Titre: Title:	Performance Evaluation of Post-Quantum Cryptography in Avionic Communication
Auteur: Author:	Jimmy Hamel
Date:	2023
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Hamel, J. (2023). Performance Evaluation of Post-Quantum Cryptography in Avionic Communication [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/55087/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Performance Evaluation of Post-Quantum Cryptography in Avionic Communication

JIMMY HAMEL
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Août 2023

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Performance Evaluation of Post-Quantum Cryptography in Avionic Communication

présenté par **Jimmy HAMEL**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Heng LI, président
Gabriela NICOLESCU, membre et directrice de recherche
Omar ABDUL WAHAB, membre

ACKNOWLEDGEMENTS

First of all, I would like to thank my research advisor, Gabriela Nicolescu, for giving me the opportunity to lead this research project and especially taking over the project when my last advisor left. I want to thank Felipe Gohring de Magalhaes as well for his incredible help during the implementation of my experimental setup.

I also want to thank Carillon Information Security Inc. and especially Patrick Patterson for the guidance and help for the avionic aspects of this research. I also want to thank Guillaume Amringer, Dave Coombs and Den Drown for all the support during my work at Carillon.

Finally, I want to thank all of my friends who encouraged me, as well as my girlfriend who supported the most during these last 2-3 years.

RÉSUMÉ

La possibilité de disposer d'ordinateurs quantiques entièrement fonctionnels représente un risque pour les systèmes cryptographiques publics utilisés aujourd'hui. En effet, grâce à l'algorithme de Shor, ces ordinateurs pourraient être en mesure de casser les algorithmes RSA et ECC. Afin de sécuriser nos communications contre de tels ordinateurs, il est nécessaire d'utiliser des algorithmes résistants à ces potentielles attaques quantiques. Le NIST a lancé un processus de standardisation de quelques candidats post-quantiques pour l'encapsulation de clés et les signatures numériques, et a actuellement sélectionné quatre d'entre eux pour la normalisation. Cependant, certains de ces algorithmes nécessitent une puissance de calcul plus élevée, une bande passante plus élevée et une utilisation de mémoire plus importante que les algorithmes classiques utilisés aujourd'hui. Cela pose un défi pour l'implémentation de tels algorithmes dans des dispositifs à faible consommation d'énergie.

Dans cette recherche, nous avons évalué les performances de ces algorithmes dans les systèmes avioniques, plus précisément dans les systèmes de communication avionique. L'avionique fait référence au matériel électronique utilisé par les aéronefs pour soutenir leur fonctionnement, leur navigation, leur communication et leur fonctionnalité globale. Les systèmes de communication avionique sont généralement considérés comme de l'électronique de faible puissance avec une bande passante très limitée, ce qui rend la mise en œuvre du post-quantique sur ces systèmes un défi.

La conception des communications air-sol englobe toutes les communications avioniques où un aéronef communique avec une station au sol. Ces communications ne sont généralement pas chiffrées. Cependant, l'infrastructure ATN/IPS est actuellement en cours de développement pour prendre en charge l'ajout de confidentialité et d'authentification via DTLS.

Dans cette recherche, nous présenterons les impacts du passage des algorithmes cryptographiques classiques aux algorithmes post-quantiques dans DTLS pour les communications air-sol en utilisant ATN/IPS comme modèle. Nos résultats ont montré que les meilleurs candidats pour les communications avioniques sont KYBER et Falcon. En émulant la phase Handshake de DTLS dans ATN/IPS sur du matériel avionique, nous concluons d'abord que l'utilisation de bande passante plus élevée de ces algorithmes sera le facteur le plus impactant dans les communications avioniques lors de la transition aux algorithmes post-quantiques. Plus spécifiquement, passer de ECC à KYBER doublerait généralement le temps de complétion de la phase Handshake et la même chose peut être dite pour la transition ECC à Falcon.

ABSTRACT

The possibility of fully functional quantum computers is causing a risk to the public cryptosystems used today. Indeed, using Shor's algorithm, these computer could be able to break RSA and ECC . To secure our communication against such computer, we need to use algorithms resistant to these potential quantum attack. The NIST started a process to standardize a few post-quantum candidate for key encapsulation a digital signature and currently selected 4 of them for standardization. However, some of these algorithms requires higher computational power, higher bandwidth and higher memory usage. This pose a challenge to implement such algorithms in low-power devices.

This research aims to evaluate the performance of these algorithms in avionic systems, more specifically avionic communication systems. Avionic refers to the electronics that is used by aircrafts to support their operation, navigation, communication, and overall functionality. Avionic communications systems are usually considered legacy hardware with very limited bandwidth which makes the implementation of post-quantum on these system a challenge.

Air-ground communication design all avionic communication where an aicraft communicates to a ground station. These communication are usually not encrypted. However, the ATN/IPS infrastructure is currently being developed to support the addition of confidentiality and authentication through DTLS.

In this research we will present the impacts of transition from classical cryptograpgic algorithms to post-quantum algorithms in DTLS for air-ground communication using ATN/IPS as a model. Our results showed that the best candidates for avionic communication are Kyber and Falcon. By emulating a DTLS handshake in ATN/IPS on avionic hardware, we first conclude that the higher bandwidth usage of these algorithm will be the most impactful factor in avionic communication when transitioning to post-quantum algorithms. More specially, transitioning from ECC to KYBER would generally double the handshake completion time and the same can be said about transitioning from ECC to Falcon.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

IETF	Internet Engineering Task Force
OSI	Open Systems Interconnection
NIST	National Institute of Standards and Technology
PQ	Post-Quantum
PQC	Post-Quantum Cryptography
OQS	Open Quantum Safe
ATN	Aeronautical Telecommunication Network
ATN/IPS	Aeronautical Telecommunication Network Internet Protocol Suite
IPS	Internet Protocol Suite
HF	High Frequency
VHF	Very High Frequency
VDL	VHF Digital Link
HFDL	High Frequency Data Link
RF	Radio Frequency
ATC	Air Traffic Control
TLS	Transport Layer Security
DTLS	Datagram Transport Layer Security
DH	Diffie–Hellman
ECDH	Elliptic-Curve Diffie–Hellman
ECDHE	Elliptic-Curve Diffie–Hellman (Ephemeral)
KEM	Key Encapsulation Mechanism
ECC	Elliptic-Curve Cryptography
AES	Advanced Encryption Standard
IOT	Internet of things
satcom	Satellite Communication

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

1.1 General Introduction

It has been shown that a large-scale quantum computer could break most of the public-key cryptosystems used today using Shor's algorithm [3]. To protect the confidentiality of our communication against such computer, we need to use quantum resistant algorithms called post-quantum algorithms. However, these algorithms are often newer and require a higher computational usage or a higher bandwidth usage than currently used algorithms such as RSA(Rivest–Shamir–Adleman) and ECC(Elliptic-Curve Cryptography). This makes it difficult to implement protocols using post-quantum algorithms on constrained devices like the ones implemented for IOT devices, mobile phones or other devices with limited resources.
It can take some time to modify our current security standards to include post-quantum algorithms which is why we need to study the impacts of using them in our current infrastructures as fast as possible so that we are prepared against large-scale quantum computers. One area specifically where standardization can take a long time to complete is the world of avionics. Simply modifying a small thing in avionic can often take a few years to be actually applied. Some standards have been in development for a few years now [4]. This means that to add something as significant as post-quantum cryptography to avionic communication, we need to plan ahead.
While most avionic communication does not support encryption of the communication channel, there is currently an updated communication architecture for avionic that is being developed called Aeronautical Telecommunication Network Internet Protocol Suite (ATN/IPS) which will introduce the support of the DTLS protocol to add confidentiality and authentication to air-ground avionic communications [4, 5]. This creates an opportunity to study the impacts of introducing post-quantum algorithms in this new architecture that is being developed for avionic communications. Since the public-key algorithms are the ones most at risk against quantum computers, [6, 7] the handshake part of the DTLS protocol is the most important step to consider as it uses both key exchange and digital signatures. In other words, we need to study the impact of using post-quantum algorithms to establish a secure communication channel for air-ground communication where an aircraft communicating to an Air Traffic Control tower. The handshake can be used both for authenticating both peers and establishing parameters for confidentiality between 2 entities. While we will consider the impact of both post-quantum Key Encapsulation Mechanisms (KEM) and post-quantum digital signatures, more efforts will be put into the study of KEM algorithms. It is more

important to secure **in advance** the confidentiality of our communication against a future quantum computer, than it is to protect **in advance** our authentication infrastructures. Indeed, we need to consider that all past communication will be threatened to be deciphered by a quantum computer, making the need to switch to post-quantum algorithms that protect confidentiality more urgent.

1.2 Definitions

1.2.1 Quantum Computer

Quantum computers are a type of computer that exploits the properties of quantum physics to execute its operations. By doing so, it opens a whole different sets of possible operations that are not possible on classical computers. However, most research that has been done in this field is still theoretical since there are many physical challenges in creating a fully functional large-scale quantum computer. One of the biggest challenges is called quantum decoherence, or simply put, the management of noise into the quantum calculation [8]. This means that it is still impossible to create a quantum computer that can hold a large enough amount of information, or qbits, to be more efficient at general tasks than a classical computer. While there is an increasingly amount of experimental work trying to overcome these limitations, there has been a lot of theoretical work that has already been done to prepare ourselves to a possible fully functional large-scale computer in the future. One important field of research has been in the field of quantum algorithms [9].
Qbits are the quantum equivalent of the classical computer bits we know. This is what is used to store information and these are the entities on which quantum computers are executing their operations. Unlike classical bits, qbits can hold the value of 0, 1 and a linear combination of 0 and 1, which is what makes quantum computation quite different than classical computation. Currently, we have not been able to build a quantum computer that has enough qbits to execute meaningful tasks more efficiently than a classical computer could. While many have claimed to have achieved "quantum supremacy" or the goal of having a quantum computer that executes tasks faster than a classical computer, none have done it in a significant way to currently impact society [9, 10].
Quantum algorithms are algorithms that run on quantum computers. These use qbits as their computational units. Two of these algorithms would have a great impact on society if a large-scale quantum computer was available. **Grover**'s algorithm is a search algorithm which would divide by 2 the time to brute force an encryption algorithm [6, 7]. While this would not be too alarming, **Shor**'s algorithm would, however, fully break the asymmetric cryptographic algorithms that we use today: RSA and ECC [3].

1.2.2 Post-Quantum Cryptography

Post-quantum designs any cryptographic processes that are considered resistant against a large-scale quantum computer. Post-quantum cryptography designs a set of cryptographic algorithms that are considered quantum resistant [11]. These algorithms are currently considered quantum resistant for two main reasons:

1. We have not found any quantum algorithms which can break these cryptographic schemes. In other words, they will remain quantum resistant until we prove them otherwise. Since Shor's algorithm is the main algorithm that is worrying for currently used cryptographic algorithms, we can also say that these algorithms are quantum resistant because Shor's algorithm cannot break them.

2. The mathematical explanation is that Shor's algorithm helps to solve the hidden subgroup problem for finite abelian groups in polynomial time. Abelian groups are currently essential to cryptography. These groups, also called commutative groups, are groups where the order in which the group operations are applied do not impact the result. A simple example would be : for all a, b in A, an abelian group: $a \cdot b = b \cdot a$. This characteristic is especially important in the definition of ECDH where the commutativity is used to generate the cryptographic keys on both sides of a communication without ever sending the key on the channel [3, 12] .

Open quantum safe The Open Quantum Safe project is an open source project started by Douglas Stebila and Michele Mosca which aims at helping with the prototyping of post-quantum projects. The main part of this project is a library called liboqs which provides a common API to execute cryptographic operations, mainly encapsulation and digital signing, using post-quantum algorithms. The supported algorithms are generally the NIST candidates for standardization. Another part of this initiative is the integration of post-quantum algorithms using liboqs in other cryptographic open source projects such as openssl [1].
KEM or Key Encapsulation Mechanisms are a category of cryptographic algorithms that are used to exchange a key between two parties. Unlike key exchange algorithms, Key Encapsulation is a concept to simple encryption, meaning that the actual key that will be exchanged will end up on the communication channel, but in an encrypted state. There is no post-quantum algorithm that can achieve a key exchange similar to ECDH. This is mostly due to the mathematical properties of Shor's algorithm.
Digital Signature is a cryptographic primitive that is used for authentication. Digital

signatures can be employed to verify the authenticity of messages or document. Unlike classical algorithms, no post-quantum algorithm can achieve both KEM and signature.
Hybrid algorithms are cryptographic mechanisms where both a post-quantum and a classical algorithm are used together. These are especially important to consider when we are thinking about implementing post-quantum algorithms in systems in production. Indeed, while post-quantum algorithms are believed to be secure against both classical and quantum attacks, most of them are still quite new, meaning that we don't have as much trust in them as we have in RSA and ECC. This means that it is currently hard to convince people to switch to post-quantum algorithms while a large-scale quantum computer still does not exist. By using hybrid algorithms, if one of the algorithms gets broken in the future, the system remains safe. As an example, if the post-quantum part of the algorithm gets broken by a classical attack, the classical part of the algorithm will still be protecting the system. The other way around is also true: if a large-scale quantum computer becomes functional, it will only be able to break the classical part of the algorithm. While this makes a lot of sense security-wise, the overhead of using 2 algorithms is obviously not something that all cryptosystems can afford to use [13].
NIST Security Levels are a measure of security developed by NIST. Usually, an algorithm's security level goes from 1 to 5. An algorithm claiming level 5 would be at least as hard to break as AES256. NIST uses this metric to recommend which parameters of an algorithm should be used in certain scenarios [14].

1.2.3 ATN/IPS

ATN The Aeronautical Telecommunication Network is a communication architecture that defines a common interface to communicate avionic data.
ATN/IPS This is the latest update to the ATN architecture that is currently being drafted. This will introduce confidentiality and authentication to air-ground avionic communication using DTLS [5].

1.2.4 (D)TLS Handshake

Handshake The process in which 2 parties establish a session key to secure a communication channel.
TLS The Transport Layer Security protocol providing secure communication between 2 peers or more. The protocol runs over TCP and uses a TLS handshake to exchange a cryptographic key to secure the communication channel.
DTLS The Datagram Transport Layer Security protocol is a TLS variant over UDP(User

Datagram Protocol) [15].

1.3 Problematics

Our scenario contains 2 main problematics. The first one is the fact that there is not much security around air-ground avionic communication [16]. Because of the lack of encryption and authentication, people can eavesdrop on messages sent between an aircraft and a control station. Information such as the flight level, the destination, the message content and the ICAO aircraft code can be accessed easily because of the lack of encryption. While the risk of harm is low, the access to this information should still be confidential but people may easily access them without being detected. The only deterrent in place is the fact that it is illegal to do. There is also a risk of impersonation using Man-in-the-middle attacks where someone could impersonate either an aircraft or the ATC. In theory, this means that fake ATC clearances could be sent to aicrafts [16]. This problematic and these risks are the ones that ATN/IPS is trying to mitigate.
The second problematic revolves around the possibility of a quantum computer with enough qbits that it would be able to use Shor's algorithm to break RSA and ECC. Because of this, even if the ATN/IPS is designed to protect against attacks from standard computers, if it is not ready to use algorithms resistant to quantum attacks, a large-scale quantum computer will be able to break the encryption and authentication from the ATN/IPS secure channel.

1.4 Possible solutions

There are multiple components that exist to secure a communication between two peers. The first one is asymmetric cryptography. Asymmetric cryptography is a type of cryptography where peers have both a public key and private key. There are usually 2 types of asymmetric cryptography: Key exchange and key encapsulation. For key exchange, we use algorithms like ECC where both the client and the server generate their own keys and use the peer's parameters to generate the same shared secret on each side. For key encapsulation like RSA however, it is much more straight forward. The client can use the server's public key to encrypt a shared secret he generated and only the server will be able to decrypt the secret using its own private key.
Asymmetric cryptography is usually costlier in terms of performances than symmetric cryptography. This is why generally, we use RSA or ECC to generate a shared secret which will then be used as a key with symmetric algorithms like AES to secure the communication channel on each side. This is how confidentiality is provided on a communication channel [17,18].
For authentication, we also use asymmetric cryptography. However, instead of encrypting with the peer's public key, the private key is used to **sign** the data. This way you can prove

that the data really came from a specific entity since the signature can only be verified using the entity's own public key. Using these signature, you can then authenticate specific messages. The concept of digital signatures is often used with digital certificates. While digital certificates have multiple purposes today, they are essentially a chain of trust that use a chain of digital signatures to prove that one's certificate can be trusted because it has been signed by an authority that is known to be trusted [19].
However, asymmetric cryptography is rarely used alone. More likely, it will be used within a protocol like TLS that will help secure the channel by providing confidentiality, authentication and data integrity. TLS is the main protocol used to protect a communication channel today. It first uses the concept of a handshake between 2 peers to establish a session key using asymmetric ciphers and concludes by securing the channel using a symmetric cipher like described earlier. During the handshake, it can also use a digital certificate to authenticate either or both sides of the communication. Finally, pre-shared keys can also be implemented instead of certificates which can be useful on constrained environment. However, it is less scalable and becomes harder to use as you increase the number of possible peers [15, 18, 20].

1.5 Proposed solution

The ATN/IPS draft is currently planning on using DTLs as the protocol to secure the channel between an aircraft and a ground tower. DTLS is the UDP equivalent of TLS and is more popular in resource-constrained environment since TCP has an extra communication overhead necessary for its reliability. The ATN/IPS working group also selected elliptic curves to be used for key exchange as well as their signature for their digital certificates [5, 15].
While they answer the first problematic, this solution is still vulnerable to quantum attacks. For this research, we are proposing to use post-quantum algorithms for both the key exchange and the signatures of DTLS to secure ATN/IPS against future quantum threat. More specifically, we are proposing to use the algorithms that were selected by NIST for standardization as well as the algorithms that are still in the run for the fourth round of submission [21] .

1.6 Research Objectives

This work wants to report the viability of adding post-quantum security to an avionic environment. To do so, we want to add post-quantum capabilities to an existing implementation of DTLS to analyze the impacts of using post-quantum algorithms instead of classic algorithms in terms of **CPU usage**, **bandwidth usage** and **total handshake times**. We also want to simulate air-ground transmission delays to have a more realistic idea of the impacts

in air-ground communication scenarios. Finally, we will offer a proposition of post-quantum algorithms to consider for the ATN/IPS network. By doing so, we hope to answer these research questions:

Q1: What are the impacts of using post-quantum algorithms in avionic communication, more specifically in ATN/IPS?

Considering post-quantum algorithms will require more hardware resources than the algorithms that the ATN/IPS initiative are currently planning to use, before even considering using post-quantum algorithms, we need to be aware of the performance impacts of switching from classical algorithms to post-quantum algorithms that require higher computational power, memory and bandwidth. Here we should expect higher delay before a communication can be established between an aircraft and an Air Traffic control towers.

Q2: Considering the impacts of switching to post-quantum algorithms, which algorithm(s) should we recommend for consideration to secure avionic communication?

By analyzing the impacts collected during our experiments, we want to come to a conclusion about which algorithms should be considered to help secure the DTLS handshake for air-ground communication in ATN/IPS. The choice will be made by considering our chosen metrics as well as considering the overall usability of these algorithms. Indeed, some algorithms might not be usable on constrained environments like avionics due to their very large key and certificate sizes.
Overall, following the current state of the art that we will present in the next chapter, we expect Kyber and Dilithium to be the PQ (post-quantum) algorithms with the best results for our experiments. We also believe that McElice will be one of the worst choices for air-ground DTLS handshakes due to its very large key sizes not suited for environments with limited bandwidth [21, 22].

