CS6355 - Cryptanalysis Project
Post-Quantum Safe Application

Mackenzie Chase, Eduardo Lazo, Arth Patel

November 2024

Abstract

The advent of quantum computing poses significant challenges to classical cryptographic systems. It has necessitated the development and implementation of quantum-resistant cryptographic protocols. In our project, we explored post-quantum cryptography and integrated the Open Quantum Safe (OQS) library with its Python wrapper to implement a quantum-safe key exchange mechanism, and digital signature mechanism. Our work focuses on the practical application of lattice-based cryptographic schemes, particularly CRYSTALS-Kyber for key encapsulation and CRYSTALS-Dilithium for digital signatures, ensuring secure communication. Using Python, we develop a client-server architecture-based messaging app that leverages OQS to establish quantum-resistant key exchanges and digitally signed message transmissions. This project provides a hands-on implementation of post-quantum cryptography's potential and challenges, contributing to the broader effort to secure digital communication against future quantum threats.

1 Introduction

Rapid advances in quantum computing technology have brought about unprecedented computational power capable of solving problems previously considered infeasible with classical computers. With that, it also presents a significant threat to current public-key cryptographic systems becoming vulnerable to quantum attacks, particularly utilizing Shor's algorithm. Moreover, Grover's Algorithm poses a threat to symmetric-key systems. However, this only reduces the computations in half, so in theory, symmetric algorithms with a large enough key size are still considered secure. This looming threat has necessitated the development and deployment of quantum-resistant cryptographic schemes to safeguard sensitive data against future quantum adversaries.

In this project, we have implemented a key-exchange mechanism using the Open Quantum Safe library to explore the practical aspects of post-quantum cryptography. Our system utilizes CRYSTALS-Kyber for secure key exchange, robust symmetric algorithms (AES256, AESGCM) for message encryption and decryption, and CRYSTALS-Dilithium for digital signatures. We have developed a client-server architecture that demonstrates the feasibility of establishing quantum-resistant communication channels in a practical setting. This report covers the mathematical background of the algorithms used in the implementation, a description of the application, UML Class diagrams and flow diagrams, the challenges we faced during the project, and future considerations for the application.

2 Mathematical background

Kyber is a quantum-safe key encapsulation mechanism (KEM). It was standardized by NIST in FIPS 203 by the name of ML-KEM(Module-Lattice-based KEM) on 13 August 2024 [13]. Kyber algorithm operates with polynomial rings module q such that $Z_q = [x]$ where all the coefficients are modulo q. Therefore, all arithmetic operations performed over those polyonomials, its coefficients are on module q. The security of Kyber relies on the hardness of the Decisional-Module Learning With Errors(D-MLWE) problem, which is related to the Module Learning with error problem, and the security of Dilithium is based on the hardness of the D-MLWE and the Module Short Integer Solutions (MSIS).

2.1 Rings

2.1.1 R_q

The polynomial ring of Kyber have the following structure $R_q = Z_q[x]/(x^n + 1)$ where the degree of $Z_q[x]$ is less than n and multiplication of polynomials are performed modulo the reduction polynomial $x^n + 1$. To represent polynomials as vectors, a polynomial function $f(x) = a_0 + a_1x + ... + a_{n-1}x^n$ in the ring $R_q = Z_q[x]/(x^n + 1)$ it can be done by its vector coefficient such that

$f(x) = (a_0, a_1, a_{n-1})$ of length n where we can perform addition and subtraction, if we want to perform multiplication we can do that multiplying first the polynomials and then transform it to its vector representation. We want the vector representation of the polynomials because that is the format of how Kyber is going to perform its operation, encryption, decryption, and key generation[3].

2.1.2 R_q^k

The module R_q^k are polynomials in R_q of length-k the addition and multiplication of those elements is close in the ring, that means the result will belong to the Ring R_q^k

2.2 Polynomials

2.2.1 Size in Z_q

The notation for the size regarding the integer in Z_q, polynomials in $Z_q[x]/(x_{n+1})$ and vectors of polynomials in R_q^k are defined as the infinity norm such that $||.||_\infty$

Let's consider $r \epsilon Z_q$,then $||r||_\infty = |r \ mod \ q|$ where $r \epsilon Z_q$,then $0 \leq ||r||_\infty \leq (q-1)/2$ if q is odd, or $r \epsilon Z_q$, then $0 \leq ||r||_\infty \leq q/2$ if q is even

2.2.2 Size of the elements in the Ring

Let's consider $f(x) = f_0 + f_1 x + ... f_{n-1} x^{n-1} \in R_q$, so then $||f||_\infty = max||f_i||_\infty$ of the coefficients module q. Now for the size of the elements of the vector $a = [a_1 ... a_k]^T \in R_q^k$, so then $||a||_\infty = max||a_i||_\infty$.

2.2.3 Small polynomials

A polynomial then $f \in R_q$ is small if then $||f||_\infty$ is "small" if all its coefficients are small, for instance let us consider η be a positive integer very small compared to $q/2$. We define $S_\eta = f \in R_\eta s.t ||f||_\infty \leq \eta$ as the set of polynomial in r_q all whose coefficients have size almost η This set is called "small" polynomials.

2.3 Lattices

Let $n \in N$. A lattice of dimension n is a set of the form
$\mathcal{L}(B) = B_x | x \in Z^n \subseteq R^n$ for some invertible matrix $B \in Gl_n(R)$. Where $dim(\mathcal{L}) = n$ sometimes B is also called lattice basis of $\mathcal{L}$ if $b_1, ... b_n$ are the columns of B, where $||B|| = max_i ||b_i||$ Let $B \in Gl_n(R)$ be a lattice basis and $\gamma \geq 1$ be an approximation parameter[11].

