Quantum-Enhanced Frequency Hopping and ASLR Mitigation: A Secure Communication Architecture

RoQteam

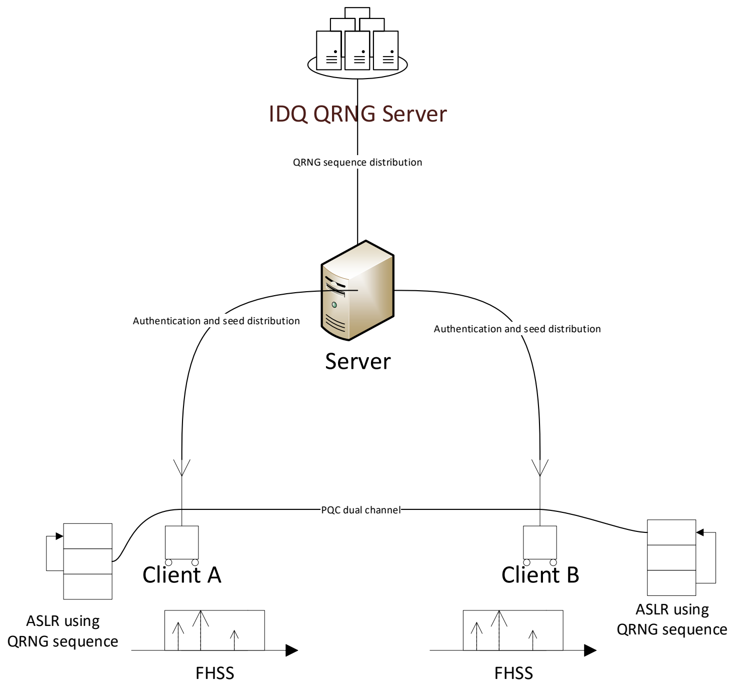
1. Introduction

This document details a novel communication architecture leveraging quantum random number generation (QRNG) and post-quantum cryptography (PQC) to enhance the security and flexibility of frequency hopping spread spectrum (FHSS) communication and address space layout randomization (ASLR) vulnerabilities. The system is designed with a modular, containerized approach using Docker for ease of deployment and scalability.

2. System Overview

The architecture comprises four primary components: the core is a QRNG-based seed generator from IDQuantique's QRNGaaS API, FHSS communication clients, an ASLR secured application layer on the client side and a client server architecture using Docker containers in order to mimic a real-world scenario.

2.1. Architecture Diagram



2.2. Components

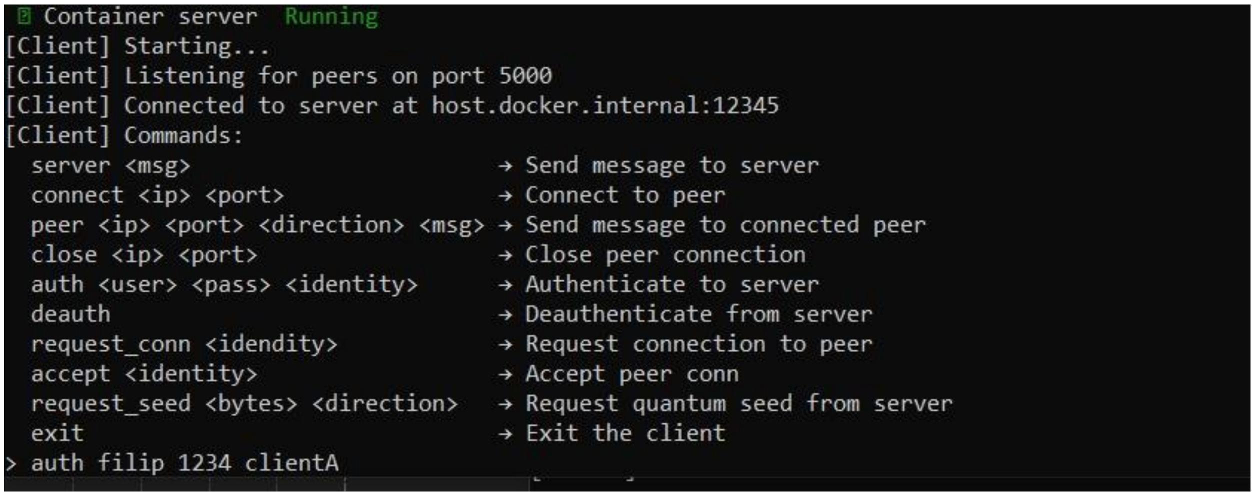
- **QRNG Seed Generator (Server):** This component utilizes the IDQuantique QRNG API to generate a unique seed for each client. The seed serves as the foundation for both FHSS frequency selection and initial key exchange.
- **FHSS Communication Clients:** These clients receive the seed from the server and use it to generate a pseudo-random sequence for frequency hopping. The frequency hopping pattern is synchronized between the client and the intended receiver.
- **Post-Quantum Cryptography (PQC) Key Exchange:** Following seed reception, clients and the server establish a secure, long-term communication channel using a PQC algorithm (e.g., Kyber). This provides forward secrecy and protects against future quantum computer attacks. *(Client-to-client PQC is planned for future development)*
- **Dual-Channel Communication:** The system employs a dual-channel approach. The initial seed drives the FHSS frequencies, while the PQC-secured channel facilitates additional data transmission and control signals.

