COSC 490 Network Security

Project 2: Cryptography algorithms implementation

This programming assignment contains 4 parts. The first three parts will use the encryption/ decryption techniques we learn to encrypt/decrypt a file. The last part will implement a hash function similar to SHA.

For the first three parts, each one should take a key from the input and use it to encrypt the plaintext into ciphertext. The program should then go into an infinite loop. At the beginning of each iteration, it asks a user if (s)he wants to encrypt or decrypt the text. Possible replies are:

e – to encrypt
d – to decrypt
q – to quit

The program then asks for a filename that contains the input text, performs requested operation and displays the output on the screen. From Part 1 to Part 3, encryption and decryption should preserve non-alphabetic characters such as commas, question marks, spaces, etc. They should also preserve case information but your mapping should be consistent, i.e. if Y maps into K then y should map into k.

In all parts of this program you can assume that maximum input text length will be 200 characters. You must test for other irregularities in the input such as long key, wrong option at the input, etc.

Note, you only write one program that contains all four choices. But it is strongly recommended that you write one by one and then integrate them together. Also you need to make sure that your program can be run in our school Linux lab.

Part 1: Caesar Cipher

The key is one character. Let us suppose that each key character from 'a' to 'z' denotes a number. Suppose 'a' stands for 0, 'b' for 1, 'c' for 2 and so on. So, in Caesar cipher, each character key value will denote the shift to be done for that particular letter.

Example: Let key = d

Plaintext alphabet:	a b c d e f g h i j k l m n o p q r s t u v w x y z
Ciphertext alphabet:	d e f g h i j k l m n o p q r s t u v w x y z a b c

A message of "Meet me after the toga party." enciphers to "Phhw ph diwhu wkh wrjd sduwb."

Part 2. Monoalphabetic Cipher

The alphabets mapping between plaintext alphabet and ciphertext alphabet is created by first writing out a keyword, removing repeated letters in it, then writing all the remaining letters in the alphabet.

Examples: Using this system, the key "zebras" gives us the following alphabets mapping:

Plaintext alphabet: abcdefghijklmnopqrstuvwxyz
Ciphertext alphabet: zebrascdfghijklmnopqtuvwxy

A message of “Flee at once. We are discovered!” enciphers to “Siaa zq lkbr. Va zoa rfpbluaoar!”

In this program, we assume the key length is 6 for monoalphabetic cipher.

Part 3. Poly alphabetic Cipher

A polyalphabetic cipher is a vast improvement over the techniques show in Parts 1 and 2. The basic principle in all polyalphabetic ciphers is the use of multiple but different monoalphabetic substitution techniques.

In this program, we can assume that the monoalphabetic substitution techniques is Caesar Cipher. For example,

Let key = ad

Plaintext = abcd
Thus cipher text = aecg (Shifts of 0,3,0,3)

Let key = abc

Let plaintext = xyz (The message to be encrypted)
Thus cipher text = xzb (Shifts of 0,1,2 respectively)

In this program, we assume the key length is 3 for polyalphabetic cipher.

Part 4. TTH - Toy Tetragraph Hash

The program will implement the simplest “serious” SHA that operates on letters instead of binary data: Toy Tetragraph Hash (TTH) (See appendix for the detail). Your sample input of the message is “I leave twenty million dollars to my friendly cousin Bill.” Your program will run TTH and get the output.

Submission:

1. Source code and make file
2. Sample input file and outputs.
3. Maintain one copy of your source code, make file, sample input file in our Linux machines and do not change them after due date.
4. A project report which contains:
    1) A description of the encryption/decryption or hash algorithm you used for each part.
    2) A discussion for each algorithm you used. What are possible problems/ weakness with your algorithm and how would you solve them?
    3) Citation of any reference/help

Submit one soft copy through MyClasses @ SU and hand in one hard copy on the class time of the due date.

Grading Rubric:

The total points for this project are 100. Each part counts 25 points. They are distributed as the following:

Algorithm implementation 15
Description of your algorithms 5
Discussion of your algorithms 5

Extra points are available for any additional fun implementations.

Appendix:

Toy Tetragraph Hash (TTH)

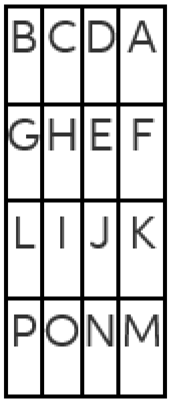
Source: "Network Security Essentials", by William Stallings, Prentice Hall, 5th edition, 2013

Given a message consisting of a sequence of letter, tth produces a hash value consisting of four letters. First tth divides the message into blocks of 16 letters, ignoring spaces, punctuation, and capitalization. If the message length is not divisible by 16, it is padded out with nulls. A four-number running total is maintained that starts out with the value (0,0,0,0); this is input to a function, known as compression function, for processing the first block. The compression function consists of two rounds. Round 1: Get the next block of text and arrange it as a row-wise 4*4 block of text and convert it to numbers (A=0, B=1, etc.). For example, for the block ABCDEFGHIJKLMNOP, we have:

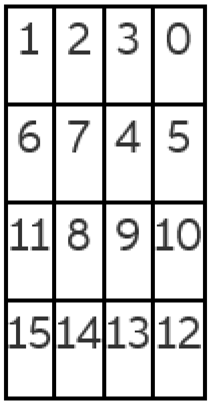
A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Then add each column mode 26 and add the result to the running total, mode 26. In this example, the running total is (24,2, 6, 10). Round 2: using the matrix from round 1, rotate the first row left by 1, second row left by 2, third row left by 2, and reverse the order of the fourth row. In our example:



B	C	D	A
G	H	E	F
L	I	J	K
P	O	N	M



1	2	3	0
6	7	4	5
11	8	9	10
15	14	13	12

Now, add each column mode 26 and add the result to the running total. The new running total is (5,7,9,11). The running total is now the input into the first round of the compression function for the next block of the text. After the final block is processed, convert the final running total to letters. For example, if the message is ABCDEFGHIJKLMN