ECE 404 Homework 1

Due: Thursday 01/19/2023 at 5:59PM

In this exercise, you will assume the role of a cryptanalyst and attempt to break a cryptographic system composed of the two Python scripts `EncryptForFun.py` and `DecryptForFun.py` described in Section 2.11 in Lecture 2. As you will recall, the script `EncryptForFun.py` can be used for encrypting a message file while the script `DecryptForFun.py` recovers that message from the ciphertext produced with the previous script. Both these scripts can be found on the ECE 404 webpage for the Lecture Notes (click on the "download code" tab for Lecture 2).

Problem

With the parameter BLOCKSIZE set to 16, the script `EncryptForFun.py` produces the following ciphertext for a plaintext message regarding the best F1 driver to walk planet Earth.

47973a7f79d201737d9b31324c96631a0da46b0817af764731ee534f30f350557fb512126fb4404b6fc9
016f7bc60677298e1f7e259f0a3e76d6183e558433385583326b4fc2236342cc6d6010c1696c148478
3905836f25148162690ece6c7606ca606c08846e2241ac211643ab20020f856e284dc0267a40c42662
4e8a272d5ac5186865c30f6f65d4475d05d22a4a40f63d515cf8201069b714126eb6005e40f82a1071
bd1f183ef0785657d8376255df367619f1785c5bb41c1d7ebc1d007da652063cbd130273a45c0d73af
130e72b250083aa35a1931f14d5c34f155190dbe701a1bf2504836f65c4c20a30f2868810a396c9e0d
256a84156c6ecd15697fd250322b93413f6e80083c6fd42d7541d52f784c90103b7786152f798f1335
35d11a787bd4547434d05f6b70d7177a38c0526b32d1596a76c4167e39c25e6f71ca11716a99123873
835c63629b5e7a7d92023561984e202e9b48272e804f3b60db5e2362c2412a2f8c44643bc350622ac7
056963cf057462d91e3139c80f3129d5067c64db05742ad14477329f5223358e483569dd35615bdd3d
6646c72a6e49ca3f225385323e54923e2854847b2d07c57c65168d742a0896272d5d86312b4d90242d
47c52c7f4ac1207b0fda6e3227955a3020944e7c0eda643203937f2f1f956973

all in one line. You can assume that the passphrase stays the same (that is, the passphrase is "Hopes and dreams of a million years").

Your job is to recover both the original quote and the encryption key by mounting a brute-force attack on the encryption/decryption algorithms.

HINT 1: The correctly decrypted message should contain the words *Sir Lewis*.
HINT 2: The logic used in the scripts assumes that the effective key size is 16 bits when the BLOCKSIZE variable is set to 16. So your brute-force attack needs to search through a keyspace of size 2^{16}.

Instructions

- To accomplish this, you need to implement the following function:

```
def cryptBreak(ciphertextFile, key_bv):
    # Arguments:
    # * ciphertextFile:   String containing file name of the ciphertext
    # * key_bv:           16-bit BitVector for the decryption key
    #
    # Function Description:
    #   Attempts to decrypt the ciphertext within ciphertextFile file using
        key_bv and returns the original plaintext as a string
```

- The function must be implemented and saved in a file named `cryptBreak.py`.
- This function must be implemented to decrypt the message *for a single key* and not to perform complete brute force analysis - the brute force analysis must be done within the code's __main__ function/statement or in a separate Python file by importing `cryptBreak.py` into that file.
- Note that the string returned by the above function may or not may not be the correct plaintext since the correct `key_bv` is unknown. Therefore to determine the correct value for `key_bv`, you will need to brute force all possible values for `key_bv` and check the returned string to find the right one.
- You need to submit only the `cryptBreak.py` file which will be auto-graded - hence make sure that the `cryptBreak.py` file does not run the entire brute force analysis or any other routine when imported.

Example of Usage

Below is an example of how your implemented function could be used - if your function is implemented correctly, the following code snippet should run without any errors:

```
import cryptBreak
from BitVector import *
someRandomInteger = 9999 #Arbitrary integer for creating a BitVector
key_bv = BitVector(intVal=someRandomInteger, size=16)
decryptedMessage = cryptBreak.cryptBreak('encrypted.txt', key_bv)
if 'Sir Lewis' is in decryptedMessage:
        print('Encryption Broken!')
else:
        print('Not decrypted yet')
```

To submit your work, please read the following two sections for instructions.
Failure to follow these instructions may result in loss of points!.

Submission Instructions

- For this assignment, you will electronically submit your work (see the Electronic Turn-In section below for more info).
- In your program file, include a header as described on the ECE 404 Homework Page (`https://engineering.purdue.edu/ece404/homework.htm`)
- As mentioned previously, put the code for your brute force analysis in an `if __name__ == "__main__"` statement (assuming you are using Python) so your test code won't be executed when `cryptBreak` function is imported.
- **Note:** This homework assignment is not the same as the one at the end of the Lecture Note 2. Please do not solve and turn in the homework assignments from the Lecture Notes - they will not be accepted.

Electronic Turn-In

- You must turn in a single zip file on Brightspace with the following naming convention: `HW01_<last_name>_<first_name>.zip` . Do not turn in files other than those listed below. Your submission must include:
  - The file containing your `cryptBreak` implementation named `cryptBreak.py`
  - A pdf named `HW01_<last_name>_<first_name>.pdf` containing:
    - The recovered plaintext quote
    - The recovered encryption key
    - A brief explanation of your code