ECE404 Introduction to Computer Security: Homework 05

Spring 2024
Due Date: 5:59pm, February 20, 2024

1 Introduction

This homework assignment consists of two part, the first of which is in regards to using block ciphers in the counter (CTR) mode. The second is in regards to cryptographically-safe pseudo-random number generators. Both parts of the assignment will require the use of your AES implemenation from homework 05.

As always, please read the homework document in its entirety before coming to office hours with your questions. The teaching staff have spent a long time writing the assignment to cover many common questions you might have.

2 Problem 1: AES Encryption In Counter Mode

In homework 02, 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, encrypt the same helicopter image using AES in the counter mode as described in Section 9.5.5 of the lecture notes [2]. In a real-world implementation, you would normally choose a random 16 byte number, however for testing and reproducibility purposes, please use a BitVector containing the ASCII encoding of the textstring 'counter-mode-ctr'. as your initialization vector.

Please note that your solution must be an object oriented python program. The bullet points below should give you an idea of how your solution should be written as an extension of the AES class from homework 04.

- The method that performs AES encryption in the CTR mode has the following format. Note that this function should be an AES class method.

```python
def ctr_aes_image(self, iv, image_file, enc_image):
    """
    Inputs:
    iv (BitVector): 128-bit initialization vector
    image_file (str): input .ppm file name
    enc_image (str): output .ppm file name

    Method Description:
    * This method encrypts the contents in image_file
                                using CTR mode AES and
                                writes the encrypted
                                content to enc_image
    * Method returns void
    """
```

- 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 noticeable slowdown to the size of the image).
- As in homework 02, the encrypted image should still be a viewable image file and as such should have an image header
- The command line syntax to invoke your `ctr_aes_image` method should be the following:

```
python3 AES.py -i image.ppm key.txt enc_image.ppm
```

- Note here that the key used is the same as the one used in homework 04
- Modify your if name equals main guard to hand the above command line syntax. Example shown below:

```
if __name__ == "__main__":
    cipher = AES(keyfile=sys.argv[3])
    if sys.argv[1] == "-e":
        cipher.encrypt(plaintext=sys.argv[2], ciphertext=
                                sys.argv[4])
    elif sys.argv[1] == "-d":
        cipher.decrypt(ciphertext=sys.argv[2],
                                recovered_plaintext=sys.
                                argv[4])
    elif sys.argv[1] == "-i":
        cipher.ctr_aes_image(iv= BitVector(textstring="
                                counter-mode-ctr"),
                                image_file=sys.argv[2],
                                enc_image=sys.argv[4])
```

3 Problem 2: X9.31 CSPRNG

Lecture 10.6 introduces the ANSI X9.31 crytographically secure pseudo-random number generator. Your task is to implement a more modern version of this PRNG with the following requirements:

- Instead of using 3DES for encrypting the 64-bit vectors as indicated in the lecture notes, use your implemenation of AES from homework 04 to encrypt 128-bit vectors.
- Interestingly enough, newer modern versions of the X9.31 algorithm use AES instead of 3DES to generate their pseudo-random numbers [1].
- Similar to problem 1, your solution for problem 2 should also be in the form of an object oriented python program. (i.e. an extension of your AES class from homework 04).
- The method that generates your cryptographically secure pseudo-random numbers has the following format. Note that this function should be an AES class method.

```python
def x931(self, v0, dt, totalNum, outfile):
    """
    Inputs:
    v0 (BitVector): 128-bit seed value
    dt (BitVector): 128-bit date/time value
    totalNum (int): total number of pseudo-random numbers
                            to generate

    Method Description:
    * This method uses the arguments with the X9.31
                            algorithm to compute
                            totalNum number of pseudo-
                            random numbers, each
                            represented as BitVector
                            objects.
    * These numbers are then written to the output file in
                             base 10 notation.
    * Method returns void
    """
```

- For testing and reproducibility purposes, please use the IV from problem 1 for v_0 and the integer value 501 represented as a 128-bit BitVector for dt.

- The command line syntax to invoke your `x931` method should be the following:

```
python3 AES.py -r 3 key.txt random_numbers.txt
```

- The argument directly following the '-r' flag indicates the number of random numbers to generate. For this assignment please generate 5 cryptographically secure random numbers.

- Note here that the key used is the same as the one used in homework 04

- Modify your if name equals main guard to handle the above command line syntax. Example shown below:

```
if __name__ == "__main__":
    cipher = AES(keyfile=sys.argv[3])
    if sys.argv[1] == "-e":
        cipher.encrypt(plaintext=sys.argv[2], ciphertext=
                                    sys.argv[4])
    elif sys.argv[1] == "-d":
        cipher.decrypt(ciphertext=sys.argv[2],
                                    recovered_plaintext=sys.
                                    argv[4])
    elif sys.argv[1] == "-i":
        cipher.ctr_aes_image(iv= BitVector(textstring="
                                    counter-mode-ctr"),
                                    image_file=sys.argv[2],
                                    enc_image=sys.argv[4])
    else:
        cipher.x931(v0=BitVector(textstring="counter-mode-
                                    ctr"), dt=BitVector(intVal
                                    =501,size=128), totalNum=
                                    int(sys.argv[2]), outfile=
                                    sys.argv[4])
```

- Using the given dt and v_0 values, your program should generate the five following numbers:

```
331374527193731622526773163027689011175
26263303708022960927873924862754889187
6213881104399286406150948824157995508
317525806849049200816126045738729418009
240080400546264647934751409092776671804
```

4 Submission Instructions

Make sure that the program requirements and submission instructions are followed. Failure to follow these instructions may result in loss of points!

- For this homework you will be submitting a zip file to Brightspace titled `hw05_<last_name>_<first_name>.zip` containing:
  - The modified `AES.py` file containing your solutions for problems 1 and 2
  - A PDF titled `hw05_<last_name>_<first_name>.pdf` containing:
    - a brief explanation of your code for both problems
    - the encrypted image from problem 1
    - the 5 pseudo-random numbers generated from problem 2

References

[1] NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms. URL https://csrc.nist.rip/cryptval/rng/931rngext.pdf.

[2] ECE 404 Lecture Notes. URL https://engineering.purdue.edu/kak/compsec/Lectures.html.