VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Khanh Hoang

POST-QUANTUM CRYPTOGRAPHY FOR PUBLIC KEY INFRASTRUCTURE

PQC - PKI

Information Technology
2023

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Information Technology

ABSTRACT

Author	Khanh Hoang
Title	Post-Quantum Cryptography for Public Key Infrastructure
Year	2023
Language	English
Pages	82 + Appendices
Name of Supervisor	Jukka Matila

The emergence of quantum computers has presented a significant challenge to the traditional encryption algorithms that are commonly employed in public key infrastructure. As a result, many applications such as email, messaging, e-commerce, and banking authentication, which rely on these encryption algorithms, are at risk of being compromised. This thesis aims to analyze and research potential post-quantum encryption algorithms that could potentially replace the traditional algorithms in the existing public key infrastructure. The thesis utilizes observation and applied methods in the field of cyber security to evaluate the effectiveness of various post-quantum encryption algorithms and identify implementations of the selected algorithms from reputable post-quantum competitions. This provides valuable insight into the efficacy of post-quantum encryption and its ability to be integrated with existing or simplified versions of public key infrastructure. To support this research, a range of implementations of each post-quantum encryption algorithm were identified in various languages and can be utilized by organizations for upcoming projects or further research if necessary.

The thesis highlights the potential danger posed by quantum computing to traditional encryption algorithms, with the possibility that existing algorithms may be rendered obsolete soon. To prepare for this eventuality, companies should source information and build a post-quantum public key infrastructure, or consider replacing existing algorithms with post-quantum algorithms as soon as practicable. This will ensure that their systems remain secure against quantum computing threats. Through providing comprehensive analysis and implementation suggestions, this thesis provides a valuable resource to the cybersecurity field and could have far-reaching implications for the future of public key infrastructure.

Keywords Post Quantum Cryptography, Public Key Infrastructure

CONTENTS

LIST OF FIGURES AND TABLES

LIST OF APPENDICES

LIST OF ACRONYMS

ACME: Automated Certificate Management Environment protocol

AES: Advanced Encryption Standard

AS: Authentication Server

ASIC: Application-Specific Integrated Circuits

CA: Certificate Authorities

CLI: Command Line Interface

CRL: Certificate Revocation List

CSR: Certificate Signing Request

DES: Data Encryption Standard

ECC: Elliptic Curve Cryptography

ECDH: Elliptic Curve Diffie-Hellman

ECDSA: Elliptic Curve Digital Signature Algorithm

EDIFACT: Electronic Data Interchange Administration, Commerce, and Transport

EJBCA CE: Enterprise JavaBeans Certificate Authority Community Edition

EST: Enrollment over Secure Transport

FPGA: Field-Programmable Gate Arrays

HFE: Hidden Field Equations

HSM: Hardware Security Modules

IoT: Internet of Things

KEM: Key Encapsulation Mechanism

GUI: Graphical User Interface

LRA: Local Registration Authority

LWE: Learning With Errors

MD5: Message Digest 5

MQ: Multivariate quadratic

NIST: National Institute of Standards and Technology

OCSP: Online Certificate Status Protocol

OpenPGP: Open Specification for Pretty Good Privacy Working Group

OQS: Open Quantum Safe

OTS: One-time Signature

PKC: Public Key Cryptography

PKE: Public Key Encryption

PKI: Public Key Infrastructure

PQC: Post Quantum Cryptography

QKD: Quantum Key Distribution

RA: Registration Authority

RBAC: Role-Based Access Control

RSA: Rivest Shamir Adleman

SCEP: Simple Certificate Enrollment Protocol

SPKI: Simple Public Key Infrastructure

SSH: Secure Shell

SSL: Secure Sockets Layer

TLS: Transport Layer Security

UOV: Unbalanced Oil and Vinegar

VA: Validation Authority

XMSS: Extended Merkle Signature Scheme

1 INTRODUCTION

Within the realm of information security, cryptography plays an essential role in protecting the integrity, confidentiality, and authenticity of digital communications and transactions. Over the years, a variety of encryption algorithms have been developed, forming the basis of modern public key infrastructure (PKI). However, with the rapid advancement of quantum computing, the traditional cryptographic algorithms that have been relied upon are facing an imminent threat.

Quantum computers possess the potential to revolutionize computing power by utilizing the principles of quantum mechanics to perform computations exponentially faster than classical computers. While this technological breakthrough holds immense promise for many fields, it also presents a significant challenge to the security landscape. The inherent computational power of quantum computers has the potential to render the currently employed cryptographic algorithms vulnerable to attacks, thus compromising the confidentiality and privacy of sensitive data.

To address this emerging threat, a new field of study known as post-quantum cryptography has gained prominence. Post-quantum cryptography focuses on developing encryption algorithms that are resistant to attacks from quantum computers, thus ensuring the continued security of digital communications and transactions even in the face of this unprecedented computing power.

This thesis aims to investigate the implementation and integration of post-quantum encryption algorithms into the existing public key infrastructure. The goal is to assess the feasibility and efficacy of these quantum-resistant algorithms as potential replacements for the traditional cryptographic algorithms currently employed. By anticipating the future need for secure encryption mechanisms, this research aims to bridge the gap between the current cryptographic landscape and the looming threat of quantum computers.

In this study, we will explore the fundamental concepts and principles of post-quantum cryptography, examining various encryption schemes, signature algorithms, and key exchange protocols that offer resistance against quantum attacks. We will analyze their strengths, weaknesses, and performance characteristics in the context of public key infrastructure. Additionally, we will investigate the challenges associated with implementing these new algorithms, such as computational complexity, key size, and compatibility with existing systems.

The research will involve a comprehensive review of the current state of post-quantum cryptographic algorithms, including lattice-based, code-based, multi-variate, and hash-based, among others. By examining their mathematical foundations, security assumptions, and practical implementations, we aim to provide a comprehensive understanding of the strengths and limitations of these algorithms.

Furthermore, the thesis will investigate the challenges associated with the transition from traditional cryptographic algorithms to post-quantum cryptography in PKI. We will explore the strategies and best practices for migrating existing systems to quantum-resistant algorithms, ensuring a smooth transition without compromising security or disrupting critical operations.

By conducting this research, we hope to contribute to the ongoing efforts of the information security community in ensuring the long-term security of digital communications and transactions. The findings and recommendations derived from this study will aid in forming a robust and quantum-resistant public key infrastructure, providing organizations and individuals with the necessary tools to protect their sensitive information against the growing threat of quantum computing.

2 PURPOSE OF THE RESEARCH

2.1 Problem and Questions

With the rapid advancements in quantum computing, it is anticipated that current state-of-the-art cryptographic algorithms will offer only short-term data security. To address this challenge, various mathematical approaches have been proposed to enhance the resilience of algorithms against both traditional and quantum computer attacks. However, the existing implementations of PKI lack support for this new generation of algorithms. As VisionSpace Technologies, a company operating in the space industry, explores the development of next-generation spacecraft and the satellite ground segment, there is a pressing need to gain a comprehensive understanding of how these technologies can be effectively combined. Furthermore, it is crucial to analyze and document the tradeoffs and limitations associated with integrating post-quantum cryptography algorithms within the context of the space sector.

Therefore, the primary objective of this research is to conduct a state-of-the-art analysis of post-quantum cryptography algorithms and explore their integration with public key infrastructure for application in next-generation spacecraft and the satellite ground segment. This investigation seeks to explore several key questions surrounding the integration of post-quantum cryptography in various sectors, with a specific focus on the space industry. Firstly, it aims to examine the current limitations of traditional cryptographic algorithms when faced with the emerging capabilities of quantum computing. Secondly, the investigation will explore how post-quantum cryptography algorithms can be effectively integrated with existing public key infrastructure implementations. Next, the investigation will consider the potential tradeoffs and considerations associated with implementing post-quantum cryptography in the space sector. Finally, the investigation will assess the implications and advantages of integrating post-quantum cryptography algorithms with public key infrastructure for next-generation spacecraft and the satellite ground segment.

By addressing these questions, this research endeavors to provide valuable insights into the state-of-the-art in post-quantum cryptography, facilitate the development of a proof-of-concept integration with public key infrastructure, and offer comprehensive documentation and analysis of the tradeoffs and limitations associated with utilizing these technologies within the space industry.

2.2 Method

In this thesis, the observation method and applied observational method will be used. Observational research, which includes exploratory studies, is highly valuable for comprehending the behavior of real-world cyber systems and their techno-social aspects. This research approach is particularly effective in addressing broad and open-ended research questions. Observational research methods involve carefully observing real-world environments and analyzing collected data to reveal interesting patterns and phenomena. Exploratory studies involve the collection, analysis, and interpretation of observations related to known designs, systems, models, abstract theories, or subjects. These studies primarily use an inductive process to gain understanding, as opposed to experimental research which progresses from a general theory to a specific understanding. Exploratory studies focus on specific phenomena to identify patterns and develop general theories of behavior, with the primary emphasis lying on evaluating and analyzing data rather than creating new designs or models. In the social and health sciences, this type of research is often referred to as qualitative studies, which prioritize gaining perspective and determining relative importance /1/.

Applied observational research is commonly conducted in the field of cybersecurity. The main distinction between applied studies and observational studies lies in their scope. Applied studies focus on observing specific subjects, such as their performance, function, or security, while fundamental observational studies observe the entire system without any presumptions about its behavior. The focus of this research is to explore how to adapt scientific techniques and approaches for studying real-world applications, with an emphasis on applied observational studies. Although various types of studies can be employed in an applied context, case-control studies, case studies, and case reports are frequently used. Applied studies aim to gain an understanding of performance or function by introducing specific changes or systems that can be measured.

In an applied study, researchers observe a new solution to assess its performance under different conditions, often involving the introduction of a new defensive feature or system change. This is accompanied by an expectation or prediction. The researcher anticipates or assumes how the subject should behave. The goal of an applied study is to understand the impact of a change or effect under observation while assuming a particular performance or behavior. For instance, studying the performance cost implications of adding a firewall to a network would be considered an applied study /1/.

2.3 Assurance and Ethics

The assurance and ethics of research are guided by three fundamental principles from the Belmont Report. Firstly, respect for persons emphasizes individual autonomy and informed consent, including privacy and confidentiality considerations. Secondly, beneficence involves minimizing harm and maximizing potential benefits, with clear communication of risks and benefits to participants. Finally, justice focuses on fairness and equal treatment, ensuring unbiased participant selection and providing access to information. These principles ensure the protection of participants' rights and well-being. In this thesis, adherence to these ethical principles will uphold integrity, safeguard autonomy, minimize harm, and promote fairness throughout the research process /2/.

For research to be considered ethical, reliable, and credible, it must adhere to the principles of responsible conduct of research. The responsible conduct of research is an essential aspect of ensuring the quality and integrity of research organizations. From the perspective of research integrity, the following premises underlie the responsible conduct of research: Research should uphold the principles of integrity, thoroughness, and accuracy throughout the entire research process, including data collection, recording, presentation, and evaluation of results. The methods employed for data acquisition, research, and evaluation should align with scientific criteria and uphold ethical standards. When disseminating research findings, they should be communicated openly and responsibly to contribute to the advancement of scientific knowledge. Researchers should demonstrate respect for the work and accomplishments of other researchers by appropriately acknowledging and citing their publications. They should acknowledge and give due credit to the contributions and significance of others' work when conducting their research and publishing its outcomes. Researchers should comply with the established standards of scientific knowledge when planning, conducting, and reporting their research. This includes accurately recording and reporting the data obtained during the research process. Research projects should obtain the necessary research permits and undergo ethical review, particularly in fields that require such scrutiny. Before commencing the research or involving researchers, all parties involved in the research project or team should reach an agreement on the researchers' rights, responsibilities, and obligations. This agreement should also address authorship principles, data archiving, and data access. Further specifications may be made as the research progresses. Any sources of funding, conflicts of interest, or relevant commitments that may influence the conduct of the research should be disclosed to all members of the research project. These should also be reported when publishing the research results. By upholding these premises, researchers can ensure ethical and responsible conduct in research, contributing to the credibility, reliability, and integrity of the scientific community /3/.

3 THEORETICAL BACKGROUND

3.1 Public Key Infrastructure

PKI is a highly secure and sophisticated system that facilitates secure communication by increasing the security of a network and providing the foundation for securing all internet-connected things, including e-mail, messaging, and e-commerce. It employs advanced encryption algorithms such as Rivest Shamir Adleman (RSA) and Elliptic Curve Cryptography (ECC), digital certificates, and certificate authorities (CA) to guarantee the integrity of data and verify the identities of both the senders and receivers of digital transactions. With the help of PKI, organizations can ensure the confidentiality, authenticity, and integrity of their digital communication and transactions /4/.

PKI is a widely used security infrastructure that employs public key cryptography to ensure secure digital communication services are delivered using public-key concepts and techniques /5/.

One of the significant advantages of public key cryptography is that it eliminates the need for exchanging secret keys. However, in public key cryptography, effective management of keys is essential. This is where public key infrastructures come into play, providing secure management of key pairs throughout their lifecycle to ensure the confidentiality and authenticity of digital transactions /6/.

PKI is a pivotal technology that plays a significant role in ensuring the necessary level of security and scalability for digital transactions. While other security approaches such as Identity-based Cryptography, Certificateless Cryptography have been either scalable but not secure, or secure but not scalable, PKI provides a comprehensive framework that offers both security and scalability for online communication. In addition, PKI is adaptable and flexible, making it a standard that can be used across different industries and applications in the coming years /7/.

PKI ensures confidentiality through encryption, authenticity via digital signatures, and integrity using hash functions. It uses both asymmetric and symmetric keys, certificates, and trust models to secure communications and verify the identity of the parties involved. Data privacy and secrecy must be protected by cryptographic encryption mechanisms to be considered confidential. Integrity refers to the assurance that data cannot be altered or corrupted and that transactions cannot be changed. By using public key certificates and digital signature envelopes, authentication means confirming that the identity of entities is provided /8/.

3.1.1 Public Key Cryptography

Public Key Cryptography (PKC), also known as public-key encryption and asymmetric encryption, is an advanced encryption technology that employs a key pair consisting of a public key and a private key. The public key is made available to anyone who needs to use it, while the private key must be kept confidential to maintain its integrity. PKC enables secure communication between two parties without prior knowledge of each other and eliminates the need for sharing a secret key. It accomplishes this by utilizing the public key cryptography process, which involves mathematical algorithms that enable secure encryption and decryption of digital data. PKC's flexibility and robustness make it a popular choice for securing digital communication channels and ensuring the confidentiality, integrity, and authenticity of digital transactions /9/.

Using public-key encryption (PKE), a sender can encrypt a message and send it to a recipient who has the corresponding secret key by using the recipient's public key. One of the main objectives would be to design effective PKE schemes with weak and reasonable computational assumptions that are provably secure in strong security notions /10/.

The main principle of a public key cryptosystem is the use of two different but related keys: one for encryption and one for decryption. Only the decryption key can be used for encryption, and vice versa, for decryption. The encryption key can be made public because the decryption key cannot be determined from the encryption key /6/.

One of the significant advantages of public key encryption is that it simplifies the distribution of keys by making public keys accessible to all parties. This feature not only ensures confidentiality but can also be used to establish secure identification protocols /6/.

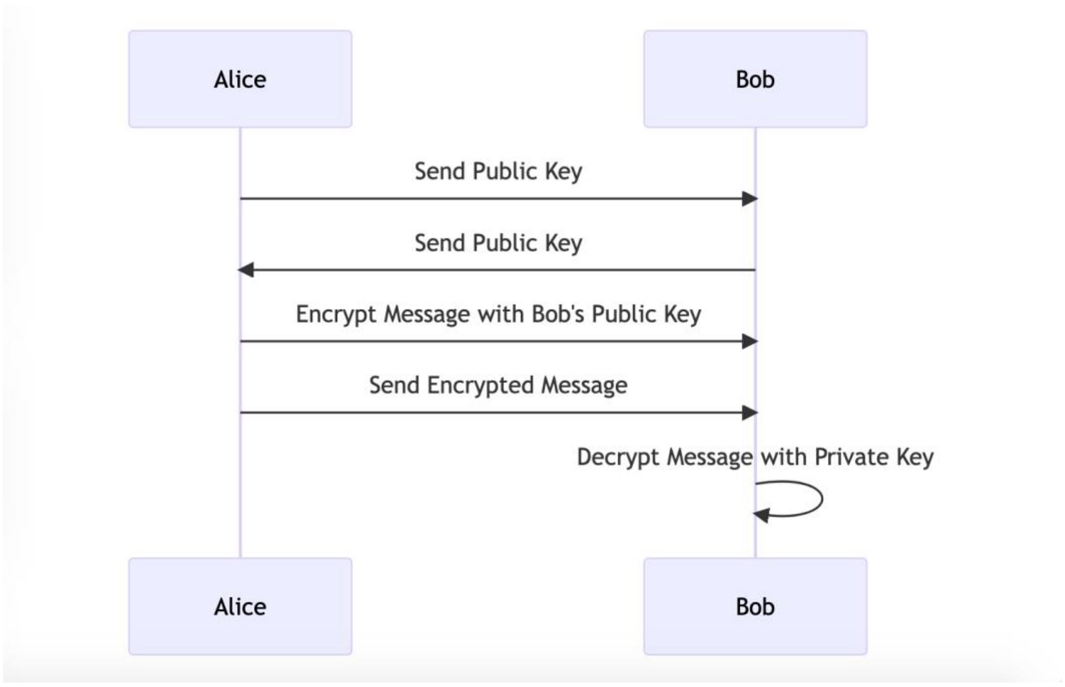

Alice
Bob
Send Public Key
Send Public Key
Encrypt Message with Bob's Public Key
Send Encrypted Message
Decrypt Message with Private Key
Alice
Bob

Figure 1. Public Key Cryptography Process

3.1.2 Digital Certificates

A digital certificate is a fundamental mechanism used in modern cryptography to establish the identity of users, devices, applications, and services. It involves the generation of a private key and the binding of a corresponding public key to its owner, which creates a unique identifier known as the certificate. This certificate serves as a trusted third-party verification of identity, which helps to establish secure communication channels between different parties. Digital certificates are widely used in various applications, including secure web browsing, email encryption as S/MIME (Secure/Multipurpose Internet Mail Extensions), and digital signatures /11/.