1.7 Thesis plan

In the second chapter, we will first present the main concepts and characteristics of air-ground communication data links that are useful for this research. This chapter aims to give the reader a better understanding of the different components of air-ground communication that are relevant to this research. It will also give the reader a deeper dive on public-key cryptographic protocols, more specifically DTLS.

In the third chapter, we give a state of the art on post-quantum cryptography and on recent experiments using DTLS and TLS in constrained environment scenarios, as well as avionic scenarios. We then describe how our research differs from what has already been done.
In the fourth chapter, we will show the solution that has been implemented by us to answer our research questions. This chapter will describe the different devices and libraries used as well as the changes we had to make to the existing DTLS implementation. The chapter will also present a simulation model and its implementation enabling the measurement of air-ground transmission delays.
In the fifth chapter, we will describe how we validated our implementation. We specify which algorithms and which variants of these algorithms were used for testing. We also detail the different validation scenarios that we have put in place to evaluate the performances of these algorithms. We follow with details on the specific metrics that are extracted from our experiments to evaluate the impacts of using post-quantum algorithms.
In chapter 6, we present the results of our experiments under the different scenarios described in chapter 5. With these results, we conclude which factors are more impactful for avionic communication. We use these factors to conclude which algorithm(s) performs better for avionic communication and we propose a few ideas to reduce their impact on DTLS handshakes when bandwidth is limited.
Finally, in the seventh and last chapter, we offer a summary of our research and of our results. We conclude by specifying limitations related to our experimental setups as well as ideas for future improvement.

CHAPTER 2 CONCEPTS

In this chapter, we will review important concepts about avionic communication and cryptography. We will start by reviewing the characteristics of ATN/IPS that are relevant for this research and then conclude by reviewing the cryptographic protocol that will be used to secure communication in this future avionic communication architecture.

2.1 Avionic communication and ATN/IPS

In this section we will describe concepts relevant to avionic communication and the ATN/IPS network. Current avionic communication does not support transmission links using a secure encrypted channel. While this feature is currently being documented in the ATN/IPS drafts, current communication specifications for avionic air-ground communication are defined in ICAO's specification document called "Digital Data & Voice Communication System" [23]. Since part of the goals of the ATN effort is the digitalization of avionic communication using a common interface, the main types of communication that will be used here are data links which are ways to communicate using digital information instead of using voice over the radio. These are usually more bandwidth efficient than using voice radio to communicate. The data are modulated before being sent over the available frequencies [23].

2.1.1 Digital Modulators

A digital modulator is a device that manipulates a digital frequency onto a carrier wave so that the signal is ready for transmission. The output of these digital modulators will be an analog signal as seen in figure 2.1 [23, 24].

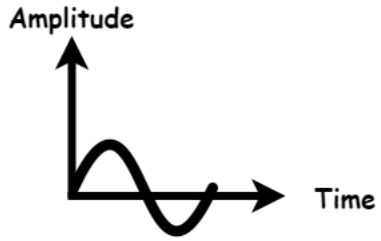

Figure 2.1 Analog signal

2.1.2 VHF and VDL

Communications on Very High Frequencies (VHF) are communications that requires a line of sight on the receiver. While this means the range of this type of communication is limited, the available bandwidth is higher than HF communication.

Bandwidth and frequency band

The available band of frequencies for VHF is 118.000–136.975 MHz. The usable frequencies are separated by 25 kHz. The bitrate for the second modulation mode is 31500 bits/s using D8PSK modulation.

Modulation

VHF Data Links(VDL) technically supports 4 different modulation modes; however, only the second mode will be studied here.

2.1.3 HF and HFDL

Communications on High Frequencies are communications that do not need a line of sight with the receiver. Those are usually used during overseas flights. While this means the range of this type of communication is almost unlimited, the available bandwidth is lower than VHF communication [23].

Bandwidth and frequency band

The available band of frequencies for HF is 2.8 to 22 MHz. HFDL supports up to 3 different bitrates depending on the value of M, the number of bits being sent simultaneously for one symbol. If M = 2, the bit rate is either 300 or 600 bits/s. If M = 4 the bitrate is 1200 and for M = 8, the bitrate is 1600 [23].

Modulation

The modulation used for HFDL is M-PSK where M is the same parameter described earlier. This means that 3 variants can be used : 2-PSK, 4-PSK and 8-PSK [23].

2.1.4 ATN

The Aeronautical Telecommunication Network is an architecture for avionic communication and its goal is to provide a common interface internationally to air-ground, ground-ground and air-air communication. One of the ways the architecture allows this is by using Data Links to provide information. The current ATN implementation is called ATN/OSI and is based on the Open Systems Interconnection (OSI) model. However, the ATN/IPS is the new generation of avionic protocol that is currently being developed and wants to replace the current OSI implementation with communications based on the IP protocol. This will introduce new security features like encryption and authentication using DTLS handshakes [2, 4, 5]. The figure 2.2 shows an overview of the targeted architecture for ATN/IPS.

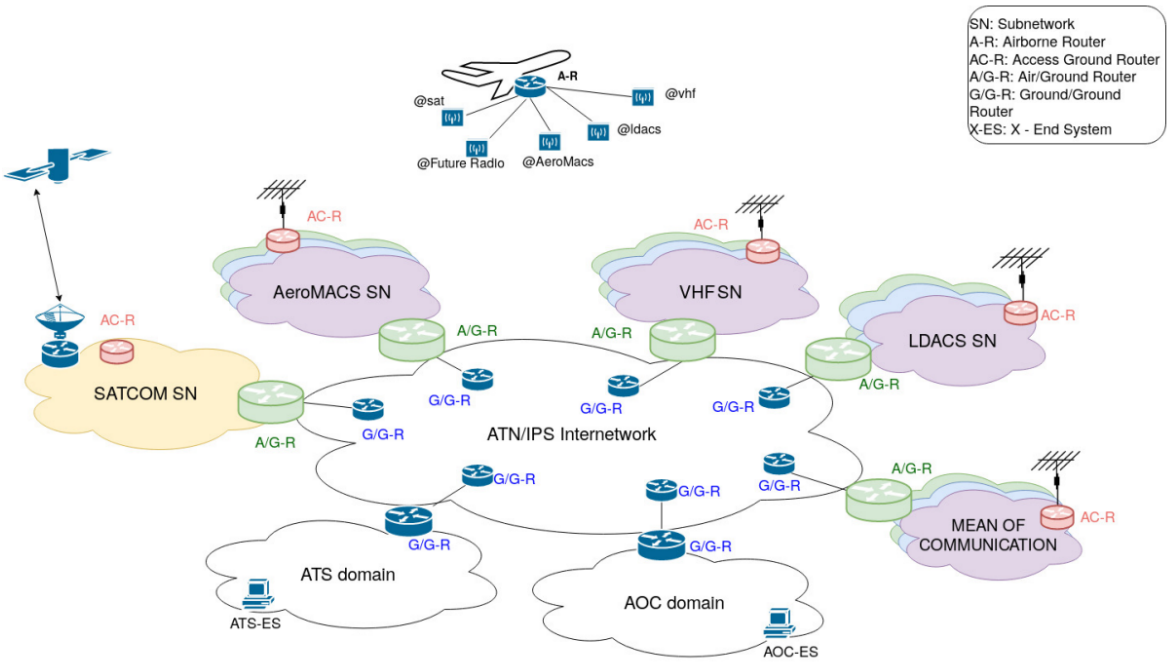

Figure 2.2 ATN/IPS future infrastructure [2]

Supported security protocols for ATN/IPS

Since specification is still being drafted, the details described here might change over time. However, it is safe to assume that the main ideas will remain the same.

The goal of ATN/IPS will be to communicate using UDP. UDP is usually chosen for applications with strong bandwidth requirements as it is more bandwidth efficient than TCP. With UDP as the base, ATN/IPS will also use DTLS to secure the communication between peers. There are 3 main types of algorithms that usually define a DTLS cryptographic suite: a key exchange algorithm, an encryption algorithm to use with the exchanged key and a signature algorithm. For air-ground communications, the currently selected algorithms are ECDH256 for key exchange, AES-GCM – 128 for encryption and ECDSA256 for digital signatures [4,15].

2.2 Cryptography

In this section we will describe relevant cryptographic concepts and protocols related to our experiments.

2.2.1 DTLS 1.2

Datagram Transport Layer Security (DTLS) is a communication protocol that helps provide confidentiality and authenticity to messages sent between 2 parties. The authenticity is usually achieved by the use of cryptographic certificates while confidentiality is achieved by the use of encryption of the messages sent. DTLS is the UDP variant of the widely used Transport Layer Security (TLS) [15].
One of the most important steps of both DTLS and TLS is the handshake which can help authenticate one or both peers while exchanging a secret key to secure the communication. However, each version of DTLS and TLS has its own variation of the handshake. For this project, we selected DTLS 1.2 which was the highest version of DTLS available in openssl at the time. DTLS 1.3 is, however, currently being worked on [25].

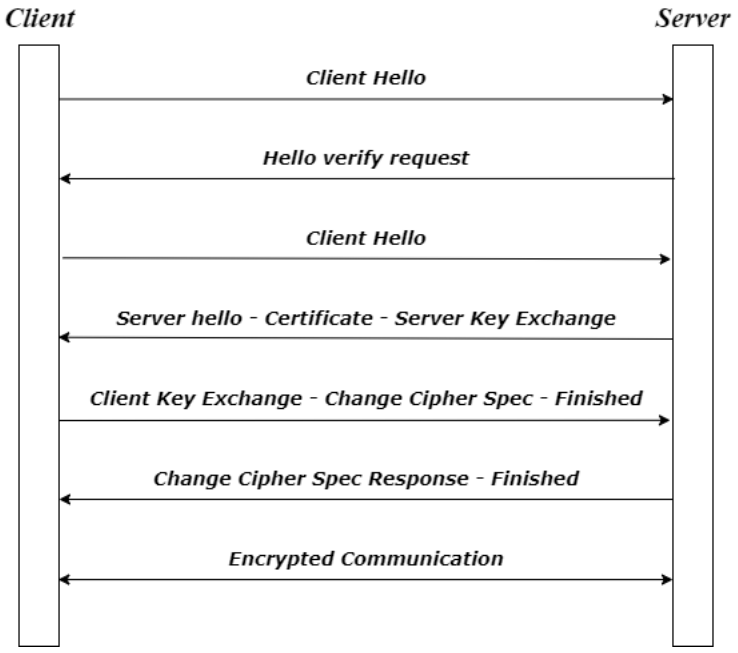

Figure 2.3 DTLS 1.2 Handshake

Figure 2.3 shows an overview of the different steps of the DTLS 1.2 handshake which executes in 3 round trips. The first round trip contains 2 steps, the **Client Hello** where the client asks to initiates the handshake, and the **Hello Verify** request where the server gives the client a session cookie. It is also during the **Hello** phase that the client tells the server of which algorithms he supports. In the **Server Hello message**, the server indicates which parameters he chose from the supported list provided by the client. The main parameters that will be negotiated are part of the cipher suite which indicates the cryptographic algorithms used for key exchange, encryption, and message authentication. The algorithms used for encryption are symmetric algorithms, usually AES.

For the second round trip, the client starts with another **Client Hello** but with the session cookie added to the request. Upon reception of **Client Hello with the cookie**, the server will initiate the key exchange process. First, the server will send its certificate for authentication. It might also initiate the handshake with the **Server Key Exchange** message. The **Server Key Exchange** message is an optional step for key exchange mechanisms which requires public key information from the server. For ephemeral variants of the protocol, the server will also generate a key pair to use for the key exchange before initiating the **Server Key Exchange** message. The server will also sign the key exchange parameters with its signing key.

For the third round trip, the client receives the certificate and verifies it. After authenticating the server, the client will then generate its part of the secret value to use between the two

parties. For RSA, or other encapsulation mechanisms, the client will generate a secret which he will encrypt using the server's public key received in the previous message. For other algorithms based on Diffie-Hellman (DH) like ECDH, the client will generate the secret value by using the server's public key. After that, the client will then send its public parameters to the server so it can generate the secret. Afterwards, the client will end its final phase by sending a **Change Cipher Spec** message which indicates to the server that the client will start to use encrypted messages using the parameters on which they agreed on. In some cases, the server can require that the client authenticate himself to the server in which case the client would also send his certificate to the server. However, we will not use this scenario in our experiment.
Finally, when the server receives the **Client Key Exchange**, it will then compute the secret value on its side using its private key and the information received from the client. The secret value will then be used to generate the key for the symmetrical algorithm. When receiving the **Change Cipher Spec message**, the server will send a response to acknowledge that they will both use the newly decided security parameters. After the response is sent, both the client and the server can now send each other encrypted messages [15].

2.2.2 Key Encapsulation Mechanism

A Key Encapsulation Mechanism is a cryptographic operation used to exchange a cryptographic key by encrypting it using your peer's public key. This means that during a handshake protocol, using a KEM will mean that the actual key that will be exchanged will be on the communication channel, but in an encrypted form. That is contrarily to key exchange algorithms where both parties use their peer's parameters to generate the same key without sending the actual final secret [26, 27].

2.2.3 Perfect Forward Secrecy

Perfect Forward Secrecy if a security feature that states that if one communication is compromised, all the other past communication would still be safe. While this is a great security feature, this increases the computational load as the protocol needs to regenerate a new key for each new sessions between the server and the client [28, 29].

CHAPTER 3 STATE OF THE ART

The chapter will present the most relevant existing works related to our research project. This presentation is organized in tree sections. In the first, we will present an overview of the theoretical research that has been done over the last few years on post-quantum algorithms. This part will present multiple categories of algorithms as well as specific implementations within these categories. In the second part, we will showcase recent experiments conducted in environments similar to the one implemented for our research, focusing on post-quantum handshakes. The third part will display a comparison between these recent experiments and this research.

3.1 Theoretical Review of Post-Quantum Cryptography

Post-quantum cryptography is the field of cryptographic algorithms that are currently considered resistant against an attack from a large scale quantum computer. Multiple types of quantum resistant algorithms have been proposed for standardization. Here we are reviewing the different category of algorithms as well as their characteristics. We will also describe the main candidates that belong to each category. We decided to review all the algorithms that were considered during the third round of selection process from NIST. While there is now a few of these algorithms that are officially going to be standardized, reviewing the candidates from the third round remain important as more selection processes are ongoing. This section will review the details of these algorithms that were drawn during the third round of selection. We will also specify is these algorithms have been selected for standardization or if they are still in the run for NIST's fourth round of selection [21, 30, 31].

3.1.1 Lattice-based cryptography

Cryptographic primitives based on lattices are using lattice problems to construct the security of their algorithm. The most known lattice problem is the shortest vector problem which defines the need to find the shortest vector in a lattice. Lattice-based cryptography is the most popular type of cryptography in the NIST call of standardization. Indeed, three of the four algorithms considered for the third round for the key encapsulation standardization are lattice-based cryptographic primitives. The main reason is that these algorithms offers great computational performance, although they generally need larger key sizes to offer as much security as currently used algorithms. Another issue with these algorithms is that most of

them are still quite young. This means that we are not as certain of their security as other algorithms that have existed for around 43 years such as RSA. These algorithms are usually based on known security assumptions over lattices like NTRU, learning with error (LWE), or even module learning with error. The various lattice-based algorithms use different types of lattices with some examples being ideal lattices or cyclic lattices. Although this presentation will focus solely on algorithms that involve key encapsulation and digital signature, it is worth noting that lattice-based cryptography can also be applied to hash functions. However, since hash functions are not vulnerable against large-scale quantum computers, we will not be describing these algorithms [31–33].

NTRU

This algorithm is the oldest of the lattice-based finalist of the NIST call for standardization. NTRU also has a great performance like the other lattice-based algorithms but has a slower key generation than most other candidates based on lattices. While most lattice-based cryptographic algorithms are based on the Learning With Error (LWE) and the Ring Learning With Error (RLWE), both known problems in lattices, NTRU is based on what is called the "NTRU assumption" [34]. The fact that this algorithm is the oldest lattice-based candidate and that fact that it is based on the NTRU assumption, a stronger difficulty assumption than the other candidates are the main reasons why the NIST kept NTRU in the race despite its slightly lower performance. It is also the only algorithm in this list that is standardized by the IEEE. This algorithm can do encryption, decryption and key encapsulation, which has been implemented in the NTRUEncrypt cryptosystem. While being a key encapsulation candidate, NTRU also defines a signature scheme. However, its implementation, also called NTRUSign, was not kept has a candidate for the call for standardization as it was outperformed by the other candidates. The candidates were evaluated by their security, cost and performance, and algorithm and implementation characteristics. It also good to know that only pqNTRUSign was considered as a candidate since the first version NTRUSign proposed was found to be insecure. One of the main issues found in the original NTRUSign was that the digital signature could leak information about the private key. Information leakage is a concept from informational theory were an attacker can extract bits of information from a message while listening on a channel. This means that an attacker listening to a channel where signatures are sent using the original NTRUSign implementation would get more and more information about the private key used to sign the messages or certificates. With enough message, the attacker could then guess the private key and forge the signature. This algorithm is however not being considered for standardization anymore [30, 34, 35].

SABER

SABER is another lattice-based cryptographic candidate for NIST call for standardization. This algorithm is also a Key Encapsulation Mechanism. The security of SABER is based on the Module-Learning with Rounding problem (M-LWR). Another specific property of SABER is its choice of moduli. All the integer moduli are chosen to be powers of 2. This allows to skip some of the steps usually executed in a lattice-based algorithm like the uplifting step. The fact that it uses power of 2 and that it can skips some of the usual steps allows faster execution. Finally, another specific implementation choice is the module. The module is an algebraic structure, an additive abelian group. A lattice is actually a specific module over a ring. In SABER, the authors use specific versions of module allowing some interpolation over original LWE/LWR problems. This optimization lowers the computation complexity and the bandwidth compared to basic lattice-based algorithms. Its key sizes are around 8000 bits or more for similar security than the algorithms we use today. Three variants are proposed: lightSABER which offers 128 bits of classical security, SABER which offers 192 bits of classical security and fireSABER which offers 256 bits of classical security. This algorithms was also not kept in the run for standardization for being too similar to other lattice-based candidates [21, 30, 36, 37].

CRYSTALS

CRYSTALS is a lattice based cryptographic suite that is going to be standardized by NIST. Its letters stand for Cryptographic Suite for Algebraic Lattices. CRYSTALS, like NTRU, defines 2 different algorithms: one for key encapsulation called CRYSTALSKYBER and another one for signatures called CRYSTALSDILITHIUM. CRYSTALS offers great performance like most lattice-based algorithms and has similar key sizes than SABER. Its main difference is that its security is based on the Module Learning with Error problem (M-LWE) instead of the Module-Learning with Rounding problem. Also, in contrary to NTRU, both algorithms from the CRYSTALS suite are still in the race for post-quantum standardization for their own category of algorithm [21, 30, 38, 39].

NTRU prime

While all the third round finalists for lattice-based Key Encapsulation Mechanisms have been presented here, the NIST had kept a few of other algorithms from previous rounds for a subsequent standardization for post-quantum algorithms. While NTRU prime was not kept after the third round of standardization, we will still be presenting its aspects.

