Design and Implementation of a Testbed Setup for Evaluating PQC and QKD Integration in Critical Infrastructure Environments

MSc Thesis

Márton Reiter

mreite23@student.aau.dk

Aalborg University
Cyber Security

AALBORG UNIVERSITY
STUDENT REPORT

Cyber Security
Aalborg University
https://www.aau.dk

Title:
Design and Implementation of a Testbed Setup for Evaluating PQC and QKD Integration in Critical Infrastructure Environments

Theme:
Cyber Security

Thesis Period:
Spring Semester 2025

Supervisors:
Edlira Dushku
Sokol Kosta

Page Numbers: 58

Date of Completion:
June 4, 2025

Key words:
post-quantum cryptography, quantum key distribution, critical infrastructure, hybrid cryptographic systems, integration, testbed

Abstract:

As quantum computing advances, classical cryptographic systems face new threats to their long-term security. This thesis investigates the integration of quantum-safe technologies (QKD and PQC) into application-layer communication. A custom testbed was developed to evaluate performance in two illustrative scenarios: a radar-based critical infrastructure use-case and a generalized high-load scenario. The radar use-case combined symmetric encryption with QKD-derived keys and PQC-based message authentication, while the high-load scenario simulated intensive QKD key requests to a single device. Experimental evaluation focused on latency and the operational stability of QKD devices under both conditions. Results show that PQC can be effectively integrated into existing systems with minimal disruption, whereas QKD integration remains challenging due to current hardware limitations. The findings offer practical insights into deploying quantum-resistant systems and emphasize design considerations for future architectures operating in constrained or performance-sensitive environments.

Contents

Acknowledgements

Aalborg University June 4, 2025

I would like to express my gratitude to my supervisors, Edlira Dushku and Sokol Kosta, for their valuable guidance and support throughout this research. I am also grateful to PhD fellows Mieszko Jan Ferens Michalek and Marius Cotiga, for their guidance and help during the testbed implementation and showcasing. I would also like to thank Shreyas Srinivasa from Terma, for providing insights on the radar use-case and organizing the showcase at the headquarters of Terma.

Acronyms

AAU Aalborg University. 28–30

AES Advanced Encryption Standard. 3, 6, 7, 16, 19, 22, 26, 31–33

AI Artificial Intelligence. ii, 13

ASTERIX All Purpose Structured Eurocontrol Surveillance Information Exchange. 19–21, 47–50

C2 Command and Control. 20–22, 24, 25, 40

CA Certificate Authority. 16

CoAP Constrained Application Protocol. 14

CPS Cyber-Physical System. 14

DER Distributed Energy Resources. 16, 45

DERMS Distributed Energy Resource Management System. 16

DoS Denial of Service. 22

DTLS Datagram Transport Layer Security. 14

ECC Elliptic-Curve Cryptography. 4, 19

ECDHE Elliptic-Curve Diffie-Hellman. 16

FORS Forest of Random Subsets. 7

GCM Galois/Counter Mode. 32

GUI Graphical User Interface. 28, 38, 39

HMAC Hash-Based Message Authentication Code. 19

HTTP Hypertext Transfer Protocol. 26, 31, 33, 35, 36, 47, 49, 50

HTTPS Hypertext Transfer Protocol Secure. 27, 29, 31, 50

IKM Input Key Material. 16

IoT Internet of Things. 14

IQR Interquartile Range. 41, 42

ISP Internet Service Provider. 50

JSON JavaScript Object Notation. 31–33

KEM Key Encapsulation Mechanism. 14, 16

KMS Key Management System. 16, 22, 25, 26, 28, 31, 33, 47, 48

LWE Learning With Errors. 6

MAC Message Authentication Code. 19

MitM Man-in-the-Middle. 21, 22

ML-DSA Module-Lattice-Based Digital Signature Algorithm. 7, 16, 22, 24, 26, 31–33, 46, 47

ML-KEM Module-Lattice-Based Key-Encapsulation Mechanism. 6, 7, 16

MLWE Module Learning With Errors. 6, 7

MoSCoW Must-have, Should-have, Could-have, and Won't-have. 12

MQTT-SN Message Queuing Telemetry Transport for Sensor Networks. 14

NIST National Institute of Standards and Technology. 1, 6, 7, 12, 22

OOB Out of Bound. 3

OPC UA Open Platform Communications Unified Architecture. 14

OSI Open Systems Interconnection. 19

PQC Post-Quantum Cryptography. i–iii, 1–3, 6–8, 11, 12, 14–18, 22, 45–48, 50, 52

QKD Quantum Key Distribution. i–iii, 1–3, 6, 8–19, 21, 22, 24–29, 31, 33, 34, 36–53

RADAR Radio Detection And Ranging. 11, 19

RSA Rivest–Shamir–Adleman. 4, 19, 50

SDN Software Defined Network. 15, 16

SFP Small Form-factor Pluggable. 12

SHA Secure Hash Algorithm. 7, 48

SLH-DSA Stateless Hash-Based Digital Signature Algorithm. 7

SSL Secure Sockets Layer. 14, 31

TCP Transmission Control Protocol. 49

TLS Transport Layer Security. 14–16, 19

UDP User Datagram Protocol. iii, 47–50

WOTS+ Winternitz One-Time Signature Plus. 7

Chapter 1

Introduction

With the rise of quantum computers [1, 2], cyberattacks by quantum-capable adversaries become a real threat to the cryptographic systems of today. As a result, technologies that can resist quantum attacks are more important than ever. Securing critical infrastructure and military systems is especially urgent, due to their key roles in public safety, economic stability, and national security.

To counter quantum adversaries, two different approaches exist today. Quantum Key Distribution (QKD) is a way to exchange keys in a secure way using quantum physics [3], while Post-Quantum Cryptography (PQC) algorithms provide security, based on mathematical problems that are hard to solve, even for quantum computers [4, 5, 6].

With the National Institute of Standards and Technology (NIST) accepting last year three PQC standards (FIPS-203 [4], FIPS-204 [5], FIPS-205 [6]), PQC became an applicable solution to secure communication against quantum attacks. These algorithms, similarly to classical cryptography solutions, take advantage of mathematical problems, that are considered to be hard to solve even with a quantum computer. PQC algorithms are also easily integrable into conventional systems, but operate with much larger key sizes.

In the last couple of years, another technology, called QKD has been commercialized. By incorporating eavesdropping detection and providing theoretically proven secure key exchange by the laws of physics, QKD is an effective way to combat quantum adversaries. Although QKD currently needs expensive special equipment and is limited in operational distance, clearly it is an approachable solution on the market, which can be integrated into different communication layers.

While both PQC and QKD provide effective security against quantum attacks, each of them lacks practicalities. To help overcome these individual difficulties, the following research question is posed: How can PQC and QKD be effectively integrated and evaluated to strengthen critical infrastructure systems? To guide investigation in the field of integrating different quantum-resistant technologies, the following sub-questions are posed:

- What are the challenges of creating a QKD testbed to perform key exchange?

- How can PQC be integrated into the QKD system to provide authenticated message exchange?
- How suitable is the developed testbed for assessing secure message exchange under a radar communication and a generalized scenario?

By addressing these questions, this thesis intends to extend both theoretical and practical insights into the deployment of integrated PQC-QKD systems in critical infrastructure to protect message exchange against quantum adversaries.

1.1 Contributions

This thesis contributes to the research of the integration of quantum-resistant technologies by designing and implementing a QKD testbed, implementing application-level integration of PQC and QKD technologies, and measuring and evluating the implementation in different use-cases, related to critical infrastructure. The main challenge of building a QKD testbed were found to be device constraints, and finding proper equipment to simulate real-life scenarios. The thesis includes a system design, where PQC and QKD are integrated on the application-level, where QKD keys serve as symmetric encryption keys, while PQC key pairs were used to create digital signatures, providing authenticity and integrity of exchanged messages. It was found, that the developed testbed is not fully suitable for assessing secure message exchange in critical infrastructure scenarios, due to unstable QKD retrieval times and quantity limitations.

1.2 Outline

This thesis is structured into nine chapters that collectively explore the integration of quantum-safe technologies into critical infrastructure systems. Chapter 2 provides the necessary background, introducing the theoretical and technical foundations of QKD, PQC, and symmetric encryption. Chapter 3 outlines the methodology, describing the development of two use-case scenarios — a critical infrastructure context and a generalized high-load environment — along with the tools, software, and evaluation approach used. Chapter 4 presents the current state of the art in integrating QKD and PQC into real-world systems, identifying key developments and research gaps. Chapter 5 details the design of the system and a QKD testbed, including both integration scenarios, while Chapter 6 covers the implementation of this design. Chapter 7 evaluates the performance of the implemented system, focusing on latency, system stability, and the behavior of QKD devices under constrained conditions. Chapter 8 provides a critical discussion of the findings, considering their implications, limitations, and relation to the original objectives. Finally, Chapter 9 concludes the thesis by summarizing the results, reflecting on the research questions, and outlining potential directions for future work.

Chapter 2

Background Research

This chapter provides an overview of the foundational technologies and theory relevant to this thesis, including the principles and algorithms of symmetric encryption, PQC and QKD. Understanding the technical and theoretical landscape surrounding these technologies is essential for evaluating their practical viability and for designing robust systems that can operate under realistic critical infrastructure use-cases.

2.1 Introduction to Cryptography

Modern cryptography plays a crucial role in secure communication and is a part of our everyday life [7]. With the proper application of cryptography, parties exchanging messages can make sure that the original content of the messages stays secret, even in the case a third party intercepts their messages. The process of making a message secure and unreadable to anyone not possessing the key is called encryption, while the reverse of this process is called decryption. In cryptography today, there are two main applications: symmetric and asymmetric cryptography.

2.1.1 Symmetric Cryptography

In symmetric cryptography, message exchanging parties must agree on a shared key prior to exchanging messages, which usually happens with the help of asymmetric cryptography or an Out of Bound (OOB) communication channel. The strength of the encryption of the messages mainly relies on the size and randomness of the used key. Only the entities possessing the key are able to decrypt the messages, which also implies that in case the key is leaked, the security of the message exchange system is compromised and messages exchanged with the given key are not secret anymore. The most widely used symmetric cryptography system is the Advanced Encryption Standard (AES) system.

2.1.2 Asymmetric Cryptography

In asymmetric cryptography, every messaging party has two keys, a private and a public one. The private key must be kept secret, while the public key can be distributed to anyone. In an Alice and Bob scenario, if Alice wants to send a message to Bob, she needs to obtain Bob's public key in a secure way, encrypt her message with it and send it to Bob. In case the right public key was obtained (which sometimes is a difficult task), only Bob will be able to decrypt the message and read its content with the help of his private key. This usage can be reverse as well, if Bob encrypts a message with his private key, the cryptography system guarantees that the message can only be decrypted with Bob's public key, therefore the authenticity of the message can be verified. The security of an asymmetric cryptography system strongly relies on private keys being kept secret and obtaining the right public key. The most widely used asymmetric cryptography systems are the Rivest–Shamir–Adleman (RSA) and the Elliptic-Curve Cryptography (ECC) systems. Due to performance reasons, in many application today asymmetric cryptography is used to generate a shared symmetric key and then the secure communication happens with the help of symmetric encryption and decryption. The security of these cryptography systems relies on hard mathematical problems that are hard to solve with conventional computers. In the case of RSA, this hard problem is prime factoring. Specifically, from a given large t composite number, it is hard to find p and q large primes, such as the equation $t = p * q$ becomes true [8]. In ECC cryptography system, the hard mathematical problem is the discrete logarithm problem, where from given b and x it is hard to find an integer y that satisfies the equation $y = log_b\,(x)$ [9, 10].

2.2 The Threat of Quantum Computing to Classical Cryptography

Quantum computing is a rapidly emerging technology starting back in the 80's and relying on quantum mechanics and physics. Today, quantum computers are not cheaper or more effective to complete regulars tasks compared to a classical (also called conventional) computers. However, they are able to perform tasks that no classical computer could complete in any reasonable time, which is called the quantum threshold.

2.2.1 Fundamentals of Quantum Computing

Compared to bits in a conventional computer [3], the computational unit of quantum computers are called qubits, which can be realized with different quantum mechanical phenomena, e.g., the polarization of photons. Qubits can be zero, one or in a linear combination of zero and one. A qubit before measurement exists in a state called superposition, which is a probability distribution between zero and one with some amplitudes. The ability of qubits to correlate their state with other qubits is called entanglement. This means that when the qubits are being measured, their results will be mathematically related, providing the advantage of gaining information about other qubits by measuring a single qubit. Also, qubits can contain significantly more information compared to classical bits. For example, 10 qubits can all exist in superposition and containing 2^{10} values at the same time, while 10 classical bits could only contain one value at a time.

A quantum system, such as a set of entangled qubits, falls into a non-quantum state when being measured, which means all qubits loose their superposition and collapse into either zero or one, and lose their information advantage against classical bits. This process of a quantum system collapsing into a non-quantum state is called decoherence. In order to extract valuable information from a quantum system, one must think about the information as waves, where amplitudes are probabilities of possible outcomes, and use interference on the waves with the help of a quantum algorithm to determine the most possible outcome of all. Despite all the different mechanisms, in general it can be said, that quantum computers are not faster in single operations than conventional computers, but they possess the ability to process data with quantum logic at parallel instances, which is a great advantage compared to classical computers, which process data with classical logic and sequentially.

2.2.2 Threats from Quantum Algorithms

Quantum algorithms were already designed in the 90's, but at that time no quantum computer existed or was stable enough to run them. By technological development, quantum computers became stable and efficient enough to be used in modern research today. The two quantum algorithms that affect the cybersecurity landscape today are Shor's algorithm for quantum factoring and Grover's algorithm for searching.

Shor's Algorithm

In 1994, Peter W. Shor published his research [11, 12] on factoring integers and finding discrete logarithms. He proposed algorithms for the aforementioned two problems where the number of steps required were polynomial in the input size. By quantum computers evolving and becoming available for more and more use cases, the application of the integer factoring algorithm is a threat to modern asymmetric cryptographic solutions. As mentioned in 2.1.2, the security of asymmetric cryptography relies on the hard mathematical problem of prime factoring. Shor's algorithm implemented on a quantum computer eliminates the difficulty in such operations, therefore their security is breakable.

Grover's Algorithm

Two years after Shor published his research, another mathematician, Lov K. Grover made his work public on a quantum searching algorithm [13] about increasing search efficiency. In his proposed algorithm, given a database with N records, he reduced significantly the maximum required searching operations from *O(N)* to *O($\sqrt{N}$)*, which can be very helpful to break symmetric cryptography systems. In itself, the algorithm is not a fatal threat, since increasing the key size, e.g., in AES, will result in similar security against quantum computers as a lower key size would result against conventional computers.

Despite the fact that quantum computers are not yet commercially available today, for the above reasons, research has been done to develop solutions that are resistant against quantum computing attacks, resulting in PQC and QKD.

2.3 Post-Quantum Cryptography

Post-quantum cryptography, or in other words quantum-resistant cryptographic solutions, are algorithms that, in an analogous manner to classical cryptographic systems, rely on mathematical problems that are considered to be hard to solve even with the help of a quantum computer. These algorithms can be implemented and used on conventional computers and provide the same security against quantum computers as classical cryptographic solutions would provide against conventional computers. The main standardizing body for such systems is NIST, who finalized three standards (FIPS-203, FIPS-204, FIPS-205) in 2024 that can be used for public key cryptography and key encapsulation.

2.3.1 PQC for Key Encapsulation

Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM), FIPS-203

The ML-KEM standard [4] can be observed as the post-quantum replacement for the Diffie-Helman algorithm [14], it ensures that two parties can exchange a shared secret key that can later be used for symmetric cryptography. The security of the algorithm lies in the Module Learning With Errors (MLWE) problem, which extends the classical Learning With Errors (LWE) problem into module structures over polynomial rings, enhancing computational complexity and security. Compared to mathematically hard problems in classical cryptography, the MLWE problem involves solving linear equations perturbed by small, random errors within a module structure, where it is hard to find a solution for a set of linear equations, where the right side of equations has been modified by errors, making it resistant to both classical and quantum attacks. When en- and decryption happens, errors given by the nature of module arithmetics, will only slightly deviate the result of the addition of a subset of the equations, which will hang around zero or one. ML-KEM offers three parameter sets, which claim to be as secure as a generic block cipher with a prescribed key size or a generic hash function with a prescribed output length:

- ML-KEM-512: Aligns with NIST security level 1, i.e., as secure as AES-128.
- ML-KEM-768: Corresponds to NIST security level 3, i.e., as secure as AES-192.

- ML-KEM-1024: Matches NIST security level 5, i.e., as secure as AES-256.

