IN2194
Interim Report

Onion Authentication
Group 40

Rui Bin Choo (03691824)
Danwen Ouyang (03692087)

1 Process Architecture

We aim to first have a single-threaded implementation, however this is open to extension for multithreading later on. The primary focus will be on implementation of all basic functionalities. We might not be able to utilize event-loop due to limited support in Java.

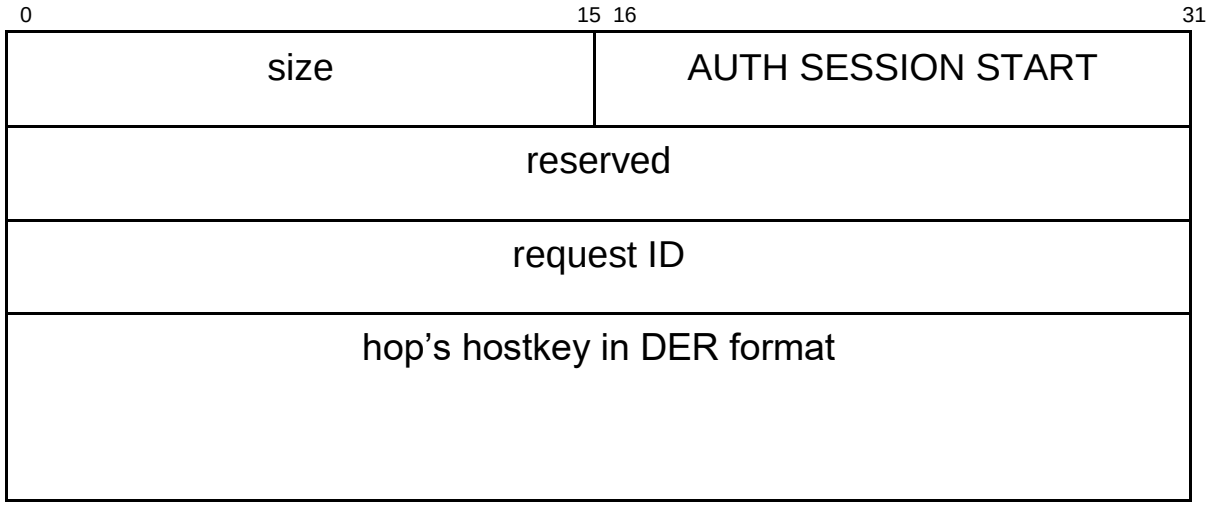
2 Inter-Module Protocol

This section comprises of the details of our inter-module protocol.

2.1 Message Formats

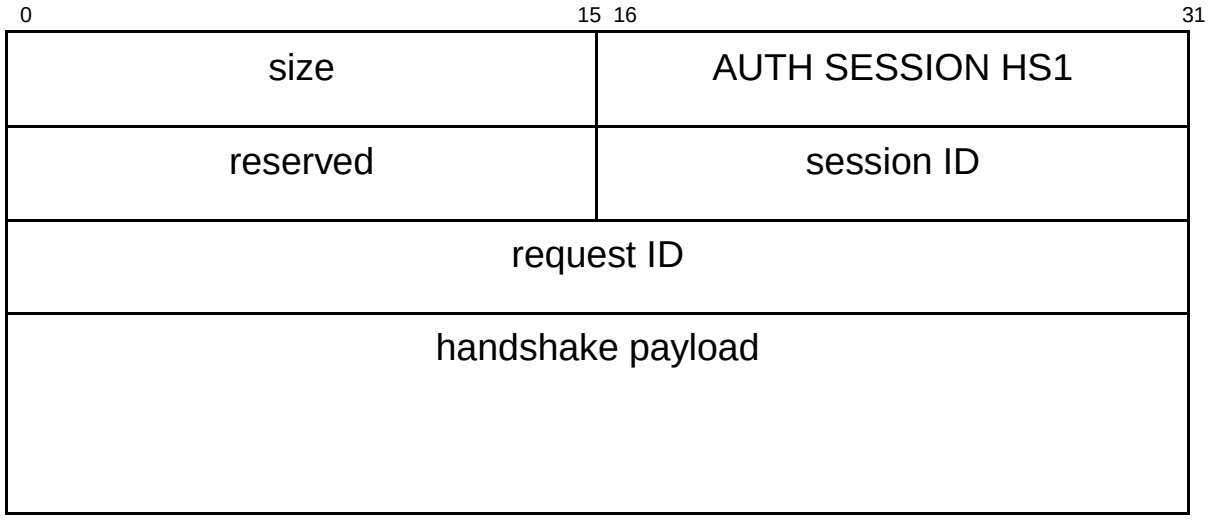
As described in the project specification, there is no direct communication between ONION AUTH modules of different peers. Instead, messages will be forwarded via ONION module of the peers. Therefore, we will abide by the message API defined for inter-module communication between ONION and ONION AUTH modules as follows.