The NTRU Prime algorithm defines two different key encapsulation mechanisms. The first one called Streamlined NTRU Prime is based on the NTRU problem, similarly to the plain NTRU candidate for KEM. The second one called NTRU LPRime is based on the Ring Learning with Rounding problem. This means that Streamlined NTRU Prime KEM is closer to the NTRU submission, while the NTRU LPRime is closer to the SABER algorithm. The actual attempt of the NTRU Prime implementation is to reduce the attack vectors possible on existing cryptographic system. To do so, the authors use what is called an irreducible noncyclotomic polynomial. By doing so, they reduces the possible attacks on NTRU and RLWE and some of the already known weaknesses of these problems have been eliminated. While this algorithm provides great performances like all the other lattice-based candidate, the fact that it is using cyclotomic structures had made him an alternate candidate for key encapsulation during the third round of submission. Indeed, cyclotomic structures are not very known in the cryptanalysis world. However, this candidates was not kept in the latest submission round [21, 30, 40].

FALCON

This lattice-based algorithm implements a post-quantum signature scheme using NTRU lattices. This algorithm uses a Gaussian sampling called GPV and implements what the authors call a Fast Fourier sampling. Its fast performances are mainly caused by the implementation of a recursive Gaussian sampling algorithm using a tree as a data structure. The author called this tree the Falcon tree. The main downside of this signature scheme is that it is really complicated to implement. It is also currently vulnerable to side-channel attacks, a type of attack where an attacker tries to crack an algorithm by collecting data on the physical system where it runs. An example would be a case where someone could do an acoustic cryptanalysis by listening to the sound emitted by a system. If an algorithm would be vulnerable to this type of side-channel attack, he could find the secret encryption key using the collected acoustic data.
The Falcon signature scheme is now confirmed to be one of the algorithms that is going to be standardized by NIST [21, 30, 41].

3.1.2 FrodoKEM

FrodoKEM was another alternate candidate for key encapsulation during the third round of standardization. Its security is based on the difficulty of the plain Learning With Error problem. The plain Learning With Error is one of the most known, studied and analysed problem. This means that FrodoKEM is based on a very strong security assumption. While

maybe being the lattice-based algorithm with the strongest security assumption, FrodoKEM is also the lattice-based algorithm with the worst performances over computational cost, key sizes and bandwidth usage. Because of these characteristics, FrodoKEM is was kept as an alternate candidate in the third round because one of the goals from NIST's process of selection is to standardize algorithms suitable for as much devices as possible. This means that performance is an important cirteria. However, FrodoKEM is a very strong candidate for implementation scenarios where the security is more important than the performance of its implementation.
It is also good to note that FrodoKEM is one of the 2 algorithms selected for post-quantum applications by the German Federal Office for Information Security (BSI). FrodoKEM was however not kept by NIST for standardization, nor for their fourth round of standardization [21, 27, 30].

3.1.3 Isogeny-based cryptography

Isogeny-based cryptography regroups cryptographic primitives using supersingular isogeny graphs which is another mathematical entity useful for public key cryptography. These algorithms use arithmetic properties of elliptic curves over finite fields. Elliptic curves are the same mathematical entities used for the classical ECC key exchange algorithm [42, 43].

SIKE

SIKE is the only cryptographic candidate for NIST standardization process that is base on isogenies. While it was kept in the run for the fourth round of standardization, SIKE is now considered insecure and should not be used [44].the candidate was an interesting choice for a replacement of our current systems for multiple reasons. The first ones were related to its mathematical properties and the implemented algorithm. Indeed, SIKE uses Elliptic curves, a mathematical entity already widely used for key exchange. The fact that SIKE uses pseudo-random walks on supersingular isogeny graphs of curves made it quantum resistant, in contrary to currently used Elliptic curves used. Another reason is that the actual implemented algorithm in SIKE is the Supersingular isogeny Diffie–Hellman key exchange (SIDH). Diffie-Hellman is the current algorithm used in most cryptosystem for key exchange. This means that SIKE implements more or less the same algorithm used today for key exchange with the difference of the mathematical entities used for the key exchange. SIKE is a specific implementation of SIDH and defines 2 algorithms: SIKE.PKE for Public Key Encryption and SIKE.KEM for a Key Encapsulation Mechanism. SIKE was also the post-quantum candidate with the smallest key sizes needing at worst around 750 bytes of key.

The fact that it used Eliptic Curve made it easy to add to existing cryptosystems. Like a lot of the other candidates, the problem on which SIKE is based, finding isogenies between supersingular elliptic curves, is not well known. Therefore, the NIST wanted to give extra time to research the algorithm and it was then found to be insecure right after the candidates for standardization were announced [21, 30, 44, 45].

3.1.4 Code-based cryptography

Code-based cryptography describes cryptographic primitives based on error correcting codes. Error correcting codes are used mainly in telecommunication to control and correct error introduced in data by unreliable or noisy channels. Usually, using these codes, we can assure that our data sent over a communication channel arrive intact to the other end. One of the first error correcting code is the Hamming code introduced in 1950 by Richard Hamming while working at Bell Telephone Laboratories. Error correcting codes are also at the base of quantum communication. Indeed, one of the big challenges of quantum computing is that the quantum channels are very noisy. There is a lot of research in quantum error correction and Hamming code is also applicable in the quantum world under the name of Steane code, an error correction code in the category of CSS codes.
Code-based cryptography are secured using the Syndrome Decoding Problem. The syndrome decoding problem is an algorithmic problem that is known to be NP-hard and which describes the difficulty of decoding random linear codes. Error correcting codes and the Syndrome Decoding Problem have been known for years, which is why we can be certain of the security of code-based cryptographic algorithms [21, 46–48].

McEliece

McEliece is another candidate for post-quantum standardization of key encapsulation that was considered for the third round. While it has not been selected for standardization yet, it is still being considered for the fourth round. McEliece is the oldest of all the other candidates as it was proposed in 1978, but was left out since it was released around the same time as RSA, one the most used algorithm for public key encryption today. This algorithm is based on error-correcting codes and is fast to execute. The biggest issue with this algorithm is the need for enormous encryption keys going from 6 kilobytes to around 1 megabyte. The fact that McElice is known since 1978 is the main reason why this algorithm is still in the race for post-quantum standardization. Indeed, having existed for so long, there has been many research and cryptoanalysis of this algorithm to prove its security. The key sizes are the biggest downside of this algorithm since it makes it difficult to implement it in memory

constrained devices like wireless sensors or raspberry pies. The original implementation used what is called Binary Goppa codes. These codes belong to the class of Goppa codes, a certain type of error-correcting codes popular in telecommunication. The classical variant of McEliece, more commonly called Classical McEliece is the second algorithm to be selected for post quantum applications by the BSI [21, 30, 49].

BIKE

The BIKE algorithm was another one of the alternate candidates that were going to be considered in subsequent rounds for key encapsulation standardization. It is now seriously being considered for the fourth round of standardization. It is a code-based algorithm implementing Bit Flipping key encapsulation. It is using what we call Quasi-Cyclic Moderate Density ParityCheck (QC-MDPC) codes. Its security is based on the Quasi-Cyclic Syndrome Decoding (QCSD) and the QuasiCyclic Codeword Finding (QCCF). One important particularity of the BIKE algorithm is that it was ally designed to uses ephemeral keys to execute Key encapsulation. This means that the keys use for this algorithm should only be used once. If the keys were to be reused, an attacker could easily do a chosen cyphertext attack on Alice and Bob and decrypt the messages. However, it has now been claimed to also support static key use [21]. Also, in contrary to McEliece, BIKE's key sizes are more reasonable and are almost comparable to key sizes used in lattice-based cryptography. With its great performance, BIKE makes a strong candidate for post-quantum KEM standardization. Its weakness, like most alternate candidates, is the fact that it is still new, having been proposed in 2010. It is set to be considered for the fourth standardization round to leave more time to researchers to do security analysis of this algorithm, improving the certainty in its security [21, 30, 50].

HQC

Hamming Quasi-Cyclic (HQC) is another code-based KEM candidate for NIST's post-quantum cryptography standardization project. It was left as an alternate candidate during the third round of standardization but is now being considered for the fourth round as well. The algorithm is based on the QCSD problem and is aiming at IND-CCA2 security. IND-CCA2 security is the strongest security definition of Ciphertext indistinguishability possible. This means that the algorithm is aiming at indistinguishability under adaptive chosen ciphertext attacks. This property is a direct consequence of its low decryption failure rate. Its keys and cyphertext sizes are slightly larger than other code-based algorithm while still being favorable over algorithms using Goppa codes like McEliece. The algorithm is still also still relatively new, even newer than BIKE. This means that the next few years will be used to increase our

confidence in HQC's security [21, 30, 51].

3.1.5 Multivariate cryptography

Multivariate cryptographic primitives are asymmetric cryptographic algorithm based on the difficulty of solving multivariate polynomials over a finite field to ensure its security. Multiple candidates were first proposed in 2016 based on these schemes, but only 2 remains in the course during the third: one finalist for digital schemes standardization, and another alternate candidate from the same category. Both candidates were competing against each other. Multivariate cryptography describes mechanism for digital signatures and key encapsulation. However, their digital signatures are usually more popular and no KEM algorithms made the third round for standardization [21, 52, 53].

Rainbow

Rainbow was the third and last digital signature candidate alongside FALCON and DILITHIUM for the third round of standardization. However, it has not been kept in the run following multiple attacks that have been found. [21]This algorithm is based on what we call the Unbalanced Oil-Vinegar (UOV) signature scheme. Algorithms based on the UOV scheme are usually very efficient. The main downside is that it requires very large signing keys. Since Rainbow is based on this same signature scheme, it inherits these same properties. Other aspects of this algorithm are its very small key sizes and its fast signature verification. The fact that is key sizes are overly large makes it difficulty to implement it in long certificate chains, as their size will grow the longer the chain will be. Nevertheless, it makes a great candidate for applications that requires a lot of signing and verifying, but not a lot of certificate and key creation. During the second round, Rainbow was competing with another multivariate signature scheme called GeMSS. However, Rainbow was kept as a finalist and GeMSS as an alternate candidate because of Rainbow's smaller key sizes and easier implementation. Before the attacks were discovered, one of the big ambush in its standardization, besides of a need of better parameters management, was the fact that it is not royalty-free, which restrained its implementation possibilities [21, 30, 54, 55].

GeMSS

GeMSS is another digital signature scheme based on multivariate cryptography. This algorithm was kept as an alternate candidate for standardization during the third round but was not kept for the fourth round for similar reasons as Rainbow. This algorithm is based on the

Hidden Field Equations (HFE) proposed by Palardin in 1996. The HFE is one of the best known and most studied multivariate schemes. This algorithm actually uses a variant of the HFE called HFEv-. This variant has increased security off the original equations, making it an interesting candidate for digital signature. The strongest argument of this algorithm is that it has the smallest signature of all the other NIST candidates. However, its large key sizes and poor signing performances makes it difficult to implement in constrained devices. Since this algorithm is competing with another algorithm based on multivariate cryptography, the Rainbow algorithm, and since GeMSS performances and key sizes are worst than Rainbow, NIST has decided to keep it has an alternate candidate for possible future standardization. The plan was that if Rainbow was found to not be suitable for standardization, because of royalties-related reasons maybe, GeMSS would have been the next multivariate-based candidate to be considered for standardization. However, this did not happen [21, 30, 56].

3.1.6 Hash-based Cryptography and zero-knowledge proof

Hash-based cryptography might be the most generic category of cryptographic primitive presented here. It defines every cryptographic algorithm which based its security on the security of hash functions. While multivariate cryptography **mostly** defines signature schemes, Hash-based cryptography only defines digital signing algorithms. Hash based signature are approximately as old as public key cryptography itself, meaning these algorithms are among the most secure digital signing schemes there is. The biggest downside of these signatures is that they can only stay secure of a specific number of signatures. The limit on the number of signature can be increased, however the resulting signatures will grow larger as this number grows.

Another unique process presented here is one where it is possible to construct a cryptographic primitive using what is called the zero-knowledge proof where one side of a party can convince the other without divulging the actual secret. One algorithm developed by Microsoft is based on this type of security [21, 57–59].

SPHINCS+

SPHINCS+ is the only hash-based algorithm that was proposed for NIST's standardization. It was listed as an alternate candidate for post-quantum digital signature in the third round but since then has been selected for standardization. This algorithm regroups 2 hash-based schemes: Winternitz One-Time Signature Plus (WOTS+) and Forest of Random Subsets (FORS). This algorithm is interesting because most algorithms that we presented here have a generally fast execution with larger key sizes than what we currently use. This is not the case

here. Indeed, SPHINCS+ public key sizes are between 32 and 64 bytes, which is smaller than what we use today. However, its signing time is extremely long, and its signature size is very large. Having poor performance and large signatures, it would make it nearly impossible to add SPHINCS+ to some existing protocol. However, in the goal of standardizing algorithms using primitives other than lattices, NIST decided to select SPHINCS+ for standardization [30, 60].

Picnic

The Picnic algorithm is the only algorithm presented here that is not based on number-theoretic, or structured hardness assumptions. This algorithm is based on the zero-knowledge proof previously mentioned. Picnic has been placed in the hashbased section of this review because it has performance similar to SPHINCS+, meaning smaller keys but slower signing and larger signatures. The secret of the algorithm is in the way its messages are signed for the zero-knowledge proof. Without going too much into the details, this special way using hashing is so that only someone knowing the actual secret key could produce the proof. Picnic has been developed by researcher from diverse university in collaboration with Microsoft. It is also open source and the library is provided under the MIT License. It is the youngest algorithm here. Indeed, the article "Post-Quantum Zero-Knowledge and Signatures from Symmetric-Key Primitives" which first defined this signature scheme was published in 2017. The strongest argument of Picnic is that it only relies on assumptions about symmetric primitives making it a unique choice. Nevertheless, the algorithm is quite young and still evolving. Therefore, it was the last alternate candidate for post-quantum digital signature during the third round. The NIST wanted to give time to Picnic's algorithm to mature and improve its performances before considering standardizing it. However, it has not been kept for consideration for standardization as NIST felt its security was not better than SPHINCS+ which has a more mature design [21, 30, 61].

3.2 Relevant Past Experiments

In this section, we will be examining the current state of the art around post-quantum experiments that are relevant to this research. We will begin by examining the research associated with the development of the liboqs (Open Quantum Safe library) fork of OpenSSL, as our own research directly builds upon their achievements. We will continue with post-quantum experiments on constrained devices and IOT since they share similar challenges as avionic systems. We will follow with a review of cybersecurity experiments in avionic communication. While the focus of this research is on a specific avionic protocol, other

studies have explored securing various avionic protocols, particularly since many of them lack encryption measures. Finally, we will present how this work differs from the current experiments that have been done and presented here.

3.2.1 Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH

In [1], the authors created a prototype of TLS1.3 using post-quantum KEM, authentication and a prototype of TLS1.2 using only post-quantum KEM algorithms. The result was the liboqs fork of OpenSSL which is used in this research. For our project however, we use their work to add support for DTLS1.2 as well. Also, while this paper was mostly about the implementation successes, the failures and the lessons learned during the prototyping, our work kept their foundation and evaluates the performance of post-quantum handshake in DTLS1.2 [13].

3.2.2 Post-quantum experiments and constrained devices

Because avionic shares similar constraints as IOT devices and embedded systems, we will show recent works that studied the performance of post-quantum algorithms while prioritizing especially the ones that explored these types of devices.

PQC Benchmark on TLS

In [22], the authors used the liboqs OpenSSL fork to achieve a complete benchmark of 3 PQ KEM algorithms and 3 PQ signature algorithms in a simulated network. This study is of particular interest as they utilize the same library as our own research. Also, TLS is the TCP equivalent of DTLS, the protocol we will be using for this research. However, their experiments do not cover all algorithms supported by liboqs, and only covers one security level for each candidates. Furthermore, while we are also simulating the network part of the protocol, we will be simulating in a different way to emulate air-ground communication, while the authors simulated a communication data-center-to-data-center going through the internet .

PQC on embedded systems

While there has been many studies that have been done to investigate the performance of PQ algorithms, the ones that interest us the most are the ones that have been made using constrained devices since they have similar limitations has avionic hardware.

In [62], the authors benchmarked 2 KEM and 2 signature algorithms that were NIST round 3 candidates. They evaluated the speed, memory requirements and communication size during a TLS 1.3 handshake on an ARM Cortex-M4 system with an local Ethernet connection. The integration was made using the wolfssl library.
In [63] on the other hand, the authors used their own implementation of a post-quantum enhanced DTLS 1.2 using NTRU. The handhsake was between a client an a sensor in a Wireless Sensor Network with a gateway in between. Sensor tags were used for the WSN and Raspberry pi 3 boards for the gateway and the clients. While they evaluated only one algorithm, they recreated life-like conditions to benchmark DTLS executing on cyberphysical systems.
In [64], which was an update to [62], the authors tested the performance of PQ algorithms on raspberry pies on ARM Cortex-M4. While they did not use the same library as us, they have one of the most complete set of algorithms that have been tested here. However, one key difference with our work is that the authors did not consider hybrid algorithms.

3.3 Cybersecurity in avionic communication

While the idea of adding post-quantum security to avionic communication is very new, people have been exploring multiple ways to add cryptography to avionic for a long time. In [65], the authors investigated the performance impacts of adding secures channel in Avionics Wireless Networks. Similarly to our research, they tested the performance of an asymmetric cryptosystem and the total time before a session key could be established. However, they compared different protocols instead of comparing different algorithms like we did. They also used some protocols that are executing over TCP, which has a bigger overhead than UDP. Finally, AWN are networks of components within the aircraft while the protocol we are testing, ATN/IPS are for communication between an aircraft and the ground stations.

3.3.1 post-quantum in avionic

The work in [66] is one of the few studies which explored the addition of post-quantum cryptography in avionic. While their research was not performance oriented like ours, they were able to implement a post-quantum algorithm into an actual aircraft. LDACS being another type of technology used for air-ground communication, it makes this achievement even more exciting for the possibility of realising the same experiment for ATN/IPS in the future. Even more interesting is the fact that they had chosen McEliece as the algorithm, which is especially known to have very large keys.

3.3.2 DTLS in ATN/IPS

The research by Niraula et al. closely aligns with the scope of our study [5]. They provided a comprehensive analysis of utilizing DTLS for securing VDL-M2 air-ground communication. They also used OpenSSL on the ground station but used wolfssl on the avionic hardware. While we will be reporting our performance for VDL-M2 as well, we will also report delays for HF links. The main difference with our research is that while we did not have access to real avionic communication systems, we will be testing post-quantum algorithms instead of only ECC. We will also be reporting performances for HF and not only VHF. However, we will be able to compare our results to theirs to see the accuracy of our emulation.

3.4 Our contribution to the state of the art

Initially, we examined the key research article that formed the basis of our study. This research paper was the foundation of the liboqs support for TLS1.3 handshake [13]. Following that, we cited multiple works where performance study has been made using post-quantum algorithms [22]. We especially showed research that used constrained environments due to the fact that these have similar limitations as avionic systems [62–64]. Some studies used their own implementation of a post-quantum variant of TLS/DTLS while some used the liboqs project to evaluate the performance of TLS 1.3 with post-quantum algorithms. We also presented one article with a scope similar to this thesis with a DTLS implementation for ATN/IPS. However, the authors did their evaluation using a limited set of algorithms [5].