2.3.2 PQC for Digital Signatures and Public Key Cryptography

These algorithms are asymmetric cryptography solutions whose main applications could be equivalent to classical systems.

Module-Lattice-Based Digital Signature Algorithm (ML-DSA), FIPS-204

The ML-DSA signature scheme [5] is also grounded in the hardness of the MLWE problem. ML-DSA also offers three parameter sets, each corresponding to different security levels as defined by NIST, where the security strength is a number associated with the amount of work (e.g., the number of operations) to break the cryptography system[1]:

- ML-DSA-44: Aligns with NIST security level 2, providing a security strength of approximately 128 bits.
- ML-DSA-65: Corresponds to NIST security level 3, offering around 192 bits of security strength.
- ML-DSA-87: Matches NIST security level 5, delivering about 256 bits of security strength.

Each parameter set defines specific configurations, including lattice dimensions and error distributions, to balance security and performance.

Stateless Hash-Based Digital Signature Algorithm (SLH-DSA), FIPS-205

SLH-DSA is based on secure hashes, utilizing the security of hash functions to generate digital signatures. It employs a hypertree structure combining Winternitz One-Time Signature Plus (WOTS+), a one-time signature scheme providing security against quantum attacks and a multi-time signature scheme designed for efficiency called Forest of Random Subsets (FORS). SLH-DSA also comes with different security levels, where each level is characterized by parameters such as the used hash function (e.g., SHA-256), tree height, number of layers of subtrees, Winternitz parameter for WOTS+, and the number of FORS trees and leaves. For example, with the hash function SHA2-128s the algorithm reaches NIST security level 1 with a signature size of 7 856 bytes. In contrast, with the hash function SHA2-256f the algorithm can reach NIST security level 5, but the signature size will also increase to 49 856 bytes. According to NIST, this standard serves as a backup for FIPS-204.

It can be seen that research was focused more on public key cryptography, namely digital signatures and key encapsulation, since Grover's algorithm can only lower the time

[1] `https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.204.pdf`, Page 4

to break symmetric encryption, which can be countered with larger key sizes. The biggest downside of PQC, is that all the above algorithms operate with much larger key sizes and ciphertext sizes compared to classical cryptography systems. On the other hand, PQC can be easily integrated with existing systems.

2.4 Quantum Key Distribution

In Quantum Key Distribution [15, 16, 17] the goal is to share a key between two parties, without anyone else eavesdropping on the shared key. The specific qubits used are realized with particles of light called photons. Photons, given their nature, can act as waves and have a polarization, which is the direction in which the electric field of the photon oscillates. The polarization can be measured for each individual photon against a given axis, and will be either up-down or side-to-side compared to the axis, which can be used as a binary representation. Polarization is a fundamental quantum mechanical property and photons can also be in a superposition, where they oscillate in both directions with a given probability. When measuring a photon in superposition, it will collapse into either one of the directions. In this thesis, the QKD devices operate with the Decoy-State BB84 protocol, therefore only this protocol is described.

2.4.1 BB84 Protocol

In the BB84 protocol [15], Alice has a polarized photon sender and Bob has a polarized photon receiver device, which both parties keep in a secure environment. Each of these devices can be turned to the side by 45°. The two states of the device define two basis, the rectilinear basis and the diagonal basis, as shown in Figure 2.1. In the rectilinear basis photons oscillating up-down mean one, photons oscillating side-to-side mean zero. In the diagonal basis photons oscillating in the form of a forward slash represent zero, the photons oscillating like a back slash represent one.

Each of the parties randomly rotates their device between the rectilinear and diagonal basis, but mathematically on average half the time they will end up choosing the same rotation. The two parties exchange bits by photons in an optical channel and after being done with this, they exchange, even in a public way, the rotation order in which they sent or measured the photons. The two parties drop the bits where they used different rotations and compare the rest of the bits, which will give the shared key if they match. This process can be seen in Figure 2.2. The security of this system relies in the superposition of photons and in the fact that, an eavesdropper has no knowledge about the rotation order of the sender or the receiver.

In the scenario where a third party tries to listen into the communication of Alice and Bob, they will have no information of whether the photons from Alice have been sent in the rectilinear or the diagonal basis. In case they mixed up the rotation of Alice's device, the photon that they see will be in the wrong diagonal basis and have a 50-50% chance

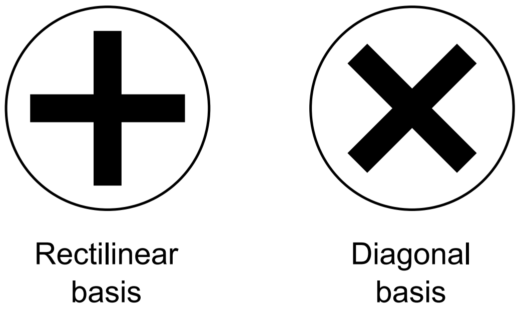
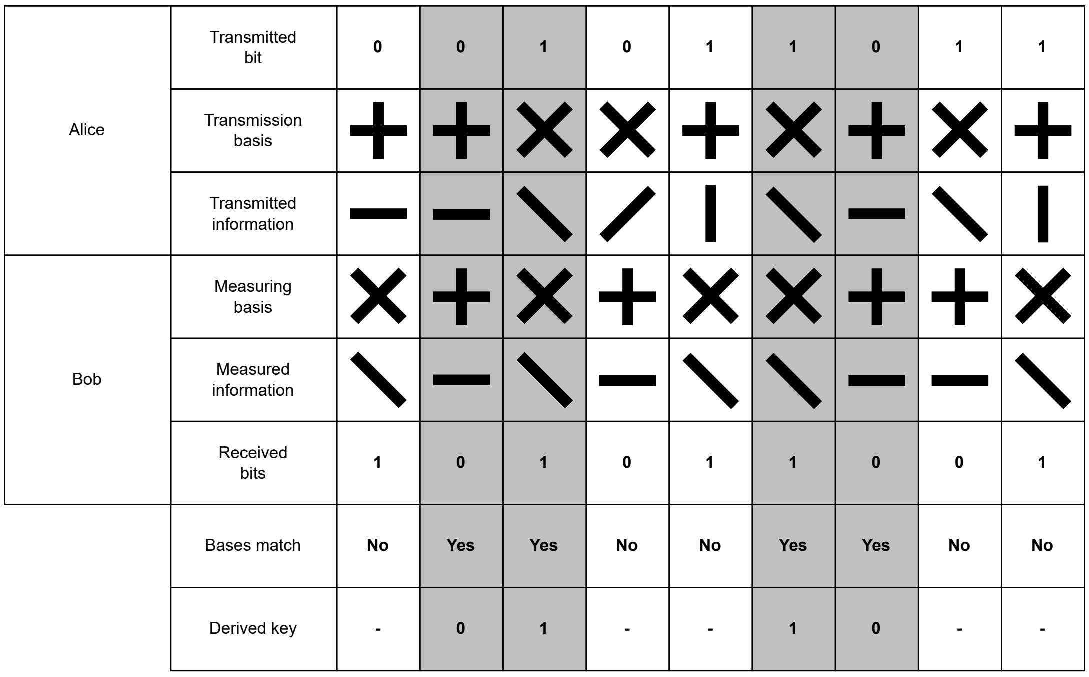


Figure 2.1: Illustrating the different rotations of a polarized photon device.

of collapsing into zero or one. Even in the case they match with Alice's device for one photon, since they lack information on the rotation of Bob's device, the same bit might be dropped at the end.

Decoy-State BB84

Due to the nature of highly attenuated lasers which are used as the source of photons in BB84, there is a chance that more than one photon is emitted into the optical channel. This opens the door for new eavesdropping attacks, such as the photon splitting attack, where an eavesdropping third party could capture one of the photons sent to Bob and keep it until Alice and Bob exchanges the rotation order of their devices and measure the photons accordingly to retrieve the key. In the article of Lo et al. [18], the authors proposed a solution for the above problem. The intensity of the laser source describes the probability of emitting one or more photons when the laser is used. In the Decoy-State BB84 protocol, Alice selects the intensity of the transmission of each photon randomly, i.e., she adds a random number of decoy states. After the key exchange is done, Alice and Bob share the polarization order of their devices, but also the different intensity Alice sent each of the bits with and the intensity Bob received them with. If the mismatch of the intensities reach a given threshold, the two parties can agree that there is an eavesdropper on the channel and the exchanged key is insecure.



Alice	Transmitted bit	0	0	1	0	1	1	0	1	1
	Transmission basis	+	+	×	×	+	×	+	×	+
	Transmitted information	—	—	\	/	|	\	—	\	|
Bob	Measuring basis	×	+	×	+	×	×	+	+	×
	Measured information	\	—	\	—	\	\	—	—	\
	Received bits	1	0	1	0	1	1	0	0	1
	Bases match	No	Yes	Yes	No	No	Yes	Yes	No	No
	Derived key	-	0	1	-	-	1	0	-	-

Figure 2.2: Illustrating the key derivation process of the BB84 protocol.

Chapter 3

Methodology

This chapter outlines the methods used to analyze the state of the art in PQC-QKD integration, choosing the QKD devices, implementing a testbed and the methodology of the performance evaluation under an emulated critical infrastructure scenario. The research follows an applied experimental design.

3.1 Analyzing the State of the Art

To showcase related works in the integration of PQC and QKD and to gain knowledge on the current status, research papers were first searched on Google Scholar in the given topic and collected into a list. Relevant papers were identified based on their abstract, contribution, authors, publication place and year of publication. After this, research papers were categorized based on their contribution to the integration of PQC, QKD, or both, and were grouped based on similar contributions in each category. Each paper were presented in correlation to the other papers in the same category, where the focus was on contribution and extraordinary achievements.

3.2 System Design

3.2.1 Critical Infrastructure Use-Case

The critical infrastructure scenario of Radio Detection And Ranging (RADAR) communication was established after a meeting with a company called Terma A/S, from now on referred to as Terma. They proposed the idea to emulate QKD key exchange, symmetric encryption and some kind of authentication in one test run, since it is a relevant data flow in radar communication of military applications. Terma also provided insight on radar flying object detection. Based on these information pieces, relevant research papers were identified to supply the problem statement. The Iron Dome missile detection and intervention system was selected by the author as a relevant application, with its capabilities investigated through news reports and articles. Given the nature of critical infrastructure, detailed documentation was not available. To estimate the number of flying missiles a radar would need to report, recent news articles and public reports were used. The relevant standard radar communication protocol was identified based on the information from Terma.

3.2.2 Generalized Use-Case

To extend the applicability of the testbed beyond a specific critical infrastructure scenario, a generalized use case was formulated. During the design discussions, the concept of simulating multiple concurrent applications interacting with a shared QKD server was proposed. Each application would independently request quantum keys at varying intervals to emulate diverse usage patterns. To introduce realistic variability in the timing of key requests, it was decided that inter-request delays should be modeled using a probabilistic distribution, thereby enabling the simulation of stochastic behavior commonly observed in real-world networked systems.

3.2.3 Requirements

Setting up the requirements involved a detailed analysis to define the functional, technical, and security needs for integrating PQC and QKD within a critical infrastructure communication. This was achieved through a combination of literature review, standards analysis (e.g., NIST PQC guidelines), attack analysis and system needs coming from Terma. The resulting requirements informed the selection of algorithms, communication design and testbed architecture to ensure the research remained aligned with realistic deployment conditions for securing critical infrastructure against quantum-era threats. To decide which requirements to fulfill during implementation, the Must-have, Should-have, Could-have, and Won't-have (MoSCoW) prioritization method was used.

3.3 Implementation

The implementation phase began with the procurement of the QKD devices, selected from multiple vendors, based on a balanced assessment of feature sets, pricing, and delivery timelines to ensure timely and practical deployment. Hands-on training was provided by the QKD vendor, complemented by additional online resources to build familiarity with device assembly and operational procedures. The physical integration of the QKD units was completed during the vendor-led sessions, where participants gained practical experience in handling and configuring the hardware. All essential network components—including switches, cables and Small Form-factor Pluggable (SFP)s were supplied by the vendor to ensure compatibility and minimize integration issues. The testbed was designed to simulate a two-site communication scenario, developed through an internal brainstorming process to reflect a realistic critical infrastructure use case involving secure data exchange between operational nodes. Applications developed for the testbed were version-controlled using Git and primarily implemented in Python, utilizing standard cryptographic libraries, including support for post-quantum algorithms.

3.4 Measuring and Evaluating Performance

3.4.1 Critical Infrastructure Use-Case

In the critical infrastructure use-case, the evaluation focused on the defined communication workflow for secure operational data exchange proposed in the system design. Application-level latency was measured independently for each step in the flow to provide fine-grained insight into performance bottlenecks and overhead introduced by quantum-safe cryptographic operations, key requests and network transport. These results were recorded, transferred to Excel, and visualized using `seaborn` and `matplotlib` in Python to create different diagrams to compare actual performance against the previously defined requirements in Subsection 5.1.3.

3.4.2 Generalized Use-Case

In the generalized use case, a simulation involving multiple QKD key requests was conducted on the application level to model scalable usage scenarios. Latency was measured at the application level for each key request to assess the responsiveness and performance of the corresponding QKD server under varying load conditions. Collected data was similarly fed into Excel and plotted using `seaborn` and `matplotlib` and compared against requirements defined in Section 5.2.1.

3.5 AI Disclosure

In this thesis the current version(s) of ChatGPT from 10th of March 2025 until 4th of June 2025 were used to rephrase writing, with all outputs checked for errors. ChatGPT did not contribute to any conclusion or results nor to any personal thoughts or scientific proofs and evaluation.

Chapter 4

State of the Art

As QKD had been commercialized, and PQC had been standardized, integration of these quantum-resistant security systems into already existing protocols and architectures has clearly began. Despite the fact that, a great portion of the integration efforts focus on the application layer, work has been done on other network levels as well. In this chapter, the thesis provides a summary on existing integrations, distinguished by the involvement of aforementioned quantum-resistant systems and focusing on hybrid integration models in industrial scenarios.

4.1 PQC Integration

PQC is often being integrated into the application layer, given its nature by which it can be used very similarly to classical cryptography systems. Industrial applications, such as the Open Platform Communications Unified Architecture (OPC UA) protocol, are an important target of integration, since they are often applied in critical environments and production lines. Paul et al. [19] investigated introducing PQC into the OPC UA protocol in Cyber-Physical Systems (CPSs) combined with classical cryptography, and measured its performance compared to classical and hybrid (PQC-classical) options. Other protocols in the application layer that are widely used in the IoT domain are Message Queuing Telemetry Transport for Sensor Networks (MQTT-SN) and Constrained Application Protocol (CoAP). Blanco-Romero et al. [20] published their work on testing PQC KEM in the TLS and Datagram Transport Layer Security (DTLS) parts of MQTT-SN and CoAP with the help of the SSL librdary wolfSSL[1]. Both research materials highlight that PQC has the best performance while it secures industrial protocols against quantum attacks.

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

4.2 QKD Integration

Application layer is also one of the easiest possibilities when it comes to integrating QKD systems. Sasaki et al. [21] in 2011 already showcased that, application layer integration of QKD is possible by using a video conference proof-of-concept where the keys used for securing the video stream were coming from QKD. The authors also gave proof of integration of different QKD protocols, such as BB84, DPS-QKD, BBM92 and SARG04.

Jumping forward in time, in the past 5 years research has been done [22, 23] to map the overall status of QKD networks. From the surveys from Mehic et al. [22] and Yuan et al. [23], it is visible, that QKD networks already exist and operate in some countries, and comparison between them is possible. Both authors also highlighted that, metropolitan or national networks spanning large distances require trusted nodes due to the limited link distance of QKD. Towards the future, the surveys emphasized integration experiments of QKD and other technologies, such as PQC.

Enabling longer distances in QKD networks, a novel approach is to use satellites. Liao et al. [24] developed a low-Earth-orbit satellite to establish QKD in the 1200 km distance using the decoy-state BB84 protocol. The researchers achieved kilohertz rate from satellite to ground, with QKD keys established between Beijing and Nanshan, which are roughly 2200 km from each other. Further developing satellite-based QKD and enabling even longer distances, in 2021, Chen et al. [25] achieved QKD over a 4 600 km distance. The network they used consisted of a 2 000 km large optical fiber backbone, with trusted nodes between the two endpoints, and two satellite nodes laying 2 600 km from each other, where from the furthest satellite link to the furthest end-node, the key share happens on 4 600 km.

4.3 Hybrid PQC-QKD Systems

A straightforward integration of PQC and QKD is to use QKD as seeding for PQC, and encrypt ongoing communication with PQC only. Ranganathan et al. [26] published their results of integrating QKD and the Kyber (FIPS-203) PQC algorithm in a scenario where the key established with the help of the BB84 QKD protocol, served as the seed for the Kyber algorithm, which was then used to establish public and private keys for Alice and Bob.

