ECE 404 Homework #1

Due: Thursday 1/23/2020 at 4:29PM

This is an exercise in you assuming the role of a cryptanalyst and trying to break a cryptographic system that consists of the two Python scripts you saw in Section 2.11 in Lecture 2. As you'll recall, the script `EncryptForFun.py` can be used for encrypting a message file and the script `DecryptForFun.py` for recovering the plaintext message from the ciphertext created by the previous script. Both scripts can be found on the ECE 404 Lecture Notes web page (click "download code" for Lecture 2).

With BLOCKSIZE set to 16, the script EncryptForFun.py produces the following ciphertext output for a plaintext message that is a quote from Mark Twain:

```
3030722b63796c722e62297c226b26393023332571397622a66347d667e7a65286e347a2
37625713333330242d323e24252939676b7575767e7168336d386b323933243f30342633
3a322d7c7f6d796e6568642a776862657f632d6237612b603c223a2f68297f2868236c2
4304b4824306b6244715468164e035d084167390841
```

all in one line.

You can see the same encrypted output in the file named `encrypted.txt` on the course website at https://engineering.purdue.edu/ece404/homework.htm The correctly decrypted message should contain "Mark Twain" (without quotation marks).

Your job is to recover both the original quote and the encryption key used by mounting a brute-force attack on the encryption/decryption algorithms. (HINT: The logic used in the scripts implies 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 only 2^{16}.)

To accomplish this, you need to implement the following function in a file named **cryptBreak.py**:

```
#Arguments:
# ciphertextFile: String containing file name of the ciphertext (e.g. encrypted.txt )
# key_bv: 16-bit BitVector of the key used to try to decrypt the ciphertext.
#Function Description:
# Attempts to decrypt ciphertext contained in ciphertextFile using key_bv and returns
    the original plaintext as a string
def cryptBreak(ciphertextFile,key_bv):
```

Keep in mind that what is returned by the above function may or not may not be the correct plaintext since you don't know the correct `key_bv`. Therefore you will have to brute force all possible `key_bv` values and check the returned string to find the right one.

Example of Usage

Below is an example of how the function you implement for this assignment could be used:

```
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 ’Mark Twain’ 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 code and a PDF as your "hard-copy" submission.
- The "hard-copy" PDF must include your code, the recovered plaintext quote, the encryption key you found through brute force methods, and a brief explanation of your code.
- In your program file, include a header as described on the ECE 404 Homework Page.
- If you use any code to test your `cryptBreak` function, put it in an `if __name__ == ’__main__’` statement so your test code won't be executed if we import your `cryptBreak` function.
- If for whatever reason BitVector is not available for Python 3 on the ECN machines, you can use the following command to install it on your ECN account:
  `pip3 install --user BitVector`

Electronic Turn-In

- We will be using the turnin command on your shay machines (shay.ecn.purdue.edu) to submit the homework electronically. You can use ThinLinc, Putty (on Windows) or the ssh command (on Linux) to access your shay machine remotely.
- To actually get your homework files onto your shay machine, you can use the WinSCP program (for Windows) or the scp command (for Linux).
- Once you have your files on your shay machine, the command you use to submit them should look like one of the following (without the dash at the beginning):

  `turnin -c ece404 -p hw01 hw01.pdf cryptBreak.py` (if using Python)
  `turnin -c ece404 -p hw01 hw01.pdf cryptBreak.pl` (if using Perl)