Quantum Key Reconciliation Application

David Cobileac
cobileacd@ua.pt
Nº 102 409

Diogo Marto
diogo.marto@ua.pt
Nº 108 298

Tiago Pereira
tfgp@ua.pt
Nº 108 546

Vítor Santos
vitor.mtsantos@ua.pt
Nº 107 186

Adviser Armando Pinto and Collaborator Diogo Matos
June, 2024

Abstract

In this report we detail the implementation process of a larger QKD application project's reconciliation layer at the Instituto de Telecomunicações's Quantum Communications group, integrating it with our own expanded concept for quantum-proof secure secrets storage and telecommunications.

Keywords Quantum Key Distribution · Cybersecurity · Secrets Management · Telecommunications

Index

1 Introduction 4
1.1 Quantum Key Distribution 4
1.2 Reconciliation 4
1.3 Project Scope 4
2 Reconciliation 4
2.1 Previous & related work 5
2.2 State of the art 5
2.3 Goals 6
2.4 Project Management 6
2.5 Outcomes 7
2.6 Emulators 8
2.7 Reconciliation Integration 9
2.8 Demonstration 10
3 QeeP: Project Expansion 11
3.1 Goals 11
3.2 Paper overview 12
4 Project management 12
4.1 Applicability 12
5 State-of-the-art 12
5.1 Related Projects 12
5.2 Quantum Key Distribution Protocols 16
5.3 Technology 16
6 Requirements Analysis 16
6.1 Methodology 16
6.2 Actors 16
6.3 Functional Requirements 17
6.4 Non-functional requirements 19
6.5 Domain model 19
7 Architecture 19
7.1 Key requirements & constraints 19
7.2 Physical model 21
7.3 Interactions 22
8 Implementation 23
8.1 System Stack 23
8.2 QKD Servers & Secrets handling 24
8.3 Entity Storage & Data Warehouse 25
9 Quality Assurance 25
9.1 Quality Steps 25
9.2 Usability Testing 26
10 Results & Outcomes 27
10.1 Limitations & Next Steps: 27
11 Team Structure, Project Management & Complications 28
11.1 Team Structure 28
11.2 Project Management 28
11.3 Complications 29

12 Conclusion 30
Bibliography 31

1 Introduction

1.1 Quantum Key Distribution

Quantum Key Distribution (QKD) has been a topic of scientific research since the 1980s, starting with the BB84 protocol [1], representing a crucial advancement in secure telecommunications. QKD leverages quantum mechanics to provide near-full-proof security in cryptographic key distribution.

The usage of QKD is motivated by inherent vulnerabilities in classical communication channels, specifically, their cryptographic key distribution schemes that rely heavily on computational complexity to protect channels against brute force decryption attacks (e.g. RSA and ECC).

The impractical feasibility of these brute force attacks is, however, only applicable to current classical computers, as quantum computers can, in theory, execute particular algorithms (e.g. Shor's Algorithm) relying on the inherent properties of quantum particles to conduct attacks in an efficient timeframe. Quantum computers have yet to reach the hardware level required to execute these due to insufficient qubit numbers and high error rates - not enough to convince current (and past) governments and institutions to ignore them, as the first quantum attacks on an unprotected Internet would be catastrophic.

Alongside post-quantum cryptographic algorithms, QKD is another alternative to solve this problem by completely foregoing asymmetric cryptography and using quantum channels to distribute these keys as quantum bits - protected against eavesdroppers by the laws of physics, as evidenced in the no-cloning theorem.

1.2 Reconciliation

The physical quantum key distribution devices provide both parties at the ends of a quantum channel with raw cryptographic keys, but they cannot be directly used in symmetric encryption algorithms with the exceedingly high error rates present in quantum channels (40-50%), especially when compared to classical channel error rates (1-2%).

The reconciliation phase is where these errors are corrected, involving a myriad of different protocols to not only correct but also amplify the privacy of these keys. Reconciliation occurs between the two parties over a classical channel, sharing a minimal amount of information to arrive at useful keys without exposing enough information for an eavesdropper to perform an attack.

1.3 Project Scope

This project was initially centered around a proposal for the development of the aforementioned reconciliation layer in a larger technological stack at the neighboring Instituto de Telecomunicações of Aveiro's Quantum Communications Group.

While the inherent intricacies of reconciliation algorithms offer much in development complexity, from a broader architectural point of view, this initial scope did not align with the Projeto em Informática course's goals of exercising the knowledge acquired throughout the Software Engineering degree, mainly because we didn't have an application we could interact with.

2 Reconciliation

2.1 Previous & related work

Aveiro's University and its partners like Portuguese Quantum Communications Infrastructure (PTQCI) and Portuguese Telecommunications Institute (IT) are developing a Quantum Key Distribution Network that allows the distribution of keys based on quantum devices. This project is related to other projects such as Quantum Genomics, PTQCI, and Discretion. Our work will contribute to this project mainly in the reconciliation layer

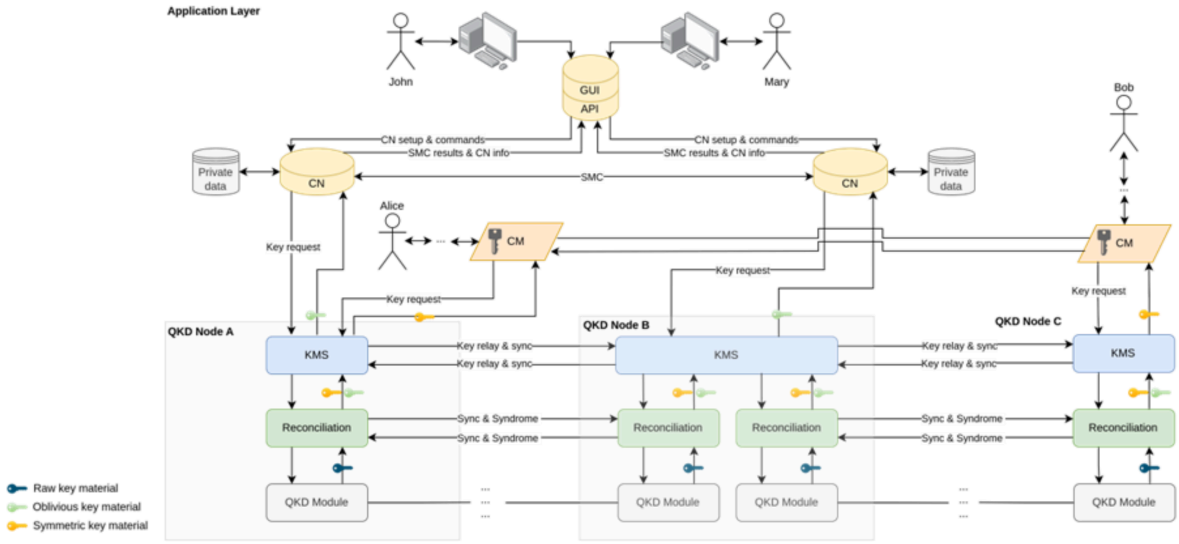


Figure 1: Overview of the QKDN being developed by IT

2.2 State of the art

In the repository we were provided with a library called NetXpto which operates under blocks and signals several versions of the reconciliation were also provided and have varying capabilities, some run on a single machine, and others run on multiple machines with different interfaces, we also created a README with the status of each version. The analysis of the NetXpto library took a significant amount of effort in the first weeks

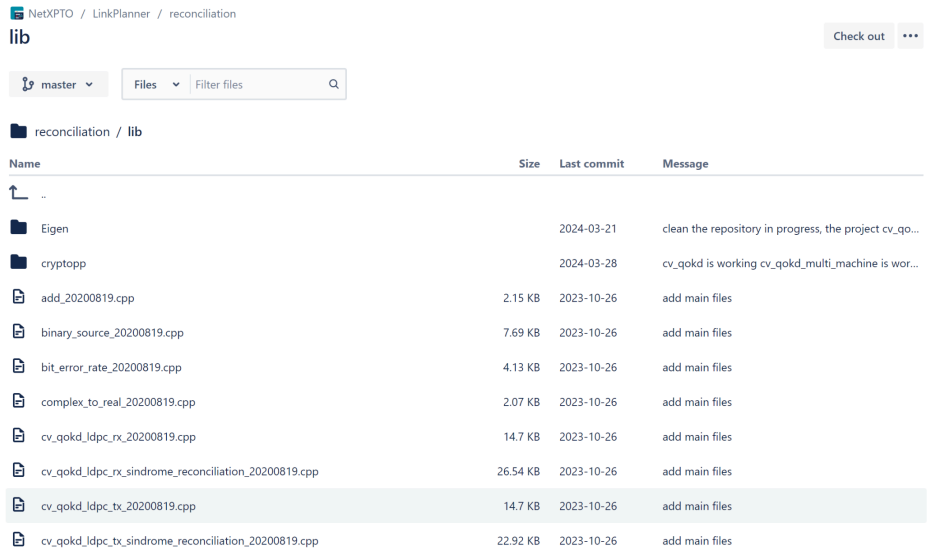


Figure 2: Netxpto library

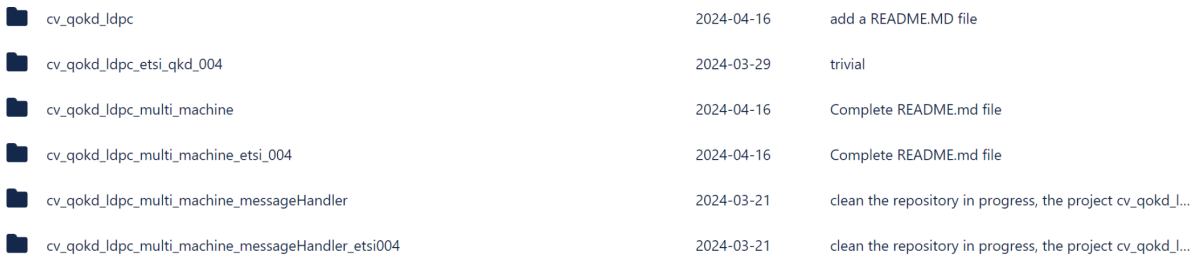


Figure 3: Previous versions of reconciliation developed by IT

2.3 Goals

The goals defined for us to accomplish were:

