Lab session 3 Part II

Securing a Distributed Messaging Service in a Distributed System

J.Gaber

November 2024

gaber@ieee.org

This lab project will give you comprehensive hands-on experience with core distributed system technologies, preparing you for applications in IoT, finance, and secure communication systems.

Throughout the session, you will build a secure messaging system in three phases, delving into various networking and security protocols within distributed systems. This approach will strengthen your understanding of secure messaging using sockets, MQTT, and blockchain technologies, illustrating how these components integrate to form secure, scalable, and reliable architectures.

Target set of this session

By the end of this lab, you will understand how to integrate:

- Sockets and MQTT for secure messaging at different levels of abstraction in distributed systems.
- ECDH, ECDSA, and HMAC for confidentiality, authenticity, and integrity.
- Blockchain for immutable logging and tamper detection.

Part A: Secure Messaging with Sockets and MQTT

In this section, we expand on the foundational concepts introduced in Lab Session 1, where you learned how to create connection-oriented services using sockets. Now, you'll enhance those skills by implementing secure key exchange and message authentication over sockets, before transitioning to MQTT for message routing.

Proceed with the following steps to implement your project:

1. Socket setup for Key Exchange:

    a) Set up a client and server using Python's socket library, where the client initiates a connection, and the server listens.
    b) Use sockets to establish an initial secure connection and perform the Diffie-Hellman (ECDH) key exchange.

2. Key Exchange using ECDH:

    c) Generate ECDH key pairs on the client and server.

d) Exchange public keys over the socket and compute a shared secret.

3. Message signing with ECDSA:

   e) Use the ECDSA private key to sign messages before sending.
   f) The receiver verifies signatures to ensure message authenticity and sender verification.

4. Message Integrity with HMAC:

   g) Using the shared secret from ECDH, create an HMAC for each message to verify integrity.
   h) Send both the signed message and HMAC over the socket to ensure the message was neither tampered with nor altered.

5. Transition to MQTT:

   i) Once the ECDH key exchange is complete, close the socket connection and establish an MQTT connection with the Mosquitto broker.
   j) Use MQTT topics to send and receive messages, including the ECDSA signature and HMAC for each message.

Part B: Secure Messaging with MQTT using paho-mqtt

In this section, you will focus exclusively on secure MQTT messaging using the paho-mqtt library. This will allow you to build a secure publish-subscribe system that leverages MQTT for efficient, scalable messaging without relying on direct sockets.

Proceed with the following steps to implement your project:

1. Install and set up paho-mqtt:

   k) Install paho-mqtt and configure your client and server to publish and subscribe to MQTT topics: pip install paho-mqtt

2. Re-establish Key Exchange over MQTT:

   l) Repeat the ECDH key exchange over MQTT. Publish public keys to a secure topic, calculate the shared secret, and use it as the HMAC key for all future messages.

3. Secure Messaging Workflow:

   m) Publish messages on an MQTT topic with both ECDSA signatures and HMACs.
   n) The recipient will verify the signature and recompute the HMAC to confirm the message's integrity and authenticity.

4. QoS and Security:

    o) Use MQTT's Quality of Service (QoS) levels to explore message delivery guarantees and resilience to network failures.
    p) Discuss how SSL/TLS would typically secure MQTT but emphasize how ECDH, ECDSA, and HMAC add an additional layer of security.

Part C: Blockchain Integration for Immutable Logging

In this final part, you will integrate your knowledge of secure messaging with blockchain technology, building on concepts from Part I of the course. The objective is to use blockchain as an immutable audit log for all messages exchanged over MQTT, ensuring a verifiable record of all transactions.

Here are the steps:

1. Blockchain Setup:

    q) Set up a simple blockchain (using libraries like pycryptodome or a provided template) where each new block logs a secure message.

2. Logging Messages to Blockchain:

    r) After verifying each received MQTT message (signature and HMAC), add the message to the blockchain with the associated metadata (timestamp, sender, message content, and HMAC).
    s) Use the blockchain to record an immutable, verifiable history of all messages exchanged, ensuring no message has been altered.

3. Verification of Integrity with Blockchain:

    t) Implement blockchain validation to detect tampering by verifying the blockchain chain's integrity.
    u) If any block is modified, the chain will break, providing a clear indication of potential data tampering.

4. End-to-End System Testing:

    v) Run the complete system: establish a secure key exchange with sockets, transmit messages over MQTT, and record each verified message to the blockchain.
    w) Conduct tests to verify message flow, blockchain integrity, and response to tampered messages.

Reporting:

Write a report covering the following:

1. How each technology (sockets, MQTT, blockchain) contributed to the overall system.
2. How ECDH, ECDSA, and HMAC ensured security.
3. Challenges faced and potential real-world applications.