A data structure known as a digital certificate links personal data to a public key. A private key is the source of the public key. There is no way to identify the owner of a public key by looking at it. A public key can be used to identify the owner of the key and confirm that it is the correct key by being combined with other identifying information, such as a name, address, and other details. Of course, we want to make sure the digital certificate is authentic by having a reliable third-party digitally certify it to make sure no one tampers with it and inserts a different key. The issuer of the digital certificate is this dependable third party.

It is not required to issue a digital certificate to an individual. In actuality, most aren't. A wide range of entities, including people, businesses, groups, organizations, governments, and things, maybe granted digital certificates. The certificate subject is the organization whose identity is linked to the public key in the certificate /12/.

A digital certificate is a method of establishing electronic identities. These certificates are pieces of digital information that allow each entity to be verified. To make this possible, an authority must control, distribute, and verify the identities. A certificate authority is one of the most commonly used authorities to accomplish this. There are numerous CA providers. During a typical Internet session, a dozen different CAs could be used without even realizing it /9/.

3.1.3 Certificate Authorities

A third party has signed or validated the data in the digital certificate of the authentication server (AS), which contains information in a standard format that identifies the owner. The AS and the supplicants both know and trust the third party, also known as a CA. To validate the certificate it receives from the AS, the supplicant must also have the CA certificate. A public key that can be used to assist in decrypting messages from the AS /13/.

A company or organization that issues certificates is called a CA. It is the CA's duty as a reputable entity to guarantee the validity and legitimacy of the certificates it issues. To comply with this requirement, the CA must prove beyond a reasonable doubt that each certificate's public key belonged to the organization claiming to have issued it. The CA must be able to demonstrate the legitimacy of any certificate it issues upon request to uphold this trust. The function of a CA is essential to the public key infrastructure (PKI) that supports secure internet communication /14/.

The two most common types of CAs are. A private CA is only trusted by members of its organization and is responsible for only issuing certificates to members of that organization. To issue certificates to any member of the public, a public CA must enjoy public trust. Depending on the type of CA that issued the certificate and the type of certificate itself, different burdens of proof apply /14/.

CA carries out the following duties in the server networks: establishes a certificate requestor's identity. To issue a certificate, the CA must first verify the identity of the requestor. The CA issues the requested type of certificate to the user, computer, network device, or service after verifying the requestor's identity. The content of the issued certificate depends on the type of certificate requested. Managing certificate revocation by publishing a certificate revocation list (CRL) at specified intervals. The CRL includes a record of certificate serial numbers that have been revoked, along with the corresponding reason codes for each revocation /15/.

3.1.4 Certificate Revocation

To ensure the security of digital communications, it is essential to maintain the integrity of certificates. To prevent compromised certificates from being used, a list of revoked certificates, known as the CRL, must be maintained and regularly updated. This allows users to check the validity of certificates before accepting them /16/.

The revocation of a certificate, as the name suggests, enables a CA to revoke a certificate and notify browsers of the change. However, the process of revocation has a complex history. CRLs were initially proposed to allow CAs to maintain a list of revoked certificates. However, the use of CRLs has been limited due to the potential for the lists to grow significantly, requiring frequent updates and increasing the burden on CAs and browsers alike. As such, other methods for revocation, such as Online Certificate Status Protocol (OCSP), have been developed to address these issues /17/.

Devices can check CRL to see if a certificate has been revoked before it expires. The serial number of the certificate (issued by the granting authority) and the date of revocation make up a list of certificates that have been revoked. The CRL database may be found on the CA or an external server. The CRL will by default be kept locally by the CA. The cs-server subconfiguration mode is used to configure this /7/.

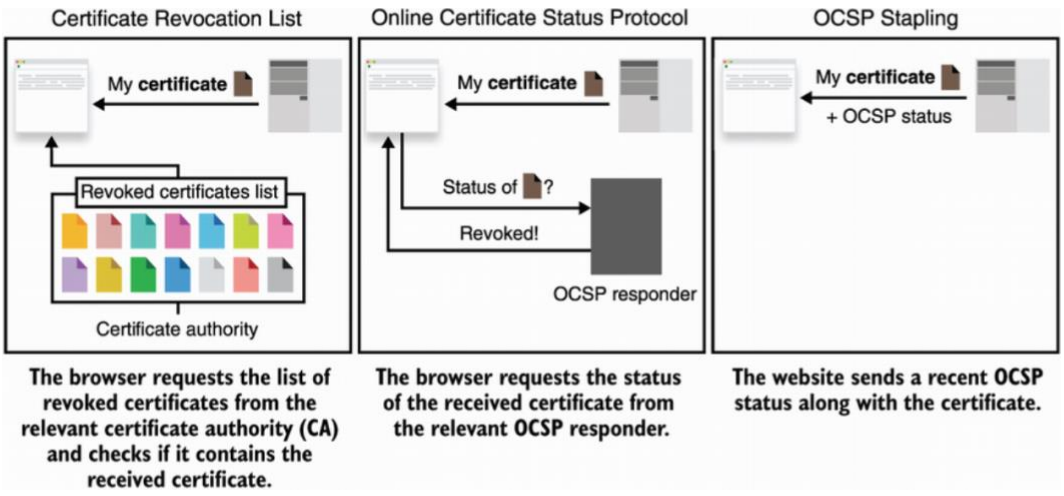

Certificate Revocation List
Online Certificate Status Protocol
OCSP Stapling
My certificate
Revoked certificates list
Certificate authority
My certificate
Status of ?
Revoked!
OCSP responder
My certificate
+ OCSP status
The browser requests the list of revoked certificates from the relevant certificate authority (CA) and checks if it contains the received certificate.
The browser requests the status of the received certificate from the relevant OCSP responder.
The website sends a recent OCSP status along with the certificate.

Figure 2. Certificate revocation: CRLs, OCSP, and OCSP stapling /17/.

3.1.5 PKI Standards

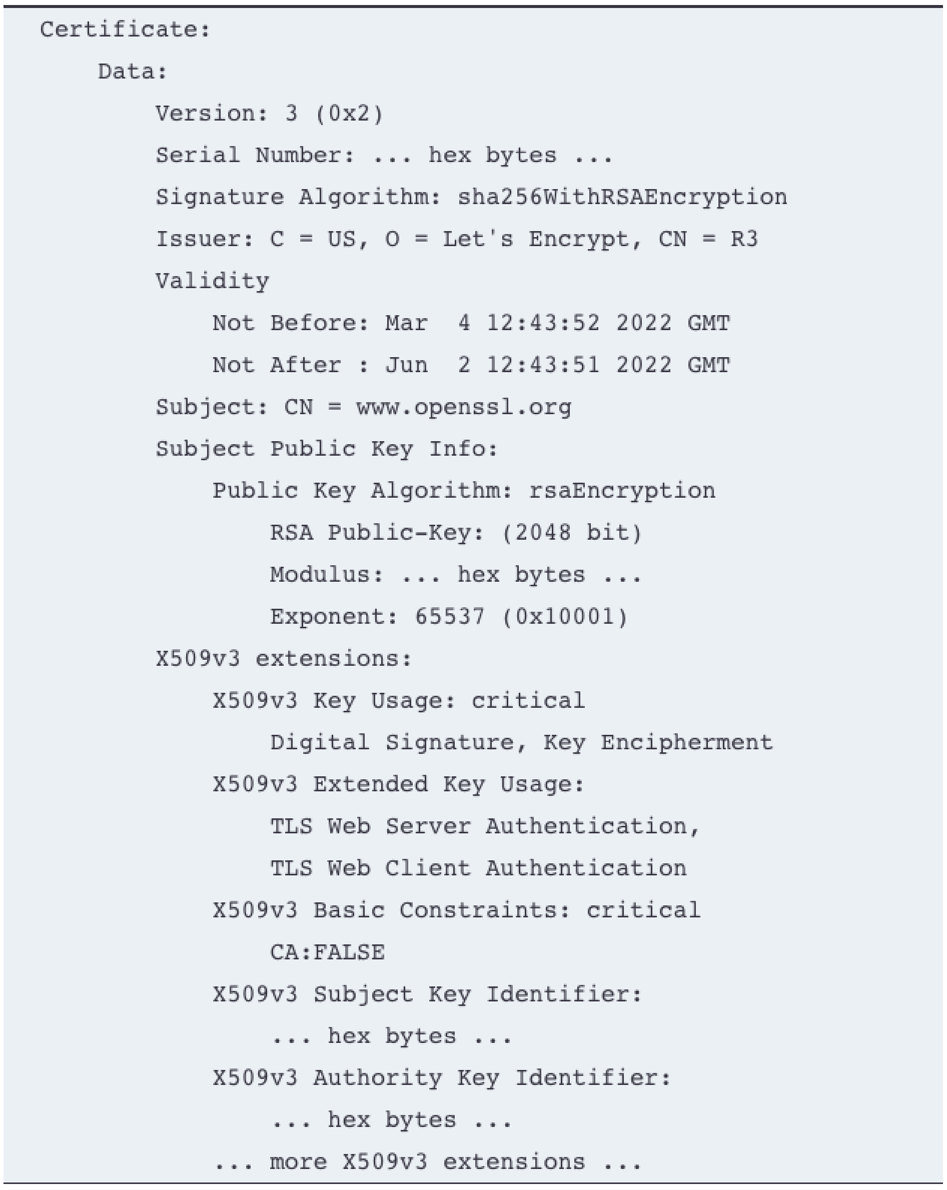
The standardization of PKIs has become increasingly important as these systems are adopted more widely. Standardization efforts have been devoted to all aspects of PKIs, including certificate formats, certificate profiling, and certificate and CRL repository issues. The potential benefits of standardization include the ability to enable large-scale interworking between PKIs, as well as lower costs through economies of scale and increased competition, ultimately contributing to a more secure and efficient digital ecosystem /18/.

The groups listed here involved in standardizing PKIs are interrelated to different degrees. Efforts focused on defining certificate formats, such as X.509, Simple Public Key Infrastructure (SPKI), Open Specification for Pretty Good Privacy Working Group (OpenPGP), and Electronic Data Interchange Administration, Commerce, and Transport (EDIFACT), are largely independent of each other, but collectively aim to provide a common language for PKI systems to communicate. On the other hand, those focused on profiling certificates for specific environments and uses, including PKIX, TC68, S/MIME, IPsec, TLS, WAP, XMLdsig, XMLenc, and SOAP, overlap to a greater degree. Finally, certificate and CRL repository issues, primarily for X.509 formats, are the focus of efforts such as X.500, LDAP, and XKMS /19/.

3.1.6 X.509 Formats

The X.509 format is the most widely accepted format for certificates and has seen three versions since its introduction in 1988, namely X.509v1, X.509v2, and X.509v3. The latter version was released in 1996 and has been widely adopted due to its increased security and support for extensions. These extensions allow for additional information to be included in the certificate such as an issuer's identity, the purpose of the certificate, and so on. This provides a greater level of trustworthiness to the certificate and is one of the main reasons why X.509v3 has become so popular /14/.

An X.509 certificate is a data structure used to securely bind a public key to an identity. It is the most widely used digital certificate format and is frequently used in a variety of security protocols, such as TLS (Transport Layer Security). It contains information about the subject (the entity being identified), such as the name of the entity, its serial number, and the digital signature of the issuing authority. A certificate may be saved to a file or transmitted via a network, and also as a part of a secure network protocol, such as TLS. An X.509 certificate binds an identity to a public key using a digital signature /20/.



```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: ... hex bytes ...
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, O = Let's Encrypt, CN = R3
        Validity
            Not Before: Mar  4 12:43:52 2022 GMT
            Not After : Jun  2 12:43:51 2022 GMT
        Subject: CN = www.openssl.org
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus: ... hex bytes ...
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Server Authentication,
                TLS Web Client Authentication
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Key Identifier:
                ... hex bytes ...
            X509v3 Authority Key Identifier:
                ... hex bytes ...
            ... more X509v3 extensions ...
```

Figure 3. An X.509 certificate /20/.

An X.509X certificate includes the following fields:

- *X.509 version:* version 3.
- *Serial number:* Unique number signed by the issuer.
- *Signature algorithm:* Denotes hash function and the digital signature algorithm used.
- *Issuer:* Entity that signed certificate in DN format.
- *Validity:* Timestamp fields define the certificate's validity.
- *Subject:* Entity certificate identifies in DN format.
- *The certificate's public key:* Important for identity verification, and supports RSA, DSA, ECDSA, and EdDSA. Note: X.509 certificate does not contain a private key, only a public key.
- *X509v3 extensions:* Optional additional information.
- *Certificate signature:* Produced by the issuer /20/.

There are several steps in the X.509 certificate generation process:

- The private and public keys for the certificate are generated by the applicant (future certificate owner).
- Creating a Certificate Signing Request is done by the applicant (CSR). The CSR includes the subject, the upcoming certificate's public key, the applicant's requested X509v3 extensions, and the CSR signature. The private key for the certificate signs the CSR.
- A CA receives the CSR from the applicant for signing.
- The applicant's identity is verified by the CA.
- Based on data from the CSR, the CA creates a certificate. The CA additionally includes the issuer, validity fields, and X509v3 extensions in the certificate.
- Finally, the certificate is signed by the CA. The applicant receives the certificate back from the CA and is now the certificate's owner or holder.

3.2 Quantum Computing

The current Information Age and all of its revolutionary digital advancements, including personal computing, Internet communication, smartphones, machine learning, and the knowledge economy in general, were ushered in by classical computers. Bits are the basic unit of data encoding and manipulation in traditional computers. To switch or amplify electrical signals, today's general-purpose machines employ billions of transistors and semiconductor components. A traditional bit, like the power button on a preferred electronic device, can only ever be in one of two states at once: 0 or 1. For this reason, traditional information processing is described as binary.

Quantum computing is an emerging field of computing that offers a fundamentally unique and powerful approach to processing information and calculating solutions to problems. Quantum computers are capable of taking advantage of the principles of quantum mechanics to operate in a seemingly infinite number of states simultaneously, exponentially increasing their computational power compared to classical computers which can operate in only one state at any given moment. Therefore, scientists believe that these quantum computers can deliver exponential speedups and solve problems that are intractable for traditional computers. While the technology is still in its early stages, the potential uses for quantum computing are vast and hold promise for revolutionizing many areas of research, such as artificial intelligence, cryptography, materials science, and more /21/.

3.2.1 Quantum Computers

Early in the 1980s, scientists learned that quantum mechanics presents fresh possibilities for data processing. The creation of a proven secure encryption key is made possible by the non-classical aspects of quantum mechanics, as shown by Charles Bennett and Gilles Brassard. The entanglement property of particles is linked to a quantum phenomenon that Richard Feynman and Yuri Manin discovered cannot be replicated by the so-called Touring's machine. This led to the question of whether the phenomenon could be applied to generally speed up calculations. The principles of quantum mechanics are the foundation of quantum computing. A quantum computer is a physical system that uses quantum computation /22/.

A quantum computer is a device that receives data as input and performs operations on said data through a process that is only explicable through the principles of quantum physics /23/.

There are currently two primary categories of quantum computers: Quantum Circuit Computers and Quantum Adiabatic Computers. Quantum Circuit Computers are made of a network of quantum gates that transform an initial guess of the solution to a computational task into one that solves the problem using quantum mechanics. In Quantum Adiabatic Computers, the energy of a particular configuration of subatomic particles is used in these computers to represent the computational task. The solution is then reached by annealing, or gradually lowering the energy. Both are computationally equal. Quantum algorithms designed for one type of hardware can be transformed to execute on the other in a comparable time /24/.

