DA51 Lab Session 3 Part II: Securing a Distributed Messaging Service in a Distributed System

Introduction

This lab session focused on building a secure, scalable, and reliable messaging service by integrating sockets, MQTT, and blockchain technologies. It highlighted the use of advanced security protocols, including ECDH, ECDSA, and HMAC, to ensure confidentiality, authenticity, and integrity. The project was implemented in three phases: secure messaging with sockets and MQTT, enhanced MQTT messaging, and blockchain integration for immutable logging.

Technologies and Security Mechanisms

1. Sockets

Sockets were utilized to establish a direct and secure connection between client and server. Using Python's socket library, an ECDH key exchange was implemented to create a shared secret, enabling encrypted communication. Additionally, ECDSA provided message authenticity through digital signatures, and HMAC ensured message integrity.

2. MQTT

MQTT, a lightweight publish-subscribe protocol, was employed for efficient and scalable message routing. The paho-mqtt library facilitated secure key exchange over MQTT topics, with subsequent message transmissions incorporating ECDSA signatures and HMACs for security.

3. Blockchain

Blockchain served as an immutable audit log for all exchanged messages. By logging verified MQTT messages with metadata, the system ensured an indelible record of communications. Blockchain integrity validation guaranteed that no messages were altered post-transmission.

Implementation Details

Part A: Secure Messaging with Sockets and MQTT

1. **Socket Setup for Key Exchange:**
    - Client and server established connections using Python's socket library.
    - ECDH key exchange generated a shared secret for secure communication.
2. **Key Exchange Using ECDH:**
    - ECDH key pairs were generated and exchanged between client and server.
    - A shared secret was computed to derive session keys for encryption and HMAC.
3. **Message Signing with ECDSA:**
    - Messages were signed using ECDSA private keys.
    - Signatures were verified on the receiving end to authenticate the sender.
4. **Message Integrity with HMAC:**
    - HMACs were generated using the shared secret to ensure message integrity.
    - Both the signed message and HMAC were transmitted for verification.
5. **Transition to MQTT:**
    - MQTT connection was established using Mosquitto broker.
    - Messages were routed through MQTT topics with ECDSA signatures and HMACs.

Part B: Secure Messaging with MQTT

1. **Setup with paho-mqtt:**
    - The paho-mqtt library was used to implement publish-subscribe messaging.
    - ECDH key exchange was re-established over secure MQTT topics.
2. **Secure Messaging Workflow:**
    - Messages were published with ECDSA signatures and HMACs.
    - Recipients verified signatures and HMACs for authenticity and integrity.
3. **QoS and Security:**
    - MQTT QoS levels were configured to explore delivery guarantees.
    - SSL/TLS was considered for transport-layer security, complementing ECDH, ECDSA, and HMAC for end-to-end security.

Part C: Blockchain Integration

1. **Blockchain Setup:**
    - A simple blockchain was created to log verified MQTT messages.
    - Each block contained metadata, including timestamps, sender details, and HMACs.
2. **Logging Messages to Blockchain:**
    - Verified messages were added to the blockchain, ensuring an immutable record.
3. **Verification of Integrity:**
    - Blockchain validation detected tampering by verifying the chain's integrity.
    - Any modifications to a block resulted in a broken chain.
4. **End-to-End Testing:**
    - The system was tested by running all components together.
    - Tests verified message flow, blockchain integrity, and response to tampered messages.

Security Analysis

1. **ECDH:**

Enabled secure key exchange to establish a shared secret for encryption and HMAC generation.

2. **ECDSA:**

Provided message authenticity through digital signatures, preventing forgery.

3. **HMAC:**

Ensured integrity by validating that messages were not altered during transmission.

Challenges and Resolutions

1. **Challenge:** Implementing seamless integration between sockets, MQTT, and blockchain.

Resolution: Modularized each component and ensured standardized communication formats.

2. **Challenge:** Verifying blockchain integrity with large datasets.

Resolution: Optimized hashing mechanisms and implemented incremental validation.

3. **Challenge:** Handling MQTT QoS levels under high network latency.

Resolution: Adjusted QoS settings and implemented retries for reliable message delivery.

Potential Real-World Applications

1. **IoT Systems:**

Secure communication between distributed sensors and actuators.

2. **Finance:**

Immutable logging of financial transactions for audit and compliance.

3. **Secure Communications:**

End-to-end encrypted messaging applications for personal and corporate use.

Conclusion

This lab session provided a comprehensive understanding of building secure distributed messaging systems. By integrating sockets, MQTT, and blockchain, it demonstrated the synergy between networking protocols and security mechanisms. The project highlights the practicality of using these technologies in real-world applications requiring confidentiality, authenticity, integrity, and accountability.