server are configured correctly.

1.2.2 TLS

The **TLS protocol** (Transport Layer Security) is used to initiate secure communications over TCP between a client and a server. It is mainly used over HTTP to provide **authenticity**, **confidentiality** and **integrity** between a user and an internet server. Today, the majority of servers does use TLS 1.2, but TLS 1.3 has arrived, does provide enhanced security, and is spreading slowly to become mainly used tomorrow. A TLS session is created by this method:

- 1 - The client sends a ClientHello message which contains the TLS supported versions and the cryptographic materials that can be used to process to the session initialization.
- 2 - The server sends a ServerHello message which contains some cryptographic choices shuch as the cipher specs.
- **At this point, on TLS 1.3, the connection between the server and the client is encrypted via asymmetric cryptography.** In TLS 1.2, we need to negotiate the key exchange mechanism that will be used. In TLS 1.3, this is not needed anymore, and the choice is implicitly made in step 1.
- 3 - The server provides a certificate containing his public key, that has been signed by a Certificate Authority which, in this protocol, plays the role of the trusted third-party.
- 4 - The client, with the CA certificate, does verify the server certificate.
- 5 - If everything is alright, then **the server is authenticated**. The client and the server will then agree on a private key, and begin the data transfer.

Here, the Asymmetric Cryptography is mainly used to **authenticate the server**, using trusted CA certificates that also use the RSA encryption. The **key exchange** is also done over asymmetric encryption.

1.3 The need of Post-Quantum Cryptography

1.3.1 Shor and Quantum Computers

Since the late 70s, the scientists are investigating the possibilities to exploit the quantum science and physics in order to solve some computational problems.

The Quantum Computing consists on the usage of Qubits, which are very little and unstable systems that, according to quantum laws of physics, can superpose states during a certain **coherence time**. A qubit

would be, which certain probability amplitudes, a 0 and a 1 at the same time. This coherence time is nowadays the main issue regarding the progress of quantum computing: it is very hard to maintain, in a quantum computer, the qubit in a coherent state.

With this strange properties that we will not investigate further in this report, it is possible to compute problems differently. The quantum computers does apply "Doors", e.g transformations, to the qubits, which allow in a way to **compute multiple calculations at the same time**.

The research in this field became major and also vital when Peter Shor did, in 1994, publish the **Shor's Algorithm**. This algorithm can **solve the RSA problem in a sub-exponential time**, by factorizing huge integers in prime numbers, which is actually what the RSA encryption relies on.

This paper had the effect of an earthquake. When, in the future, a practical quantum computer will be achieved, it will be possible to break the RSA keys, getting for example the private key from a public key, in a way faster time than with classical computers.

Now, industry giants such as IBM and Google are trying to create the computers that will allow this attacks to become a reality. The race is happening: with this technique, in 2016, 291 311 was factored using an adiabatic quantum computer, while in 2019, Zapata Computing and IBM announced having factorized 1 099 551 473 989 which is equal to 1 048 589 x 1 048 601.

If it is possible, from a public RSA key, to recover the private key, the consequences would be catastrophic:

- For **TLS**: CA usurpation, fake certificates creation, breaking of the forward secrecy by recovering shared keys exchanges that had been shared via TLS.
- For **SSH**: user usurpation and shared key recovery.

Researchers now have to find other ways to use asymmetric encryption, before a quantum computer is achieved.

Other papers and other techniques would also allow a quantum computer owner to break other security mechanisms such as Diffie-Hellman, which relies on the discrete logarithm problem (DLP).

1.3.2 Post-Quantum Algorithms

As we have seen, the Quantum Computation does not break the Asymmetric Cryptography, but more precisely the RSA problem. The simpler way to address this issue would be to find other ways to achieve to asymmetric encryption that have to be as safe and as efficient as the RSA problem. If researchers manage to find the good equivalent in time, then the issue would be addressed simply by making the protocols such as TLS and SSH

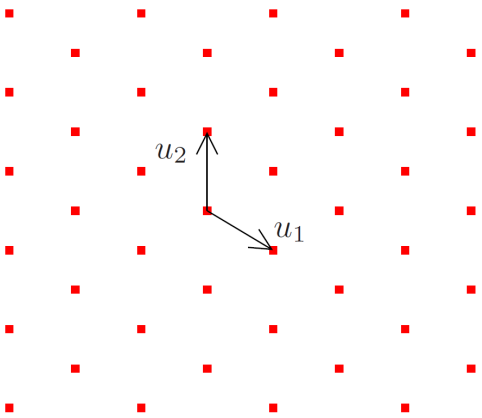


Figure 1.1: Example of a lattice in $\mathbb{R}^2$

incorporate this equivalent. People would have to generate new public keys, and use different algorithms, but the protocols would basically stay the same.

We call an algorithm that relies on a Quantum-Safe problem a **Post-Quantum Algorithm**. Such algorithms already exist, and use some well known techniques that we will briefly present. Note that if, with RSA, the same algorithm does allow encryption and signature, this is not the case of all the algorithms.

- **Lattice Based Cryptography**: One of the best hopes to construct a post quantum algorithm. Lattices are discrete substracts of $\mathbb{R}^N$, defined by an N vector base. *[Figure 1.1]* In the Lattice Based problems, we define a basis which stays secret. While, in a Lattice, there is **an infinite number of exploitable basis**, we can compute problems very simply when knowing a useful basis that are hard to compute without the right basis. Among these problems, we find the Shortest Vector Problem (to find a non zero vector in the lattice whose length is minimal), the Closest Vector Problem (to find the vector that is the closest to a define vector), and the Short Integer Solution (a problem based on standard worst case complexity assumptions). Again, what would be considered **the secret key here is the basis that allows the problem to be solved simply**.

- **Code-Based Cryptography**: With this solution, the private key is a generator matrix that is used by the owner to create codes to which errors are appended that can not be corrected without this private key. The public key would be a randomized matrix that allows to create codes, but not to decipher them.

There are of course other mechanisms that can be used by Post-Quantum algorithms.

Chapter 2

State of the Art

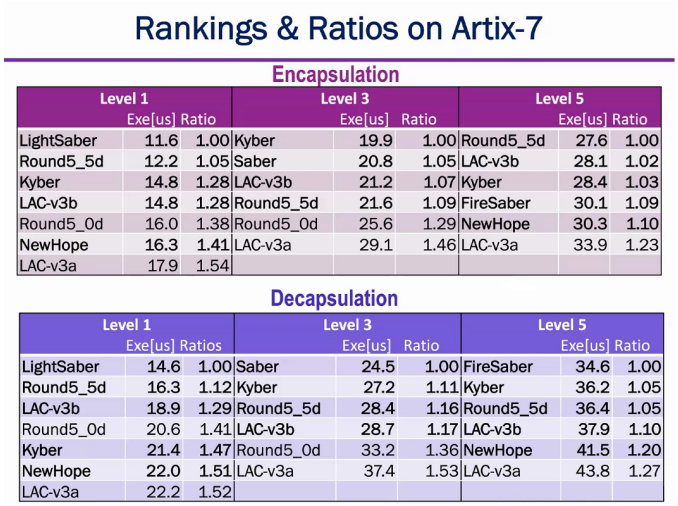
2.1 NIST Selection Process

While IBM promises a 1000-qubits quantum computer by 2023[1], the USA National Institute of Standards and Technology launched in December 2016 the Post-Quantum Cryptography Standardization process. This aims to find new Post Quantum Algorithms for Signature and Key Exchange - that are generally two different features addressed by two different algorithms - in order to generalize the use of the best ones and standardize them. As the organization did before, the process can be seen as a competition between several algorithms, presented by researchers, in a multiple round "contest". Of course, it is still a scientific research process, and all the people involved in this multiple rounds experiment participate in order to collaborate and cooperate to achieve the best-level post-quantum security. Many "contestants" where proposed in response to the NIST request, and big corporations also proposed their own algorithms (NewHope for Google, Picnic for Microsoft...). There were 82 submissions received, and only 64 algorithms were studied in the first round (19 signature algorithms and 45 key exchange algorithms).

