ECE 404 Homework 1

Due: Thursday 01/20/2022 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 that is a quote from Douglas Adams:

3c2b223a71277173636930742f6c296b33702e2a7d127b086b146c09721821083d092c112
645265e7b202574126f147c0b690b3d392d2b342b40

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 *Douglas Adams*.
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:

```
1 import cryptBreak
2 from BitVector import *
3 someRandomInteger = 9999 #Arbitrary integer for creating a BitVector
4 key_bv = BitVector(intVal=someRandomInteger, size=16)
5 decryptedMessage = cryptBreak.cryptBreak('encrypted.txt', key_bv)
6 if 'Douglas Adams' is in decryptedMessage:
7         print('Encryption Broken!')
8 else:
9         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.
- If using Python, please denote the Python version in your code with a shebang line (e.g. #!/usr/bin/env python2)

- 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`
- **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 two files electronically. Do not turn in files other than those listed below. Your submission must include:
  - The file containing your `cryptBreak` implementation
  - A pdf containing:
    * The recovered plaintext quote
    * The recovered encryption key
    * A brief explanation of your code
- 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)