Cryptographic Secure Messaging System using AES

November 5, 2024

1 Introduction and Motivation

In secure communications, confidentiality, integrity, and data protection are fundamental. Applications like WhatsApp and Signal use *end-to-end encryption* (E2E) to ensure that only authorized users can read messages. This is crucial for privacy and security in our interconnected world.

In this assignment, you'll design a cryptographic messaging application that mirrors such E2E systems. The key security objectives are:

- **Confidentiality**: Only the intended recipient should access the message content.
- **Integrity**: Messages should be protected against tampering, ensuring the content is exactly what was sent.
- **Message Freshness and Replay Attack Prevention**: Using a unique Initialization Vector (IV) with a fixed length of 16 bytes for each message.
- **Username Privacy**: Encrypting usernames with a fixed length (16 bytes maximum) to ensure privacy.

This assignment will familiarize you with the fundamentals of secure messaging while learning about practical cryptographic implementations.

2 Cryptographic Design Components

2.1 Password-Derived Encryption Key

Using a password-based key derivation function (e.g., PBKDF2) to generate a strong encryption key from a shared password ensures that only users with this password can participate in secure communication.

Note: You can assume the password is pre-shared between the server and client.

2.2 AES Encryption with Random IV for Confidentiality

AES encryption is used in Cipher Block Chaining (CBC) mode, with a fixed size 16-byte Initialization Vector (IV) for each message. This is critical for:

- Ensuring message confidentiality and preventing patterns from being recognizable.
- Since AES-CBC is a block cipher, ensure to add padding if the message size is not a multiple of the block size.

2.3 Integrity and Replay Prevention with Random IVs

To prevent replay attacks, every message must use a unique, random 16-byte IV. This ensures that identical messages produce distinct encrypted outputs, making it impossible for attackers to reuse intercepted messages.

2.4 Socket-Based Client-Server Communication

The server will run continuously, accepting a client connection over a fixed IP address and port. This allows clients to securely communicate with the server, similar to real-world messaging app servers.

2.5 Database Storage for Encrypted Messages

To securely log encrypted messages, use an SQLite database. Encrypt both the username (limited to 16 bytes) and message content for privacy and integrity.

```
import sqlite3
from datetime import datetime

def setup_database():
    conn = sqlite3.connect("messages.db")
    cursor = conn.cursor()
    cursor.execute("""
    CREATE TABLE IF NOT EXISTS messages (
        id INTEGER PRIMARY KEY,
        user_id BLOB,
        message BLOB,
        iv BLOB,
        timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
    )
    """)
    conn.commit()
    conn.close()

def store_message(user_id, message, iv):
    conn = sqlite3.connect("messages.db")
    cursor = conn.cursor()
    cursor.execute("INSERT␣INTO␣messages␣(user_id,message,iv,
       timestamp)VALUES(?,?,?,?)",
                   (user_id, message, iv, datetime.now()))
    conn.commit()
    conn.close()
```

3 Assignment Implementation

3.1 Encrypting the Username for Privacy

Limit the username to a maximum of 16 bytes and encrypt it using AES-CBC to prevent unauthorized access to user identities.

Example: Encrypt the username separately from the message, combining it with the encrypted message before transmission.

4 Database and Networking Guidance

To simplify message storage and retrieval:

- Store the encrypted username, message, and IV in the database.
- Ensure that each message includes its own IV for decryption.
- Use SQLite's in-built functions to timestamp each message for tracking.

5 Running the Application

1. Start the server to listen for incoming messages.
2. A client can connect to the server and send encrypted messages.
3. The server securely stores each encrypted message in the database and decrypts each message for verification before printing it.

6 Conclusion

Through this assignment, you'll gain experience with cryptographic principles and practical implementation:

- Understanding encryption standards for confidentiality, integrity, and freshness.
- Implementing replay prevention with unique IVs.

These skills align with real-world secure applications, building a foundation for designing privacy-focused systems.

Submission Guidelines

- Submit your code along with a report detailing your approach in a zip file.
- **Deadline : 12th November 2024 23:59**
- Plagiarism i.e. similarity (more than 15 %) with any part of the code available on the internet or the similarity of the code of two different students will lead to a heavy penalty (may be an F grade in the course or -10 marks).
- 5% marks will be deducted for each day late submission; after a week, you will get zero.