3. Technical Details

3.1. QRNG Seed Generation

The IDQuantique QRNG API is used to generate a unique seed for each client. The seed is a high-entropy random number, ensuring unpredictability and resistance to statistical attacks.

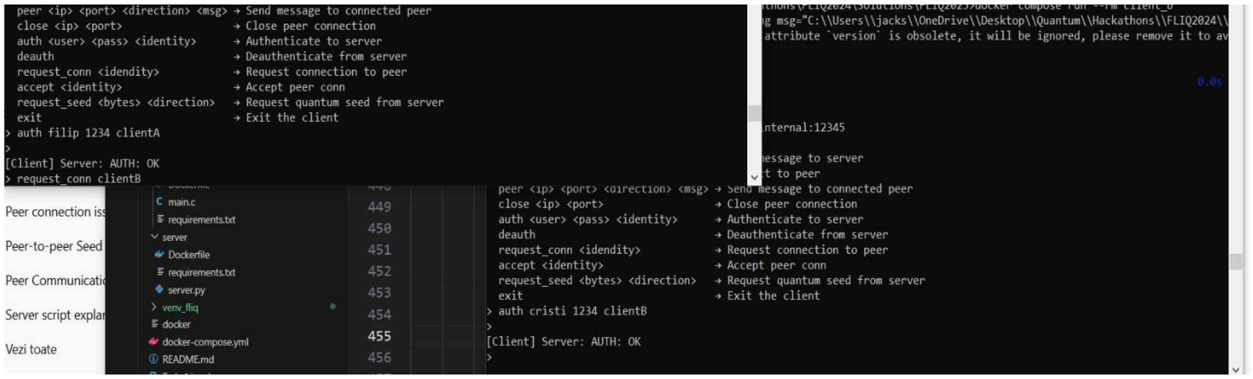
The configured commands for the clients are the following:



```
 ? Container server  Running
[Client] Starting...
[Client] Listening for peers on port 5000
[Client] Connected to server at host.docker.internal:12345
[Client] Commands:
  server <msg>                      → Send message to server
  connect <ip> <port>               → Connect to peer
  peer <ip> <port> <direction> <msg> → Send message to connected peer
  close <ip> <port>                 → Close peer connection
  auth <user> <pass> <identity>     → Authenticate to server
  deauth                            → Deauthenticate from server
  request_conn <idendity>           → Request connection to peer
  accept <identity>                 → Accept peer conn
  request_seed <bytes> <direction>  → Request quantum seed from server
  exit                              → Exit the client
> auth filip 1234 clientA
```

We can observe the client steps:

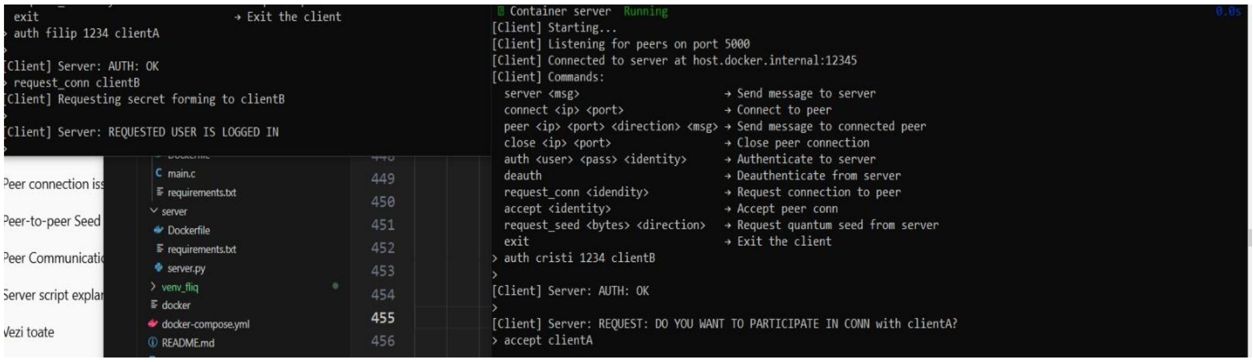
- Authentification:



```
  peer <ip> <port> <direction> <msg> → Send message to connected peer
  close <ip> <port>                  → Close peer connection
  auth <user> <pass> <identity>      → Authenticate to server
  deauth                             → Deauthenticate from server
  request_conn <idendity>            → Request connection to peer
  accept <identity>                  → Accept peer conn
  request_seed <bytes> <direction>   → Request quantum seed from server
  exit                               → Exit the client
> auth filip 1234 clientA
>
[Client] Server: AUTH: OK
> request_conn clientB
```

```
g msg="C:\\Users\\jacks\\OneDrive\\Desktop\\Quantum\\Hackathons\\FLIQ2024\\
 attribute `version` is obsolete, it will be ignored, please remove it to av
0.0s
nternal:12345
essage to server
t to peer
 peer <ip> <port> <direction> <msg> → Send message to connected peer
 close <ip> <port>                  → Close peer connection
 auth <user> <pass> <identity>      → Authenticate to server
 deauth                             → Deauthenticate from server
 request_conn <idendity>            → Request connection to peer
 accept <identity>                  → Accept peer conn
 request_seed <bytes> <direction>   → Request quantum seed from server
 exit                               → Exit the client
> auth cristi 1234 clientB
>
[Client] Server: AUTH: OK
>
```

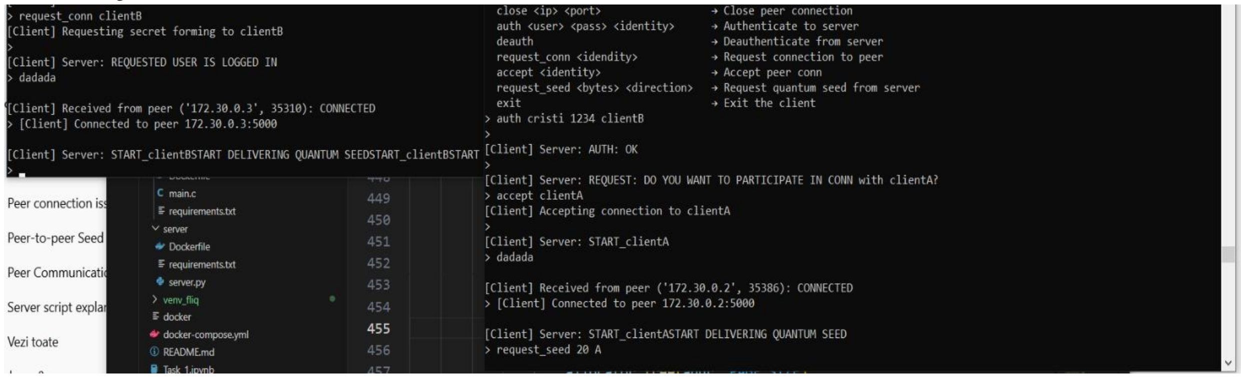
- Connection between clients:



```
 exit                               → Exit the client
> auth filip 1234 clientA
>
[Client] Server: AUTH: OK
> request_conn clientB
[Client] Requesting secret forming to clientB
>
[Client] Server: REQUESTED USER IS LOGGED IN
>
```

```
 Container server  Running                                                0.0s
[Client] Starting...
[Client] Listening for peers on port 5000
[Client] Connected to server at host.docker.internal:12345
[Client] Commands:
  server <msg>                       → Send message to server
  connect <ip> <port>                → Connect to peer
  peer <ip> <port> <direction> <msg> → Send message to connected peer
  close <ip> <port>                  → Close peer connection
  auth <user> <pass> <identity>      → Authenticate to server
  deauth                             → Deauthenticate from server
  request_conn <idendity>            → Request connection to peer
  accept <identity>                  → Accept peer conn
  request_seed <bytes> <direction>   → Request quantum seed from server
  exit                               → Exit the client
> auth cristi 1234 clientB
>
[Client] Server: AUTH: OK
>
[Client] Server: REQUEST: DO YOU WANT TO PARTICIPATE IN CONN with clientA?
> accept clientA
```