2.3.1 Lattice problem

The most well-known approximation problems on this lattice are the following:

2.3.2 Shortest Vector Problem (SVP_γ)

Find a non zero vector $v \in \mathcal{L}(\mathcal{B})$ such that $||v|| \leq \gamma.min_w \in \mathcal{L}(\mathcal{B})||w||w \neq 0$

2.3.3 Closest Vector Problem (CVP_γ)

For $t \in R^n$, find a lattice point $v \in vL(B)$ s.t $||v - t|| \subseteq \gamma min_{w \in L(B)}||w - t||$

3 Kyber Key Generation and Encapsulation

Algorithm 1 Kyber.CPA.KeyGen(): key generation

$\rho, \sigma \leftarrow \{0,1\}^{256}$
$\mathbf{A} \sim R_q^{k \times k} := \mathsf{Sam}(\rho)$
$(\mathbf{s}, \mathbf{e}) \sim \beta_\eta^k \times \beta_\eta^k := \mathsf{Sam}(\sigma)$
$\mathbf{t} := \mathsf{Compress}_q(\mathbf{As} + \mathbf{e}, d_t)$
return $(pk := (\mathbf{t}, \rho), sk := \mathbf{s})$

Algorithm 2 Kyber.CPA.Enc($pk = (\mathbf{t}, \rho), m \in \mathcal{M}$): encryption

$r \leftarrow \{0,1\}^{256}$
$\mathbf{t} := \mathsf{Decompress}_q(\mathbf{t}, d_t)$
$\mathbf{A} \sim R_q^{k \times k} := \mathsf{Sam}(\rho)$
$(\mathbf{r}, \mathbf{e}_1, e_2) \sim \beta_\eta^k \times \beta_\eta^k \times \beta_\eta := \mathsf{Sam}(r)$
$\mathbf{u} := \mathsf{Compress}_q(\mathbf{A}^T\mathbf{r} + \mathbf{e}_1, d_u)$
$v := \mathsf{Compress}_q\left(\mathbf{t}^T\mathbf{r} + e_2 + \left\lceil \frac{q}{2} \right\rfloor \cdot m, d_v\right)$
return $c := (\mathbf{u}, v)$

Figure 1: Algorithms key Generation and Encapsulation [3]

4 Kyber Decryption

Algorithm 3 Kyber.CPA.Dec($sk = \mathbf{s}, c = (\mathbf{u}, v)$): decryption

1: $\mathbf{u} := \mathsf{Decompress}_q(\mathbf{u}, d_u)$
2: $v := \mathsf{Decompress}_q(v, d_v)$
3: **return** $\mathsf{Compress}_q(v - \mathbf{s}^T\mathbf{u}, 1)$

Figure 2: Algorithm Decryption [3]

5 Advantages and Limitations of PQC

Implementing Post Quantum Cryptography (PQC) now provides security against emerging threats with the advent of quantum computing and make systems more resilient [15]. In addition, implementing PQC before quantum computing becomes a mainstream phenomenon is a proactive measure to transition from Classical Cryptographic schemes to PQC [6]. However, not all Classical cryptosystems face the same threats from Quantum computing. A hybrid system using both PQC and Classical cryptographic schemes can provide a transition phase [16]. Regulatory bodies, such as the US government have recognized the pending shift and are actively working towards standardizations and guidelines, namely through NIST [15].

Quantum threats require current symmetric algorithms to expand their key size to maintain security causing an increase in computational power. This will impact system performance [9]. Moreover, legacy systems and protocols may not support PQC without significant modifications, and integration with current systems can be complex [15]. With PQC standards and algorithms still in their infancy, there is hesitation for companies to implement them [4]. In August 2024, the SIKE and SIDH algorithms were broken and deemed insecure despite making it through the first three rounds of NIST's call for proposal [8]. The cost of PQC deployment is also substantial, which includes training staff, redesigning and integrating protocols, and demand for resources. Since large-scale quantum computers capable of breaking RSA/ECC are not an imminent threat, it may be premature measure to some organizations [14].

6 Quantum Threats to Existing Systems

Quantum era threats to current cryptographic systems include adversaries intercepting and storing encrypted data today to decrypt it later with quantum computers (Harvest Now, Decrypt Later), breaking encryption immediately when

quantum capabilities mature (Break Now, Exploit Later), and forging digital signatures to compromise trust (Fake Now, Trust Later). Legacy protocols like TLS, VPNs, and PKI will become obsolete, while blockchain systems face threats to transaction integrity [14]. Quantum algorithms like Grover's will weaken symmetric encryption by reducing effective key strength, and encrypted messaging platforms risk having private communications decrypted in the future. State-sponsored actors could leverage quantum computing for espionage and attacks on critical infrastructure [5]. To mitigate these threats, organizations must adopt post-quantum cryptography, implement hybrid cryptographic systems, rotate keys regularly, and prioritize securing long-term sensitive data. These proactive steps will ensure resilience in the quantum era [10].

7 Description of the Application

Our application falls into two versions. The first version is a very simple case of two clients, one acting as a host, and the other as a "customer". The first client launches the server.py, this initializes a TCP socket listening for incoming connections. The second client launches client.py which connects to the other client. They then perform a Key Exchange, sharing their public keys for KEM and digital signing. They acquire a shared secret (the 256 bit AES key) so they can then communicate securely using AES256 with the cryptography library. They sign their messages and verify the received ciphertext using the signature algorithm. We do not implement a robust authentication protocol. We assume the two parties are who they say they are and can be trusted.

The next version of the messaging app includes a trusted server that collects the public keys of connected clients, manages chats, and forwards messages between clients. With the server running, it awaits incoming connections. When a client connects, the server creates a ClientHandler for the client, the client sends both its public keys (KEM, Signing), the client can then connect to other online clients and chat securely. When a client requests to chat with another client, the server sends the public keys to each client and they perform a key exchange. The server has no way of decrypting the ciphertext from the KEM, nor the following ciphertext messages between the two clients. It never obtains the shared secret either, ensuring the two clients can communicate securely via the server.

The applications are running on localhost. Both utilize the liboqs library, cryptography library, TCP sockets, as well as threads for sending and receiving messages. We provide some UML diagrams and flow diagrams of both versions. We further provide screenshots of the two versions running in a Kali Linux virtual machine in the appendix. We also provide some trivial Wireshark capture screenshots showing the KEM public key(s) are sent, as well as encrypted messages and signatures in a payload.

It should be noted that Open Quantum Safe do not recommend using liboqs in production environments nor for securing sensitve data. It is merely for research and prototyping.

