CMPSC443: Introduction to Computer Security and Network Security
Project 1 - Encryption

General Instructions

1. There are **4 tasks** in this project with many sub-tasks under each one. There is one optional **bonus** question (task-1b) which will be awarded with bonus points that add up to your score.
2. The maximum possible score in this project will be **270 points**. Out of which 250 points constitutes to 100% and there is a scope for 20 bonus points.
3. To complete all the tasks in this lab, we recommend using a Linux based operating system. While we have tested the exercises in MacOS and Windows, they might need additional set up for some tasks.
4. If you want to use Windows, set up "minGW" and run all commands in it. This is required for "openssl" application to be installed.
5. All commands, keys, execution of tasks needs to be recorded in your report using screenshots. Make sure it is legible to the grader.
6. Be ready for Demo! Keep your project environment ready for executing one or more tasks to the course staff.

Task 1 – *(50 points + 20 Bonus points)*

a. Frequency Analysis - 1 ***(50 points)***
In this task, you are given a ciphertext encrypted with a mono-alphabetic substitution cipher. It means each English letter will be replaced by another letter. From class, you will learn that this kind of encryption is weak and can easily be cracked by frequency analysis. You should perform frequency analysis to the given ciphertext (ciphertext.txt) and try to decipher the original article.

You can use some online tools or the provided Python program (freq.py) to perform the frequency analysis. freq.py usage: `python3 ./freq.py <filename>`

Once you have the frequency information, you will need to check the common frequency in the link below. And you may want to change some letters in the ciphertext back to plaintext. For example, if we want to replace the letters a, c, and f in in.txt with letters Z, K, and J and output the result to out.txt.

On Linux, you can use the "tr" command such as:
```
$ tr 'acf' 'ZKJ' < in.txt > out.txt
```

You may be able to guess some words when most letters in that word are decoded, so you will find more clues in the decoding process.

Submission:
You need to write all the steps you did, the substitution key, and the decoded text in the report. You may need to attach screenshots to help explain the steps.

Online resources:
https://en.wikipedia.org/wiki/Frequency_analysis
https://en.wikipedia.org/wiki/Bigram
https://en.wikipedia.org/wiki/Trigram

Notes:
- Some letters may not appear in the ciphertext, you do not need worry to about them.
- We will recommend you use uppercase letters to substitute ciphertext. Since the ciphertext will only contain lowercase letters, it is easier to distinguish between original letters and substituted letters.
- The frequencies in our ciphertext may be different from the common frequencies.

[BONUS] b. Frequency Analysis – 2 ***(20 points)***

In this part, you are given a ciphertext encrypted with Vigenère cipher. The ciphertext provided is encrypted with a 3-letter English word. You will need to perform frequency analysis to the ciphertext and find out the key.

Feel free to check the Wikipedia page below to learn more about Vigenère cipher.
https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher

Cipher Text:

VHX IRXCT MJIGI AUQUM QBCGCM QRBGNMGD VQDX KS MJAM KT VCN FCKX UMTNL, LKMINE ITOUNEFU LHQK EKKX NAKIE, VQMINEQ QNXU.

Notes:
- The most often letters in English are E, T, A, O, I.
- Group letters encrypted by the same letter in the key so that you can do the correct frequency analysis.
- The key is a legal 3-letter English word and decrypted sentence is also a legal English sentence.

Submission:
- You need to write all the steps you did, the substitution key, and the decoded text in the report. You may need to attach screenshots to help explain the steps.

Task 2 – *(30 points)*

Hashing is the process of converting a plain text secret data into another value that can be sent to a verifier to verify the authenticity of the sender. A good hash function uses a one-way hashing algorithm, or in other words, the hash cannot be converted back into the original key.

MD5 and SHA-1 are some examples of hashing algorithms that are now deemed to be too weak for hashing purposes. Hence, many organizations have deprecated these algorithms for hashing.

a. Plain Hashing – 1 ***(10 points)***
Following is the hash generated for some secret data –
`7634aead3978678f19debc8e7e5d7d43`

Find the original text that was used to generate this hash. Report the original text, the hashing algorithm used and how did you find the same!

b. Plain Hashing – 2 ***(10 points)***
Following is the hash generated for some secret data –
`4170ac2a2782a1516fe9e13d7322ae482c1bd594`

Find the original text that was used to generate this hash. Report the original text, the hashing algorithm used and how did you find the same!

c. Hashing – 3 ***(10 points)***
From the previous steps, it's evident that simple character strings and easy passwords can easily be "un-hashed". Without using stronger passwords or stronger hashing algorithms, can you suggest one technique to build stronger Hashes that cannot be easily un-hashed? Explain.

Task 3 – *(95 points)*

a. Explain ECB (Electronic Code Block) encryption in 2-4 lines. ***(10 points)***

b. Explain CBC (Cipher Block Chaining) encryption in 2-4 lines. ***(10 points)***

c. Which one is better and more secure? Why? ***(5 points)***

d. What is "Salting" in cryptography? Why do we use it in encryption? ***(10 points)***

e. Experiment task for two types of encryptions. Follow the steps:- ***(35 points)***

Step 1 - Using the "openssl" tool in command line (Should work in all platforms. Linux preferred) create two sets of keys:-

i -> ECB key without salting. Encryption algorithm should be AES 128 bit.
ii -> CBC key and IV (Initial Vector) pair without salting. Encryption algorithm should be AES 128 bit.
Figure out how to create the keys. Take a screenshot showing the command used to create the keys and the keys themselves. These screenshots must be recorded in your report.

