HY-457: Introduction to Information Security Systems

Computer Science Department Spring Semester 2025

Assignment 2 “Cryptography Algorithms”

Tutorial: 29/04/2025

Deadline: 15/05/2025

Part A: Cryptography Algorithms

For this section of the assignment, you will implement encryption algorithms and also their decryptions. You are required to create two files: **cs457_crypto.h**, which should contain function declarations, and **cs457_crypto.c**, where you will implement these functions. Only the C programming language should be used, without relying on any external libraries. If you want to examine the corresponding examples of cryptanalysis, you may see the associated Tutorial that is held on the site https://www.csd.uoc.gr/~hy457/assignments.html. You should implement a demo (test file) for the functions described in part A of the assignment, demonstrating successful encryption and decryption processes.

1. The one-time pad is an encryption technique that utilizes a randomly generated key, often referred to as a one-time pad, with a length at least equal to that of the plaintext. During encryption, each bit or character of the plaintext is XOR-ed with the corresponding bit or character of the key.
    - Implement the functions one_time_pad_encr and one_time_pad_decr. These functions should accept the plaintext or ciphertext, its length, and the randomly generated key as arguments, and return the result accordingly.
    - Use a pseudorandom number generator to generate the key, such as /dev/urandom. Since /dev/urandom provides a new random value upon each read, it is essential to generate a random secret key and store it in memory to successfully decrypt the encrypted message.
2. The affine cipher is an encryption method where each letter (either in upper or in lower case) is mapped to its numeric equivalent (’x’ in plaintext and ’y’ in ciphertext), encrypted using a simple mathematical function, especially **(3x + 8) mod 26**, and then converted back to a letter using the function **9(y - 8) mod 26** for decryption.
    - Implement the functions affine_encr and affine_decr. These functions should accept the plaintext or ciphertext and return the result accordingly.
    - It is assumed that the plaintext consists only of letters and/or spaces, and the program should handle letters in both upper and lower cases.
3. Write a decryptor for the simple substitution algorithm that decrypts a ciphertext without knowing the key (cipher alphabet).
    - Decrypt the following: ‘Vrq wdgvr mati, ichhqmm, cz Lqbqem' mct, Gyrabbqm, vrgv hqmvdeyvanq wdgvr wrayr pdceirv ycetvbqmm wcqm elct vrq Gyrgqgtm, gth mqtv zcdvr vc Rghqm kgto ngbagtv mcebm cz rqdcqm, gth kghq vrqk vrqkmqbnqm mlcab zcd hcim gth qnqdo padh;’

   The decryptor will execute the following functionalities in each iteration:

a) Read the ciphertext and prompt the user to enter a mapping (alphabet to cipher alphabet). The tool will display the plaintext that can be decoded so far.

b) Ask the user to enter a partially decrypted word and the program will print the common words that match the pattern in the ciphertext. To do that, you will follow the below steps:
   i. Take as argument a text file and the program will print the frequency of each character.
   ii. Calculate the frequency of the English Dictionary (https://github.com/dwyl/english-words) and the ciphertext using the functionality at (i).

For example: Given the first cipher text as input and the mapping q -> e the following intermediate plaintext should be printed:

```
**e ***** **** ****e** ** *e*e** *** ******e* **** *e********* ***** ***** *******
******e** **e* **** *** ****e*** *** *e** ***** ** ***e* **** ******* ***** ** ****e* ***
***e **e* **e****** ***** *** **** *** e**** ****
```

Enter partially decrypted word: c*t
cat
Next mapping: g -> a

```
**e **a** **** ****e** ** *e*e** *** A*****e* **a* *e********* **a** ***** *******
******e** **e* **** *** A***ea** a** *e** ***** ** *a*e* *a** *a***** ***** ** ****e* a**
*a*e **e* **e****** ***** *** **** a** e**** ****
```

Enter partially decrypted word: th**
this,that,them
Repeat this process until you are able to decode the whole plaintext, you should include the plaintext in the Readme file.

4. The Scytale cipher is an encryption method using a simple device known as a Scytale, which consists of a rod and a parchment strip. The rod is typically a cylindrical object, such as a wooden stick, with a specific diameter. The parchment strip is a long, narrow piece of material, such as parchment or leather.
   - Implement the functions scytale_encr and scytale_decr. These functions should accept the plaintext or ciphertext, as well as the diameter of the rod, and return the result accordingly.
   - During encryption, the parchment strip is wrapped tightly around the rod, and the message is then written along the length of the strip.
   - Decryption involves wrapping the parchment strip around a rod of the same diameter used for encryption. When the strip is wrapped around the rod, the letters align properly, allowing the original message to be read.
   - It is assumed that the plaintext consists of letters, punctuation, and/or spaces, and the program should handle letters in both upper and lower cases. During encryption, spaces and punctuation should be omitted, and they should be added back to the generated plaintext after decryption.

Part B: MD5 Hashing

MD5 a widely used cryptographic hash function that produces a 128-bit (16-byte) hash value. It's often used to store passwords securely by hashing them so that the original password is not stored in plain text.

For this section of the assignment, suppose that you are a cybersecurity analyst testing how quickly you can recover passwords hashed with MD5. You will:

1) Use a dictionary attack with the famous rockyou.txt password list. (Can be downloaded here)
2) Implement a basic brute-force attack for short passwords.
3) Compare the execution time of both approaches.

In order to compute the MD5 hashes you have to use the openssl library

Here are the MD5 hashes:

- d077f244def8a70e5ea758bd8352fcd8
- df53ca268240ca76670c8566ee54568a
- 6b718641741f992e68ec3712718561b8

Part C: RSA Implementation

RSA is one of the first public-key cryptosystems and is widely used for secure data transmission. It relies on the difficulty of factoring the product of two large prime numbers. RSA uses a public key for encryption and a private key for decryption.

RSA is based on some neat number theory:

- It uses two large prime numbers to create a modulus for both encryption and decryption.
- The public key is made of this modulus and an exponent.
- The private key is mathematically related to the public key, but it's kept secret.

Your task is to implement a simplified RSA system step-by-step with **two** functions, one for encryption and one for decryption.

Steps

1) Choose two large prime numbers, p and q.
2) Compute n:
   n = p * q
   (This is the modulus used in encryption and decryption.)
3) Compute Euler's Totient Function:
   ϕ(n) = (p - 1) * (q - 1)
4) Choose a public exponent e such that:
5) 1 < e < ϕ(n) **and** gcd(e, ϕ(n)) = 1
   Some common choices include 3,17,65537

6) Compute the private exponent d such that:
   (d * e) mod ϕ(n) = 1
   (Use the modular inverse of e mod ϕ(n).)
7) Encrypt a message m using:
   c = m^e mod n
8) Decrypt the message using:
   m = c^d mod n

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.
- Use of A.I. generated code is **strictly prohibited**!
- You can submit the assignment by executing the command
  "turnin assignment_2@hy457 $directory_name"
  where "$directory_name" is the directory that contains the source code.