8 Architecture

8.1 KEM

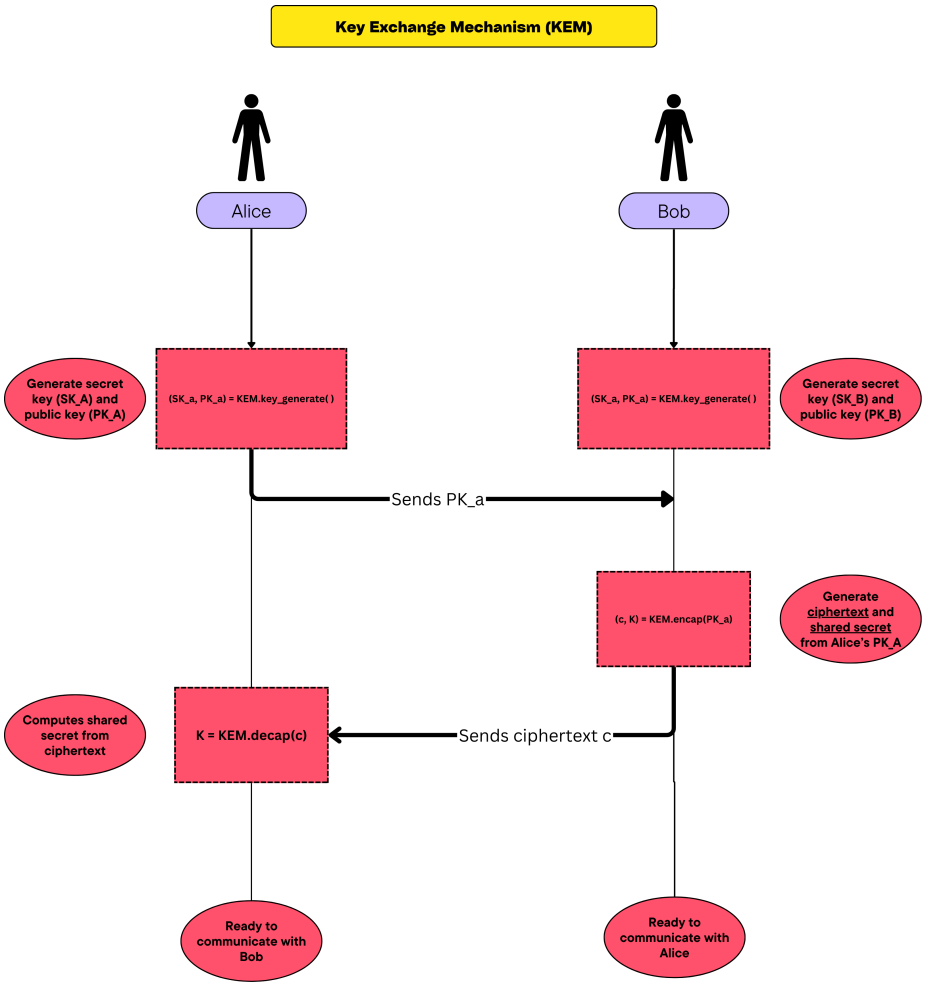

Figure 3: Key Exchange Mechanism

8.2 Simple Case

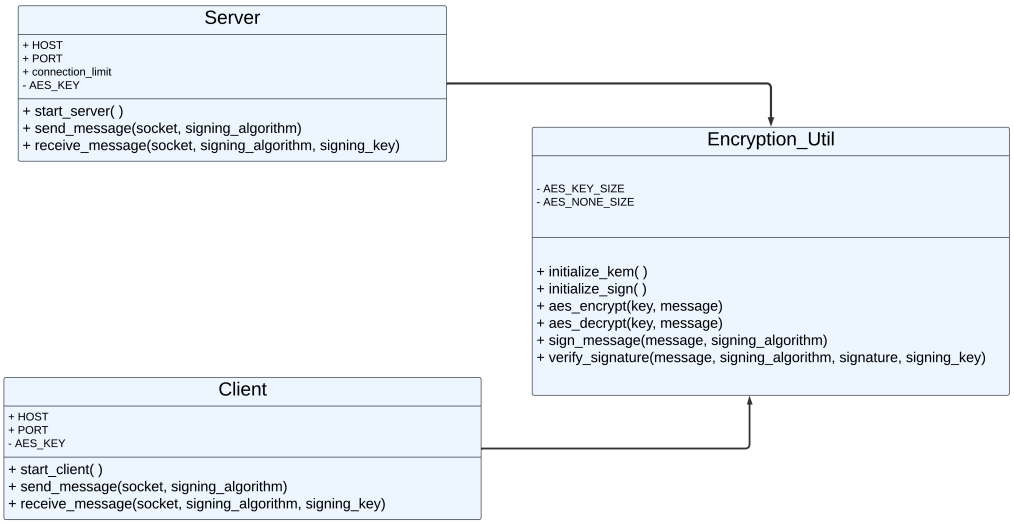

Figure 4: UML for simple case

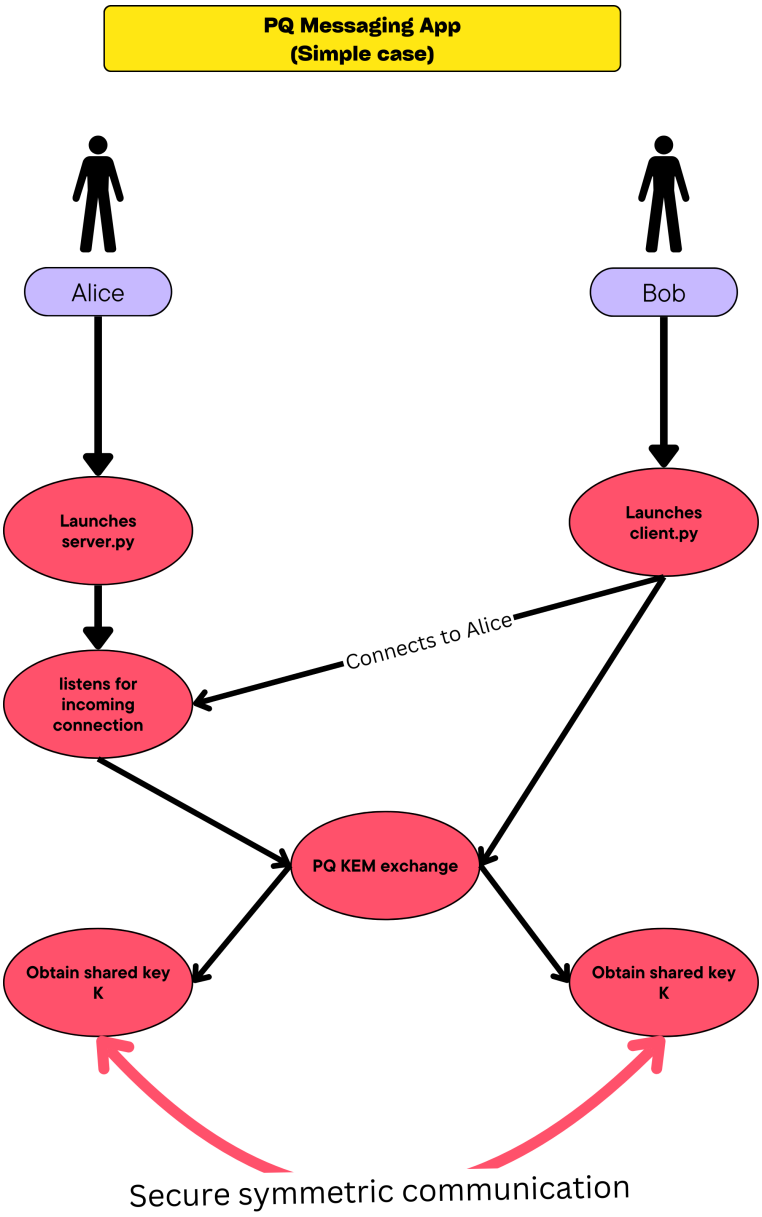

Figure 5: Flow diagram for simple case

8.3 Clients via Server Case

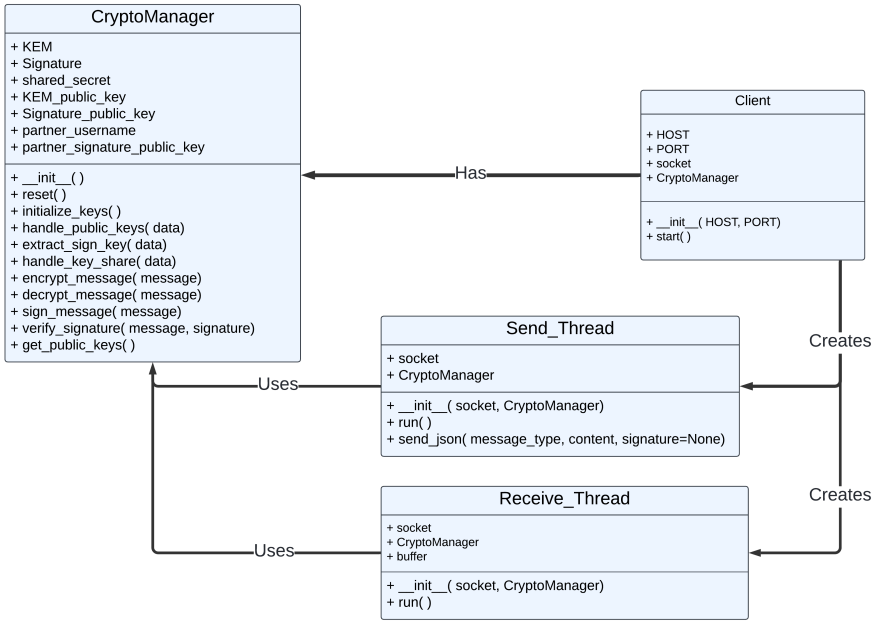

Figure 6: UML for clients

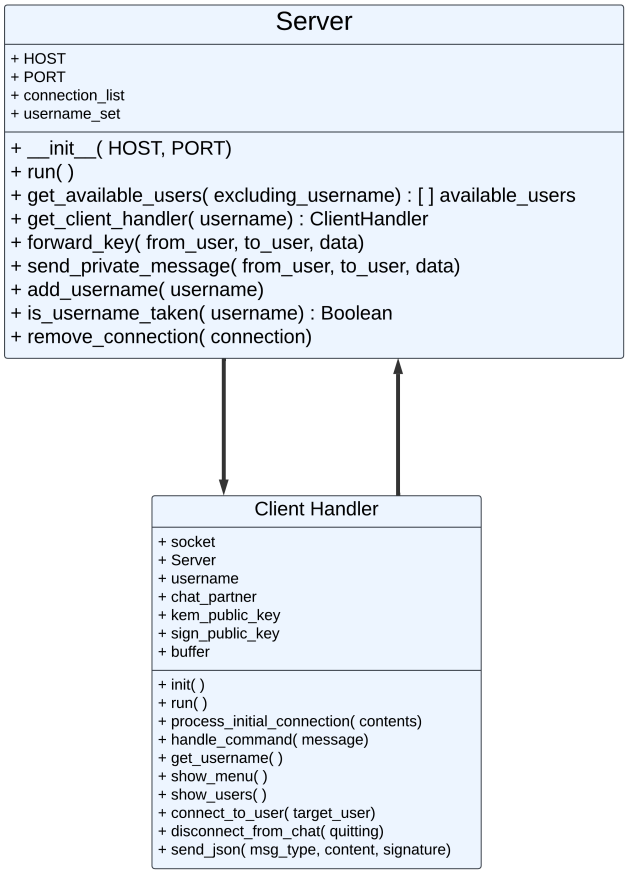

Figure 7: UML for server

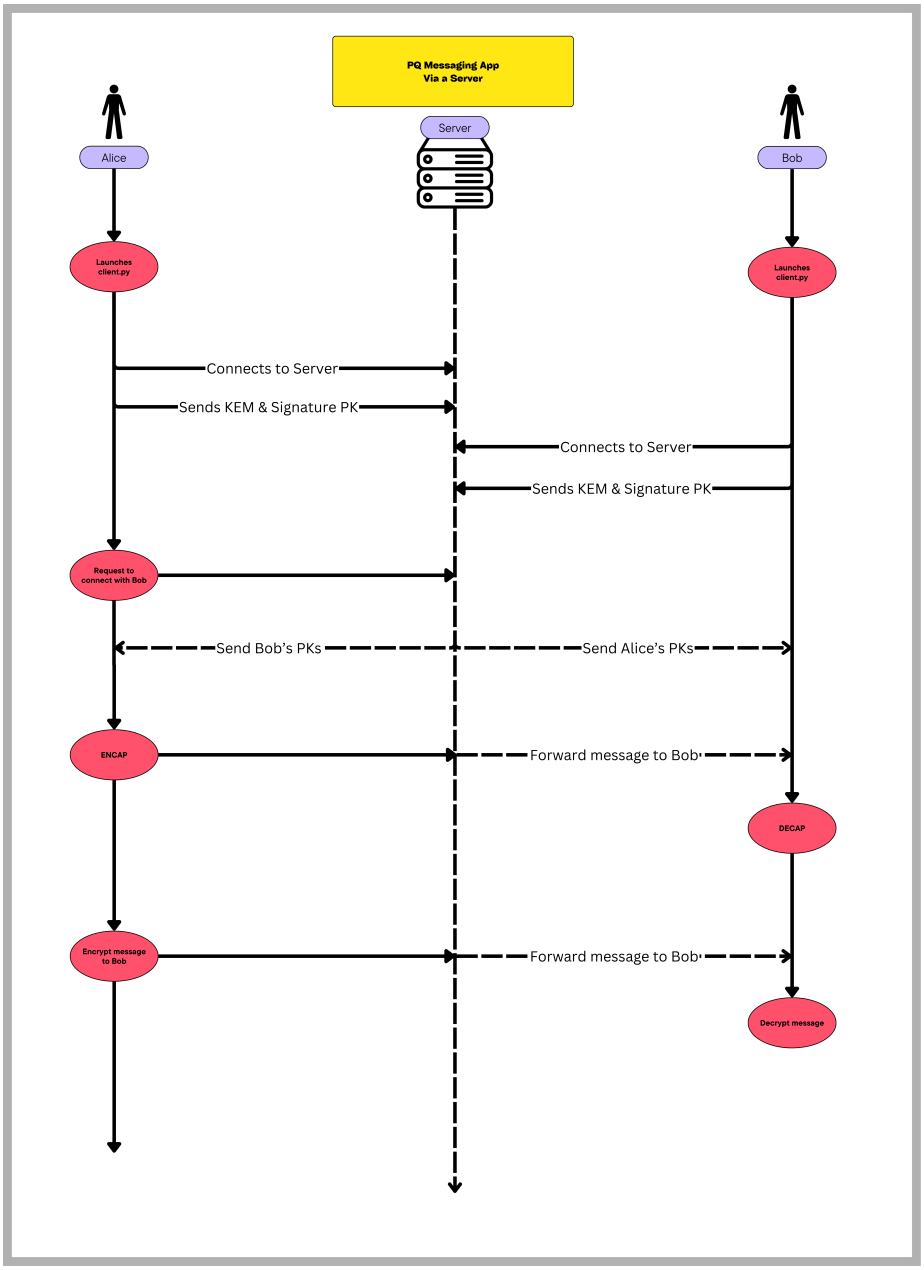

Figure 8: Flow diagram

9 Challenges Faced during implementation

We had a difficult time getting liboqs and its python wrapper (liboqs-python) installed properly, especially on Windows machines, however we were finally able to get it working on an Debian based Kali Linux virtual machine. In our project proposal, we set out to create a simple messaging application. There would be a host server, and several clients would be able to connect to it. The clients and server would perform a key exchange utilizing post-quantum safe algorithms to obtain a secure shared secret key to then communicate using a quantum safe symmetric algorithm. Our initial goal was to have clients join a chatroom where all other clients could communicate, however we quickly realized that was a difficult task to properly share the same secret key while using the individual KEM of every client. We reevaluated our goal to start simple by only having two clients communicate with each other without a centralized server. Once, that was working properly, we expanded our objective to have the same functionality, however