At the end of round 1, there were 26 algorithms left (9 signature algorithms, 17 key exchange algorithms).

Then began, on July 22 2020, the Round 3, with only 7 main finalists: **Key Exchange Algorithms**: Classic McEliece, Crystals-Kyber, NTRU and Saber. **Digital Signature Algorithms**: Crystals-Dilithium, Falcon and Rainbow.

[1] IBM promises 1000-qubit quantum computer—a milestone—by 2023, www.sciencemag.org



Rankings & Ratios on Artix-7

Encapsulation

Level 1			Level 3			Level 5		
	Exe[us]	Ratio		Exe[us]	Ratio		Exe[us]	Ratio
LightSaber	11.6	1.00	Kyber	19.9	1.00	Round5_5d	27.6	1.00
Round5_5d	12.2	1.05	Saber	20.8	1.05	LAC-v3b	28.1	1.02
Kyber	14.8	1.28	LAC-v3b	21.2	1.07	Kyber	28.4	1.03
LAC-v3b	14.8	1.28	Round5_5d	21.6	1.09	FireSaber	30.1	1.09
Round5_0d	16.0	1.38	Round5_0d	25.6	1.29	NewHope	30.3	1.10
NewHope	16.3	1.41	LAC-v3a	29.1	1.46	LAC-v3a	33.9	1.23
LAC-v3a	17.9	1.54						

Decapsulation

Level 1			Level 3			Level 5		
	Exe[us]	Ratios		Exe[us]	Ratio		Exe[us]	Ratio
LightSaber	14.6	1.00	Saber	24.5	1.00	FireSaber	34.6	1.00
Round5_5d	16.3	1.12	Kyber	27.2	1.11	Kyber	36.2	1.05
LAC-v3b	18.9	1.29	Round5_5d	28.4	1.16	Round5_5d	36.4	1.05
Round5_0d	20.6	1.41	LAC-v3b	28.7	1.17	LAC-v3b	37.9	1.10
Kyber	21.4	1.47	Round5_0d	33.2	1.36	NewHope	41.5	1.20
NewHope	22.0	1.51	LAC-v3a	37.4	1.53	LAC-v3a	43.8	1.27
LAC-v3a	22.2	1.52						

Figure 2.1: Rankings of Round 2 key exchange algorithms, from CERG. [1]

2.2 Benchmarks

These algorithms were benchmarked and tested. The evaluation criteria are Security, Software efficiency and Hardware efficiency, that get along with flexibility, simplicity and licensing. These benchmarks has been made by a team from the NIST: the **C**ryptographic **E**ngineering **R**esearch **G**roup. They delivered, during a seminar on Oct 27, 2020, their results. [1] I'll then present some of the results that we find useful to think about future implementations.

One of the interesting result that came out of this process was this table *[Figure 2.1]*, which allow us to know the security level and the efficiency of each **Key Exchange algorithm**.

We see that in term of Time efficiency, Saber and Crystals-Kyber are much more efficient than their concurrent. But the speed is not the only thing the future users will have in mind, especially with the high capacity computers and servers that are often used today.

For example, we also look at the size of the key related to the size of the ciphertext for a defined input. *[Figure 2.2]*

We immediately see the trade-offs: do we consider the ciphertext size as important as the public key size, so we could choose an algorithm such as Sike, Saber or NTRU? Or could we make the assumption that a large public key is not an issue if we send shorter ciphertexts and we keep the key for multiple communications? In this case, Classic McEliece would also be a good choice.

This graph has also been computed for **Signature Algorithms**. *[Figure 2.3]* Trade-offs are even more visible here. We could choose a solution which

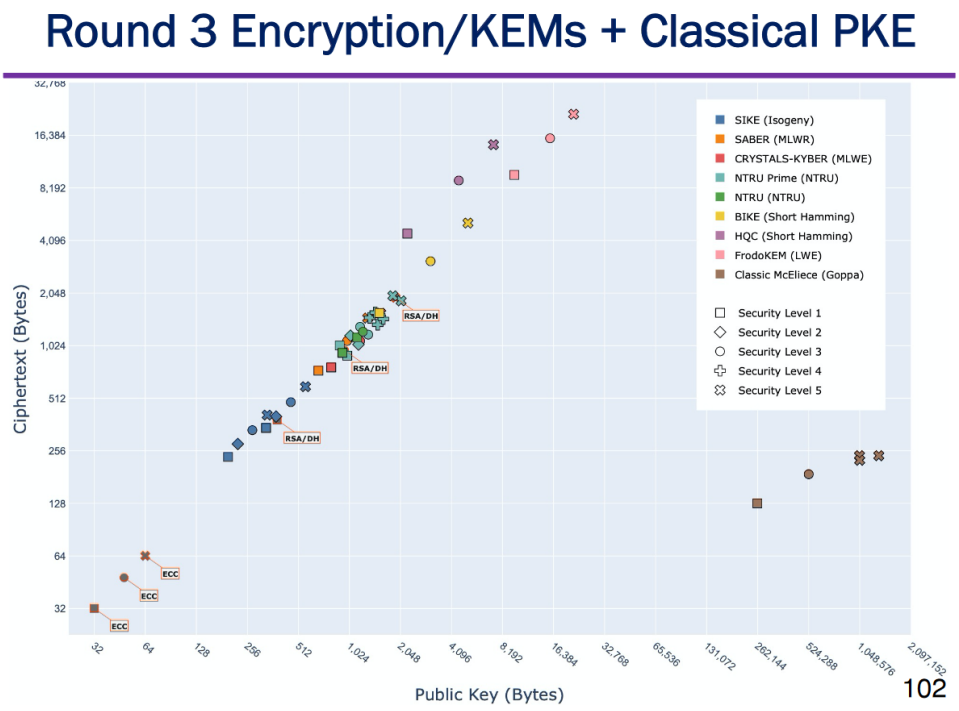


Figure 2.2: Key and ciphertext sizes of Round 3 key exchange algorithms, from CERG. [1]

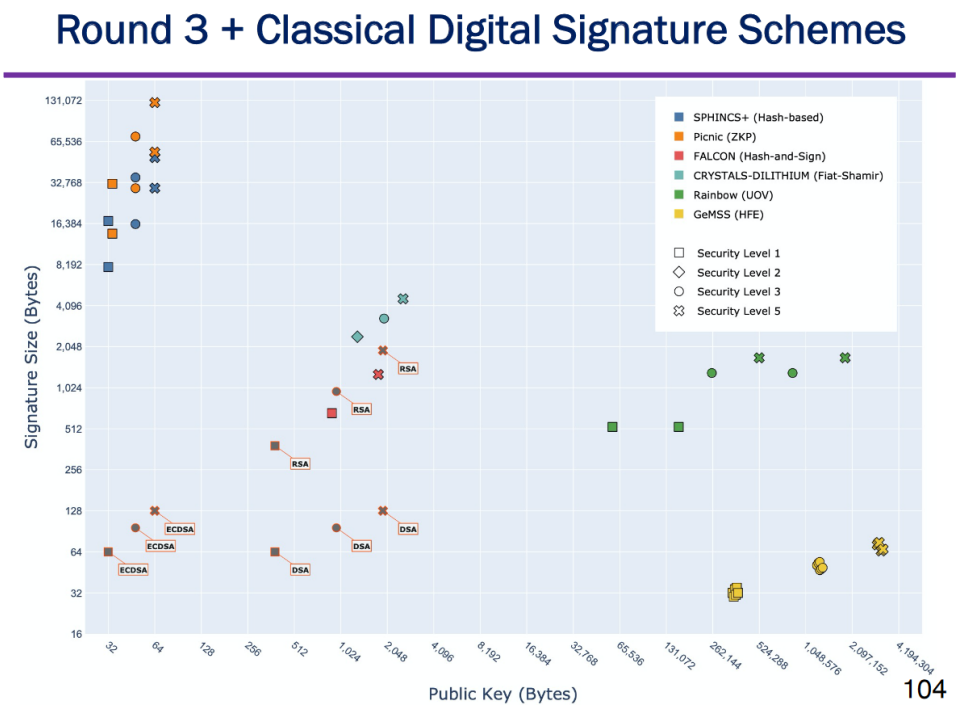


Figure 2.3: Key and signature sizes of Round 3 Signature algorithms, from CERG. [1]

looks like RSA, that would be Crystals-Dilithium or Falcon. We could also want to reduce to the minimum the signature sizes by choosing Rainbow. The other algorithms are alternative algorithms that are out of the "main competition", that might be standardized but that have other issues.