However for this thesis we modified the liboqs OpenSSL fork to support post-quantum algorithms for openssl's DTLS 1.2 implementation which, to our knowledge, as never currently been done before. Most current research with post-quantum and DTLS on constrained devices are using a limited set of algorithms and variants. By customizing the liboqs integration into OpenSSL, we have the capability to extensively test numerous post-quantum algorithms and their variants on DTLS 1.2. Similar to various IoT experiments, we will also showcase an asymmetric setup involving communication between a constrained device and a more capable server. However, we specifically chose the hardware for the DTLS client to be of similar performance to what avionic systems would have which differs from common IOT experiments with PQC. Finally, most setups used either real network conditions or a network simulator to evaluate the transmissions delays. However, we will present our way to emulate the total delay for air-ground communication by using a Simulink simulation model. In summary, this research aims to contribute to the existing body of knowledge by providing additional per-

formance data on post-quantum DTLS1.2 handshakes using OpenSSL. It will also include an emulation of an avionic environment for post-quantum handshakes, offering valuable insights into the performance of these algorithms. Furthermore, the research will present performance data on various post-quantum algorithms running on future avionic hardware. In the end of this research, we will have showcased a performance study of the integration of post-quantum algorithms to an infrastructure that relates to the future ATN/IPS network.

CHAPTER 4 METHODOLOGY FOR POST-QUANTUM SECURITY ALGORITHMS ASSESSMENT

In this chapter we will describe the different components used to realize our experiment as well as how these components relate to avionic communications. We will start by describing the hardware and the operating systems that were used to run post-quantum algorithms and the DTLS handshake. We will then describe the libraries and softwares that were used to test the performance of our implementation. From there, we will explain the changes that were necessary to realize our experiments. Finally, we will describe how we used Mathlab to create a simulation model that can estimate the transmission delay for air-ground communication in function of the number of bytes to send.

4.1 General view

In the case of ATN/IPS, an air-ground communication is typically a communication between an aircraft in flight and a ground-based Air Traffic Control system. Using this scenario, we are using a card with constrained resources to represent the performance constrained hardware in avionic. We also use a standard PC to represent the ground-based system.
It is currently planned that ATN/IPS will use DTLS to secure the air-ground communication [5]. DTLS is a protocol that helps secure communication over UDP by using a handshake to establish session keys. After the key is established using asymmetric cryptography, the communication is secured using symmetric encryption. This is the most common way to encrypt a communication channel since asymmetric cryptography is way more resources consuming than symmetric cryptography. Here we are concentrating our efforts on the handshake part of DTLS, since symmetric encryption should barely be impacted by a large-scale quantum computer.
In a DTLS handshake there is usually 2 parties, one acting as the client and one acting as the server where the client will be the one initiating the handshake with the server [15]. Figure 4.1 shows the parallel between our setup and the avionic scenario.

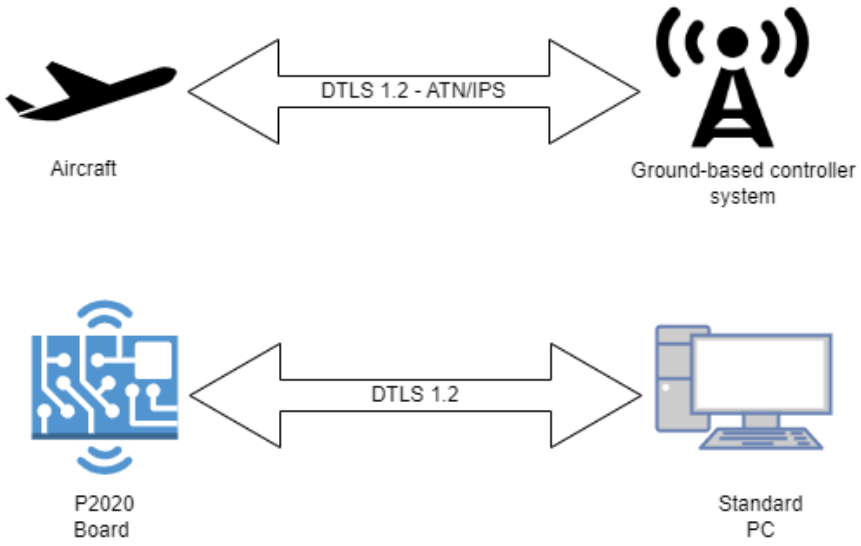

Figure 4.1 General Setup

For this research, we are planning to replicate a DTLS handshake executing in the ATN/IPS network. To do so, we will first modify the liboqs fork of OpenSSL to add PQ capabilities to its DTLs implementation. The choice of using OpenSSL has been made because of the fact that this is one of the most used cryptographic libraries in the world. More specifically, we will add the capability of using post-quantum KEM algorithms for the key exchange phase of the DTLS handshake as well as the capability of using post-quantum certificates and signing keys for authentication and integrity. By using the liboqs fork of OpenSSL, we already have access to post-quantum algorithms within OpenSSL as well as an implementation for TLS 1.3 which we will use as an example [13].

Subsequently, we will create a Mathlab model to simulate air-ground avionic communication delays. Using these 2 components, we will first execute multiple DTLS handshakes using both classical and PQ algorithms. One of the measurement that will be taken is the total size of the handshake over the network. This will then be used in our simulink model to simulate air-ground transmission delays. We will also measure the CPU times for both the client and the server during the handshakes to finally estimate a total completion time for DTLS handshakes inside ATN/IPS. This will be done by adding both CPU times to the simulated transmitted delays for each algorithm. In the end of our research, we will have emulated the total completion times for DTLS handshakes inside a simulated ATN/IPS for both PQ algorithms and classical algorithms for comparison. This final results will contain information on hardware performance, bandwidth usage and transmission delays and these will be used to determine the best algorithms to use in ATN/IPS as well as the overall impacts of using PQ over classical algorithms.

Apart from the DTLS experiments, we will also be using the liboqs speed benchmark tool to report the overall performance of PQ algorithms on avionic hardware. While this won't help evaluate the impacts of switching to PQ algorithms from classical algorithms, these results will help the selection of which algorithm(s) will be preferable in an avionic context like ATN/IPS.

4.2 P2020 card with Yocto

We selected this card considering what will be recommended to be used for future aircrafts [67]. While this means the card won't be as constrained in resources as the legacy systems used in some older aircrafts, this allowed us to test more algorithms. This also means that the results will be directly applicable to future avionic systems.
Our card is based on a QorIQ P2020 processor which is a system-on-chip (SoC) produced by NXP. It has two e500 Power Architecture cores running at up to 1.33 GHz. The e500 core is a 2-bit RISC-based microprocessor architecture. It also has 1gb of RAM and uses flash memory for storage.
The Yocto project is an open source initiative that can help create custom Linux distributions. This makes these tools useful when doing Linux-based experiments on embedded devices with specific requirements that cannot be met with standard Linux distributions.
Using the Yocto project, we were able to create a custom 32-bit Linux distribution for our board, as well as a crosscompiler to build our binaries for this specific environment [68].

4.3 Standard Linux PC

The PC is running Ubuntu 22.04 with a CPU Intel I5-8400 with 5 cores at 4.0 GHZ. It also has 12 GB of DDR4 RAM and a 500GB SSD. This PC will be used to replicate the ATC's PC which won't have the same hardware restrictions as the avionic system. This is an important aspect of this setup because the client and the server won't execute exactly the same operations in a DTLS handshake.

4.4 Tcpdump

Tcpdump is a packet analyzer tool that can capture all packets going in and out of a network interface. We have an instance of the tool running on both the card and the PC to capture all the packets sent during the handshakes. Using filters, we are only capturing the packets that

are being sent **between** the PC and the card. By doing so, we will be able to simulate the transmission delay for air-ground communication using Mathlab models after the handshake executed by recording the total size of the communication. We also used Tcpdump to validate that the handshakes are properly executed and does not have any bugs after our changes [69].

4.5 Liboqs and OpenSSL

The Open Quantum Source project (OQS) is an open source project which aims to help with the prototyping of projects using post-quantum cryptography. The project is an initiative of the University of Waterloo and was started by Michele Mosca and Douglas Stebila. The main project of this initiative is called liboqs. The liboqs library offers a common API to use with a large collection of post-quantum algorithms. The actual implementation of the algorithms either comes directly from the team that are working on them, or the PQClean project. The liboqs library also has a test framework already implemented which makes the benchmark of post-quantum algorithms on different platforms simple.
Their second biggest project is the liboqs fork of OpenSSL. Openssl is one of the biggest and widely used cryptographic libraries which is why we chose to use its liboqs fork for our experiments. The openssl library implements multiple cryptographic algorithms but also multiple cryptographic protocols. More specifically, the version of the forked OpenSSL has already implementations of TSL going up to TLS 1.3 and DTLS up to DTLS 1.2 with classical algorithms. Furthermore, the liboqs fork of OpenSSL also supports TLS 1.3 using post-quantum algorithms. [1]

4.6 Implementation and methodology

While Liboqs and its OpenSSL fork offered multiple tools that helped us achieve our goals, one of the features it was still lacking was the support of post-quantum algorithms for DTLS 1.2. Currently, the fork only supports post-quantum algorithms for TLS 1.3. The protocol is supported through the use of the s_client and s_server applications. While we were able to re-use a lot of the code that was already in place for our TLS 1.3, a few changes were necessary because of fundamental differences between DTLS 1.2 and TLS 1.3. Due to the high similarity between TLS and DTLS protocols, their version numbers not only correspond in value but also reflect consistent changes across both. In other words, DTLS 1.2 is similar to TLS 1.2 and DTLS 1.3 is similar to TLS 1.3. Because of this, if OpenSSL supported DTLS 1.3, adding support for post-quantum algorithms to it would have been easier than what it

has been to DTLS 1.2 due to the current support of PQ algorithms in TLS 1.3. Indeed, there are 2 very important changes between D(TLS) 1.2 and D(TLS) 1.3. The first is that D(TLS) 1.3 only supports ephemeral keys for the key exchange phase of the handshake. Ephemeral keys are key exchange keys that are generated during the handshake and which will only be used once for the purposed of this handshake only. Using ephemeral keys provides a higher security level to the protocol since it helps provide perfect forward secrecy with the use of the ECDHE protocol as opposed to non-ephemeral keys. This means that RSA is no more used as a Key Encapsulation mechanism for (D)TLS1.3 because it is not forward secret [29]. The second difference is removable of one round trip necessary to complete the handshake. With this difference, this meant that the post-quantum implementation of TLS 1.3 was not directly usable for D(TLS) 1.2 since the handshake steps were different [15, 25]. Figure 4.2 shows the handshake steps that required modifications.

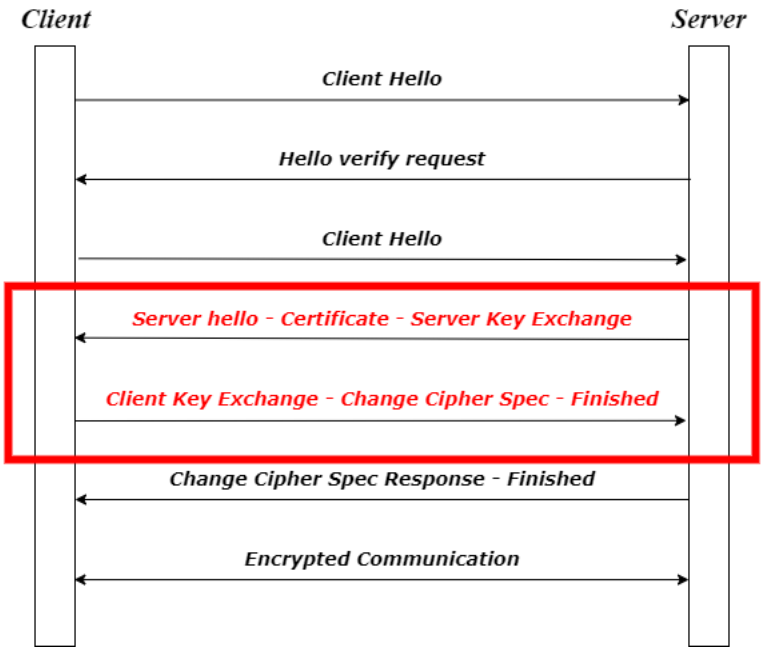

Figure 4.2 DTLS 1.2 Handshake

This last figure illustrates the primary segments of DTLS 1.2 which needed changes for our experiments : the certificate and key exchange phase and the client key exchange phase. A significant portion of the work accomplished in the liboqs fork of OpenSSL, specifically regarding post-quantum and hybrid certificates, was highly reusable. As a result, there was minimal additional effort required in that area. Most of the logic of operations that will be described here is already applicable to the liboqs implementation of TLS 1.3 that was done

in [13]. The main difference here is **where** and **when** the operations are executed.

4.6.1 Post-quantum key exchange

For the post-quantum key exchange, we are using ephemeral keys. This means that during the server key exchange phase, the server will generate its own post-quantum key pair of a KEM algorithm. The server then sends its public key after it finishes sending its certificate. The client first authenticate the server by verifying its certificate. Then it uses the received KEM key from the **Server Key Exchange** message to encrypt a secret it generated. Finally, the client sends the encrypted secret to the server and the rest of the protocol is unchanged. In this scenario, The server executes one key generation operation and one decryption operation (decapsulation). The client will execute one encryption operation. For signing operations, the server will sign the key exchange message and the client will verify both that the message has been signed by the server's certificate and that the certificate has been signed by a trusted CA(Certificate Authority).

4.6.2 Hybrid key exchange

For the classical part of the hybrid key exchange, the only classical algorithms that will be used is ECDHE. For the **Server Key Exchange** phase, the server first generates both a post-quantum KEM key pair and an EC key pair. The specific parameters for both algorithms are chosen by the client using the *"-groups"* parameter of the *s_client* OpenSSL application. To send both the public key of the PQ algorithm and the public EC parameters, the **Server Key Exchange** message will concatenate both values as shown in figure 4.3 below. In this case, the final public key of the hybrid algorithm is considered to be the EC public parameters concatenated to the post-quantum public key. For signing operation, it will execute the same operations described in the previous section [13].

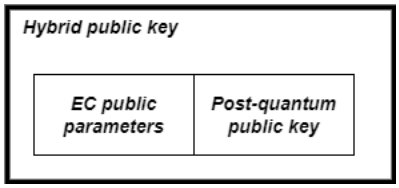

Figure 4.3 Hybrid public key (server)

Upon receiving the server's public key, the client first extract the classical key, generate its own EC key pair and the classical shared secret. It also generates a secret for the quantum

part of the key and, much like the post-quantum handshake, encrypts it with the post-quantum public key received from the server. The full secret will then be a concatenation of both the classical and the post-quantum part as well. Finally, as shown in figure 4.4 below, the client creates its **Client Key Exchange** message by concatenating its newly generated public EC parameters and the secret encrypted with the server's PQ public key.
In this scenario, the server will do one PQ key generation and one classical key generation. It will also do one post-quantum decryption and the generation of the secret using both EC parameters. The client will also do a classical key generation as well as a post-quantum encryption operation.

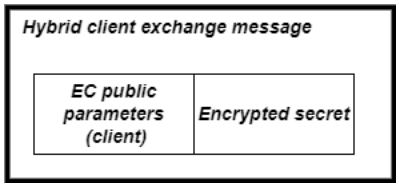

Figure 4.4 Hybrid public key (client)

4.6.3 Server authentication

To authenticate the server to the client, we used X509 certificates. The liboqs fork of OpenSSL supports the generation of X509 certificates for signing algorithms. The generation of certificates for KEM algorithms is, however, not supported and trickier since KEM algorithms cannot self-sign the certificate request since they not support digital signatures at all. While there is workarounds possible to generate X509 certificates for KEM algorithms for experimental purposes, since we decided to use ephemeral keys for our experiments, these workarounds were not needed. However, in real life scenarios, to generate X509 certificate, a self-signed certificate request is usually necessary to prove the ownership of the key. Since this method is not possible for post-quantum KEM algorithms, research is being done to explore other methods to prove key ownership to generate x509 certificates [70].
For our certificate chain, we decided to have a chain of 2 certificates to simplify the experiment. While a standard PKI would have at least three certificates, the root CA, a signing CA and the actual certificate that is used, we decided to follow the base example of the liboqs fork and simply have the server certificate be directly signed by the CA. This way, while we are still doing server authentication using a certificate chain, which is close to what would happen in a real case scenario, while skipping a level to the chain to reduce the chain size which and reduce the verification time. Figure 4.5 and Figure 4.6 shows the difference between a standard certificate chain and the chain that we used for this experiment.

Figure 4.5 Standard Certificate Chain

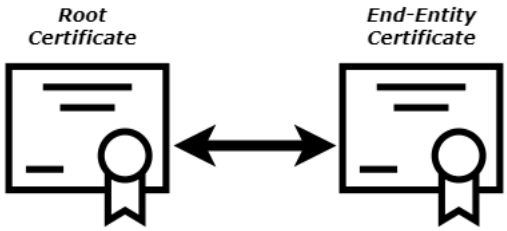

Figure 4.6 Certificate Chain Used For These Experiments

4.6.4 Client authentication

For this experience we decided to not include client authentication to simplify the experiment. Like we said in the previous sections, we focus on the implementation of KEM algorithms so we only tested post-quantum signatures on server-side authentication. Because of this assumption, we believe that testing post-quantum signatures in DTLS 1.2 solely focusing on server authentication will sufficiently fulfill the objectives of this project.

4.7 Mathlab and Simulink model

For our simulation, we used primarily the Radio Frequency (RF) toolbox that is part of Simulink[1]. The RF toolbox offers multiple tools to help model and simulate different types of wireless communication models. Using the different components provided by the toolbox, we created a model to simulate air-ground communication based on the details specified in the avionic communication section of chapter 2. The modelled system takes as input the total size of the communication between the client and the server during the handshake. It outputs a delay in a number of bytes caused by the transformation necessary to communicate these packets through RF. In 5, we will describe how we use this delay number to calculate the total transmission delay simulated by the model. We also modelled 2 different systems

[1]Simulink is a Mathlab-based tool that can help simulate dynamic systems. Simulink can provide multiple toolsets depending on the type of environment being simulated.

to simulate both HF communications and VHF communications. All the information on the different components presented here can be found in the official Simulink and RF toolbox documentation [71, 72].

4.7.1 Inputs

The input to our model is a bit stream generated by 2 blocks shown in figure 4.7: a random integer generator and an integer to bit converter. The random integer generator is an input block can be configured to send a configurable amount of integer per seconds by setting the *Samples per frame* parameter. We then use the second block to convert the output into bits to input to our modulator block. To simplify the simulation of our delays, we will run the simulation once with the *Samples per frame* parameter set to the total size of the captured DTLS handshake. More details in chapter 5.

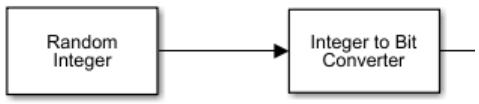

Figure 4.7 Model input

After the passing through the integer to bit converter, the bus will form a digital signal between 0 and 1 which the modulator(s) will transform to fit the desired frequency needed for air-ground communication. Figure 4.8 shows an example of a resulting signal after modulation.