with a centralized server in between the clients to represent a more realistic scenario (such as WhatsApp, Signal, etc.). While working with Python we need to get reacquainted with python sockets, threading, and properly parsing messages and data.

10 Future Improvements

In our current implementation, we hard coded the chosen algorithms to Kyber512 for KEM and Dilithium2 for signing, although this should be modified to allow clients to choose different algorithms especially if tomorrow these two algorithms are broken and no longer deemed secure. As we do not implement an authentication protocol, it would be wise to do so with a robust authentication process using certificates such as OQS OpenSSL. Another possible area of improvement is to implement a hybrid approach with classical algorithms as proposed by Dr. Vikas Chouhan from the CIC [2]. We lacked thorough testing in our implementation due to time constraints, therefore it would be beneficial to do so. We also do not sanitize user input which can lead to injection attacks. We would need to develop an input sanitizing procedure. Additionally, a cosmetic improvement would be a proper GUI. With the current solid foundation, we could aim to incorporate group chat functionality. Finally, this application could be converted to a web application for wide spread use.

11 Current Industry Use

Post-Quantum Cryptography (PQC) is being aggressively adopted by a number of industry leaders in order to get ready for quantum attacks. IBM, Google, and Microsoft are incorporating PQC into their services, with IBM actively supporting NIST's efforts for standardization [12]. Intel is working at the hardware-level to achieve PQC, in addition to contributing to one of the three new PQ standards released by NIST [13]. Certificate authorities such as DigiCert and

Entrust are currently testing quantum-resistant certificates [7]. Both Apple, and Signal advertise using quantum secure cryptography in their messaging applications [1] [17]. This highlights the onset of a slow shift from classical cryptography to post quantum.

12 Conclusion

In this project we implemented a chat application using Post-Quantum Algorithms to provide confidentiality using encryption, and message integrity and non-repudiation using digital signatures between two users. We did not achieve a robust user authentication scheme. Future improvements will focus on implementing certificates using OQS OpenSSL in order to provide entity authentication. A GUI would be a beneficial cosmetic upgrade to the application. Through this project, we additionally explored TCP sockets, multi-threading, and proper message parsing. Despite having difficulties setting up liboqs, we were able to achieve a working proof-of-concept, and a more matured version reflective of real-world messaging applications. Our application is resistant to attacks on the encryption of messages as Kyber and AES, with a suitable key size, are used. Since quantum cryptography is the cutting edge technology, there are still potential challenges when implementing these new cryptography schemes. Liboqs is not recommended for production environments.