Step 2 - Create a ".txt" file and name it "assign1.txt". Add 4-5 lines of content in the file.
Take a snapshot of the file and record it in your report.

Step 3 - ENCRYPTION using ECB
Using the Key generated for ECB in Step 1, encrypt the file assign1.txt and name it as "assign1_ecb.txt.enc"
Figure out how to do this and record the command used here in your report.
Open the encrypted file using your "vim" editor or any other command line editor. Take a snapshot of the encrypted file's contents and record it in your report.

Step 4 - DECRYPTION using ECB
Decrypt the "assign1_ecb.txt.enc" file and print it out in your standard output.
Figure out how to do this and record the command and output in your report.

Step 5 - ENCRYPTION using CBC
Using the Key and IV generated for CBC in Step 1, encrypt the file assign1.txt and name it as "assign1_cbc.txt.enc"

Figure out how to do this and record the command used here in your report.

Open the encrypted file using your "vim" editor or any other command line editor. Take a snapshot of the encrypted file's contents and record it in your report.

Step 6 - DECRYPTION using CBC
Decrypt the "assign1_cbc.txt.enc" file and print it out in your standard output.
Figure out how to do this and record the command and output in your report.

f. Image File Encryption using different encryption modes. ***(20 points)***

We have provided an image file named “psu.bmp” in the handout files for this assignment. Image files can also be encrypted using the same keys that were generated in the previous steps.

Encrypt the image file “psu.bmp” using ECB mode (key from Step 1) and name the file as “ecb.bmp”. Similarly, Encrypt the image file “psu.bmp” using CBC mode (key and iv from Step 1) and name the file as “cbc.bmp”.

Open all the three files - psu.bmp, ecb.bmp, cbc.bmp using any preferred Image Viewer software and take a screenshot of them individually. Record these screenshots in your report.

Observe the encrypted images “ecb.bmp” and “cbc.bmp”. Can you infer anything about the original picture from these image files? Explain.

NOTES: You might not be able to readily open the encrypted files in any image viewer as the header of these files are inconsistent with bmp extension. Try the following commands on your terminal to be able to open the encrypted image files:

```
dd if=psu.bmp of=ecb.bmp bs=1 count=54 conv=notrunc
```

g. What type of Cryptography encryption was this? Symmetric or Asymmetric? Explain your answer. ***(5 points)***

NOTES: For this task, record all commands, contents of files, outputs in your report.

Task 4 – *(75 points)*

You are one of the employees at "Crypto443" company that exchanges data with certain authorized customers/clients over the internet. For a long time, you have been using the key generated in your previous task (CBC) to do so.

For example, you generate a CBC AES 128 bit key & IV pair and share it secretly with your authorized customer and they use it to encrypt data while sending it to you.

Recently, you have observed that these customers are not very responsible with the shared keys. Attackers can get these keys and decrypt confidential data sent in the communication. To resolve this, you want to adopt public-key RSA encryption.

We use the "openssl" tool in command line (Should work on all platforms. Linux preferred).

a. Create a private key using openssl with the following specifications and save it in a file named "private_key.pem" – *(5 points)*

RSA encryption algorithm
2048-bit long modulus
Triple DES encryption for protecting private key itself

Hint: Use the command: `openssl genrsa -des3 -out private_key.pem 2048`
Keep a copy of the private key and the passphrase in your report.

b. Extract a public key from the private key generated in the previous step and name it as "public_key.pem". *(15 points)*

The company "Crypto443" keeps the private key safe and shares the public key to its customer. Now as a customer for the company, you need to encrypt any data sent to the company.

You are given two files "task4_big.txt" and "task4_small.txt". (See Lab files)

c. Encrypt the file "task4_small.txt" using the public key generated in step b and save it in a file called "task4_small_encrypted.txt". *(15 points)*

Were you successful in completing this encryption?

If No, why do you think so and how can you overcome this limitation? Explain in 1-2 lines.

If yes, Open the encrypted file using your "vim" editor or any other command line editor. Take a screenshot of the encrypted file's contents and record it in your report.

d. Encrypt the file "task4_big.txt" using the public key generated in step b and save it in a file called "task4_big_encrypted.txt". *(15 points)*

Were you successful in completing this encryption?

If No, why do you think so and how can you overcome this limitation? Explain in 1-2 lines.

If yes, Open the encrypted file using your "vim" editor or any other command line editor. Take a screenshot of the encrypted file's contents and record it in your report.

e. If any of the previous steps were successful, then decrypt back the files using the private key generated in step "a" and compare the result with the original file contents. Record the same in your report. ***(15 points)***

This exercise works great when the customer (with your public key) wants to send data to you (Crypto443 company). Since you possess your private key, ONLY you can decrypt the data. The same exercise will not work when you want to send data to the customer.

f. How can you use this exercise (Steps a - e) for the entire two-way communication between you and your customer? ***(10 points)***