SIL765: Networks and System Security

Semester II, 2021-2022

Assignment-1

January 19, 2022

Problem-1: Basic Cryptanalysis

Background

Substitution is one of the essential ingredients of a secure encryption algorithm. In fact, the first few ciphers in the human history utilized only substitution. However, such substitution-only ciphers can be easily broken using basic methods of cryptanalysis. In this assignment, you will break a substitution cipher where the original characters (e.g., letters) are replaced with other characters (e.g., numbers, symbols, and other letters).

To-Do List

Let us consider the substitution cipher where the set of the plaintext characters (the letters of the English alphabet) and the set of the ciphertext characters are as follows.

Plaintext Characters	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 Characters	1	2	3	4	5	6	7	8	9	0	@	#	$
	z	y	x	w	v	u	t	s	r	q	p	o	n

You have been given two ciphertexts that are encrypted using this substitution cipher.

1. Using any cryptanalysis method, decipher the secret key, i.e., the sequence of the 26 ciphertext characters corresponding to the traditional sequence of the English alphabet. For example, the secret key is "3456120987pqrstzyxwvu$#@no" when the plaintext character "a" could be mapped to the ciphertext character "3", the plaintext character "b" could be mapped to the ciphertext character "4", and so on.

2. Also, decipher the ciphertext to obtain plaintext. Note that your deciphering algorithm should be generic, i.e., even if you are given a ciphertext other than the given ones, it should output the corresponding plaintext.

Hint: You can try frequency analysis of characters. Also, the space, comma, semicolon, exclamation mark and full-stop characters are not part of the encryption/decryption process. Hence, you can use them as anchors to decrypt other characters.

Given Files

- **ciphertext-1.txt** - ciphertext-1 in the .txt format file.
- **ciphertext-2.txt** - ciphertext-2 in the .txt format file.
- **script.zip** - Illustrative example in python

Expected Submission

- **decipher_text** (the source code in any appropriate format): Given a ciphertext as the input, the code should
  - print the ciphertext, deciphered plaintext, and deciphered key
  - output the deciphered plaintext and deciphered key
- You can also share any other relevant files. For instance,
  - **Makefile** (the make file to cleanly execute your source code)
  - **dictionary.txt** (a dictionary file containing words)
- **readme.pdf**: This should include how you deciphered the plaintext, i.e., the approach used for cracking the cipher. You have to include the mapping of plaintext characters to the ciphertext characters for each ciphertext. Also, you have to present the deciphered plaintext obtained from each ciphertext.

Grading

- In addition to the two ciphertexts, we will use four other ciphertexts using your code to check the generality of your solution.
- If your script is not in python, you need to be present in a schedule session to demonstrate the functionality of your code.
- If your script is in python, Gradescope has been configured to auto-grade your submission. To facilitate that, please follow the following steps.
  - Unzip "script.zip"
  - You will find "decipher_text.py" in the "script" folder. In this file, there are two sets of comments "Do not change this" and "Write your script here". Please follow them and add your code for deciphering the plaintext and the key given the ciphertext.
  - At any time, you can upload your submission on Gradescope to see if your code is passing the tests given there.
  - To help you understand the directory structure utilized in the Gradescope auto-grading tool, please look into the "example_autograder" folder.
    - This contains an auto-grading functionality similar to Gradescope. With this, you can check your "decrypt_text.py" code by placing it in the folder "example_autograder/source/submissions"". There are two important things to note.
      - You must run your code from the "example_autograder" folder.
      - The path to any file must be relative to the "example_autograder" folder.
    - Try executing "python3 ./source/example_test.py". You should be able to see the desired output on the terminal.