ECE 404 Homework #5

Due: Tuesday 02/25/2020 at 4:29 PM

In this homework, you will become familiar with the ANSI X9.31 pseudo-random number generator (PRNG) and the Counter (CTR) mode for using block ciphers.

Both parts will require the use of your AES implementation from homework 4. If for some reason you did not finish homework 4 and are unable to finish your AES implementation by the deadline for homework 5, you may use the implementation found in the PyCryptoDome/PyCrypto package. **If you import this package in your code, you will incur a 40% penalty to your homework score.**

Part 1: X9.31 Pseudo-Random Number Generator

Section 10.6 of Lecture 10 talks about the ANSI X9.31 cryptographically secure PRNGs. Your task is to implement a more modern version of this PRNG with the following requirements:

1. Instead of using 3DES for encrypting the 64-bit vectors as indicated in the lecture notes, use your implementation of AES from homework 4 to encrypt 128-bit vectors. By the way, AES is used instead of 3DES in the newer implementations of X9.31.

2. Your script needs to define the PRNG function in the following manner:

```
#Arguments:
# v0: 128-bit BitVector object containing the seed value
# dt: 128-bit BitVector object symbolizing the date and time
# key_file: String of file name containing the encryption key (in ASCII) for AES
# totalNum: integer indicating the total number of random numbers to generate
#Function Description
# Uses the arguments with the X9.31 algorithm to generate totalNum random
    numbers as BitVector objects
#Returns a list of BitVector objects, with each BitVector object representing a
    random number generated from X9.31
def x931(v0, dt, totalNum, key_file):
```

3. For simplicity's sake in this homework, you can use the same `dt` vector for each random number generated in a given call.

4. Obviously, `dt` is supposed to contain the date and time. But for testing purposes, you can set it to a predetermined value (hence, `dt` is a specifiable argument to the `x931(...)`. function).

How Your X9.31 Code Will Be Tested

Your `x931` function will be tested with a script similar to the one below:

```
import x931
from BitVector import *
v0 = BitVector(textstring='computersecurity') #v0 will be 128 bits
#As mentioned before, for testing purposes dt is set to a predetermined value
dt = BitVector(intVal=99, size=128)
listX931 = x931.x931(v0,dt,3,'keyX931.txt')
#Check if list is correct
print('{}\n{}\n{}'.format(int(listX931[0]),int(listX931[1]),int(listX931[2]))
```

The correct result for the above script (as well as the `keyX931.txt` file) can be found in the Homework Section of the ECE 404 website.

Part 2: AES Encryption in Counter Mode

In Homework 2, the sudden changes in the image of the helicopter allowed you to see the helicopter's outline even after encrypting the image. To prevent this from happening, implement AES in CTR mode as described in section 9.5.5 of the lecture notes. Use your implementation of AES from homework 4 as a starting point. Use a random number generated from your X9.31 implementation in Part 1 of this homework for the CTR mode initialization vector.

1. The encryption function should have the following format:

```
#Arguments:
# iv: 128-bit initialization vector
# image_file: input .ppm image file name
# out_file: encrypted .ppm image file name
# key_file: String of file name containing encryption key (in ASCII)
#Function Descrption:
# Encrypts image_file using CTR mode AES and writes said file to out_file. No
    required return value.
def ctr_aes_image(iv,image_file='image.ppm',out_file='enc_image.ppm',
    key_file='key.txt'):
```

2. For those who are unaware: the above syntax involving the equals sign in the arguments of the function definition indicates **default arguments** for a Python function definition. For example, when calling `ctr_aes_image(...)`, if no value is specified for the `image_file` argument, the value `'image.ppm'` is used by default (similar logic applies to `out_file` and `key_file`).

3. To ensure that the encryption does not take too long, write each block to the output image file as you encrypt it. Do not store the entire encrypted image in a BitVector as you encrypt it (this will cause a slowdown due to the size of the image).

4. As in homework 2, the encrypted image should still be a viewable image file and as such should have an image header (though due to the usage of CTR mode encryption the image should be indistinguishable from the original).

How Your CTR AES Code Will Be Tested

Your `ctr_aes_image` function will be tested with a script similar to the one below

```
from AES_image import ctr_aes_image
from BitVector import *
iv = BitVector(textstring='computersecurity') #iv will be 128 bits
ctr_aes_image(iv,'image.ppm','enc_image.ppm','keyCTR.txt')
```

Further testing can be done by comparing your encrypted image with the encrypted image found on the ECE 404 homework page (which is an encrypted version of the helicopter image from homework 2). You can use
`xxd enc _image.ppm | less`
to view the encrypted image data in hexadecimal and compare with what the encrypted image your code generates (you can also use the `diff` command to see if the two images are identical).

Submission Instructions

Please read below. **Failure to follow these instructions may result in loss of points!**.

- This assignment will **have an electronic (code) submission only**.
- In your program file, include a header as described on the ECE 404 Homework Page.
- If using Python, please denote the Python version in your code with a shebang line (e.g. #!/usr/bin/env python3)
- You must turn in two files electronically. Do not turn in files other than those listed below.
- As previously mentioned, you can use the AES implementation in PyCryptoDome or PyCrypto if you are somehow unable to finish implementing AES. However, if you import either package in your code, **you will incur a 40% penalty to your homework score**. As an example, if you would otherwise get 100% for this homework, but used the PyCryptoDome/PyCrypto implementation of AES, your final grade for this homework would instead be 60%.
  - For those who are curious: PyCrypto was first released around 2002, and for many years has been used in Python applications for cryptographic needs. However, PyCrypyto is no longer actively maintained by its creator, and vulnerabilities have been discovered in its code. PyCryptoDome is a "drop-in replacement" for PyCrypto that is actively maintained.
- Please include comments in your code.

Electronic Turn-in

`turnin -c ece404 -p hw05 x931.pl AES_image.pl` (if using Perl)
`turnin -c ece404 -p hw05 x931.py AES_image.py` (if using Python)