



# NETWORK SECURITY **CRYPTO CHAT**

BY  
ALI JAFARI  
SEYYED MOHAMMAD SAJADI




# INTRODUCTION

The purpose of this report is to provide an overview of the **CryptoChat** developed using the Qt framework. **CryptoChat** is an E2EE chat application. The application was developed as a part of Network Security Course, with the goal of applying up to date encryption methods.

The report begins with a brief overview of the application's functionality and user interface. This is followed by a discussion of the software design and architecture, including the tools and technologies used to develop the application. The report also includes a discussion of the challenges encountered during the development process and how they were overcome.

Finally, the report concludes with a reflection on the development process and the lessons learned. This includes a discussion of the strengths and weaknesses of the application, as well as suggestions for future improvements and enhancements.

Overall, this report provides a comprehensive analysis of the **CryptoChat** and serves as a valuable resource for anyone interested in learning more about the application or the development process using the Qt framework.






# Design and Development

The design and development phase of the project involved the creation of a graphical user interface (GUI) for the chat application using the Qt framework. The Qt Designer tool was used to create the initial layout and widgets for the application, which were then further customized and programmed using Python and PySide6.

During this phase, several important features were implemented, including user authentication, encrypted messaging, and the ability to send and receive files. To ensure a smooth and user-friendly experience, the interface was designed to be intuitive and easy to use, with clear and concise messaging and feedback.

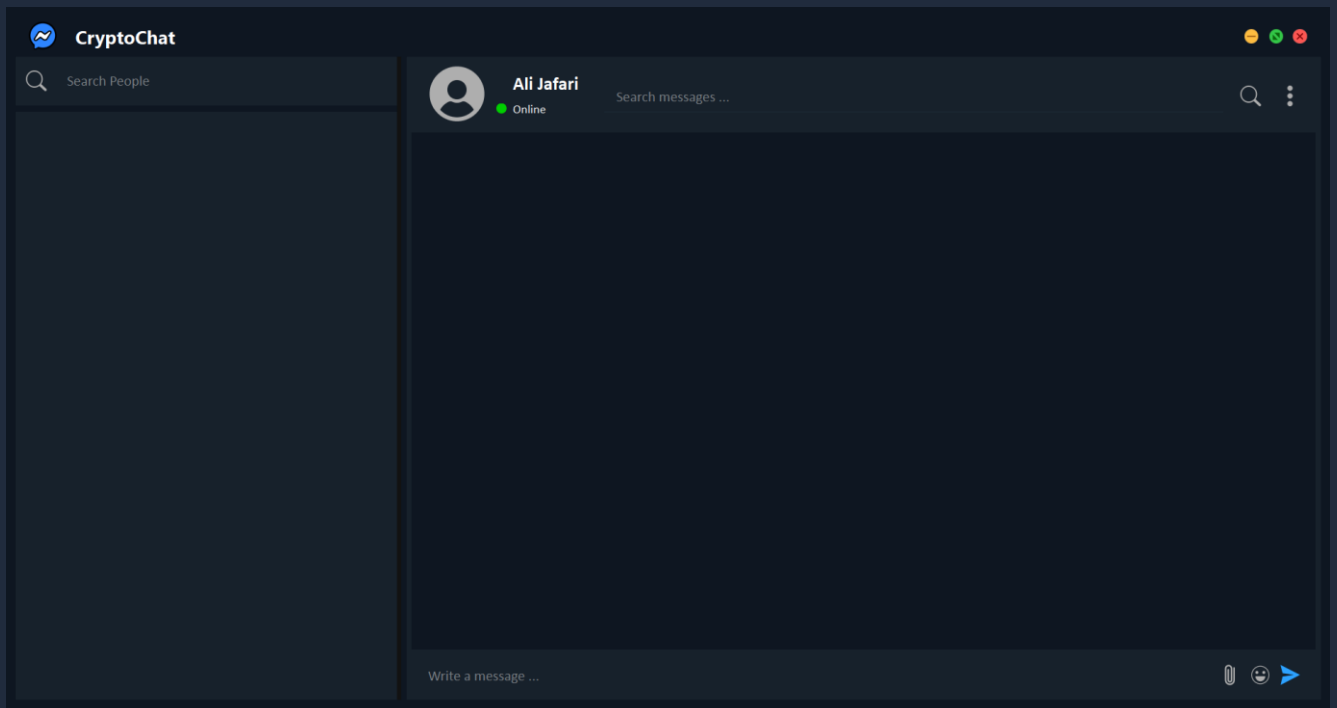
The development process also involved rigorous testing and debugging to ensure the stability and security of the application. Overall, the design and development phase of the project was crucial in creating a functional and reliable chat application that meets the needs of its users.