AUTH SESSION START



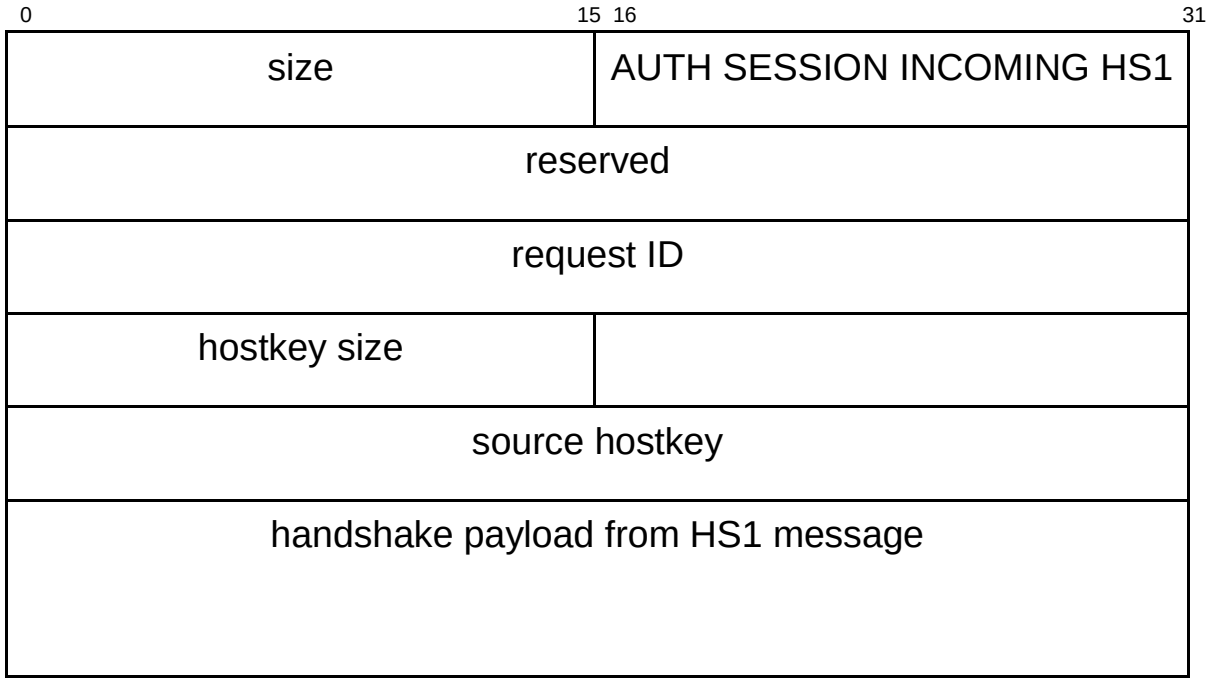
- This message is sent to the peer’s own ONION AUTH module and used to start a session key establishment process with a another peer(a hop) which is identified by a hostkey in DER format.