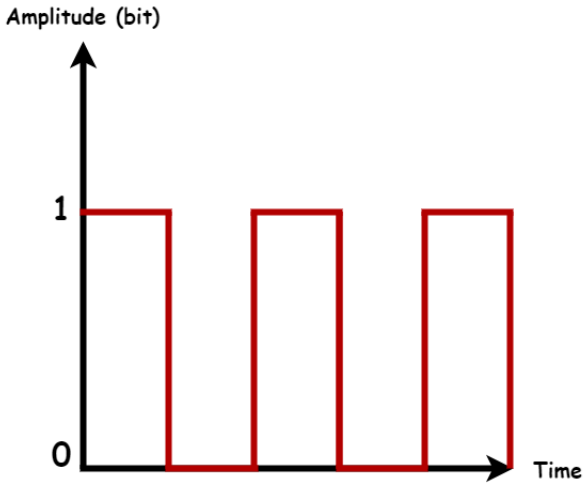

Figure 4.8 Digital Signal between 0 and 1

4.7.2 Digital Modulators

In this section we will present the 2 modulator blocks used for both simulation models. We tested our simulations with both blocks to properly simulate the HF and the VHF protocols. In this case, both modulators have the same data rates. Since we won't explicitly introduce errors in our channel, we are expecting both simulation models to produce the same results. Figure 4.8 and 4.9 shows both Simulink blocks as seen in our simulation model.

8-PSK Modulator

PSK or Phase Key Shifting is a specific method of digital modulation. 8-PSK specifically shifts the signal by 8 possible degrees to represent 3 bits with 8 possibles states: 000, 001, 010, 011, 100, 101, 110, and 111. These states are usually called *symbols*. Encoding states using 3 bits at the time allows to have higher data rates but is also more susceptible to errors. This is the modulation used for HF communication. The modulator is also the only module that was changed between our HF and our VHF model. The demodulator block is similar but simply has its input and output reversed.

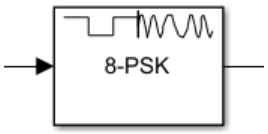

Figure 4.9 8-PSK Modulator

8-DPSK Modulator

8-Differential Phase Shift Keying is another specific method of digital modulation. Like the name suggest, it is a variant of the 8-PSK modulation where instead of representing absolute phase values, the encoding is made using the difference between consecutive symbols. The 8 possible phase differences are 0, 45, 90, 135, 180, 225, 270, and 315. This modulation scheme is usually less noisy than 8-PSK but requires more complex hardware. However, since we will not introduce noise in our simulation model, these characteristics won't matter for our experiments. Much like the 8-PSK modulator, the demodulator block is similar but simply has the input and output reverse.

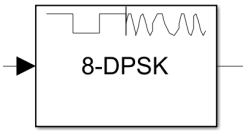

Figure 4.10 8-DPSK Modulator

4.7.3 Complex to real-image

The output of the digital modulators will be a complex signal of the form

$$I + j * Q \tag{4.1}$$

where I is the In-phase component, j is the imaginary unit, and Q is the quadrature components of the modulated signal which is a decomposed form of the modulated signal. The complex to real-image block is simply a way to split the complex signal into 2 different signal with the real and the imaginary part as seen on the Simulink block in figure 4.11. This way we will be able to create our RF signal using the IQ modulator which takes the I and the Q input separately [73].

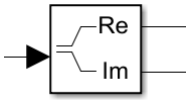

Figure 4.11 Complex to real-image

4.7.4 Inport

This Inport block converts the Simulink input signal into a signal compatible with the RF blockset of Simulink. This section block is necessary to link the Simulink part which generates the inputs and modulate the signal, and the RF part which will simulate the RF communication. Both output of the complex to real-image bloc will be the input to an Inport block that will transform both signals into a complex type before entering the IQ modulator block. The transformation can be seen on the Simulink block in figure 4.12.

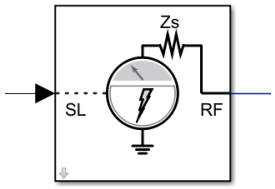

Figure 4.12 Inport

4.7.5 IQ Modulator

The figure 4.13 below shows the IQ Modulator which is the actual block that will generate the RF signal. It takes the output of separated IQ signal that went through the Inport and transforms it into a single RF signal which will be the simulated wireless communication. We can then configure the RF signal using a configuration block. In this block we also configure the signal frequency output by the block by configuring the *Local oscillator frequency* parameter. For the HF simulation, we configured it to 11 MHz and for the VHF simulation we configured it to 121.5 MHz.

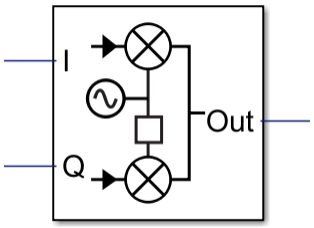

Figure 4.13 IQ Modulator

4.7.6 IQ To RF

To recapitulate these last 2 blocks, the IQ components that we as the output of the digital modulator get split into two separate signals that represent both components of the complex signal. We then pass these signal through the inport block which makes the component compatible with the RF toolset components and finally inputs the new complex signals into the IQ modulator that creates an RF signal. This connection can be seen in figure 4.14.

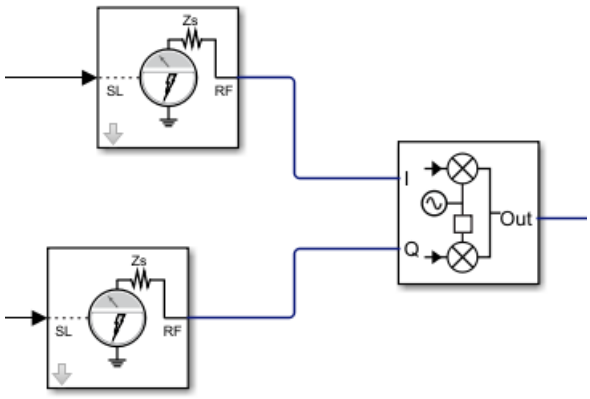

Figure 4.14 IQ to RF

4.7.7 RF Configuration

The block shows in figure 4.15 is the RF Configuration block of the RF toolset which sets the conditions of the simulated model. Each subset of blocks from the RF toolbox needs to have a configuration block to, at the very least, set the default parameters of the RF signal.

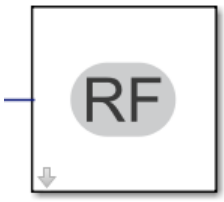

Figure 4.15 RF Configuration block

4.7.8 Outport

The Outport block from the RF toolset is the reverse block of the inport block. This is taking an RF signal as input and convert it to a Simulink signal as shown in figure 4.16. This is used here to create the reverse circuit to receive and decode the RF signal. This can also be used to introduce specific noise models between the sender and the receiver which we have not done for this experiment.

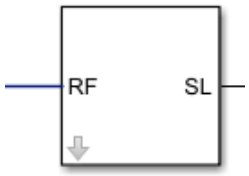

Figure 4.16 Outport block

4.7.9 Find delay

The Find Delay is also from the RF toolset and helps calculate a delay between 2 signals. This is what we will use to compute the final transmission delay of our DTLS handshakes. The block takes 2 signals as input. For our model, we are calculating the delay between the signal bus before the modulator block, and the one after the demodulator block. The output delay is a value represented in "units of samples" which in our case will be in bytes. In other words, the delay is the number of "bytes" that is being read by the simulation before the 2 signals match. Figure 4.17 shows the Simulink block. In 5, we will describe how we will be using this value to estimate a transmission delay for our handshake.

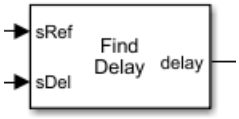

Figure 4.17 Find Delay block

4.7.10 Complete models

In this section, we present both completed models for VHF and HF simulation with all the components and links described in the previous sections. The only differences between both models are the modulator and the demodulator blocks which are using 8-PSK for the HF model and 8-DPSK for the VHF model. Figure 4.18 shows the complete module for the HF simulation model and figure 4.19 for the VHF simulation model.

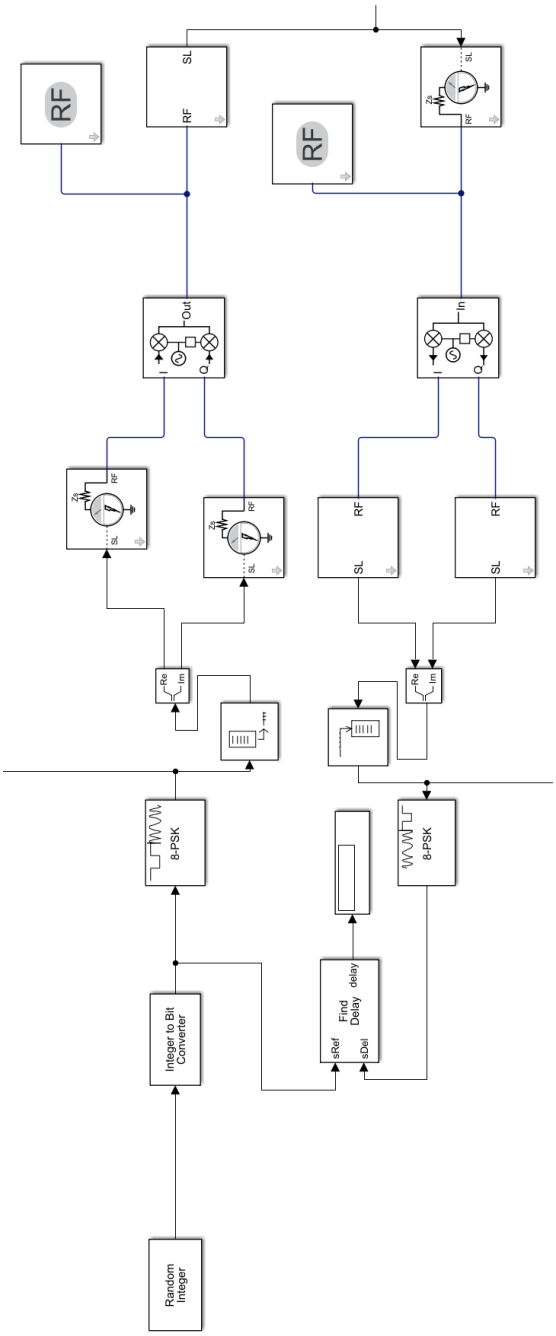

Figure 4.18 HF Simulation Model

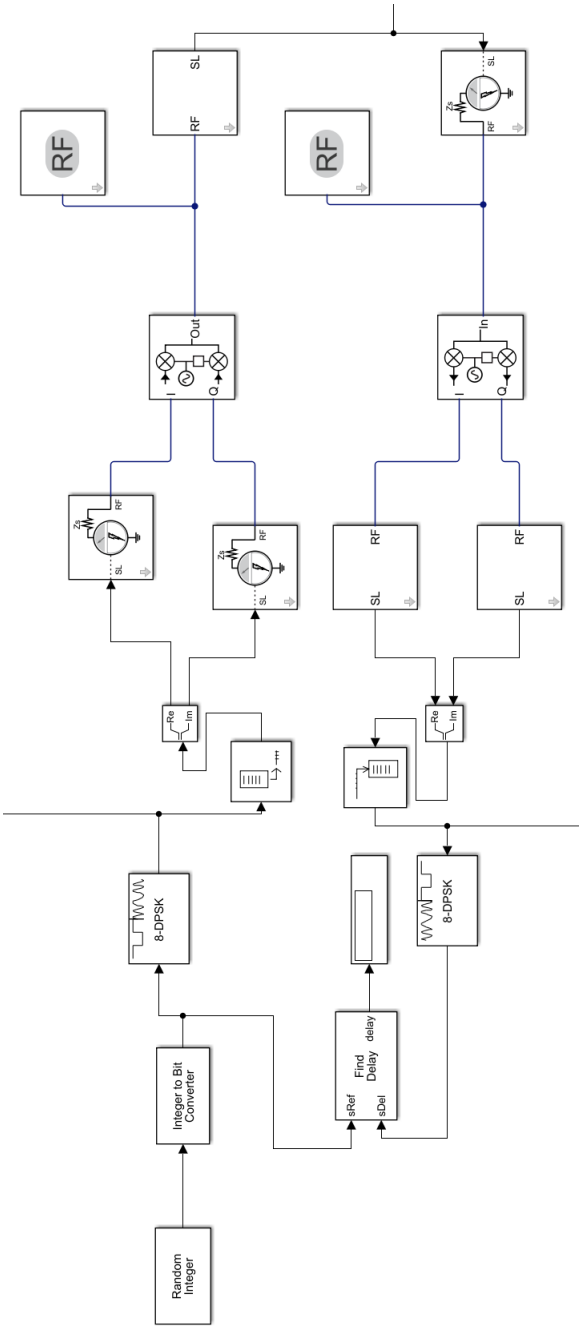

Figure 4.19 VHF Simulation Model

4.8 Discussion

In this chapter, we have described our approach to replicating the conditions of a DTLS handshake within the ATN/IPS environment. We have introduced the various components, libraries, and hardware utilized, along with the modifications made to the liboqs OpenSSL library to enable DTLS handshakes with post-quantum capabilities. Additionally, we have demonstrated the utilization of Simulink and the ref toolbox to simulate transmission delays inherent in air-ground communication during a DTLS handshake. Moving forward, the subsequent section will showcase how we leveraged this setup to validate the effectiveness of our modifications. We will also outline our testing plan, which focuses on a specific set of post-quantum algorithms.

CHAPTER 5 VALIDATION

In this chapter, we will present how we validated the functionality of our implementation. Next, we will be describing the testing plan implemented for our experiments, including the selection of algorithms that we will be experimenting with. Our plan can be separated in 3 parts. The first part involves conducting a performance benchmark of the selected algorithms. The second par of our experiment focuses on the execution of the DTLS 1.2 handshakes and the last part involves the simulation of the transmission delays.

5.1 Setup Validation

While openssl has internal processes to indicate if a handshake failed or succeeded, because of the modification we made to the liboqs fork, we needed extra validation before starting our performance experiments. To do so, we examined the handshakes carefully using packet captures and analyzed the packets using Wireshark, a packet analyzer. We were then able to see the modified handshake with the messages that we had modified. We were also able to confirm that a session was correctly established by seeing that the messages sent after the handshake were encrypted. Finally, we could also confirm that after the session was established, the messages sent from the client could be decrypted and shown correctly on the server. Following this initial validation, we were able to start our testing scenarios which will be described in the next sections. An example of a packet capture for Kyber512 is shown in figure 5.1 below.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	132.207.28.197	132.207.28.200	DTLSv1.2	323	Client Hello
2	0.000923	132.207.28.200	132.207.28.197	DTLSv1.2	90	Hello Verify Request
3	0.001143	132.207.28.197	132.207.28.200	DTLSv1.2	343	Client Hello
4	0.001866	132.207.28.200	132.207.28.197	DTLSv1.2	270	Server Hello, Certificate (Fragment)
5	0.001887	132.207.28.200	132.207.28.197	DTLSv1.2	270	Certificate (Fragment)
6	0.006273	132.207.28.200	132.207.28.197	DTLSv1.2	270	Certificate (Reassembled), Server Key Exchange (Fragment)
7	0.006288	132.207.28.200	132.207.28.197	DTLSv1.2	270	Server Key Exchange (Fragment)
8	0.006314	132.207.28.200	132.207.28.197	DTLSv1.2	270	Server Key Exchange (Fragment)
9	0.006339	132.207.28.200	132.207.28.197	DTLSv1.2	270	Server Key Exchange (Fragment)
10	0.006359	132.207.28.200	132.207.28.197	DTLSv1.2	250	Server Key Exchange (Reassembled)
11	0.006366	132.207.28.200	132.207.28.197	DTLSv1.2	67	Server Hello Done
12	0.031230	132.207.28.197	132.207.28.200	DTLSv1.2	912	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
13	0.033227	132.207.28.200	132.207.28.197	DTLSv1.2	247	New Session Ticket, Change Cipher Spec
14	0.033250	132.207.28.200	132.207.28.197	DTLSv1.2	103	Encrypted Handshake Message

Figure 5.1 Example of packet capture for Kyber512 handshake

5.2 Algorithms selected for testing

An important part of developing our testing plan was to select which algorithms to test. We decided to concentrate our effort on the algorithms that are currently selected for NIST standardization as well as the algorithms that are still in consideration for NIST's fourth round of evaluation. NIST specified that one of the main criteria for these first rounds of standardization was performance, meaning these algorithms should be the best choices for constrained environments [21].

Following these criteria, the KEM algorithms that were are planning on testing with are:

- McEliece
- CRYSTALS-KYBER
- BIKE
- FrodoKEM
- HQC

and the digital signature algorithms that also fit these criteria for our testing are:

- CRYSTALS-DILITHIUM
- FALCON
- SPHINCS+

At the beginning of our research, we were planning include the SIKE and SIDH algorithms in our experiments. However, this algorithm is not considered safe anymore after a paper was published presenting an efficient classical key recovery algorithm against SIDH and SIKE. The authors also presented their implementation and demonstrated its practicality [44]. This means that we should not consider both these algorithms for standardization in the ATN/IPS project. We first also considered to test other algorithms from the third round of NIST standardization [59]. However, the algorithms were not kept in the run for either being not secure enough anymore, not being performant enough or for needing significant change before being considered seriously [21]. In addition, other algorithms like SABER were considered too similar to Kyber since they were also lattice-based and produce similar performance. Because of this we thought it would not be worth it to test all these algorithms and we are trusting the selection process that NIST has made over the last few years.

5.2.1 Hybrid Algorithms

In the case of hybrid algorithms, the classical portion exclusively supports elliptic curve signing and key exchange algorithms, while other classical algorithms are not supported. For NIST level 3 security, the algorithms will be using the P384 curves for their classical part. Consequently, the hybrid signing algorithms combine a post-quantum signing algorithm with p384-ECDSA, while the Key exchange algorithms combine a post-quantum KEM with p384-ECDH. For instance, the hybrid variant of DILITHIUM-3 would be named P384_DILITHIUM3.

5.3 General performance

One of the advantages of using the liboqs library is that the team made some benchmarking tools available to use. These tools allow us to accurately measure the performance of these algorithms when running on future avionic hardware.
To benchmark these algorithms using liboqs, we followed the default approach of executing each operation within an algorithm for as many times as possible within 3 seconds. These parameters are the default ones presented by the liboqs benchmarking tool and we decided to keep them as they were to compare more easily to other research. The specific information that is collected are :

- Number of iterations
- Total mean time in microseconds
- Standard deviation

The total mean time is calculated by the sum of the measured time for the operation divided by the total number of iterations for a run as shown in equation 5.1.

$$\mu = \frac{\sum OperationTime}{nbIterations} \tag{5.1}$$

Where μ is the calculated mean. After calculating the mean, we can use the value obtained to compute the standard deviation as shown in equation 5.2:

$$\sigma = \sqrt{\frac{\sum (OperationTime - \mu)^2}{nbIterations}} \tag{5.2}$$

Three operations were used to benchmark each algorithms. For signing algorithms, we are benchmarking the **keypair generation**, the **signature** and the **verification** times. For KEM, we are also benchmarking the **key generation**, as well as the **encapsulation** and **decapsulation** times. All the benchmarking as only been done on the card.

5.4 DTLS experiments and performance impacts

To measure the DTLS handshakes, we divided our experiments into three categories: classical execution, KEM algorithm execution, and fully post-quantum execution. The classical

executions will employ the algorithms currently planned for ATN/IPS, while the other experiments will involve switching the algorithms used for key exchange or both key exchange and digital signing.