What we see with these benchmarks is the number of criteria to consider. The sizes and the computation times are just parts of what we need to make the right choice that will probably dominate the SSH and TLS cryptographic materials in the next decades.

Some researchers have tried to perform larger scale experiments to **test networks behavior**. "Wide-scale" experiments, at the scale of the Internet, are really rare. However, they still exist: in 2019 was conducted the "TLS Post-Quantum Experiment" with Google and Cloudfare, which allowed to test some Post Quantum Cryptosystems by integrating post quantum clients in Google Chrome while Cloudfare did host servers to handle some minimal TLS requests. The cryptosystems that have been tested are CECPQ2 (which was based upon NTRU) and CECPQ2b (which was based upon SIKE). These two mechanisms have a level of security of 1. CECPQ2 uses larger public keys and ciphertexts while CECPQ2b is lighter. Besides, CECPQ2 is faster that CECPQ2b. *[Details in Figure 2.4]* It revealed an-

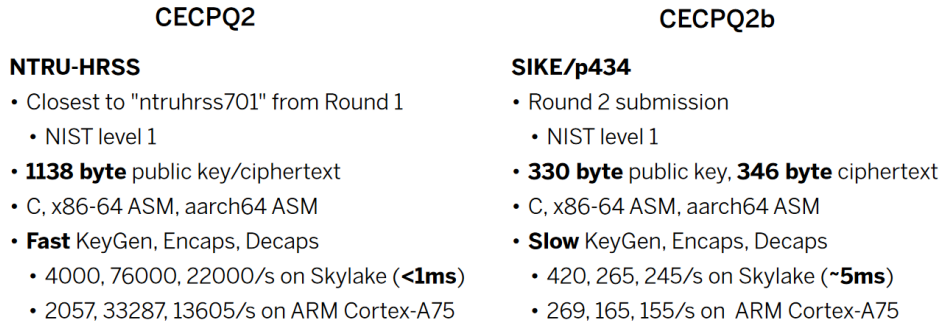


Figure 2.4: CECPQ cryptosystems details, from Cloudfare x Google presentation during the second PQCS NIST Conference. [2]

other trade-off: while CECPQ2b is slower... It is less exigent, and can be run faster on older computer! If the network is composed of slower devices and peers, then CECPQ2 will be less efficient. Another criteria to consider before standardizing the next PQ algorithm.

There are not a lot of other experiments. In a smaller scale, Christian Paquin from the Microsoft Research Team, Douglas Stebila and Goutam Tamvada from the University of Waterloo did publish the February 6th, 2020, a paper that dispensed publicly some tools and a method to benchmark the algorithms on our own. [3] The Framework they provided make use of the Network Namespaces of the Linux kernel to create nodes, routes, firewall rules, and even some limitations such as delays, RTT and packet loss rates. Their framework is available on Github and has been a very useful contribution to begin this project.

2.3 Hybrid schemes

While some cryptographic algorithm transitions went really quick in the past, we have examples of transitions that took a long time, such as the Elliptic Curve Cryptography, invented during the 1980s and used by default by Google in 2011. To avoid this issue regarding to Post Quantum Cryptography, we have to prepare this switch with some research and by providing solutions to the users that will be quicker to use.

As detailed in a paper from Eric Crockett, Christian Paquin and Douglas Stebila named *"Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH"*[4], a way to solve this problem is to propose an **hybrid mode**. This would mean: combine multiple algorithms to ensure a security level at least equal to the weaker algorithm used. We can then imagine strong associations of algorithms, using a classical one such as RSA, and a Post-Quantum one such as Saber. This proposition has a great

future: it allows the early adopters to get the potential of Post-Quantum security while allowing them to use a strong and robust algorithm for more confidence and security.

The Hybrid mode is harder to implement, and some questions are not yet perfectly resolved. The main issue that has to be addressed is about **negotiation**. How to let two peers agree on a specific algorithm? We can consider three ways the negotiation would be done:

- **Separately**: we get one PQ algorithm and one traditional algorithm, negotiated separately, but we have the risk to use weak combinations.
- **Informally**: we get two algorithms that can be PQ or traditional, the protocol does not check anything about the algorithms, and we still have the possibility of weak combinations.
- **Explicitly**: we attribute to each pair of protocol an identifier, which would be easier, but could lead on a quadratic number of identifiers.

The paper then introduces and explains implementation details about how to get the hybrid mode working. What we learn is that some protocols **fail due to too large messages**, bigger than what the protocol allows. The implementation details also prove us that implementing the Hybrid mode in TLS 1.3 is harder than in TLS 1.2, but that the **optimization of the negotiation in TLS 1.3 does solve the issue of the combinatory explosion of cryptographic primitives**, which is good news.

We also see that the way we execute the combination of the algorithms has yet to be decided.

2.4 The Open Quantum Safe project

To allow and facilitate the research on this subject, a team named the Open Quantum Safe [5], leaded from the University of Waterloo by Douglas Stebila and Michele Mosca, has been founded on the principles of collaborative research and open source development.

Aiming to create softwares for prototyping quantum-resistant cryptography, several contributors worked on online and public github repositories while writing and publishing a dozen of research papers.

The source code created by this team helps anyone who wants to start to create prototypes, and contribute to the research on this topic. The main contributions are **liboqs** and **oqs-openssl** which integrate in OpenSSL the Post Quantum algorithm in a simple way.

Starting from this, we can already build more complex tools. For example, it is possible to build nginx from scratch using oqs-openssl to create websites that use the post quantum algorithms.

The algorithms that we can setup are taken from the NIST Selection Process. The complete list and further details are available at https://openquantumsafe.org/ and on the github page, directly accessible from the project page.

Chapter 3

Realisations

3.1 Quantumnet - A toolkit to benchmark PQ algorithms

When we look at the state of the art, we see that there are few experimental benchmarks, on simulated or real conditions, to test which algorithms are better to guide the choice of the algorithms we should standardize or not. When it comes to the Hybrid Schemes, this kind of research is even more rare and hard to find.

The idea would be to build together some tools, with the help of the projects the OQS team provides us, and construct benchmarks that would:

- Let us have some interesting results to interpret and to share.
- Be modifiable and open-source so future researchers could fork and build their own experiments.

We will then introduce the tools that have been created, which are a part of a suite named Quantumnet. This toolkit should be available to use on Ubuntu or Fedora distributions.

3.2 A minimal openssl server creator

A tool that has been created to start manipulating the tools provided by the OQS team is a minimal OpenSSL server creator.

It consist on a Flask server and a webpage that allows the creation of a minimal OpenSSL server that uses a certificate signed by a CA using a user defined Post Quantum Signature Algorithm. Then, the user can try to initiate a TLS handshake with this server.

This simple experiment has these parameters:

- The IPv4 only **address** of the Flask server (default: 127.0.0.1:5000)

- The IPv4 only **address** of the TLS server (default: 127.0.0.1:4433)
- A **signature algorithm** from the list.
- A **key exchange algorithm** from the list.

3.3 Quantumnet: use Mininet along with PQ algorithms

The main tool of this toolkit is **Quantumnet**, a program that couples a build of nginx which is using oqs-openssl and **Mininet** [6].

Mininet is a Virtual Network creator. It allows the user to build nodes, switches, controllers and links, specify their characteristics and monitor their inputs and outputs. This can be used as a CLI tool and as an API in Python based applications. We will use the second option to build networks topologies and run a server that will answers to multiple nodes at the same time.

All the nodes that are not the server use the command **stime** that creates and discard as may TLS Handshake at it can during a certain time.