- Creating emulators for both 'Binary' and 'Real' modes has a way to simulate the behavior of physical devices this would allow us to test the reconciliation layer without the physical devices.
- Integrate our emulators with the reconciliation layer.
- Integrate the reconciliation layer with the KMS layer.
- Elaborate a strategy to continuously generate keys.

2.4 Project Management

Our project management strategy incorporates regular weekly meetings with our advisors to keep them updated on the project's progress and seek their guidance. We maintain a Bitbucket repository containing the entire reconciliation layer, allowing for organized version control and collaboration among team members. For communication, we utilize a Slack channel dedicated to interactions with our advisors and other project teams. Additionally, we used Discord as a platform for communication with the KMS team (KMS RECON channel).

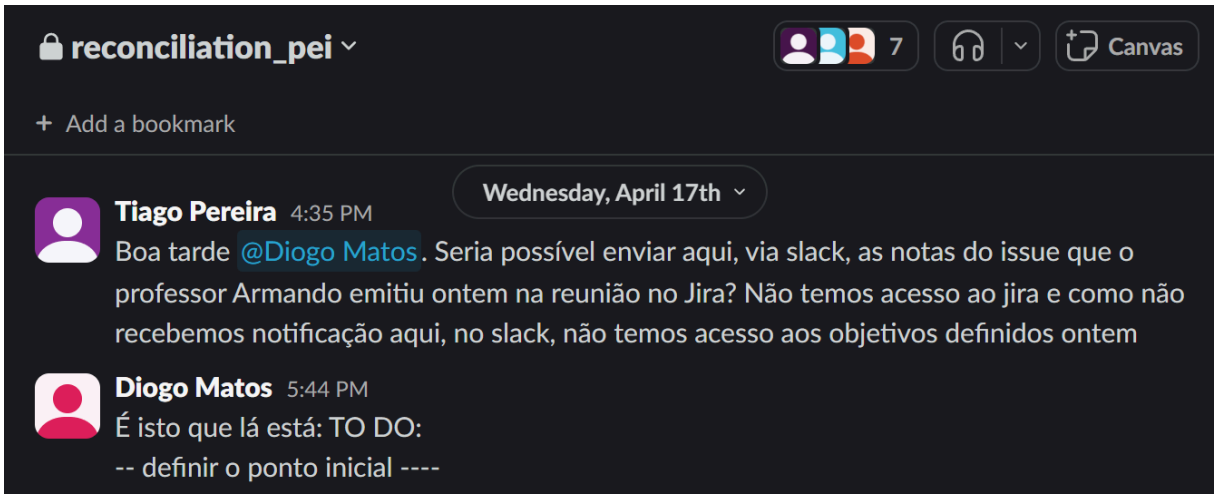


Figure 4: Slack channel used for communication between the team and advisers.

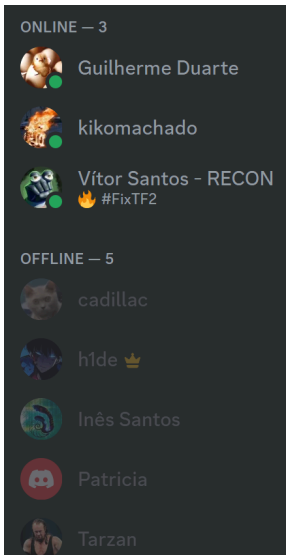


Figure 5: Members of KMS Recon discord server

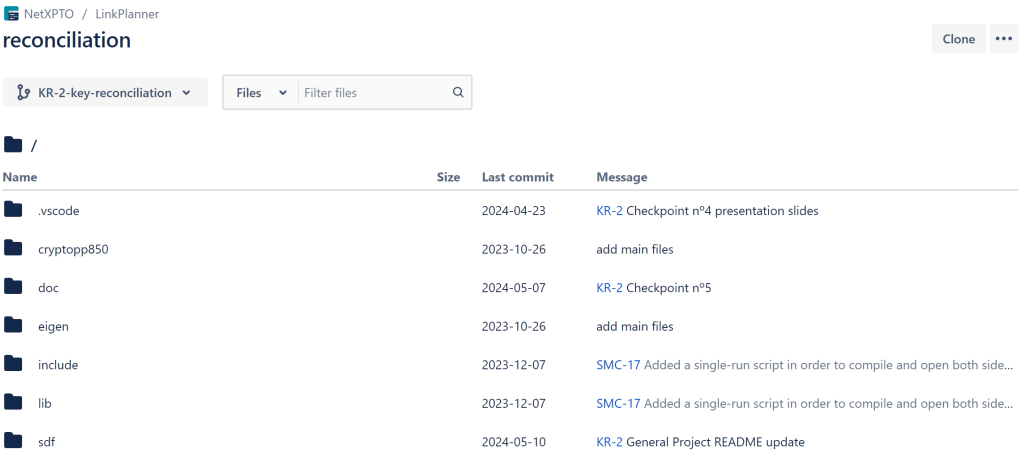


Figure 6: Reconciliation repository

2.5 Outcomes

The defined outcomes expected for this project were:

- The actual source code and implementation of the emulator and other adjacent functionality
- Documentation of our work each includes this report and several READMEs in the bitbucket repository.
- A lab demonstration to demonstrate our work
- A presentation in the IT Quantum Communications group meeting to expose our work to the rest of the IT group.

Figure 7: Slides made for IT Quantum Communications group meeting

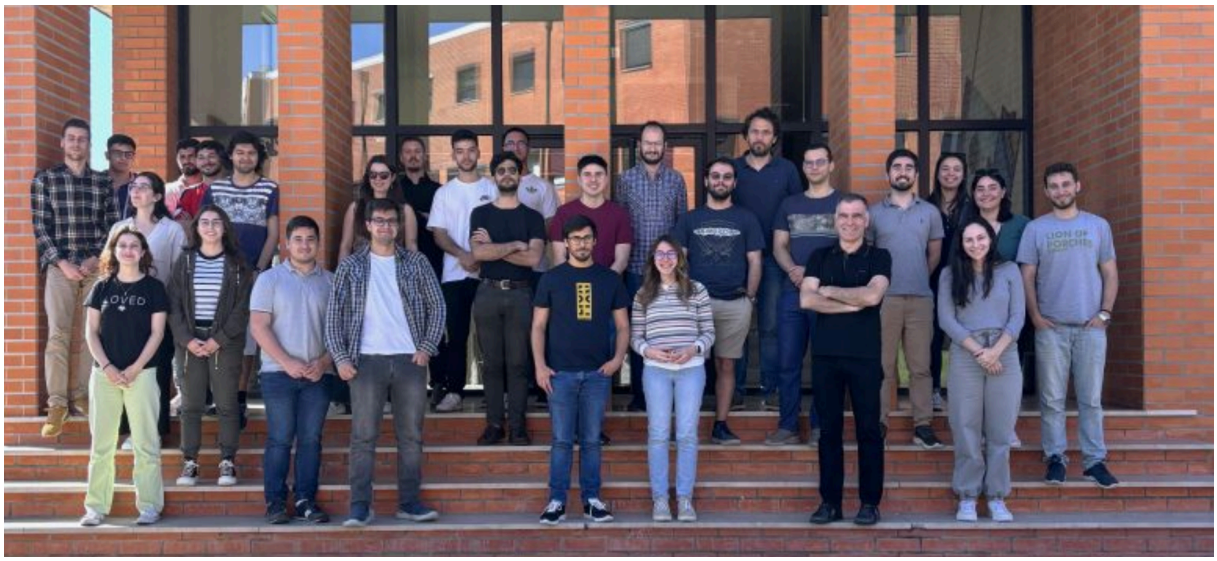


Figure 8: Picture at IT Quantum Communications group meeting

2.6 Emulators

We developed one class called *qkd_emulator* that can simulate physical QKD devices both discrete variable and continuous variable. The arguments of the simulator should be specified in the *input_parameters.txt* and they are:

- keyType: specifics the type of generated raw material either 'Real' or 'Binary'
- fileType: specifics the desired format to save the generated keys, right now it only supports 'ASCII'
- counterFirstValue: an integer that specifies the first incremental value to append to the key name.
- counterLastValue: an integer that specifies the last incremental value to append to the key name.
- txFilesName: a string that prefixes tx key names.

