Programming Project 1 Exhaustive Key Search CSCE 5050

Group 9

Members:

Govind Naidu Pulakhandam - 11479985

Nitin Varma Siruvuri - 11513261

Required Libraries

```
!pip3 install pycryptodomex
!pip install pycryptodome
!pip install tqdm
'''We are installing the pycryptodome libraries and their dependencies

to work with the AES cipher as per the question requirements'''
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
Requirement already satisfied: pycryptodomex in /usr/local/lib/python3.8/dist-pack
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
Requirement already satisfied: pycryptodome in /usr/local/lib/python3.8/dist-packa
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packages (4.6
'We are installing the pycryptodome libraries and their dependencies\n\nto work w
ith the AES cipher as per the question requirements'
```

Defined functions to use in the program

```
'''Defining the function to read the contents of the files in the binary format.
This function is used to read the data fromt the .bin files
mainly from c1.bin,c2.bin,c3.bin,nonce1.bin,nonce2.bin,nonce3.bin

'''

def read_binary_files(binary_file_name):
    binary_fp = open(binary_file_name, "rb")
    '''The argument "rb" is used to read the binary data from the files or to read the .bin file in general '''
    binary_file_content = binary_fp.read()
    binary_fp.close()
    '''reading the content of the binary file using the binary file pointer and also closing the opened file and returin

    return binary_file_content


#Using the code fromt he utils_demo.py file given

def write_bytes(fn, value):
    f = open(fn, "wb")
    f.write(value)
    f.close()

def write_file(fn, value):
    f = open(fn, "w")
    f.write(value)
    f.close()
```

```
!unzip /content/sample_data/challenge.zip

#To unzip the contents of the files that are given to perform the project task

     Archive:  /content/sample_data/challenge.zip
     replace c_c.bin? [y]es, [n]o, [A]ll, [N]one, [r]ename:


from Crypto.Cipher import AES
from tqdm import tqdm
import binascii #Library to use to convert the file contents or values from binary to asscii and vice cersa



# Load plaintext and ciphertext from files
#To perfrom the activity we first need to read the data from the cipher files and the corresponding
#nonce and plain text files.

#Reading the data from the cipher text files with .bin extension.
#This cipher data is stored in the variables cipher_texgt_c1, cipher_text_c2 and cipher_text_c3
cipher_text_c1 = read_binary_files("c1.bin")#read_binary_files is a custom method wriiten to read the files in binary fo
cipher_text_c2 = read_binary_files("c2.bin")
cipher_text_c3 = read_binary_files("c3.bin")

#Now since the cipher text is read in binary format, lets try to read the text files in the same format
#These messages from each file are stored in the variables message_text_m1,message_text_m2,message_text_m3.
message_text_m1 = read_binary_files("m1.txt")
message_text_m2 = read_binary_files("m2.txt")
message_text_m3 = read_binary_files("m3.txt")

#Now lets try to load the nonce content in to the desired variables in the binary format
# Load nonce for CTR mode from files
nonce1 = read_binary_files("nonce1.bin")
nonce2 = read_binary_files("nonce2.bin")
nonce3 = read_binary_files("nonce3.bin")




#Now, defining the key
#in the questionn it is given that the key format follows first 13 bytes be fixed
#Meaning first digit is 1 in binary and next 12 zeros
#Total key length is of 16 bytes meaning only last 3 bytes are changing
#Key pattern is already given as x80 for initial value and x00 for next 12
# Based on that we are giving the keylength as 2 and Key space as 2**(key length*8) also equal to 2^24
key_length = 3
key_space = 2**(key_length * 8) # 2^24
print("\nEXHAUSTIVE KEY SEARCH RESULTS\n\n")
# Writing a for loop to iterate over to check for the possibilities of all the keys

for k in tqdm(range(key_space), desc="Searching for key ..."):
    # Giving the custom intial key format since the initial staring 13 bytes are known, that is given already
    key = b"\x80\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00" + k.to_bytes(key_length, byteorder="big")

    #Now checking each cipher text fromm plain text
    #In the straight forward approach mentioned in the lecture notes 4-2, given
    # encrypt the messages witht the keys to generate the cipher text and match for the same.
    #We can also verify by decryption also lets try first with encryption.

    #The c1, c2, c3 are the ciphes that are created to encrypt or decrypt the message / cipher texts
    c1 = AES.new(key, AES.MODE_CTR, nonce=nonce1)
    c2 = AES.new(key, AES.MODE_CTR, nonce=nonce2)
    c3 = AES.new(key, AES.MODE_CTR, nonce=nonce3)

    #Encrypting the message texts with the current generated secret key
    encrypt_1 = c1.encrypt(message_text_m1)
    encrypt_2 = c2.encrypt(message_text_m2)
    encrypt_3 = c3.encrypt(message_text_m3)
    if cipher_text_c3 == encrypt_3 and cipher_text_c2 == encrypt_2 and cipher_text_c1 == encrypt_1:
        #write_key = binascii.hexlify(key) #storing the key in hex fomat to a file to write to text file
        print("\n\nThe key used to encrypt 3 message files is:: ", binascii.hexlify(key))
        write_bytes("key.bin", key)#writing key in binary format
        write_file("key_in_text.txt",key.hex())#writing key to different file in hex format for backup
        print("\n The key is written to key.bin file\n")
        break
```

```
EXHAUSTIVE KEY SEARCH RESULTS


Searching for key ...:  37%|███▋      | 6286948/16777216 [07:47<12:59, 13452.09it/s]

The key used to encrypt 3 message files is::  b'800000000000000000000000005fee64'

 The key is written to key.bin file
```

Program to read the key and then decrypt the challenge cipher

```
from Cryptodome.Cipher import AES

#reading the challenge cipher and challenge nonce

challenge_cipher_text = read_binary_files("c_c.bin")#Reading the challenge cipher
challenge_nonce = read_binary_files("nonce_c.bin")#Reading chalenge nonce
key = read_binary_files("key.bin")#readig the challenge key from the file key.bin
#The read_binary_files method is used to read files in binary format
cc = AES.new(key, AES.MODE_CTR, nonce=challenge_nonce)#Creating ciper to decrypt


decrypt = cc.decrypt(challenge_cipher_text)#decrypting the cipher text using cipher
#and storing the value in decrypt variable

print("\nThe decrypted challenge cipher is: " ,decrypt)#value of decrypt is displayed
```

```
The decrypted challenge cipher is:  b'UNT is a community of dreamers and doers.'
```

✓ 0s completed at 9:48 PM