References

[1] Apple. imessage with pq3: The new state of the art in quantum-secure messaging at scale, 2024. https://security.apple.com/blog/imessage-pq3/.

[2] Baseri, Y., Chouhan, V., and Hafid, A. Navigating quantum security risks in networked environments: A comprehensive study of quantum-safe network protocols. *Computers & Security 142* (2024), 103883.

[3] Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., and Stehlé, D. Crystals-kyber: a cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)* (2018), IEEE, pp. 353–367.

[4] Buchmann, J., Lauter, K., and Mosca, M. Postquantum Cryptography—State of the Art . *IEEE Security and Privacy 15*, 04 (2017).

[5] Chen, L. e. a. Report on post-quantum cryptography, 2016. https://doi.org/10.6028/NIST.IR.8105.

[6] CISA. Quantum-readiness: Migration to post-quantum cryptography, 2023. https://www.nccoe.nist.gov/sites/default/files/2023-08/quantum-readiness-fact-sheet.pdf.

[7] Digicert. Quantum computing will change everything. https://www.digicert.com/tls-ssl/post-quantum-cryptography.

[8] et al., D. J. Sike – supersingular isogeny key encapsulation, 2024. https://sike.org/.

[9] et al., L. C. Report on post-quantum cryptography, 2016. https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf.

[10] Hasan, K. F., Simpson, L., Baee, M. A. R., Islam, C., Rahman, Z., Armstrong, W., Gauravaram, P., and McKague, M. A framework for migrating to post-quantum cryptography: Security dependency analysis and case studies. *IEEE Access 12* (2024), 23427–23450.

[11] Hüttenhain, J., and Wallenborn, L. Topics in post-quantum cryptography.

[12] IBM. Make the world quantum safe, 2024. https://www.ibm.com/quantum/quantum-safe.

[13] Intel. Be ready for post-quantum security with intel cryptography primitives library, 2024. https://www.intel.com/content/www/us/en/developer/articles/technical/post-quantum-cryptography.html.

[14] Mosca, M. Cybersecurity in an era with quantum computers: Will we be ready? *IEEE Security & Privacy 16*, 5 (2018), 38–41.

[15] NIST. Post-quantum cryptography, 2024. https://csrc.nist.gov/projects/post-quantum-cryptography.

[16] Ricci, S., Dobias, P., Malina, L., Hajny, J., and Jedlicka, P. Hybrid keys in practice: Combining classical, quantum and post-quantum cryptography. *IEEE Access 12* (2024), 23206–23219.

[17] Signal. Quantum resistance and the signal protocol, 2024. https://signal.org/blog/pqxdh/.