- rxFilesName: a string that prefixes rx key names.
- numberOfValuesPerFile: an integer number that specifies the number of values to store in each file
- seed: an integer that specifies a seed to the random number generators
- qber: a float that specifies the QBER of the generated raw key, only works on 'Binary' mode
- noisePower: a float that specifies the noise power of the noise, only works in 'Real' mode

Both modes of the simulators are implemented with the following architecture:

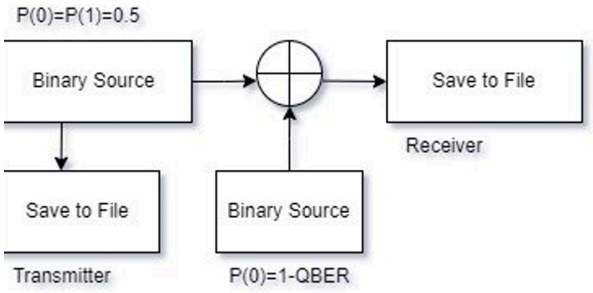


Figure 9: Binary simulator architecture

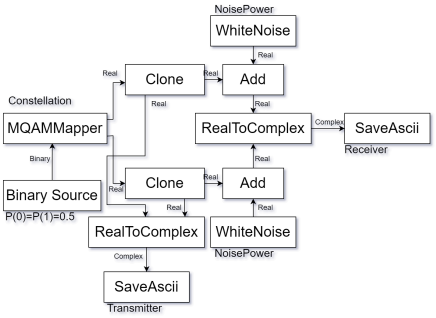


Figure 10: Real simulator architecture

Note: the defined constellation is hard coded with a value of 4.

To implement these architectures we needed to:

- Alter the block *ADD* to support operations between binary signals the implement operation is equivalent to xoring the signals.
- Alter the block *SaveAscii* to support saving complex signals. The resulting file has a number per line every odd row is the real component and even lines are the imaginary component.
- Create the block *Clone* that copies the input signal to the output signals.

We obtained the following plot by running the *plots.py* which opens the generated signals from running the simulator in 'Real' mode and plots those signals. With this plot, we can verify the correct functionality of our simulator.

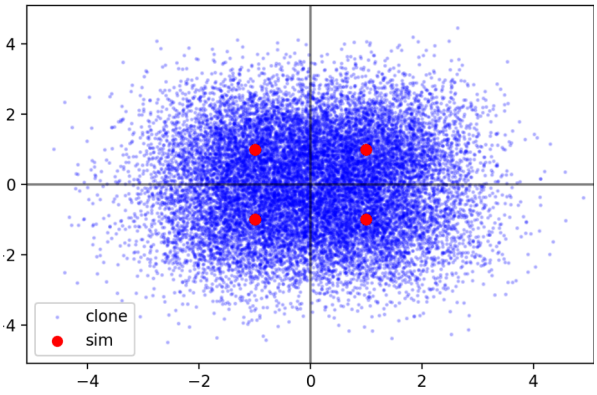


Figure 11: Results obtained from Real simulator

2.7 Reconciliation Integration

Initial versions of the KMS utilized pre-generated raw key files. Once the emulators were ready, we developed a simple server/client application to transmit raw key data from our emulator to the KMS. We then tested the system on our reconciliation layer to ensure more accurate results.

After developing a functional prototype of the reconciliation layer with our raw key emulators, our next step was to ensure it communicated seamlessly with the Key Management System (KMS). To do this, we coordinated with the KMS team to agree on an architecture for the communication between the two layers. They had already developed a key provider server class (key_provider) that simulated communication with our layer using the ETSI014 interface. Our task was to adapt this class to meet our layer's specific requirements and serve our symmetric and/or oblivious keys using the ETSI014 interface to KMS's south interface.

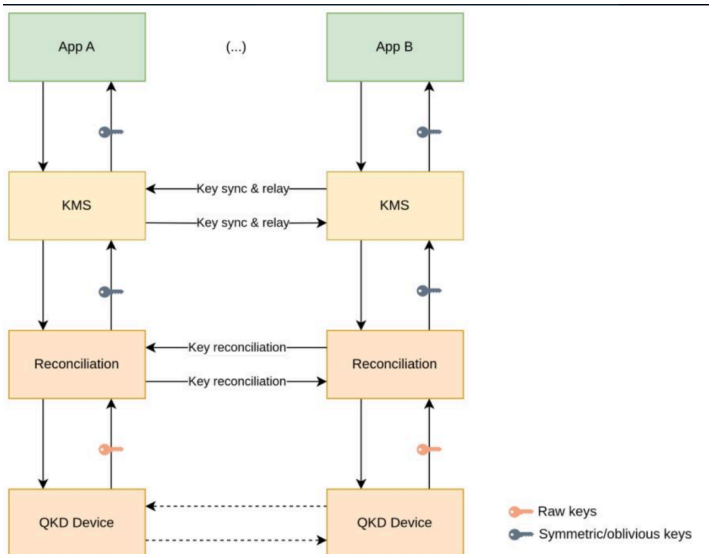


Figure 12: High-level two node QKDN architecture

2.8 Demonstration

We had a demonstration aimed to test all the system's components working together, as well as, to create an informative video for the general public about the technology and system.

It successfully processed raw keys into symmetric keys and stored them on the KMS.

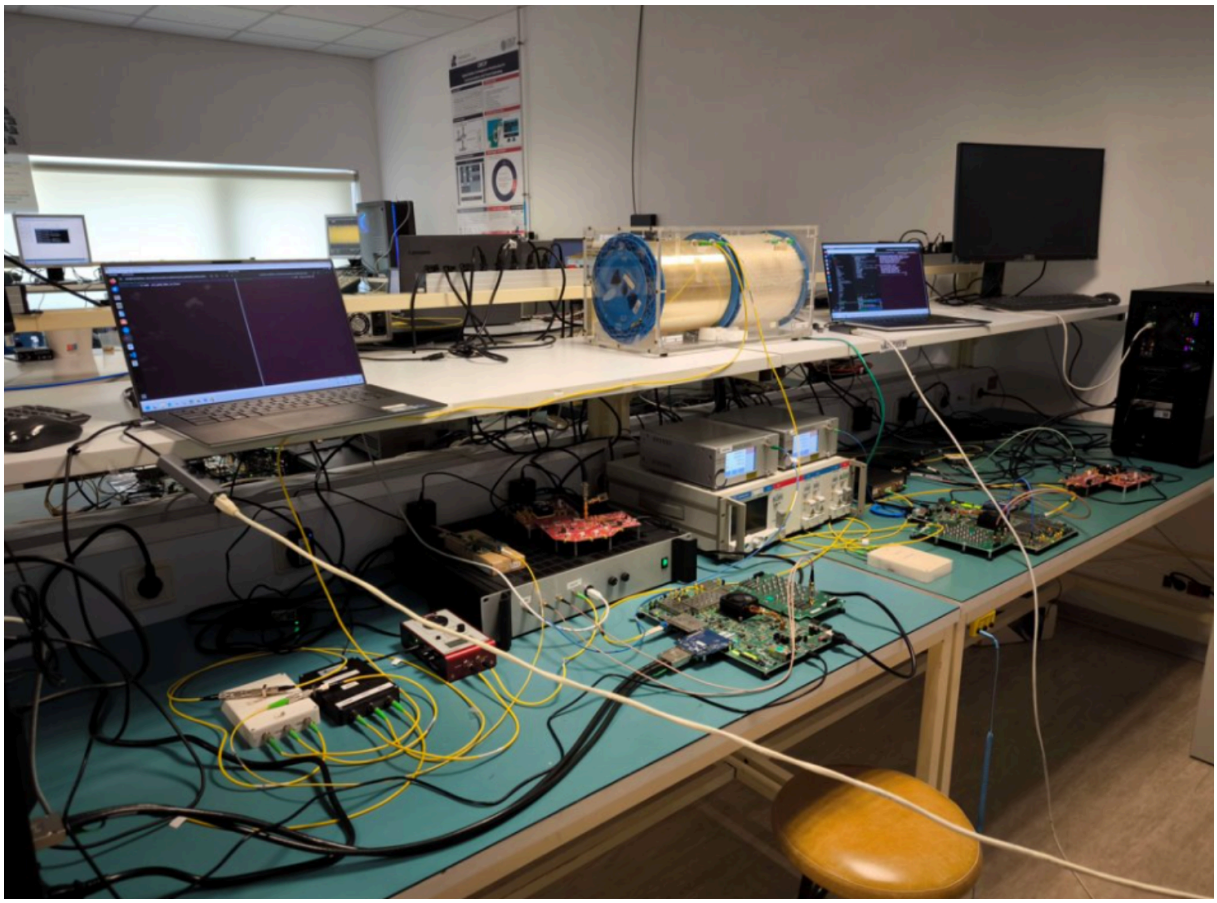


Figure 13: Picture taken at the demonstration

3 QeeP: Project Expansion

To add more value and higher-level complexity to this project, we expanded the initial scope to include a secrets storage and management application - QeeP - envisioned to bring the innovations of quantum key distribution to a broader public, comprised of both organizations and individuals.