Quantum computers use the properties of subatomic particles such as electrons, ions, or photons to process information. Information is stored in quantum registers, which consist of qubits, or quantum bits. Qubits are not limited to the binary states of 0 and 1 but can exist in a superposition of states until measured. This is similar to Schrödinger's cat, which can be both dead and alive until observed. When a qubit is measured, it collapses into a classical bit with a binary state.

Entanglement is another property of quantum physics where particles become connected in such a way that they cannot be described independently, even over great distances. This is in contrast to classical bits, which are independent of each other. In quantum computing, entangled qubits can fall into a shared quantum state, and manipulating one qubit can affect the whole system.

Interference is a mathematical description of qubits in quantum mechanics, represented by the wave function. The wave functions of entangled qubits can interfere constructively or destructively, increasing or decreasing the probability of obtaining the correct solution when the quantum computer is measured. Quantum algorithms are designed to choreograph this interference to increase the probability of obtaining useful measurement states.

One of the tricks used in quantum computing is called inversion about the mean, where finding the optimal number of iterations through the circuit can increase the probability of identifying the correct solution when the measurement is taken. Theoretically, a quantum computer with enough qubits and free from decoherence and noise could possess immense processing power, capable of examining more possibilities than the number of atoms in the observable universe /21/.

3.2.2 Quantum Algorithms

Algorithms are utilized in computation and data processing for modeling and solving real-life situations through the use of heuristics and methods. Generally, an algorithm is comprised of a set of instructions that, when executed, achieve a specific task. Quantum algorithms are particularly useful for providing solutions to complex problems that are unable to be solved by classical computers.

To further facilitate the development of quantum software, quantum programming languages have been created to enable the implementation of quantum algorithms. Generally, these algorithms consist of steps such as encoding quantum data into quantum bits, the application of a unitary quantum gate chain operating on the quantum bits, and the termination of the algorithm following the measurement of the quantum bits. It is important to note that a unitary quantum gate is analogous to a unitary matrix, which operates on the quantum bits, and that the inverse of a unitary matrix is equal to its conjugate transpose /25/.

The utilization of quantum algorithms for obtaining speedups in solving problems that necessitate searching for inputs to a function is of great importance. These functions may be obfuscated, such as hash functions, or computationally difficult to evaluate, which is common in the study of mathematical problems. Consequently, the utilization of quantum computers for such problems necessitates an understanding of how to program and provide input to quantum algorithms. A quantum program is an implementation of a quantum algorithm, which consists of a classical program that sends instructions to a quantum device to prepare a particular state or measurement result /23/.

Quantum algorithms are a series of mathematical steps that, when executed on a quantum device, will yield a specific result. With the advent of a functional quantum computer, researchers can take a problem that is addressed by a quantum algorithm, apply the algorithm, and observe the results. Most quantum algorithms are deemed revolutionary due to the substantial speed increases they provide when compared to traditional computers, as well as the types of complex problems they can solve. Ultimately, many modern cryptographic techniques can be broken by a combination of quantum properties, computers, and algorithms /26/.

3.2.2.1 Shor's Algorithm

Peter Shor is widely recognized as a figurehead of quantum computing and the breaking of traditional asymmetric cryptography in the modern era. Shor's algorithm is believed to have provided at least an exponential improvement, and potentially a polynomial time improvement, for the factorization of large primes. Utilizing quantum computers with a sufficient number of stable qubits, Shor's algorithm is capable of factoring prime number equations of considerable magnitude in a matter of seconds to minutes. Through the implementation of an equation that takes a random guess at one of the prime numbers and subsequently turns it into a much more accurate guess, Shor's algorithm can significantly reduce the number of guesses needed, in comparison to a classical brute-force method, by exploiting the mathematical relationship between the two prime numbers involved /27/.

Far from being merely a mathematical curiosity, the ability to quickly factorize large numbers is beneficial in cracking the RSA public-key cryptosystem /28/.

The RSA Algorithm, which is employed to encrypt a large portion of today's data, is based on the factorization of prime numbers. Examples of information that use this algorithm as part of their security infrastructure include bank records, passwords, and health records. The popularity of this algorithm is that it would take a significant amount of time to determine the two factored prime numbers of a product, particularly when the product is of considerable size. Shor's Algorithm has gained recognition for its capability to quickly factor out integers, particularly its capacity to solve period-finding problems in polynomial time /29/.

3.2.2.2 Grover's Algorithm

Search algorithms are distinguished by their ability to be employed by various algorithms to locate information, whether in a data repository or a list of values such as features in an image. The advantage of quantum computing lies in the potential to expedite the search process. Grover's algorithm utilizes a well-known technique that allows the use of interference to amplify certain states in our quantum circuit in a manner that will amplify the amplitude of the value being sought and diminish those that are not /29/.

Grover's Search Algorithm is a quantum variant of the searching algorithm that is utilized to achieve an unordered search, which entails locating one or multiple elements in a database or array. This algorithm demonstrates the capabilities of quantum computing by minimizing the number of operations required to complete the search process, as compared to its classical search versions. Grover's

Algorithm provides a boost in the search process and employs the technique of amplitude amplification for the item that is being sought /30/.

This is an unstructured quantum search algorithm developed by Lov Grover which is capable of finding an input with a high probability using a black box function. Grover can find an item in N to the power of 1/2 steps as opposed to a classical average of N/2 steps /31/.

Grover's algorithm can solve a phone book search on the order of the square root of the number of phone book numbers. For example, if a phone book contains 100 million phone numbers, Grover's algorithm can find a number with 10,000 steps. This quantum algorithm utilizes quantum mechanics-based methods to accelerate the search /25/.

Grover's search algorithm can be divided into two main components: Grover's oracle and the Grover diffusion operator /29/. Grover's oracle can be conceptualized as a black box. The Grover diffusion operator is responsible for the process of amplitude amplification, wherein the amplitude of a designated item is augmented while the amplitude of other elements is reduced. The process of amplitude amplification can be interpreted as augmenting the amplitude of the marked element, thus making it more likely to be identified during measurement /30/.

3.2.3 Algorithms Break

It has been demonstrated that a quantum computer-based algorithm can break the symmetric key cryptographic algorithm by a factor of the square root of the size of the key. For instance, to find an image of a 256-bit hash function, a quantum algorithm will take only 2128 times. In addition, Shor's algorithm assists in the quantum factorization of large numbers, necessitating that security systems be quantum-resistant. Furthermore, Elliptic Curve Cryptography has been demonstrated to augment the cracking process by 21 percent, as it is based on multiplying polynomials and adding random noise.

Financial institutions must begin to invest in post-quantum cryptography to ensure the security of blockchain-based solutions. Examples of existing cryptographic systems that will be rendered ineffective by quantum algorithms and computing power include DES (Data Encryption Standard), Triple DES, AES (Advanced Encryption Standard), RSA, Merkle hash-tree signatures, Merkle–Hellman knapsack encryption, Buchmann–Williams class-group encryption, ECDSA, HFEv-, and others. In particular, Shor's algorithm is a quantum algorithm that poses a threat to RSA, DSA, and ECDSA cryptographic algorithms /25/.

Shor's Algorithm has rendered RSA, ECC, and Diffie-Hellman vulnerable in the quantum computing world, as the attacker can use the algorithm to resolve a hard-mathematical problem in polynomial time and reconstruct the private key with ease. Grover's Algorithm, which finds with high probability the unique input to a function that produces a particular output value, further aids the brute attack for key searching by reducing the effective key strength by half.

Symmetric cryptography is not greatly impacted, as AES is presumed to be quantum-safe, with Grover's Algorithm only aiding the brute force search. To counter this, the key size must be doubled, from AES-128 to AES-256. Asymmetric cryptography, however, is heavily impacted, as RSA/ECC/DH and their derivatives are not quantum-safe due to Shor's Algorithm. Therefore, a replacement must be found, with potential options including hash-based digital signature, lattice-based cryptography, code-based cryptography, and Multivariate Public Key Cryptography /32/.

The security of Diffie-Hellman is contingent upon the difficulty of resolving the discrete logarithm problem. Shor's algorithm can also be utilized to solve discrete logarithms. This implies that, with a quantum computer executing Shor's algorithm, Diffie-Hellman can be compromised in a reasonable timeframe /33/.

It has been determined that hashing algorithms, such as SHA-1 and MD5, are vulnerable to quantum attacks since the Grover Quantum Search Algorithm can be utilized to obtain hash collisions at a much faster rate. As a result, Grover's Quantum Algorithm has the potential to drastically reduce the security level of symmetric cryptographic systems /34/.

Shor's solution for determining the order has been demonstrated to be far more efficient than the Grover algorithm in terms of breaking RSA. The most straightforward solution is to seek the prime factors of N through the utilization of combined classical-quantum algorithms. Furthermore, it is worth noting that public key cryptosystems based on discrete logarithms are also highly vulnerable as the discrete logarithm problem can be linked to period finding, which is a common practice in order finding through factorization /35/.

3.3 Post Quantum Cryptography

The utilization of immense computing power could be revolutionary in the field of cryptography. Generally, brute force attacks, which involve attempting all potential key values, are unfeasible. This was demonstrated in 1997 and 1998 when the DES was broken due to its 56-bit key length, which was deemed satisfactory in 1977 but was surpassed by the continuous acceleration of computing technology. The success of the brute force attack was only made possible by the rapid development of hardware. In response to this breach, the AES was developed, featuring keys of 128 bits or more, thus rendering brute force attacks virtually impossible at that time /36/.

The sufficiently large quantum computer implementing Shor’s algorithm would be capable of breaking the public key algorithms currently in use. As a result, the world must begin the process of transitioning to replacement algorithms based on mathematical problems that are not solvable by classical or quantum computers in a reasonable amount of time.

These algorithms are also referred to as quantum-resistant, quantum-safe, or post-quantum cryptography (PQC). It is therefore essential to begin the migration away from existing public key algorithms before the potential existence of a sufficiently large quantum computer /37/.

Post-quantum cryptography is a field of study that focuses on the development of public-key algorithms that are resistant to being broken by quantum computers. These algorithms would be quantum-safe and could be used to replace RSA and elliptic curve-based algorithms in a future where quantum computers can easily break 4096-bit RSA moduli /38/.

PQC is an area of research in which new quantum-resistant (primarily) public-key algorithms are developed for a wide range of devices and applications. Quantum Cryptography, on the other hand, typically refers to the utilization of Quantum Key Distribution (QKD) in combination with modern encryption techniques /39/.

It is not advisable to design algorithms that are based on a hard problem that is known to be efficiently solvable by Shor's algorithm, as this would effectively nullify the hardness of factoring and discrete logarithm problems. Symmetric algorithms, such as block ciphers and hash functions, would only experience a reduction in their theoretical security by half in the event of a quantum computer, as opposed to RSA which would be severely compromised. As such, these algorithms could potentially form the foundation of a post-quantum scheme /38/.

Classical cryptographic algorithms are based on either integer factorization (for example, RSA) or discrete logarithms (for instance, ElGamal). These approaches provide robust security within the current computing environment; however, their future may be uncertain due to the discovery of algorithms such as Shor's algorithm which could potentially be used to break contemporary encryption schemes. As powerful quantum computers become available, measures must be taken to protect against them. The first step is to increase the key sizes used in modern encryption algorithms from, for example, 128 to 256 bits (and beyond).

The next step is to transition to full-scale encryption which is intrinsically quantum-resistant. Most proposed PQC protocols employ an asymmetric approach. There are several major implementations of PQC which are currently being researched /39/.

In the following sections, we shall analyze four of the most renowned families of schemes: hash-based, lattice-based, code-based, and multivariate cryptography.

3.3.1 Hash-based Cryptography

Hash-based cryptography is based on cryptographic hashes, as the nomenclature implies, and is typically utilized in digital signature schemes (as opposed to encryption). A hash is a one-way function that converts the hashed content into a representative set of bits (commonly referred to as a hash, hash result, signature, or message digest) that is unique for each unique content.

It has been established that hash-based cryptography is quantum-resistant, as hashes are not vulnerable to Shor's algorithm. However, they are susceptible to Grover's algorithm, which, when implemented on quantum computers, offers a square root improvement in comparison to binary computers for certain tasks, such as cracking hashes. This, in turn, reduces the strength of hash-based cryptography by half. To counter this, it is necessary to double the key size of the hash, thereby nullifying the benefits of Grover's algorithm and quantum computing /26/.

Hash-based signatures are often characterized by a variety of parameters that can affect factors such as security, signature size, and computational complexity /37/.

Hash-based cryptography is vulnerable to the potential of repeating a one-time key for two different inputs, which could grant attackers access to the private key. For this reason, developers of hash and hash-based cryptography take extensive measures to avoid the repetition of one-time keys. A common method to prevent such repetition is to make the hash stateful, as opposed to stateless. A stateful hash keeps a record of all one-time secret keys used and ensures that none of them are reused. Most traditional signature-based hashes are stateful. If a repeated key is detected, the algorithm is re-run or a different part of the longer keystream is chosen to generate a unique one-time key /26/.

Stateful schemes are significantly more efficient than their stateless counterparts; however, stateful implementations must be exceptionally diligent in maintaining an accurate state. It is possible for an implementation to misplace the count of items that have been signed if a process is interrupted and restarted, or if multiple services are utilizing the same public key /37/.

3.3.1.1 One-time Signature with Lamport

On October 18, 1979, Leslie Lamport introduced his concept of one-time signatures (OTS): key pairs that can only be utilized for signing once. Most signature schemes are partially dependent on the security of one-way functions (commonly hash functions) for their security proofs. The appeal of Lamport's scheme is that his signature is exclusively reliant on the security of such one-way functions.

A Lamport signature is a type of OTS based solely on hash functions. To generate a key pair that is capable of signing a bit, two random numbers must be generated, which will serve as the private key. Subsequently, each of these numbers must be hashed individually to produce the two digests of the public key. To sign a bit that is set to 0, the first random number must be revealed; conversely, to sign a bit that is set to 1, the second random number must be revealed /17/.

The individual who generated the keys now desires to sign a message. The initial step is to hash the message so that a 256-bit hash or digest is produced. Subsequently, for each bit in the hash, one number from the pairs that constitute the private key is chosen. This usually implies that if the bit is 0, the first number is utilized; if it is 1, the second number is utilized. This will generate a sequence of 256 numbers, each one 256 bits long. Consequently, the signature is 64 kilobytes.

What distinguishes this from other signature algorithms is that the signature is utilized one time only. The sender will then discard the private keys used. To authenticate the signature, the recipient will also hash the message to acquire a 256-bit hash/digest. Then, the recipient utilizes the bits in that hash/digest to select 256 of the hashes in the sender's public key. The same selection method that the sender utilized is employed (i.e., if it is 0, use the first hash; if it is 1, use the second). Now, the recipient will hash each of the 256 random numbers in the sender's signature. If all of these precisely match the 256 hashes selected from the sender's public key, the signature is confirmed. If any do not match, the signature is rejected /33/.

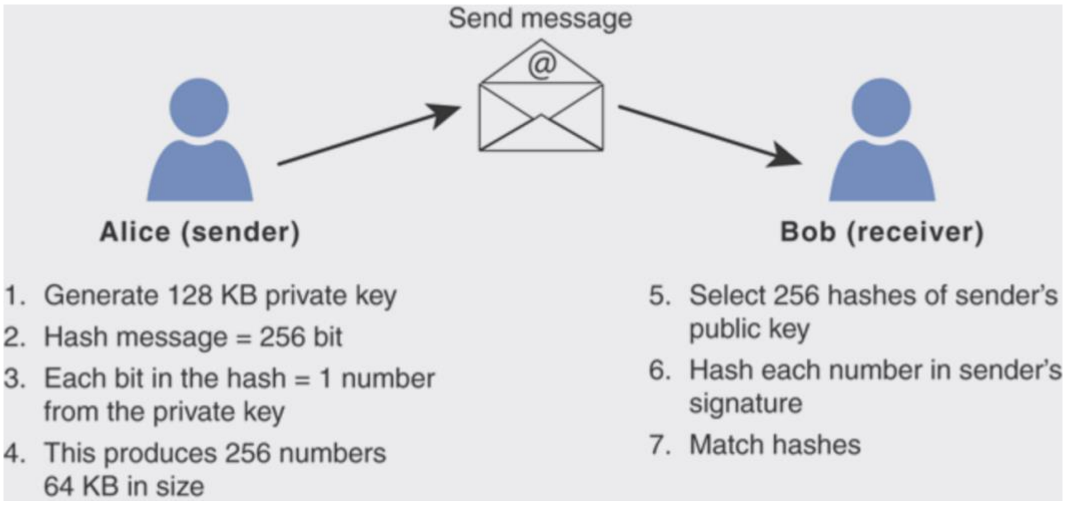

Send message
@
Alice (sender)
Bob (receiver)
1. Generate 128 KB private key
2. Hash message = 256 bit
3. Each bit in the hash = 1 number from the private key
4. This produces 256 numbers 64 KB in size
5. Select 256 hashes of sender's public key
6. Hash each number in sender's signature
7. Match hashes

Figure 4. Lamport's Steps /33/.