Quantumnet allows the user to build his own topology, and to change some variables such as:

- The **port** that you want the node server to use (default: 4433)
- The **Signature algorithm** to benchmark (default: dilithium2)
- The **Key Exchange algorithm** to benchmark (default: saber)
- The **bandwith** of the switch - server link in Mbps (default: 8)
- The **delay** of the switch - server link (default: 10ms)
- The **loss rate** of the switch - server link (default: 0)
- The **CPU percentage** allowed to the server (default: 1)
- The **number of clients** (default: 1)
- The **maximum size of the queue** for the switch - server link (default: 14)
- The **time of the experiment** during which stime will be executed on the clients (default: 60s).

Like the Openssl server creator, Quantumnet - Quantumnet is provided with flask server that runs an interface accessible via a web browser to define and launch experiments.

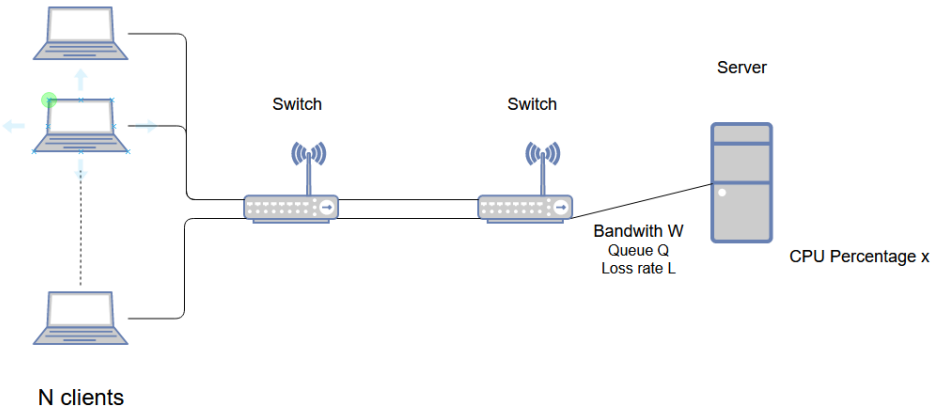


Figure 3.1: Quantumnet Topology

Two additional flags have been added to support the Hybrid Schemes. The OQS-Openssl code has been provided with a support for this functionality: depending on the claimed Level of security of the signature and key-exchange algorithms, it is possible to add a second layer of computation with a similar security level. Here is the list of traditional algorithms that are implemented in addition to the quantum-safe algorithms[5] :

- For **Key Exchange** algorithms : We use ECDH with P256 curve if the Post Quantum algorithm has L1 security, P384 curve for L3 security and P521 curve for L5 security.

- For **Signature** algorithms : We use ECDSA using NIST's P256 curve if the Post Quantum algorithm has L1 or L2 security, P384 curve for L3 security and P521 curve for L5 security.

Finally, the last feature, implemented as the flag -www, is making the server distribute a webpage to the nodes to simulate the transfer of data over SSL.

The network constructed by Quantumnet has the topology described if Figure 5.1.

3.4 Tests on a remote server

As running the tests on my local personal computer would not provide significant results, it has been decided to run the tests on a remote server paid by the LASEC lab.

3.4.1 Protocol

The "Quantumnet" Post-Quantum Test Server has been extended with an API that allow the Flask server to receive HTTP POST requests, allowing it to launch the experiment without using the web interface. With this addition, it is possible to launch many tests and to let the server compute all the experiments and results.

Client-side, we will write scripts that will launch these experiments. It will have to make evolve different parameters, and store all the results as JSON objects. The important inputs that will vary are :

- The Signature Algorithm, and whether it is hybrid or not.
- The Key Exchange Algorithm, and whether it is hybrid or not.
- The Bandwith
- The Loss rate
- The number of nodes
- The webpage to download after the handshake. For all the tests, we chosed to download the wikipedia page of Post Quantum Cryptography which is 636 KB in size.

The main output will be **the number of connections that are achieved by the nodes** via the openssl stime command. With this data, we will be able to compute the average time for an handshake. Each node will launch the stime command with the same options, at the same time. We will take one of the outputs, and consider all the nodes got the same output. The nodes will all connect to the same server. The TLS handshake will reset after each connection, so it has to be started over again each time.

When all the computations have been done on the remote server, it has been analyzed and gathered into graphs that we will present below. As the servers could not have been used continuously during the projects, the machines were the same for the different phases. We can consider we stayed on the same machine all along, and discard the uncertainty of the machine characteristics changes.

The characteristics are the following :

- CPU : 2x Intel Xeon E5-2680 v3 (Haswell), 2x 12 cores, 24 threads, 2.5 GHz, 30 MB cache
- RAM : 16x 16GB DDR4-2133 256 GB total
- DISKS : 2x 200 GB SSD SATA3, 2x 2TB SATA3 HDD 7200 RPM
- NETWORK : 2x 10 GbE

- No GPU

The goals of the experiments are to :

- Compare the results of Quantumnet with the one of previous research papers.
- Observe the behavior of the algorithms in different situations (bad network, etc).
- Compare different post quantum algorithm, with similar or different claimed security levels.
- Compare, for the same algorithms, the performance with the hybrid scheme functionality implemented.

3.4.2 Results

Correctness of Quantumnet

The first objective is to compare Quantumnet results with our expectations, to see if it is a useful tool to benchmark quantum algorithms. If we look at Figure 5.2 from the previous subsection, we see that the average time to execute an handshake and download the webpage is growing in a linear way with the number of nodes, which seems normal. We could argue that, the servers having high computation capabilities with a 16 core processor, this growth may be too heavy. Maybe does Quantumnet not take all the concurrent possibilities of the machine.

Let's then have a look on the changes on the Handshake and download completion time introduced by the evolution of the loss rate factor. When we take a look at former research that aimed at prototyping the Dilithium2 signature algorithm on different loss rate factors, we observe that in most of the cases, the completion time does not evolve a lot between 0% and 10% of loss rate [3], followed by an exponential growth of the curve. Quantumnet has been used to construct a curve of this evolution depending on the loss rate for different levels of claimed security levels (Figure 3.2).

As we retrieve the exponential curve that we were waiting, we can first conclude that claimed level of security is not a factor that matters for defining the curve according to Quantumnet, as no curve appears to be way superior to any other. This is quite surprising, as the length of the keys and ciphertexts tends to grow with the claimed security level : the public key, secret key and ciphertext lengths are doubled between Lightsaber and Firesaber for example. We can doubt the precision of the results provided by Quantumnet, but they seem to validate the over-all behavior of the Quantumnet software as the curves are similar to the ones that we can find in former research papers.

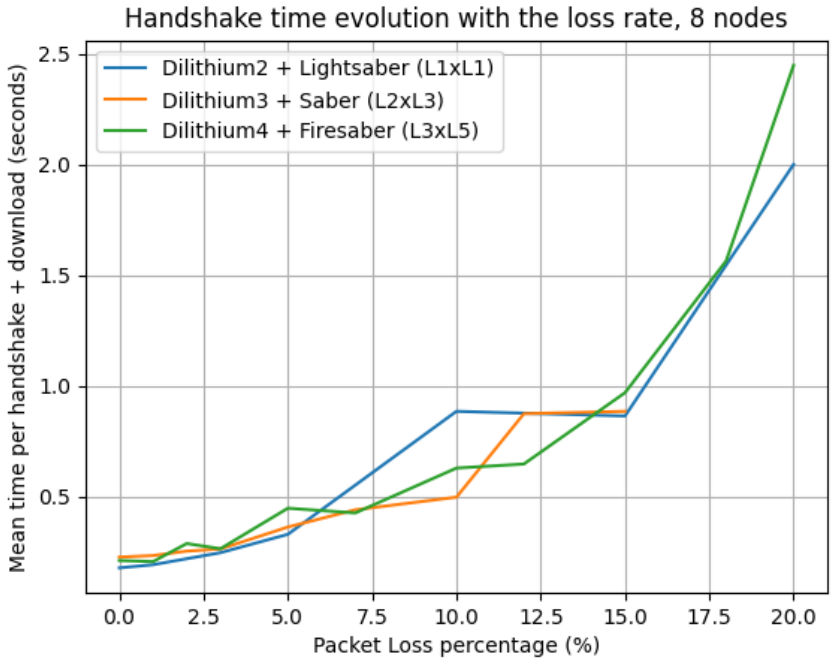


Figure 3.2: Average completion time for different claimed security levels

Conclusion : Quantumnet provides coherent results, but they bear uncertainties. Maybe does this differences occur because of the small cost of the TLS crypto computation (ciphertexts are always small, for all the key exchange algorithms, for example) compared with the download, the protocol...

