Cairo University- Faculty of Engineering

Computer Engineering Department

Cryptography – Spring 2025

Project Report

Ahmed Hamed Gaber 9220027	Somia Saad Elshemy 9202666
LCG	Seed Encryption
Stream Cipher	Seed Authentication
Communication	Key Exchange
Report	Report

1. Introduction

This project implements a secure communication system that encrypts and authenticates a message using a combination of:

- **Diffie-Hellman key exchange**
- **AES for seed encryption**
- **HMAC for seed authentication**
- **Linear Congruential Generator (LCG)** based stream cipher

The communication simulation is implemented in Python and tested via local file I/O.

2. System Overview

The system simulates two parties:

- **Sender**: Encrypts a message using a pseudo-random keystream, securely shares the seed using AES, and authenticates it using HMAC.
- **Receiver**: Decrypts and verifies the seed, then decrypts the message using the same keystream.

3. Design Choices

Component	**Design Decision**
Key Exchange	Diffie-Hellman over small prime for simplicity
Seed Encryption	AES in ECB mode (16-byte block, padded)
Authentication	HMAC with SHA-256 to detect tampering
Stream Cipher	LCG to simulate keystream for XOR ciphering
File Exchange	Used binary files for encrypted seed, HMAC, and ciphertext
Language & Libs	Python with `pycryptodome`, NumPy

4. Module Functionalities

4.1 KeyExchange

- `generate_keys()`: Random private key + Computes public key
- `compute_shared_secret_key(other_public)`: Returns shared key.

4.2 SeedEncryption

- `encrypt(seed_bytes, key)`: Encrypts using AES.
- `decrypt(encrypted_seed, key)`: Decrypts seed.

4.3 SeedAuthentication

- `generate_hmac(seed_bytes, key)`: Produces HMAC.
- `verify_hmac(data, key, hmac)`: Verifies HMAC.

4.4 LCG Stream Cipher

- `next()`: Generates one number.
- `n_next(n)`: Generates n numbers for the keystream.