For all the experiments, we will first run our modified version of the openssl s_client app on the card and our modified version of the openssl s_server app on the PC. While both will communicate through WIFI, we will measure the *mean* CPU execution time for both the client and the server during the handshake, and will will add to those our simulated transmission delay. This implies that our network setup will not have any impact on this particular experiment. We will run each experiments **1000** times and compute the **mean**. The number of run was chosen to benchmark similarly to [74]. This will be used to record the impacts in terms of CPU usage as well as calculate the total emulated handshake time for air-ground communication. Additionally, we will record the total handshake size, which serves as an indicator of the impact on the required bandwidth usage for each algorithm. This value will be utilized to calculate the overall simulated transmission delay. The total handshake size is the input of our Simulink model. With our simulated delay, we will finally calculate the total estimated handshake time using the recorded mean CPU times and the simulated transmission delay as presented in equation 5.3 below.

$$T_{handshake} = T_{card} + T_{PC} + T_{mathlab} \tag{5.3}$$

In other words, the total completion time for the handshake will be the total mean CPU time on the card added to the total mean execution time on the PC added to the simulated transmission delay calculated using our Simulink models. The total handshake time will be one of the main factors that will indicate the impact of using PQ algorithms instead of ECC.

5.4.1 Classical algorithms handshake

The first run of our DTLS experiments will be a run using classical algorithms only. The signing algorithm will be ECDSA and the key exchange algorithm will be ECDHE. For the signing algorithm, we will only use the P384 curve. This is because doing all the possible combination for Key exchange algorithms with the signing algorithms would require running our experiments a large number of times. Because of this, we will try to keep the certificate part of our experiment as stable as possible. This decision was made to reflect the fact that we consider the testing of the PQ KEM algorithms to holds greater significance. As a result, we allocated a larger portion of our efforts towards testing the various KEM candidates. These choices are reflected here to hold consistency across our different testing scenarios. In

the classical run of our experiments, we will execute our handshake 3 times, each with a different curve for key exchange: P256, P384 and P521.
Starting the experiment, we will generate a root certificate using the openssl fork. We will then generate the server certificate that we will sign it using the root certificate, as well as use it as the server certificate for the s_server application using the -cert parameter. On the client side, we will select the curve to use for key exchange using the -Curves option. The specific curve implementations that were used were the secp<curvesize>r1 curves in openssl and secp<curvesize>k1 for P256.

5.4.2 KEM algorithms handshakes

For the KEM algorithms, we will use the same certificate generated for the classical run. Like mentioned in the previous section, testing all the KEM algorithms with all the signing algorithms would require too many experiment runs. In this experiment, we are executing the DTLS handshake for all the main variants of all the KEM candidates. To select the right KEM algorithm we enfore it in the handshake by using the -groups option on the client application.

5.4.3 Fully post-quantum

Since the NIST made their first selection for the 4 algorithms that are going to be standardized, we decided to have a third round of the experiment where we will be using both post-quantum KEM algorithms and post-quantum signing algorithms. We decided to prioritize the algorithms that we know are going to be standardized as of this day. The current list of selected candidates contains 3 signing algorithm and one KEM algorithms. Contrarily to the other 2 experiments, here the key exchange algorithm will stay as a constant while we will generate a root certificate and a server certificate for all the selected PQ signing algorithms. The selected KEM is KYBER and the selected algorithms for digital signatures are DILITHIUM, FALCON and SPHINCS+ [21].

5.4.4 Hybrid algorithms handshakes

While hybrid algorithms are considered to be a category on their own, we will consider them as part of the other KEM and signing algorithms. This means that for the KEM section of our experiments, we will also test all the hybrid KEM variants. For the fully post-quantum section of our experiment, we will also test all the hybrid variants of the select candidates for digital signing. However, we will not use the KYBER hybrid variant for this part of the

experiment

5.4.5 Calculation of the transmission delay

The transmission delay will be calculated by simulating an air-ground communication with our Simulink models. Both VHF and HF communication will be simulated.
The input of our simulation models will be the total size of the communication for a DTLS handshake. To retrieve this value, we will capture the packets received and sent between the PC and the card by using Tcpdump. We will use the total size of the packet capture as the input of our models. We assume that the content of the packets will not influence the simulation models. We also assume that the total transmission delay is equal to the sum of transmission delay for each packet. Because of this assumption, we will be running the simulation only once with the total size of the communication to simulate the total transmission delay of the handshake.
The output of our simulation model is the number of bytes that the **Find Delay** block has had to read before the input and the output signal matched. My multiplying this value with the bitrate corresponding to each type of air-ground communication, we can estimate their total transmission delay. As a reminder, for the way we configured our simulated HF and VHF models, the bitrate for VHF communication is **31500 bits per seconds** while the bitrate for HF communication is **1600 bits per seconds**.

5.5 Other benchmarking methodologies

Others have already tested impacts in similar ways. If we look at [74], the authors used similar scenarios as previous described with one simple difference. While they do have a similar KEM scenario where they measures handshake completion times of KEM algorithms with classical signatures, they also tested the impact of PQ signatures with a classical key exchange algorithm. This is a straightforward way to extract the impact of using PQ signatures instead of classical signatures. However, since we stated that having PQ digital signatures without PQ KEMs might not be worth the efforts here, we used a PQ KEM algorithm for our equivalent of their "signature scenario" instead. The impact over classical algorithm can still be extracted by comparing our fully post-quantum scenario to our KEM scenario. Another difference is that the authors used a network emulator and extracted the median and 95th percentiles of their results. The network emulator is a great way to simulate network conditions but is not applicable for our air-ground scenario which is why we went the Simulink route. Finally, instead of the 95th percentiles and median, we decided to extract the means. Apart from these specific details, our testing scenarios and validation processes

are similar [22].
In the publication "*Post-Quantum Authentication in TLS 1.3: A Performance Study*", the authors present their findings in a manner similar to our approach. Their Table II is presented in the same format as our general performance evaluation because they used the same benchmark tools as us. Also, their *Table IV* where they showed their handshake results using mean completion times is similar to what we will present in the next chapter for DTLS completion times. Finally, they collected their mean completion time from running their handshake 1000 times exactly like we will be doing.

5.6 Discussion

In this chapter, we initially demonstrated the validation process for the implementation outlined in the previous chapter. Subsequently, we introduced the selected algorithms and described the methodology for testing our DTLS handshakes across three different configurations. Furthermore, we detailed how the outcomes of these experiments would be utilized to simulate a transmission delay for air-ground communication. In the following section, we will present the results of these experiments and provide answers to our research questions. Additionally, we will propose several potential solutions that could be explored to minimize the overall handshake delay under specific circumstances.

CHAPTER 6 RESULTS AND ANALYSIS

In this chapter we will demonstrate the results from the experiments described in the previous section. We will first start by showing the general performances of our algorithms on both the card and the PC. Then, we will show the results of the DTLS handshake as well as an estimation of the handshake completion time for both HF and VHF communication.

6.1 General performances on the P2020

First, we will present the results from the benchmark of the KEM and signature algorithms on the P2020. These results will serve as indicators of how these algorithms are going to perform on future avionic hardware.

6.1.1 KEM algorithms

Here we are presenting the performances of 4 different algorithms: Frodo, HQ, KYBER and McEliece. We tested all the different security parameters that were available to us. See Annexe A for reference. Table 6.1 shows the performance for key generation and table 6.2 shows the performance for key encapsulation and decapsulation.

Algorithm	**Key Generation**		
	Iteration	**Average Time (us)**	**pop. stdev**
Frodo640aes	37	82 947,57	24,59
Frodo976aes	16	189 656,38	40,43
Frodo1344aes	9	352 477,33	42,22
hqc128	481	6 244,25	124,48
hqc192	274	10 957,56	233,14
hqc256	177	16 957,15	270,16
Kyber512	945	3 176,65	14,10
Kyber768	773	3 383,19	37,94
Kyber1024	606	4 957,56	17,64
Classic-McEliece-348864	2	1 603 548,00	183 838,00
Classic-McEliece-460896	1	6 059 713,00	0,00
Classic-McEliece-6688128	1	13 502 757,00	0,00
Classic-McEliece-6960119	1	9 872 566,00	0,00
Classic-McEliece-8192128	1	12 742 318,00	0,00

Table 6.1 Key generation benchmark of KEM algorithms

Algorithm	Encapsulation			Decapsulation		
	Iteration	Average Time (us)	pop. stdev	Iteration	Average Time (us)	pop. stdev
Frodo640aes	35	85 991,06	17,02	37	83 126,30	29,14
Frodo976aes	16	195 530,94	33,30	16	190 604,81	39,93
Frodo1344aes	9	358 975,22	47,17	9	352 644,78	67,91
hqc128	386	7 791,60	201,87	309	979,26	18,23
hqc192	191	15 735,40	329,98	155	19 399,71	29,82
hqc256	115	26 172,92	270,16	91	33 224,40	36,27
Kyber512	1208	2 483,70	12,04	2280	1 316,33	17,12
Kyber768	889	3 378,23	9,98	1402	2 141,29	19,17
Kyber1024	656	4 577,33	12,06	918	3 269,18	20,78
Classic-McEliece-348864	704	4 262,37	1 181,99	1064	2 821,57	10,39
Classic-McEliece-460896	363	8 276,86	2 376,86	401	7 491,96	12,48
Classic-McEliece-6688128	227	13 254,99	2 958,59	356	8 434,00	26,22
Classic-McEliece-6960119	246	12 216,86	1 687,98	366	8 198,37	10,91
Classic-McEliece-8192128	198	15 209,50	2 008,65	354	8 492,62	13,36

Table 6.2 Encapsulation and Decapsulation benchmark of KEM algorithms

As expected, the main outliers here are the McEliece variants. Most of their key generation operations could only be run once since they took well over 3 seconds to execute. It is also the only algorithm that has such a significant performance difference among its various types of operations. However, there is something that is worth noting considering the architecture of this experiment where the DTLS server can be more powerful than the client. In this scenario, the only KEM operation the client would have to do is encapsulation and the table shows that McEliece performed the second best for that operation. Nevertheless, we anticipate that the transmission delays will be the most limiting factor and considering McEliece key sizes, this is where the algorithm will face challenge in performing.
We also predicted that KYBER will be the algorithm giving us the best overall best results and we were not wrong. As the only KEM algorithm being selected for standardization by NIST, these performance measurements only strengthen the choice behind this selection.
Apart from that, HQC also demonstrated excellent overall performance. However its key generation operation results were more inconsistent than all the other algorithms, not counting McEliece. Finally, while Frodo offered some of the worst performances overall, its performances were stable across all the 3 different types of operations. This means asymmetrical architecture from our experiments won't advantage nor disadvantage this algorithm.

6.1.2 Signing Algorithms

After the previous overview of the performance of our KEM algorithms, the next tables will present the performance of the 3 signing algorithms that were selected for standardization by NIST. It is important to note that although we are presenting the performance for key generation, in a DTLS handshake, considering that PQ algorithms cannot simultaneously execute digital signing and key exchange operations, we assume that the signing keys and

certificate will be generated in advance. This means that the performance shown for the key generation of PQ signing algorithms won't impact our DTLS experiments, only the signing and verifying operations will. Table 6.3 shows the performance for key generation and table 6.4 for signature and signature verification.

Algorithm	Key Generation		
	Iteration	Time (us)	pop. stdev
Diltihium2	596	5 040,79	50,31
Diltihium3	368	8 168,86	10,58
Dilithium5	227	13 244,65	86,95
Falcon512	17	180 874,65	64 690,96
Falcon1024	5	668 715,00	187 739,91
SPHINCS+-SHA256-128f-robust	71	42 620,72	30,42
SPHINCS+-SHA256-192f-robust	48	62 884,85	37,95
SPHINCS+-SHA256-256f-robust	14	225 753,43	55,17

Table 6.3 Key generation benchmark of signing algorithms

Algorithm	Sign			Verify		
	Iteration	Average Time (us)	pop. stdev	Iteration	Average Time (us)	pop. stdev
Diltihium2	265	11 343,82	5 544,50	758	3 958,94	17,53
Diltihium3	171	17 553,57	9 840,41	442	6 802,55	19,82
Dilithium5	136	22 149,35	8 203,72	252	11 916,08	33,46
Falcon512	80	37 629,44	66,41	3762	797,57	5,50
Falcon1024	38	79 600,05	94,87	1940	1 546,85	6,75
SPHINCS+-SHA256-128f-robust	3	1 019 872,67	444,89	51	59 941,18	30,42
SPHINCS+-SHA256-192f-robust	2	1 712 849,00	2 097,00	34	89 971,38	62,32
SPHINCS+-SHA256-256f-robust	1	4 702 093,00	0,00	23	133 707,26	43,62

Table 6.4 Signing and signature verification benchmark of KEM algorithms

After the great results from KYBER, it is no surprise that its signing counterpart Dilithium is also the one offering the best overall results here. Contrarily to the KEM algorithms, we don't have any **major** outliers in these last two tables. However, SPHINCS+ is by far the worst algorithm both in terms of signing and verifying digital signatures. Its highest security variant could not sign under 3 seconds and while its verification completion times are acceptable overall, it is still largely underperforming Dilithium and Falcon across all variants.

One advantage of solely performing server authentication is that the client will never need to perform signing operations; its role is limited to verifying the server certificate. Since our client is our least powerful device in our DTLs experiments, the fact that falcon has an fast verification time is a great asset for this scenario.

6.1.3 Unsupported algorithms

One other algorithm that we planned on using for our experiments is Bike. However, the bike implementation from liboqs does not support 32-bit systems so it could not be executed

on our card.

6.2 DTLS results

In this section, we will be presenting the results for all of our DTLS experiments. For both our KEM-only experiments and our fully PQ experiments, we will display the results using three tables. The first tables (table 6.5, 6.8 and 6.11) will show an overview of the bandwidth requirement for each algorithm by presenting the complete size of all the information sent by both the client and the server. While the individual sizes can give a general idea of the memory usage necessary for each device, the total size is used to simulate the transmission delay of air-ground communication.
The second tables (table 6.6, 6.9 and 6.12) will show the simulated data for air-ground communication. The first column will show the delay in terms of bytes that the transformation for air-ground communication adds to the system, while the 2 other columns show the simulated transmission delay in seconds for both VHF and HF communications. The values in these two columns represent the results of the first column multiplied by the corresponding data rates for VHF and HF communications, respectively. In this table, we simulated the delays using both of our models. However, due to the fact that the data rate of 8-PSK and 8-DPSK is identical, the simulated delays in terms of bytes were also the same. Therefore, we have only included a single column for both models. If we were to introduce errors in our communication models, then we could expect that both models would have produced different results and 2 columns would have been necessary.
For the third tables (table 6.7, 6.10 and 6.13), we will be showing the total results of the emulated air-ground handshake by using the method described in the previous section. We first show the total CPU times on both the card and the PC during the handshake. We then add to these values their simulated transmission delays to generate a total emulated completion time of each handshake.

6.2.1 Unsupported algorithms and failures

Like mentioned earlier, the bike algorithm was not supported on our card so it was not used for the DTLS experiments either. Furthermore, like explained in [13], some algorithms and variants does not meet the specification requirement of (D)TLS and some are limited by implementation specifics of openssl. In the case of Classical-McEliece, its public key sizes are larger than the TLS limit which is why we excluded it from our DTLS experiments. Also like it is mentioned in this article, a lot of the PQ key and signature sizes that do respect the (D)TLS specification were still initially blocked by the openssl implementation which

has a lower limit than what the specification allowed. However, this could simply by fix by increasing this limit in our implementation.
However, despite implementing these workarounds, we encountered issues with two algorithms. The first one is with the Frodo1344aes variant which has the biggest public key and ciphertext sizes of all the KEM algorithms that were tested for these experiments. During the phase when our card attempted to transmit the encrypted shared secret to the server, we encountered the error "write:errno=90." This error indicated that we were attempting to send a message that exceeded the maximum length supported by our network interface.
The other algorithm is SPHINCS+. However, contrary to the Frodo algorithm, none of its variants could be tested. Instead of returning the same error as the Frodo variant, however, the handshake executed at least twice, sometimes more, before establishing a secret key. The handshake could never be completed the first try.
For both algorithms, we verified with Tcpdump and Wireshark that the handshakes were failing. This validation step proved to be useful since handshake sometimes completed for Frodo's variant. However, with Wireshark we could clearly see that there were too many packets being sent and that it was caused by the the handshake restarting a couple of times. During the investigation, we tested both algorithm locally by executing the handshake on a standard Linux PC and they both succeeded the handshake normally. As a result, we classify both of these bugs as limitations of our setup that were caused by our card. Considering that both these algorithms did not work due to their high bandwidth usage, however, we can conclude that they are not the best choices and variants for a bandwidth constrained environment like avionic communication to begin with.

6.2.2 Classical Handshakes

Table 6.5 shows the results of our classical experiment using ECC as the key exchange algorithm. These results should give an idea about DTLS performances for air-ground communication as it is currently being drafted for ATN/IPS. We will then use these results as a baseline of comparison for the PQ algorithms.

Algorithm	**Handshake Size Server (bytes)**	**Handshake size Client (bytes)**	**Handshake Size total (bytes)**
secp256k1	1106	748	1854
sec384r1	1137	780	1917
secp521r1	1174	816	1990

Table 6.5 Communication sizes for DTLS1.2 handshakes with ECC and ECDSA

Algorithm	Handshake simulated delay (bytes)	Simulated VHF Delay (s)	Simulated HF Delay (s)
secp256k1	14830	3,77	74,15
sec384r1	15101	3,84	75,51
secp521r1	15936	4,05	79,68

Table 6.6 Simulation of the transmission delays for the classical handshakes

Algorithm	Mean CPU time card (ms)	Mean CPU time PC (ms)	Total VHF time (s)	Total HF time (s)
secp256k1	32,55	1,77	3,80	74,18
sec384r1	45,64	8,28	3,89	75,56
secp521r1	94,48	17,29	4,16	79,79

Table 6.7 CPU times and total classical emulated handshake times for HF/VHF

These will be the same 3 tables we will use to show the results for our other 2 scenarios. As a reminder, the total columns are calculated by summing both the CPU times and the respective simulated delay.

Here there is not that much to declare as these algorithms have been tested and used for a very long time. However, it is interesting to see in table 6.6 and tale 6.7 that even with ECC, simulated times for air-ground communication are still relatively high which makes sense considering the low bandwidth of the environment. With these results, it is safe to assume that all the post-quantum algorithms would take over one minute to execute a DTLS handshake over High Frequencies.

6.2.3 Post-quantum KEM handshakes

In this section we will present our results for our handshake scenario where we used a post-quantum KEM algorithm with a classical certificate and signing algorithm (ECDSA-p384). The results will show the impact of switching only the key exchange mechanism from the previous experiment to use PQ algorithms.

Algorithm	Server Handshake Size (bytes)	Client Handshake Size (bytes)	Total handshake size (bytes)
Frodo640aes	11 830	10 554	22 384
Frodo976aes	18 572	16 678	35 250
Frodo1344aes	X	X	X
hqc128	3 563	5 240	8 803
hqc192	6 112	9 860	15 972
hqc256	9 161	15 403	24 564
Kyber512	1 915	1 452	3 367
Kyber768	2 348	1 772	4 120
Kyber1024	2 782	2 277	5 059
P256_Frodo640aes	11 897	10 619	22 516
P384_Frodo976aes	18 694	17 075	35 769
P521_Frodo1344aes	X	X	X
P256_hqc128	3 630	5 305	8 935
P384_hqc192	6 210	9 957	16 167
P521_hqc256	9 319	15 536	24 855
P256_Kyber512	2 006	1 517	3 523
P384_Kyber768	2 472	1 869	4 341
P521_Kyber1024	2 940	2 410	5 350

Table 6.8 Communication sizes for PQ KEM handshakes with ECDSA certificates

As expected and shown in table 6.8, Frodo is the algorithm that requires the highest bandwidth out of all 3 algorithms tested here. On the other hand, much like the performance benchmark that was done on the card, KYBER is the algorithm that also performed the best in terms of communications sizes as well. HQC also has respectable sizes but they increase rapidly as we used variants of higher security levels. We can also see that the hybrid variants do not add too much to the communication size in comparison to what the PQ algorithms already require.