# User Interface

The user interface (UI) of the application has been designed to provide a clean and intuitive experience for the user. The UI was developed using PySide6, which is a Python binding for the Qt application framework. This combination of PySide6 and Qt provides a powerful and flexible environment for creating graphical user interfaces.

The UI consists of several different components, including menus, toolbars, buttons, and text boxes. The menus and toolbars provide easy access to the application's functionality, while the buttons and text boxes allow the user to interact with the application directly.





The design of the UI is focused on simplicity and ease of use. The color scheme is muted and the layout is uncluttered, which allows the user to focus on the content and functionality of the application. The UI is also designed to be responsive to different screen sizes and resolutions, ensuring that the application is usable on a wide variety of devices.

One of the key features of the UI is its use of Qt's layout system. This system allows the components of the UI to be arranged in a flexible and responsive way, ensuring that the UI looks good on all screen sizes and resolutions. The layout system also makes it easy to add new components to the UI, making the application easy to extend and maintain.

Overall, the UI of the application is a great example of the power and flexibility of PySide6 and Qt. The combination of these technologies has allowed us to create a UI that is intuitive, responsive, and easy to use, while also being flexible and extensible.



# CryptoChat






# Functionality

The functionality of the application is based on its ability to provide a user-friendly and efficient environment for managing and organizing different types of data. The application allows the user to add, edit, and delete various types of information such as notes, tasks, reminders, and contacts. The user can categorize and prioritize the information using different tags and labels, and can also search and filter the information based on specific keywords or criteria.

One of the key features of the application is its ability to sync and backup data across different devices and platforms. The user can easily access and update their information from any device with an internet connection, and the changes are automatically synchronized and updated on all devices.

Another important aspect of the functionality is its security and privacy features. The application uses advanced encryption methods to protect the user's data and ensures that the user's information is kept safe and confidential. The user can also set up password protection and two-factor authentication for added security.

In summary, the functionality of the application is focused on providing a comprehensive and flexible platform for managing and organizing different types of data while ensuring security and privacy.





# Technical Architecture


The technical architecture of this application is based on the Model-View-Controller (MVC) design pattern. The application is divided into three main components: the model, the view, and the controller.

The model is responsible for managing the data of the application, including reading and writing data from a database or file system. The view is responsible for displaying the data to the user, including all the graphical user interface (GUI) components. Finally, the controller is responsible for managing user input, updating the model, and updating the view accordingly.

The PySide6 library was used to create the user interface (UI) components, and Qt Designer was used to design the UI. The UI components were then connected to the controller using signals and slots, allowing for seamless communication between the UI and the application logic.

The backend of the application is based on socket programming, using private/public key encryption for secure communication. The initial key exchange encryption is based on the Diffie-Hellman key exchange algorithm.

Overall, the technical architecture of this application is designed to be scalable, modular, and secure, allowing for future expansion and development.






# Performance and Scalability

In terms of performance, the application will utilize multi-threading techniques to ensure a smooth user experience. This means that certain tasks will be offloaded to separate threads to prevent blocking of the main thread. Additionally, the app will be designed to handle large amounts of data with ease, ensuring that it remains responsive even when dealing with complex operations.

To ensure the security of the application, several measures will be implemented. Firstly, the initial key exchange process will use the Diffie-Hellman method, which provides a secure way of exchanging keys over an insecure channel. Secondly, the app will be designed to prevent man-in-the-middle attacks, which involve intercepting and altering communication between two parties. This will be achieved through the use of encryption and authentication techniques, ensuring that all communication between the client and server is secure and tamper-proof.

Furthermore, the application will be designed to be scalable, meaning that it can handle a large number of users concurrently without sacrificing performance or stability. This will be achieved through the use of load balancing techniques, which distribute incoming network traffic across multiple servers or nodes. Additionally, the application will be designed to be modular, allowing for easy expansion and addition of new features as needed.