Knowing hybrid systems are achievable, research is also focused on securing internet communication, with a great focus on the Transport Layer Security (TLS) protocol, operating in the application, presentation, session, and transport layer of network communication. Software Defined Networks (SDNs) are a nieche part of networking with focus on controling and monitoring network resources, building on the security provided by TLS. Rubio Garcia et al. [27] made a proposal to secure SDNs by integrating both QKD and PQC systems into Transport Layer Security (TLS). In the proposed architecture, SDNs supply TLS-based applications with QKD keys, but also elevate security by combining PQC, QKD and classical cryptography in the network segments where optical fiber is available. For wireless links, the authors propose the use of PQC combined with classical cryptography. For a more general approach, Rubio Garcia et al. [28] implemented a quantum-resistant version of TLS 1.3, by applying Elliptic-Curve Diffie-Hellman (ECDHE) in combination with ML-KEM for key encapsulation, ML-DSA for digital signature generation and verification, and QKD as a complement to the aforementioned cryptography systems in a concatenation-based approach. The key generated by the combination of different cryptography systems then served as the Input Key Material (IKM) in TLS. To smoothen the integration of QKD into TLS, Blanco-Romero et al. [29] proposed a QKD Key Encapsulation Mechanism (KEM) to accommodate stateful (ETSI 004) and stateless (ETSI 014) QKD key management interfaces, where the QKD key is combined with a PQC shared secret, enabling direct QKD key retrieval from TLS implementations.

When PQC and QKD are not directly integrated, they are often used in a setup, where authentication happens by PQC and key exchange happens by QKD. Marchsreiter et al. [30] proposed PQC authenticated QKD key exchange. The authors experimented with QKD nodes, each node having a sender and a receiver device capable of the T12 QKD protocol. The experiment consisted of authentication, key generation and encryption (AES-CBC), authenticating with both pre-shared keys and keys shared by PQC KEM. Aquina et al. [31] also proposed a hybrid authentication mechanism composed of classical and post-quantum cryptography, and QKD based on the PRF-then-XOR split-key KEM combiner from the work of Giacon et al. [32]. Authentication can also be done with Certificate Authoritys (CAs). Geitz et al. [33] described QKD nodes, each incorporating a node Certificate Authority (CA), a node KMS and a QKD device, where the chain of trust is established through a root CA and PQC using the Falcon-1024 digital signature algorithm. The nodes can authenticate using PQC with the help of node certificates and the Kyber KEM, and the key exchange can happens by QKD, enabling easy network expansion.

Hybrid architectures are also considered in infrastructure communication. Hoque et al. [34] proposed a network architecture encapsulating both QKD and PQC technologies in order to establish quantum-resistant communication between mobile parties and network towers. In the architecture, QKD plays a vital role in securing communication between network towers, while PQC is used for endpoint security. Garcia et al. [35] published an architectural vision design for quantum-enabled 6G systems, where QKD plays a role in SDN control communication, and PQC and QKD ensures secure data transfer across all levels of the architecture. Ahn et al. [36] proposed quantum-resistant defense strategies including PQC and QKD to protect Distributed Energy Resources (DER). In the proposed architecture, QKD was used on the DER control network for securing communication between DER sectors and the Distributed Energy Resource Management System (DERMS), while PQC was introduced in the communication channels between the quantum transceivers of each sector and the smart inverters of each DER device in the sector, but still keeping encryption in the application layer.

Summary

To summarize conducted research, from the previous paragraphs it can be seen that, experiments and measurements in the integration of quantum-resistant technologies focus on integration in different layers of communication. This thesis contributes to research in the critical infrastructure sector and application-level integration, by implementing a QKD-PQC testbed and measuring it against given radar communication requirements proposed in Chapter 5, with focus on latency, stability and interoperability.

Chapter 5

System Design

This chapter presents the design of a testbed developed to evaluate the integration of PQC and QKD within critical infrastructure communication scenarios, creating evaluations use-cases and communication flow. The design process begins with the construction of a QKD-enabled test environment, serving as the foundation for all subsequent experimentation. A specific use case is then introduced: a radar communication scenario representing a critical infrastructure application, in which symmetric encryption, PQC, and QKD are jointly deployed to secure end-to-end message exchange. To support this use case, a structured communication flow is designed to model realistic message passing and key management between distributed components. In addition to this focused scenario, a generalized design is developed to simulate a higher-load environment, where a large volume of QKD key requests are directed toward a single QKD device, enabling stress testing and performance analysis under constrained resource conditions. Together, these designs provide the groundwork for implementing and evaluating the security and performance characteristics of the system in subsequent chapters.

5.1 Radar Communication Use-Case

To balance the advancement in quantum computing, significant progress has been made in quantum-resistant technologies, which enabled the application of PQC and QKD. Securing communication against quantum-capable adversaries is especially important in military critical infrastructure, since these devices and services have strict requirements to ensure secure communication by staying up to date with latest protocols and technologies. Such applications are, e.g., radar systems, which are responsible for surveillance monitoring on land, in air, and on the sea. Acquisition of data transmitted by radar can lead to exploitable military advantage, therefore a fast deployment of quantum-resistant technologies is needed in this area, especially when considering "harvest now, decrypt later" attacks, where encrypted data is collected today, and planned to be decrypted when quantum computers become commercially available.

While exact radar specifications are not relevant in this thesis, it is assured, that any radar device has at least one component that is transmitting data outside the radar. A component like this, could be a tracking device, that is computing the information collected through sensors and signals, and forms the data into tracks (a data structure for a single detection describing many details about a single target). The tracks then are accessible through some kind of Ethernet interface. Given this, data transmission can be imagined through wired or wireless communication, i.e. for wired communication using the IEEE 802.3 (Ethernet) [37] protocol. This protocol does not provide any security, therefore authentication and encryption needs to be done in another Open Systems Interconnection (OSI) layer, with the help of TLS for example. To transfer track data, the All Purpose Structured Eurocontrol Surveillance Information Exchange (ASTERIX) data exchange protocol is known in the industry as a standard [38, 39]. However, ASTERIX assumes a secure communication channel, therefore no authentication or encryption is provided in this protocol, and because of this, security again relies on other communication protocols.

Given the sensitive nature of the data transmitted by radar stations, to ensure secure communication, quantum-resistant technologies need to be integrated both into radar systems and surrounding communication infrastructure, for the means of authentication, key exchange and data encryption:

- **Authentication**: In radar communication it is especially important to make sure the two parties communicating know each other and are convinced of the identity of the other party. A standard method is to use some kind of Message Authentication Code (MAC), e.g. the Hash-Based Message Authentication Code (HMAC). To achieve this, public-key cryptography with digital signatures is used in today's infrastructure, utilizing RSA or ECC. To integrate quantum-resistance into authentication, FIPS-204 or FIPS-205 can be used.
- **Key exchange**: To enable data encryption with symmetric cryptography, encryption keys need to be exchanged in a secure way. For this purpose, RADAR applications today use public-key cryptography (RSA or ECC). To elevate key exchange to a post-quantum level, FIPS-203 or some QKD protocol needs to be integrated.
- **Data encryption**: to encrypt data effectively in radar communication, symmetric cryptography algorithms are used today, such as AES. As stated in 2.2.2, these algorithms are less vulnerable to quantum adversaries compared to public key cryptography, therefore an elevated key size can be enough to counter quantum attacks.

To formalize the previous statements, a missile detection scenario of the Iron Dome system of Israel is considered, to estimate the data load needed to be handled. Although, there are no exact numbers published, from various sources it is estimated that as of today the Iron Dome itself still consist of only 10 batteries (an operational unit of a radar, launchers and missiles), which means 10 radars in use, each covering an area of 155 square kilometers [40, 41, 42]. It is also assumed, that the batteries transmit surveillance data

Data Item	Description	Size (bytes)
I062/010	Data Source Identifier	2 (0)
I062/040	Track Number	2 (0)
I062/070	Time Of Track Information	3 (0)
I062/080	Track Status	1 (5)
I062/100	Calculated Track Position (Cartesian)	6 (0)
I062/105	Calculated Track Position (WGS-84)	8 (0)
I062/130	Calculated Track Geometric Altitude	2 (0)
I062/135	Calculated Track Barometric Altitude	2 (0)
I062/136	Measured Flight Level	2 (0)
I062/185	Calculated Track Velocity (Cartesian)	4 (0)
I062/200	Mode of Movement	1 (0)
I062/210	Calculated Acceleration (Cartesian)	2 (0)
I062/220	Calculated Rate Of Climb/Descent	2 (0)
I062/270	Target Size & Orientation	1 (2)
I062/500	Estimated Accuracies	2 (17)
		40 (24)

Table 5.1: Relevant data items of the ASTERIX CAT62 v1.20 protocol. The minimum number of required bytes is presented, and optional bytes are given in parenthesis. The maximum considered when calculating data load, is the sum. Other data items of the ASTERIX description are considered irrelevant and therefore excluded.

to a centralized Command and Control (C2) center, which requires confidentiality and integrity. Based on the documentation ASTERIX [39], using the biggest possible size of all mandatory data items and data items in relation to non-piloted flying objects, a single track for air targets in military applications consist of at most 64 bytes (see table 5.1). The biggest reported attack from the recent years on the Iron Dome contained around 2000 missiles [41, 43]. Taking into consideration, that the Iron Dome is a critical system, estimations should be pessimistic, therefore a larger missile swarm, 3000 missiles are used for calculations. It is estimated, that the missiles distribute evenly, close to populated areas, and it is assumed that no two radars track the same missile, therefore a single radar tracks at most 300 missiles. Unintended targets (i.e., birds) can also appear on radar, therefore the amount of flying, tracked objects multiplies by 4. This leads to an estimated data load of 75 kB on each radar update. How often radars provide an update is not public information, but it is assumed to be 1 second, since missiles are fast flying objects and the range of radars are relatively small, providing a 60 rpm radar rotation for full surveillance. Combining all the previous details, a throughput of 75 kB/s is considered in the scenario, which requires authentication, key generation and encryption. Furthermore, it is also assumed, that the data travels in a classical channel, either in wireless or wired methods.

5.1.1 Visualizing Communication

After reviewing the use-case description in Section 5.1, communication between a radar and its C2 is visualized on Figure 5.1. The flow starts from the radar device generating track messages in the ASTERIX protocol and transmitting them to an application connected to the radar. This application then requests a QKD key from the corresponding QKD server and encrypts the data with it. The encrypted message is being signed by the application and being sent to the site of the C2. At this site, another application receives the data, verifies the signature and requests the same QKD key to decrypt the message. After this, the message can be process, for example being presented.

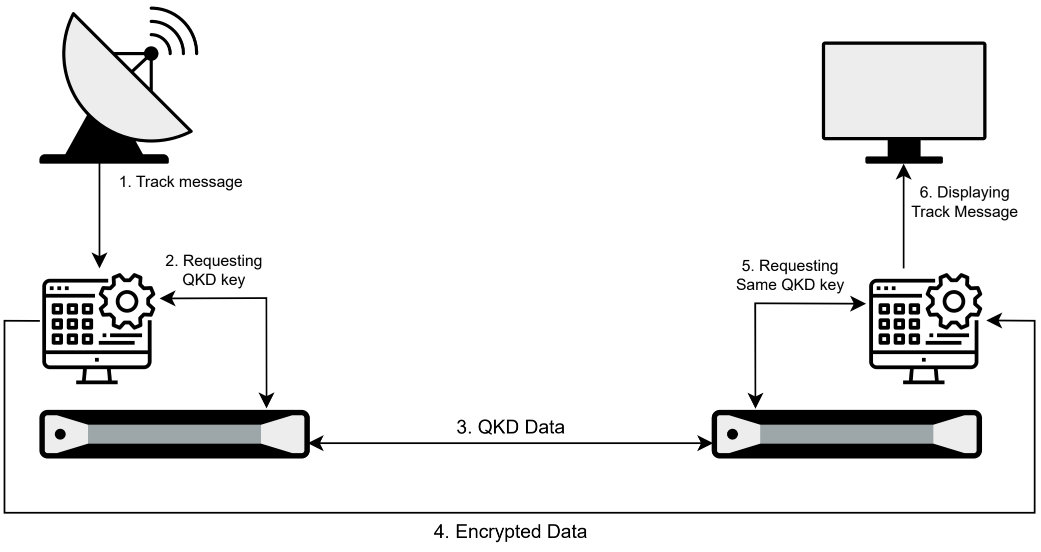


Figure 5.1: Visualization of the communication between a radar and its C2.

5.1.2 Attack Scenarios

Based on the proposed use-case in the previous section, passive eavesdropping, replay and Man-in-the-Middle (MitM) attacks are considered, each working with an adversary that has quantum computing capabilities.

Passive eavesdropping

During passive eavesdropping an attacker listens on a communication channel, which they should not observe, but only collects information undetected and not interfere with the communication itself. It is imaginable, that an attacker can listen to radar messages coming from the radar device and analyze the data, or they are capable of listening on the channel where key exchange happens.

- **Radar communication channel**: It is possible that all messages sent from a radar device are intercepted by an adversary, and their content is analyzed. These messages are considered secret in military applications, therefore it is required that their content is only visible to authorized parties. To counter this attack, using AES-256 is proposed, which provides sufficient security even against a quantum adversary.
- **Key exchange channel**: No matter what type of key exchange two parties using, it is possible that an adversary observes they key exchange and exfiltrates the key. Later, this key could be used to decrypt radar messages to gain access to their content. QKD is one of the current technologies that can be used to protect against leaking the exchanged key, by using the advantages quantum mechanics provide, therefore QKD is proposed to be used in the solution.

Replay Attacks

In replay attacks, an adversary somehow intercepts a valid message between to legitimate parties, and sends this message again to the recipient in a later point of time to achieve its goals. It is imaginable, that an adversary replays a radar communication message coming from the radar device to the C2 center and alerting armed forces or mislead the ongoing efforts of the center, in order to gain tactical advantage. A way to counter against these attacks is to keep track of a unique ID of each received message on the side of the recipient, but then liability lies in the implementation of the communication protocol. Liability can also be transferred to QKD, since the ETSI GS QKD 014 protocol describes, that each retrieved key by the recipient is removed from the KMS. It is proposed to generate new keys for each message sent in the communication, to prevent processing replayed messages.

MitM Attacks

In MitM attacks, an attacker impersonates both communicating parties to each other in order to modify messages and forward them finally to the recipient. An adversary with such capabilities can be assumed to act as a MitM and provide invalid information to the C2 center. To prevent this, it is proposed to use a PQC digital signature, since the signature can only originate from the owner of the private key it was made with, therefore it provides authentication, and makes sure the integrity of the message is not compromised. For digital signature purposes FIPS-204 (ML-DSA) with the highest security parameter set is proposed based on the advice of NIST.

Jamming, physical compromise and Denial of Service (DoS) attacks are not considered to be in the scope of the thesis, since they are specifically dependent on the specific capabilities of radar devices and their C2 center.

5.1.3 Requirements for the Radar Use-Case

Based on the use-case presented in Section 5.1, the attacks presented in Subsection 5.1.2 and on security standards and considerations, requirements were formalized towards the system and are presented in Table 5.2. System refers to the testbed and its components, together with the application layer communication.

ID	Type	Description	Source	Priority
RR1	Functional	The system shall be able to simulate messages coming from a radar device	S1	CH
RR2	Functional	The system shall be able to transmit the processed radar messages	S1	MH
RR3	Functional	The system shall be able to receive and process messages from a radar device	S1	WH
RR4	Functional	The system shall be able to simulate a C2 center receive the transmitted messages	S1	SH
RR5	Functional	The system shall be able to process the messages received from transmission	S1	MH
RR6	Functional	The system shall be able to simulate communication between the radar site and the C2 site	S1	MH
RR7	Functional	The system shall be able to use keys generated in a quantum-safe way	S3	MH
RR8	Functional	The system shall be able to symmetric encryption with the messages	S1	MH
RR9	Functional	The system shall be able to use quantum-safe authentication	S3	MH
RR10	Functional	The system shall use HTTP to transmit messages	S3	CH
RR11	Non-functional	The system shall simulate radar messages within less or equal than a second	S3	SH
RR12	Non-functional	The system shall end-to-end process messages in less or equal than a second	S1	MH
RR13	Non-functional	The system shall process all messages	S3	MH
RR14	Non-functional	The system shall transmit all messages and all messages shall arrive	S3	MH
RR15	Non-functional	The system shall maintain at least 99.9% uptime in key management and message processing	S3	MH
RR16	Security	The system shall use AES-256 for symmetric encryption	S2	MH
RR17	Security	The system shall use ML-DSA-87 for quantum-safe digital signing	S2	MH
RR18	Security	The system shall use QKD for quantum-safe key generation	S2	MH
RR19	Security	The system shall use an ETSI GS QKD 014 complient QKD device	S2	MH
RR20	Security	The system shall preserve confidentiality, integrity and authenticity of messages during transmit and at rest	S1	MH

Table 5.2: Table of requirements for the radar use-case. Prioritizing were created based on the MoSCoW method. Sources are the following: **S1** Terma, **S2** Attack Scenarios, **S3** Brainstorming.