3.3.1.2 Many Times Signature with XMSS

The Extended Merkle Signature Scheme (XMSS) was introduced in 2011, and it shares many similarities with the Merkel Signature Scheme. XMSS is a more secure version of the aforementioned scheme, and it often produces smaller signature sizes as well. XMSS MT is a more powerful, multitree variant of this scheme and it is capable of signing an infinite number of messages; compared to XMSS, this variant typically produces signatures faster, albeit at the cost of a larger file size /39/.

The XMSS, as standardized in RFC 8391, was developed to improve upon Merkle's signature scheme by introducing several optimizations. One such optimization is the reduction of the size of the private key required to sign N messages. Rather than having to generate N OTS private keys and store the public key as a root hash, XMSS allows for the deterministic generation of OTS keys using a seed and the leaf position in the tree. This means that only the seed needs to be stored as the private key, and any OTS key pair can be quickly regenerated from its position in the tree and the seed. To keep track of which leaf/OTS was used last, the private key also contains a counter which is incremented each time it is used for signing.

The XMSSMT Stateful Hash-Based Signature Scheme utilizes multiple trees to enhance the number of signatures supported by the Scheme, while simultaneously diminishing the workload during the generation of keys and signing of messages. Each tree is generated in a deterministic manner only when it is employed in the path leading to the ultimate leaf that contains the OTS utilized to sign a message /17/.

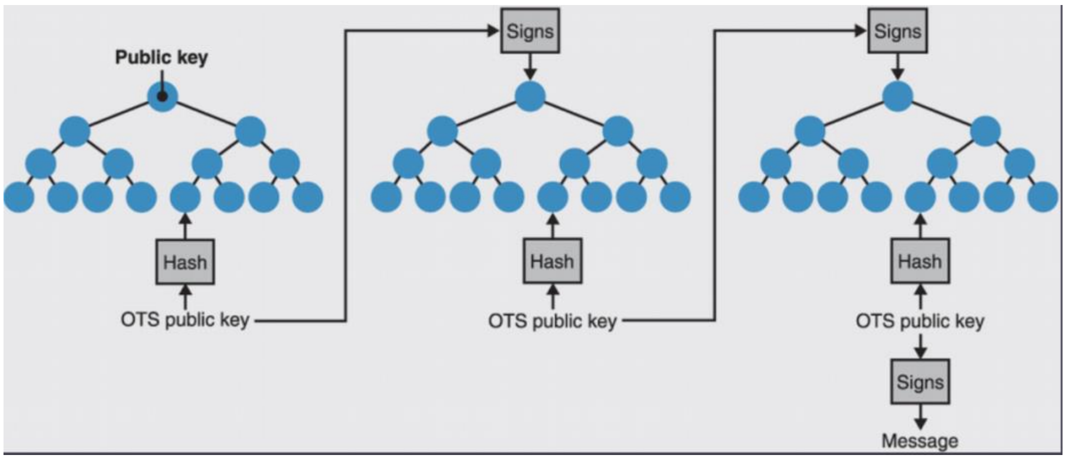


Figure 5. XMSS signature scheme /17/.

3.3.2 Lattice-based Cryptography

Lattice-based cryptography is regarded as being very reliable because it illustrates the learning with errors (LWE) idea. LWE encrypts datasets and introduces a small, controlled error that renders them impossible to decrypt (at least within a reasonable amount of time). A significant component of machine learning is this method. Oded Regev, a theoretical computer scientist who later won the 2018 Gödel Prize, first proposed the LWE problem in 2005.

A lattice is a concept in mathematics related to set theory, which is characterized by a repeating pattern of points. It is also known as a coordinate vector in an n-dimensional space, where 'n' stands for the number of vectors in this space. An example of a lattice in the real world is a crystal. This type of lattice is defined by its number of vectors in a vector space /39/.

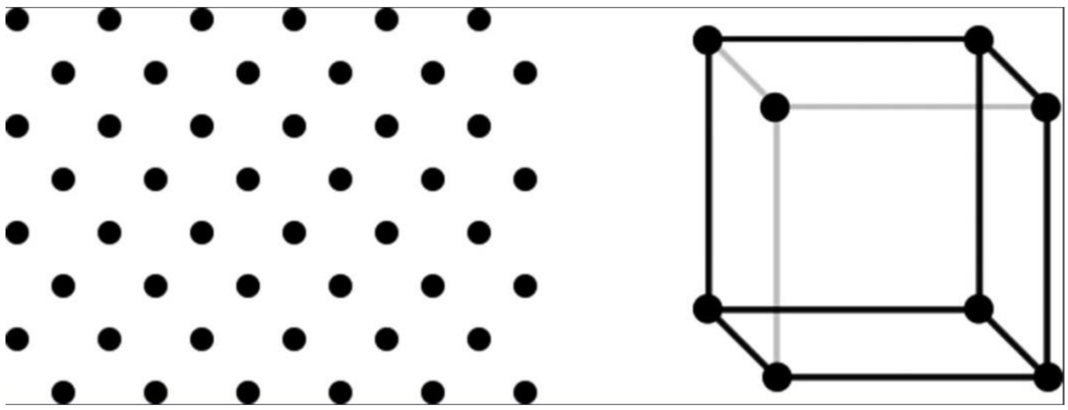


Figure 6. Two-dimension versus three-dimension lattice /39/.

Lattice-based cryptography may need slightly bigger key sizes than other encryption techniques, yet this is not necessarily the case when it comes to many of the lattice-based proposals submitted to the National Institute of Standards and Technology (NIST), such as CRYSTAL-Kyber, LAC, NewHope, NTRU, NTRUPrime, Round5, SABER, and ThreeBears. FRODO-KEM is the only one with an appreciably larger key size, though some code-based ciphers have much bigger ones.

The first lattice-based cipher was NTRU, introduced in 1998, followed by several ciphers based on LWE and RLWE math problems. Today, lattice-based ciphers are the most commonly submitted type of post-quantum crypto submitted to NIST. Additionally, Craig Gentry used lattice-based cryptography in his 2009 dissertation to create the first fully homomorphic real-world encryption system, allowing a third party to correctly tamper with encrypted data without first decrypting it /26/.

Two NIST finalist schemes are closely related: CRYSTALS-Kyber and CRYSTALS-Dilithium, which are candidates from the same research team, and both are based on the LWE problem. Kyber is a public key cryptographic primitive that can be used as a key exchange primitive, which will explain in this section. Dilithium is a signature scheme that will explain in the next section. Also note that since these algorithms are still in flux, will only write about the ideas and intuitions behind both schemes /17/.

3.3.3 Code-based Cryptography

Error-correcting codes are a component of code-based cryptographic schemes. Error-correcting codes were initially developed to address the issue of communication or storage systems that could have a small number of bits that were corrupted. A k-bit string called a codeword is expanded into an n-bit string by an error-correcting code. An n-bit codeword with up to t errors (flipped bits) can be given to an inverse function, which can then identify the flipped bits and restore the original k-bit string. The more redundant bits that are added to a string, the more mistakes the error-correcting code can fix. If there are too many errors, the error-correcting code might report that the error cannot be recovered or it might recover the wrong data /37/.

A Russian/Soviet mathematician named Valery Denisovich Goppa connected geometric shapes and combinations to error-correcting codes in the 1970s and 1980s. The Goppa codes are now well-known and were adopted by cryptographers. The majority of code-based ciphers in general, including McEliece, one of the most popular code-based ciphers, are built on binary Goppa codes. Code-based ciphers came in at number two in terms of popularity among the asymmetric encryption cipher types submitted to NIST. The code-based ciphers BIKE, Classic McEliece, HQC, LEDAcrypt, NTS-KEM, Rollo, and RQC are among those that have been submitted to the NIST PQC competition and qualified for the second round /26/.

The first application of the code-based approach was the asymmetric McEliece cryptosystem, which dates back to 1978. This method employs the binary Goppa error correction code. The foundation of a Goppa code is modular arithmetic, in which numbers that reach a predetermined target value return to zero once more. The twelve-hour clock is the most prevalent example of this in everyday life. Aside from the McEliece, many cryptographic algorithms, such as RSA and AES, make use of modular arithmetic in different ways.

The Goppa codes are extremely reliable, but they frequently require very large public keys. Possibly more work needs to be done on this aspect of the McEliece plan. Although other, lighter codes have been suggested to take the place of the trapdoor functions offered by the Goppa codes, none have shown to be as resistant to cryptanalysis.

An updated version of the McEliece cryptosystem is the Niederreiter cryptosystem. The method, created in 1986 by Harald Niederreiter, is regarded as having an equivalent level of security to that of its forerunner. Niederreiter, however, encrypts data more quickly than the McEliece cryptosystem /39/.

3.3.4 Multivariate Cryptography

Multivariate polynomial-based asymmetric cryptographic primitives are referred to as multivariate cryptography. Post-quantum cryptography is seen as having been very robustly interpreted by it. By 2020, the majority of this technology may have reached full maturity, but it's still possible that attack vectors that have not yet been identified pose a threat /39/.

Multivariate is an abbreviation for "multiple variables." Multivariate math equations, like $x + y + z = n$, are used to create the cryptographic primitives in asymmetric encryption and signature schemes known as multivariate cryptography. Multivariate quadratic (MQ) polynomial equation cryptography is another name for cryptography based on multivariate polynomial mathematics.

This means that at least one of the variables has been raised to the second power (for instance, x^2 + y + z = n). Correctly designed multivariate cryptography is not protected by large primes and cannot be cracked in polynomial time. They are therefore regarded as quantum-resistant. They are a strong performance candidate for hardware implementations like application-specific integrated circuits (ASIC) and field-programmable gate arrays (FPGA) due to their inherent qualities.

HFE, Gui, Balanced Oil & Vinegar, Unbalanced Oil & Vinegar, and Tame Transformation Signature are examples of multivariate cryptography. The multivariate digital signature protocols GeMSS, LUOV, MQDSS, and Rainbow have all been submitted. Unbalanced Oil & Vinegar is implemented in several layers as Rainbow /26/.

In 1996, cryptographer Jacques Patarin unveiled the Hidden Field Equations (HFE) public-key cryptosystem. This is still a widely used type of multivariate cryptography, and it may have been the first. The foundation of HFE is the complexity of the problems posed by a system of quadratic equations. To obscure the relationship between private and public keys, HFE is based on polynomials over finite fields of various sizes.

Based on his earlier work, Patarin created the signature scheme known as Unbalanced Oil and Vinegar (UOV). UOV is based on how difficult it is to distinguish between two different types of variables, known as "oil" and "vinegar." The phrase "unbalanced" refers to the use of this scheme of varying ratios of the two aforementioned variables. UOV requires the solution of a minimal quadratic equation system to generate and receive signatures. This plan is thought to be quantum-resistant. Since UOV is primarily based on addition and multiplication, it is also very easy to implement in even the most basic hardware devices. This scheme typically has rather long key lengths, resulting in fairly large key file sizes. An earlier version of UOV, known as Balanced Oil and Vinegar, was broken in 1998, calling for this updated version /40/.

The McEliece cryptosystem has been updated as the Niederreiter cryptosystem. The method, created in 1986 by Harald Niederreiter, is regarded as being on par with its predecessor in terms of overall security. When encrypting data, Niederreiter is quicker than the McEliece cryptosystem /41/.

4 APPROACH AND IMPLEMENTATION

4.1 Review of Available PKI Software

PKI software provides the requisite tools to generate, disseminate, and supervise digital certificates and public keys, as well as to authenticate digital signatures and ensure secure communication between parties.

The fundamental operation of PKI software involves the utilization of a CA which issues digital certificates to entities such as individuals, devices, or organizations. These certificates incorporate the entity's public key and other pertinent data and are signed by the CA to guarantee their legitimacy.

When an individual desires to communicate securely with another entity, they must first acquire the other entity's public key from their digital certificate. Subsequently, they utilize this public key to encrypt the message before transmitting it. Upon receipt, the other entity utilizes its private key to decrypt the message.

Furthermore, PKI software provides additional functions such as certificate revocation, which permits the revocation of compromised or invalid certificates, and certificate validation, which verifies the authenticity and validity of digital certificates.

There are several PKI software solutions available on the internet that support post-quantum algorithms, a majority of them are either closed-source or commercial PKI solutions. The present chapter entails an extensive investigation and comparative analysis of various open-source PKI software solutions to offer a comprehensive overview of the current state of development in the field of PKI software. The primary objective is to select the most suitable PKI software that can seamlessly integrate with post-quantum algorithms, or serve as crucial information for future research endeavors.

4.1.1 Candidates

OpenXPKI is designed to function as an online Registration Authority (RA) and CA for X.509 version 3 certificates, but its flexibility allows it to serve a wide range of cryptographic key management use cases. OpenXPKI has a mature and stable code base and is continually evolving to meet the needs of its growing user base. Furthermore, the development team actively supports multiple professional installations that have been in operation since 2009, hosting multiple logical CAs and hundreds of thousands of active certificates /42/.

OpenXPKI is a PKI software solution that is written in the Perl programming language and is licensed under the Apache 2.0 license. Notably, OpenXPKI also offers a Graphical User Interface (GUI) for ease of use. There are 33 active developers of the software who continue to actively maintain and update the software's repository, with the latest commit being made in February 2023 upon the release of a new version 3.24. This highlights the ongoing development of the software and commitment to providing users with an up-to-date and reliable PKI solution /43/.

OpenXPKI supports Simple Certificate Enrollment Protocol (SCEP) and Enrollment over Secure Transport (EST) and allows for the use of Hardware Security Modules (HSMs). It also allows for easy customization of workflows and can run multiple separate CAs with a single installation. Additionally, it can issue certificates with publicly trusted CAs and is based on OpenSSL and Perl, making it compatible with most *nix platforms /42/.

EJBCA is one of the most extensively utilized PKI platforms in the world, providing comprehensive certificate management, registration, enrollment, and certificate validation capabilities. It encompasses all essential PKI components, including CA, RA, and Validation Authority (VA).

EJBCA provides flexibility and scalability to accommodate nearly every PKI use case, including DevOps, Internet of Things (IoT), Industrial IoT, Enterprise PKI, and more. Additionally, EJBCA offers effortless integrations into third-party systems for full automation and easy operations. It also features multitenancy, which enables the hosting of multiple CAs and PKIs in a single server installation /44/.

EJBCA is a Java-based software that operates under the LGPL 2.1 license and incorporates a graphical user interface. The project currently benefits from the contributions of 14 active contributors, with the most recent commit having taken place four months ago which is in February 2023. The latest iteration of EJBCA is version 7.12.

EJBCA provides a comprehensive turnkey PKI solution that offers pre-packaged components, protocols, and software for quick and easy deployment. It offers various deployment models, including turnkey software and hardware appliances, cloud-based solutions, and hybrid models, enabling flexible and scalable PKI implementation. Additionally, it supports SaaS-based PKI deployment /45/.

For EJBCA, public key algorithms, including RSA, DSA, and ECDSA (Elliptic Curve Digital Signature Algorithm), are implemented to facilitate secure key operations and digital signatures. Key exchange algorithms, such as Diffie-Hellman and Elliptic-curve Diffie–Hellman (ECDH) are employed during the TLS handshake process to establish secure sessions. Hash functions, utilized for generating message digests and certificate signatures, encompass MD5, SHA-1, and the SHA-2 family, encompassing SHA-256, SHA-384, and SHA-512 while the use of MD5 and SHA-1 is regarded as a legacy, and their vulnerability to attacks is recognized.

The Dogtag Certificate System is an open-source CA designed for enterprise use. This full-featured system has been rigorously tested through real-world deployments, resulting in a hardened solution capable of managing all aspects of the certificate lifecycle. The system capabilities include key archival, OCSP management, and smartcard integration, making it a comprehensive tool for certificate management in large-scale environments /46/.

The Dogtag PKI is a Java-based software system designed for PKI deployment. The system is licensed under the GPL 2.0 license and comes with GUI support. Currently, there are 44 active contributors involved in the development of this project, ensuring a constantly evolving and robust platform. The project has been actively maintained, with the latest commit made in May 2023, coinciding with the release of this thesis. The current version available is 11.0.0, highlighting the continued evolution of the system to meet the changing needs of enterprise-level PKI management /47/.

Dogtag is a comprehensive suite of technologies that facilitates the implementation of PKI on a large scale. It offers a wide range of features, such as certificate issuance, revocation, and retrieval, as well as CRL generation and publishing. Furthermore, Dogtag is compatible with various Certificate Profiles. It integrates the SCEP to simplify the certificate enrollment process. Additionally, Dogtag provides a Local Registration Authority (LRA) for organizational authentication and policy enforcement.

Dogtag also enables organizations to effectively manage their encryption keys, which includes Smartcard lifecycle management and Token profiles. Furthermore, Token enrollment, on-hold, key recovery, format, and face-to-face enrollment with the security officer workstation interface, ensure the highest levels of security during the enrollment process /46/.

