CS-457 Introduction to Secure Systems

Spring 2026

Prof. Evangelos Markatos

Assignment 1 - Cryptography & Security

Assigned: 26/02/2026

Due: 20/03/2026

This assignment must be implemented in the C programming language using only the standard library. No external cryptographic or third-party libraries are allowed.

Part 1 – One-Time Pad (OTP) Implementation and Cryptanalysis

In this part, you are required to implement the One-Time Pad (OTP) encryption and decryption algorithm. Encryption and decryption must be performed using the XOR operation. A pseudorandom number generator should be used to generate the keystream (https://en.wikipedia.org/wiki/One-time_pad). The produced ciphertext must be encoded and displayed in hexadecimal format.

Assume that a programmer reused the same one-time pad key to encrypt two English phrases. You are given the following two ciphertexts in hexadecimal format.

- 72814c04ba04a4f8a05f458d2e26bcef7e433c4ee43d5d7f1c9c0bc8acf8d714 c6c9324c30da3dc13f3411e08769b2949535dfd386
- 6f904804a949f6febc5c48c83d75f3e3690b294ca9311b2f04975e83acf79e46 89c9774a26dc68c73e2548ab8b63e187883edb9087

Recover the encryption key and determine the two original plaintext phrases. To assist in identifying valid English words, you may use the following word list: https://raw.githubusercontent.com/dwyl/english-words/master/words.txt.

Part 2 – Caesar Cipher

In this part, you must implement the Caesar cipher encryption and decryption algorithm. Letters should be shifted by a fixed key between 0 and 25. The implementation must preserve uppercase and lowercase letters and ignore non-alphabetic characters. After implementing encryption and decryption, you must write code to break the Caesar cipher using two different methods.

First, implement a brute-force attack that tries all 26 possible shifts and outputs all candidate plaintexts.

Second, implement a frequency analysis attack based on English letter frequency statistics to automatically determine the most likely shift.

Part 3 – Stream Cipher Using XOR and PRNG

Implement a stream cipher by generating a keystream using a seeded pseudorandom number generator (PRNG). Encryption must be performed by XORing the plaintext with the generated keystream. Decryption must be performed by reapplying the XOR operation with the same keystream.

You must demonstrate with examples what happens when the same seed is reused to encrypt two different plaintexts.

Part 4 – Password Guessing Timing Attack

Consider a server that checks passwords character by character and introduces a small delay after each correct character comparison (code below). You are required to implement this vulnerable server logic in C.

Then, implement an attacker program that measures the response time of the server and recovers the secret password one character at a time. Demonstrate experimentally how the attack succeeds even though the server does not directly reveal any information about the password.

```
def insecure_check(guess):
      for i in range(len(SECRET)):
            if i >= len(guess):
                  return False
            if guess[i] != SECRET[i]:
                  return False
            time.sleep(0.03) # intentional leak
      return len(guess) == len(SECRET)
```

Part 5 – Secret Sharing

Three friends have a common bank account. They deposit some money and agree to withdraw it after 10 years. To make sure that no one of them will get the money alone, they decide to split the account password (D) in three pieces.

One of them suggested that if something happens to any of the three, the other two will not be able to withdraw the money. Thus they agreed that any two of them will be able to withdraw the money.

They decided to split the password D as follows: They create the function of a straight line: **y=ax+D**. Each of them will take one point in the line: x1, y1, x2, y2, and x3, y3. Any two points will be enough to reconstruct the line and the secret D.

Write a program to create the three points and a program to reconstruct the secret out of any two points.

Notes

- You need to submit all the .c and .h files you created, a Makefile that compiles them, a Readme file explaining your implementation or unimplemented parts and a test file that utilizes the implemented functions.
- If you implement more helper functions or macros, explain their functionality in the Readme file.
- This assignment should be implemented using C on Linux-based machines.
- You can use the course's mailing list or the Office Hours for questions. However, read the previous emails first since your question might have already been answered.
- Do not send private messages with questions to the TAs, since other students might have the same question and everyone deserves the answer.
- Do not send code snippets or pieces of your implementation to the mailing list when asking a question.
- Submitted code will be tested for plagiarism using plagiarism-detection software.
- The submission is done through **e-learn** platform and the submitted file should be a **single zip** one that contains all of the files of this assignment.
- The use of **AI generated code** is **strictly** prohibited.