5.1.4 Proposed Communication Flow

Based on the attacks described in Subsection 5.1.2 and the requirements presented in Table 5.2, the communication flow visible on Figure 5.2 is proposed to facilitate communication between a radar device and its C2 in a secure way. Client A represents the radar device, KMS A is the underlying QKD component connected to the radar and Client B is the C2 center and KMS B is the underlying QKD component of the center, directly connected to KMS A. The communication represents the use-case, where the radar device sends encrypted data to the C2 center without expecting response.

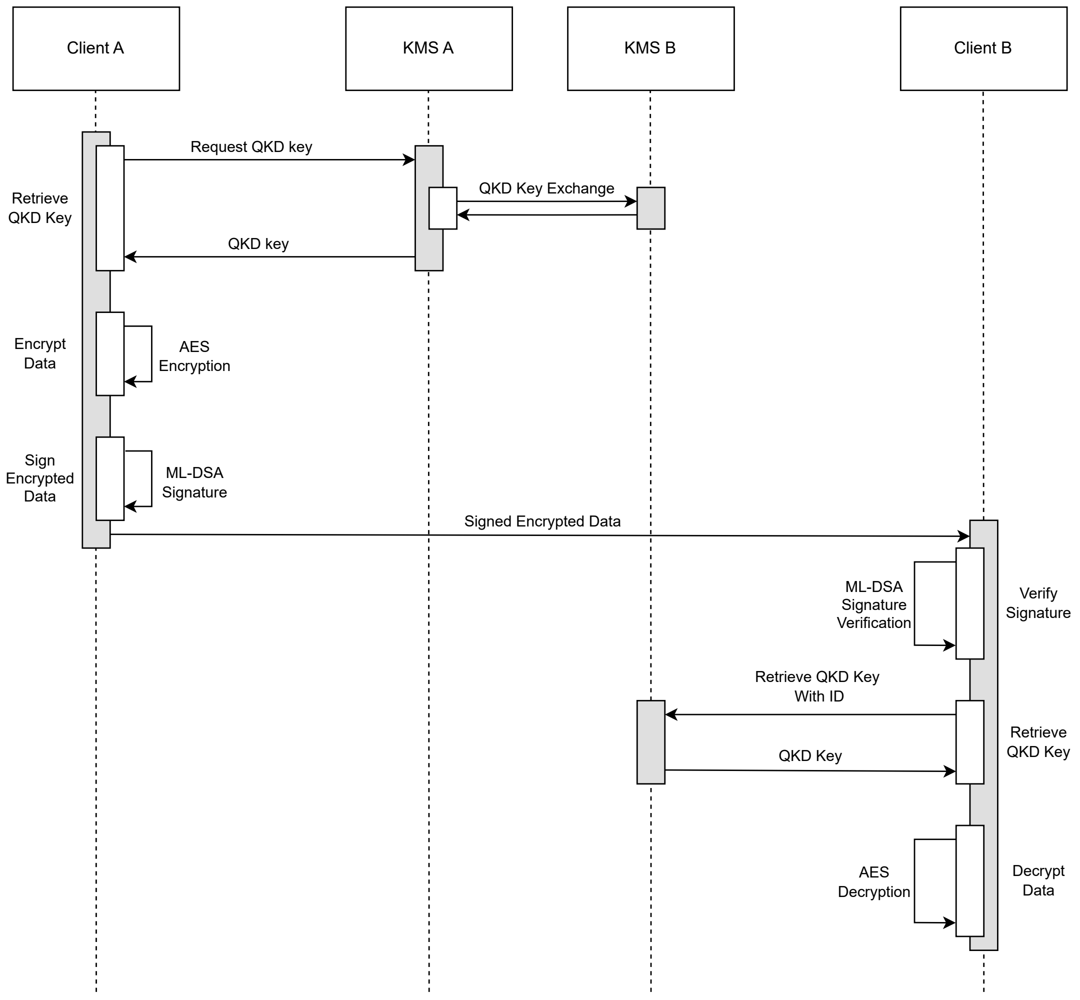


Figure 5.2: Communication flow on the application level.

The communication begins with Client A having some data to transfer, therefore it requests a 256 bit long key from KMS A. After this, QKD happens between the two sides, and the generated key appears in KMS B as well. Using the generated key, Client A AES-256 encrypts the 75 kB data payload. It is assumed, that ML-DSA-87 public-private key pairs has been generated before the communication begins for both Clinet A and Client B, and each client knows the public key of the other side, which was exchanged in a out-of-bound channel. When the data is encrypted, Client A appends the metadata of the encryption to the ciphertext and the QKD key ID, and signs this data with its own private key. The payload after this is transmitted to Client B through Hypertext Transfer Protocol (HTTP). Client B verifies the signature on the payload with the pre-shared public key of Client A. If the signature is valid, Client B proceeds to retrieve the QKD key from KMS B based on the ID in the payload. After the key is retrieved, Client B AES-256 decrypts the payload with helps of the key and the encryption metadata. At this point, the communication reached one full cycle, which is the base set of operations that needs to be measured in latency.

Communication happening on the quantum level is not in the scope of this thesis and is facilitated by the Decoy-State BB84 2.4.1 protocol.

5.2 Generalized Use-Case

Stepping away from a specific use-case, it is easily imaginable that quantum-secure communication needs to be established between two sites in different locations. In this case, each site might have one QKD server with a link between the sites, and applications on each site only have access to the QKD instance of their site. Applications then would need to request QKD keys from the QKD server on-site in different amounts and different loads.

To simulate such a request load from applications, based on brainstorming a generalized use-case is proposed, where 9000 QKD key requests are sent to one of the QKD servers. A larger communication is assumed, (e.g. a data center), therefore the amount of requests is considered to be a realistic number. To simulate key requests as random events from the perspective of the QKD box, each request is sent after waiting for seconds based on drawing a random number from an exponential distribution defined by a scale value. The exponential distribution is a good fit to model events that occur continuously and independently, such as packet arrivals in network traffic [44]. In a real world example it can happen easily, that one application requests multiple keys during its running, and different applications do this with different frequency. To model this, different exponential distribution scale values can be used. In order to simulate a quantum-resistant use-case, in theory the applications would use the keys for symmetric encryption with AES.

5.2.1 Requirements for the Generalized Use-Case

Based on the use-case described in Section 5.2 and the capabilities of the QKD devices, the requirements towards the system are presented in Table 5.3. System refers to the testbed and its components, together with the application layer communication.

ID	Type	Description	Source	Priority
RG1	Functional	The system shall be able to request QKD keys	S1	MH
RG2	Functional	The system shall be able to simulate a large parallel volume of key requests	S1	MH
RG3	Functional	The system shall be able to make HTTPS requests	S2	MH
RG4	Functional	The system shall simulate applications requesting keys	S1	MH
RG5	Functional	The system shall simulate applications with different key-requirements	S1	SH
RG6	Functional	The system shall use the retrieved keys for symmetric encryption	S1	WH
RG7	Functional	The system shall transmit encrypted messages	S1	WH
RG8	Non-functional	The system shall use the key request protocol provided by the QKD system	S2	MH
RG9	Non-functional	The system shall not have failed key requests	S1	SH
RG10	Non-functional	The system shall request keys in an asynchronous way	S1	SH
RG11	Non-functional	The system shall simulate request delay with exponential distribution	S1	CH
RG12	Constraint	The system shall not request more than 10 keys per second	S1	MH
RG13	Constraint	The system shall use the ETSI GS QKD 014 protocol to retrieve keys	S2	MH
RG14	Constraint	The system shall request keys through HTTPS	S2	MH

Table 5.3: Table of requirements for the generalized use-case. Prioritizing were created based on the MoSCoW method. Sources are the following: **S1** Use-Case, **S2** QKD Device.

Chapter 6

Implementation

6.1 QKD Testbed

Deciding together with AAU, the Sceptre LINK QKD devices (one transmitter, one receiver) of the company called HEQA Security were assigned for procurement (specifications are not public and will remain undisclosed in this thesis). HEQA provided the best delivery time and had the devices delivered in seven weeks. The QKD devices were designed to be used out-of-the-box and were assembled during a 2-day online training provided by HEQA, where we connected the machines and tested if they operate properly. On the training the vendor explained mechanisms of their QKD devices, warnings and pre-cautions (such as treating fiber links carefully, or being cautious during IP address changes). They also explained their implementation of the decoy-state BB84 protocol, which is the time bin and phase encoding. During the assembling, the devices were connected by the 2 meter direct fiber cables provided by the vendor. Unfortunately, the ordered 20 kilometer fiber cables did not arrive on time, therefore during the thesis the 2 meter fiber cable were used in the testbed. Manual attenuators of 5 and 10 dB were also purchased during this time to simulate longer distances. After the training, we assembled the boxes in a laboratory of the university with the following system architecture seen on Figure 6.1. Each QKD server has three bidirectional Ethernet interfaces: one for service, one for accessing the integrated KMS and one for connecting the devices to network management. The service channel connects the QKD transmitter with the receiver and transmits service data, key reconciliation data and KMS data (which does not include the keys). The devices also provide a web GUI, which is also accessible through this channel. The KMS channel connects the integrated KMS of a QKD device to the key-requester. In our testbed, we did not use the management interface of the devices, since in the scope of the thesis they were not connected to any network. The Ethernet switches were added to the setup for practical reasons, they could be removed if the service channel is connected directly to the QKD devices and KMS channels are connected directly to the key-requesters.

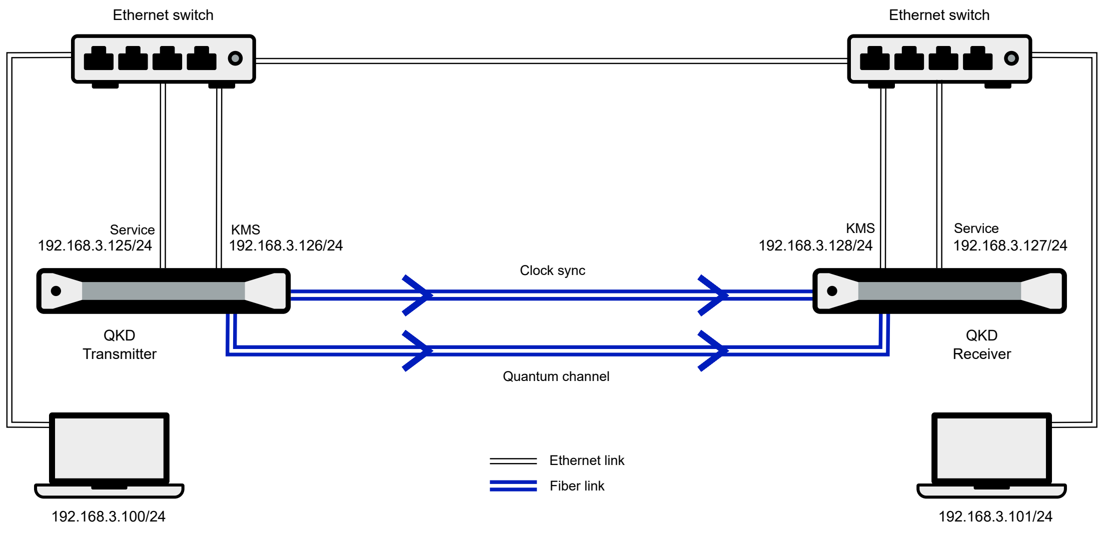


Figure 6.1: Architecture of the testbed.

In order to operate, the QKD devices need two unidirectional fiber links: one for clock synchronization and one for facilitating the key exchange protocol.

The connected PC devices specification are out of the scope of this thesis, since any device can request keys which has the capability to send HTTPS requests, because the devices are also compatible with the ETSI GS QKD 014 REST API[1].

On Figure 6.2 the temporary laboratory setup can be seen at AAU without using a rack at the time of this thesis was written.

[1]`https://www.etsi.org/deliver/etsi_gs/QKD/001_099/014/01.01.01_60/gs_qkd014v010101p.pdf`

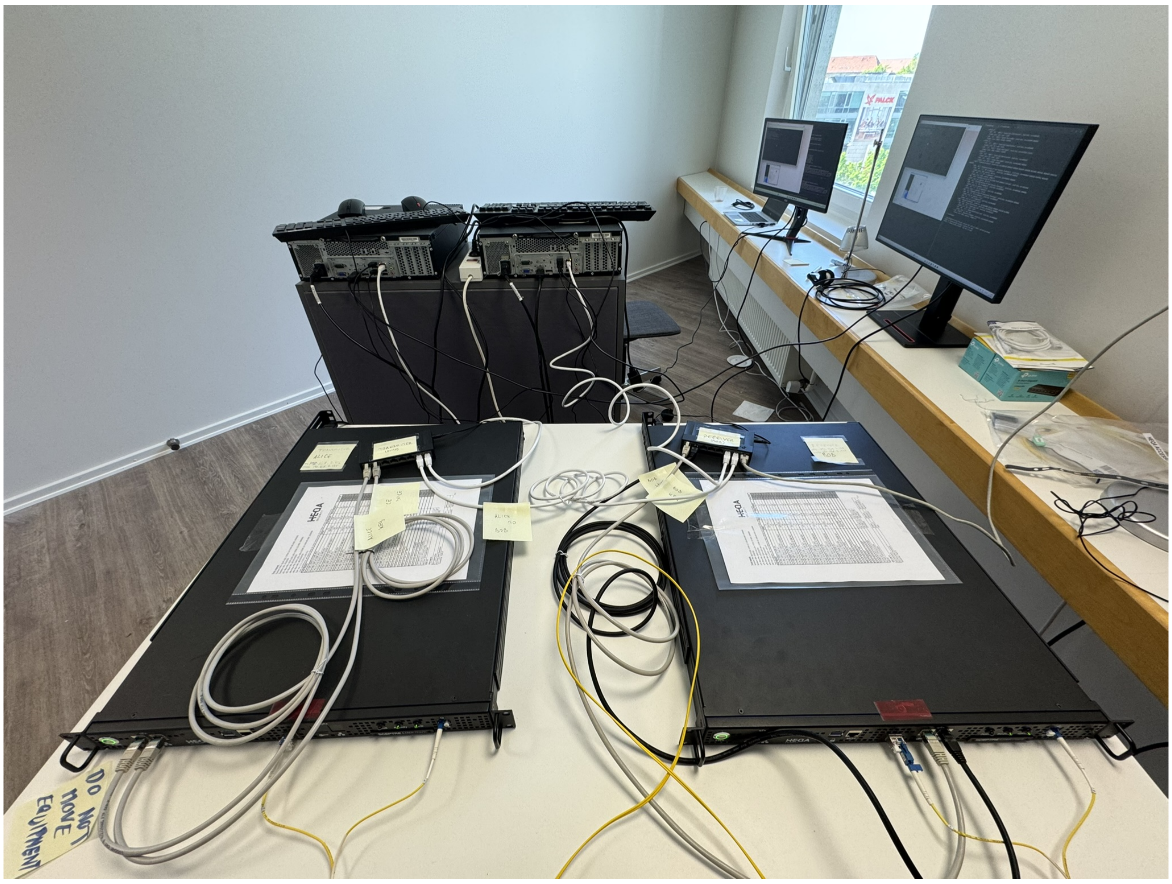


Figure 6.2: Temporary physical testbed setup at AAU.

