Report on One-Time Pad Cryptography

By Mitra Sasanka Darbha, M#13523468

1. Introduction

Cryptography is a method of protecting information and communications through use of codes so that only intended users are able to process and read it.

One-Time Pad is an encryption technique that cannot be cracked but requires a use of a one-time pre-shared key the same size as, or longer than, the message being sent. In this technique, the plaintext is paired with a random secret key (also referred to as a one-time pad). Then each bit or character of plaintext is encrypted by combining it with the corresponding bit or character form the pad using modular addition. The resulting ciphertext will be impossible to break if the key is:

1. Truly Random
2. At least as long as the plaintext
3. Never reused in whole or in part
4. Kept completely secret.

In this project, XOR is used to encrypt and decrypt data as follows:

$$\text{Encryption: } c \leftarrow m \oplus sk$$

$$\text{Decryption: } m \leftarrow c \oplus sk$$

Where ***m*** is message, ***c*** is cipher text and ***sk*** is the secret key.

2. Environment

The following system configuration is used in this project.

Processor	AMD A8 64-bit 2.20GHz
Operating System	Microsoft© Windows® 10 Home Single Language
Language	Python 3.7.4 (64-bit) with IDLE.

3. Execution and Results

Name of the project folder is **otp_m13523468.** It contains two subfolders – **src** and **data –** and the report.pdf file. The sources files reside under **src** folder and the files required and/or generated by the programs reside under **data** folder. This folder also contains the screenshots of the program execution.

Since Python Language is used for programming, there is no **build** folder as there will be no .cpp or .h or .class files.

Details of the individual files in the folder:

File Name	*Description*	*Path*
encrypt.py	To perform OTP encryption	otp_m13523468\src\
encrypt_128.py	To perform OTP encryption for 128 bits input	
decrypt.py	To perform OTP decryption	
keygen.py	To randomly generate keys of specified length	
distro.py	To calculate frequency of key distribution	
plaintext.txt	Contains plain text that needs to be encrypted	otp_m13523468\data\
key.txt	Contains secret key for cryptography	
ciphertext.txt	Contains the cipher text after encryption	
result.txt	Contains the plaintext after decryption	
newkey.txt	Contains the key generated by keygen.py	

Encryption:

Command Prompt is used for the execution of python files.

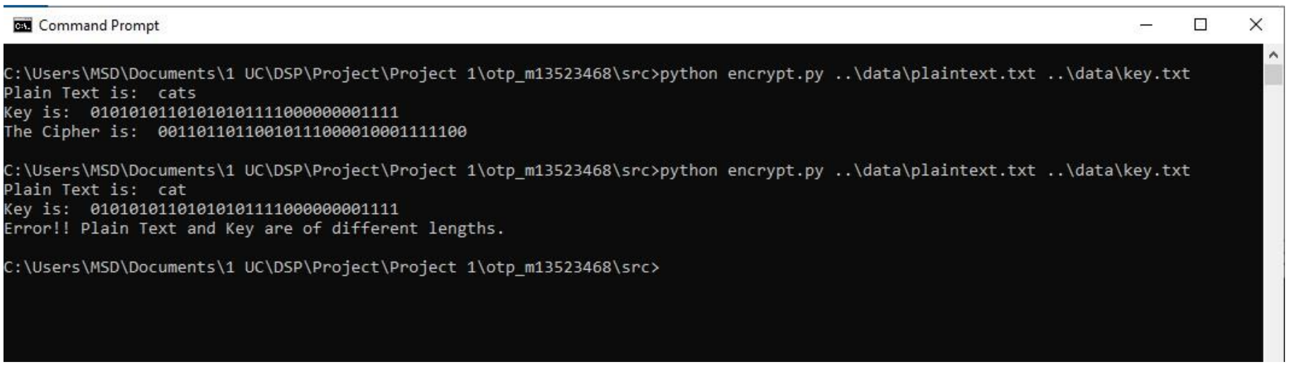
In Command Prompt, traverse to the **src** folder in **otp_m13523468.** In my system it is C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>

The command to perform encryption is:

C: …\otp_m13523468\src > python encrypt.py ..\data\plaintext.txt ..\data\key.txt

This displays the text in plaintext.txt, displays the key in binary format and also performs OTP encryption and displays the cipher text on terminal. The cipher output is also written to ciphertext.txt. A new file is created if ciphertext.txt does not exist.