AUTH SESSION HS1



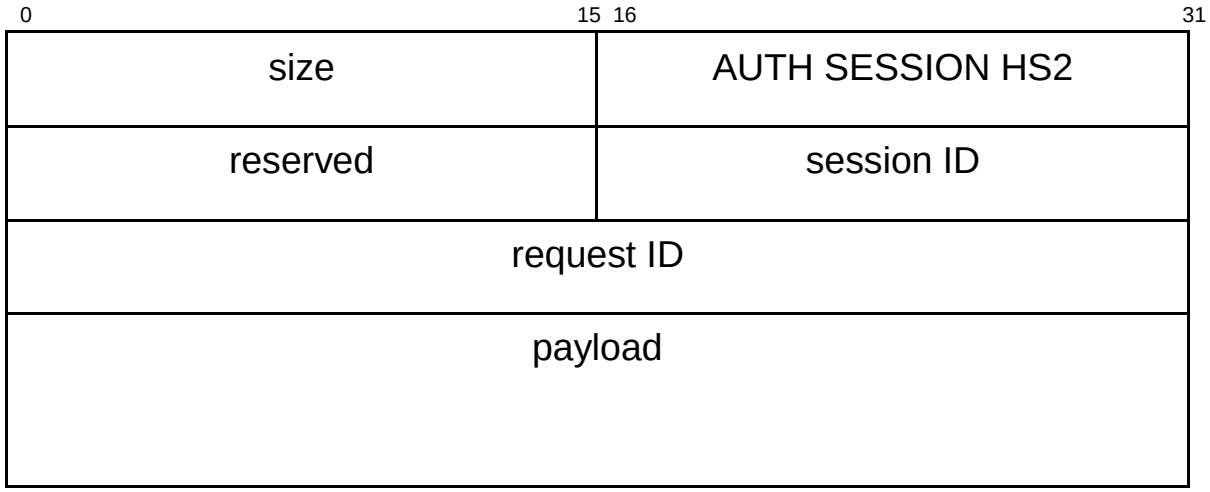
- This message is reply to AUTH SESSION START and will be forwarded via ONION FORWARDING to the hop identified earlier. The field “session ID” uniquely identifies the session.

AUTH SESSION INCOMING HS1



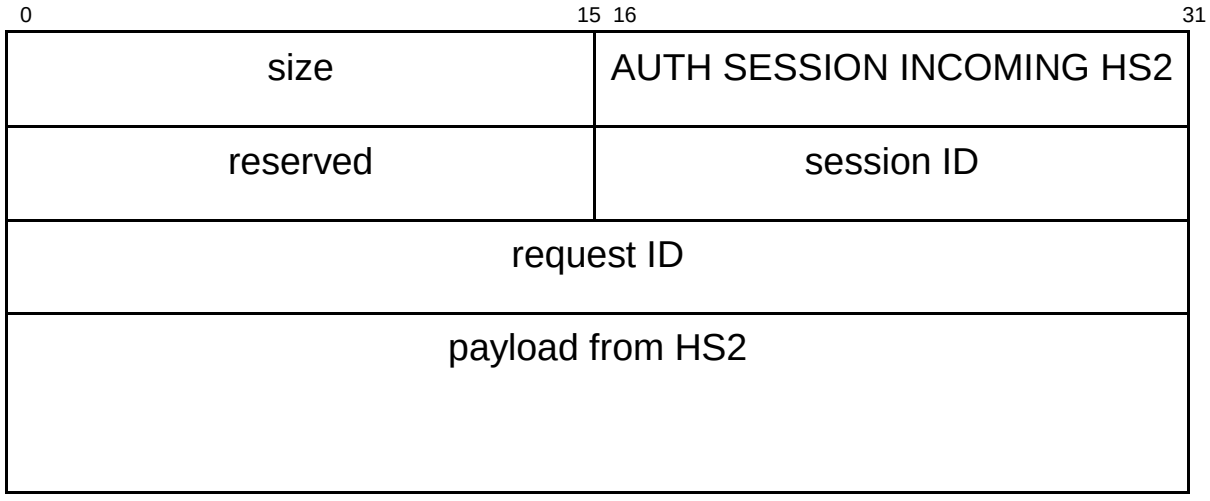
- This message is sent(forwarded) to from the hop's ONION module to its own ONION AUTH module, when it receives AUTH SESSION HS1.

