**IMPLEMENTING AES-256 GCM AND RSA-2048 IN A SIMPLE CHAT APP WITH SOCKET PROGRAMMING, SOMEWHAT ADOPTING WHATSAPP KEY MANAGEMENT KEY SYSTEM**

1. **Work Phases**
2. Creating multi user application with multi-threaded socket server.
3. Data communication with Java serialization and socket binary stream, formatted by using Type-Length-Value (TLV format).
4. Securing private chat with encryption AES-256 GCM.
5. Securing AES-256 key with RSA-2048 with PKCS#1 padding.
6. End-to-end testing.
7. **Concept**
   1. When a user connects, the client application will automatically generate an RSA-2048 keypair and AES-256 key.
   2. RSA public key is submitted to the server, while the RSA private key and AES-256 key are kept in each client application memory (cache).
   3. Ordinary message will be sent without encryption to all users.
   4. Private message only will be sent to a specific user with encryption process.
   5. When a user wants to send a private chat to someone, they need to get the list of public keys of all users from the server by submitting a special command: LIST\_PK, then the client application will store the name and public key pairs in its memory (cache).
   6. Then the user (sender) types command: ENCRYPT [recipient] [message].
   7. The client application of sender will:
      1. Look up the [recipient] from public key list in its memory.
      2. Encrypt the message with AES-256 in GCM mode.
      3. Encrypt AES-256 key by using RSA public key of the recipient.
      4. Wrap the cipher text, encrypted AES-256 key, initialization vector (IV), recipient, and sender.
      5. Serialize them into byte array and then send them to socket binary stream with TLV format.
      6. Send them to the server application.
   8. The server application will:
8. Deserialize the message in TLV format.
9. Inspect who the recipient is.
10. Send the message to the recipient.
11. NOTE: Server cannot open the cipher text (end-to-end encryption).
    1. The client application of recipient will:
       1. Deserialize the binary stream with TLV format.
       2. Decrypt AES-256 key with private key.
       3. Decrypt the cipher text with the decrypted AES-256 key.
       4. Show the plaintext to the recipient.
12. **TLV Format**

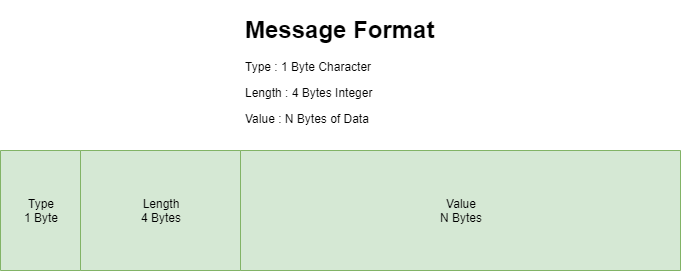


Figure 1TLV Schema from Baeldung

1. **READING LIST**
   1. A Guide to Java Sockets, <https://www.baeldung.com/a-guide-to-java-sockets>
   2. Java Concurrency and Multithreading Tutorial, <https://jenkov.com/tutorials/java-concurrency/index.html>
   3. Java IO Overview, <https://jenkov.com/tutorials/java-io/overview.html>
   4. Read an InputStream using the Java Server Socket, <https://www.baeldung.com/java-inputstream-server-socket>
   5. Introduction to Java Serialization, <https://www.baeldung.com/java-serialization>
   6. RSA in Java, <https://www.baeldung.com/java-rsa>
   7. Java AES Encryption and Decryption, <https://www.baeldung.com/java-aes-encryption-decryption>
   8. Java AES encryption and decryption (with GCM mode), <https://mkyong.com/java/java-aes-encryption-and-decryption/>