The growing advancement and innovation in the field of quantum computing, the ever-increasing relevance of cybersecurity in the public mind, and the lack of a broadly accessible application for future-proof sensitive data handling motivated this conceptualization of this expansion.

3.1 Goals

The primary goals of this project are multifaceted, aiming to not only fulfill the requirements of the Projeto de Informática course but also to deliver a robust and secure application leveraging cutting-edge quantum technologies. The objectives include:

- Fulfilling the Projeto de Informática course's goals:
  - Familiarize students with the informatics engineering project methodology by integrating skills developed throughout the first cycle of studies.
  - Develop teamwork skills through collaborative efforts.
  - Enhance communication, documentation, and demonstration skills via presentations, reporting, and prototyping.

- Designing and implementing a full-stack application utilizing QKD:
  - Secure internal communications from quantum security threats.
  - Integrate quantum random number generators for enhanced security features.

3.2 Paper overview

The following paper presents the entire design and development process of the aforementioned project, describing the current state-of-the-art, employed methodologies, results & outcomes, problems, and conclusions.

4 Project management

The QeeP project is developed using the Unified Process Framework (UPF), specifically its variant known as the Open Unified Process (OpenUP). We chose this framework because it perfectly fits the unique requirements and complexities of integrating Quantum Key Distribution (QKD) into a BitWarden-like secrets management system. UPF's iterative and incremental approach is especially well-suited for handling the high levels of uncertainty and the need for frequent reassessment that are inherent in QKD-based projects.

4.1 Applicability

QKD technology is based on state-of-the-art quantum mechanics that involve substantial technical risks and uncertainties. The iterative nature of the Unified Process Framework enables early detection and mitigation of these risks by way of ongoing feedback and incremental enhancements. In this manner, technical difficulties are speedily addressed minimizing the risk of expensive mistakes in later stages of the project life-cycle.

5 State-of-the-art

5.1 Related Projects

To properly develop QeeP, it is important to analyze what already exists in the quantum key distribution (QKD), secrets management and secure chatting platform ecosystems. To this end, this section looks at other notable projects and technologies related to QeeP's domain by examining their distinguishing characteristics, strong points, and drawbacks. The understanding of these related works will help us determine the best practices and weaknesses in existing implementations thereby influencing the development and differentiation of QeeP.

5.1.1 Bitwarden

Bitwarden is a popular password manager that aims to provide security and convenience of use for both individuals and businesses. Bitwarden has established itself as dependable software that allows users to securely save and manage passwords using end-to-end encryption (E2EE) and an open-source policy that is completely transparent in its operations.

Differentiating Features

1. E2EE (End-to-End Encryption):

The data of all Bitwarden users is protected with E2EE, so that it is encrypted even before it leaves the user's device and remains in the same state at rest and in transit. The system uses

AES-256 for this encryption process, key derivation takes place using PBKDF2 with a master password defined by the end-user, while data integrity is checked using HMAC-SHA256 [2].

2. MFA (Multi-Factor Authentication):

Bitwarden has various MFA options like Time-based One-Time Password (TOTP) generators; email; Duo; YubiKey and FIDO2 WebAuthn which enables high account security. These provide further protection to mitigate scenarios whereby a slight compromise in the master password would allow unauthorized access into one's account [2].

3. Cross-platform Compatibility:

In addition to browser extensions for easy access and autofill abilities, Bitwarden supports web browsers as well as desktop applications (Windows, MacOS, and Linux) and mobile applications (iOS & Android). This compatibility therefore makes it possible for users to confidently access their passwords and other data from any device while maintaining a consistent level of security across these platforms.

When logging into Bitwarden from a separate device, the user must give a master password and any additional MFA actions. After that, the user downloads their encrypted vault and derives encryption keys locally to decrypt the vault data. The objective is to ensure data privacy and security regardless of the device being used.

4. Encrypted Backups:

Bitwarden enables users to safely export their vault contents. This guarantees protection for backups since it makes sure that only the user having the correct decryption key will be able to access them by encrypting these exports. Various formats can be utilized when exporting in Bitwarden which allows users to save them with confidentiality [2].

Shortcomings

Despite its strong security features, Bitwarden requires users to remember their master password, which is necessary for accessing their encrypted vault. Bitwarden's zero-knowledge architecture prevents it from recovering data if the master password is forgotten and no recovery measures are in place. Furthermore, while Bitwarden's open-source nature promotes transparency, it relies on user vigilance to successfully deploy updates and monitor security settings [3], [2].

Takeaway

Bitwarden's approach to cybersecurity could provide useful insights to QeeP, which features a secure platform for storing and managing sensitive data using Quantum Key Distribution (QKD). To ensure that its communications and data storage are secured, QeeP can adopt strong cryptographic protocols like Bitwarden employs AES-256 and PBKDF2. There is also an improvement in the safety of user account passwords by incorporating Multi-Factor Authentication (MFA) that hinders unauthorized access.

Additionally, by incorporating end-to-end encrypted backup solutions, QeeP can ensure that any user data including backups remains encrypted and secure. This preserves data privacy and integrity in case of a data breach. Furthermore, it may increase transparency and user privacy through open-source development practices together with limited metadata collection policies as means of improving security in communication. Overall, this will help gain users' trust since it reduces the chances of exposing or misusing their details.

5.1.2 WhatsApp

WhatsApp is one of the world's most popular messengers with over 2 billion active users and owned by Meta Platforms Inc. By implementing Signal Protocol for its end-to-end encryption (E2EE), WhatsApp has created a benchmark in secure messaging while striking a highly successful balance with usability.

Differentiating Features

1. End-to-End Encryption (E2EE):

WhatsApp applies the Signal Protocol to provide end-to-end encryption (E2EE) for instant messaging. Only the recipient can decrypt these messages as well as access them since they are encrypted or decrypted locally [4].

For instance, the Double Ratchet Algorithm is one of these cryptographic methods used in securing each message. In addition to this, Curve25519 is used for key agreement while AES-256 ensures encryption and HMAC-SHA256 achieves message authentication. Furthermore, WhatsApp ensures forward secrecy by frequently rotating these encryption keys. This means that all conversations held using WhatsApp provide complete secrecy between users since even if a key is disclosed it cannot be used to decrypt past or future messages which will keep communication intact. Green and Moxie Marlinspike (2018), argue that perfect secrecy can be achieved through frequent rotation of such keys thereby safeguarding against compromise of a particular key that could lead to past or future communication snooping.

2. Multi-Factor Authentication (MFA):

Regarding the security of the account, WhatsApp has an option for two-step verification. This enables users to set a six-digit PIN that must be entered when they activate WhatsApp on their phone with the same phone number again. Unauthorized access can be prevented by this additional layer of security [5].

3. Cross-Platform Compatibility:

WhatsApp also offers a compatible interface across various platforms such as iOS, Android, Windows, and macOS in the form of WhatsApp Web, WhatsApp Desktop, and mobile applications. More often than not, users will log in with their accounts on different devices, prompting a mechanism for securely transferring message history without compromising the E2EE principle of the app.

When one connects another device to WhatsApp by scanning a QR code with the primary device, this begins a secure key exchange using public key cryptography. WhatsApp creates session keys to encrypt and decrypt messages that use ephemeral keys for forward secrecy so that even if the session key has been breached, previous messages are still protected. This is done via prekey bundles which are used in the initial key exchange process and support asynchronous messaging, with regularly updated encryption keys. The primary device securely synchronizes encryption keys with new devices, maintaining end-to-end encryption across all linked devices [5].

4. Backup Encryption:

WhatsApp users can back up their chats to cloud services like Google Drive for Android and iCloud for iOS (WhatsApp, 2021). This is because it recently launched cloud-hosted E2EE backups, which can only be accessed by the user with the correct encryption key. The key is

generated from a password entered when E2EE backups are enabled, using a key derivation function (KDF) such as PBKDF2 and salting. WhatsApp might produce this key at random, with no correlation to passwords that you, as an end user, must remember and be accountable for all on your own. As a result, there are specific dangers connected with any lost or forgotten keys [5].

Shortcomings

WhatsApp's reliance on phone numbers for registration can limit user anonymity and pose problems if the number is lost or changed. Despite robust E2EE, its centralized infrastructure, controlled by Meta Platforms Inc., subjects the service to company policies and legal pressures that may impact privacy. Additionally, while WhatsApp minimizes message content collection, it still collects metadata, such as communication logs, which can reveal user behavior and connections and raise privacy concerns [6], [4], [7].

Takeaway

End-to-end encryption (E2EE) and comprehensive cyber security measures, such as those used by WhatsApp, can be extremely beneficial to the QeeP project, which comprises of a basic messaging platform with Quantum Key Distribution (QKD) for secure communication. It is critical to understand and apply robust cryptography protocols. For example, adopting strong cryptographic methods such as those used in Signal Protocol will secure QeeP messages from numerous types of attacks. Installing MFA, or Multi-Factor Authentication, could also improve user account security by adding another layer of defense against unauthorized access.