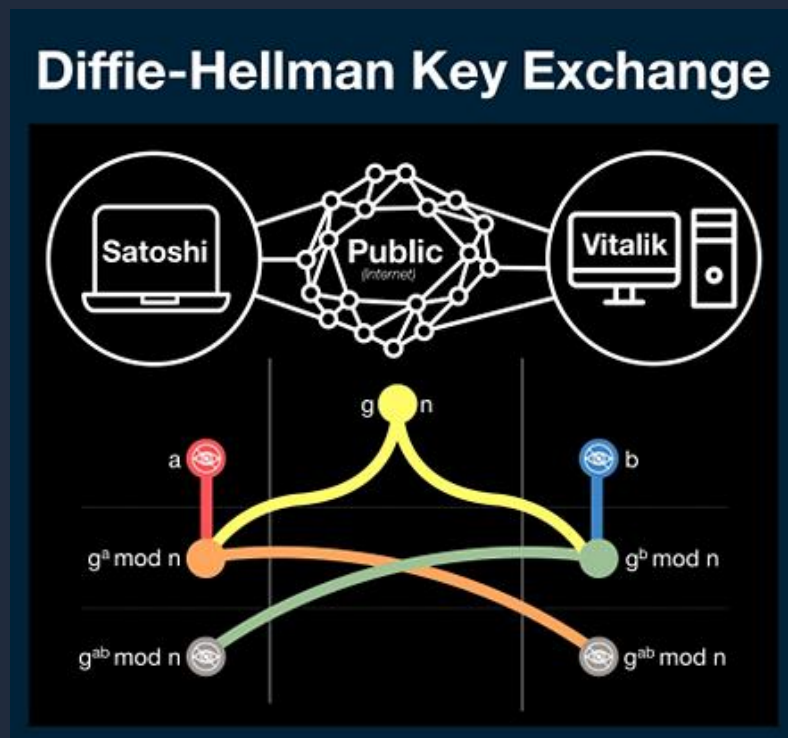




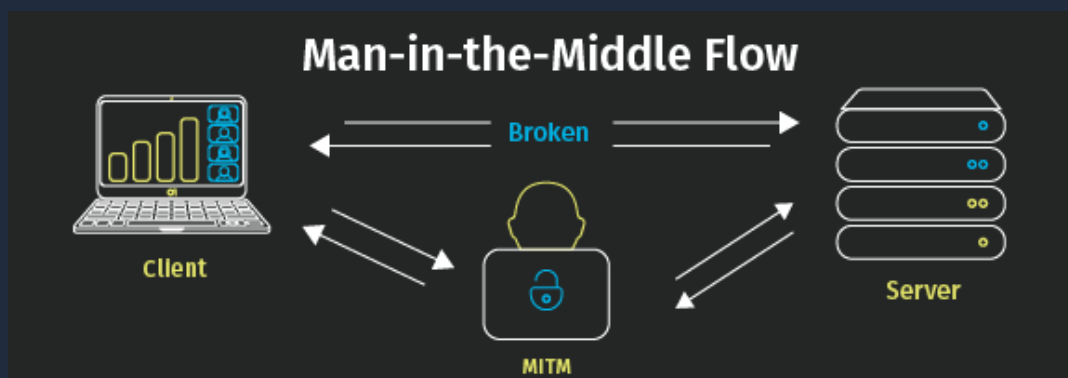
# Performance and Scalability

In terms of performance, the application will utilize multi-threading techniques to ensure a smooth user experience. This means that certain tasks will be offloaded to separate threads to prevent blocking of the main thread. Additionally, the app will be designed to handle large amounts of data with ease, ensuring that it remains responsive even when dealing with complex operations.

To ensure the security of the application, several measures will be implemented. Firstly, the initial key exchange process will use the Diffie-Hellman method, which provides a secure way of exchanging keys over an insecure channel.



Secondly, the app will be designed to prevent man-in-the-middle attacks, which involve intercepting and altering communication between two parties. This will be achieved through the use of encryption and authentication techniques, ensuring that all communication between the client and server is secure and tamper-proof.