- Quantum seed distribution:



```
> request_conn clientB
[Client] Requesting secret forming to clientB
>
[Client] Server: REQUESTED USER IS LOGGED IN
> dadada

[Client] Received from peer ('172.30.0.3', 35310): CONNECTED
> [Client] Connected to peer 172.30.0.3:5000

[Client] Server: START_clientBSTART DELIVERING QUANTUM SEEDSTART_clientBSTART
>
```

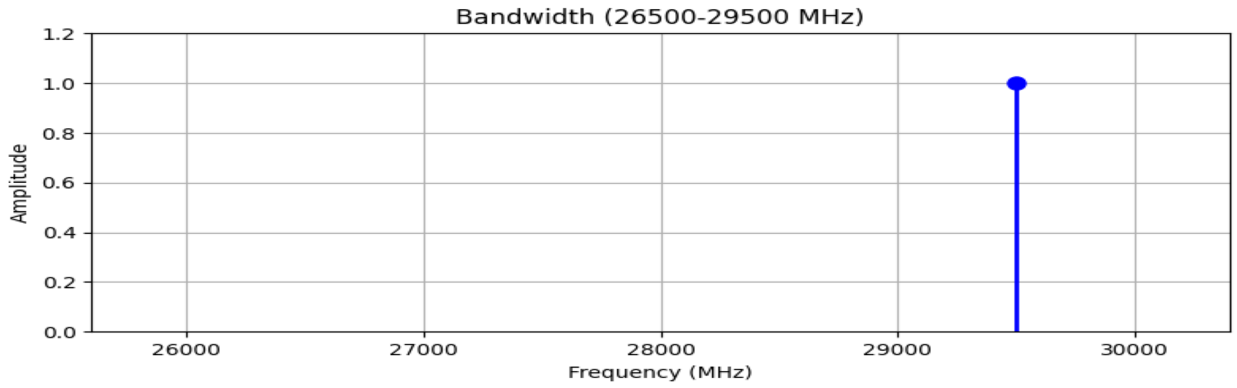
```
 close <ip> <port>                  → Close peer connection
 auth <user> <pass> <identity>      → Authenticate to server
 deauth                             → Deauthenticate from server
 request_conn <idendity>            → Request connection to peer
 accept <identity>                  → Accept peer conn
 request_seed <bytes> <direction>   → Request quantum seed from server
 exit                               → Exit the client
> auth cristi 1234 clientB
>
[Client] Server: AUTH: OK
>
[Client] Server: REQUEST: DO YOU WANT TO PARTICIPATE IN CONN with clientA?
> accept clientA
[Client] Accepting connection to clientA
>
[Client] Server: START_clientA
> dadada

[Client] Received from peer ('172.30.0.2', 35386): CONNECTED
> [Client] Connected to peer 172.30.0.2:5000

[Client] Server: START_clientASTART DELIVERING QUANTUM SEED
> request_seed 20 A
```

3.2. Frequency Hopping Spread Spectrum (FHSS)

The seed is used to generate a pseudo-random sequence that dictates the hopping frequencies. The sequence is synchronized between the client and receiver, allowing for secure communication even in noisy environments.