Different Loss rate situations

We wanted to show the influence of ciphertexts or signature sizes by running different signature algorithms with high loss rate factors (we run different experiments with loss rates evolving with an average of 10%). In this situation, we put the results in perspective with the size of the signature sizes, which are obviously the data that will be sent in any signature procedure - while public keys could be stored and should not been sent at each signature. The results of this experiment is showed in Figure 3.3.

We see that algorithms with large signatures are slower. The value of "RainbowIa" seems incoherent here, but it is important to keep in mind that the Rainbow Classic signature scheme use very large private and public keys that slows down the computation a lot (RainbowVc, the L5 alternative of Rainbow Classic, take more than 10 seconds to conclude a handshake for a node) (Figure 3.5)

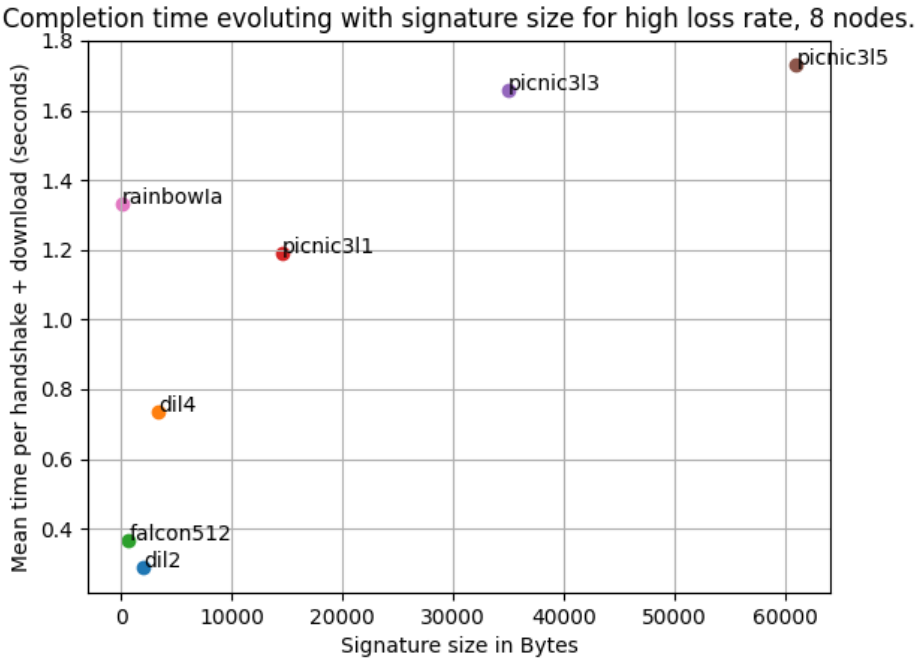


Figure 3.3: Average completion time for signature algorithms on high loss rate.

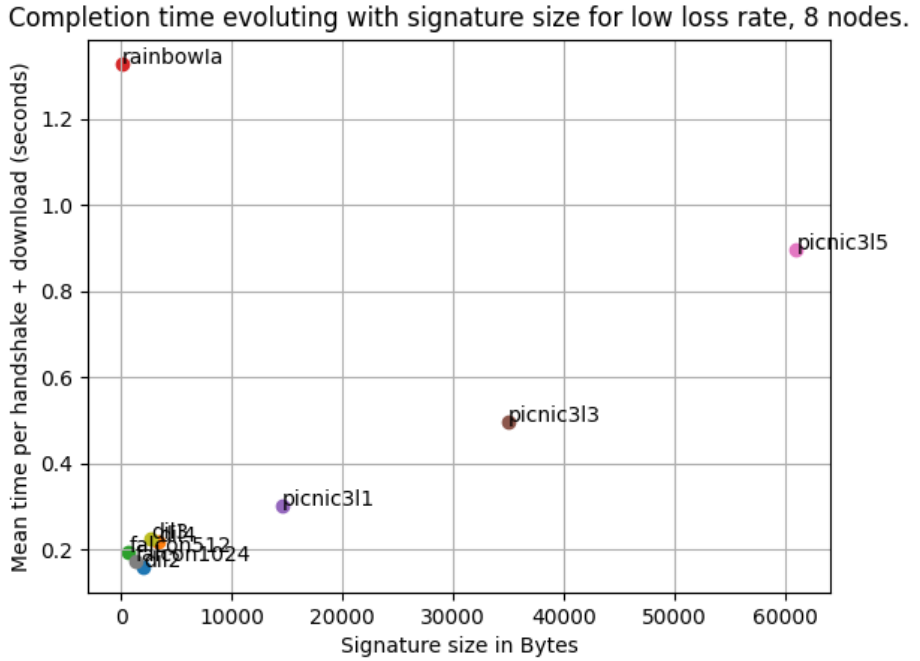


Figure 3.4: Average completion time for signature algorithms on low loss rate.

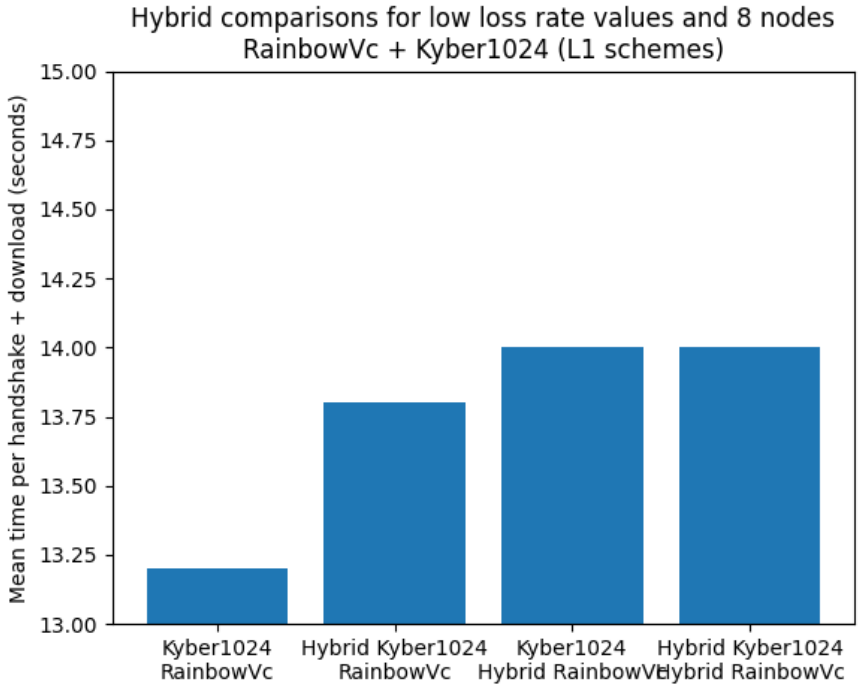


Figure 3.5: RainbowVc + Kyber performances.

As expected, in low loss rate situations, the signature or ciphertext sizes are still relevant parameters, as shown in Figure 3.4 for example, which present key exchange algorithms in low loss rate situations, but have way less impacts on the performance. We can see that on high loss rate situations, Picnic3 L5 completion and download time goes up around 1.8s while in low loss rate situation it rises to 0.9s.

On these pictures, the most surprising values are the ones of Rainbow Classic, an algorithm with two variants : RainbowIa, which has claimed L1 security level, and RainbowVc, which has claimed L5 security level. On these graphs, the results of RainbowVc, that uses a very little signature of 64 Bytes like RainbowIa, have been removed for visibility because they are too high. It shows that signature size are not the only factor in these cases : Rainbow Classic uses well known problems that are proved to be secure, but the cost of this security is that the handshakes are way slower than any other algorithms.