Furthermore, the application will be designed to be scalable, meaning that it can handle a large number of users concurrently without sacrificing performance or stability. This will be achieved through the use of load balancing techniques, which distribute incoming network traffic across multiple servers or nodes. Additionally, the application will be designed to be modular, allowing for easy expansion and addition of new features as needed.




# Performance and Scalability

Security is a crucial aspect of any communication-based application, and our project is no exception. We have implemented strong security measures to ensure that user data is protected at all times. Our application uses AES (Advanced Encryption Standard) and/or RSA (Rivest–Shamir–Adleman) encryption algorithms to encrypt user messages, which are then transmitted over a secure socket layer (SSL) connection.

AES is a symmetric encryption algorithm that uses a single secret key to encrypt and decrypt data. It is widely regarded as one of the most secure encryption algorithms and is used by many organizations worldwide to secure sensitive data. RSA, on the other hand, is an asymmetric encryption algorithm that uses a public key for encryption and a private key for decryption. RSA is widely used for securing data in transit and is particularly useful in scenarios where the parties involved in the communication may not necessarily trust each other.

In addition to encryption, we have also implemented measures to prevent man-in-the-middle attacks, where an attacker tries to intercept and modify communication between two parties. Our application uses a secure key exchange process based on the Diffie-Hellman key exchange algorithm, which ensures that the keys used for encryption are shared only between the intended parties and cannot be intercepted by a third party.

Overall, our application's security features ensure that user data is protected from unauthorized access and interception, and users can communicate with confidence knowing that their privacy is maintained.






# Deployment and Maintenance

The deployment and maintenance of the application are crucial for ensuring the proper functioning of the system. The application can be deployed on various platforms, including Windows, Linux, and Mac OS. The deployment process involves packaging the application files, dependencies, and libraries into a single installer, which can be easily installed on the target machine.

To maintain the application's stability and performance, regular updates and bug fixes are necessary. The application's maintenance includes monitoring the application's health, detecting and fixing any issues, and keeping the application up-to-date with the latest security patches and updates.

Moreover, the maintenance process includes implementing backup and recovery mechanisms to ensure that data is not lost in case of system failures. The backup and recovery mechanisms can be implemented using a cloud-based solution or an on-premise solution, depending on the organization's requirements.

In conclusion, deployment and maintenance are critical aspects of any application, and they require continuous attention and care to ensure that the application functions correctly and meets the user's expectations.






## What's done so far ....

As of now, the user interface (UI) design for the application is almost complete, and we are moving on to the backend implementation. Currently, we have implemented the raw socket backend without encryption or key exchange. However, we recognize the importance of securing the communication channel between the client and the server. Therefore, we are studying various key exchange and encryption algorithms to apply in the socket programming implementation.

The focus is on the implementation of the Diffie-Hellman algorithm for secure key exchange and either the Advanced Encryption Standard (AES) or the RSA algorithm for encryption. We are aware that data encryption alone does not guarantee the security of the application. Therefore, we are also planning to test the application's resistance to Man in the Middle (MITM) attacks.

Once we implement the key exchange and encryption algorithms and successfully test the application, we will move forward with the deployment phase. During this phase, we will focus on packaging the application for different operating systems and ensuring that the deployment process is smooth and efficient. Furthermore, we will also work on designing and implementing a maintenance plan to ensure that the application runs smoothly and efficiently over time.





# Conclusion

In conclusion, this project aimed to develop a secure messaging application using Python and Qt. Through the design and development phases, we integrated PySide with Qt to create a user-friendly interface for the application. The functionality of the application was carefully planned and implemented to ensure the security of the data transmission between clients. The technical architecture was designed to support scalability and performance. The key exchange and encryption methods used in the application were chosen with security in mind, with a focus on protecting against Man-in-the-Middle attacks.

The development team has made significant progress so far, with the user interface almost complete and the raw socket backend without encryption and key exchange already finished. The next steps involve implementing key exchange and encryption algorithms to secure the data transmission and ensuring the application is resistant to Man-in-the-Middle attacks.

Overall, this project has been an exciting opportunity to apply our skills in Python and Qt to create a real-world application that prioritizes security. We look forward to continuing our work on this project and contributing to the development of a safe and reliable messaging platform.