A
Application Screenshots

A.1
Simple version

Figure 9: Simple case terminal view

A.2 Client via server version

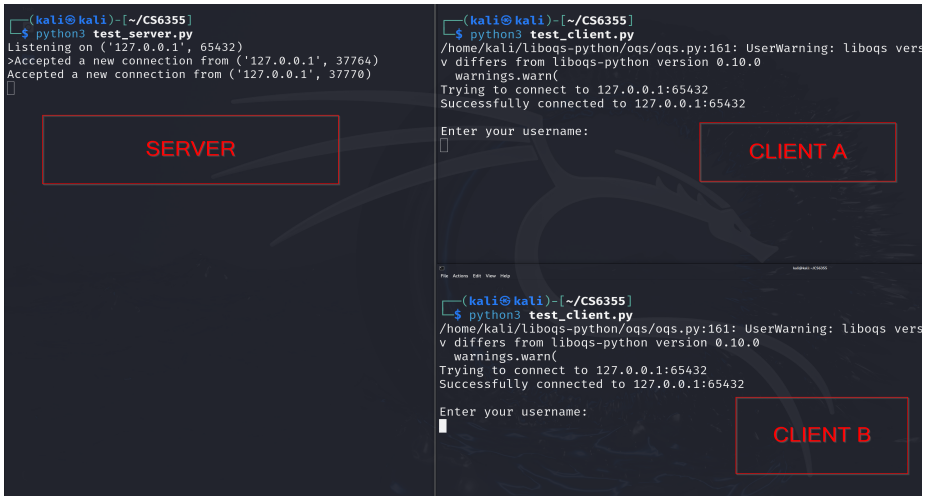

Figure 10: Initial connections

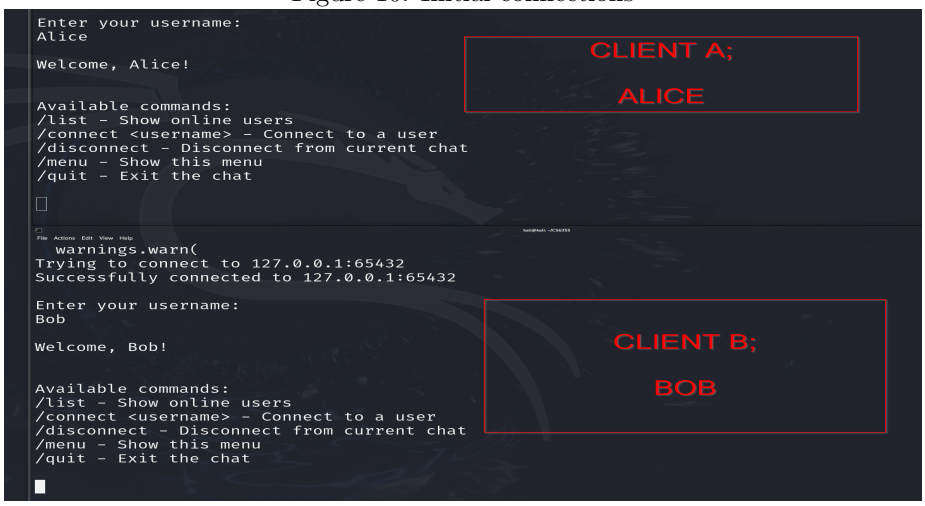

Figure 11: Menu view

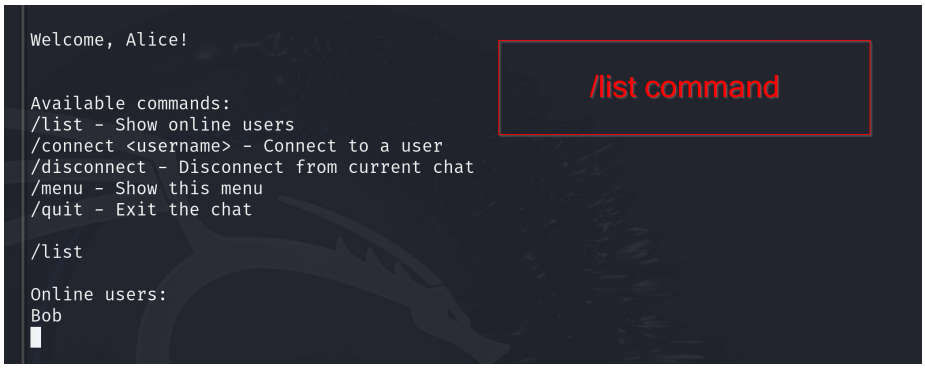

Figure 12: List online users

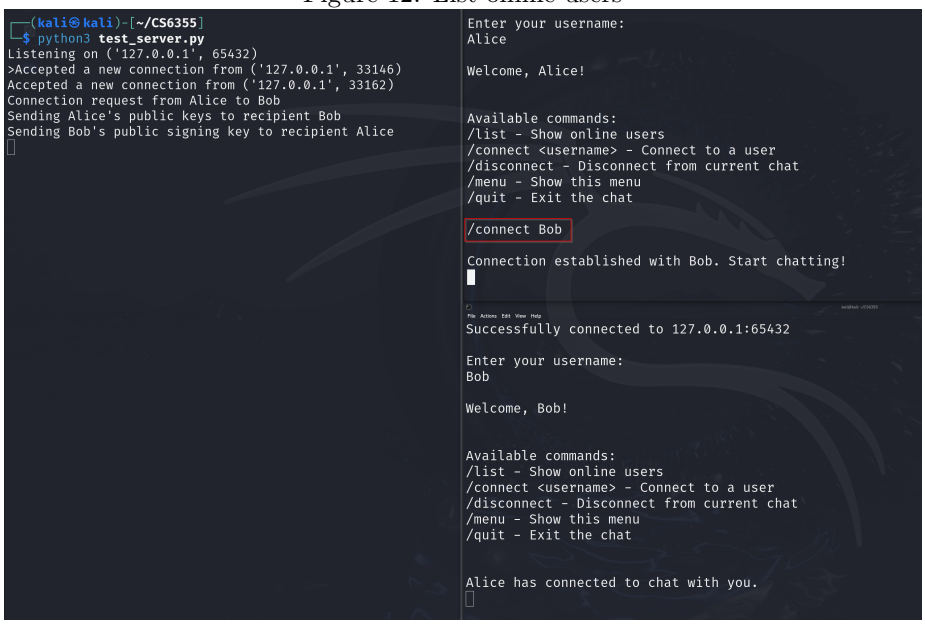

Figure 13: Connect to user