Conclusion : In high loss rate conditions, the signature size is an important factor of completion time for signature algorithm.

This study did not show anything about Key Exchange algorithms variations in different loss rate situations, because all ciphertext sizes are quite small and in the same range (600-1600B) (see Figure 3.6).

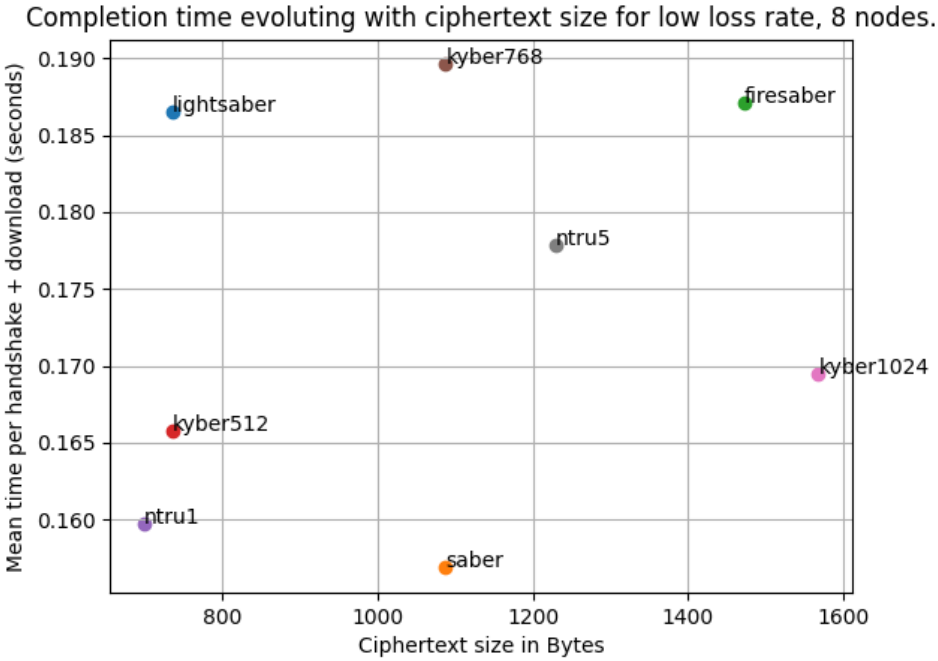


Figure 3.6: Figure 6 - Average completion time for key exchange algorithms on low loss rate.

Algorithms comparison

The last section let us distinguish some low performance algorithms that are Rainbow Classic and Picnic (which has been eliminated during the second round and competes in the third round as an "alternate candidate").

It has to be kept in mind that a low performance algorithm is not always a low quality algorithm : some features can be more interesting in some usages. For Rainbow Classic, the signature size is only 64 Bytes which is way lower than any of the other competitors. Moreover, some of the fastest algorithms such as Dilithium does rely on newer mathematical problems that have been less studied. There are some algorithms that are way faster, but that could also bear vulnerabilities and this is why the third round does still include some slower algorithms.

Having this idea in mind, let's compare the algorithms in terms of speed. We added in Quantumnet the possibility to run DSA and RSA, then we can compare the post quantum algorithms with the DSA/RSA couple that we use a lot today, and see how much the efficiency differs from one algorithm to another.

To compare the signature and the key exchange algorithms, we will run Quantumnet, pairing them with the same other algorithm to estimate the differences between them.

For the Signature algorithms, we use the Key Exchange Algorithm Saber (L3). We choose to not add Rainbow Classic or Picnic because we already know that they are way slower. For the Key Exchange algorithms, we use the Signature Algorithm Dilithium 2 (L1).

With the Figures 3.8 and 3.7 we can observe that :

- **Nothing beats the RSA + DSA duet**, that is still the fastest way to achieve security. This is the fastest duo by far, and it can provide very high security.
- **In general, the more secure an algorithm is, the longer it will take to complete an handshake.** At the beginning of the NIST Selection process, it has been stated that the searchers had to focus on the L1 and L3 levels of security, preserving L5 for most important security issues.
- **Some algorithms are better, in general, than others, even at a high level of security.** We will discuss these differences in detail in the following.

When we look in the detail for the Key Exchange algorithms, we can clearly see that NTRU L5 (which real name is NTRU HPS 4096821) is, when we aim for an higher security, the best choice among others. NTRU also seems to be better for L1 security with its variant NTRU HPS 2048509. It seems then possible to assume that NTRU is the best choice when aiming

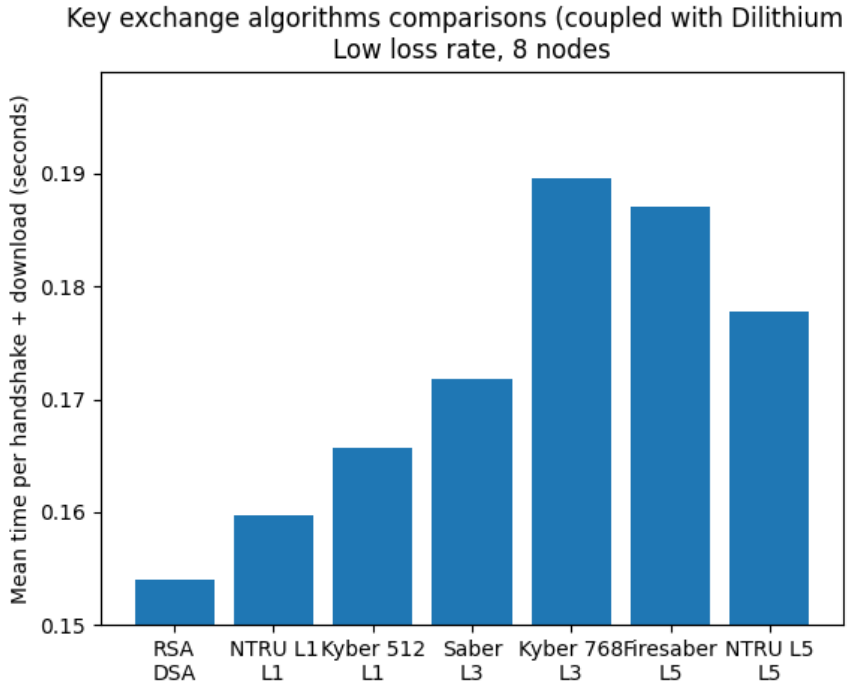


Figure 3.7: PQ Key Exchange algorithms comparisons

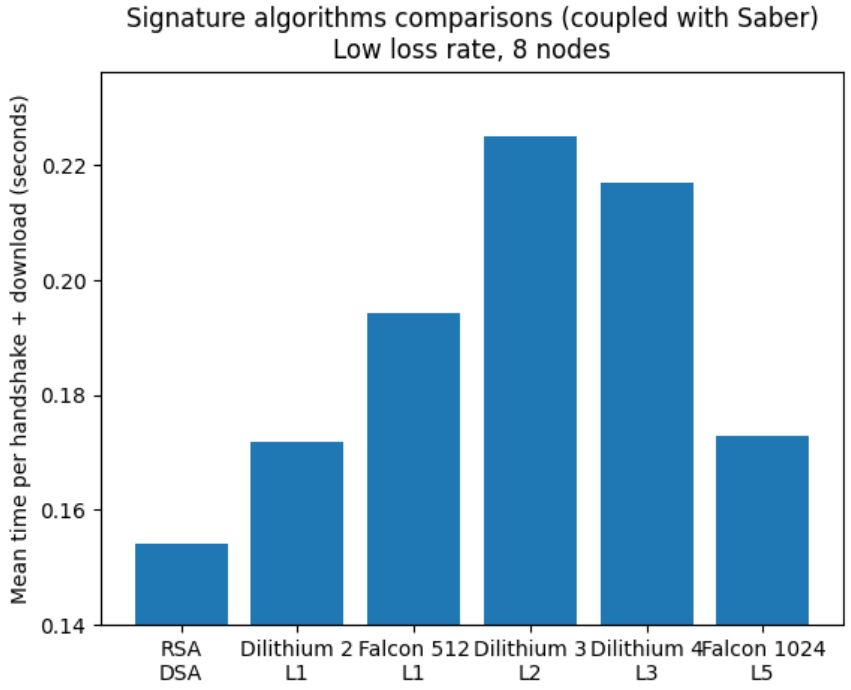


Figure 3.8: PQ Signature algorithms comparisons

for speed. Moreover, as NTRU has been invented in 1996 and got a lot of upgrades during the last decades, it also seems to be a secure choice, based on years of research and tests.