If lengths of binary plaintext and binary key are different then an **Error** is displayed on screen.



```
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python encrypt.py ..\data\plaintext.txt ..\data\key.txt
Plain Text is:  cats
Key is:  01010101101010101111000000001111
The Cipher is:  00110110110010111000010001111100

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python encrypt.py ..\data\plaintext.txt ..\data\key.txt
Plain Text is:  cat
Key is:  01010101101010101111000000001111
Error!! Plain Text and Key are of different lengths.

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>
```

Decryption:

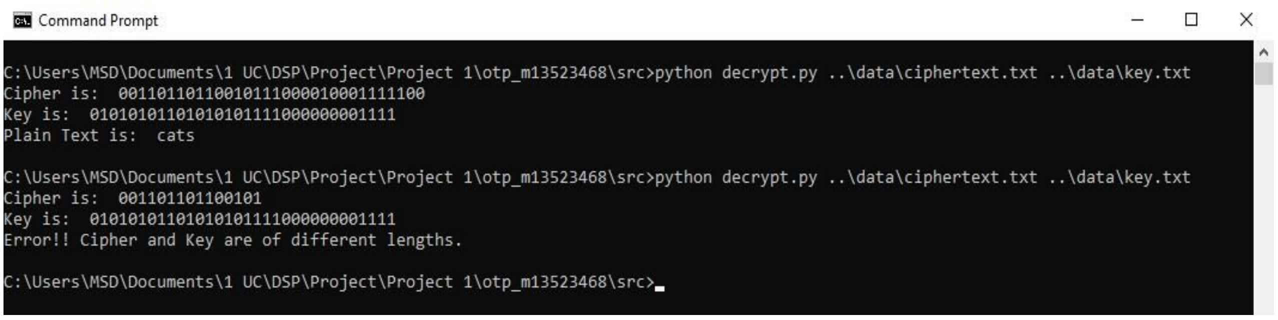
This is a similar to execution of encryption.py

The command for Decryption is

C: …\otp_m13523468\src> python decrypt.py ..\data\ciphertext.txt ..\data\key.txt

This displays the text present in ciphertext.txt and also the key. It performs decryption using OTP and the result is stored in result.txt and also displayed on the terminal.

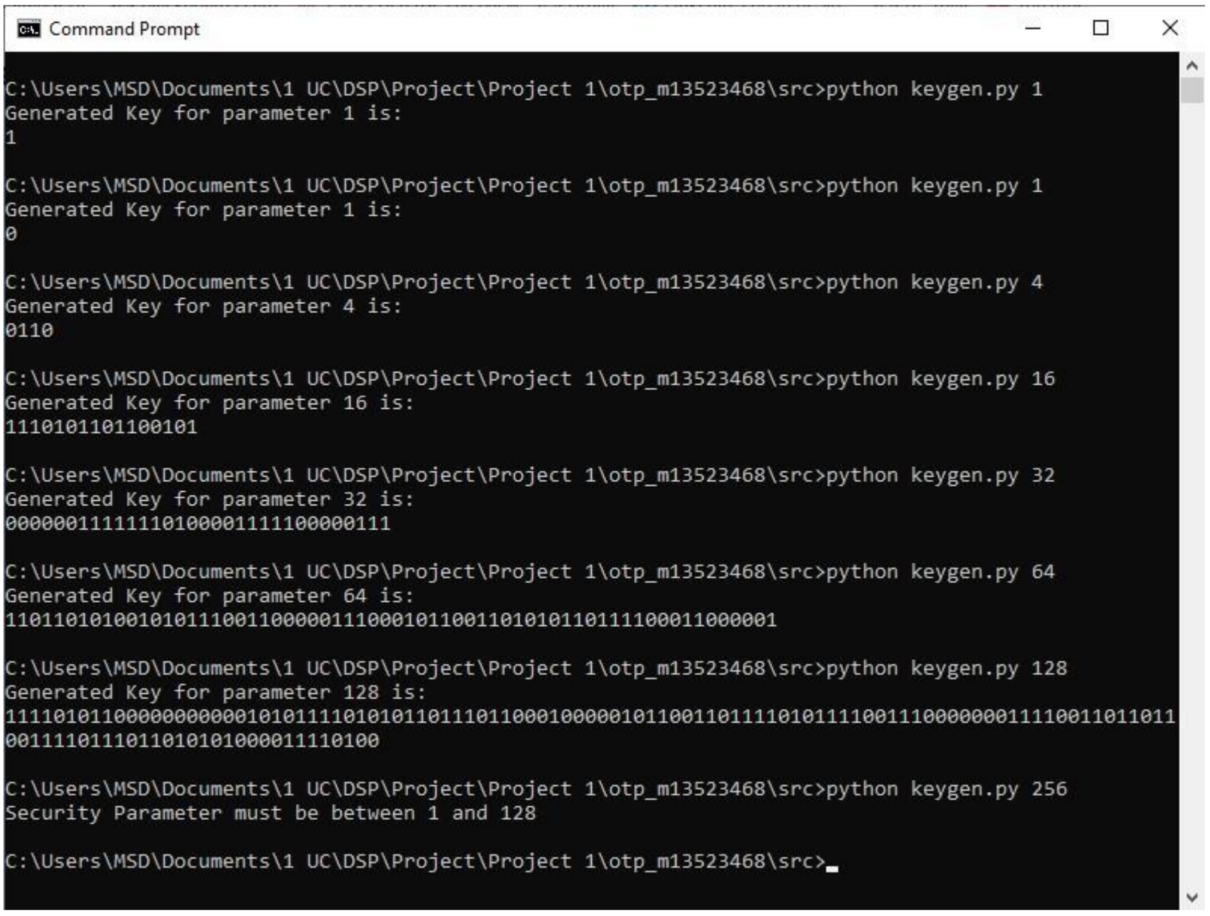
If the lengths of ciphertext and key do not match, then an **Error** is displayed on the terminal.


```
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python decrypt.py ..\data\ciphertext.txt ..\data\key.txt
Cipher is:  00110110110010111000010001111100
Key is:  01010101101010101111000000001111
Plain Text is:  cats

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python decrypt.py ..\data\ciphertext.txt ..\data\key.txt
Cipher is:  001101101100101
Key is:  01010101101010101111000000001111
Error!! Cipher and Key are of different lengths.

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>
```