AUTH SESSION HS2



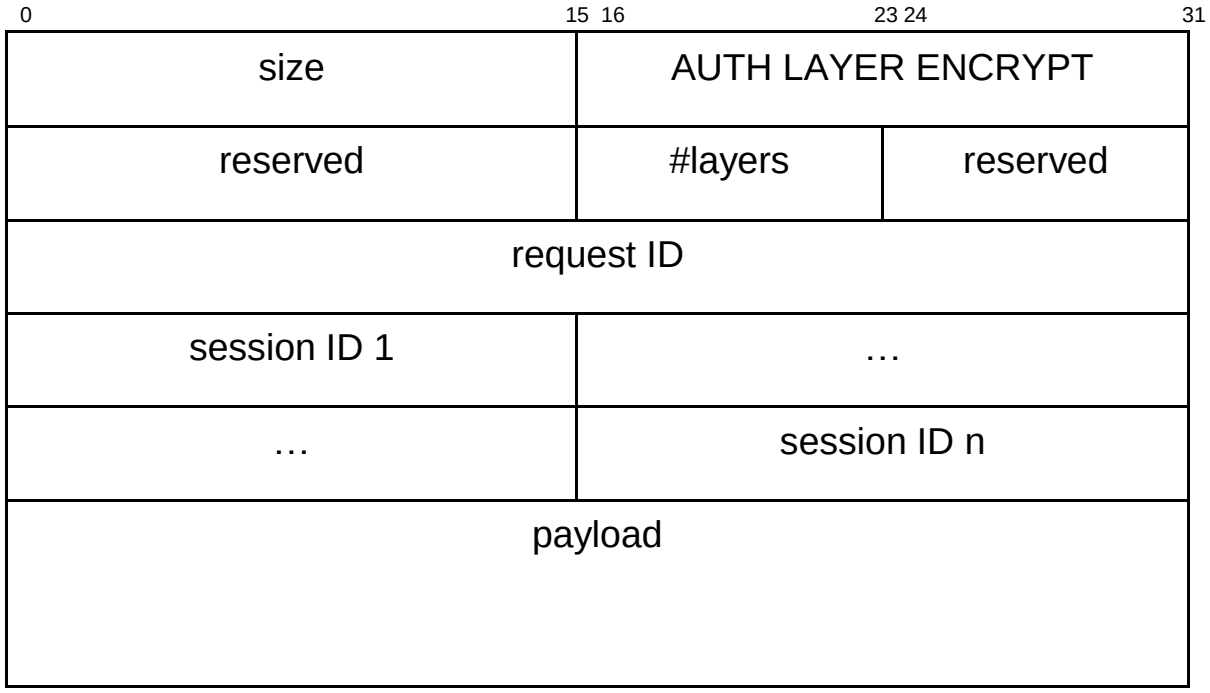
- This message is sent from by the hop's ONION AUTH module in response of AUTH SESSION INCOMING HS1 from its ONION module.This message is then forwarded back to the originator's ONION module.