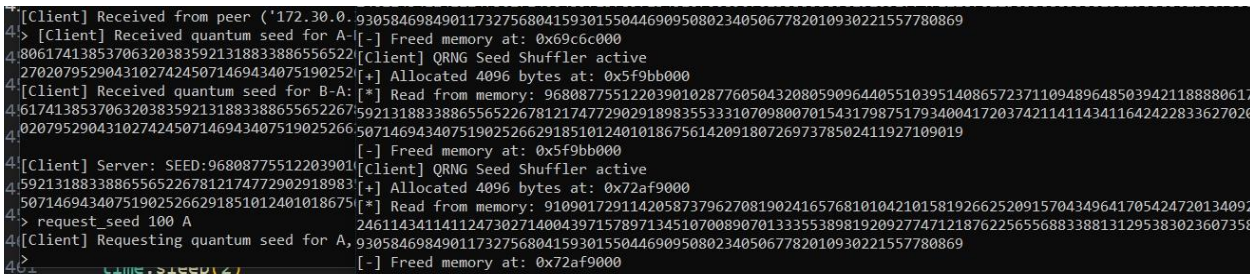


3.3. Post-Quantum Cryptography (PQC)

The Kyber key encapsulation mechanism (KEM) is selected for its efficiency and security. The server and clients establish a secure channel using Kyber, protecting the initial seed exchange and subsequent control data. *(Client-to-client PQC is planned for future development)*

3.4. ASLR Proof-of-Concept (PoC)

The FHSS system, and the ASLR techniques provide a solid demonstration of the QRNG-based security. Our PoC demonstrates how the unpredictable hopping frequencies, derived from the QRNG seed, can be used alongside dynamically shifted memory regions during runtime. This makes it significantly more difficult for attackers to reliably predict memory addresses and exploit ASLR vulnerabilities. Specifically, we're exploring the use of the FHSS sequence to influence the placement of critical data structures, making static analysis and memory mapping techniques less effective. The PQC-secured channel ensures that the control signals used to manage this dynamic memory shifting are protected from eavesdropping and tampering.



```
[Client] Received from peer ('172.30.0.
> [Client] Received quantum seed for A-
80617413853706320383592131883388655652
27020795290431027424507146943407519025
[Client] Received quantum seed for B-A:
6174138537063203835921318833886556522679
0207952904310274245071469434075190252661

[Client] Server: SEED:968087755122039010
5921318833886556522678121747729029189833
5071469434075190252662918510124010186750
> request_seed 100 A
[Client] Requesting quantum seed for A,
>
```

```
93058469849011732756804159301550446909508023405067782010930221557780869
[-] Freed memory at: 0x69c6c000
[Client] QRNG Seed Shuffler active
[+] Allocated 4096 bytes at: 0x5f9bb000
[*] Read from memory: 968087755122039010287760504320805909644055103951408657237110948964850394211888806l7
59213188338865565226781217477290291898355333107098007015431798751793400417203742114114341164242283362702
50714694340751902526629185101240101867561420918072697378502411927109019
[-] Freed memory at: 0x5f9bb000
[Client] QRNG Seed Shuffler active
[+] Allocated 4096 bytes at: 0x72af9000
[*] Read from memory: 910901729114205873796270819024165768101042101581926625209157043496417054247201340922
24611434114112473027140043971578971345107008907013335538981920927747121876225655688338813129538302360735
93058469849011732756804159301550446909508023405067782010930221557780869
[-] Freed memory at: 0x72af9000
```

4. ASLR PoC Details

- **Dynamic Memory Shifting:** The QRNG sequence is used to offset the base addresses of key data structures by a small, unpredictable amount at regular intervals.
- **Control Signal Protection:** The signals controlling the memory shifting process should be transmitted over the PQC-secured channel, preventing attackers from manipulating the shifting pattern.
- **Limited Predictability:** While the shifting is periodic, the QRNG-derived seed makes it computationally infeasible to predict the exact memory layout at any given time.

5. Future Improvements

- **Client-to-Client PQC:** Implementing PQC between clients will further enhance the security of the system and enable secure direct communication.
- **Adaptive Shifting:** Developing an adaptive shifting algorithm that adjusts the shifting pattern based on observed attack attempts.
- **Hardware Acceleration:** Exploring hardware acceleration for the FHSS sequence generation and memory shifting process.

6. Conclusion

This architecture demonstrates a novel approach to enhancing security through the combination of QRNG, FHSS, and PQC. The ASLR PoC highlights the potential for mitigating ASLR bypass techniques by dynamically shifting memory regions. Future improvements will focus on further strengthening the system's security and adaptability.