QeeP can also benefit from implementing end-to-end encrypted backup systems, which ensure that all user data, including backups, is protected and only accessible by authorized users. This technique ensures that data integrity and privacy are protected even in the case of a data breach. Adopting limited metadata collection standards can also significantly improve user privacy, matching QeeP with best practices in secure communications. By limiting metadata collection, QeeP can lower the chance of sensitive user information being revealed or exploited, hence increasing overall user trust and security.

5.2 Quantum Key Distribution Protocols

- Most modern up-to-date QKD protocols being used and deployed in the industry

– Including reconciliation algorithms and protocols
- Each protocol's motive to exist, the advantages it brings, and potential shortcomings
- The selected QKD protocols for this project, based on presentations

5.3 Technology

- Underlying technological hardware that supports quantum key distribution

6 Requirements Analysis

6.1 Methodology

In the pursuit of defining and refining the requirements for QeeP's project expansion, we adopted a structured approach grounded in the principles of use case analysis and thorough research. Our methodology encompasses the following steps:

- **Use Case Analysis Methodology**: We employed a use case analysis approach to identify and articulate the functional requirements of the QeeP application expansion. This methodology allows us to capture the interactions between users (actors) and the system, providing a comprehensive understanding of system behavior.
- **Research on Target Users and Project Domain**: Understanding the needs, preferences, and behaviors of our target users is paramount. Through thorough research, we delved into the project domain, exploring the intricacies of how quantum key distribution an secrets management can work together to further improve everyday lives. Our design decisions take into consideration the collected user expectations.
- **Determining and Describing Actors**: Actors represent the various entities interacting with the QeeP system. These may include individual users, organizations, administrators, and external systems. By identifying and describing these actors, we laid the foundation for defining use cases and understanding their roles within the system.
- **Modeling Functional Requirements as Use Cases**: Functional requirements describe the specific behaviors and functionalities expected from the system. We modeled these requirements as use cases, depicting how actors interact with the system to achieve their goals. Each use case encapsulates a specific functional requirement, providing clarity and structure to the development process.
- **Determining Non-functional Requirements**: In addition to functional requirements, we identified and prioritized non-functional requirements that define the system's quality attributes. These include not only performance, reliability, scalability, and usability criteria but, mainly, security. By addressing these requirements early in the development process, we ensure that the QeeP application meets the highest standards of excellence.

6.2 Actors

In the QeeP project expansion, various actors interact with the system, each with distinct roles and responsibilities tailored to their needs. The identified actors are as follows:

1. **Individual User**: The individual user represents the typical end-user of the QeeP application, seeking to leverage its basic secret management and messaging features. This actor interacts with the system to securely store and manage personal secrets, as well as engage in encrypted communication with other users through the integrated chat system.
2. **Organizational User**: Organizational users comprise entities such as businesses, institutions, or groups that utilize QeeP's advanced features tailored to organizational needs. These users benefit from enhanced functionalities designed to facilitate secure collaboration, data sharing, and access control within their respective organizations.
3. **Organizational Administrator**: Organizational administrators hold privileged roles within their respective organizations, responsible for managing and overseeing the use of QeeP's advanced features within their organizational context. This actor has access to administrative tools and functionalities for configuring user permissions, managing organizational vaults, and enforcing security policies.
4. **QeeP Administrator**: The QeeP administrator is tasked with the management and maintenance of the QeeP website and underlying infrastructure. This actor ensures the smooth operation of the platform, handles statistical data, monitors system performance, and addresses any technical issues that may arise.
5. **QeeP Personalized Role**: QeeP provides an advanced customizable group/role system, allowing Owners to define custom permissions to the organization and to each organization vault individually. This allows, for example, a client like UA to create personalized Organizations, each representing a Department, and each containing their respective Vaults, each representing a subject, with Administrators with the roles **Professor**, **Regente**, **Investigador**, etc. Each one of these roles can possess distinct access permission at the organization management level, similar to the business model that UA adopts in its everyday operations.

Each actor plays a vital role in the ecosystem of the QeeP application, contributing to its functionality, usability, and security in their respective capacities. By understanding the roles and responsibilities of these actors, we can effectively design and implement features that cater to their specific needs and requirements.

The identified actors encompass a spectrum of user roles, ranging from individual users to organizational administrators, each with distinct responsibilities and access levels within the QeeP application.

6.3 Functional Requirements

6.3.1 Authentication Package

6.3.1.1 Logging In

All actors must be able to authenticate themselves securely to access the QeeP application. *Disclaimer*: This is an automated process that occurs in a synchronous manner when a user starts the QeeP application.

6.3.1.2 Signing Up

New users, including individual users, organizational users, and administrators, must be able to create accounts with appropriate credentials. This is done in a presential manner. However, for PI practicality, we emulate this process through an API endpoint.

6.3.2.1 Secrets

Package related to operations over Sensible Information.

- **Storing**:

All actors should be able to securely store sensitive information (secrets) within the QeeP application.

- **Retrieving**:

All actors must have the ability to retrieve stored secrets, ensuring seamless access to their encrypted data.

- **Deleting**:

All actors should be able to delete unwanted or obsolete secrets from their vaults, maintaining data hygiene and privacy.

6.3.2 Organization Management Package

Package related to Organization Operations.

- **Vault Operations:**

Organizational administrators must have the capability to create, configure, and manage organizational vaults, which may include setting user access permissions and encryption policies.

- **Inviting Users and Organizations:**

Organizational administrators should be able to invite new users to join their organization, as well as, invite a new organization to share a vault, facilitating seamless collaboration and data sharing within the organizational context.

6.3.3 Messaging Package

Package associated with operations over the quantum chat system.

- **Sending Simple Text:**

All actors should be able to exchange encrypted messages containing simple text, ensuring secure communication within the QeeP application. Device Linking:

- **Loading Message History onto Different Devices**:

All actors must have the ability to link multiple devices to their QeeP accounts and synchronize message history across these devices, ensuring continuity and accessibility of communication.

These functional requirements delineate the core functionalities of the QeeP application, encompassing authentication, secrets management, organization management, and messaging capabilities tailored to the needs of individual users, organizational users, and administrators.

6.4 Non-functional requirements

The following list presents requirements specifying non-functional requirements, i.e. a description of a property or characteristic that a software system must exhibit or a constraint that it must respect.

6.4.1 Security

Related to the fact the system focuses on secrets management and encrypted communication, comes security concerns. With a system like the one being developed, there are many security concerns to consider. One of them is about controlling access to the personal and organizations' vaults, private data will be available on the web application and it should only be accessed by an authorized user; therefore, a role-based access control module should be implemented. We also need to ensure that no keys are leaked by defining a secure protocol for the distribution of the same.

6.4.2 Usability

Usability requirements focus on the user experience and how easily users can interact with the system. Regarding the web app, the application should have a clean, intuitive, and responsive user interface that simplifies navigation and usage for all users.

6.4.3 Reliability

The simple fact that the system deals with user and organization's secrets, has a direct impact on the importance of the system operating without failing. Error handling must be treated carefully and, in case of system errors, the web application should be responsive enough so the user is not led into misunderstandings and abandoning of the system.

6.5 Domain model

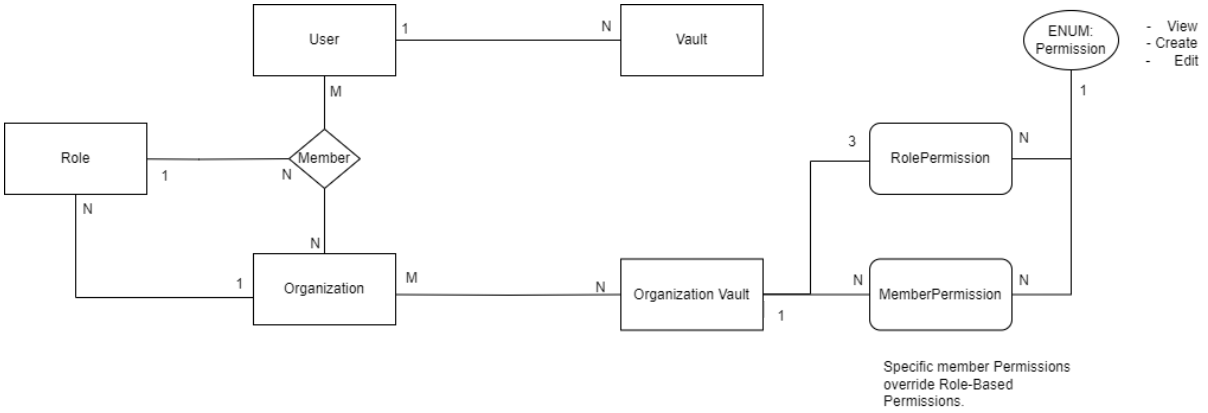


Figure 14: Domain model

7 Architecture

7.1 Key requirements & constraints

7.1.1 QKD Scalability

Quantum key distribution networks (QKDN) normally face problems in scaling communication distances, as quantum channel noise and errors transfigure the raw key beyond practical use in

reconciliation, requiring intense computational resources to process into a usable output [8]. In CV-QKD systems, the maximum effective practical range between two nodes is far from certain and depends on various environmental factors, but it does not tend to exceed 50 km [9], [10]. This error rate over distance also has an exponential tendency.