Key Generation Function:

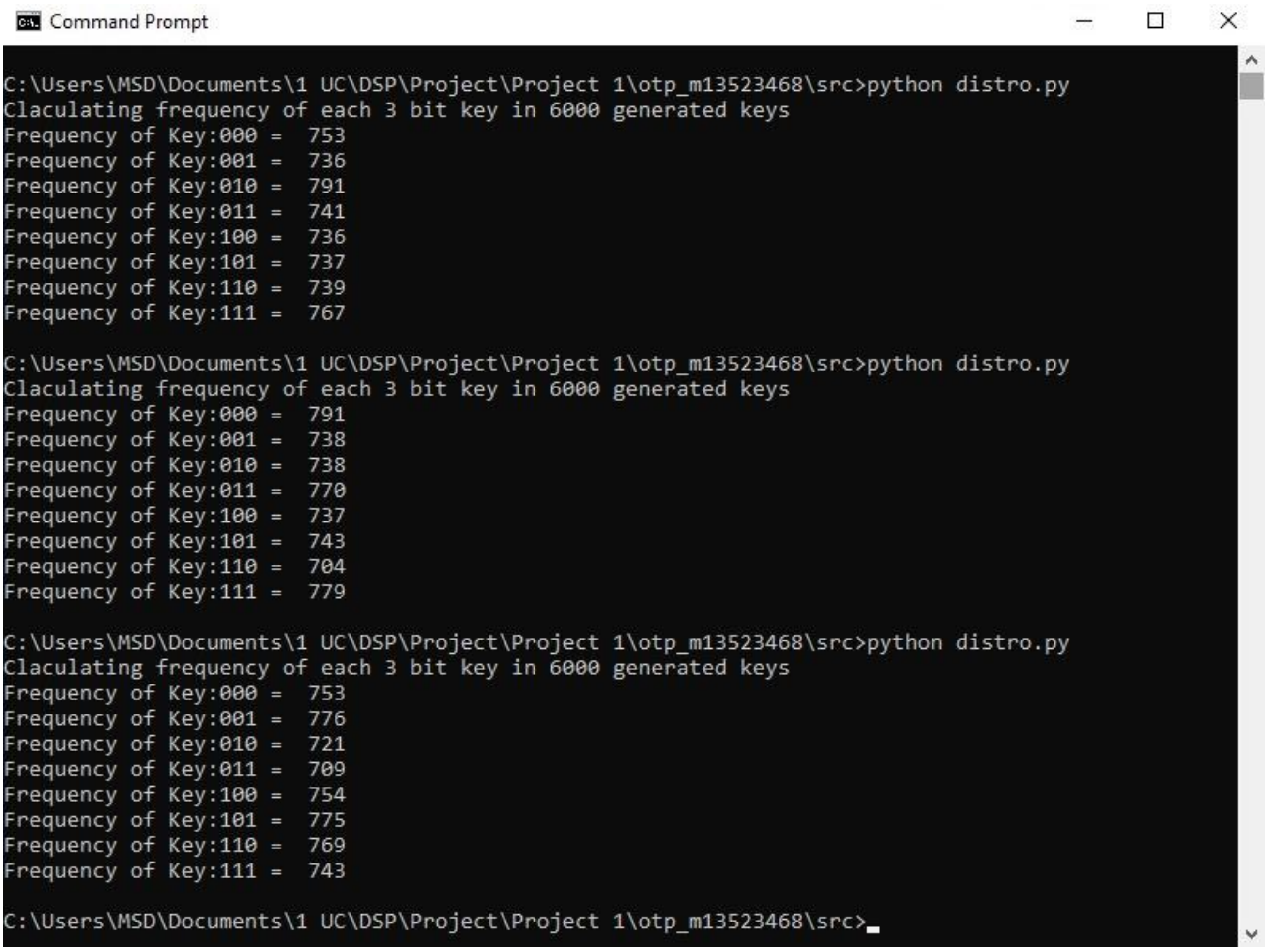
The keygen.py program randomly generates a secret key with λ, where λ is an integer given as a command-line argument. The newly generated random key is displayed in terminal and also stored in the file newkey.txt