The OpenCA PKI Project is a collaborative initiative aimed at creating a comprehensive, fully-featured, and open-source Certification Authority with robust capabilities, implementing widely-used protocols utilizing state-of-the-art cryptography globally. The project draws upon several open-source software projects, including OpenLDAP, OpenSSL, Apache Project, and Apache mod_ssl, to achieve its objectives.

The development of the project is structured around two primary objectives: rigorous analysis and refinement of the security framework, ensuring the selection of the optimal model for deployment in a CA, and the development of software tools to facilitate the efficient establishment and management of a CA /48/.

OpenCA PKI is developed using the Perl programming language and is licensed under the Apache license. The web-based management tool for OpenCA PKI, OpenCA-NG, offers a user-friendly interface for configuring and managing the certification authority. While the project's Github repository lists two contributors, it appears that the development of OpenCA PKI has been inactive for approximately nine years. Additionally, the latest version of the software, OpenCA PKI V1.5.0, was released in August 2013 /49/.

Step-CA is an internet-based Certificate Authority (CA) that facilitates the secure and automated management of X.509 and SSH certificates. Serving as the server-side counterpart to the Step CLI, Step-CA offers robust TLS security. The system features default algorithms and attributes that are designed to be both practical and secure, making it accessible to users who may not possess extensive expertise in security engineering.

Step-ca is a widely utilized tool among developer teams for a range of critical functions. It enables the generation of TLS certificates for private infrastructure via the Automated Certificate Management Environment (ACME) protocol. Step-ca facilitates automated TLS certificate renewal, which is essential for ensuring the continued secure operation of online systems. Step-ca allows for the integration of ACME support with legacy subordinate CAs, offering a seamless solution to a common challenge in modern security contexts and the issuance of short-lived Secure Shell (SSH) certificates through the use of OAuth OIDC single sign-on. Finally, Step-ca empowers teams to issue customized X.509 and SSH certificates, thereby providing them with greater flexibility in managing their security infrastructure /50/.

Step-ca is a command-line tool that is primarily used through a terminal interface and is implemented in the Go programming language under the Apache 2.0 license. It provides a solution for managing digital certificates, including X.509 and SSH certificates, and features a user-friendly web-based interface for the efficient management and monitoring of certificates issued by the CA. The project has a substantial following, with 65 active contributors to the codebase, reflecting a vibrant and dynamic community committed to improving the PKI. Notably, the most recent commit to the codebase was made in May 2023, indicating that the project is highly active and focused on delivering cutting-edge PKI solutions. The latest version, Step CA v0.24.2, was also released in May 2023, providing users with the most up-to-date features and functionality /51/.

CFSSL is an open-source toolkit that caters to all aspects of TLS and Secure Sockets Layer (SSL). CloudFlare, a leading internet security company, has integrated CFSSL into its infrastructure for aggregating TLS/SSL certificate chains and for maintaining an internal Certificate Authority infrastructure. One of CFSSL's key advantages is its ability to balance performance, security, and compatibility, which makes it an ideal tool for website owners and large software-as-a-service companies alike. CFSSL offers a JSON API web service and a convenient command-line interface, which can be utilized in diverse settings /52/.

CFSSL is a program that is authored in the popular programming language, Golang. The tool requires version 1.16 or newer of Go to compile successfully and is distributed under the permissive BSD-2-Clause open-source license. CFSSL offers both command-line functionality and HTTP API server capabilities, making it a versatile tool for signing, verifying, and bundling TLS certificates. A large and diverse community of contributors has contributed to the project, resulting in a robust and reliable software package. The most recent commit to the project was made in May 2023, demonstrating the continued commitment of the project's maintainers to ensure its ongoing development. The current stable release version of Cfssl is v1.6.4, which provides a stable and reliable toolset for developers /53/.

At present, Cloudflare is running CFSSL on highly secure and tightly controlled computing systems. However, to further bolster the security of our infrastructure, Cloudflare plans to incorporate the software with cost-effective Trusted Platform Modules (TPMs) to offer a more advanced level of hardware security. This strategy is intended to protect private keys from unauthorized access, even in the improbable event of a breach /52/.

The public key algorithms employed by CFSSL include RSA and DSA, which are utilized for cryptographic key operations and digital signatures. CFSSL also incorporates key exchange algorithms, such as Diffie-Hellman and its elliptic curve variants ECDH, which are crucial components of the TLS handshake process for secure session establishment. In terms of hash functions, CFSSL employs the Message Digest 5 (MD5) and SHA-1, which play vital roles in generating message digests and creating certificate signatures.

X Certificate and Key Management is a powerful software application developed to facilitate the effective creation and management of X.509 certificates, certificate requests, RSA, DSA and EC private keys, Smartcards, and CRLs. This software provides the comprehensive functionality necessary for a Certificate Authority, enabling the signing of sub-CAs in a recursive manner with a clear display of the resulting certificate chains. Additionally, the software offers configurable templates for generating certificates or requests, which is especially advantageous for the smooth implementation of this application on a corporate level /54/.

XCA is a Certificate Authority software application developed using the C/C++ programming language and licensed under the BSD-3-Clause license. Its GUI facilitates the generation, signing, revocation of certificates, and the management of private keys and configuration of certificate authorities. The software has been enhanced by the collaborative efforts of 36 active developers. The most recent commit to the project was made approximately seven months ago, in November 2022. The latest version is XCA 2.4.0 /55/.

The software application offers a comprehensive set of features that enable users to create, manage, and export private keys, certificates, requests, or CRLs in various formats, including PEM, DER, PKCS#7, and PKCS#12. The certificates generated can be utilized for various certificate-based setups, such as IPsec, OpenVPN, and TLS. The application provides a convenient way to manage Smart-Cards via the PKCS#11 interface. Additionally, the software offers the ability to create Subject and/or Extension templates that make issuing similar certificates more straightforward, converting existing certificates or requests to templates as required. The x509v3 extensions are supported, allowing for a broad range of customization options while remaining user-friendly /54/.

In the context of PKI systems, XCA employs a range of encryption algorithms for different purposes. XCA utilizes traditional encryption algorithms such as RSA, DSA, ECDSA, Diffie-Hellman, ECDH, and hash functions such as MD5, SHA-1, and SHA-2 family (including SHA-256, SHA-384, and SHA-512) for public key operations and message digests. These algorithms play a vital role in ensuring the security and integrity of data within the XCA PKI environment, allowing for secure authentication, encryption, and verification processes.

Easy-RSA is a command-line interface (CLI) tool that is specifically designed to facilitate the management of PKI, using the widely recognized X.509 standard. With Easy-RSA, it is possible to establish and manage a PKI CA with ease. In essence, PKI is built around the principle of relying on a trusted authority to verify the authenticity of a remote peer. Therefore, Easy-RSA represents a reliable utility for ensuring secure and authenticated communication in a networked environment /56/.

Easy-RSA is a highly versatile utility that has been developed to run on a variety of different host systems, owing to its reliance on the platform-neutral POSIX shell. Moreover, the official Windows release is bundled with all the required programs for seamless integration with Easy-RSA. The philosophy software aims to reduce dependencies on external programs, with OpenSSL serving as the backend for crypto-related tasks. Although Easy-RSA does not offer a GUI by default, it provides a powerful command-line interface that requires manual input and terminal interaction. The project boasts active development by 60 contributors, with the latest commit having been created in May 2023, and the most recent release version being v3.1.2 in January 2023 /57/.

One of its notable advantages is the ability to manage multiple PKIs independently, with different configurations and storage directories. Easy-RSA supports several Subject Name formatting options, allowing for a cleaner setup for VPNs using commonName only. Easy-RSA employs a single backend that is compatible with various platforms, including Unix-alikes (BSD, Linux, etc.) and Windows, ensuring all platforms have access to its rich features. Its X.509 support includes advanced features such as CRL, CDP, and keyUsage/eKu attributes, with the option to add or modify features as needed. Easy-RSA offers flexible operation modes, with both interactive and automated (batch) modes of operation. The configuration is also flexible, with features enabled through command-line options, environment variables, a config file, or a combination of these. Built-in defaults enable Easy-RSA to be used without having to first edit a configuration file /56/.

Django-ca is a tool that facilitates the management of TLS certificate authorities and streamlines the process of issuing and revoking certificates. The tool is built on top of the highly secure cryptography framework and integrated with the powerful Django web framework. Notably, the entire system can be effectively administered through the command line interface, leveraging the manage.py commands /58/.

Django-ca is built upon Django, a full-stack framework widely recognized in the software development community. Licensed under the GPL-3.0 license, Django-ca offers a rich set of features for managing TLS certificate authorities. It is worth noting that Django-ca does not provide GUI.

The project boasts a vibrant development community, with a dedicated team of 10 contributors actively advancing its capabilities. As evidenced by the latest commit in May 2023, the project is continuously evolving and improving with regular updates. The most recent release, version 1.24.0, was launched in May 2023, further solidifying the commitment to ongoing development and enhancement of Django-ca /59/.

A key feature is the ability to quickly and easily set up a secure local CA, streamlining the process of issuing and managing certificates. Furthermore, the software provides for CRLs and OCSP to ensure the efficient and secure revocation of certificates when necessary. The application allows for certificate issuance through multiple mediums, including ACMEv2, command line, and web interface, offering users flexibility and ease of use. Management of the certificates can be done through the command line and/or Django's admin interface, providing a simple and efficient management process. To ensure the timely renewal of certificates, the software can be configured to send email notifications regarding certificates that are nearing expiration. The application uses Python 3.8+, Django 3.2+, and cryptography 37.0+, providing a stable and dependable platform for certificate management /58/.

Django-ca incorporates various encryption algorithms within its PKI system. Django-ca utilizes traditional encryption algorithms such as RSA, DSA, ECDSA, Diffie-Hellman, ECDH, and hash functions like MD5, SHA-1, and SHA-2 family (including SHA-256, SHA-384, and SHA-512) for public key operations and message digests. Furthermore, Django-ca also places notable emphasis on symmetric key algorithms, including AES, DES, 3DES, RC4, Blowfish, and Twofish.

The Open Quantum Safe (OQS) project is a collaborative effort aimed at advancing and exploring quantum-resistant cryptography. It consists of two main components: liboqs, an open-source C library that provides quantum-resistant cryptographic algorithms, and the integration of prototypes into various protocols and applications. Notably, the project is integrated with the widely utilized

OpenSSL library, allowing developers to incorporate quantum-resistant cryptography into their projects. The OQS project is a valuable resource for enterprises seeking to create their own post-quantum PKI. Its C-based source code makes it easy to integrate with future algorithms, guaranteeing adaptability as new cryptographic methods emerge. By utilizing widely used algorithms originally written in C, developers can easily incorporate novel cryptographic approaches into their systems. The project has an active community of developers, as evidenced by the release of 12 versions as of May 2023. Its repository contains a wide range of renowned and efficient post-quantum algorithms, including CRYSTAL-kyber, BIKE, FrodoKEM for key exchange, and Dilithium, Falcon, and SPHINCS+ for signature algorithms. While the OQS project does not provide a complete PKI solution on its own, it presents an appealing integration option due to its broad selection of post-quantum algorithms and compatibility with OpenSSL, a widely accepted library.

The Open Quantum Safe (OQS) project is a collaborative effort aimed at advancing and exploring quantum-resistant cryptography. It consists of two main components: liboqs, an open-source C library that provides quantum-resistant cryptographic algorithms, and the integration of prototypes into various protocols and applications. Notably, the project is integrated with the widely utilized OpenSSL library, allowing developers to incorporate quantum-resistant cryptography into their projects. The OQS project is a valuable resource for enterprises seeking to create their own post-quantum PKI. Its C-based source code makes it easy to integrate with future algorithms, guaranteeing adaptability as new cryptographic methods emerge. By utilizing widely used algorithms originally written in C, developers can easily incorporate novel cryptographic approaches into their systems. The project has an active community of developers, as evidenced by the release of 12 versions as of May 2023. Its repository contains a wide range of renowned and efficient post-quantum algorithms, including CRYSTAL-kyber, BIKE, FrodoKEM for key exchange, and Dilithium, Falcon, and SPHINCS+ for signature algorithms. While the OQS project does not provide a complete PKI solution on its own, it presents an appealing integration option due to its broad selection of post-quantum algorithms and compatibility with OpenSSL, a widely accepted library /60/.

4.1.2 Selected PKI

After conducting a comprehensive evaluation and analysis of various PKI solutions, a carefully curated selection has been compiled, resulting in a refined list of potential candidates. These candidates exhibit numerous advantageous qualities that make them highly suitable for integration and further research purposes. Notably, they have garnered a commendable reputation, with active contributions from developers. Moreover, their seamless integration capabilities with different platforms and programming languages, coupled with their robust security measures, further enhance their suitability for deployment.

The first candidate is CFSSL which developed and maintained by Cloudflare, is a highly reputable and widely utilized PKI toolkit, renowned for its dependability and industry recognition. CFSSL boasts an active community and robust support, enabling consistent development, frequent updates, and efficient issue resolution for improved security and user assistance. CFSSL is particularly known for its scalability, accommodating organizations with extensive certificate needs. Its strong architecture efficiently manages high volumes of certificate issuance and management for large-scale deployments. CFSSL provides seamless integration with a variety of systems, platforms, and programming languages, affording flexibility for straightforward incorporation into existing infrastructures. This allows developers and system administrators to integrate CFSSL into their preferred environments with ease. CFSSL provides extensive documentation and resources, including tutorials, guides, and examples. These valuable references assist users in comprehending CFSSL's capabilities and implementing them effectively for their particular use cases.

The second candidate is Django-ca which offers seamless integration with the Django web framework, taking advantage of its powerful features and ecosystem. This integration enables developers to easily incorporate Django-ca into their applications, benefitting from Django's robust development practices and saving time and effort in the construction and maintenance of a PKI solution. Django-ca provides a highly customizable and extensible architecture, allowing users to customize certificate fields, user roles, and signing policies to meet their unique requirements. Furthermore, Django-ca features a user-friendly administrative interface, facilitating efficient management of the PKI system. This interface simplifies essential tasks such as certificate issuance, revocation, and management, utilizing the familiar Django admin interface to enhance usability and reduce the learning curve for administrators and users. Additionally, Django-ca enables comprehensive certificate lifecycle management, supporting CSRs, issuance, revocation, and renewal. This streamlined approach simplifies PKI infrastructure management, ensuring efficient handling of certificates throughout their lifecycle. Finally, Django-ca implements role-based access control (RBAC) to enforce fine-grained user permissions and actions in the PKI system. RBAC ensures secure access while maintaining data integrity, enabling organizations to enforce least privilege principles and enhance system security.

The third candidate is EJBCA CE which is a comprehensive PKI solution that offers a range of features for certificate management. It provides customizable certificate profiles, robust features for issuance, revocation, and validation, and comprehensive certificate lifecycle management. EJBCA CE is highly scalable, making it suitable for organizations with large-scale PKI needs. It is capable of efficiently managing high volumes of certificate issuance and management, thereby ensuring seamless operations in complex environments. Furthermore, EJBCA CE is supported by an active community of users and contributors, which facilitates continuous development, support, and regular updates. This vibrant community also provides valuable resources, documentation, and support channels, thus promoting knowledge sharing and facilitating prompt issue resolution. In addition, EJBCA CE prioritizes security and compliance with industry standards, including features such as CRLs, OCSP support, and granular access control. Furthermore, it follows best practices for cryptographic operations and certificate management, ensuring a robust and secure infrastructure. Lastly, EJBCA CE offers seamless integration with diverse systems and platforms, thereby enhancing flexibility and adaptability in various environments. It supports integration with popular software such as LDAP directories, databases, and external certificate authorities, making it easy to incorporate into existing infrastructures.