Figure 14: Chat with other user

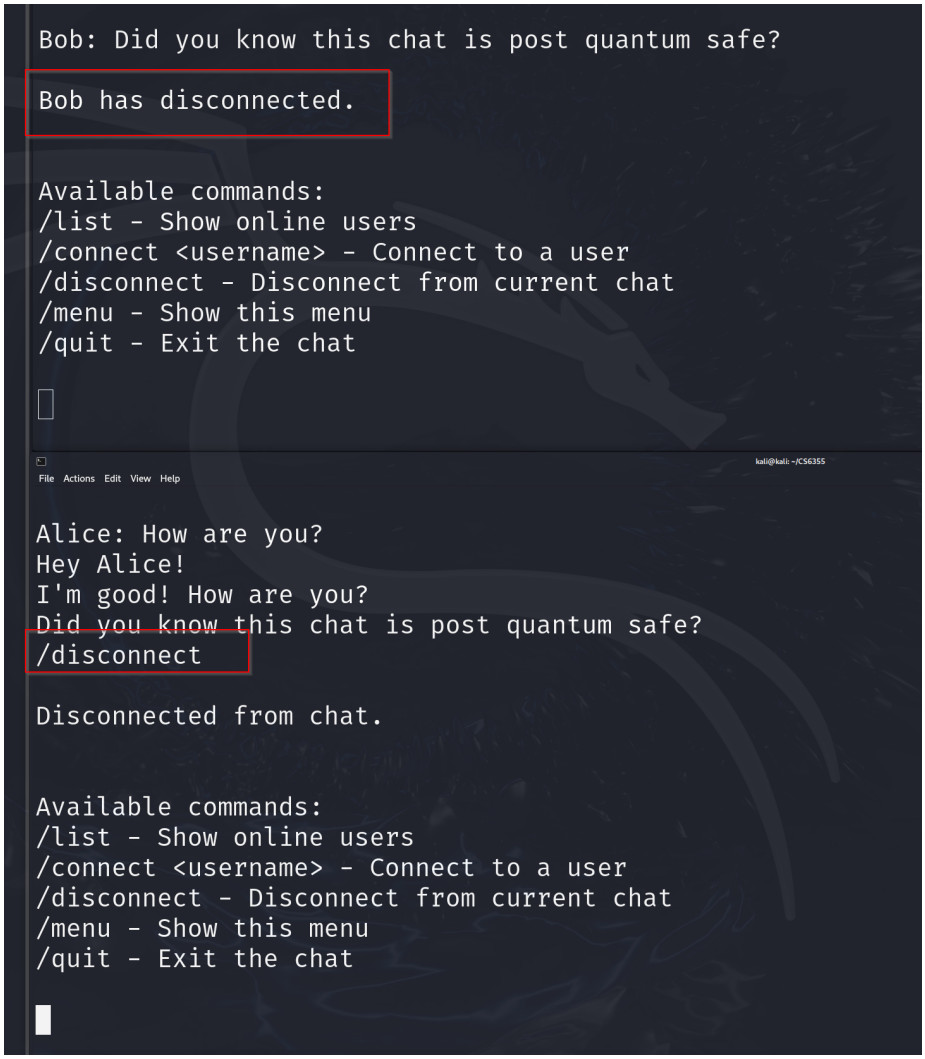

Figure 15: Disconnect from chat

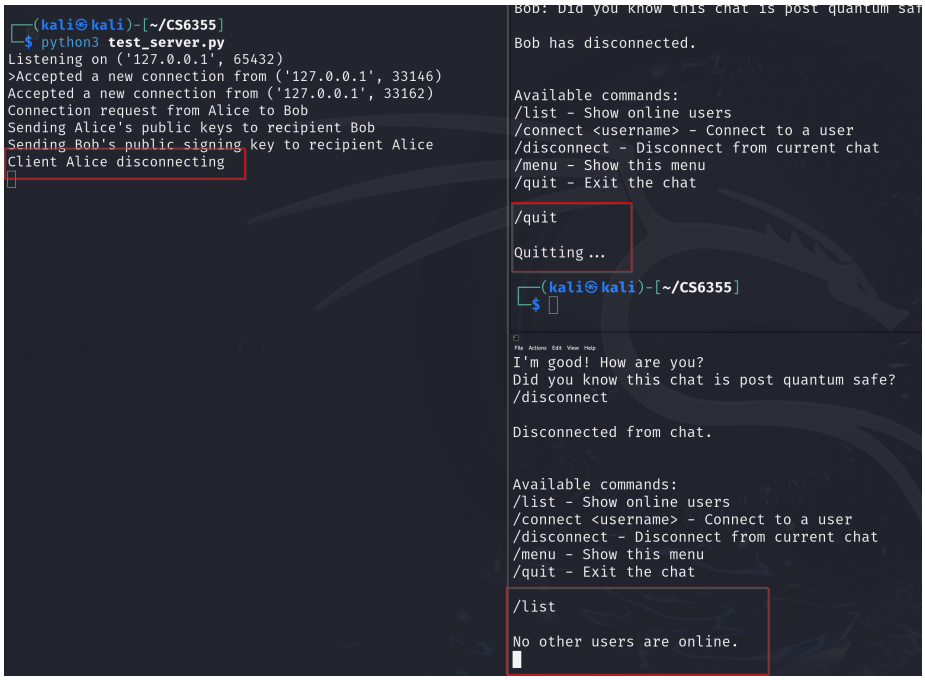

Figure 16: Quit from server

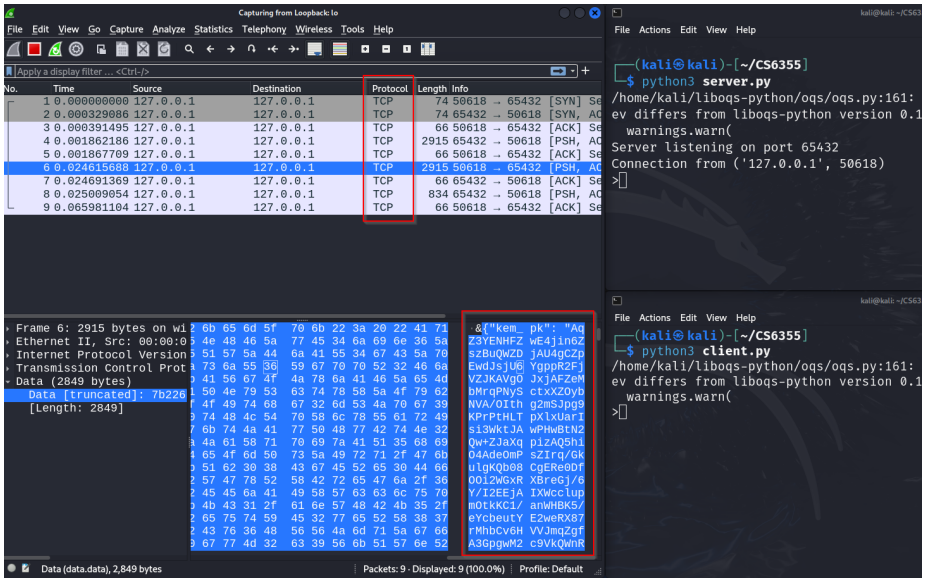

Figure 17: Wireshark capture showing TCP use and KEM public key being sent

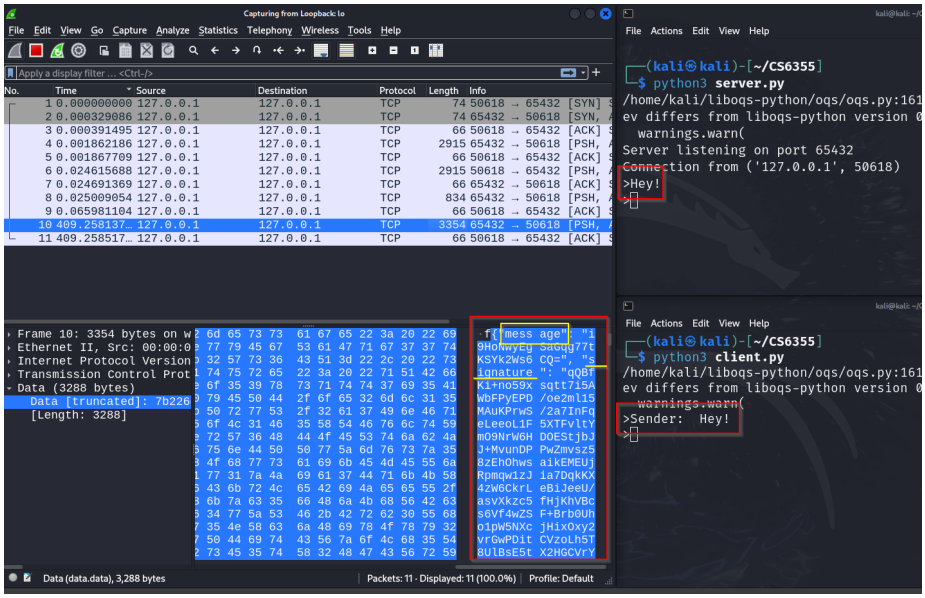

Figure 18: Wireshark capture showing message and signature being sent