CSCI 458 – Network Security & Management

Lab 2 –Introduction to Cryptography

Objective

To experiment with substitution ciphers and to practice with Python scripting language. In this lab, you will start from a sample Python program, and then proceed to edit it to create a more generic cryptographic algorithm using substitution cipher. After that, you will edit the code again to produce a more efficient and flexible substitution cipher.

READ EVERYTHING CAREFULLY TO AVOID UNNECESSARY MISTAKES AND TO FULLY UNDERSTAND WHAT YOU ARE DOING

Background

In earlier times, before computers, cryptography consisted of character-based algorithms. Different cryptographic algorithms either substituted characters for one another (*Substitution Ciphers*) or transposed characters with one another (*Transposition Ciphers*).

To describe a cryptographic system, we usually adopt the following terminology, as illustrated in Figure 1:
- Plaintext message (M): the message to be encrypted
- Cipher (E): the encryption algorithm
- Ciphertext (C): the encrypted message

Hence, E(M) = C.

Plaintext can be any type of message, such as a text file, binary data, etc. In this lab, we will work with simple text messages.

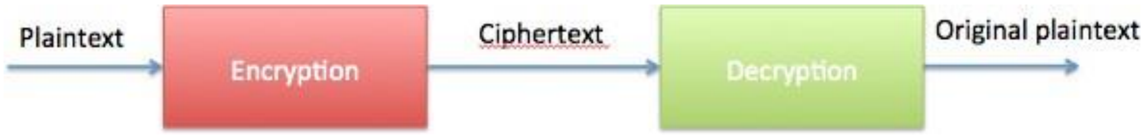

Figure 1: Encryption and Decryption

Task 1: Substitution Cipher – Caesar cipher function

The famous Caesar Cipher, in which each plaintext character is replaced by the character three to the right modulo26 ("A" is replaced by "D," "B" is replaced by "E," …., "W" is replaced by "Z," "X" is replaced by "A," "Y" is replaced by "B", and "Z" is replaced by "C", and so on) is a simple substitution cipher.

Note that it rotates. That's why it's we need to add the modulo26, because we have 26 letters in the alphabet (in Python, the modulo operator is the percent symbol, so modulo26 is written as "%26")

1 – Download substitution_functions.py from D2L.
2 – Open the file using the IDLE editor (or your IDE of choice) in your desktop.
3 – To run the program, click on Run Module.

As you can see, the code is very simple and it is meant to help you understand basic Python syntax and a simple substitution cipher algorithm. It uses functions which can be re-used in other programs.

Encryption algorithms are written as functions (or methods_, in the following format:

```
def encryptCaesar(plaintext):
  #a sample code was given to you
  return  ciphertext

def decryptCaesar(ciphertext):
  #here you use the inverse operation of your encryption)
  return  plaintext
```

Also, please pay attention to the indentation under the "while", "for", "if", and "return" statements.

Play with the program, and run it a few times to examine its behavior.

Edit the program so that it won't crash if plaintext has different characters (such as spaces and numbers). You could consider different approaches to resolve this issue. In one possible approach, if the plaintext character is not in the alphabetList, you can choose to just copy it to ciphertext without making any substitutions. Feel free to make your own choices but your code SHOULD NOT crash in the presence of spaces or numbers.

Question 1: Encode the following plaintext messages (M) using Caesar cipher encryption.
M = zoo, E(M) = ________________________________
M = xray, E(M) = ________________________________
M = rellis, E(M) = ________________________________
M = college station, E(M) = ________________________________
M = csci458, E(M) = ________________________________

Create the decryption function as well. It should be the inverse operation of Ceasar cipher. Test the previous words and see if you can recover the plaintext.

Task 2: Call Caesar Cipher from your client and server program

In this task, edit the client and server code you did in the Lab 1, to use the Caesar cipher encryption you just completed. Hence, all messages transmitted from client to server and from server to client will be encrypted with the general substitution cipher encryption.

Make sure to import your functions file. For instance, you can use this command:

```
from substitution_functions import *
```

Note: In the client and server, you call the encrypt function before you send the message, and the decrypt function when you receive it.

