International Institute of Information Technology Hyderabad

System and Network Security (CS5.470)

Lab Assignment 1:
Secure Multi-Client Communication with Symmetric Keys
Hard Deadline: 22-01-2026, 11:59 PM
Total Marks: 100

Note:- *It is strongly recommended that no group is allowed to copy programs from others. Hence, if there is any duplicate in the assignment, simply both parties will be given zero marks without any compromise. The rest of the assignments will not be evaluated further, and assignment marks will not be considered towards final grading in the course. No assignment will be taken after the deadline.* ***You can use C, C++ or Python programming language only.*** *Blind use of cryptographic libraries without protocol logic will result in heavy penalties.*

1. Objective

The objective of this assignment is to design and implement a **stateful, symmetric-key-based secure communication protocol** between a server and multiple clients operating in a hostile network environment.

Definition of Stateful Protocol:
The server and each client must remember and update cryptographic and protocol information from previous messages. A message is considered valid only if it is consistent with the stored state.

Operational Meaning of State:
For each client, both the client and the server must maintain:

- Current round number R
- Current encryption and MAC keys for each direction
- Current protocol phase (INIT, ACTIVE, TERMINATED)

A message is accepted **if and only if**:

- The round number equals the expected round
- The opcode is valid for the current protocol phase
- The HMAC verifies using the current MAC key

Failure of any check must permanently terminate the session.

This assignment emphasizes:

- Protocol state management
- Key evolution (ratcheting)
- Replay and desynchronization resistance
- Explicit handling of active adversarial attacks

2. Problem Description

Consider a centralized analytics server communicating with multiple distributed clients deployed in a hostile network (e.g., industrial control systems or secure sensor networks).

Due to operational constraints:

- Public-key cryptography is unavailable
- Each client is provisioned with a pre-shared symmetric master key
- The network is fully controlled by an active attacker

The goal is to ensure confidentiality, integrity, freshness, and synchronization using only symmetric cryptographic techniques.

3. Threat Model

Assume an active network adversary who can:

- Replay old messages
- Modify ciphertexts and MACs
- Drop or reorder packets
- Reflect messages back to the sender

The adversary cannot break AES or HMAC primitives.

4. System Model

- One server
- Multiple clients
- Each client shares a unique long-term master key with the server
- Communication occurs in discrete rounds
- Loss of synchronization must result in session termination

5. Cryptographic Primitives

Component	**Requirement**
Block Cipher	AES-128
Mode of Operation	CBC
MAC	HMAC-SHA256
Padding	Manual PKCS#7
Randomness	OS-level secure RNG

ECB mode, automatic padding, authenticated encryption modes (e.g., AES-GCM, Fernet)are strictly forbidden.

Manual PKCS#7 padding means that students must explicitly implement the logic to add and remove padding bytes according to the PKCS#7 specification, rather than relying on cryptographic libraries to perform padding automatically.

5.1 Encryption and Authentication Procedure

Encryption (Sender Side)

1. Construct plaintext payload
2. Apply PKCS#7 padding manually
3. Generate a fresh random IV (16 bytes)
4. Encrypt padded plaintext using AES-128-CBC
5. Construct message header fields
6. Compute HMAC over **(Header || Ciphertext)**
7. Transmit (Header|| Ciphertext || HMAC)

Decryption (Receiver Side)

1. Verify round number and direction
2. Verify HMAC **before decryption**
3. If HMAC fails, terminate session
4. Decrypt ciphertext using AES-128-CBC
5. Remove PKCS#7 padding
6. Validate plaintext format

Decrypting before HMAC verification is strictly forbidden.

5.2 Padding Requirements

For AES block size of 16 bytes:

- Padding must always be applied
- Each padding byte must equal the padding length
- Incorrect padding must be treated as data tampering

6. Key Initialization

Each client C_i shares a master key K_i with the server.

Client → Server Keys:

```
C2S_Enc_0 = H(K_i || "C2S-ENC")
C2S_Mac_0 = H(K_i || "C2S-MAC")
```

Server → Client Keys:

```
S2C_Enc_0 = H(K_i || "S2C-ENC")
S2C_Mac_0 = H(K_i || "S2C-MAC")
```

7. Stateful Communication Rounds

Communication proceeds in rounds $R = 0, 1, 2, \ldots$.

Client → Server Key Evolution:

```
C2S_Enc_R+1 = H(C2S_Enc_R || Ciphertext_R)
C2S_Mac_R+1 = H(C2S_Mac_R || Nonce_R)
```

Server → Client Key Evolution:

```
S2C_Enc_R+1 = H(S2C_Enc_R || AggregatedData_R)
S2C_Mac_R+1 = H(S2C_Mac_R || StatusCode_R)
```

Key Evolution Rule:
Keys must be updated only after successful verification, decryption, and validation. Any failure must terminate the session without updating keys.

8. Message Format (with byte size)

```
| Opcode (1) | Client ID (1) | Round (4) | Direction (1) | IV (16)|
| Ciphertext (variable) | HMAC (32) |
```

The HMAC covers all preceding fields.

9. Protocol Opcodes

Opcode	Type	Description
10	CLIENT_HELLO	Client initiates protocol
20	SERVER_CHALLENGE	Encrypted server challenge
30	CLIENT_DATA	Encrypted client data
40	SERVER_AGGR_RESPONSE	Encrypted aggregate result
50	KEY_DESYNC_ERROR	Desynchronization detected
60	TERMINATE	Session termination

10. Mandatory Error Scenarios

- Incorrect HMAC
- Replay attacks
- Message reordering
- Key desynchronization

11. Server-Side Aggregation

The server aggregates numeric data per round and encrypts the result separately for each client using evolved keys.

12. Main Tasks

1. Implement AES-CBC encryption, PKCS#7 padding, and HMAC
2. Implement strict protocol state management
3. Implement secure multi-client aggregation
4. Simulate and demonstrate potential security attacks (based on your understanding) and illustrate how the implemented system effectively mitigates or resists these attacks.

5.3 Library Usage

Students are permitted to use cryptographic libraries **only** for the raw AES block cipher and HMAC primitives.

The following are **allowed**:

- AES-128 block cipher in CBC mode
- HMAC-SHA256 hash function
- Secure random number generation

The following are **strictly forbidden**:

- Automatic padding or unpadding functions
- Authenticated encryption modes (e.g., AES-GCM, AES-CCM)

Violations will result in zero marks for the cryptographic component.

13. Submission Guidelines

Submit:

`<group_number>_lab1.zip`

Containing:

- server.py
- client.py
- crypto_utils.py: Contains cryptographic primitives only (AES-CBC, PKCS#7 padding, HMAC). No protocol or networking logic is allowed.
- protocol_fsm.py: Implements the protocol finite state machine, including round tracking, key evolution, opcode validation, and session termination logic.
- attacks.py: All the attacking scenario
- README.md
- SECURITY.md (must include explanation about how your designed protocol is secure against different attack scenarios)

14. Evaluation Criteria

- Protocol correctness
- Key evolution logic
- Attack handling
- Cryptographic correctness

- Code quality
- SECURITY.md reasoning
- Demo and viva

— End of Assignment —