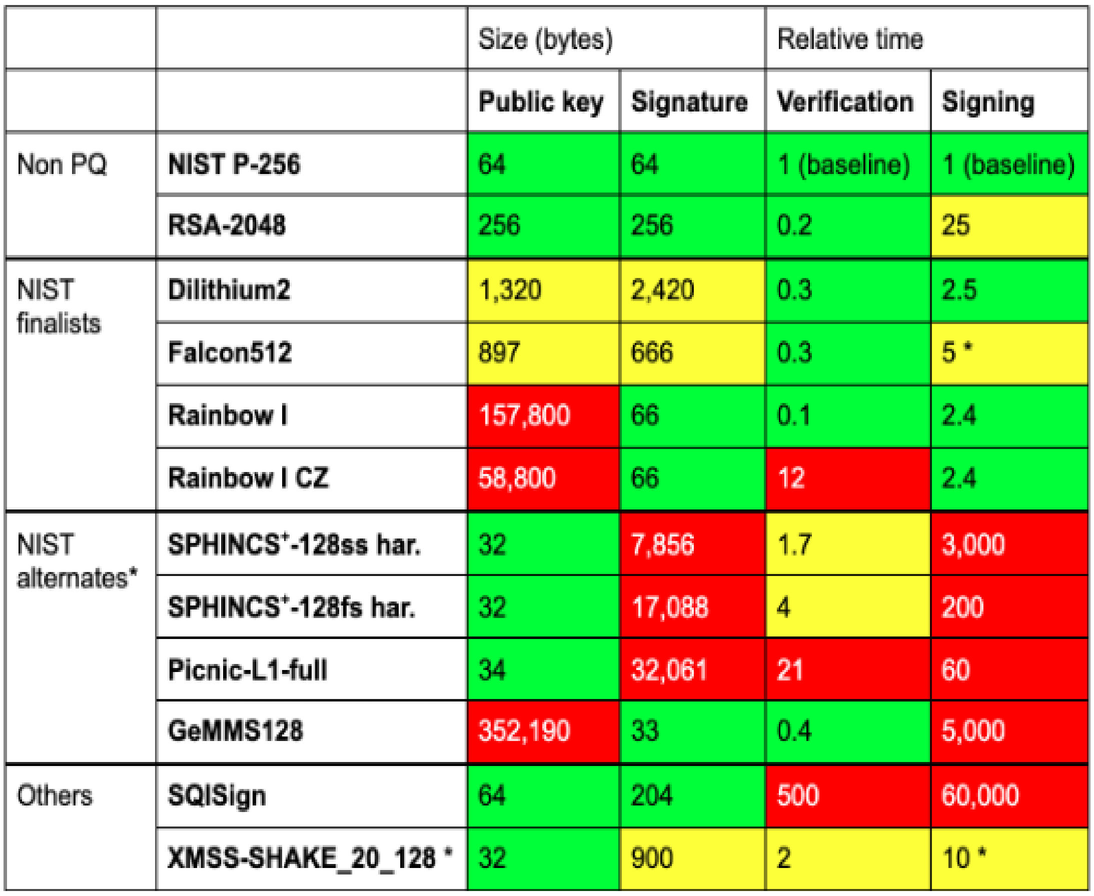
The fourth candidate is XCA which offers an intuitive and user-friendly interface, making PKI system management easier for administrators and users. The simplified interface facilitates tasks such as certificate issuance, revocation, and management, thus reducing the learning curve and improving usability. XCA is lightweight and portable, allowing for compatibility with a variety of environments and operating systems. It is deployable on both desktop and server platforms, providing flexibility and seamless integration with existing infrastructure. XCA provides advanced cryptographic support, offering a wide range of algorithms and protocols. Users can take advantage of the latest security standards and encryption techniques, including secure options for key pair generation, certificate signing, and encryption. This ensures strong security for digital certificates. XCA has a highly customizable architecture, allowing organizations to customize the PKI system to their individual needs. It supports custom certificate fields, extensions, and integration with external systems through APIs, thus promoting flexibility and adaptability. XCA is an open-source project with a vibrant community of users and contributors. The active community guarantees continuous development, support, and regular updates, providing valuable resources, documentation, and support channels. The open-source nature encourages transparency and facilitates community-driven enhancements and improvements.

4.2 Possible Alternatives for Traditional Encryption Algorithms in PKI

Quantum computers can out-calculate traditional encryption algorithms, making existing cryptographic systems vulnerable to attack. To counter this, the NIST has evaluated post-quantum algorithms to potentially replace existing ones. It is important to consider and integrate post-quantum algorithms into the current infrastructure or explore their potential. Doing so is essential for bolstering PKI security against quantum computer attacks. A selection of encryption algorithms commonly used in PKI software will be briefly examined.

Within the third round of the NIST post-quantum encryption competition, a selection of encryption algorithms emerged as noteworthy candidates for replacing traditional cryptographic algorithms. These algorithms have undergone rigorous evaluation and demonstrate promising characteristics. In the category of Public-key Encryption and Key-establishment Algorithms, the finalist algorithms include Classic McEliece, CRYSTALS-KYBER, NTRU, and SABER. Moreover, alternative options such as BIKE, FrodoKEM, HQC, NTRUPrime, and SIKE are available. In terms of Digital Signature Algorithms, the finalists encompass CRYSTAL-DILITHIUM, Rainbow, and Falcon, while choices consist of GeMSS, Picnic, and SPHINCS+. Each of these algorithms possesses distinct advantages and trade-offs in various scenarios. Collectively, they present a viable approach to achieving quantum-resistant security, thus rendering them suitable for integration within PKI software or any application requiring quantum-safe cryptographic mechanisms /61/.

Cloudflare recently published a blog post that conducted a comprehensive assessment and comparison of Digital Signature Algorithms concerning their traditional counterparts. The evaluation was based on two key factors: size and relative time. These factors were carefully analyzed to determine the effectiveness and efficiency of the Digital Signature Algorithms in practical applications. By considering the size of the signature and the relative time required for signature generation and verification, Cloudflare aimed to provide a thorough understanding of the performance characteristics and feasibility of these algorithms compared to traditional alternatives. This analysis from Cloudflare's blog post provides valuable insights for evaluating the suitability and practicality of Digital Signature Algorithms in real-world scenarios, thus enhancing the understanding of cryptographic choices in the context of the thesis /62/.



		Size (bytes)		Relative time	
		Public key	**Signature**	**Verification**	**Signing**
Non PQ	**NIST P-256**	64	64	1 (baseline)	1 (baseline)
	RSA-2048	256	256	0.2	25
NIST finalists	**Dilithium2**	1,320	2,420	0.3	2.5
	Falcon512	897	666	0.3	5 *
	Rainbow I	157,800	66	0.1	2.4
	Rainbow I CZ	58,800	66	12	2.4
NIST alternates*	**SPHINCS$^+$-128ss har.**	32	7,856	1.7	3,000
	SPHINCS$^+$-128fs har.	32	17,088	4	200
	Picnic-L1-full	34	32,061	21	60
	GeMMS128	352,190	33	0.4	5,000
Others	**SQISign**	64	204	500	60,000
	XMSS-SHAKE_20_128 *	32	900	2	10 *

Figure 7. Digital Signature Algorithms comparison /62/.

4.3 Review of Available Post-quantum Encryption Algorithms

The field of cryptography encompasses numerous algorithms for public key encryption and digital signature schemes, as previously mentioned. In this thesis, our focus will primarily be on the algorithms recommended by the NIST. The selected algorithm for public key encryption and the key establishment is CRYSTALS-KYBER, while for digital signature schemes, the chosen algorithms are CRYSTALS-DILITHIUM, falcon, and SPHINCS+ /63/.

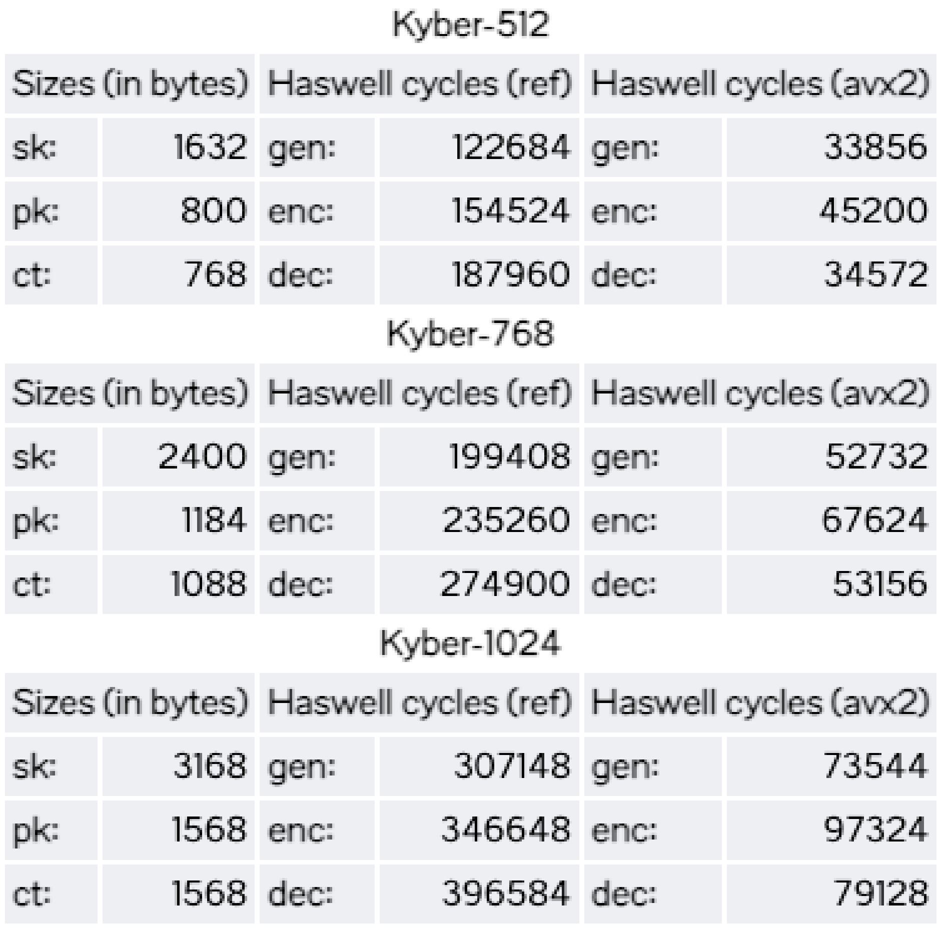
Given NIST's endorsement, our objective is to identify and examine a wide range of implementations for each of these selected algorithms. This will provide us with a comprehensive set of options to either support our implementation or facilitate further research. By delving deeper into the characteristics and intricacies of these algorithms, we aim to foster their integration and explore their potential for future advancements.

In the subsequent section, we will delve into a comprehensive discussion of the aforementioned algorithms, shedding light on their distinct features and providing valuable insights to aid in their integration and further investigation.

4.3.1 Public-key Encryption and Key-establishment Algorithms

Kyber is an IND-CCA2-secure key encapsulation mechanism (KEM), which is based on the hardness of solving the LWE problem over module lattices. It provides different levels of security depending on the size of the key being used. Specifically, Kyber-512 provides security roughly equivalent to AES-128, Kyber-768 provides security roughly equivalent to AES-192, and Kyber-1024 provides security roughly equivalent to AES-256. Due to its impressive performance, Kyber is already being integrated into libraries and systems by the industry. For example, Cloudflare has integrated Kyber alongside other post-quantum algorithms into CIRCL. The performance of Kyber can be seen in Figure 8 below, which demonstrates its efficiency compared to other algorithms.

The main repository contains the official reference implementation of the Kyber key encapsulation mechanism, as well as an optimized implementation specifically designed for x86 CPUs that support the AVX2 instruction set. While the original implementation is written in the C programming language, it is worth noting that numerous implementations in other languages are available on the internet. These alternative implementations can be leveraged to suit our specific needs if the original implementation is not compatible with the language requirements of our PKI software.



Kyber-512

Sizes (in bytes)		Haswell cycles (ref)		Haswell cycles (avx2)	
sk:	1632	gen:	122684	gen:	33856
pk:	800	enc:	154524	enc:	45200
ct:	768	dec:	187960	dec:	34572

Kyber-768

Sizes (in bytes)		Haswell cycles (ref)		Haswell cycles (avx2)	
sk:	2400	gen:	199408	gen:	52732
pk:	1184	enc:	235260	enc:	67624
ct:	1088	dec:	274900	dec:	53156

Kyber-1024

Sizes (in bytes)		Haswell cycles (ref)		Haswell cycles (avx2)	
sk:	3168	gen:	307148	gen:	73544
pk:	1568	enc:	346648	enc:	97324
ct:	1568	dec:	396584	dec:	79128

Figure 8. Kyber Performance /64/.

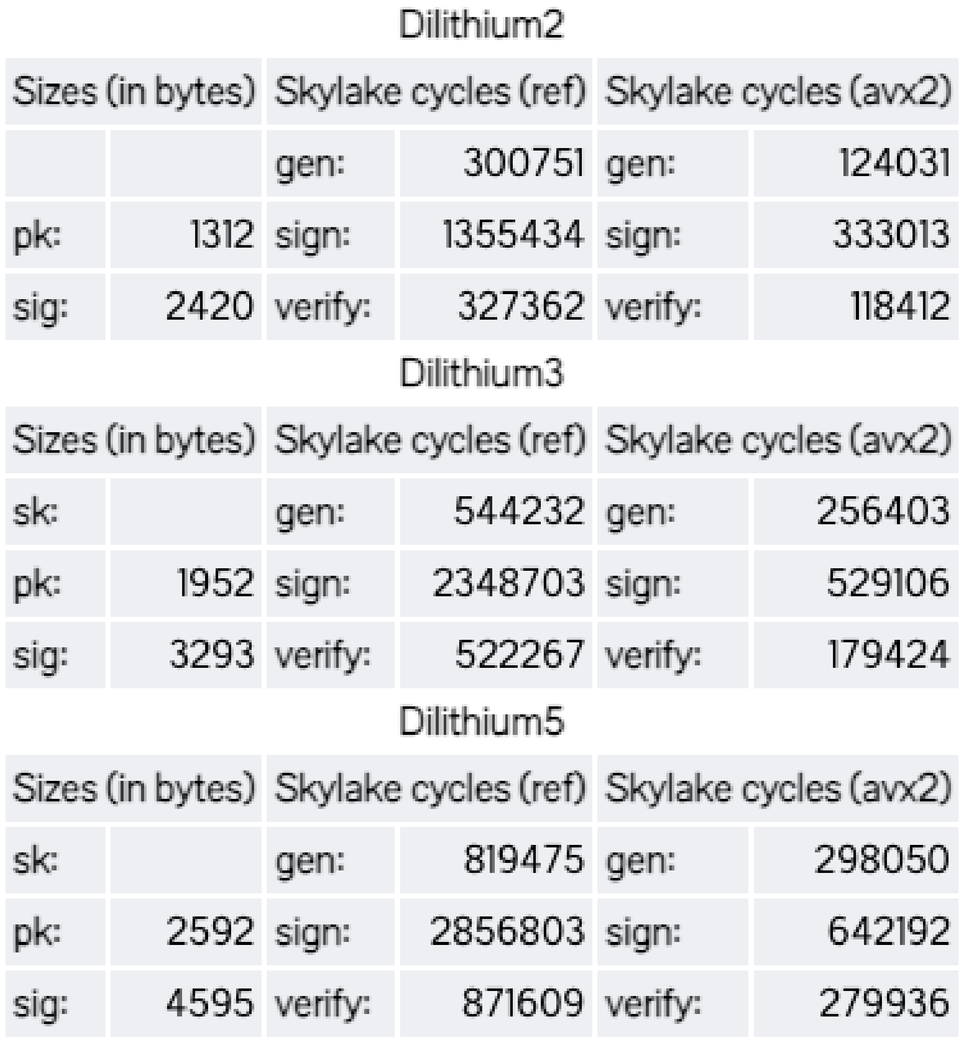
The primary implementation of the Kyber algorithm within the repository was initially developed three years ago in 2020. However, it is noteworthy that active development efforts have continued, as evidenced by the most recent commit made in May 2023. This ongoing development highlights the commitment to refining and enhancing the implementation of the algorithm, ensuring it remains up to date with the latest advancements and addressing any potential vulnerabilities or performance optimizations /65/.

In addition to the official C implementation, a wide array of implementations in various programming languages are available. These include but are not limited to C++, Rust, Python, Java, TypeScript, Golang, JavaScript, and VHDL. This diverse range of language options provides ample choices for developers seeking to utilize the Kyber algorithm in their projects, enabling them to integrate it seamlessly without the need to develop an implementation from scratch.

4.3.2 Digital Signature Algorithms

The first algorithm is Dilithium. The hardness of lattice problems over module lattices makes Dilithium a digital signature scheme that is strongly secure under chosen message attacks. This security notion means that an adversary, even with access to a signing oracle, cannot produce a signature of a message whose signature he has not yet seen, nor produce a different signature of a message that he already saw signed. Lyubashevsky's "Fiat-Shamir with Aborts" technique, which uses rejection sampling to make lattice-based Fiat-Shamir schemes compact and secure, serves as the design basis for Dilithium. Dilithium improves on the most efficient scheme of Bai and Galbraith, which only uses uniform distribution, by using a new technique that shrinks the public key by more than a factor of 2. To the best of our knowledge, Dilithium has the smallest public key + signature size of any lattice-based signature scheme that only uses uniform sampling. We offer two different implementations: a C reference implementation and an optimized implementation using AVX2 vector instructions. The efficiency of the algorithms is depicted in Figure 9 below.

The main repository hosts the official reference implementation of the Dilithium signature scheme, along with an optimized implementation specifically tailored for x86 CPUs that support the AVX2 instruction set. These implementations include various test and benchmarking programs, as well as a Makefile to streamline the compilation process. The primary codebase of the repository was developed approximately between 2020 and 2021. However, it appears that the project has become relatively inactive, with the most recent commit dating back to 2022 /67/.