Generally, in this table, the total size sent from the server is impacted by the increase in size of the public key while the change in total size sent by the client is impacted by the ciphertext size of the algorithm. Contrarily to KYBER, HQC has ciphertexts larger than its public key which results in higher bandwidth usage on the client than on the server. While its bandwidth usage his still way lower than Frodo's, this characteristic is not optimal considering our model.

Algorithm	Handshake simulated delay (bytes)	Simulated VHF Delay (s)	Simulated HF Delay (s)
Frodo640aes	179 123	45,49	895,62
Frodo976aes	282 232	71,68	1 411,16
Frodo1344aes	X	X	X
hqc128	68 405	17,37	342,03
hqc192	127 300	32,33	636,50
hqc256	193 240	49,08	966,20
Kyber512	25 320	6,43	126,60
Kyber768	32 450	8,24	162,25
Kyber1024	40 488	10,28	202,44
P256_Frodo640aes	180 144	45,75	900,72
P384_Frodo976aes	286 152	72,67	1 430,76
P521_Frodo1344aes	X	X	X
P256_hqc128	70 906	18,01	354,53
P384_hqc192	129 336	32,85	646,68
P521_hqc256	198 840	50,50	994,20
P256_Kyber512	28 200	7,16	141,00
P384_Kyber768	34 728	8,82	173,64
P521_Kyber1024	42 488	10,79	212,44

Table 6.9 Simulation of the transmission delays for KEM handshakes

In table 6.9, the first column shows the resulting delay simulated by our Simulink model. This delay is then multiplied by the HF and VHF data rate respectively to simulate their total transmission delays in seconds. In VHF, we can see that most algorithms would be able to execute the handshake under 1 minute. For Frodo, we can see that while VHF as a higher bandwidth than HF, its transmission time in VHF would be close to the transmission delay with ECC but on HF. As usual, KYBER is the algorithm with the best results with most variant executing in 10 seconds or less. By transitioning from the most secure ECC variant to the most secure KYBER variant, there is an approximate increase of six seconds in the VHF communication. For the least secure variants, however, there is only a difference of 3,16 seconds.

For HF, however, the impact is much higher. If we look at Frodo, the worst performer, its middle variant would take around 23 and a half minutes to complete the handshake. The significant duration needed to establish a secure communication channel is notably long. As a reminder, these simulated values are also under perfect transmission conditions. However, compared to sec384r1, this is over 22 minutes slower than the classical implementation. For the KYBER variants, while their impact is not as big as Frodo's, their overall transmission delay would still be approximately at least double or more the delays from ECC. Overall, if KYBER were to replace ECC for HF handshake, the transmission delay would increase from approximately one minute to two or three minutes.

Furthermore, it is evident that transitioning from PQ to hybrid algorithms does not have a significant impact on VHF communication, with most P521 hybrids resulting in less than 1 second of additional delay. However, in HF communication, the increased communication size contributes an additional delay of approximately 10 to 20 seconds to an already lengthy process.

Finally, HQC continues to deliver average results by being worst than KYBER but definitely better than Frodo in terms of performance. It is also apparent that each of its variant differs a lot in performance with each having 5 minutes of difference with each other for the HF results.

Algorithm	Mean CPU time card (ms)	Mean CPU time PC (ms)	Total VHF time (s)	Total HF time (s)
Frodo640aes	120,76	17,99	45,63	895,75
Frodo976aes	244,51	26,47	71,95	1 411,43
Frodo1344aes	X	X	X	X
hqc128	31,21	11,00	17,41	342,07
hqc192	40,17	13,84	32,38	636,55
hqc256	51,49	14,11	49,14	966,27
Kyber512	25,08	6,67	6,46	126,63
Kyber768	25,83	7,02	8,27	162,28
Kyber1024	27,03	7,53	10,32	202,47
P256_Frodo640aes	143,46	18,03	45,91	900,88
P384_Frodo976aes	290,72	20,67	72,98	1 431,07
P521_Frodo1344aes	X	X	X	X
P256_hqc128	43,23	7,69	18,06	354,58
P384_hqc192	77,76	11,95	32,94	646,77
P521_hqc256	134,87	32,97	50,67	994,37
P256_Kyber512	35,45	6,09	7,20	141,04
P384_Kyber768	51,19	8,48	8,88	173,70
P521_Kyber1024	101,95	19,00	10,91	212,56

Table 6.10 CPU times and total emulated handshake times for HF/VHF (KEM)

Table 6.10 shows the total CPU times during the DTLS handshakes for both the client (the card) and the server (the PC). By combining both values with the simulated delays from the previous table, we obtain the last two columns of this table, which represent the total handshake completion times for VHF and HF communication.

Table 6.10 also confirms our hypothesis that the most important factor for the ATN/IPS scenario is the total size of the communication generated by the handshake. Indeed, even considering the slow performance of the card, especially with Frodo, the additional delays caused by CPU execution are all under half a second. The impact of these CPU times on the total completion time is minimal, to the extent that all the conclusions drawn from the previous table remain applicable here.

One interesting value in this table is that KYBER had better results in terms of delays caused by CPU executions than ECC, with the exception of Kyber512 and only on the PC. We can see that comparatively to ECC, The CPU requirement from Kyber is more stable across its variants which results in its highest secure variant having a significant advantage on P521. In addition, we can conclude that while the hybrid variants have a minimal impact on the overall completion time, they do significantly increase the required CPU times. Specifically, the inclusion of P521 demands more than double the required CPU time on the card.

Impacts over classical algorithms and best choice for ATN/IPS

Upon analyzing these results, we can conclude that the most suitable choice for a KEM algorithm in ATN/IPS would be KYBER. Even without considering its great performance on hardware, it is its lower bandwidth requirement which makes him the best choice for our scenario. As observed, even when considering a low-powered client and the potentially higher computational requirements of certain algorithms, the most significant factor influencing air-ground communication is the size of the transmitted data. These conclusions also aligns with the reality that that upgrading hardware on aircrafts is more feasible than changing the available bandwidth on HF and VHF channels.

While simulating is not as precise as actually executing the handshakes over HF and VHF, we can estimate that switching from P384-ECC to Kyber768 would result in an increase of latency **between 112% and 114%** for both HF and VHF communications. Also, switching from Kyber768 to its hybrid variant would result in an increase of latency of approximately **7%**.

6.2.4 Fully post-quantum handshakes

In this section, we will analyze the results from the last part of our experiment the same way we did in the previous section. As a reminder, for our final experiments, we ran our DTLS handshake with Kyber768 as the Key exchange and executed with different certificates using different PQ signing algorithms.We did not conduct experiments involving PQ signing algorithms combined with classical key exchange algorithms as we believe that scenarios involving only the switch of certificates in a DTLS handshake to use PQ algorithms are unlikely to occur in practice. We need to prioritize the switch to PQ KEM algorithms as they will help provide confidentiality against future attacks on current information. However, the switch to PQ certificates can be done when the quantum threat will become active.
Even with that in mind, evaluating the impact of switching to post-quantum certificates and signatures especially in low-bandwidth environments like avionic communication remains an important task. For this section, the results that were gathered from Kyber768 in the previous section will be used as the point of comparison to evaluate the impacts of switching to post-quantum signature algorithms.

Algorithm	Handshake Size Server (bytes)	Handshake size Client (bytes)	Handshake Size total (bytes)
Diltihium2	8 934	1 772	10 706
Diltihium3	11 620	1 772	13 392
Dilithium5	15 264	1 772	17 036
Falcon512	4 475	1 772	6 247
Falcon1024	6 876	1 772	8 648
SPHINCS+	X	X	X
p256_Diltihium2	9 168	1 772	10 940
P384_Diltihium3	11 952	1 772	13 724
P521_Dilithium5	15 727	1 772	17 499
P256_Falcon512	4 752	1 772	6 524
P521_Falcon1024	7 344	1 772	9 116
P256_SPHINCS+	X	X	X

Table 6.11 Communication sizes for handshakes with Kyber768 and PQ certificates

Unfortunately, due to our inability to successfully execute SPHINCS+ handshakes on our card, our comparison is limited to only two algorithms: Dilithium and Falcon. However, taking into account the results from the previous experiments, the fact that SPHINCS+ certificate and signature sizes are the largest amongst the three would most likely make it the worst contender here.
In table 6.11, while KYBER emerged as the best algorithm in our KEM experiments, when it comes to communication sizes, we can observe that its signing counterpart, Dilithium, is not the most bandwidth-efficient option in this last table. While, in terms of security level, falcon does not have an equivalent to compare to Dilithium3, even its highest secure variant Falcon1024 has a lower communication size than the lowest secure Dilithium.
We can also see that much like the previous experience, the hybrid variants do not add much

communication overhead with the exception of P521 curves which adds a bit under 500 of extra bytes. Furthermore, the communication sizes in this last table are way closer values compared to the different KEM algorithms. With results much closer, CPU times might have a bigger impact than it had for KEM algorithms.
Lastly, since we have only enabled server authentication, the selection of a certificate does not affect the transmission size sent by the client.

Algorithm	handshake simulated delay (bytes)	VHF Simulated Delay (s)	HF Simulated Delay (s)
Diltihium2	84 700	21,51	423,50
Diltihium3	102 702	26,08	513,51
Dilithium5	130 002	33,02	650,01
Falcon512	50 200	12,75	251,00
Falcon1024	68 926	17,51	344,63
SPHINCS+	X	X	X
p256_Diltihium2	85 656	21,75	428,28
P384_Diltihium3	109 800	27,89	549,00
P521_Dilithium5	139 992	35,55	699,96
P256_Falcon512	52 200	13,26	261,00
P521_Falcon1024	72 936	18,52	364,68
P256_SPHINCS+	X	X	X

Table 6.12 Simulation of the transmission delays for fully post-quantum handshakes

Table 6.12 reveals an intriguing finding: without considering delays caused by cryptographic operations, even with a post-quantum KEM algorithm, each handshake using post-quantum certificates would run way significantly under one minute on VHF. This means that the extra delay caused by choosing either signing algorithm is less less impactful compared to the overhead caused by choosing other KEM algorithms.
Comparing to our previous results with Kyber768, we can observe that the switch from ECDSA-P384 to Dilithium 3 would result in an additional delay of **17.84 seconds** in VHF communication, while switching to Falcon1024 would result in an additional delay of **9.27 seconds**. In HF communication, both Dilithium 3 and Falcon1024 would result in significant increases in delay compared to ECDSA-P384. Specifically, Dilithium 3 would introduce an additional delay of **351.26 seconds** (almost 6 minutes), while Falcon1024 would introduce an additional delay of **182.38 seconds** (almost 3 minutes).

Algorithm	Mean CPU time card (ms)	Mean CPU Time PC (ms)	Total VHF time (s)	Total HF time (s)
Diltihium2	14,71	3,93	21,53	423,52
Diltihium3	20,65	4,55	26,11	513,54
Dilithium5	31,02	4,79	33,05	650,05
Falcon512	7,87	4,41	12,76	251,01
Falcon1024	9,79	5,75	17,52	344,65
SPHINCS+	X	X	X	X
p256_Diltihium2	22,89	3,97	21,78	428,31
P384_Diltihium3	40,80	8,47	27,93	549,05
P521_Dilithium5	88,98	9,29	35,65	700,06
P256_Falcon512	16,31	4,58	13,28	261,02
P521_Falcon1024	67,17	8,57	18,60	364,76
P256_SPHINCS+	X	X	X	X

Table 6.13 CPU times and total emulated handshakes times for HF/VHF (SIG)

Finally, we can observe in table 6.13 that Falcon outperforms Dilithium overall in our emulated DTLS handshake for air-ground communication. This makes sense because 1) as we saw in our KEM experiment, the most impactful factor in our scenario is the total communication size for our handshake and Falcon has smaller key and signature sizes than Dilithium. 2) Looking back at the benchmark, the biggest downside of Falcon is its key generation time which is more than 100 times slower than Dilithium. However, in (D)TLS handshakes, the signing keys are pre-generated, meaning that this disadvantage does not impact our scenario. Furthermore, as mentioned previously, Falcon has faster verification times compared to signing times, which has a positive impact on our experiment since we only perform server authentication. However, even if client authentication was done as well, tts superior bandwidth efficiency compared to Dilithium would likely give it the advantage in terms of performance.
Similar to the KEM experiments, the hybrid algorithms does not show a significant difference in terms of total delay compared to their base PQ algorithm.However, the hybrid algorithms still have a considerable impact on the required CPU times for the handshake, particularly on the card and especially in the case of P521. Apart from that, most conclusions drawn from the previous table are also applicable here.

Impacts over classical algorithms and best choice for ATN/IPS

Overall, we can conclude that the best choice for a post-quantum signing algorithm in ATN/IPS is Falcon. Despite having slow key generation, this does not impact DTLS handshakes. Despite having slightly longer signing times than Dilithium, Falcon does have better verification times than its competitor. However, the most significant advantage of Falcon is its lower bandwidth requirement. As we have emphasised before, this is the most impactful factor for this scenario.
In summary, the transition from P521-ECDSA to Falcon 1024 would increase the handshake

completion time by approximately **112%** for both HF and VHF communications. However, switching from Falcon1024 to its hybrid variant would result in a smaller increase of less than **6%**, similarly to the KEM algorithms.

6.3 Comparison with relevant research

In [5], the authors reported that the time to establish a secure connection using ECDHE-ECC was of 4.471 seconds. Since they used Secp384r1, we can compare their result to our time of 3.89 seconds in our Table 6.7. The difference in time can be explained by the fact that the authors implemented two-way mutual authentication instead of one way like use. Although our testing environment was not as close to reality as theirs, it is encouraging to observe that our results are still comparable to theirs, even though we simulated the transmission delays.

6.4 Discussion

In this chapter, we presented our results in 2 parts. In the initial section, we presented a benchmark of several post-quantum algorithms running on our future avionic hardware. The results showed that generally, KYBER and Dilithium were the algorithms delivering the best results. We could also conclude that while Dilithium had better results than Falcon overall, its lower verification time is an advantage in our scenario without client authentication.
In the second part, we presented the results of our handshakes in three different scenarios: one utilizing ECC for both key exchange and certificate, one where we switched the key exchange mechanism to post-quantum algorithms, and one where we modified the certificate to incorporate various post-quantum algorithms while maintaining Kyber768 as the key exchange mechanism.
Overall, these results showed that the increased computational requirements of certain PQ algorithms have a much smaller impact compared to the effect of their higher bandwidth usage. We were also able to conclude that KYBER would be the best choice for a KEM algorithm in ATN/IPS while Falcon would be the best choice for authentication.
When considering the impacts of using post-quantum algorithms instead of ECC, we observe that the handshake time would at least double for both VHF and HF communications. Additionally, the use of hybrid algorithms, compared to the base PQ algorithms, would result in an increase in handshake time of approximately 6-7%.

6.4.1 Reducing the handshake time

After recognizing the significant impact of communication size on our scenario, it becomes crucial to explore potential measures to mitigate this impact in the future. Therefore, we present several propositions aimed at reducing the communication size and therefore improve the handshake completion times.

1. Use DTLS1.3 instead of DTLS1.2
2. Only add use PQ for key exchange
3. Doing one way authentication
4. Consider doing handshakes over VHF as much as possible.
5. For falcon specifically, use a curve of lower security for its hybrid variant.
6. Non-ephemeral keys.
7. Pre-shared keys.

In regard to the first proposition, DTLS1.3 is designed to be more efficient by eliminating a round trip that was previously required during the handshake process. When DTLS1.3 will completely be finalized and available, using is over DTLS1.2 should provide mostly benefits to ATN/IPS, unless non-ephemeral keys are used [15, 25].
For point #2, switching our key exchange algorithms to use PQ algorithm is more urgent than switching our certificate. Indeed, handshake done today could be recorded and kept to be decrypted when a fully working quantum computer is available. However, the day such computer will be available, there will be no impact on the authenticity and integrity provided by past certificates. By switching only the key exchange part of the handshake like we have done in the second part of our DTLS experiment, we can reduce the impact on communication sizes and maintain a more efficient transmission.
Regarding the third proposition, it is essential to highlight that implementing such changes may have security implications that extend beyond the scope of this research. However, on a performance point of view, if it is viable to authenticate only one side of the handshake similarly to what we have done, this would also reduce the communication size comparatively to a two-way authentication mechanism.
For the fourth point, as anticipated, switching to PQ algorithms has a more significant impact on HF channels compared to VHF. HF is usually necessary for overseas flights. If possible, a logic that would ensure that the handshake is done before going overseas could reduce the

overall impact of using PQ algorithms. The handshakes would require a few extra seconds which is less noticeable than a few minutes on HF. For the fifth proposition, the comparison analysis of Falcon was done using its 1024 variant because it does not currently have a variant of security level three. However, if the scenario does not specifically require the extra security levels provided by Falcon1024, and if the implementation includes hybrid algorithms, it would be beneficial to lower the classical security level by introducing a Falcon1024 variant that utilizes the P384 curve. This would reduce the handshake time by reducing the handshake size and the computational power needed.

Considering the sixth proposition, the idea would require a standardize way to prove ownership of a key for a post-quantum KEM algorithm. This would allow saving the computing time necessary for generating the post-quantum key during the handshake. However, we would lose the forward secrecy of the protocol. Considering the conclusion that the CPU time required for the handshake has a less significant impact on the total handshake time than the transmission delay, the trade-off might not be worth the consideration [29, 70].

And finally, using pre-shared keys in ATN/IPS would obviously be the most efficient scenario as there would be no need to use public-key cryptography. However, pre-shared keys are known for their lack of scalability. Indeed, each time a new peer needs to be communicated with, all the communication nodes must be updated with the new key. This is because there are significantly more new aircraft being introduced than new ground stations.

CHAPTER 7 CONCLUSION

In this research, we conducted a performance evaluation of multiple post-quantum algorithms within a testing environment that replicated the ATN/IPS architecture. In this chapter, we begin by providing a summary of our experimental results that are directly relevant to our objectives. The questions that guided our research objectives were:

Q1: What are the impacts of using post-quantum algorithms in avionic communication, more specifically in ATN/IPS?

For this question, we established that switching to post-quantum KEM algorithms would result in at least doubling the handshake completion time in ATN/IPS. The same conclusion was also made for signing algorithms. We also established that, while there was a slight computing performance impact, the overall slower CPU performance for PQ algorithms was much less important than the impact on bandwidth usage. This last impact is the main the reason for the slower handshake completion time for PQ algorithms in avionic communication.

Q2: Considering the impacts of switching to post-quantum algorithms, which algorithm(s) should we recommend for consideration to secure avionic communication?

For this question, we established that considering the impacts on the bandwidth usage and the total handshake completion time, Kyber would be the best choice for a PQ KEM candidate in ATN/IPS while Falcon would be the best choice for the signing algorithm candidate.

Following the summary of our work, we then highlight the limitation of our experiment and propose potential enhancements to our methodology for future investigations.