Quantum repeaters are a solution to this, but they're not commercially available [11]. Authors have suggested the use of secure quantum relay nodes using post-quantum cryptography [12].

7.1.2 Real-time messaging

In order to guarantee a secure and efficient real-time messaging system for QeeP, Quantum Key Distribution (QKD) must be used to provide strong encryption and protection against quantum threats. In handling huge message traffic with minimal latency, performance and scalability play crucial roles. However, QKD introduces time delays due to the nature of the quantum states and the need for error correction; this can be dealt with by strategies such as high-bandwidth quantum channels and parallel key distribution. In practice, ensuring that it interfaces with external systems is done through standard APIs and protocols allowing seamless integration thereby enhancing functionality and user experience. To enable robust performance tracking, reliability must be a prime consideration in implementing monitoring tools that will make the messaging system of QeeP secure, and efficient, and meet the needs of today's communications [13].

7.2 Physical model

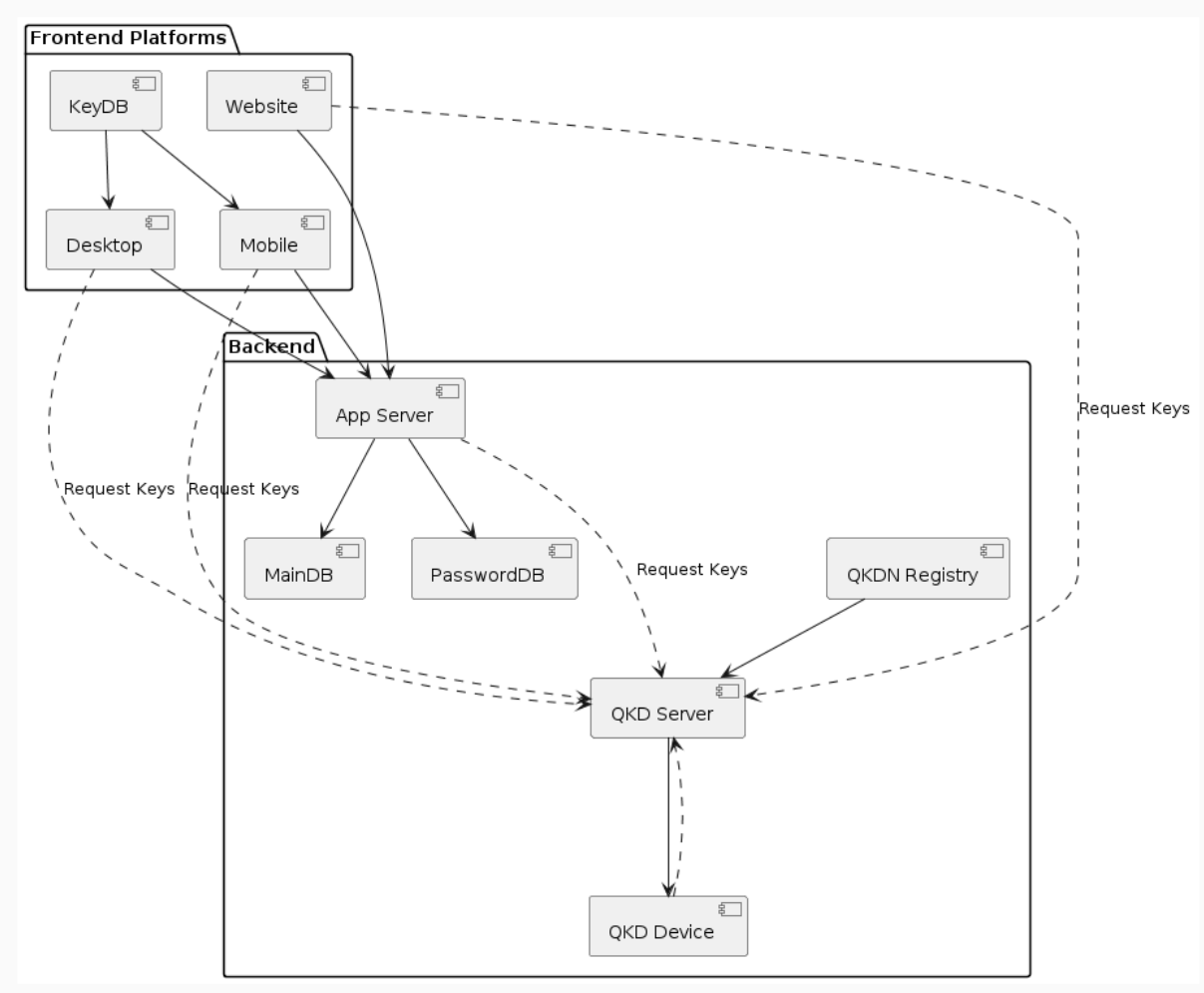


Figure 15: Physical Model

Website / Desktop / Mobile: The frontend platform for all GUI end-user interactions.

KeyDB: Encryption keys for E2EE messaging and secrets storage are stored locally, using the browser's localStorage API or a minimal DBMS like SQLite for desktop and mobile applications. 0′« **PasswordDB:** For storing E2EE messaging session keys, E2E-encrypted at the user level using the master key. The PasswordDB is part of QeeP's internal infrastructure. Users communicate with this database when messaging session keys need to be exported to a new authenticated device.

App Server: Central server for most business end-user operations, including account authentication, CRUD secrets management operations and others. It supports an internal Websocket component for real-time messaging.

QKD Server: Server wrapper on the QKD physical device used to serve quantum symmetric keys to other backend components.

QKDN Registry: Reliable, dynamic, cached service for looking up a respective user's QKD server to begin the key exchange.

MainDB: Database for account information details, E2EE message history, and stored secrets.

7.3 Interactions

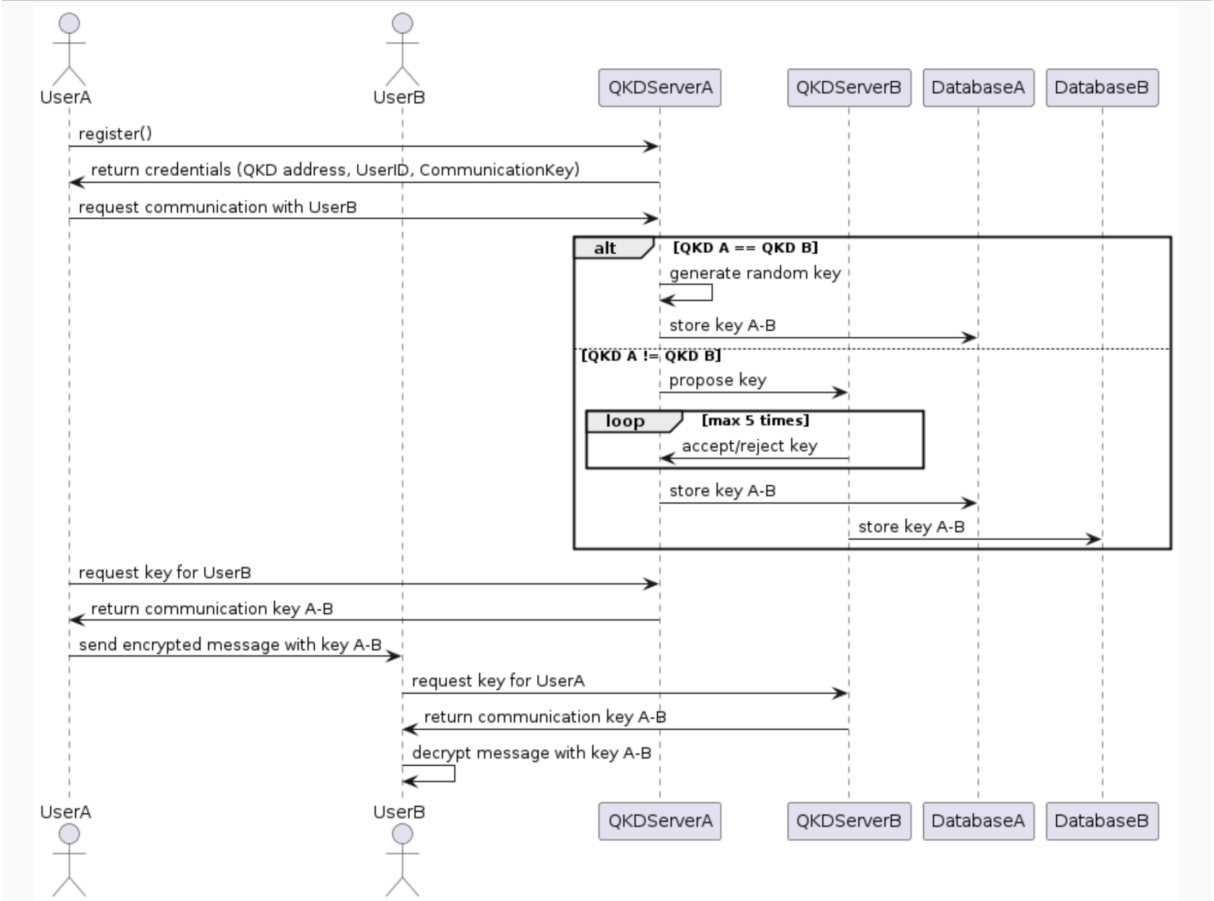


Figure 16: Flow illustring how user A can be communicate with user B safely.