Question 2: Show the results of your client and server program running. For instance, your client sends the words we used before, such as "zoo", "xray", "rellis", "college station", "csci458". Then you can show what the server receives and the results of the decryption for each message.

Task 3: ROT13 Cipher – simple test + client and server

In this task, edit the code to create the ROT13 cipher by adding a new function. Here is how it works.

ROT13 is a simple encryption program commonly found on Unix systems. In this cipher, “A” is replaced by “N,”, “B” is replaced by “O,” and so on. Every letter is rotated 13 places. Encrypting a file twice with ROT13 restores the original file. So the encryption is the same as the decryption. Now write your own ROT13 program and run it a few times.

First test your ROT13 without using the client and server program (at the end of the functions file).
Question 3: Encode the following plaintext messages using ROT13 encryption.
M1 = zoo
M2 = xray
M3 = rellis
M4 = college station
M5 = csci458

C1 = ROT13(M1)= ________________
ROT13(C1) = ____________________

C2 = ROT13(M2)= ________________
ROT13(C2) = ____________________

C3 = ROT13(M3)= ________________
ROT13(C3) = ____________________

C4 = ROT13(M4)= ________________
ROT13(C4) = ____________________

C5 = ROT13(M5)= ________________
ROT13(C5) = ____________________

Now test the code using your client and server.

Question 3: Show the results of your client and server program running for the new function ROT13. For instance, your client sends the words we used before, such as “zoo”, “xray”, “rellis”, “college station”, “csci458”. Then you can show what the server receives and the results of the decryption for each message.

Task 4: A simple Substitution Box (S-Box) - simple test with client and server

Many encryption algorithms nowadays use substitutions in one of its steps to encrypt the data. They often have a substitution box (S-box) with fixed substitution values. It is meant to add “non-linearity” to the encryption operation.

Let’s create a simple “S-box” function to substitute each character. For the encryption, every letter x in Table 1 is replaced with the value S(x).

Table 1 - S-Box for our character-by-character encryption – the input is the plaintext x

x	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
S(x)	c	l	i	m	h	x	e	z	s	t	b	p	v	o	u	d	r	y	j	q	k	a	g	w	n	f

For decryption we should use the inverse S-box as in Table 2. But the steps in your function should be the same.

Table 3 – Inverse S-Box for our character-by-character encryption – the input c is the ciphertex

c	c	l	i	m	h	x	e	z	s	t	b	p	v	o	U	d	r	y	j	q	k	a	g	w	n	f
S^{-1}(c)	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

First test your S-box encryption and decryption without using the client and server program (at the end of the functions file).

Question 4: Encode the following plaintext messages using S-Box encryption and inverse S-Box for decryption. .
M1 = zoo
M2 = xray
M3 = rellis
M4 = college station
M5 = csci458

C1 = sBox(M1)= ________________
inv_sBox(C1) = _______________

C2 = sBox(M2)= _____________
inv_sBox(C2) = ___________________

C3 = sBox(M3)= _____________
inv_sBox (C3) = _______________

C4 = sBox(M4)= _____________
inv_sBox (C4) = _________________

C5 = sBox(M5)= _____________
inv_sBox (C5) = _________________

Now test it using your client and server.

Task 5: Three rounds of encryption in your client and server code

Apply multiple encryption steps (or “encryption rounds”) before you send your message over the network using your client. Please follow this algorithm:

C1 = encryptCaesar(message)
C2 = ROT13(C1)
C3 = sBox(C2)

You should send C3 over the network.

The receiver (e.g., the server) should apply the decryption operation to C3 in the reverse order:

inv_sBox
ROT13
decryptCaesar

The receiver (e.g., the server) should apply the decryption operation to C3 in the reverse order. Make sure you can recover and print the original plaintext.

<u>Question 5:</u> Attach the results of your client and server windows showing the messages that you sent. To make sure, test them with the words we used before:

Zoo, xray, rellis, college station, csci458

<u>Question 6:</u> Attach your final client, server and functions file. Make sure to add comments to fully explain/document your code. Code without comments will not be reviewed.