6.2 Radar Use-Case Application-Layer Communication

To store all related scripts and files a private GitHub repository was created with the following folder structure:

```
/
├── data/
│   └── data.bin
├── keys/
├── out/
├── src_radar/
│   ├── clientA.py
│   └── clientB.py
├── src_generalized/
│   └── app.py
└── util/
```

```
│   └── gen_keys.py
└── requirements.txt
```

The `data.bin` file contains 75 kB of random bytes, which represents the data load and was generated with the `head -c 76800 /dev/urandom > data.bin` bash command. The key pairs for each node should be placed into the `keys/` folder in a JSON format. The scripts generate the measured output into the `out/` folder. To facilitate communication between Client A and Client B, two python scripts were created in the `src/` folder, `clientA.py` and `clientB.py` respectively, using Python 3.13.2. Test scripts are found in the `test/` folder, both for Client A and Client B. To generate public-private key pairs, the `gen_keys.py` Python script was used, which is using the `liboqs-python` library according to its documentation to generate two key pairs, and outputs them base64 encoded into a JSON file. The required packages and their version are listed in the `requirements.txt` file and are the following:

```
certifi==2025.1.31
charset-normalizer==3.4.1
Flask==3.1.0
idna==3.10
pycryptodome==3.22.0
requests==2.32.3
urllib3==2.4.0
```

6.2.1 Client A

As the first step, the data bytes are read from the `data.bin` file, and stored in a variable. After this comes a QKD key generation, AES-256 encryption, ML-DSA-87 signature generation and transferring the data happens. To get more precise results, this process is repeated 10 000 times in a loop. Inside the loop, a new thread is started with each test round, which after a 500 ms pause is implemented to simulate 2 messages per second, which is more than what a radar based on the use-case description in Section 5.1 could do. Elapsed time of each separated action is measured, by the `time.perf_counter_ns()` standard Python function. Initiating QKD key generation happens by using the ETSI GS QKD 014 REST API provided by KMS A, so technically an HTTPS POST message is sent with the `requests` Python package using the SSL certificate of the KMS, and the returned JSON object contains the key and its ID. It is possible to request a specific-size key from the KMS, therefore the key length is set to 256 bit in the payload of the HTTP requests, according to section 5.3 in the official documentation[2]:

```
payload = {
    "number": KEY_AMOUNT,
    "size": KEY_LENGTH,
```

[2]`https://www.etsi.org/deliver/etsi_gs/QKD/001_099/014/01.01.01_60/gs_qkd014v010101p.pdf`

```
    }
    response = requests.post(
        url = get_key_url,
        json = payload,
        verify = 'ca-cert.crt'
    )
    response.raise_for_status()
    key_info = response.json()

    key_ID = key_info['Keys'][0]['key_ID']
    key = base64.b64decode(key_info['Keys'][0]['key'])
    return (key_ID, key)
```

After the key is retrieved, the PyCryptodome Python library is used to AES-256-GCM encrypt the data loaded from data.bin following the documentation of PyCryptodome[3]. The following JSON payload is created to be signed:

```
{
    "nonce": <base64 encoded nonce>,
    "ciphertext":<base64 encoded ciphertext> ,
    "tag": <base64 encoded tag>,
    "key_ID": <UUID of generated key returned by the REST API>,
}
```

ML-DSA-87 signature generation happens with the python-liboqs library, based on the provided example[4] in the GitHub repository of the library, where *SIGALG* is *"ML-DSA-87"*:

```
def sign_data(encrypted_data: object, key_id: str, private_key: bytes) -> object:
    signer = oqs.Signature(SIGALG, private_key)

    encrypted_data["key_ID"] = key_id
    signature = signer.sign(json.dumps(encrypted_data, sort_keys=True).encode())
    encrypted_data["signature"] = base64.b64encode(signature).decode()

    return encrypted_data
```

After this, the signature is base64 encoded and appended to the payload object presenting the final payload being sent over to Client B:

```
{
```

[3]https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html#aes

[4]https://github.com/open-quantum-safe/liboqs-python/blob/328ca5322943df037cad4d49f3db1076e7f80b92/examples/sig.py#L37

```
    ...<PREVIOUS DATA>,
    "signature": <base64 encoded signature>
}
```

Sending the payload to Client B happens by sending a simple HTTP POST request. At the end of a loop iteration, measured time in nanoseconds is added to the collection of measurements. This also means that the transmission time from Client A to Client B is measured in the HTTP return time of sending the payload to Cleint B. When all the iterations are done, the script outputs the measurements into the `out/output_clientA.txt` file.

6.2.2 Client B

When being run, the script starts a Flask HTTP server instance and listens for incoming POST requests only. After decoding the received JSON payload, a `threading.Thread` is started to handle the request. In each request handling, ML-DSA-87 signature verification, QKD key retrieval and AES-256 decryption happens. During verification, the payload is decomposed into the signature itself, and the remaining fields in the Python object, resulting in the same object as in 6.2.1. The signature is decoded from base64 encoding. The signature verification happens according to the example in the `python-liboqs` GitHub repository[5]:

```
message = json.dumps(signed_data, sort_keys=True).encode()
with oqs.Signature(SIGALG) as verifier:
    return verifier.verify(message, signature, public_key)
```

In case the signature is invalid, processing of the request stops. If the signature was verified, the script retrieves the generated QKD key from KMS B with the ID retrieved from the payload received, according to section 5.4 in the ETSI GS QKD 014 documentation[6]:

```
payload = {
    "key_IDs": [
        {
            "key_ID": key_id,
        }
    ],
}
response = requests.post(
    url = get_key_with_IDs_url,
    json = payload,
```

[5]https://github.com/open-quantum-safe/liboqs-python/blob/328ca5322943df037cad4d49f3db1076e7f80b92/examples/sig.py#L40

[6]https://www.etsi.org/deliver/etsi_gs/QKD/001_099/014/01.01.01_60/gs_qkd014v010101p.pdf

```
        verify = 'ca-cert.crt'
    )
    response.raise_for_status()
    key_info = response.json()
    return base64.b64decode(key_info['Keys'][0]['key'])
```

After the QKD key has been retrieved, the process continues by decrypting the ciphertext according to the official documentation of PyCryptodome[7]. The application measures time similarly to Client A, after processing a request time in nanoseconds is added to the collection of measurements. When all the requests were received, upon server shutdown the script outputs the measurements into the `out/output_clientB.txt` file.

6.3 Generalized Use-Case Application

In the generalized use-case, the system architecture stayed the same as described on Figure 6.1. To get good measurement results, 9000 requests are sent to one of the QKD boxes. The application was implemented using the `asyncio` and `aiohttp` Python libraries, to introduce parallelism into the system. To prevent more than 10 requests per second, a token queue architecture is used, where each request can only happen, if the queue contains at least one token. After starting the application, an `asyncio.Queue` object is being created to serve as the token queue, and a background task is started to add a token to the queue every 100 ms.

```
async def token_refiller():
    while True:
        try:
            rate_limiter_queue.put_nowait(1)
        except asyncio.QueueFull:
            pass  # ignore overflow
        await asyncio.sleep(1 / MAX_REQUESTS_PER_SECOND)
```

After this task has started, the 9000 tasks are created and parameterized.

```
    for i in range(TOTAL_REQUESTS):
        group_id = i // REQUESTS_PER_GROUP
        scale = SCALES[group_id]
        delay = np.random.default_rng().exponential(scale=scale)
        tasks.append(asyncio.create_task(
            make_request(group_id, delay, i, ssl_context)))
```

To simulate different application key-needs, scale 0.5, 1.0 and 5.0, distributed equally between the requests, were used with exponential distribution as the random delay before

[7]https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html#aes

a request can be executed. When a request task is executed, it requests a token from the queue, and then proceeds. In case there is no token, the request handling waits until there is one available. To simulate a better real-life scenario, each request builds its own session.

```
duration = -1
try:
    async with aiohttp.ClientSession() as session:
        start = time.perf_counter_ns()
        async with session.post(URL, json=payload, ssl=ssl_context) as response:
            end = time.perf_counter_ns()
            duration = (end - start) / NANO_TO_MILLI
            status = response.status
except Exception as e:
    status = f"Error: {e}"
```

Each request latency is measured in milliseconds with the standard `time.perf_counter_ns()` Python function, and in case an error happened during the request, the duration is set to -1. After the applications is done with all the requests, it outputs the Group ID, Request ID, delay, duration and HTTP status into the `request_logs.csv` file. Group ID refers to the scale the request has used to draw a random number to delay itself, where Group ID 1 means scale 0.5, Group ID 2 means scale 1.0 and Group ID 3 means scale 5.0.

Chapter 7

Evaluation

7.1 Radar Use-Case Evaluation

Latency was measured on the application level with a computer with an Apple M1 chip and 16 GB of RAM, and the results were plotted manually based on 10 000 full communication flow test rounds. Latency results from each communication step are visible on Figure 7.1.

It is clearly visible based on Figure 7.1, that the symmetric encryption and digital signature generation or verification is significantly faster compared to QKD key retrieval or transmitting the data over HTTP. It is concerning, that the 1000 ms threshold given by requirement RR12 in Table 5.2 is violated multiple times already by the QKD key retrieval on Site A. Another anomaly worth to note is, that all the steps that involve some kind of HTTP communication have considerably more outliers in Q4, this communication protocol might be a bottleneck in the proposed system. A tendency in key retrieval time in comparison between Site A and Site B is also visible, namely retrieval on Site B is faster and has less extreme outliers, for which might be the reason that less communication between the QKD devices is involved.

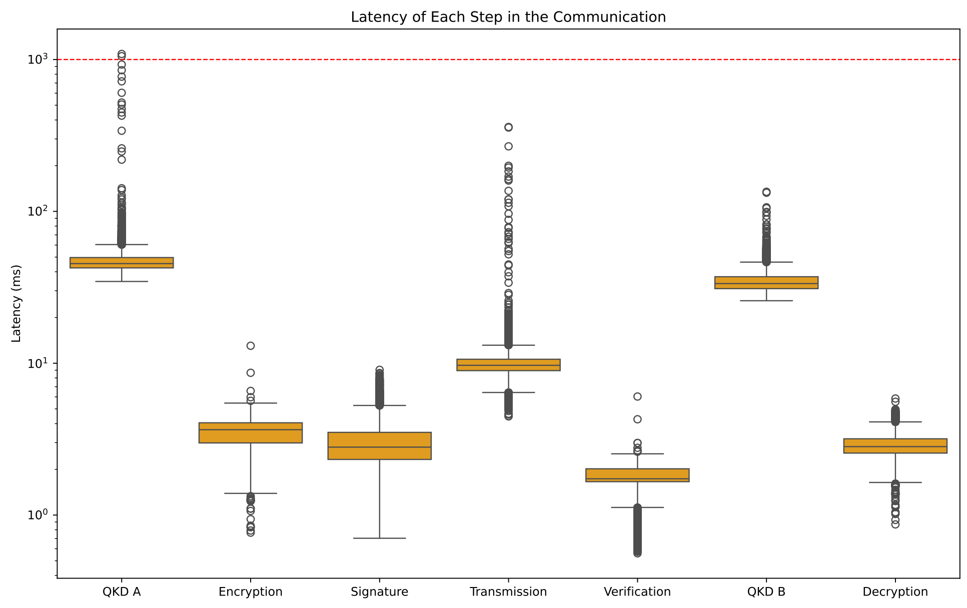


Figure 7.1: Box plot of latency projected to each step in the communication over 10 000 test rounds (see full-resolution version as Figure A.2 in Appendix A). 1000 ms threshold marked with red.

Looking at the box plot of total elapsed time during each test round on Figure 7.2, it can be seen that many of the test rounds did not complete under the 1000 ms threshold posed by requirement RR12. Additionally, these results are not considering the QKD key generation as a latency factor, since the procured QKD boxes constantly generate new keys after being switched on. Using other type of devices would possibly increase the overall latency of the system depending on the distance between the QKD nodes.

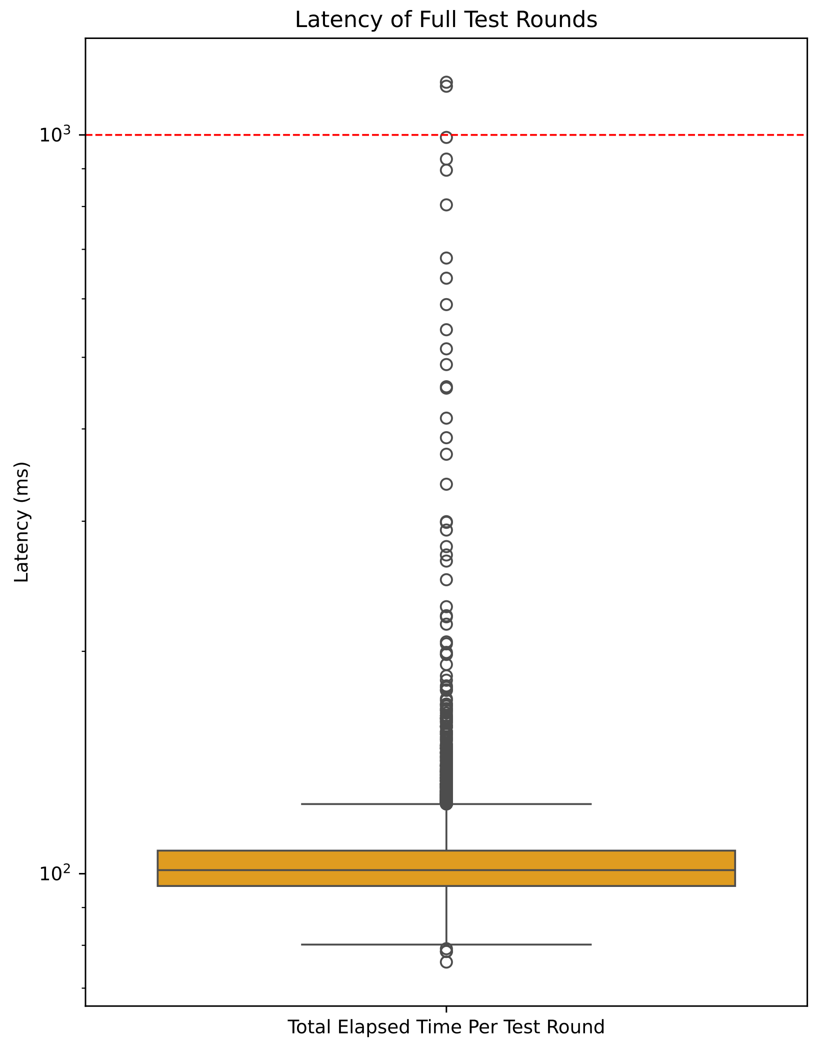
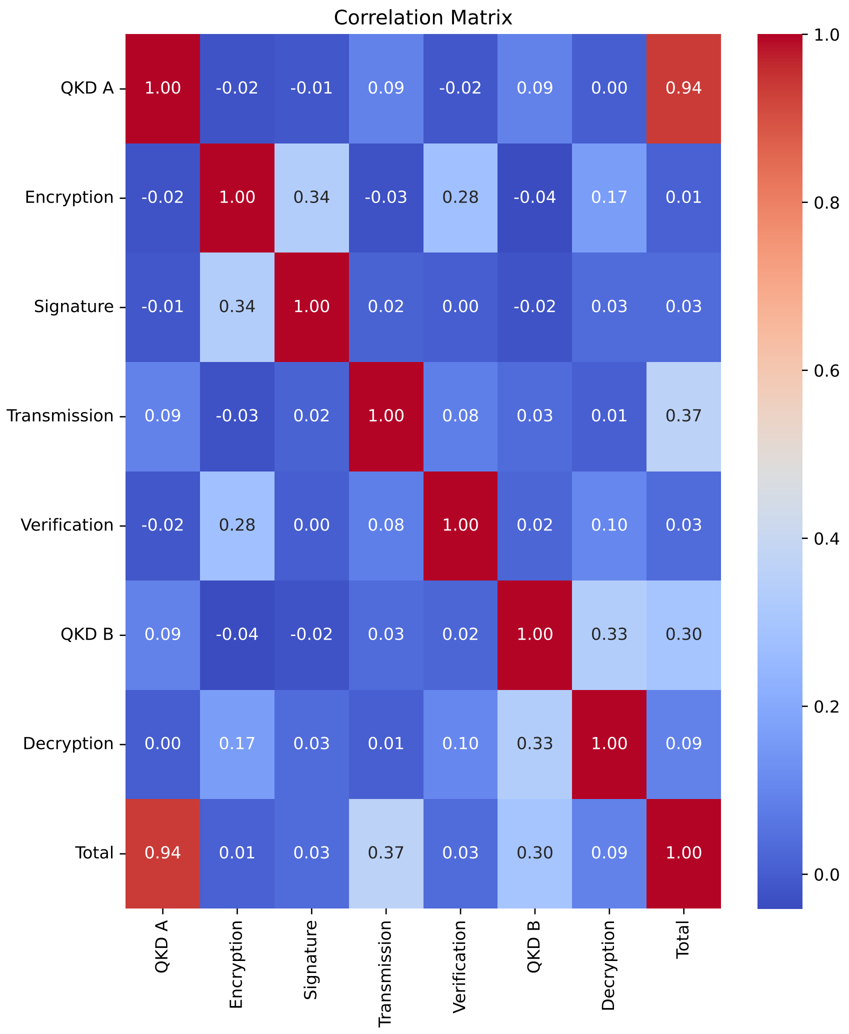


Figure 7.2: Box plot of total latency measured during 10 000 test rounds.