8 Implementation

8.1 System Stack

In our quest to deliver a robust, widely distributed, and cross-platform-available application within our constrained timeframe, the selection of an appropriate system stack was paramount. We aimed for a stack that strikes a delicate balance between functionality, automation, and ease of use, ensuring efficiency and effectiveness in the development process. Moreover, given the nature of our project, which demands broad accessibility across multiple platforms, we sought technologies that could seamlessly accommodate such requirements. Consequently, after careful consideration, we decided to leverage the following tech stack for our main system workflows:

- **Ionic-React**: As the cornerstone of our front-end development, Ionic-React emerged as an optimal choice due to its unparalleled support for multi-platform deployment and compilation. With Ionic, we could develop applications that run smoothly on various platforms, including mobile devices, web browsers, and desktop environments, all from a single codebase. This aligns perfectly with our architectural requirements, facilitating widespread distribution and accessibility for our users. Additionally, Ionic's rich ecosystem of pre-built components, coupled with its intuitive development framework, empowers our team to rapidly prototype and iterate on the user interface, ensuring a seamless and engaging user experience.

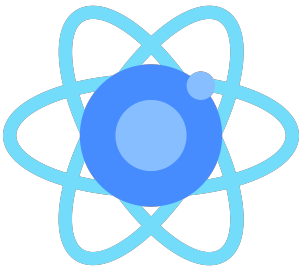


Figure 17: Ionic React Framework

- **Django**: Powering our back-end entities and handling sensitive data management and storage operations, Django stood out as the ideal framework to meet our project's demands. With Django's robust and comprehensive feature set, we gain access to a plethora of tools and functionalities that streamline the development of complex back-end systems. From user authentication and authorization to database management and RESTful API creation, Django offers a versatile and efficient platform for building scalable and secure web applications. Moreover, Django's emphasis on best practices, coupled with its extensive documentation and vibrant community, ensures that our development team can navigate the intricacies of back-end development with ease, reducing development time and minimizing potential pitfalls.

Figure 18: Django Python Framework

By embracing Ionic-React for our main front-end workflows and Django for our back-end infrastructure, we possess the ability to deliver a high-quality, highly scalable, feature-rich, comprehensive system that meets the diverse needs of our users.

8.2 QKD Servers & Secrets handling

- **FAST API for QKD Servers API**: To facilitate seamless communication with QKD servers, we employ FAST API, a modern web framework for building APIs with Python. FAST API's asynchronous capabilities and performance optimizations make it an ideal choice for handling the intricate demands of quantum key distribution. With FAST API, we can develop highly responsive and efficient API endpoints that facilitate secure communication between the QKD servers and the QeeP application, allowing QKD servers to act also as a **DNS Server** and **KMS**.

Figure 19: SQlite embedded database

- **SQLite for QKD Servers Addresses & Keys Databases**: Leveraging SQLite for storing QKD server addresses and encryption keys ensures lightweight and efficient data management. SQLite's serverless architecture and minimal overhead make it an excellent choice, providing fast and reliable access to critical information required for the system quantum key distribution operations, acting as the storage unit for both the KMS & DNS embedded server.

Figure 20: FastAPI framework

8.3 Entity Storage & Data Warehouse

In addition to the system stack outlined above, we employ specialized technologies for entity storage and data warehousing to ensure robustness and scalability:

- **Entities in Postgres Cluster**: Utilizing a Postgres cluster for entity storage ensures reliability and scalability for our application's data. Postgres's robust relational database management system offers ACID compliance, data integrity, and extensive support for complex queries, making it suitable for storing critical application data such as user profiles, organizational information, and statistical information.
- **Warehouse in Separated Postgres Cluster**: For data warehousing and analytical purposes, we leverage a separate Postgres cluster optimized for data aggregation, reporting, and business intelligence. By segregating the warehouse from the primary entity storage, we ensure optimal performance and scalability for both transactional and analytical workloads, enabling comprehensive data analysis and decision-making capabilities.

By incorporating these specialized technologies into our system architecture, we lay the groundwork for a highly performant, scalable, and secure application that meets the diverse needs of our users and stakeholders. Through careful selection and integration of each component, we aim to deliver a cutting-edge solution that exceeds expectations and drives tangible value for our users and the broader community.

Figure 21: Postgres DBMS

9 Quality Assurance

Disclaimer: This topic will also revolve around the reconciliation part of the project.

9.1 Quality Steps

In our Quality Assurance process for the informatics project, we meticulously ensured the integrity and excellence of our work through the following measures:

- **Adherence to Defined Standards:**

We meticulously followed defined and standardized standards throughout the project's development, ensuring consistency, clarity, and compliance with industry best practices.

- **Implementation Details Reconciliation and Peer Review:**

We engaged in thorough discussions with supervisors to reconcile implementation details, ensuring alignment with project objectives and specifications. Additionally, peer review sessions were conducted for both the reconciliation process and the QeeP project, facilitating constructive feedback and refinement of our approaches.

- **User Stories Satisfaction and Usability Testing:**

We prioritized user satisfaction by rigorously assessing the fulfillment of user stories and conducting extensive usability testing(to be discussed in the following sub-topic). This allowed us to validate the functionality and usability of our solutions, identifying areas for improvement and optimization.

- **Github Repository and Version Control:**

We utilized **Github** as our primary repository and version control system, enabling seamless collaboration, version tracking, and code management throughout the project's lifecycle.

Figure 22: Github Version Control and Repository

- **Standards in External System Interfaces:**

We implemented standards in external system interfaces to ensure compatibility and interoperability with a wide variety of end applications, particularly for our multi-platform front-end solution.

- **Regular Meetings and Feedback Incorporation:**

Weekly meetings with advisers provided opportunities for progress updates, feedback exchange, and guidance. In the QeeP project, consultations with subject matter experts and professors were instrumental in refining our approach. Furthermore, feedback from checkpoints and milestone presentations in the Projeto de Informática class informed iterative improvements to our work.

- **Continuous Improvement:**

By actively gathering feedback from various sources, including advisers, teachers, and usability tests, we iteratively implemented changes aimed at enhancing the quality and effectiveness of our product. These continuous improvements were reflected in successive usability tests and positive feedback received from stakeholders.

Through these comprehensive quality assurance practices, we ensured the robustness, usability, and alignment of our informatics project with stakeholder expectations and industry standards.

9.2 Usability Testing

In order to receive feedback on our expansion proposal throughout the development process, we conducted several usability testing sessions.

- **Sample Group**:

1. **Number of Reviewers**: 8.

2. **Ages**: **21 - 63** (approx.).
3. **Reviewers Technological Expertise**: Ranged from beginners to experts.

Main Takeaways: Simplify UI, mainly in the organization's dashboard. Improve color schemes. Abstract specific, complex operations from being **1-click away**.

10 Results & Outcomes

Our informatics project culminated in the development of a cutting-edge **Quantum Chat System** & **Quantum Vaults** in a Highly-Modular **Secret Data Management Ecosystem**. These achievements are a representation of the significant advancements in the field of quantum computing and secure data management, laying the groundwork for future innovations and applications. Below, we outline the key results and outcomes of our project:

1. **Quantum Chat System:** Our Quantum Chat System leverages the principles of quantum key distribution to ensure secure and confidential communication between users. By harnessing the power of quantum cryptography, we have created a messaging platform that offers unprecedented levels of security, protecting sensitive information from eavesdropping and interception.
2. **Quantum Vaults:** The Quantum Vaults feature of our project provides users with a secure and reliable means of storing their sensitive data. Utilizing quantum encryption techniques, we have developed a robust vault system that safeguards confidential information against unauthorized access and cyber threats. This ensures the confidentiality and integrity of user data, even in the face of evolving security challenges.
3. **Highly-Modular Secret Data Management Ecosystem:** Our project also encompasses a Highly-Modular Secret Data Management Ecosystem, designed to cater to the diverse needs of users and organizations through customizable **roles** and **permissions**. This ecosystem offers a comprehensive suite of tools and functionalities for managing, organizing, and accessing secret data securely. From password management to file encryption, our solution provides a versatile platform for safeguarding sensitive information across various use cases.

10.1 Limitations & Next Steps:

Despite the significant progress made in our project, certain limitations and areas for improvement remain to be addressed in future development iterations. These include:

- **Organizations Vaults Not Quantum Ciphered:** Due to time constraints, organizations' vaults are not currently quantum-ciphered. Future iterations of the project will focus on implementing quantum encryption for organizational data storage, enhancing security and confidentiality.
- **Incomplete Update/Delete Operations:** Some functionalities, such as update and delete operations, are not fully implemented in the current version of the application. These features will be prioritized in future development cycles to improve the user experience and data management capabilities.
- **Quantum Password Generator:** The implementation of a password generator feature, aimed at enhancing password security and convenience, remains pending. This feature will be incorporated in subsequent iterations to provide users with a comprehensive set of tools for secure data management.

