Cryptography [ECE5632]
Lab 1: Random-Key Mono-Alphabetic Classic Substitution Cipher

Ahmed H. Soliman
ahatem@msa.edu.eg

February 25, 2024

1 Introduction

Mono-alphabetic substitution ciphers encrypt the plaintext by swapping each letter or symbol in the plaintext by a different symbol as directed by the key[2]. This encryption can be broken using statistical methods such as frequency analysis, because the characters in every language appear with a particular probability. Figure 1 shows the frequency of letters in English language.

In this lab, we are going to implement a random-key mono-alphabetic classic substitution cipher. For

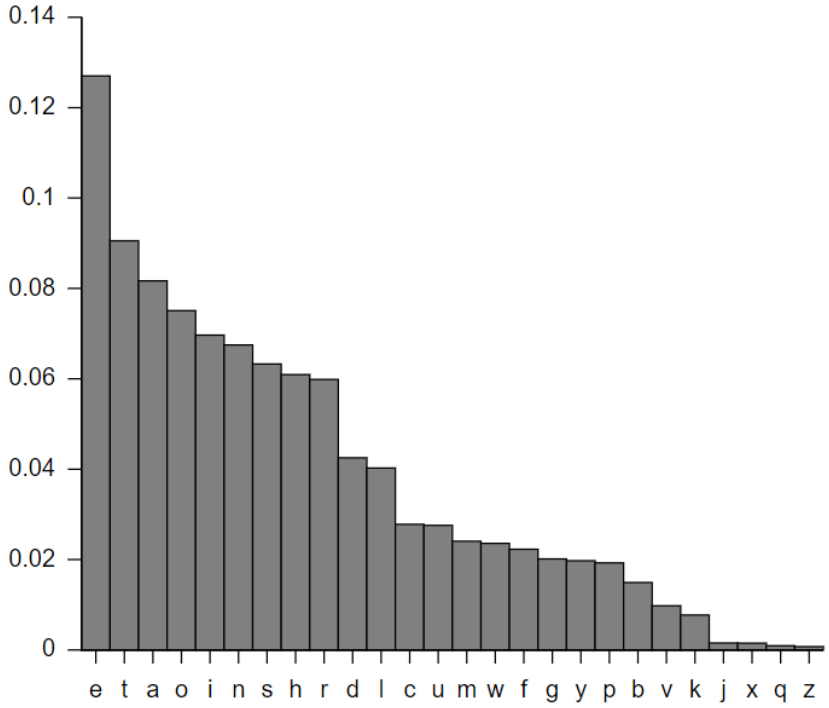

Figure 1: Characters frequency in English.[1]

simplicity, the cipher will keep spaces, commas, numbers, and special characters unchanged.

2 Lab Objectives

After completing this lab you should be able to:

- Implement a mono-alphabetic classic substitution cipher using Python.
- Analyze the frequency of english letters in ciphertext.

3 Lab Procedure

In the following sub-sections the precise procedure that should be done in order to achieve the aforementioned objectives.

3.1 Key Generation

```
import random
from string import ascii_lowercase


# Key generation
#alphabet = "abcdefghijklmnopqrstuvwxyz"
alphabet = ascii_lowercase
keyList = random.sample(alphabet, len(alphabet))

# This is needed for display only (not part of encryption)
key = ''.join(keyList)
print(alphabet)
print(key)

# Map plaintext letter  to ciphertext letter
keyMap = dict(zip(list(alphabet), keyList))
print(keyMap)
```

3.2 Read Plaintext File

The easiest and most safe method of opening a file for read/write in Python is using `with` statement. That's because it ensures closing the file after the read/write operation.

```
with open('plaintext.txt', 'r') as file:
    plaintext = file.read().lower()
```

3.3 Encryption

```
ciphertext = "".join(keyMap.get(letter, letter) for letter in plaintext)

# Write ciphertext to file
with open('ciphertext.txt', 'w') as file:
    file.write(ciphertext)
```

3.4 Letters Frequency Analysis

In this section, we are going to count the number of occurrences of each letter in the ciphertext. In order to perform a letter-frequency attack you must refer to the frequency distribution in Fig. 1.

```
lettersFreq = {}
for c in ascii_lowercase:
    lettersFreq[c] = ciphertext.count(c)
e
lettersFreqSorted = dict(
    sorted(lettersFreq.items(), key=lambda item: item[1], reverse=True)
)
print(lettersFreqSorted)
```

4 Self-Practice

- Write Python script that decrypts the `ciphertext.txt` file that you have encrypted before using the same key.
- Perform letters frequency analysis attack to a given ciphertext only (you have no idea about the key).

References

[1] File: English letter frequency (frequency).svg - wikimedia commons, 04 2010.

[2] Hassan, N. A., and Hijazi, R. Chapter 1 - introduction and historical background. In *Data Hiding Techniques in Windows OS*, N. A. Hassan and R. Hijazi, Eds. Syngress, Boston, 2017, pp. 1–22.