Conclusion : Our study points the NTRU cryptosystem as a good candidate for standardization, which seems secure and faster than its opponents.

Considering the Signature Algorithms, we can see that Dilithium is a good choice when aiming for a standard security. When it comes to L5 security, there is less choices available as Dilithium does not provide any L5 solution, and multiple factors have to be taken in account : as stated before, Rainbow Classic and Picnic have been discarded from these tests as it has been shown that they are way slower. Nevertheless, Rainbow Classic offers strong security algorithms with a very light signature (only 64 Bytes). Another fact that has to be taken in account is that Dilithium has been paired with the Key Exchange algorithm Kyber, creating a larger cryptosystem named "Crystals". If Kyber has been chosen for the Key Exchange, Dilithium could still be the best candidate when aiming for a standard security.

Conclusion : We assume that Falcon provides a fast L5 security, and that Dilithium is sufficient for a standard security, but other factors has to be taken in account, as the signature size, when it comes to choose for a signature algorithm.

Hybrid vs Pure Post Quantum

We stated that a goal of the research was to **characterize the difference between pure-quantum and hybrid computation**. A curve showing the evolution of the completion time with the loss rate has been computed using Dilithium2 and Lightsaber, but with all possible combinations of pure-quantum and hybrid schemes. The combined EC curve for each of these algorithms is the p256 elliptic curve. (Figure 3.9).

What this Figure tends to confirm is that while the duets using Hybrid Lightsaber are slower than duets using Pure Quantum Key Exchange Mechanism, Hybrid Dilithium2 seems slightly faster than Pure Dilithium2. This result is very interesting : there must be a trade-off between computation, key size and ciphertext size that works better for a specific configuration. Of course, as the Packet loss percentage increases, key and ciphertext sizes are the factors that are the more important to define which solution to chose.

When we look a the graph of the average completion time for different loss rates with all the Picnic variants - an alternative concurrent of the 3rd round, we clearly see that the hybrid variants that use lighter signatures are faster than the pure post-quantum variants for high loss rate values (Figure 3.10).

Conclusion : The trade-off between Hybrid and Pure post quan-

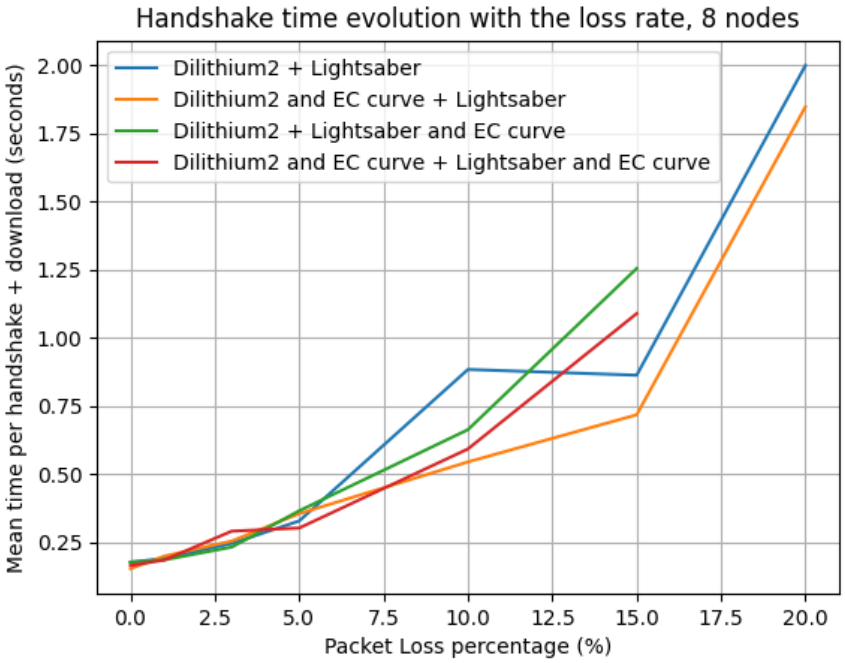


Figure 3.9: Average completion time with pure quantum and hybrid computation.

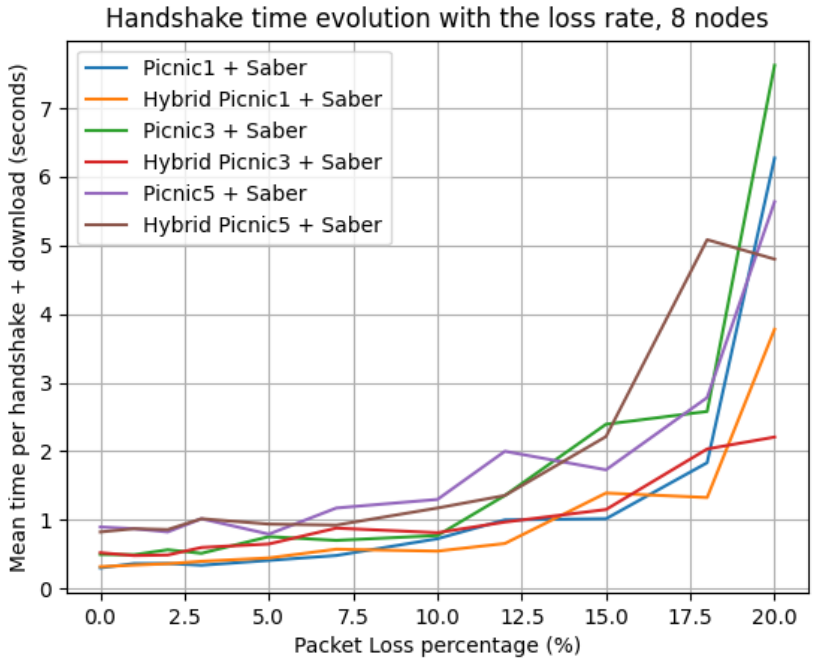


Figure 3.10: Average completion time with pure quantum and hybrid computation for Picnic algorithms.

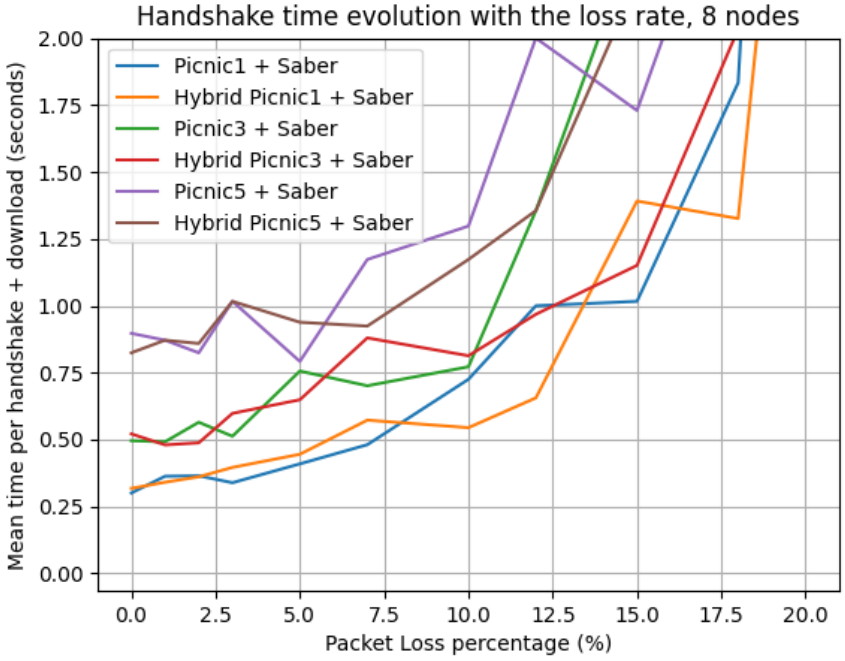


Figure 3.11: Average completion time with pure quantum and hybrid computation for Picnic algorithms.

tum schemes does depend on the network conditions. For the Picnic Algorithm, in low quality networks, the Hybrid variants are way better.