- **Master Secret Container:** The creation and distribution of end-users' master secrets containers in the form of QR Codes, SmartCards, Passwords, etc, defined at the moment the Users register physically on our Platform.

11 Team Structure, Project Management & Complications

11.1 Team Structure

We adopted a **AGILE** methodology throughout the whole development process, with each team member possessing a specific role and responsibilities.

Team Leader: *Tiago Pereira*. Leads the planning of the project, coordinates interactions with the stakeholders, and keeps the project team focused on meeting the project objectives.

Architect: *Vítor Santos*. Coordinates the technical design of the system and has overall responsibility for facilitating the major technical decisions expressed as software architecture.

DevOps: *Diogo Marto*. Responsible for the combination of practices and tools that create an infrastructure that increases the team's ability to deliver applications and services at high velocity, without sacrificing quality.

Quality Assurance: *David Cobileac*. Responsible for ensuring the quality and reliability of the delivered system.

11.2 Project Management

We used **Discord** as our primary team communication platform, as well as **JIRA** for task distribution and **Github** for code versioning and storage.

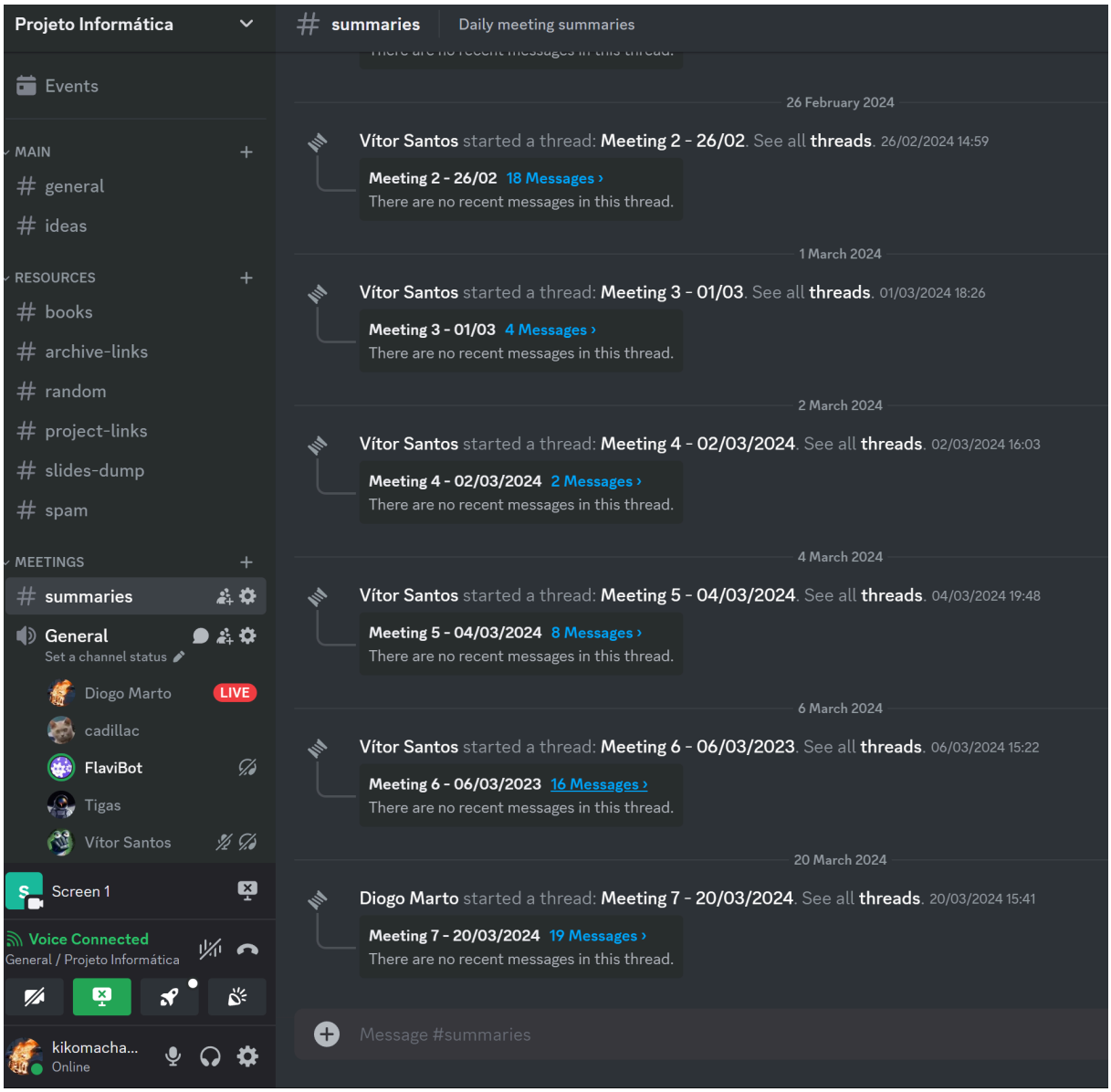


Figure 23: Discord Server used for communication between members.

11.3 Complications

Throughout the project, we found ourselves against 2 main complications:

1. **Confusion with relation to the Project Objective**. During the first weeks, development was delayed since meetings with the advisor were mainly about the desired solution context, and not so much about its intrinsic properties.

2. **Reconciliation Repository High Complexity**. Our given started code base, developed with **cpp** proved to be a special challenge, not only due to language complexity but also due to unfamiliarity with a large code source developed by different people over multiple iterations.

3. **Lost of a Team Member.** Halfway through the project, one member simply disappeared without any kind of previous indication and didn't respond to multiple communication attempts. This reduced our man-power and demanded changes in the team-paradigm,

readjusts on task distributions and deliveries calendar. This member has since been removed from our project.

12 Conclusion

In conclusion, our informatics project has achieved significant milestones in the development of quantum-based communication and data management solutions. Through the Quantum Chat System & Quantum Vaults, present in a Highly-Modular Secret Data Management Ecosystem, we have demonstrated the potential of quantum technologies in addressing modern cybersecurity challenges. Moving forward, we are committed to addressing the identified limitations and advancing our project to new heights, ultimately delivering innovative solutions that empower users to protect their sensitive information in an increasingly digital world.

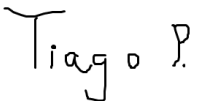
Though we were faced with unanticipated challenges, we are happy with our performance, especially taking into account the time context around the project expansion development, the reduction in man-power, and the high complexity of the project given to us.

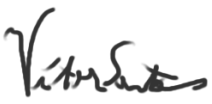
This project helped us better understand the importance and significance of engineering to better everyday lives by solving everyday problems, such as secure information exchange between points **A** and **B**.

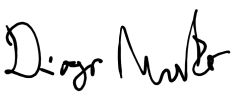
Was especially fulfilling to observe the culmination of capabilities acquired over the last **3 years**.

Our area of engineering is indeed extremely large. Communication and collaboration are essential to achieve success, both at the project and professional level. This course and project helped us to understand precisely that.

Our Team,







Bibliography

[1] C. H. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *Theoretical Computer Science*, vol. 560, pp. 7–11, 2014, doi: https://doi.org/10.1016/j.tcs.2014.05.025.

[2] Bitwarden, "Bitwarden Security Whitepaper." 2021.

[3] Bitwarden, "Bitwarden Privacy Policy." 2021.

[4] M. Marlinspike and T. Perrin, "The Signal Protocol." 2016.

[5] WhatsApp, "WhatsApp Privacy Policy." 2021.

[6] S. Foundation, "Signal Privacy Policy." 2021.

[7] M. Green and M. Marlinspike, "Formal Verification of the Signal Protocol." 2018.

[8] C.-W. Tsai, C.-W. Yang, J. Lin, Y.-C. Chang, and R.-S. Chang, "Quantum Key Distribution Networks: Challenges and Future Research Issues in Security," *Applied Sciences*, vol. 11, no. 9, p. 3767–3768, 2021, doi: 10.3390/app11093767.

[9] T. Wang, P. Huang, H.-L. Yin, W. Zhang, and G. Zeng, "High key rate continuous-variable quantum key distribution with a real local oscillator," *Optics Express*, vol. 27, no. 1, pp. 262–274, 2019, doi: 10.1364/OE.27.000262.

[10] D. Huang, P. Huang, D. Lin, and G. Zeng, "Long-distance continuous-variable quantum key distribution by controlling excess noise," *Scientific Reports*, vol. 6, p. 19201–19202, 2016, doi: 10.1038/srep19201.

[11] C. Cicconetti, L. Maccari, and G. Bianchi, "Quality of Service in Quantum Networks," *arXiv preprint arXiv:2204.09538*, 2022, [Online]. Available: https://arxiv.org/abs/2204.09538

[12] S. Pirandola *et al.*, "Advances in Quantum Cryptography," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 547–615, 2020, doi: 10.1109/COMST.2019.2961423.

[13] S. Ayoade, P. Lio, D. Gill, F. Aldoseri, and F. Awan, "Quantum Cryptography for Enhanced Network Security: A Comprehensive Survey of Research, Developments, and Future Directions," *Journal of Analytical Science and Technology*, 2022, [Online]. Available: https://jast-journal.springeropen.com/articles/10.1186/s40543-022-00302-0