7.1 Summary of Work

In this thesis, we presented a way to simulate a DTLS1.2 handshake executed within ATN/IPS, a new architecture for avionic communication that is currently being developed. We developed our experiment in the goal of having a platform to evaluate the impacts of adding post-quantum security in avionic communication systems. These impacts were evaluated by considering the additional CPU time, bandwidth usage and completion time for a DTLS 1.2 handshake. To make this possible, we modified the liboqs fork of openssl to support post-

quantum algorithms in its DTLS1.2 implementation. To replicate the avionic aspects of our research, we used a card that has performances similar to future avionic hardware. We then used a standard PC as the handshake server to replicate the ground-based control system. To evaluate the impacts on the transmission delays, we created a Simulink model for HF and VHF communication which we used to simulate the air-ground transmission part of our handshakes.

For our experiments, we had the goal to experiment with all the algorithms select by NIST for standardization, as well as all the candidates still being considered for the fourth round of standardization. However, we were not able to run our experiments with all of these algorithms due to limitations from our experimental setup.

To evaluate the impact on CPU times, our initial step involved running a benchmark tool on our card to measure the performance of the three operations associated with each algorithm. The results showed that KYBER was the most performant KEM algorithm overall. HQC had great performance as well but was slightly slower than KYBER overall. We also saw that McEliece's key generation would not be sustainable on avionic hardware with results ranging from approximately 1.6 to 13.5 seconds. For signing algorithms, we saw that Dilithium was the most performant algorithm on avionic hardware for key generations and signing. However, falcon had way faster verification. Overall SPHINCS+ showed that it was not the best choice for avionic hardware. With signing times from 1 to 4.7 seconds. After this round of testing, we saw that, purely in terms of performance on avionic hardware, KYBER and then HQC would be the two best choice for KEM algorithms while Dilithium and Falcon would be the 2 best choices for signing algorithms.

Following our initial benchmark, we proceeded to present the results obtained from running our DTLS handshake experiments. The data that was gathered from these handshakes were the total communication size, the CPU times and the simulated transmission delays.

One of the key observations drawn from these results is that, in avionic communication, the increased computational requirements of post-quantum algorithms have a lesser impact compared to their higher bandwidth usage. With these data, we could conclude that the best choice for a KEM algorithm in ATN/IPS would be KYBER and the best choice for a signing algorithm would be Falcon. Furthermore, for ATN/IPS, choosing Kyber768 instead of classical algorithms for key exchange would result in a DTLS handshake 113% slower. Similarly, choosing Falcon1024 instead P384-ECDSA would result in a DTLS handshake being 112% slower. In other words, transitioning from ECC to the most efficient post-quantum algorithms would lead to a doubling of the handshake completion time for each algorithm type.

7.2 Limitations

During this research, we mentioned a few limitations that we came across. Some were directly related to our experimental setup while some others were simply due to lack of time.
First off, we failed to experiment with all the algorithms that we aimed to use initially. Indeed, the implementation that we had for BIKE was not available on 32-bit systems which meant that it could not run on our P2020 card. Furthermore, while we were able to benchmark SPHINCS+ and kyber1344, we encountered a bug that was only present when using the card. And finally, since McEliece was not initially part of the liboqs openssl fork due to its key sizes being larger than the limit in the DTLs spec, more time would have been required to add its support to our implementation.

Another limitation is the fact that we had to simulate our transmission delay to estimate the handshakes in air-ground communications. While we do compare our results to a baseline using ECC, the actual results does not represent reality accurately. Additionally, our simulation could have been more complete by introducing errors in our communication model.

Ultimately, although we were able to estimate certain memory requirements based on our results, our analysis would have been more comprehensive if we had conducted a memory analysis concurrently with our handshakes. However, due to the complexity of our card and its operating system, we were not able to crosscompile such tool in time.

7.3 Future Research

Our limitations demonstrated that there are numerous opportunities for improvement within our research. First, while refining the Simulink model would be a good step towards improvement, using real radios would replicate ATN/IPS more realistically. When we tried to use Software Defined Radios for our experiment, we came to the conclusion that using them with our implementation was not feasible in the allocated time for this research.
Secondly, a more in-depth analysis of the performance taken by the post-quantum handshakes on the card would be valuable to deepen our understanding of the complete impact of transitioning to post-quantum algorithms.

An additional scenario that could be covered in the analysis of the impact for 2 way authentication with post-quantum certificates.
Moreover, a few implementations of DTLS1.3 became available recently. While the feature

it is still not available to OpenSSL to the best of our knowledge, it has been made available in WolfSSL, an SSL/TLS library specifically made for embedded systems. It would also be interesting to compare our results with DTLS1.2 with results generated using DTLS1.3 [75]. Furthermore, it would be intriguing to compare our findings with wolfSSL's implementation of DTLS1.2.

Finally, after we explored the impacts on transmission delays using VHF and HF, the satellite communication route could be explored in the future. Satcom is usually more expensive, however there are newer projects like Starlink from SpaceX which offer low end-to-end latency rates and high data-rates at relatively low-cost [76]. Consequently, if the communication overhead of post-quantum algorithms becomes excessively significant, it may be worth considering cost-effective satellite communication options for the DTLS handshake.

REFERENCES

[1] D. Stebila and M. Mosca, "Post-quantum key exchange for the internet and the open quantum safe project," in *Selected Areas in Cryptography–SAC 2016: 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*. Springer, 2017, pp. 14–37.

[2] T. N. Alexandre, P. Alain, and L. Nicolas, "Managing aircraft mobility in a context of the atn/ips network," in *2019 IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*. IEEE, 2019, pp. 1–9.

[3] P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997, arXiv: quant-ph/9508027. [Online]. Available: http://arxiv.org/abs/quant-ph/9508027

[4] International Civil Aviation Organization I.C.A.O, "Manual on the aeronautical telecommunication network (atn) using internet protocol suite (ips) standards and protocols 2nd edition," 2011.

[5] M. Niraula *et al.*, "Atn/ips security approach: Two-way mutual authentication, data integrity and privacy," in *2018 Integrated Communications, Navigation, Surveillance Conference (ICNS)*, 2018, pp. 1A3–1–1A3–17.

[6] L. K. Grover, "A fast quantum mechanical algorithm for database search," *arXiv:quant-ph/9605043*, Nov. 1996, arXiv: quant-ph/9605043. [Online]. Available: http://arxiv.org/abs/quant-ph/9605043

[7] C. H. Bennett *et al.*, "Strengths and Weaknesses of Quantum Computing," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1510–1523, Oct. 1997, arXiv: quant-ph/9701001. [Online]. Available: http://arxiv.org/abs/quant-ph/9701001

[8] M. Schlosshauer, "Quantum decoherence," *Physics Reports*, vol. 831, pp. 1–57, 2019.

[9] G. Brassard *et al.*, "Quantum computing," *Proceedings of the National Academy of Sciences*, vol. 95, no. 19, pp. 11 032–11 033, 1998.

[10] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[11] D. Stebila, "The current status of post-quantum cryptography," p. 22.

[12] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[13] E. Crockett, C. Paquin, and D. Stebila, "Prototyping post-quantum and hybrid key exchange and authentication in tls and ssh," *Cryptology ePrint Archive*, 2019.

[14] E. B. Barker, M. Smid, and D. Branstad, "A profile for us federal cryptographic key management systems," 2015.

[15] E. Rescorla and N. Modadugu, "Datagram transport layer security version 1.2," Tech. Rep., 2012.

[16] M. S. Ben Mahmoud, A. Pirovano, and N. Larrieu, "Aeronautical communication transition from analog to digital data: A network security survey," *Computer Science Review*, vol. 11-12, pp. 1–29, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574013714000021

[17] R. Shirey, "Rfc 4949: Internet security glossary, version 2," 2007.

[18] T. Dierks and E. Rescorla, "Rfc 5246: The transport layer security (tls) protocol version 1.2," 2008.

[19] R. Housley *et al.*, "Rfc2459: Internet x. 509 public key infrastructure certificate and crl profile," 1999.

[20] E. Rescorla, "Rfc 8446: The transport layer security (tls) protocol version 1.3," 2018.

[21] G. Alagic *et al.*, "Status report on the third round of the nist post-quantum cryptography standardization process," *US Department of Commerce, NIST*, 2022.

[22] C. Paquin, D. Stebila, and G. Tamvada, "Benchmarking post-quantum cryptography in tls," in *Post-Quantum Cryptography: 11th International Conference, PQCrypto 2020, Paris, France, April 15–17, 2020, Proceedings 11*. Springer, 2020, pp. 72–91.

[23] International Civil Aviation Organization I.C.A.O, "Digital data & voice communication systems ed 2," vol. Annex 10 Volume III, 2007.

[24] ——, "Manual on vhf digital link (vdl) mode 2," 2015.

[25] E. Rescorla, H. Tschofenig, and N. Modadugu, "Rfc 9147: The datagram transport layer security (dtls) protocol version 1.3," 2022.

[26] D. Hofheinz and E. Kiltz, "Secure hybrid encryption from weakened key encapsulation," in *Advances in Cryptology-CRYPTO 2007: 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007. Proceedings 27*. Springer, 2007, pp. 553–571.

[27] E. Alkim *et al.*, "Frodokem learning with errors key encapsulation," 2017.

[28] "Ieee standard specifications for public-key cryptography," *IEEE Std 1363-2000*, pp. 1–228, 2000.

[29] W. Diffie, P. C. Van Oorschot, and M. J. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes and cryptography*, vol. 2, no. 2, pp. 107–125, 1992.

[30] D. Moody *et al.*, "Status report on the second round of the NIST post-quantum cryptography standardization process," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep. NIST IR 8309, Jul. 2020. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8309.pdf

[31] L. Chen *et al.*, "Report on Post-Quantum Cryptography," National Institute of Standards and Technology, Tech. Rep. NIST IR 8105, Apr. 2016. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf

[32] D. Micciancio and O. Regev, *Lattice-based Cryptography*, 2008.

[33] R. A. Perlner, "Quantum Resistant Public Key Cryptography: A Survey," p. 9.

[34] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *International Algorithmic Number Theory Symposium*. Springer, 1998, pp. 267–288.

[35] J. Hoffstein *et al.*, "Choosing Parameters for NTRUEncrypt," in *Topics in Cryptology – CT-RSA 2017*, H. Handschuh, Ed. Cham: Springer International Publishing, 2017, vol. 10159, pp. 3–18, series Title: Lecture Notes in Computer Science. [Online]. Available: http://link.springer.com/10.1007/978-3-319-52153-4_1

[36] J.-P. D'Anvers *et al.*, "Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM," Tech. Rep. 230, 2018. [Online]. Available: http://eprint.iacr.org/2018/230

[37] G. Alagic *et al.*, "Status report on the first round of the NIST post-quantum cryptography standardization process," National Institute of Standards and

Technology, Gaithersburg, MD, Tech. Rep. NIST IR 8240, Jan. 2019. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2019/NIST.IR.8240.pdf

[38] J. Bos *et al.*, "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. London: IEEE, Apr. 2018, pp. 353–367. [Online]. Available: https://ieeexplore.ieee.org/document/8406610/

[39] L. Ducas *et al.*, "CRYSTALS-Dilithium," p. 30.

[40] D. J. Bernstein *et al.*, "Ntru prime: reducing attack surface at low cost," in *International Conference on Selected Areas in Cryptography*. Springer, 2017, pp. 235–260.

[41] P.-A. Fouque *et al.*, "Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU," p. 67.

[42] C. Peng *et al.*, "Isogeny-based cryptography: A promising post-quantum technique," *IT Professional*, vol. 21, no. 6, pp. 27–32, 2019.

[43] L. De Feo, "Mathematics of isogeny based cryptography," *arXiv preprint arXiv:1711.04062*, 2017.

[44] S. Team *et al.*, "Sike and sidh are insecure and should not be used," 2022.

[45] R. Azarderakhsh *et al.*, "Supersingular isogeny key encapsulation," *Submission to the NIST Post-Quantum Standardization project*, 2017.

[46] R. Overbeck and N. Sendrier, "Code-Based Cryptography," Feb. 2009, pp. 95–145.

[47] R. W. Hamming, "Error detecting and error correcting codes," *The Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, Apr. 1950, conference Name: The Bell System Technical Journal.

[48] A. M. Steane, "Error correcting codes in quantum theory," *Physical Review Letters*, vol. 77, no. 5, p. 793, 1996.

[49] R. J. McEliece, "A public-key cryptosystem based on algebraic," *Coding Thv*, vol. 4244, pp. 114–116, 1978.

[50] N. Aragon *et al.*, "Bike: bit flipping key encapsulation," 2017.

[51] C. A. Melchor *et al.*, "Hamming quasi-cyclic (hqc)," *NIST PQC Round*, vol. 2, pp. 4–13, 2018.

[52] J. Ding and B.-Y. Yang, "Multivariate public key cryptography," in *Post-quantum cryptography*. Springer, 2009, pp. 193–241.

[53] J. Ding and A. Petzoldt, "Current state of multivariate cryptography," *IEEE Security & Privacy*, vol. 15, no. 4, pp. 28–36, 2017.

[54] J. Ding and D. Schmidt, "Rainbow, a new multivariable polynomial signature scheme," in *International Conference on Applied Cryptography and Network Security*. Springer, 2005, pp. 164–175.

[55] A. Petzoldt, S. Bulygin, and J. Buchmann, "Selecting parameters for the rainbow signature scheme," in *International Workshop on Post-Quantum Cryptography*. Springer, 2010, pp. 218–240.

[56] A. Casanova *et al.*, "Gemss: A great multivariate short signature," *Submission to NIST*, 2017.

[57] D. Butin, "Hash-based signatures: State of play," *IEEE security & privacy*, vol. 15, no. 4, pp. 37–43, 2017.

[58] O. Goldreich and H. Krawczyk, "On the composition of zero-knowledge proof systems," *SIAM Journal on Computing*, vol. 25, no. 1, pp. 169–192, 1996.

[59] D. Moody *et al.*, "Status report on the second round of the nist post-quantum cryptography standardization process," 2020.

[60] D. J. Bernstein *et al.*, "Sphincs: practical stateless hash-based signatures," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 368–397.

[61] M. Chase *et al.*, "Post-quantum zero-knowledge and signatures from symmetric-key primitives," in *Proceedings of the 2017 acm sigsac conference on computer and communications security*, 2017, pp. 1825–1842.

[62] G. Tasopoulos *et al.*, "Performance evaluation of post-quantum tls 1.3 on embedded systems."

[63] J. Sepúlveda, S. Liu, and J. M. Bermudo Mera, "Post-Quantum Enabled Cyber Physical Systems," *IEEE Embedded Systems Letters*, vol. 11, no. 4, pp. 106–110, Dec. 2019, conference Name: IEEE Embedded Systems Letters.

[64] G. Tasopoulos *et al.*, "Performance evaluation of post-quantum tls 1.3 on resource-constrained embedded systems," in *Information Security Practice and Experience: 17th International Conference, ISPEC 2022, Taipei, Taiwan, November 23–25, 2022, Proceedings*. Springer, 2022, pp. 432–451.

[65] R. Naeem Akram *et al.*, "Security and performance comparison of different secure channel protocols for avionics wireless networks," *arXiv e-prints*, pp. arXiv–1608, 2016.

[66] D. M. Mielke *et al.*, "Getting civil aviation ready for the post quantum age with ldacs," 2021.

[67] SYSGO. (2012) Sirio panel selected sysgo for avionics equipment. [Online]. Available: https://www.sysgo.com/press-releases/sirio-panel-selected-sysgo-for-avionics-equipment

[68] O. Salvador and D. Angolini, *Embedded Linux Development with Yocto Project*. Packt Publishing Ltd, 2014.

[69] D. A. Joseph, V. Paxson, and S. Kim, "tcpdump tutorial," *University of California, EE122 Fall*, 2006.

[70] T. Güneysu *et al.*, "Proof-of-possession for kem certificates using verifiable generation," *Cryptology ePrint Archive*, 2022.

[71] mathworks. (2022) Rf toolbox. [Online]. Available: https://www.mathworks.com/help/rf/

[72] ——. (2022) Simulink documentation. [Online]. Available: https://www.mathworks.com/help/simulink/

[73] C. M. Rader, "A simple method for sampling in-phase and quadrature components," *IEEE Transactions on Aerospace and Electronic Systems*, no. 6, pp. 821–824, 1984.

[74] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Post-Quantum Authentication in TLS 1.3: A Performance Study," in *Proceedings 2020 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2020. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2020/02/24203.pdf

[75] G. Restuccia, H. Tschofenig, and E. Baccelli, "Low-power iot communication security: On the performance of dtls and tls 1.3," in *2020 9th IFIP International Conference on Performance Evaluation and Modeling in Wireless Networks (PEMWN)*. IEEE, 2020, pp. 1–6.

[76] N. Mäurer *et al.*, "Security in digital aeronautical communications a comprehensive gap analysis," *International Journal of Critical Infrastructure Protection*, vol. 38, p. 100549, 2022.

APPENDIX A GENERAL INFORMATION ON POST-QUANTUM ALGORITHMS

Parameter set	NIST Level	Public key size (bytes)	Secret key size (bytes)	Signature size (bytes)
Dilithium2	2	1312	2528	2420
Dilithium3	3	1952	4000	3293
Dilithium5	5	2592	4864	4595

Table A.1 Information on Dilithium [1]

Parameter set	NIST Level	Public key size (bytes)	Secret key size (bytes)	Signature size (bytes)
Falcon-512	1	897	1281	666
Falcon-1024	5	1793	2305	1280

Table A.2 Information on Falcon [1]

Parameter set	NIST Level	Public key size (bytes)	Secret key size (bytes)	Signature size (bytes)
SPHINCS+-SHA256-128f-r	1	32	64	17088
SPHINCS+-SHA256-192f-r	3	48	96	35664
SPHINCS+-SHA256-256f-r	5	64	128	49856

Table A.3 Information on SPHINCS+ [1]

Parameter set	NIST Level	Public key size (bytes)	Secret key size (bytes)	Ciphertext size (bytes)	Shared secret size (bytes)
Kyber512	1	800	1632	768	32
Kyber768	3	1184	2400	1088	32
Kyber1024	5	1568	3168	1568	32

Table A.4 Information on Kyber [1]

Parameter set	NIST Level	Public key size (bytes)	Secret key size (bytes)	Ciphertext size (bytes)	Shared secret size (bytes)
FrodoKEM-640-AES	1	9616	19888	9720	16
FrodoKEM-976-AES	3	15632	31296	15744	24
FrodoKEM-1344-AES	5	21520	43088	21632	32

Table A.5 Information on FrodoKEM [1]

Parameter set	NIST Level	Public key size (bytes)	Secret key size (bytes)	Ciphertext size (bytes)	Shared secret size (bytes)
HQC-128	1	2249	2289	4481	64
HQC-192	3	4522	4562	9026	64
HQC-256	5	7245	7285	14469	64

Table A.6 Information on HQC [1]

Parameter set	NIST Level	Public key size (bytes)	Secret key size (bytes)	Ciphertext size (bytes)	Shared secret size (bytes)
C-McEliece-348864	1	261120	6452	128	32
C-McEliece-460896	3	524160	13568	188	32
C-McEliece-6688128	5	1044992	13892	240	32
C-McEliece-6960119	5	1047319	13908	226	32
C-McEliece-8192128	5	1357824	14080	240	32

Table A.7 Information on Classical-McEliece [1]