To continue our study, we can zoom on the first part of the Figure 3.10 (Figure 3.11).

As expected, we can differentiate the duets : Picnic3 L1 is faster than Picnic3 L3, which is faster than Picnic3 L5, hybrid or pure-quantum. This hierarchy is very clear between 0% and 10% of loss rates. We also see that if the network has good quality, the difference between Hybrid and Pure Quantum is not determinant.

Let's compare more generally these two choices (Figures 3.12 and 3.13).

In general, nothing can be showed with this kind of comparisons because of the variance of the experiments : there is no general rules in low loss rates. What we know is that it depends on the algorithm used, and that the evolution of the mean time will not be very impacted by the choice. It is kind of surprising, because we could argue that as the Hybrid computation adds a step of computation for the server and for the client, the Hybrid Mode should always be slower. It appears, with our experiment, that it is not the case.

Conclusion : In low loss rate conditions, there is not general rules for the choice of Quantum Only or Hybrid computation. The Hybrid computation is not always longer.

This result can be discussed : as the Hybrid Schemes does always add a computational step, this result is quite surprising.

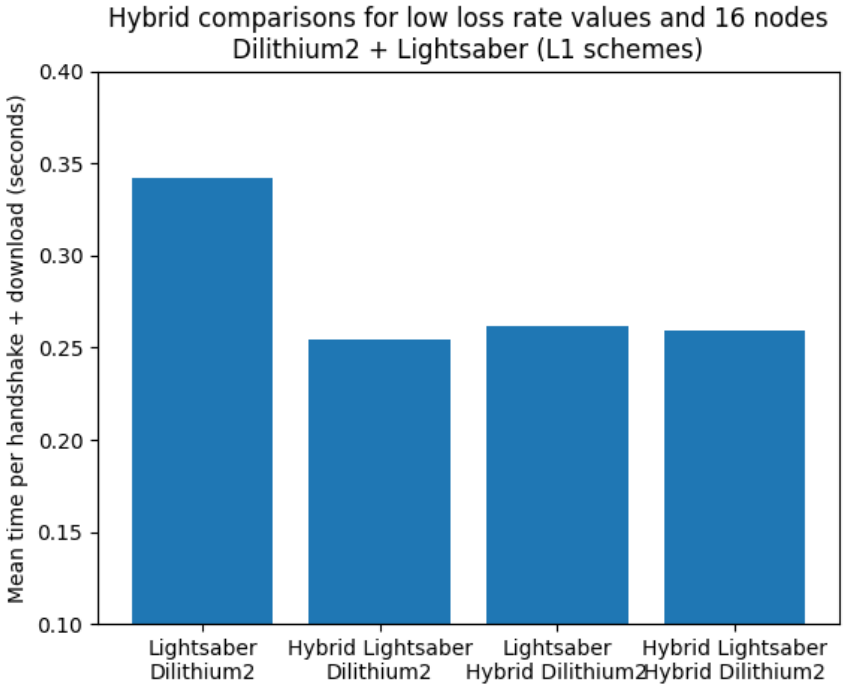


Figure 3.12: Comparison between Pure Quantum and Hybrid modes for Dilithium2 x Lightsaber.

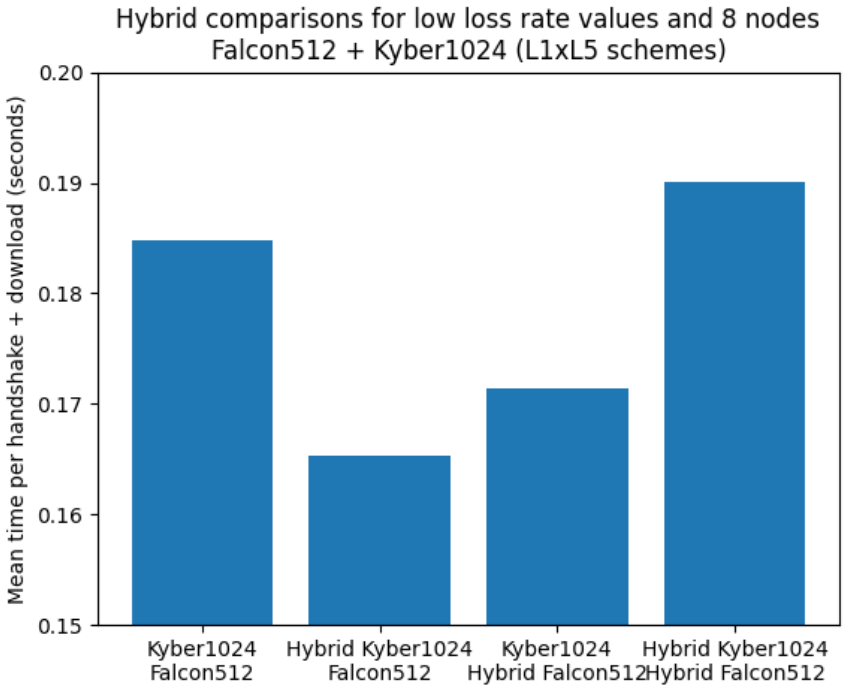


Figure 3.13: Comparison between Pure Quantum and Hybrid modes for Falcon512 x Kyber1024.

Chapter 4

Conclusion and Future Work

Quantumnet answers to a practical need of the future : the benchmarking of post-quantum algorithms by users that do not have the possibility to construct real size networks, and to create custom personalized tests using the code and the hard work of the community. We have shown that the results are coherent, can be exploited easily to construct graphs, but we also reached some uncertainties regarding to the protocol that we used for this report.

We hope that this project will be used by other searchers that will build over it other network topologies or implement new features. Regarding to our own studies and analysis, we managed to point the fastest Key Exchange and Signature algorithms while putting this in perspective with all the other factors to make a choice.

To continue on this project, the following subjects could be taken :

- The LASEC laboratory has developed an alternative hybrid combiner, to create hybrid schemes by combining a post-quantum scheme and a classical scheme together in a way that should be faster than what is done today. What could be done now is implementing into OQS-liboqs and OQS-openssl this combiner and benchmark them. This is a typical low level programming work that was too long for this semester, but that could be a great evolution for Quantumnet.

- As the time with the servers was limited, all the possibilities of Quantumnet have not been exploited. New experiments could be conducted with throughput constraints, or more complex network topologies. One could also add new algorithms to OQS-liboqs and OQS-openssl (round 2 candidates, alternate candidates...),

Bibliography

[1] Kris Gaj, from the CERG Team. *Implementation and Benchmarking of Round 2 Candidates in the NIST Post-Quantum Cryptography Standardization Process Using FPGAs*. 3rd Round of the NIST PQC standardization process Seminar, Oct 27, 2020.

[2] Krzysztof Kwiatkowski, Luke Valenta et al. *The TLS Post-Quantum Experiment*. Cloudfare blog, Oct 30, 2019. More at: https://blog.cloudflare.com/the-tls-post-quantum-experiment/

[3] Christian Paquin, Douglas Stebila and Goutam Tamvada. *Benchmarking Post-Quantum Cryptography in TLS*. IACR Cryptology ePrint Archive, Report 2019/1447, Feb 6, 2020.

[4] Eric Crockett, Christian Paquin and Douglas Stebila. *Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH*. IACR Cryptology ePrint Archive, Report 2019/858, Jul 19, 2019.

[5] Douglas Stebila, Michele Mosca. *Post-quantum key exchange for the Internet and the Open Quantum Safe project*. In Roberto Avanzi, Howard Heys, editors, Selected Areas in Cryptography (SAC) 2016, LNCS, vol. 10532, pp. 1–24. Springer, Oct, 2017.

[6] Mininet Project. *Mininet*. http://mininet.org/