IN2194
Final Report

Onion Authentication
Group 40

Rui Bin Choo (03691824)
Danwen Ouyang (03692087)

Table of Contents

1 Software Documentation **2**

1.1 Requirements 2

1.2 How to use 2

2 Protocol **2**

2.1 Session key establishment scheme 2

2.2 Encryption scheme 3

2.3 API messages 3

3 Potential Enhancements **11**

4 Work Distribution **11**

1 Software Documentation

1.1 Requirements

- Java 7 and above

1.2 How to use

- Navigate to the directory *src/OnionAuth/* which contains the two main source files PeerOnionAuth.java and MessageType.java.
- Compile PeerOnionAuth.java which specifies the API for the Onion Auth module by *"javac PeerOnionAuth.java"*
- Instantiate an instance of this class by supplying a port to be listened on. An instance of Onion Auth will be running indefinitely listening for incoming messages.
- To test, an Onion module instance is required, which sends API messages as specified in the project specification to this running instance of Onion Auth. These messages will be then handled accordingly and the responses can be checked.

2 Protocol

2.1 Session key establishment scheme

The process of establishing a session between two peers utilises the Diffie-Hellman key exchange protocol. Each peer will have its own Diffie-Hellman (DH) key pair and a RSA key pair.

The process is as follows.

1. A sends its DH public key to B.
2. B receives A's DH public key and generates a secret key (256-bit AES) using A's DH public key and B's DH private key.
3. B generates a SHA-256 hash of the secret key.
4. B uses its RSA private key to sign the handshake payload containing
    a. The hash of the generated secret key
    b. B's DH public key
5. B sends back its DH public key and the signed handshake payload to A.

6. A reads in B’s DH public key and generates a secret key using B’s DH public key and A’s DH private key.
7. A generates a SHA-256 hash of its generated secret key and compares it to the hash in the handshake payload that had been generated by B.
8. If the hashes match, it indicates that the secret key generated by both peers is identical.
9. A then uses its own RSA private key and signs the payload containing
    a. The hash of its own generated secret key
    b. B’s DH public key that had been sent by B
10. A compares the digests of the 2 signatures.
11. If the signatures match, it indicates that the payload sent by B has not been tampered with.
12. The session key establishment process is now complete.

2.2 Encryption scheme

As specified in the project specification, once sessions are established by a peer to multiple hops, the individual sessions can be chained together to layer-encrypt a given payload. Given that the session IDs run from 1...n, the payload is first encrypted with the session key corresponding to session ID 1, all the way up till that of session ID n.

Each individual cipher encryption is done using GCM mode with a randomly generated IV. The IV is appended to the start of the generated ciphertext and the entire byte array is returned as the encrypted payload.

Additionally, a hash of the original cleartext is generated at the start and appended to the end of the encrypted payload. During the decryption process, after each layer of decryption, the decrypted payload will be hashed and the hashed value is compared to the hash value of the original cleartext. If the hashes match, it indicates that the payload has been fully decrypted to cleartext; otherwise, it indicates that the payload still remains encrypted. The encryption status will be indicated as such in the “E” flag in the AUTH CIPHER DECRYPT RESP message and the corresponding AUTH CIPHER DECRYPT messages.

2.3 API messages

For each incoming message, the message size and message type are first read and a switch-statement based on the message type determines the method to be called.

AUTH SESSION START

0 … 15	16 … 31
size	AUTH SESSION START
reserved	
request ID	
hop’s hostkey in DER format	

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

AUTH SESSION HS1

0 … 15	16 … 31
size	AUTH SESSION HS1
reserved	session ID
request ID	
handshake payload	

- This message is a reply to AUTH SESSION START and returned to the peer’s own Onion module. It will then be forwarded via Onion forwarding to the hop identified earlier.
- The field “session ID” uniquely identifies the session.
- The field “request ID” contains the same value as that in the AUTH SESSION START message, so that the Onion module is able to match the request and response accordingly.
- The handshake payload contains the peer’s DH public key.

AUTH SESSION INCOMING HS1

0 15	16 31
size	AUTH SESSION INCOMING HS1
reserved	
request ID	
handshake payload from HS1 message	

- This message is forwarded from the peer’s Onion module to its own Onion Auth module, when it receives AUTH SESSION HS1 from another peer.
- The handshake payload contains the DH public key of the sending peer.

AUTH SESSION HS2

0 15	16 31
size	AUTH SESSION HS2
reserved	session ID
request ID	
payload	

- This message is sent from the Onion Auth module in response to AUTH SESSION INCOMING HS1 from its Onion module.This message is then forwarded back to the originator’s Onion module.
- The field “session ID” uniquely identifies the session.
- The field “request ID” contains the same value as that in the corresponding AUTH SESSION INCOMING HS1 message.
- The payload contains the peer’s public key and a signed payload containing the hash of the generated secret key and the peer’s public key (more details in Section 2.2 above).

AUTH SESSION INCOMING HS2

0 15	16 31
size	AUTH SESSION INCOMING HS2
reserved	session ID
request ID	
payload from HS2	

- This is the forwarded message to the originator’s Onion Auth module, when its Onion module receives AUTH SESSION HS2.
- The field “session ID’ is used to verify the correctness of the session.

AUTH LAYER ENCRYPT

0 15	16 23	24 31
size	AUTH LAYER ENCRYPT	
reserved	#layers	reserved
request ID		
session ID 1	…	
…	session ID n	
Cleartext payload		

- 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 specific to each peer for identifying request and response message pairs.