Standalone QKD key generation rate read from the GUI can be seen in Table 7.1 measured by manual attenuation on the QKD devices.

	0 dB	5 dB	10 dB	15 dB	20 dB
Secure Bit Rate Per Second	21000	18000	15000	7000	3000
Keys Generated Per Second (256 bit)	82	70	58	27	11

Table 7.1: Approximate secure bit rate and key generation speed per second with different manual attenuation, based on the information provided by the GUI of the QKD servers.



	QKD A	Encryption	Signature	Transmission	Verification	QKD B	Decryption	Total
QKD A	1.00	-0.02	-0.01	0.09	-0.02	0.09	0.00	0.94
Encryption	-0.02	1.00	0.34	-0.03	0.28	-0.04	0.17	0.01
Signature	-0.01	0.34	1.00	0.02	0.00	-0.02	0.03	0.03
Transmission	0.09	-0.03	0.02	1.00	0.08	0.03	0.01	0.37
Verification	-0.02	0.28	0.00	0.08	1.00	0.02	0.10	0.03
QKD B	0.09	-0.04	-0.02	0.03	0.02	1.00	0.33	0.30
Decryption	0.00	0.17	0.03	0.01	0.10	0.33	1.00	0.09
Total	0.94	0.01	0.03	0.37	0.03	0.30	0.09	1.00

Figure 7.3: Correlation matrix between the communication flow steps.

Figure 7.3 showcases a correlation matrix between the communication flow steps. The matrix shows, that there is a visible correlation between the QKD key retrieval on Site A and the total elapsed latency. However, there is no remarkable correlation between QKD key retrievals on Site A and Site B, or between any other communication steps.

Requirement RR1 were not implemented in the system, since the source of the messages was considered not important in the scope of the thesis. For similar reasons, requirement RR3 was also dismissed and a dummy data message was used instead. Requirement RR4 was not implemented, since no information of a C2 center or its behavior was available. It is to be noted, that requirement RR15 and RR19 were guaranteed by the QKD vendor.

To give an evaluation summary, it can be stated that the current system is not fulfilling the must have requirements of the radar use-case, reasoned by the violation of requirement RR12, and future research is needed to stabilize the system and reduce latency.

7.2 General Use-Case Evaluation

Latency was measured on the application level with a computer with an Apple M1 chip set and 16 GB of RAM, and the results were plotted manually based on 9000 requests. Results of each test round can be seen on Figure 7.4.

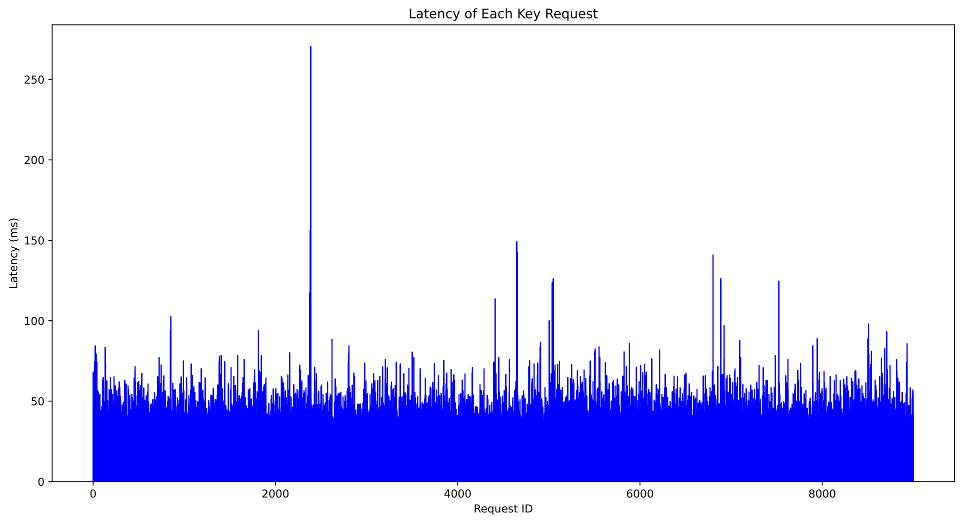


Figure 7.4: Latency of each request in the test (see full-resolution version as Figure A.1 in Appendix A).

It is visible that although most of the requests were around 50 ms, there are many clear outliers, with latency up to almost 300 ms. It is also notable, that the outliers were not coming in bursts, but they appear to be isolated events. To showcase outliers and general summary statistic, see Figure 7.5.

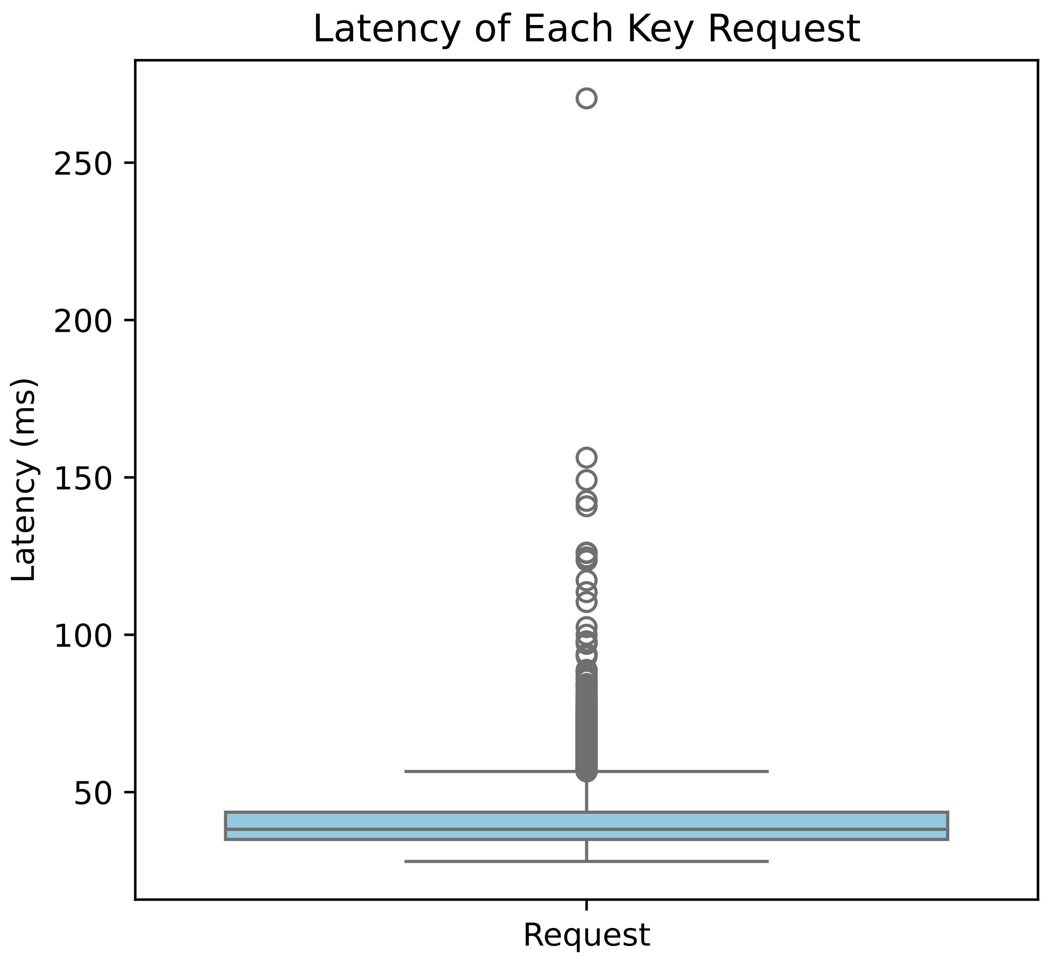


Figure 7.5: Box plot of latency measured in 9000 QKD key requests.

7.5 shows that the median latency is around 37 ms, indicating that half of the key requests complete at or below this value. The Interquartile Range (IQR) spans from approximately 35 ms to 42 ms, suggesting that the middle 50% of latencies are fairly consistent and tightly clustered. The whiskers extend from roughly 25 ms to 60 ms, capturing the tighter range of typical request durations. However, there are clear outliers visible here as well. The distribution appears relatively symmetric with a bit of positive skew, indicating a roughly stable system with no significant skew in latency performance besides outliers.

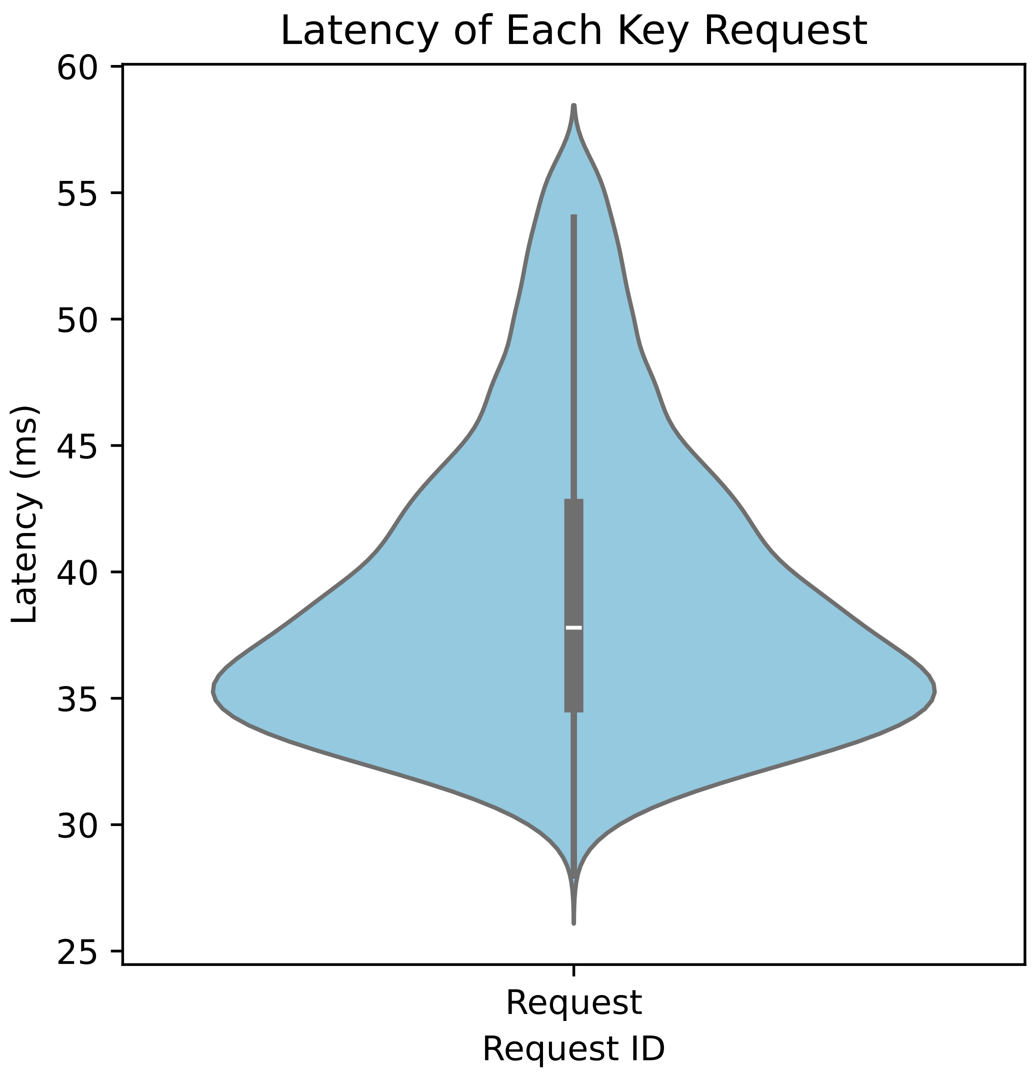


Figure 7.6: Violin plot of latency measured in 9000 QKD requests, where outliers were removed at both ends of the results based on distance compared to 1.5 * IQR.

Figure 7.6 shows a violin plot of the distribution of the latency of requests. As described earlier, the data is centered around 36-37 ms with slight density bulges near 42 ms and 46 ms, producing a asymmetric shape with clustering on the lower end. The box inside the violin shows that the median slightly deviates from the peak density, which suggests variation in performance. Outliers although must be considered.

Requirement RG6 and RG7 were not implemented, since they were tested in the radar use-case already, and they would not give value to the latency measurement concluded here. The rest of the requirements found in Table 5.3 are satisfied. It needs to be noted, that there were no requirement for the latency, since the goal of the test was to see how latency changes if the system is under stress from multiple requests, however clear outliers are visible, which indicates that further investigation might be needed. Before the final measurement, previous measurements failed due to exceeding the key request limit guaranteed by the manufacturer, in some cases even resulting in the restart of the QKD devices. This phenomenon is discussed deeper in Subsection 8.2.4.

To give a summary to the evaluation of the measurements in the generalized use-case, it can be said that the system over all provides a fast and roughly stable QKD retrieval latency when the supported key request threshold is not violated, but there are clear and visible outliers.

Chapter 8

Discussion

This chapter provides an in-depth discussion of the results presented in Chapter 7. The aim is to interpret the key findings in relation to the original research questions proposed in Chapter 1 and the broader context of system performance and latency analysis. Particular attention is given to the observed latency trends, variability, and system behavior under rate-limited and delay-injected workloads. The implications of these results are considered both from a practical implementation standpoint and within the scope of existing literature. Furthermore, potential limitations of the measurement approaches and implementation are discussed, along with considerations for the future work after this thesis. Where applicable, comparisons are drawn to expected performance baselines or known architectural behaviors, providing a critical lens through which to evaluate the system's response characteristics.

8.1 Main Findings and Interpretations

8.1.1 Implementing the Testbed

By proposing the research question about the challenges of creating a QKD testbed to facilitate key exchange, the goal was to gain experience and knowledge in the operation of these devices and their surrounding network elements. Setting up the devices was easier than expected, especially with the training provided by the vendor. The two servers can be used out-of-the-box, starting them takes roughly 15 minutes and from the point the go into active state, they generate data blocks constantly, so available keys are quickly provided. Knowledge gained by reading the state of the art, reinforced with details provided by the vendor, was more than enough to understand the high-level mechanics of the devices and the way they operate. Apart from some non-trivial limitations, all details were straightforward to understand. It was interesting to gain knowledge about a different implementation of decoy states in the BB84 protocol, which is called Time Base and Phase encoding by the vendor. This protocol gives the same results as decoy-state BB84 would give by transmitting the photon source intensity next to the rotation of the photon emitter and detector, but using a different approach, which is out of the scope of this thesis. Other experiences are stated in Section 8.2. The devices should be placed in a sound-isolated locations, since they are very loud during operation due to fan noise.

8.1.2 Integrating PQC and QKD

There are different ways to integrate PQC and QKD technologies, such as using QKD as the seed for PQC encryption [26]; using PQC for wireless while QKD for wire-enabled channels [27]; or creating a single key from QKD, PQC and classical keys [28]. Integration is also considered towards authentication, such as using PQC to authenticate messaging parties and then using QKD to encrypt communication[30]; or using PQC certifications as chain of trust, and using QKD keys for encrypting communication [32, 33]. Other integration efforts were more focused on different quantum-resistant technologies in different communication layers, i.e. using PQC for endpoint security and QKD for securing communication nodes [34]; or using PQC for communication inside DER networks, while using QKD between DER networks.

Compared to previous work done on integrating these specific technologies, the proposed communication contributes to the authenticated message exchange in the application layer, where message authenticity and integrity is provided by pre-shared PQC keys and digital signatures, and confidentiality is ensured by symmetric encryption established by QKD keys. This setup was considered to be the most easily implementable, yet effective, which secures message exchange against the attack scenarios introduced in Subsection 5.1.2.

To answer the research question on how PQC and QKD can be integrated to provide authenticated message exchange, it can be said that the proposed communication flow shown on Figure 5.2 is suitable for this purpose with pre-shared public key pairs, even that security could be increased during QKD key retrieval.

8.1.3 Evaluating the Setup

Radar Use-Case

To re-estate the findings of Section 7.1, it was found that the proposed communication flow and the implemented testbed do not fulfill all the requirements posed in Table 5.2. Interpreting the results on Figure 7.1 shows that although QKD key retrievals perform with stable latency, many outliers can be found in the Q4 region of the measured results. The reason for this is unknown, yet some assumptions are posed in Section 8.2. It was expected to see a very smooth convergence in the latency, since the limitations of the devices were taken into consideration during the final measurements. Having some of the QKD requests lasting longer than 1000 ms is worrying, since critical infrastructures could have very strict requirements on stability and availability.