If λ is not in the range of 1 to 128, then an error message is displayed.


```
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python keygen.py 1
Generated Key for parameter 1 is:
1

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python keygen.py 1
Generated Key for parameter 1 is:
0

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python keygen.py 4
Generated Key for parameter 4 is:
0110

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python keygen.py 16
Generated Key for parameter 16 is:
1110101101100101

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python keygen.py 32
Generated Key for parameter 32 is:
00000011111110100001111100000111

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python keygen.py 64
Generated Key for parameter 64 is:
1101101010010101110011000001110001011001101010110111100011000001

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python keygen.py 128
Generated Key for parameter 128 is:
11110101100000000000101011110101011011101100010000010110011011110101111001110000000111100110110110
00111101110110101010000111110100

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python keygen.py 256
Security Parameter must be between 1 and 128

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>
```

Distribution of Keys:

Security parameter, λ is 3 and 6000 keys are generated. Frequency of all uniquely generated 3-bit keys is calculated. The result is displayed on the terminal.



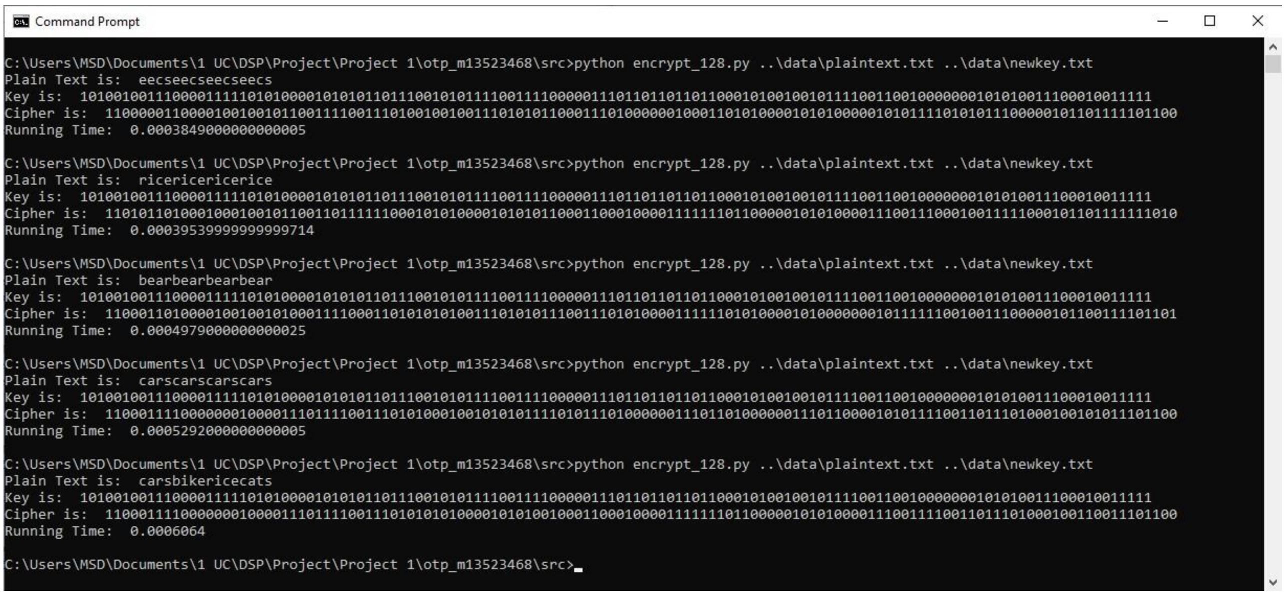
```
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python distro.py
Claculating frequency of each 3 bit key in 6000 generated keys
Frequency of Key:000 =  753
Frequency of Key:001 =  736
Frequency of Key:010 =  791
Frequency of Key:011 =  741
Frequency of Key:100 =  736
Frequency of Key:101 =  737
Frequency of Key:110 =  739
Frequency of Key:111 =  767

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python distro.py
Claculating frequency of each 3 bit key in 6000 generated keys
Frequency of Key:000 =  791
Frequency of Key:001 =  738
Frequency of Key:010 =  738
Frequency of Key:011 =  770
Frequency of Key:100 =  737
Frequency of Key:101 =  743
Frequency of Key:110 =  704
Frequency of Key:111 =  779

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python distro.py
Claculating frequency of each 3 bit key in 6000 generated keys
Frequency of Key:000 =  753
Frequency of Key:001 =  776
Frequency of Key:010 =  721
Frequency of Key:011 =  709
Frequency of Key:100 =  754
Frequency of Key:101 =  775
Frequency of Key:110 =  769
Frequency of Key:111 =  743

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>
```