AUTH LAYER ENCRYPT RESP

0 15	16 31
size	AUTH LAYER ENCRYPT RESP
reserved	
request ID	
encrypted payload	

- This message is the reply message in response to AUTH LAYER ENCRYPT.
- The field “request ID” contains the same value as that in the corresponding AUTH LAYER ENCRYPT message.
- This message is then forwarded to other peers and their respective Onion Auth modules.

AUTH LAYER DECRYPT

0 15	16 23	24 31
size	AUTH LAYER DECRYPT	
reserved	#layers	reserved
request ID		
session ID 1	…	
…	session ID n	
encrypted payload		

- 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 the encryption process.

AUTH LAYER DECRYPT RESP

<table>
<tr><th>0</th><th>15</th><th>16</th><th>31</th></tr>
<tr><td colspan="2">size</td><td colspan="2">AUTH LAYER DECRYPT RESP</td></tr>
<tr><td colspan="4">reserved</td></tr>
<tr><td colspan="4">request ID</td></tr>
<tr><td colspan="4">decrypted payload</td></tr>
</table>

- This message is the response to AUTH LAYER DECRYPT, which is sent back to the peer's Onion module.
- The field "request ID" contains the same value as that in the corresponding AUTH LAYER DECRYPT message.

AUTH CIPHER ENCRYPT

<table>
<tr><th>0</th><th>15</th><th>16</th><th>23</th><th>24</th><th>31</th></tr>
<tr><td colspan="2">size</td><td colspan="4">AUTH CIPHER ENCRYPT</td></tr>
<tr><td colspan="5">reserved</td><td>E</td></tr>
<tr><td colspan="6">request ID</td></tr>
<tr><td colspan="2">session ID</td><td colspan="4"></td></tr>
<tr><td colspan="6">Payload</td></tr>
</table>

- This message is used by each individual hop of a tunnel to encrypt a payload.
- The flag "E" indicates if the payload is already encrypted or cleartext; 1 for encrypted payload, 0 for cleartext.
- The "session ID" indicates the corresponding session key to be used to encrypt the payload.

AUTH CIPHER ENCRYPT RESP

<table>
<tr><th>0</th><th>15</th><th>16</th><th>31</th></tr>
<tr><td colspan="2">size</td><td colspan="2">AUTH CIPHER ENCRYPT RESP</td></tr>
<tr><td colspan="4">reserved</td></tr>
<tr><td colspan="4">request ID</td></tr>
<tr><td colspan="4">encrypted payload</td></tr>
</table>

- This message is the reply message in response to AUTH CIPHER ENCRYPT.
- The field “request ID” contains the same value as that in the corresponding AUTH CIPHER ENCRYPT message.
- This message is then forwarded to the next hop of the tunnel for the next layer of encryption.

AUTH CIPHER DECRYPT

<table>
<tr><th>0</th><th>15</th><th>16</th><th>23</th><th>24</th><th>31</th></tr>
<tr><td colspan="2">size</td><td colspan="4">AUTH CIPHER DECRYPT</td></tr>
<tr><td colspan="5">reserved</td><td>E</td></tr>
<tr><td colspan="6">request ID</td></tr>
<tr><td colspan="2">session ID</td><td colspan="4"></td></tr>
<tr><td colspan="6">Payload</td></tr>
</table>

- This message is used by each individual hop of a tunnel to decrypt a payload.
- The “E” flag is ignored here.
- The “session ID” indicates the corresponding session key to be used to decrypt the payload.

AUTH CIPHER DECRYPT RESP

0 15	16 23	24 31
size	AUTH CIPHER DECRYPT RESP	
reserved		E
request ID		
decrypted payload		

- This message is the reply message in response to AUTH CIPHER DECRYPT.
- The flag “E” indicates if the payload is still encrypted or already fully decrypted to cleartext; 1 for encrypted payload, 0 for cleartext
- The field “request ID” contains the same value as that in the corresponding AUTH CIPHER DECRYPT message.
- This message is then forwarded to the next hop of the tunnel for the next layer of decryption.

AUTH SESSION CLOSE

0 15	16 31
size	AUTH SESSION CLOSE
reserved	session ID

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

AUTH ERROR

0 15	16 31
size	AUTH ERROR
reserved	

request ID
message type to which the error message corresponds to
Additional message depending on the error (optional)

- This message is used to raise and identify an error in case of having one.
- The “request ID” field contains the ID of the request which caused the error.
- Depending on the type of exception, an additional message might be displayed at the end of the message.

3 Potential Enhancements

The following points are some features that we would have liked to include in our implementation:

- More in-depth error handling
- Multithreading

4 Work Distribution

This section details the individual work that we have undertaken for this project.

- Danwen Ouyang:
    - Implementation of network sockets, streams (DataInputStream and DataOutputStream), and skeletons of API messages.
    - Conversion and reading of hostkey in DER format.
    - MessageType.java.
    - Diffie-Hellman key exchange protocol.
    - Crypto setup including choices of schemes, key-pair generation, and cipher configurations.
    - (Layered) Encryption and (layered) decryption with AES in GCM mode.
    - Handling of mapping of session keys to session IDs.
    - Report writing.
- Rui Bin Choo:
    - Additional implementation of API messages.

- Diffie-Hellman key exchange protocol.
- Mechanism to check on the encryption status of the payload being decrypted.
- Error messages.
- Report writing.