AUTH SESSION INCOMING HS2



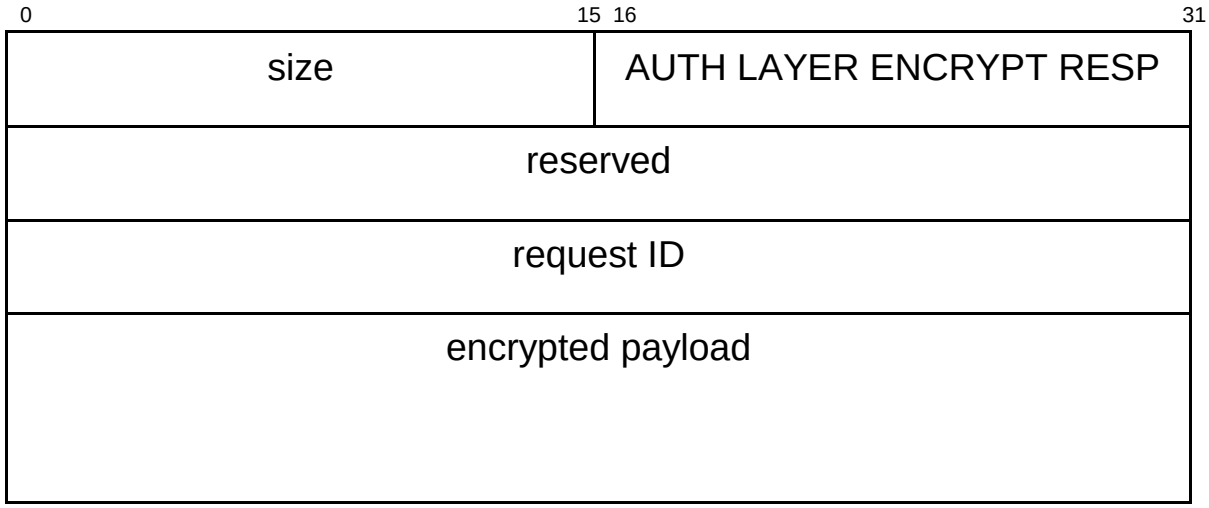
- This is the forwarded message to the originator's ONION AUTH module, when it ONION module receives AUTH SESSION HS2. The field "session ID' is used to verify the correctness of the session.

AUTH LAYER ENCRYPT



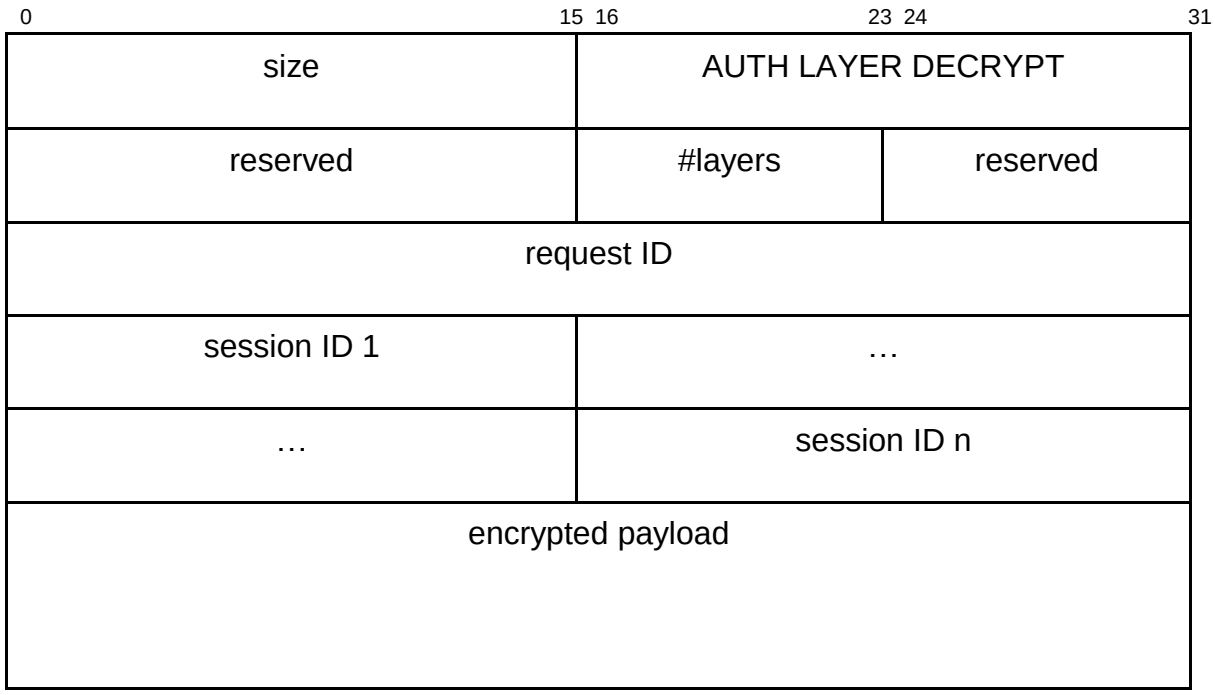
- This message is sent to the originator's own ONION AUTH module and used to initialize chained layered encryption of payload data, once sessions are established with all intermediate hops in order to reach the target remote peer. The field "request ID" is pairwise unique for identifying request and response message pairs.