Seeing the results for each full test round on Figure 7.2 and interpreting it together with the correlation matrix seen on Figure 7.3, it can be said that requesting QKD keys on the transmitter site has the biggest affect on the total latency. A reason for this could be, that at requesting a QKD key, the device the key was requested from needs to identify the bits of the key in the secure data blocks shared between the two devices, and needs to communicate this with the receiver QKD device. Compared to this, when a key is being retrieved on the other side, the QKD device on that side already knows where to look for the key.

In comparison with QKD, PQC integration seems to be an easier challenge, due to well-tested and widely-used standard libraries already existing, in which performance does not depend on a specific vendor. It is interesting to see that PQC signature generation has more outliers in Q4, while signature verification has its most outliers in Q1, which means based on the measured latency, that signature verification is generally a faster process in ML-DSA.

There is also a small ~0.3, but strange correlation between the latency of encryption and signature generation. This is unexpected, since every test round was executed with the same data, which resulted in the same size of encrypted data and same size of signature. A valid reason could be, that the measuring PC was busy at the time these operations happened, and therefore both of them took longer in this period.

Generalized Use-Case

Summarizing the results of the generalized use-case it can be stated that although the system fulfills the requirements posed in Table 5.3, the performance of the QKD servers is not constant over time and clear outliers can be identified. Figure 7.4 shows that there are outliers with latency around 150 ms, with the highest outlier being over 250 ms in latency. No clear reason could be identified in the scope of the thesis for having such high deviations, however it is suspected that the reason might lie in he application-level implementation of the tests, one of the network components or the QKD devices itself. Looking at Figure 7.5, without outliers the QKD servers provide a stable key request time, supplying keys in roughly 40 ms, which can be considered good performance, if creating a new session with each request and having 9000 requests are taken into consideration. Looking at Figure 7.6 it is also visible, that even with removing outliers from both Q1 and Q4, there is significant density of requests over the median, which can indicate disparity in performance of the QKD devices.

Requirements RG6 and RG7 of Table 5.3 were not implemented, since they additional value to the measurement was considered insignificant. Also, encryption capabilities were tested in the radar use-case already. It is to be noted on the other hand, that all the 9000 requests were completed successfully with maximum key request amount per second.

To answer the research question on how suitable the developed testbed for assessing secure message exchange under a radar communication and a generalized scenario, it is visible that future research is needed, due to the unstable latency results in both cases.

8.2 Limitations

8.2.1 Distance

The original plan for this thesis was to facilitate QKD on a 20 km long fiber cable where additional manual attenuation can be added to simulate even longer distances, and to measure the latency of the radar use-case communication in different setups. Unfortunately, the long cable did not arrive on schedule, therefore secure bit rate measurement happened using manual attenuators only, where 5 dB of attenuation was considered to be roughly equivalent to 25 km distance on fiber.

8.2.2 Constant QKD Key Generation

The purchased QKD devices constantly generate secure data blocks from the point they are switched on and in active state. This means, the retrieval time of a key is independent from the distance the QKD devices are set up on, and solely relies on the network setup between the key-requester and the QKD device. Depending on the distance, after 1-2 hours being powered on, there are more than 3 million keys ready to be used. Even with 20 dB manual attenuation, 11 keys (256 bit) were generated per second, which could be more than enough for most of the use cases.

8.2.3 PQC Signature Size Matters

In the early stages for planning the showcase, for the use-case it was imagined that messages are transmitted between Site A and Site B via UDP and are also PQC digital signed. The two sites were both simulated on the same macOS computer, and in some cases connection error was noticed. It turned out, that on macOS the default UDP packet size is 9216 bytes, and that an ASTERIX message generator tool[1] can easily generate message around a couple thousand bytes. When the ML-DSA-87 digital signature of 4627 bytes size was added to the payload, it became easily larger than 9216 bytes. This indicates that switching current digital signature schemes with PQC is not a trivial task, because the key size can be even ten times larger than before, and network limitations may apply.

8.2.4 QKD Key Retrieval Limitation

While testing the generalized use-case for the first time with 9000 requests, it was experienced that the KMS of the addressed QKD device stopped working and returned either HTTP error 500 or connection error. After discussing the issue with the vendor, it became clear that the purchased QKD devices only support ~10 key requests per second. Becoming aware of this limitation of the devices, both the radar and the generalized use-case

[1]`https://github.com/zoranbosnjak/asterix-tool`

were re-implemented with restrictions on the amount of requests sent per second and re-measured. In the radar use-case a 500 ms pause between the processing of messages were implemented, while in the generalized scenario a token queue was introduced to enforce maximum 10 key requests per second. Unfortunately, at this point of measurement there was only one PC available with an Apple M1 chip to conduct measurements on.

This limitation might have been the cause for extreme outliers measured at the first testing in the radar use-case, such as ~61000 ms for a key request. The KMS of the QKD device became operational again after restarting the device.

8.2.5 Equipment

The equipment seen on Figure 6.2 and Figure 8.1 can be far away from a real-life hardware setup in critical infrastructure or a data center. In the testbed the brand and age of patch cables were different, and Tp-Link Litewave LS1008g switches were used, these might have also affected the measurement results and could have been a bottleneck. As it is also mentioned in Subsection 8.2.4, for the measurements presented in this thesis only one PC with Apple M1 chip set was used, this is also far from realistic setups. This machine only has USB-C ports, therefore a BlitzWolf BW-TH11 11 in 1 USB-C hub were used to attach the patch cable connected to the testbed setup, which might have been a physical bottleneck as well.

8.3 Showcasing QKD Integration at Terma

Terma is a partner in the project in which terms the QKD devices were purchased, therefore they requested to showcase the devices at their headquarters in Søborg. The devices were prepared to showcase the message flow seen on Figure 5.1. The setup did not involve PQC digital signatures, but radar track messages were generated with the ASTERIX generator tool and received via UDP at the application level. On Figure 8.1 the setup at Terma can be seen, where the PC and QKD on the left represents the radar generating messages and the secure node connected to the radar, and the right side represents the secure receiver node and a decoder for the radar track messages. In the terminal on the left monitor the SHA-256 hashes of the received and encrypted radar messages can be seen. On the left terminal on the left monitor the SHA-256 hashes of the received and decrypted messages are visible, and on the right terminal the decoded ASTERIX CAT62 message is presented.

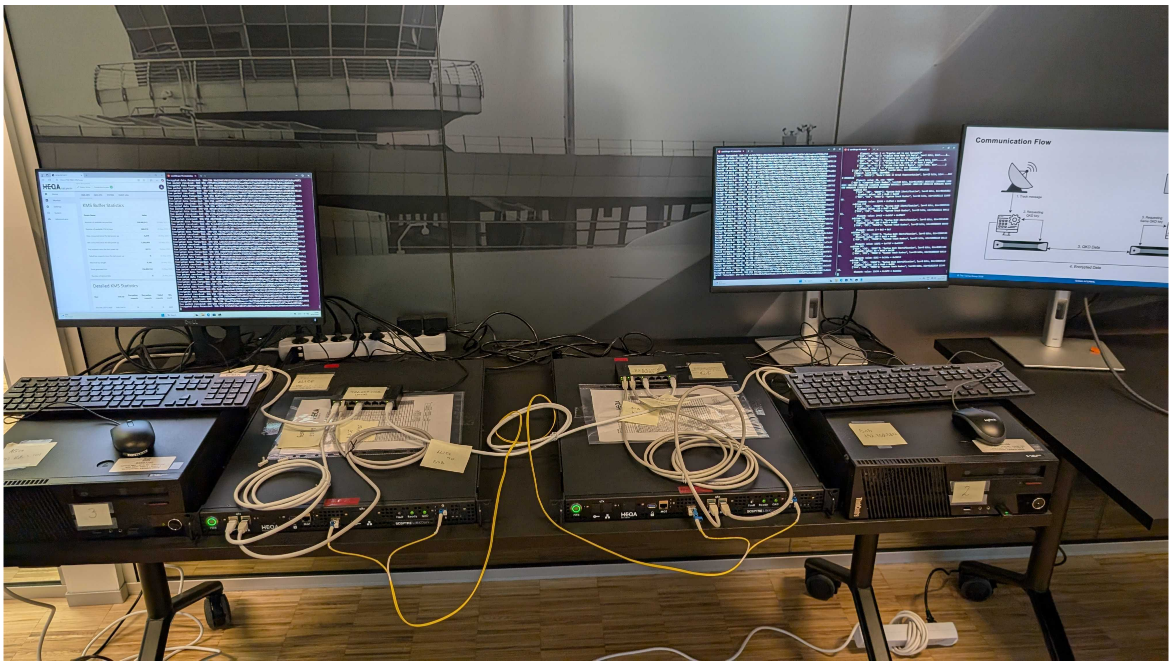


Figure 8.1: QKD setup at Terma headquarters in Søborg.

8.3.1 Unreadable Data Decoded

During the showcase the system was running for circa 2 hours, through which it was observed that the decoded track messages periodically in some cases turned into nonsense gibberish data on ther receiver side. This might be an issue with the ASTERIX generator, since the rest of the implementations relies on well-tested pieces.

8.3.2 Dropped UDP Packets

During testing the system before showcasing it, first UDP transmission were used between the two sites, but we noticed dropped packets. The reason for the package loss could not be resolved, therefore we switched to TCP, more precisely to HTTP, so package fragmentation is handled by the protocol.

8.4 Future Work

With the experience gained during this thesis, the following research extensions are proposed:

- Using a long fiber cable
- Using industry-quality network equipment

- Using the ASTERIX generator as message source
- Using a different communication protocol between sites
- Switching to PQC certifications on the QKD devices
- Requesting multiple keys at once

Using long fiber cable

To make the setup even closer to reality, a long fiber (even 50 km long) cable could be used to measure actual secure bit rate between the QKD devices. To push this further, it could be considered to rent an already deployed fiber cable link from one of the available ISPs in Denmark, where noise will be realistic.

Using industry-quality network equipment

As mentioned in Subsection 8.2.5, the network equipment used in the testbed do not qualify to be used in realistic industrial scenarios. For better results, a different network architecture, segmentation and high-quality devices could be used.

Using the ASTERIX generator as message source

To provide an even more realistic setup, the generator could be used to provide messages via UDP. With this, proper message content can be used and the tool can be customized to send messages even every 100 ms. This would create a better source-simulation of a radar or other equipment.

Using a different communication protocol between sites

In the radar use-case, the two sites communicate via HTTP. The reason for this are the limitations found in Subsection 8.3.2 and Subsection 8.2.3. To speed up the implementation, HTTP was used as transmission protocol, but as it can also be seen in Figure 7.1, it is a large bottleneck in message processing.

Switching to PQC certifications on the QKD devices

The purchased QKD devices deliver keys via HTTPS, for which certifications were generated using RSA key pairs. In many use-cases, QKD devices are expected to be placed in a secure environment, where all accessing applications are also secure, but if this requirement is not fulfilled, then in the presence of a quantum-capable adversary the key request is not considered to be secure. It is proposed to experiment with switching the certifications of the devices to PQC certifications, e.g. with the help of OpenSSL[2].

[2]`https://github.com/openssl/openssl/discussions/27332#discussioncomment-12789592`

Requesting multiple keys at once

In the proposed communication flow in the radar use-case, each message is encrypted with a new QKD key. The reason for this is to measure how the QKD boxes perform in a longer period with many requests, and to simulate maximum security against compromising a key. The whole point of the generalized use-case was to measure QKD key retrieval latency and stability. In both cases, research could be extended with requesting multiple keys at once, since the QKD devices are capable of this, and measure how latency and stability changes. Reusing keys for a fixed amount of messages or a time constraint could also be another factor to look at on.

Chapter 9

Conclusion

This thesis set out to investigate the challenges of integrating existing quantum-resistant technologies, such as PQC and QKD, to provide practical knowledge and experience over implementing a QKD testbed and to measure the performance and stability of the testbed towards a critical infrastructure and generalized use-cases. This was achieved by designing a system and a communication flow based on a radar use-case and a generalized scenario, where both PQC and QKD are facilitated on the application level, and measuring system performance in both cases in the application layer. The goal was to analyze how reliable the system is during simulating messages produced by a radar device, and to analyze latency and performance indicators under a heavier concurrent request load.

The results demonstrated that purchasing and operating QKD devices can be done just like any other server appliance, however there are very specific limitations. Even with taking the key request limitations of the QKD devices into consideration in all scenarios, and never requesting more than 10 keys per second, the devices have not shown a very stable performance and many outliers were found in key request latency, in both the radar and the generalized use-cases. Despite this, integration of PQC and QKD is a significantly easier task than expected, due to the availability and interoperability of existing PQC implementations. The plots and visualizations provide additional insights on the system performance and occasional outliers manifested throughout the experiments.

These findings emphasize the value of research on applying quantum-safe technologies into real-world scenarios, where reliability, stability and latency are key performance indicators, and suggest that this is still not a trivial task to complete. They also show that network components and implementation quality play a crucial role in integration of such technologies.

While the testbed was designed to be isolated and reproducible, it does not yet account for real-world radar communication, protocol implementation, full quantum-resilience or cross-system interference. Future work could extend this approach by incorporating testing with long fiber cables, using high-quality network equipment, using a proper radar message source, implementing a custom communication protocol, enabling PQC-supported communication encryption in different layers and testing different QKD device behaviors to further analyze performance under dynamic and unpredictable conditions (see Section 8.4 for more details).

In summary, this thesis contributes practical knowledge and experience on handling QKD devices, integrating quantum-resistant technologies in the application layer, and latency and stability testing analysis in constrained scenarios, bridging the gap between experimental quantum communication systems and their real-world deployment in performance-sensitive, secure network environments.

Bibliography

[1] IBM. *IBM Quantum Roadmap*. Accessed on April 17, 2025. 2025. URL: https://www.ibm.com/roadmaps/quantum.

[2] Microsoft. *2025: The year to become Quantum-Ready*. Accessed on April 17, 2025. 2025. URL: https://azure.microsoft.com/en-us/blog/quantum/2025/01/14/2025-the-year-to-become-quantum-ready.

[3] Josh Schneider; Ian Smalley. *What is quantum computing?* Ed. by IBM. Aug. 2024. URL: https://www.ibm.com/think/topics/quantum-computing.

[4] NIST. *FIPS-203*. 2024. DOI: 10.6028/NIST.FIPS.203. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf.

[5] NIST. *FIPS-204*. 2024. DOI: 10.6028/NIST.FIPS.204. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf.

[6] NIST. *FIPS-205*. 2024. DOI: 10.6028/NIST.FIPS.205. URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf.

[7] Fortinet Inc. *What Is Cryptography?* Accessed on June 3, 2025. 2025. URL: https://www.fortinet.com/resources/cyberglossary/what-is-cryptography.

[8] R. L. Rivest, A. Shamir, and L. Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: https://doi.org/10.1145/359340.359342.

[9] Victor S. Miller. "Use of Elliptic Curves in Cryptography". In: *Advances in Cryptology — CRYPTO '85 Proceedings*. Ed. by Hugh C. Williams. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 417–426. ISBN: 978-3-540-39799-1.

[10] Neal Koblitz. "Elliptic Curve Cryptosystems". In: *Mathematics Of Computation* 48.177 (1985), pp. 203–209. ISSN: 0025-5718. DOI: 10.1090/S0025-5718-1987-0866109-5. URL: https://www.ams.org/journals/mcom/1987-48-177/S0025-5718-1987-0866109-5/S0025-5718-1987-0866109-5.pdf.

[11] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. 1994, pp. 124–134. DOI: 10.1109/SFCS.1994.365700.

[12] Peter W. Shor. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. Jan. 1996. DOI: 10.1137/S0097539795293172. URL: https://www.arxiv.org/pdf/quant-ph/9508027.

[13] Lov K. Grover. *A fast quantum mechanical algorithm for database search*. 1996. DOI: 10.1145/237814.237866. URL: https://dl.acm.org/doi/pdf/10.1145/237814.237866.

[14] W. Diffie and M. Hellman. "New directions in cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: 10.1109/TIT.1976.1055638.

[15] Charles H. Bennett; Gilles Brassard. *QUANTUM CRYPTOGRAPHY: PUBLIC KEY DISTRIBUTION AND COIN TOSSING*. 1984. DOI: 10.48550/arXiv.2003.06557. URL: https://arxiv.org/pdf/2003.06557.

[16] C. E. Shannon. "Communication theory of secrecy systems". In: *The Bell System Technical Journal* 28.4 (1949), pp. 656–715. DOI: 10.1002/j.1538-7305.1949.tb00928.x.

[17] Marco Tomamichel and Anthony Leverrier. "A largely self-contained and complete security proof for quantum key distribution". In: *Quantum* 1 (July 2017), p. 14. ISSN: 2521-327X. DOI: 10.22331/q-2017-07-14-14. URL: https://doi.org/10.22331/q-2017-07-14-14.