Dilithium2

Sizes (in bytes)		Skylake cycles (ref)		Skylake cycles (avx2)	
		gen:	300751	gen:	124031
pk:	1312	sign:	1355434	sign:	333013
sig:	2420	verify:	327362	verify:	118412

Dilithium3

Sizes (in bytes)		Skylake cycles (ref)		Skylake cycles (avx2)	
sk:		gen:	544232	gen:	256403
pk:	1952	sign:	2348703	sign:	529106
sig:	3293	verify:	522267	verify:	179424

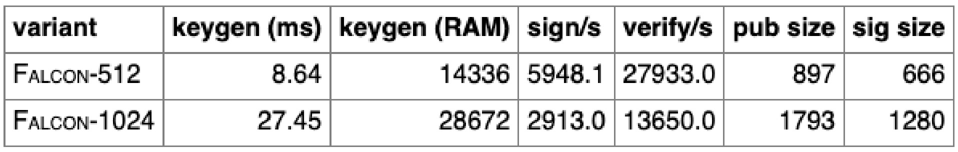
Dilithium5

Sizes (in bytes)		Skylake cycles (ref)		Skylake cycles (avx2)	
sk:		gen:	819475	gen:	298050
pk:	2592	sign:	2856803	sign:	642192
sig:	4595	verify:	871609	verify:	279936

Figure 9. Dilithium Performance /66/.

Similar to Kyber, Dilithium has garnered attention in the cryptographic community, leading to the availability of implementations in multiple programming languages. The official repository provides the reference and optimized implementations of Dilithium, supplemented with test programs and a Makefile. Although not as abundant as Kyber, the languages commonly associated with Dilithium implementations include Java, Golang, Rust, C++, and JavaScript. These alternative language implementations broaden the accessibility and usability of Dilithium, enabling developers to adopt the scheme in their preferred programming languages, alongside the original C implementation.

The second algorithm is Falcon. Falcon builds upon the theoretical framework of Gentry, Peikert, and Vaikuntanathan for lattice-based signature schemes. It instantiates this framework using NTRU lattices and a trapdoor sampler known as "fast Fourier sampling." The foundation of Falcon relies on the computational challenge of solving the short integer solution problem (SIS) over NTRU lattices. To date, no efficient algorithm exists to solve this problem, even with the assistance of quantum computers. Falcon offers advanced security with negligible key leakage, compact signatures, fast processing, scalability, and efficient RAM usage. It utilizes true Gaussian sampling, NTRU lattices, and fast Fourier sampling for high-security signatures, shorter signature sizes, rapid operations, scalability, and compatibility with memory-constrained devices. The performance of Falcon is depicted in the accompanying figure.



variant	keygen (ms)	keygen (RAM)	sign/s	verify/s	pub size	sig size
Falcon-512	8.64	14336	5948.1	27933.0	897	666
Falcon-1024	27.45	28672	2913.0	13650.0	1793	1280

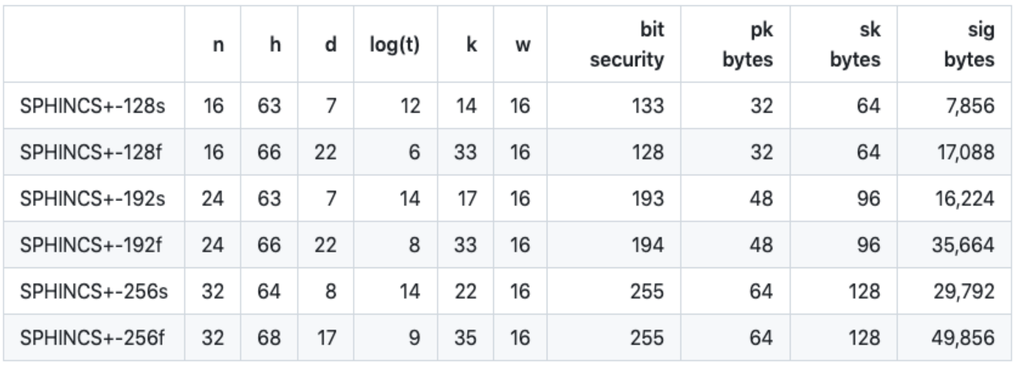
Figure 10. Falcon Performance /68/.

The primary repository of Falcon is implemented in C, demonstrating the initial development and focus of the project. However, an additional repository exists specifically for the Python implementation of Falcon, indicating efforts to expand its availability in different programming languages. While the main repository appears relatively inactive, the Python implementation continues to receive updates and maintenance, indicating ongoing development /69/.

Furthermore, alternative versions of Falcon can be found on various platforms with implementations available in Golang, C++, C (besides the official one not in GitHub), Rust, and Javascript. Although the number of resources for Falcon is not as abundant as those for Kyber, it is comparable to the resources available for Dilithium.

Finally, SPHINCS+ is a stateless hash-based signature scheme that was originally proposed as part of the NIST post-quantum crypto project. It builds upon the advancements of the SPHINCS signature scheme, which was initially presented at EUROCRYPT 2015. SPHINCS+ incorporates several enhancements, primarily focused on reducing the size of signatures.

These signature schemes are derived by instantiating the SPHINCS+ construction with three different hash functions: SHAKE256, SHA-256, and Haraka. In the second round submission of SPHINCS+, a split is introduced for each of the three variants, resulting in a simplified and robust version of the scheme for each hash function choice. The robust variant corresponds to the SPHINCS+ version submitted in the first round and retains all the conservative security guarantees provided earlier. The performance of SPHINCS+ is represented in Figure 11.



	n	h	d	log(t)	k	w	bit security	pk bytes	sk bytes	sig bytes
SPHINCS+-128s	16	63	7	12	14	16	133	32	64	7,856
SPHINCS+-128f	16	66	22	6	33	16	128	32	64	17,088
SPHINCS+-192s	24	63	7	14	17	16	193	48	96	16,224
SPHINCS+-192f	24	66	22	8	33	16	194	48	96	35,664
SPHINCS+-256s	32	64	8	14	22	16	255	64	128	29,792
SPHINCS+-256f	32	68	17	9	35	16	255	64	128	49,856

Figure 11. SPHINCS+ Performance /70/.

Similar to the aforementioned algorithms, the original implementation of SPHINCS+ was initially written in the C programming language. The repository was established approximately a year ago and is released under the CC0 1.0 Universal license. The developer and author of the repository continue to actively contribute, as evidenced by the latest commit made in April 2023.

Additionally, like other digital signature algorithms, alternative implementations of SPHINCS+ in various programming languages have been identified. These languages include Java, Go, Rust, Python, and C++. While the availability of resources for SPHINCS+ is substantial compared to other digital signature algorithms, it falls short in comparison to the extensive range of resources available for Kyber. While the resources for SPHINCS+ may not be as abundant as those for Kyber, they still offer a considerable selection for developers seeking to utilize this signature scheme /71/.

4.4 Integration of Post-quantum Algorithms with PKI

This chapter outlines the implementation and testing of the integration of Post-Quantum PKI, which has several significant use cases. Firstly, it ensures the confidentiality and integrity of data against potential quantum attacks, making it essential for secure online transactions and e-commerce, as it safeguards sensitive information such as credit card details. Governments and defense organizations can benefit from post-quantum PKI by securing their classified communications, document signing processes, and identity management systems. Additionally, post-quantum PKI is essential in protecting critical infrastructure systems, preventing potential disruptions caused by quantum attacks. Cloud service providers can leverage post-quantum PKI to enhance the security of their infrastructure and protect customer data. In healthcare and medical systems, post-quantum PKI secures electronic health records, medical device communications, and telemedicine platforms, ensuring privacy and protection against quantum-based threats. These use cases illustrate the importance of implementing post-quantum PKI to protect sensitive information, secure communications, and maintain the integrity of various systems and applications in the face of quantum computing advancements.

4.4.1 Planning

Following the evaluation of selected PKIs and attempts to inspect their source code and integrate them with post-quantum algorithms, it was determined that such endeavors exceeded the time constraints established for this thesis. While the selected PKIs offered a wealth of resources for future research, their complex nature, consisting of extensive lines of code, rendered them unsuitable for implementation within the scope of this study. As a result, a simpler PKI implementation will be pursued, leveraging various open-source resources and guidelines. The key aspect of this implementation is its reliance on the Open Quantum Safe (OQS) framework, as previously mentioned in the Candidates subchapter. The implementation will involve building a straightforward PKI system that encompasses crucial components such as a Root Certificate Authority (rootCA), capable of issuing server certificates, mechanisms for certificate revocation, and the inclusion of a CRL to store information on revoked certificates. As highlighted in the review of post-quantum algorithms, this implementation emphasizes the adoption of specific algorithms derived from the NIST competition. Notably, the chosen algorithms include CRYSTALS-kyber, CRYSTALS-dilithium, Falcon, and SPHINCS+. These algorithms, provided by the OQS framework, are central to the implementation's design and execution /72/.

4.4.2 Implementation

This thesis outlines the implementation of a project on Linux Ubuntu version 20.04.6 LTS. Therefore, the implementation and command line instructions provided are Linux-based. The initial phase of the implementation involves the installation of the OQS repository github.com/open-quantum-safe/, which serves as the foundation for the PKIs before additional components are incorporated. All the command lines will be demonstrated following each paragraph to clarify the process.

The installation process of the OQS can be summarized as follows. To build the OQS, several libraries must be acquired. These libraries include cmake, gcc, libtool, libssl-dev, make, ninja-build, and git. The installation process can be completed with the installation command (1). It is imperative to remember to run the update (2) before installation to ensure all components are up to date. The following steps involve the cloning and installation of the liboqs library. This process is composed of several sub-steps. To begin, the liboqs repository should be cloned from the official GitHub with the suffix liboqs. Subsequently, a build directory should be created within the liboqs directory. After navigating to the build directory, the next step involves executing the command (3). To complete the installation of liboqs, the ninja commands (4) and (5) should be executed in sequence. The concluding step of the installation process necessitates the construction of the OQS repository through the utilization of the make command. This step encompasses the following substeps: Initially, the aforesaid repository should be cloned, with the addition of the suffix "openssl" to its name. Subsequently, one must navigate to the directory containing OpenSSL and execute the command (6). Lastly, command (7) should be employed to initiate the build process.

```
$sudo apt install                                    (1)

$sudo apt update                                     (2)

$cmake -GNinja -DOQS_USE_OPENSSL=1 ..                (3)

$ninja                                               (4)

$sudo install ninja                                  (5)

$./Configure no-shared linux-x86_64 -lm              (6)

$make -j                                             (7)
```

Upon the successful completion of the installation process, the next step is to proceed to construct our PKI based on the installed components. Furthermore, there are more to incorporate additional components functional PKI system. To establish PKI, it is necessary to have a Root CA and a Signing CA. Furthermore, the Signing CA should be utilized to generate a certificate for a sample server or email. Moreover, command line tools to revoke certificates and create CRLs should be incorporated, as well as the capability to view issued certificates. All of these components are indispensable for a successful and secure PKI implementation.

To begin the process, navigate to openssl directory and create the RootCA by executing the (8) command. To provide further clarity, it is essential to understand the significance of each parameter used in the command line. The "req" parameter denotes the request, while "-x509" signifies the output format as the x509 structure. The "-new" parameter indicates a new certificate request, while "-newkey" specifies the type of the new key to be generated. The "-keyout" parameter determines the destination file for the key, and "-out" specifies the output file for the certificate. The "-nodes" flag ensures that the output key is not encrypted. The "-subj" parameter sets the subject for the certificate, and the "-days" parameter defines the validity period of the certificate in days. Besides, the "-config" parameter refers to the request template file, which plays a crucial role in the certificate creation process. Lastly, it is important to note that the placeholder "<SIG>" in the aforementioned command represents the type of signature algorithm to be utilized for encryption. The OQS repository offers support for various signature algorithms recommended by NIST, such as Dilithium, Falcon, and SPHINCS+. The selection of a specific algorithm is contingent upon the preferences of the creator or the requirements of the PKI system being implemented. The following step involves the creation of a Signing CA, which follows a similar procedure as that of the RootCA setup. However, slight modifications are made to differentiate the Signing CA; these modifications include changing "_CA" to "_signingCA" in the file names and adjusting the Common Name to "SigningCA" within the same command line as mentioned earlier.

In the subsequent step, the server generates its key pair and a certificate request, which is then forwarded to the CA. This can be achieved by executing the (9) command. This command generates a new key pair and creates a certificate request. The private key is saved in the file specified by "<SIG>_server.key", while the certificate request is stored in "<SIG>_server.csr". To sign and generate the server's certificate, the command (10) is used. To provide a more comprehensive explanation, this command performs the signing process and generates the server's certificate. The "-in" parameter denotes the input file as "<SIG>_server.csr", which contains the certificate request. The "-out" parameter designates the output file as "<SIG>_server.crt", where the signed certificate will be stored. The "-CA" and "-CAkey" parameters respectively indicate the RootCA certificate file "<SIG>_CA.crt" and its corresponding private key file "<SIG>_CA.key" used for signing. The "-CAcreateserial" flag creates a serial number file for the certificate.

```
$apps/openssl req -x509 -new -newkey <SIG> -keyout
<SIG>_CA.key -out <SIG>_CA.crt -nodes -subj "/CN=Root CA" -
days 365 -config apps/openssl.cnf                          (8)

$apps/openssl req -new -newkey <SIG> -keyout
<SIG>_server.key -out <SIG>_server.csr -nodes -subj
"/CN=Server" -config apps/openssl.cnf                      (9)

$apps/openssl x509 -req -in <SIG>_server.csr -out
<SIG>_server.crt -CA <SIG>_CA.crt -CAkey <SIG>_CA.key -
CAcreateserial -days 365                                   (10)
```

To establish a basic server capable of key exchange with enabled algorithms, the previously created key and certificate are utilized. The following command lines demonstrate the creation of a server and client for key exchange within the PKI. For the server, execute (11). For the client in a separate terminal, execute (12). The server command initiates a server instance, utilizing the specified server certificate "<SIG>_server.crt" and its corresponding private key "<SIG>_server.key". The "-www" flag enables a basic web server functionality, while "-tls1_3" ensures that TLS version 1.3 is used. On the other hand, the client command creates a client instance in a separate terminal. The "-groups" parameter specifies the desired key exchange algorithm "<KEX>" such as kyber512, and the "-CAfile" parameter references the RootCA certificate "<SIG>_CA.crt" to establish trust. These commands facilitate the establishment of a secure key exchange environment within the PKI, allowing secure communication between the server and client.

The final step in the establishment of a basic PKI using the OpenSSL repository involves generating a revocation list. This can be achieved by executing the (13) command, which generates a revocation list denoted as "ca.crl" using the OpenSSL configuration file "openssl.cnf". The "-gencrl" option specifies the generation of a revocation list, ensuring that a comprehensive record of revoked certificates is maintained within the PKI system.

Furthermore, it is essential to understand how to view certificates and revoke them when necessary. To view the certificate in a human-readable format, the command (14) should be executed, which provides a detailed textual representation of the certificate "<SIG>_server.crt", allowing for easy examination and analysis of its contents. To revoke a certificate, the command (15) should be used, which is performed by the Certification Authority (CA) and allows for the designation of a specific reason for revocation, with "superseded" used as an example here. These command lines enable the viewing and managing of certificates within the PKI system, guaranteeing transparency and control over the certificate lifecycle.

```
$apps/openssl   s_server   -cert   <SIG>_server.crt   -key
<SIG>_server.key -www -tls1_3                          (11)

$apps/openssl  s_client  -groups  <KEX>  -CAfile  <SIG>_CA.crt
                                                       (12)

$openssl ca -gencrl -config openssl.cnf -out ca.crl    (13)

$apps/openssl x509 -in <SIG>_server.crt -noout -text   (14)

$apps/openssl    ca    -config    apps/openssl.conf    -revoke
<SIG>_server.crt -crl_reason superseded                (15)
```

In conclusion, a basic PKI has been established using the OpenSSL repository. This PKI includes the RootCA, and SigningCA, and covers essential procedures such as generating certificates for servers, enabling key exchange between servers and clients, creating CRLs, viewing certificate details, and revoking certificates using OpenSSL-based commands. This implementation demonstrates the fundamental components and operational aspects of a PKI, providing a solid foundation for secure communication within a networked environment.

4.4.3 Testing

The Testing Phase of Integration PKI involves three distinct tests: Verification Testing, Decoding Testing, and Time (Performance) Testing. The initial phase of the Testing Process entails conducting Verification Tests to validate the authenticity and integrity of the certificates generated in the preceding section. This Verification is carried out by utilizing a command line instruction, namely: $apps/openssl verify -CAfile <SIG>_CA.crt <SIG>_server.pem. This command line execution performs a comprehensive examination of the certificates, comparing them against the corresponding PEM or cert files. Upon successfully verifying the certificates using the aforementioned command line and obtaining the "OK" output, the subsequent step involves creating a revoke command to invalidate the certificate. This is achieved by employing the revoke command line specified in the implementation section. After a revocation, the verification command line is executed once more, resulting in a "fail" outcome. This outcome validates the effectiveness of the verification command line in accurately determining the validity of certificates. Hence, the verification command line successfully passes the test by effectively assessing the certificate's verifiability. Through this rigorous verification procedure, the validity and trustworthiness of the certificates are established, thereby ensuring the robustness and reliability of the Integration PKI System.

