Department of Telecoms and Networking

Course Title: Cryptography

Term 1 | Year 3

Assignment Title:

Encrypted Messaging CLI (RSA + AES Hybrid)

Lecturer: Mr. Meas Sothearath

Submission date: [20/12/2025]

Student: Seng Sosetha

Student ID's: IDTB100028

Group: 2

I. Introduction / Background

Digital communication is widely used today, but many messaging platforms do not provide full end-to-end encryption. This project implements a secure command-line messaging system using hybrid cryptography:

- **RSA**: for exchanging encryption keys securely
- **AES**: for fast symmetric encryption of messages

The goal is to protect message confidentiality, integrity, and simulate how real-world secure messaging applications like Signal or WhatsApp work.

Motivation: Learn how modern secure messaging works and apply classroom concepts such as encryption, key exchange, and secure coding practices.

II. System Design / Architecture

Architecture:

- Two peers: client and server
- Key generation and exchange (RSA)
- AES used to encrypt/decrypt messages
- Messages sent over TCP socket

Flowchart / Steps:

1. Generate RSA key pair for each peer

2. Exchange public keys between peers
3. Generate random AES session key
4. Encrypt AES key with peer's public RSA key and send
5. Use AES key to encrypt/decrypt chat messages
6. Repeat until session ends

III. Implementation Details

Files and Functions:

- `crypto_utils.py`
  - `generate_rsa_keys()` — generate RSA public/private keys
  - `encrypt_rsa()` / `decrypt_rsa()` — encrypt/decrypt AES session keys
  - `encrypt_aes()` / `decrypt_aes()` — encrypt/decrypt messages with AES
- `peer.py`
  - `server_mode()` — listen for connections
  - `client_mode()` — connect to server
  - Handles key exchange, message encryption/decryption, and chat loop

Libraries Used:

- `pycryptodome` for cryptography
- `socket` for networking

IV. Usage Guide

Setup:

```
python3 -m venv .venv
source .venv/bin/activate   # Git Bash / Linux
pip install -r requirements.txt
```

Run Server:

```
python peer.py --mode server --host 127.0.0.1 --port 5000
```

Run Client:

```
python peer.py --mode client --host 127.0.0.1 --port 5000
```

Example Messages:

```
[Client] Hello!
[Server] Hi! Secure message received.
```

V. Conclusion and Future Work

- Successfully implemented hybrid encrypted messaging CLI
- Demonstrates RSA key exchange + AES message encryption
- **Future improvements:** GUI interface, file transfer, multi-peer chat

---