[18] Hoi-Kwong Lo, Xiongfeng Ma, and Kai Chen. "Decoy State Quantum Key Distribution". In: *Phys. Rev. Lett.* 94 (23 June 2005), p. 230504. DOI: 10.1103/PhysRevLett.94.230504. URL: https://link.aps.org/doi/10.1103/PhysRevLett.94.230504.

[19] Sebastian Paul, Patrik Scheible, and Friedrich Wiemer. "Towards post-quantum security for cyber-physical systems: Integrating PQC into industrial M2M communication1". In: *Journal of Computer Security* 30.4 (2022), pp. 623–653. DOI: 10.3233/JCS-210037. eprint: https://doi.org/10.3233/JCS-210037. URL: https://doi.org/10.3233/JCS-210037.

[20] Javier Blanco-Romero et al. "Integrating Post-Quantum Cryptography into CoAP and MQTT-SN Protocols". In: *2024 IEEE Symposium on Computers and Communications (ISCC)*. 2024, pp. 1–6. DOI: 10.1109/ISCC61673.2024.10733716.

[21] M. Sasaki et al. "Field test of quantum key distribution in the Tokyo QKD Network". In: *Opt. Express* 19.11 (May 2011), pp. 10387–10409. DOI: 10.1364/OE.19.010387. URL: https://opg.optica.org/oe/abstract.cfm?URI=oe-19-11-10387.

[22] Miralem Mehic et al. "Quantum Key Distribution: A Networking Perspective". In: *ACM Comput. Surv.* 53.5 (Sept. 2020). ISSN: 0360-0300. DOI: 10.1145/3402192. URL: https://doi.org/10.1145/3402192.

[23] Yuan Cao et al. "The Evolution of Quantum Key Distribution Networks: On the Road to the Qinternet". In: *IEEE Communications Surveys Tutorials* 24.2 (2022), pp. 839–894. DOI: 10.1109/COMST.2022.3144219.

[24] Sheng-Kai Liao et al. "Satellite-to-ground quantum key distribution". In: *Nature* 549.7670 (Sept. 2017), pp. 43–47. ISSN: 1476-4687. DOI: 10.1038/nature23655. URL: https://doi.org/10.1038/nature23655.

[25] Yu-Ao Chen et al. "An integrated space-to-ground quantum communication network over 4,600 kilometres". In: *Nature* 589.7841 (Jan. 2021), pp. 214–219. ISSN: 1476-4687. DOI: 10.1038/s41586-020-03093-8. URL: https://doi.org/10.1038/s41586-020-03093-8.

[26] Swetha Ranganathan et al. "Integrating Quantum Key Distribution (QKD) with Post-Quantum Cryptography (PQC): Combining BB84 Protocol with Lattice-Based Cryptographic Techniques". In: *2024 IEEE 4th International Conference on ICT in Business Industry Government (ICTBIG)*. 2024, pp. 1–9. DOI: 10.1109/ICTBIG64922.2024.10911719.

[27] Carlos Rubio Garcia et al. "Enhancing the Security of Software Defined Networks via Quantum Key Distribution and Post-Quantum Cryptography". English. In: *Distributed Computing and Artificial Intelligence, Special Sessions I, 20th International Conference*. Ed. by Rashid Mehmood et al. Lecture Notes in Networks and Systems, LNNS. 20th International Conference on Distributed Computing and Artificial Intelligence, DCAI 2023, DCAI 2023 ; Conference date: 12-07-2023 Through 14-07-2023. Germany: Springer, July 2023, pp. 428–437. ISBN: 978-3-031-38317-5. DOI: 10.1007/978-3-031-38318-2_42. URL: https://www.dcai-conference.net/.

[28] Carlos Rubio Garcia et al. "Quantum-Resistant TLS 1.3: A Hybrid Solution Combining Classical, Quantum and Post-Quantum Cryptography". English. In: *2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD 2023*. 2023 IEEE 28th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD) ; Conference date: 06-11-2023 Through 08-11-2023. United States: Institute of Electrical and Electronics Engineers, Mar. 2024, pp. 246–251. DOI: 10.1109/CAMAD59638.2023.10478407.

[29] Javier Blanco-Romero et al. *QKD-KEM: Hybrid QKD Integration into TLS with OpenSSL Providers*. 2025. DOI: 10.48550/arXiv.2503.07196. URL: https://arxiv.org/pdf/2503.07196.

[30] Dominik Marchsreiter and Johanna Sepúlveda. "A PQC and QKD Hybridization for Quantum-Secure Communications". In: *2023 26th Euromicro Conference on Digital System Design (DSD)*. 2023, pp. 545–552. DOI: 10.1109/DSD60849.2023.00081.

[31] Nick Aquina, Simon Rommel, and Idelfonso Tafur Monroy. "Quantum secure communication using hybrid post-quantum cryptography and quantum key distribution". In: *2024 24th International Conference on Transparent Optical Networks (ICTON)*. 2024, pp. 1–4. DOI: 10.1109/ICTON62926.2024.10648124.

[32] Federico Giacon, Felix Heuer, and Bertram Poettering. "KEM Combiners". In: *Public-Key Cryptography – PKC 2018*. Ed. by Michel Abdalla and Ricardo Dahab. Cham: Springer International Publishing, 2018, pp. 190–218. ISBN: 978-3-319-76578-5. DOI: 10.1007/978-3-319-76578-5_7.

[33] Marc Geitz, Ronny Döring, and Ralf-Peter Braun. "Hybrid QKD PQC Protocols implemented in the Berlin OpenQKD testbed". In: *2023 8th International Conference on Frontiers of Signal Processing (ICFSP)*. 2023, pp. 69–74. DOI: 10.1109/ICFSP59764.2023.10372894.

[34] Sanzida Hoque, Abdullah Aydeger, and Engin Zeydan. "Exploring Post Quantum Cryptography with Quantum Key Distribution for Sustainable Mobile Network Architecture Design". In: *Proceedings of the 4th Workshop on Performance and Energy Efficiency in Concurrent and Distributed Systems*. PECS '24. Pisa, Italy: Association for Computing Machinery, 2024, pp. 9–16. ISBN: 9798400706448. DOI: 10.1145/3659997.3660033. URL: https://doi.org/10.1145/3659997.3660033.

[35] Carlos Rubio García et al. "Secure and Agile 6G Networking – Quantum and AI Enabling Technologies". In: *2023 23rd International Conference on Transparent Optical Networks (ICTON)*. 2023, pp. 1–4. DOI: 10.1109/ICTON59386.2023.10207418.

[36] Jongmin Ahn et al. "Toward Quantum Secured Distributed Energy Resources: Adoption of Post-Quantum Cryptography (PQC) and Quantum Key Distribution (QKD)". In: *Energies* 15.3 (2022). ISSN: 1996-1073. DOI: 10.3390/en15030714. URL: https://www.mdpi.com/1996-1073/15/3/714.

[37] IEEE. *IEEE Standard for Ethernet*. Accessed on April 20, 2025. 2025. URL: https://standards.ieee.org/ieee/802.3/10422/.

[38] Giacomo Longo et al. "Attacking (and Defending) the Maritime Radar System". In: *IEEE Transactions on Information Forensics and Security* 18 (2023), pp. 3575–3589. DOI: 10.1109/TIFS.2023.3282132.

[39] Eurocontrol. *Specification for Surveillance Data Exchange ASTERIX*. Accessed on April 20, 2025. 2025. URL: https://www.eurocontrol.int/publication/cat062-eurocontrol-specification-surveillance-data-exchange-asterix-part-9-category-062.

[40] BBC. *What are Israel's Iron Dome, David's Sling, Arrow and Thaad missile defences?* Accessed on April 20, 2025. Oct. 2024. URL: https://www.bbc.com/news/world-middle-east-20385306.

[41] The Associated Press Julia Frankel. *Is Israel's Iron Dome missile defense system ironclad?* Available: https://apnews.com/article/iron-dome-missile-defense-israel-palestinians-rockets-hamas-f57508d0e53945d1071d4582fcdae8d2, Accessed on April 20, 2025. Oct. 2023.

[42] Reuters Patricia Zengerle. *U.S. House approves* $1billionforIsrael's'IronDome'missile - defensesystem$. Accessed on April 20, 2025. Sept. 2021. URL: `https://www.reuters.com/world/us/us-house-backs-bill-provide-1-billion-israel-iron-dome-system-2021-09-23/`.

[43] The Times of Israel. *Iron Dome is facing its greatest test in war with Hamas*. Accessed on April 20, 2025. Oct. 2023. URL: `https://www.timesofisrael.com/iron-dome-is-facing-its-greatest-test-in-war-with-hamas`.

[44] FasterCapital. *Interarrival Times: Timing the Unexpected: Interarrival Times in Exponential Distribution*. Accessed on May 20, 2025. Mar. 2025. URL: `https://fastercapital.com/content/Interarrival-Times--Timing-the-Unexpected--Interarrival-Times-in-Exponential-Distribution.html`.

Appendix A

High Resolution Plots

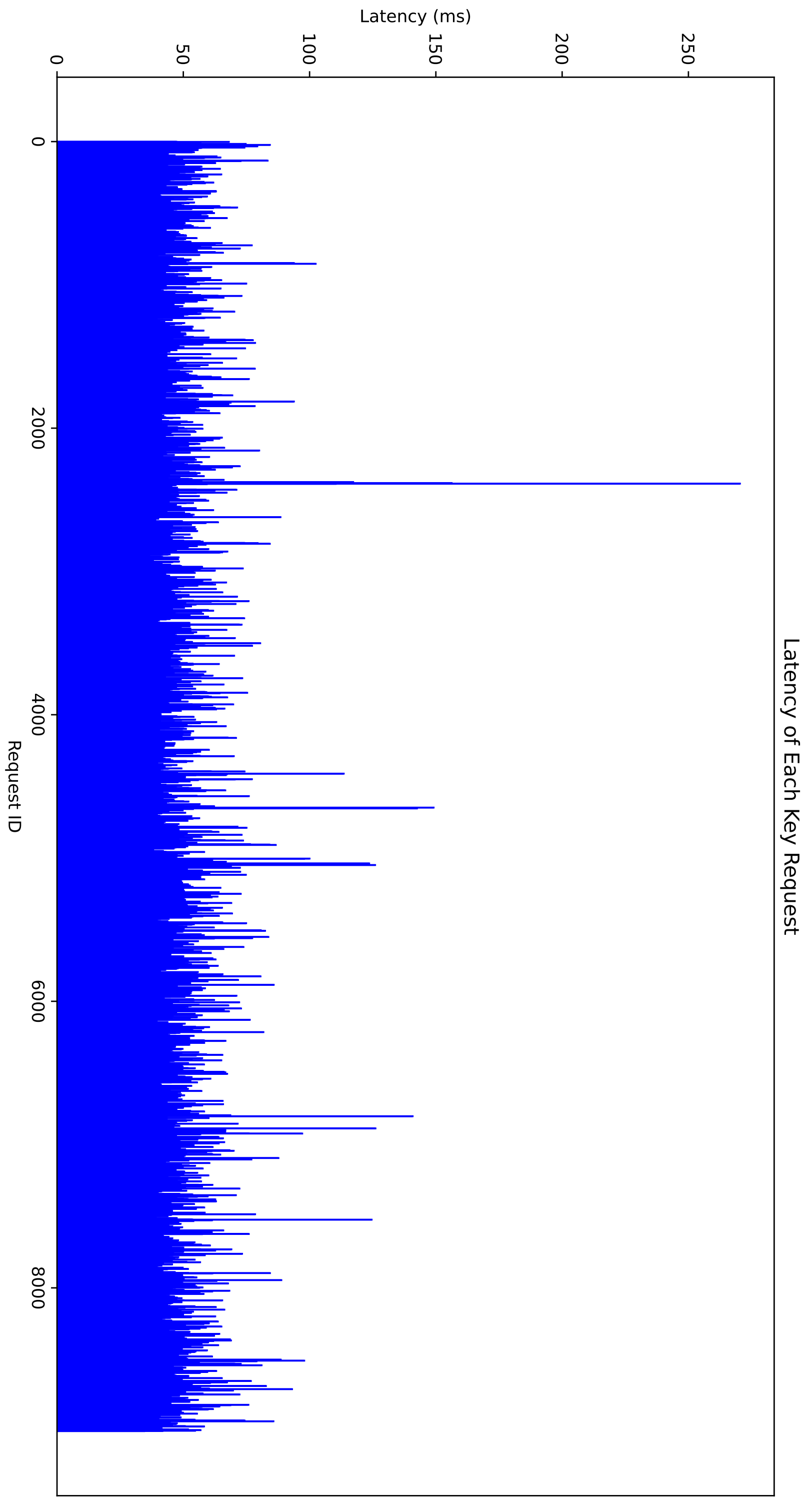


Figure A.1: Latency of each request in the test.

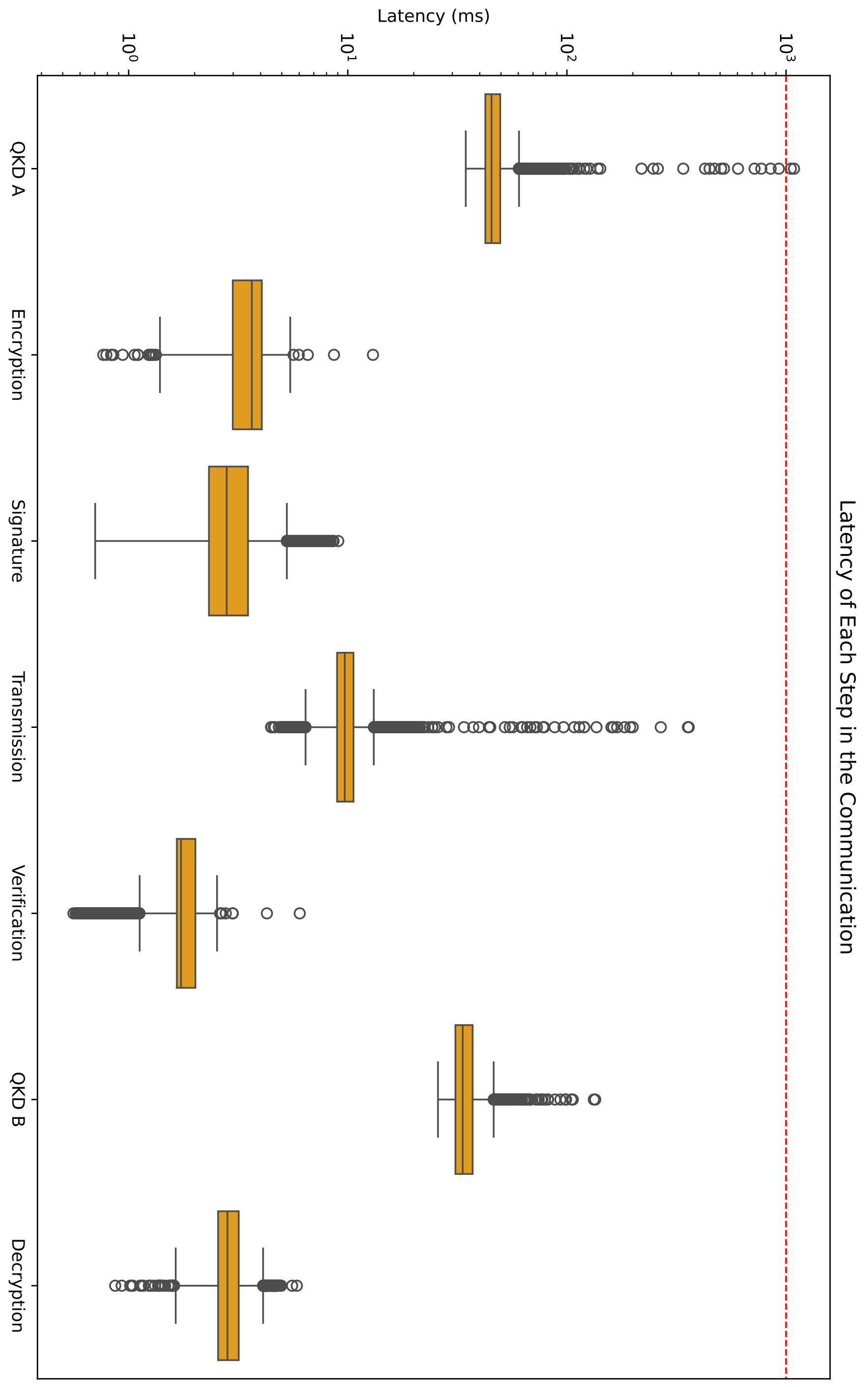


Figure A.2: Box plot of latency projected to each step in the communication over 10 000 test rounds. 1000 ms threshold marked with red.