AUTH LAYER ENCRYPT RESP



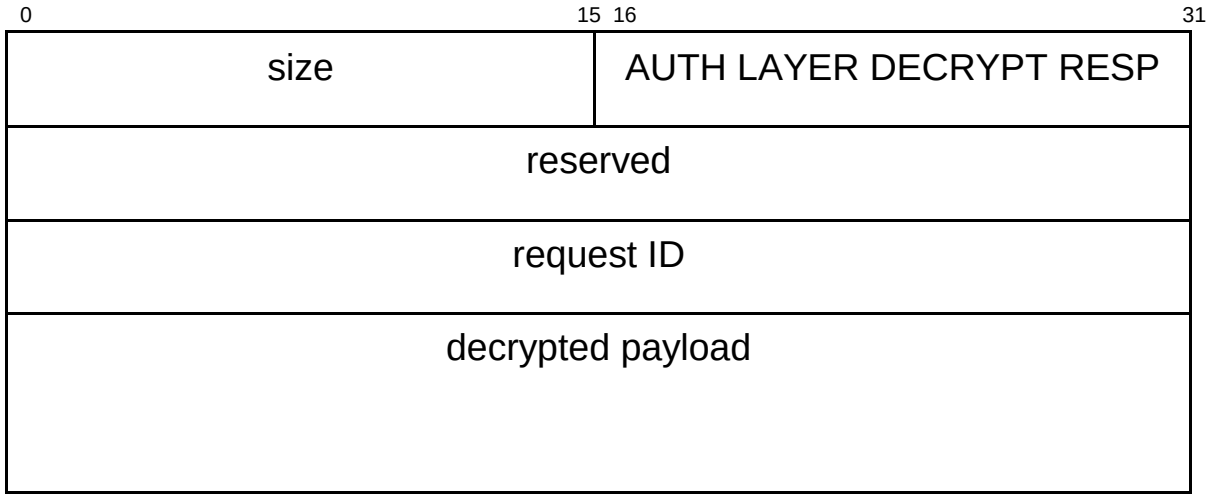
- This message is the reply message in response of AUTH LAYER ENCRYPT to be forwarded to other peers and their respective ONION AUTH modules.

AUTH LAYER DECRYPT



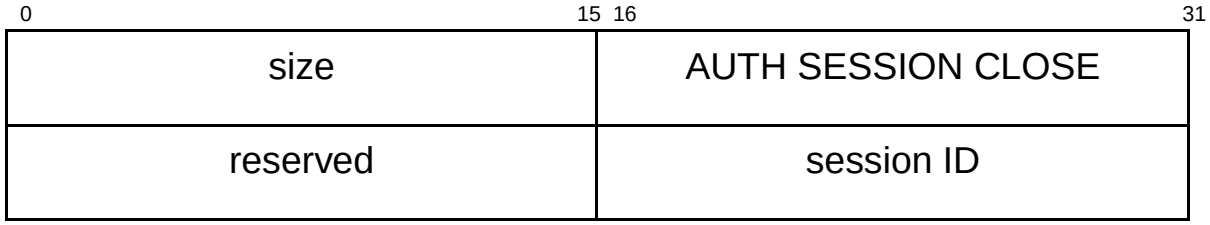
- This message is used for layered decryption of the encrypted payload. The process is again chained and proceeded in the reverse order with respect to encryption.