We know that 8 unique 3-bit keys are possible: 000, 001, 010, 011, 100, 101, 110 and 111.

The program is run for 3 times and it can be observed that each key occurs for more than 700 times and the maximum possible frequency is less than 800. We can say that these keys are (almost) uniformly distributed with a mean value of 750. The standard deviation is approximately 20.

Average Running Time:

The Encryption function is executed with 128 bits of key and 16 characters of plaintext. The running time is calculated for 5 samples. The average running time is obtained as 0.0004 secs.



```
C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python encrypt_128.py ..\data\plaintext.txt ..\data\newkey.txt
Plain Text is:  eecseecseecseecs
Key is:  10100100111000011111010100001010101101110010101111001111000001110110110110110001010010010111100110010000000101010011100010011111
Cipher is:  11000001100001001001011001111001110100100100111010101100011101000000100011010100001010100000101011110101011100000101101111101100
Running Time:  0.0003849000000000005

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python encrypt_128.py ..\data\plaintext.txt ..\data\newkey.txt
Plain Text is:  ricericericerice
Key is:  10100100111000011111010100001010101101110010101111001111000001110110110110110001010010010111100110010000000101010011100010011111
Cipher is:  11010110100010001001011001101111110001010100001010101100011000100001111111011000001010100001110011100010011111000101101111111010
Running Time:  0.00039539999999999714

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python encrypt_128.py ..\data\plaintext.txt ..\data\newkey.txt
Plain Text is:  bearbearbearbear
Key is:  10100100111000011111010100001010101101110010101111001111000001110110110110110001010010010111100110010000000101010011100010011111
Cipher is:  11000110100001001001010001111000110101010100111010101110011101010000111111010100001010000000101111110010011100000101100111101101
Running Time:  0.0004979000000000025

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python encrypt_128.py ..\data\plaintext.txt ..\data\newkey.txt
Plain Text is:  carscarscarscars
Key is:  10100100111000011111010100001010101101110010101111001111000001110110110110110001010010010111100110010000000101010011100010011111
Cipher is:  110001111000000001000011101111001110101000100101010111101011101000000111011010000001110110000101011110011011101000100101011101100
Running Time:  0.0005292000000000005

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>python encrypt_128.py ..\data\plaintext.txt ..\data\newkey.txt
Plain Text is:  carsbikericecats
Key is:  10100100111000011111010100001010101101110010101111001111000001110110110110110001010010010111100110010000000101010011100010011111
Cipher is:  110001111000000001000011101111001110101010100001010100100011000100001111111011000001010100001110011110011011101000100110011101100
Running Time:  0.0006064

C:\Users\MSD\Documents\1 UC\DSP\Project\Project 1\otp_m13523468\src>
```

4. References

1. https://docs.python.org/3/library/functions.html#ord
2. https://docs.python.org/3/library/functions.html#chr
3. https://docs.python.org/3/library/functions.html#zip
4. https://www.pythonforbeginners.com/cheatsheet/python-file-handling
5. https://docs.python.org/3/library/sys.html
6. https://docs.python.org/3.7/library/timeit.html
7. https://en.wikipedia.org/wiki/One-time_pad