The second phase of testing involves conducting Decoding Tests to assess the certificate's ability to accurately decode and present the information provided by the CA. For this purpose, a Python script is employed, which is included in the Appendices section. The Python code leverages the pyasn1 and pyasn1-modules libraries, which can be installed via the pip3 package manager. In Ubuntu 20.04, the pip3 installation can be executed using the following command line: $sudo apt install python-pip3. To ensure compatibility with the specific file requiring decoding, the script is modified accordingly. The Python script is executed using the command python3 filename.py to obtain the results. Upon conducting the

Decoding Test, the system successfully passes the assessment. The test verifies that the system can correctly decode the certificate and extract essential information such as valid time, invalid time, creation time, and the name of the CA. The obtained results align accurately with the information provided in the corresponding certificate file.

The final phase of testing involves conducting Time (Performance) Tests, focusing on evaluating the speed and efficiency of specific algorithms without employing quantum computers for algorithm cracking. The objective of this testing is to measure the signing time and verification time for each algorithm selected in the previous section. The Performance Testing is conducted using the built-in command line functionality available in the repository. Specifically, the command line structure is as follows: $apps/openssl speed <algorithms>. The results obtained from the Performance Testing will be presented in a figure, providing a visual representation of the performance characteristics of the tested algorithms. This Performance Testing phase serves to assess and compare the signing and verification times of the selected algorithms, providing valuable insights into their efficiency and suitability for the Integration PKI System.

Algorithms	*Signing (s)*	*Verification(s)*	*Signing/s*	*Verification/s*
dilithium2	0.0001	< 0.0001	8561	22399
dilithium3	0.0002	0.0001	5895	15291
dilithium5	0.0002	0.0001	4424	8011
falcon512	0.0003	0.0001	2964	17198
falcon1024	0.0007	0.0001	1349	8841
sphincs-sha2	0.0099	0.0009	101	1131
sphincs-shake	0.0221	0.0016	45	611

Figure 12. Speed Testing.

5 DISCUSSION AND CONCLUSION

5.1 Discussion

This thesis aimed to explore the state-of-the-art post-quantum algorithms and integrate them with the PKI to address the increasing threat posed by quantum computers to conventional cryptographic systems. The objective of the study was to pave the way for the next generation of spacecraft and satellite ground segment systems, where secure communication is paramount.

To this end, the student conducted a comprehensive analysis of nine different PKIs and the OpenSSL library, evaluating their compatibility and suitability for integration with PQC. By comparing their features, performance, and security, the study identified the recommended PKIs that can be effectively combined with PQC for practical implementation. Furthermore, the thesis delved into the NIST competition for post-quantum algorithms, offering a detailed examination of the selected algorithms. Through extensive research, various implementations of these algorithms in different programming languages were identified. Additionally, a practical aspect of the thesis involved the implementation of a simple PKI using the OQS library, serving as a proof-of-concept, demonstrating the feasibility of integrating PQC with PKI.

The findings of this research have significant implications for the future of secure communication systems, particularly in the context of spacecraft and satellite ground segment operations. As quantum computers continue to advance in computational power, the vulnerabilities of current cryptographic systems become increasingly evident. The integration of PQC with PKI offers a promising solution to mitigate these risks and ensure the confidentiality, integrity, and authenticity of sensitive data. It is essential to acknowledge the limitations of this research. While the selected PKIs and algorithms were thoroughly evaluated, there may be other viable options that were not included in this study.

Additionally, the practical implementation was limited to a simplified version, and further research is required to develop more comprehensive and robust systems.

In summary, by evaluating different PKIs, exploring the NIST competition algorithms, and implementing a proof-of-concept PKI, this research has laid the groundwork for the future development and implementation of secure communication systems resilient against quantum threats. The findings and recommendations of this study can guide organizations and researchers in adopting suitable PKIs and effectively integrating them with PQC, ensuring secure and quantum-resistant communication in the next generation of spacecraft and satellite ground segment systems.

5.2 Future Challenges

As quantum computing continues to advance and garner substantial investments, there is an impending threat posed by quantum computers to the so-called "quantum-resistant" algorithms. It is only a matter of time before quantum computers can potentially break these algorithms, making current cryptographic systems vulnerable. To address this challenge, it is essential to allocate more resources and investments to research and develop more robust post-quantum algorithms shortly. The ongoing advancements in quantum computing necessitate a proactive approach to stay ahead of potential cryptographic breaches.

Furthermore, the integration of post-quantum algorithms with intricate PKIs presents another significant challenge. Many PKIs currently in use run and verify numerous applications and web browsers, making them inherently complex systems. While the thesis focused on the implementation of a simple PKI, integrating post-quantum algorithms with such complex PKIs is essential for ensuring their security in real-world applications, particularly in domains such as spacecraft or satellite operations. The challenge lies in effectively integrating post-quantum cryptography into these intricate PKIs without compromising their functionality or introducing vulnerabilities.

To effectively address these challenges, it is important to adopt a forward-thinking approach that anticipates future advancements in quantum computing and their potential impact on cryptographic systems. By proactively investing in research and development, fostering interdisciplinary collaborations, and continually evaluating and updating PKIs, we can enhance the security and resilience of our digital infrastructure in the face of emerging quantum threats.

To this end, it is necessary to allocate more resources and investments to research and develop stronger algorithms to withstand quantum attacks and integration of these algorithms into complex PKI frameworks. By addressing these challenges head-on, we can pave the way for a secure and quantum-resistant digital future.

5.3 Conclusion

In conclusion, this thesis has revealed several key findings. Firstly, there is a wide range of PKIs available on the internet, each with distinct features, reputation, and complexity. When integrating post-quantum algorithms into a PKI, it is essential to take into account the most suitable PKI and compatible post-quantum algorithms for ensuring a secure and efficient integration. Secondly, the strength of quantum computers will continue to increase shortly, thus posing a significant risk to organizations relying on traditional cryptographic algorithms. Consequently, it is recommended that organizations replace their existing PKIs and transition to post-quantum algorithms to enhance their resilience and protect their sensitive information against the imminent threat of quantum computing. Lastly, the most effective approach to withstanding the challenges posed by quantum computing lies in the integration of complex and sophisticated PKIs with post-quantum algorithms. By adopting such a strategy, organizations can significantly prolong the security and longevity of their cryptographic systems in the quantum era.

Stakeholders must recognize the urgency of this transition and take appropriate steps to protect their systems and data in the face of quantum computing advancements.

APPENDICES

APPENDIX 1. Python Script for Decoding Testing

```
from pyasn1_modules import pem, rfc2459

from pyasn1.codec.der import decoder

substrate = pem.readPemFromFile(open('cert-file.pem'))

cert              =              decoder.decode(substrate,
asn1Spec=rfc2459.Certificate())[0]

print(cert.prettyPrint())
```

6 REFERENCES

/1/ Edgar, T. W. and Manz, D. O. 2017. Research Methods for Cyber Security - Syngress.

/2/ Roper, L. D., Renn, K. A. and Biddix, J. 2018. Research Methods and Applications for Student Affairs - Jossey-Bass.

/3/ Varantola, K., Launis, V., Helin, M., Spoof, S.-K. and Jappinen, S. 2012. FINNISH ADVISORY BOARD ON RESEARCH INTEGRITY.

/4/ Suranjan, Choudhury; Kartik, Bhatnagar; Wasim, Haque. 2002. Public Key Infrastructure Implementation and Design - United Kingdom - Wiley.

/5/ Carlisle, Adams; Steve, Lloyd. 1999. Understanding Public-Key Infrastructure: Concepts, Standards, and Deployment Considerations - Sams.

/6/ Buchmann, J. A., Karatsiolis, E. and Wiesmaier, A. 2013. Introduction to Public Key Infrastructures - Darmstadt, Germany - Springer.

/7/ Andre, Karamanian; Srinivas, Tenneti; Francois, Dessart. 2011. PKI Uncovered: Certificate-Based Security Solutions for Next-Generation Networks - Cisco Press.

/8/ Weise, J. 2001. Public Key Infrastructure Overview - Sun BluePrints™ OnLine.

/9/ Boyles, T. 2010. CCNA® Security Study Guide - Canada - Sybex.

/10/ Pointcheval, D. 2023. Asymmetric Cryptography - Wiley-ISTE.

/11/ Greene, S. 2020. CompTIA Security+ SY0-601 – Pearson - IT Certification.

/12/ Windley, P. J. 2023. Learning Digital Identity - O'Reilly Media, Inc.

/13/ Hucaby, D. 2015. CCNA Wireless 200-355 Official Cert Guide - Cisco Press.

/14/ John, Viega; Matt, Messier; Pravir, Chandra. 2002. Network Security with OpenSSL - O'Reilly Media, Inc.

/15/ Komar, B. 2008. Windows Server® 2008 PKI and Certificate Security - Microsoft Press.

/16/ Michael, Watkins; Kevin, Wallace. 2008. CCNA Security Official Exam Certification Guide (Exam 640-553) - Cisco Press.

/17/ Wong, D. 2021. Real-World Cryptography - Manning Publications.

/18/ Mitchell, C. 2000. PKI standards, Information Security Group, Royal Holloway - University of London, United Kingdom.

/19/ Carlisle, Adams; Steve, Lloyd. 2002. Understanding PKI: Concepts, Standards, and Deployment Considerations, Second Edition - Addison-Wesley Professional.

/20/ Khlebnikov, Alexei; Adolfsen, Jarle. 2022. Demystifying Cryptography with OpenSSL 3.0 - Packt Publishing.

/21/ Viggiano, Greg; Brin, David. 2023. Convergence: Artificial Intelligence and Quantum Computing - Wiley.

/22/ Thompson, R. C. 2003. Physics with Trapped Charged Particles Lectures from the Les Houches Winter School - World Scientific.

/23/ Kaiser, Sarah; Granade, Chris. 2021. Learn Quantum Computing with Python and Q# - Manning Publications.

/24/ Mehta, N. 2020. Quantum Computing, Pragmatic - Bookshelf.

/25/ Kommadi, B. 2020. Quantum Computing Solutions: Solving Real-World Problems Using Quantum Computing and Algorithms - Apress .

/26/ Grimes, R. A. 2019. Cryptography Apocalypse - Wiley.

/27/ Sharkey, Keeper L.; Chancé, Alain; Khan, Alex. 2022. Quantum Chemistry and Computing for the Curious - Packt Publishing.

/28/ Johnston, Eric R.; Harrigan, Nic; Mercedes Gimeno-Segovia. 2019. Programming Quantum Computers - O'Reilly Media, Inc.

/29/ Loredo, R. 2020. Learn Quantum Computing with Python and IBM Quantum Experience - Packt Publishing.

/30/ Ganguly, Srinjoy; Cambier, Thomas. 2021. Quantum Computing with Silq Programming - Packt Publishing.

/31/ Silva, V. 2018. Practical Quantum Computing for Developers: Programming Quantum Rigs in the Cloud using Python, Quantum Assembly Language and IBM QExperience - Apress.

/32/ Jiewen, Yao; Zimmer, Vincent. 2020. Building Secure Firmware: Armoring the Foundation of the Platform - Apress.

/33/ Easttom, C. 2021. Quantum Computing Fundamentals - Addison-Wesley Professional.

/34/ Ghonge, Mangesh M.; Pramanik, Sabyasachi; Mangrulkar, Ramchandra. 2022. Cyber Security and Digital Forensics - Wiley-Scrivener.

/35/ Imre, Sandor; Balazs, Ferenc. Quantum Computing and Communications: An Engineering Approach.

/36/ Pfleeger, Charles; Pfleeger, Shari Lawrence; Coles-Kemp, Lizzie. 2023. Security in Computing, 6th Edition - Addison-Wesley Professional.

/37/ Kaufman, Charlie; Perlman, Radia; Speciner, Mike. 2022. Network Security: Private Communications in a Public World, 3rd Edition - Addison-Wesley Professional.

/38/ Aumasson, J.-P. 2017. Serious Cryptography - No Starch Press.

/39/ Ciesla, R. 2020. Encryption for Organizations and Individuals: Basics of Contemporary and Quantum Cryptography - Apress.

/40/ Kipnis, Aviad; Shamir, Adi. 2006. Cryptanalysis of the oil and vinegar signature scheme - Springer - Berlin, Heidelberg.

/41/ Ferozpuri, Ahmed; Gaj, Kris. 2018. "High-speed FPGA Implementation of the NIST Round 1 Rainbow Signature Scheme," 2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig).

/42/ OpenXPKI, OpenXPKI Main Page. http://www.openxpki.org/. Accessed 11 May 2023.

/43/ OpenXPKI, OpenXPKI Source Code. https://github.com/openxpki/openxpki. Accessed 11 May 2023.

/44/ EJBCA, EJBCA Main Page. https://www.ejbca.org/. Accessed 6 May 2023.

/45/ EJBCA, EJBCA Source Code. https://github.com/Keyfactor/ejbca-ce. Accessed 6 May 2023.

/46/ Dogtag, Dogtag PKI Main Page. https://www.dogtagpki.org/wiki/PKI_Main_Page. Accessed 5 May 2023.

/47/ Dogtag, Dogtag PKI Source Code. https://github.com/dogtagpki/pki. Accessed 5 May 2023.

/48/ OpenCA, OpenCA Main Page. https://www.openca.org/projects/openca/. Accessed 10 May 2023.

/49/ OpenCA, OpenCA Source Code. https://github.com/openca/openca-base. Accessed 10 May 2023.

/50/ Step-ca, Step-ca Main Page. https://smallstep.com/docs/step-ca/. Accessed 14 May 2023.

/51/ Step-ca, Step-ca Source Code. https://github.com/smallstep/certificates. Accessed 14 May 2023.

/52/ Cfssl, Cfssl Main Page. https://cfssl.org/. Accessed 1 May 2023.

/53/ Cfssl, Cfssl Source Code. https://github.com/cloudflare/cfssl. Accessed 1 May 2023.

/54/ XCA, XCA Main Page. https://hohnstaedt.de/xca/. Accessed 15 May 2023.

/55/ XCA, XCA Source Code. https://github.com/chris2511/xca/. Accessed 15 May 2023.

/56/ Easy-RSA, Easy-RSA Main Page. https://easy-rsa.readthedocs.io/en/latest/. Accessed 7 May 2023.

/57/ Easy-RSA, Easy-RSA Source Code. https://github.com/OpenVPN/easy-rsa. Accessed 7 May 2023.

/58/ Django-ca, Django-ca Main Page. https://django-ca.readthedocs.io/en/latest/. Accessed 4 May 2023.

/59/ Django-ca, Django-ca Source Code. https://github.com/mathiasertl/django-ca. Accessed 4 May 2023.

/60/ Open-Quantum-Safe, Open-Quantum-Safe Main Page. https://openquantumsafe.org/. Accessed 23 May 2023.

/61/ National-Institute-of-Standards-and-Technology, NIST Post Quantum Cryptography Round 3, 03 January 2017.

https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions. Accessed 9 May 2023.

/62/ Cloudflare, Sizing Up Post-Quantum Signatures, 08 November 2021. https://blog.cloudflare.com/sizing-up-post-quantum-signatures/. Accessed May 2023.

/63/ National-Institute-of-Standards-and-Technology, NIST Selected Algorithms 2022, 03 January 2017. https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022. Accessed 9 May 2023.

/64/ Crystals-Kyber. Kyber Main Page. 2017. https://pq-crystals.org/kyber/index.shtml. Accessed 3 May 2023.

/65/ Crystals-Kyber. Kyber Source Code. https://github.com/pq-crystals/kyber. Accessed 3 May 2023.

/66/ Crystals-Dilithium. Dilithium Main Page. 2017. https://pq-crystals.org/dilithium/index.shtml. Accessed 2 May 2023.

/67/ Crystals-Dilithium. Dilithium GitHub. https://github.com/pq-crystals/dilithium. Accessed 2 May 2023.

/68/ Falcon. Falcon Main Page. 2017. https://falcon-sign.info/. Accessed 8 May 2023.

/69/ Falcon. Falcon Source Code. 2020. https://falcon-sign.info/impl/README.txt.html. Accessed 8 May 2023.

/70/ SPHINCS+. SPHINCS+ Main Page. 2017. https://sphincs.org/index.html. Accessed 12 May 2023.

/71/ SPHINCS+. SPHINCS+ Source Code. https://github.com/sphincs/sphincsplus. Accessed 12 May 2023.

/72/ Open-Quantum-Safe, Open-Quantum-Safe Source Code. https://openquantumsafe.org/openssl. Accessed 23 May 2023.