Cairo University
Faculty of
Engineering

Department of Computer
Engineering

Computer Security

One Time Pad Stream cipher

Submitted by

Name	BN	ID
Nesma Osama	60	9220912
Mohamed Ashraf	48	9220658

Introduction

The goal of this system is to securely transmit a message contained in an input file from a sender to a receiver, using the One-Time Pad (OTP) encryption algorithm. The receiver then decrypts the message and saves it into an output file. This ensures the confidentiality of the message during transmission.

In the One-Time Pad (OTP) scheme, the initial seed must be shared secretly with the receiver for decryption. To achieve this securely, two encryption options are available:

ElGamal (asymmetric) for direct public-key encryption of the seed, or

AES (symmetric) for faster encryption, paired with a pre-shared key.

To guarantee the seed's integrity and authenticity, an HMAC (e.g., HMAC-SHA-256) is computed over the seed before transmission

Stages

Establishing a connection between sender and receiver

Established a reliable connection between sender and receiver using Python's socket library with TCP protocol to ensure message delivery.

Shared Configuration File for Sender and Receiver

There are certain attributes that must be shared between the sender and receiver, which can remain constant throughout the communication. These attributes are stored in a shared configuration file. These attributes include:

p: A large prime number used in the ElGamal encryption system.

g: A primitive root modulo **p**, used as the base for the encryption process.

a: The multiplier in the Linear Congruential Generator (LCG), which is used for generating pseudo-random numbers.

c: The increment in the LCG, added to the product of the multiplier and the previous number in the sequence.

m: The modulus for the LCG, defining the range of generated random numbers.

For p → large one

g → 2

They are taken from the parameters provided in the doctor's slide

For a → 16807

m → 2147483647

c →0

We share on the best parameters for LGC

Generate keys for the ElGamal encryption algorithm

Both the sender and receiver randomly generate their private keys. Each private key, denoted as b for the sender and for the receiver, must be an integer in the range $1 \leq a \leq p-1, 1 \leq b \leq p-1$.

From the private keys, both the sender and receiver compute their respective public keys. The public key Kb for the sender and ka for the receiver are calculated using the following formula

$$ka = g^a mod\ p$$

$$kb = g^b mod\ p$$

The sender and receiver now exchange their public keys kb and ka

Encrypt Seed

Generate Random Seed

- The sender generates a random seed between $1 \leq seed \leq m$

Convert Seed to String

- Convert the list of encrypted characters back to a string for transmission.

ElGamal Encryption:

$$m * (ka\text{^}b) mod\ p$$

AES Encryption:

Implement AES as it is a symmetric encryption Algorithm.

Implement HMAC

Generate Key

The sender uses the public key of the receiver (kb) and their own private key (b) to compute the shared key using the formula

$$shared = ka^b \, mod \, p$$

Similarly, the receiver uses the public key of the sender (ka) and their own private key (a) to compute the same shared key:

$$shared = kb^a mod \, p$$

The **shared key** computed by both sender and receiver will be the **same**

Generate HMAC

- Generate HMAC from the shared key and the cipher seed.
- Concatenate the result of HMAC with the cipher seed.
- Send the concatenated result to the receiver.

Verify The Seed and Decrypt It

Receive the Message:

The receiver gets the concatenated message which includes the cipher seed and HMAC.

Verify the HMAC:

- Use the received cipher seed and the shared key to recompute the HMAC.
- Compare the recomputed HMAC with the received HMAC. If they match, it means the message has not been tampered with.

Decrypt the Cipher Seed

If the HMAC is verified, decrypt the cipher seed using the shared key to get back the original seed.

Encrypt the input file

Generate Seed Using LCG

$$x = a * x + c \, mod \, p$$

Read Input File

Read the input file in chunks where each chunk corresponds to the number of bytes in the seed.

XOR Operation

XOR each byte of the message with the corresponding byte of the generated seed.

Send Encrypted Data

First, send the length of the message, then the XOR encrypted message.

Receiver's Side:

- The receiver reads the encrypted file, starting by retrieving the **message length** (which was sent by the sender as the first 4 bytes).
- The opposite of the sender and save the message in output file