AUTH LAYER DECRYPT RESP



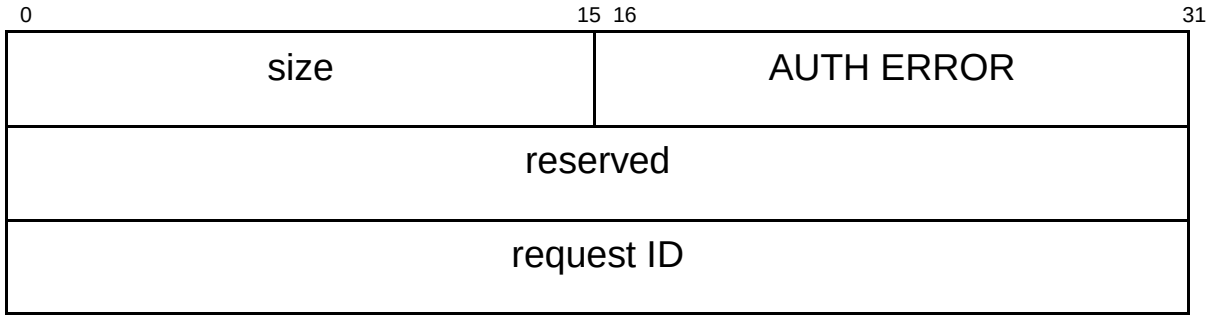
- This message is the response to AUTH LAYER DECRYPT, which is sent back to the peer's ONION AUTH module and forwarded subsequently to the next peer.

AUTH SESSION CLOSE



- This message is used to terminate the given session, with all session related setups being destroyed.

AUTH ERROR



- This message is used to raise and identify an error in case of having one.

Illustration of important concepts:

- Each session will hold a symmetric session key in AES.
- This session key is used to encrypt payload data.
- Key sharing is protected by asymmetric cryptography, i.e. using public key and private keys.
- We adopt the following key sharing protocol between Peer A and Peer B:
    1. ONION AUTH of Peer A generates a symmetric key and encrypts it with Peer B's public key
    2. ONION AUTH of Peer A encapsulates this encrypted payload inside a message and sends it to ONION of Peer A for forwarding to Peer B
    3. ONION of Peer A forwards the message, which is received by ONION of Peer B
    4. ONION of Peer B then forwards this message to ONION AUTH of Peer B
    5. ONION AUTH of Peer B extracts the encrypted payload and decrypts it with Peer B's private key
    6. Now both Peer A and Peer B have the same AES key, thus concluding the key sharing process

2.2 Authentication Process

A hash-based message authentication code will be used to authenticate the modules running on different peers. The hash function utilised is SHA-256 and the secret key is the unique session key shared between 2 peers.

Java offers a MessageDigest class that enables SHA-256 hashing:

```
MessageDigest digest = MessageDigest.getInstance("SHA-256");
byte[] hash = digest.digest(
              sessionKey.getBytes(StandardCharsets.UTF_8));
```

2.3 Exception Handling

The below table lists a few exceptions encountered and the corresponding handling mechanisms.

Exception	Handling
Peer disconnected	An attempt to reconnect will be sent every 20 seconds; if the reconnection fails after 5 attempts, then the connection will be terminated.
Connection interrupted	An attempt to reconnect will be sent every 10 seconds; if the reconnection fails after 10 attempts, then the connection will be terminated.
Corrupted data	The entire message is resent.