Name(s): ____

Modern Cryptography Lab

CMSC 426/626 - Computer Security

You may work individually or in groups of 2 - 3.

In this lab, we analyze the encrypted communications of Gru and Dr. Nefario – with luck, we will decrypt some of their messages. The lab is focused on analysis rather than programming.

Hex Viewing on Linux

It is nearly impossible to analyze binary data without hex viewing tools. On Linux, there are two tools that I like to use: `hexdump` and the `emacs` editor. Before diving into the lab, we will walk through using these tools.

1. Login to `linux.gl`, a Linux VM, or Mac and download the lab files:

   a. Copy the lab files from my AFS directory:

   ```
   cp /afs/umbc.edu/users/c/m/cmarron/pub/msg1.enc .
   cp /afs/umbc.edu/users/c/m/cmarron/pub/msg2.enc .
   cp /afs/umbc.edu/users/c/m/cmarron/pub/msg3.enc .
   cp /afs/umbc.edu/users/c/m/cmarron/pub/metadata.txt .
   cp /afs/umbc.edu/users/c/m/cmarron/pub/genkey.py .
   ```

   On a Linux VM or personal Linux laptop, you should be able to use `wget` to download the lab files:

   ```
   wget http://www.csee.umbc.edu/~cmarron/pub/msg1.enc
   wget http://www.csee.umbc.edu/~cmarron/pub/msg2.enc
   wget http://www.csee.umbc.edu/~cmarron/pub/msg3.enc
   wget http://www.csee.umbc.edu/~cmarron/pub/metadata.txt
   wget http://www.csee.umbc.edu/~cmarron/pub/genkey.py
   ```

   On a Mac, use `curl` and redirect standard output to a file:

   ```
   curl http://www.csee.umbc.edu/~cmarron/pub/msg1.enc > msg1.enc
   curl http://www.csee.umbc.edu/~cmarron/pub/msg2.enc > msg2.enc
   curl http://www.csee.umbc.edu/~cmarron/pub/msg3.enc > msg3.enc
   curl http://www.csee.umbc.edu/~cmarron/pub/metadata.txt > metadata.txt
   curl http://www.csee.umbc.edu/~cmarron/pub/genkey.py > genkey.py
   ```

b. At the command prompt, enter the command:

```
hexdump -C msg1.enc
```

You should see something like the following:

```
00000000  b3 b5 c9 70 a8 b7 7e f9  3b d9 77 39 ca 80 06 f9  |...p..~.;.w9....|
00000010  3b d9 77 39 ca 80 06 f9  af 8b 5d c6 e0 5d c7 33  |;.w9......]..].3|
00000020  53 75 62 a1 84 43 c0 02  6e 0c e8 f8 a1 84 af 2e  |Sub..C..n.......|
00000030  9f fc bc c6 2d fc e4 78  fe 96 a3 ea 01 e9 8b fa  |....-..x........|
00000040  9a e6 36 4d f6 df 07 a5  3b d9 77 39 ca 80 06 f9  |..6M....;.w9....|
00000050  9d d5 2f 8b a1 f1 bb da  9d d5 2f 8b a1 f1 bb da  |../......./.....|
*
00000200  f3 24 5e a8 9d bc 33 1d  82 05 0a 03 08 26 54 ba  |.$^...3......&T.|
00000210  34 21 ec 4f 4f 18 25 d7  2e ba 12 3f 3e 2f a8 9b  |4!.OO.%....?>/..|
```

This is just the beginning of the output, showing the start of the file. The asterisk (*) indicates a section of repeated rows: there are a number of rows after the row labeled `0000050` with the same values as row `0000050`.

c. Try the command again, piping to `less`:

```
hexdump -C msg1.enc | less
```

Press `<space>` to scroll through the output, or “`q`” to quit.

d. From the Linux command line, enter the following:

```
emacs msg1.enc
```

Once emacs has started, press `Esc`, then `x` (written `Esc x` or `Meta-x`). This will place Emacs in command mode.

Type `hexl-mode` and press `Enter`. You should now have a hex view of the file. You can move around and edit the file using Emacs commands. If you don’t know or can’t remember Emacs commands, see the Emacs Quick Reference Card.

Problem 1: What type of encryption?

When Gru is "on the road," he and Dr. Nefario communicate by posting encrypted messages to a bulletin board. It is known that Gru's user name is BIRDMASTER. Dr. Nefario's user name is FALLENEAGLE, undoubtedly a reference to his prior affiliation with the U.S. Air Force. Our goal is to analyze the encrypted messages and, with luck, decrypt some of them.

a. Examine the file `msg1.enc` using a hex viewing tool. Based on material presented in the lectures, you should be able to make an educated guess as to what algorithm is being used to encrypt the data.

   Does the encryption algorithm appear to be a block cipher or a stream cipher? If a block cipher, which one, and which mode of operation? Justify your answers!

b. Continuing with your analysis of `msg1.enc`, try to determine the format of the underlying plaintext. Find a .doc file, a .docx file, a pdf file, and some image files; use a hex viewing tool to determine if any of these types of file have patterns matching what you observe in the ciphertext.

   What are some likely formats for the underlying plaintext? What is your best guess for the underlying format? Justify your answers!

Problem 2: Something has changed...

Later on the January 30, 2013, several additional encrypted messages were posted: FALLENEAGLE posted the file `msg2.enc` and BIRDMASTER posted `msg3.enc`. Soon after `msg3.enc` was posted, the unencrypted file `genkey.py` was posted by FALLENEAGLE, but quickly removed (we can only assume that Dr. Nefario had intended to encrypt the file, but realizing his error, removed the file soon after posting it).

a. Examine the file `genkey.py` using a text editor. The comment suggests that there has been a recent change to the encryption program. Analyze the key generation scheme.

   Describe how the encryption keys are being generated. You should recognize the method from lecture.

b. Focus on the seeding of the key generation algorithm. You will need to read about the Python `time` module and `time()` function to fully understand how the seeding works (see http://docs.python.org/2/library/time.html)

Describe in words how the seed is being computed. Is this a good method? Why or why not?

c. Consider *all* you know about the files `msg2.enc` and `msg3.enc`. How can you use this information, along with what you have learned from `genkey.py`, to attack the encrypted messages?

What crucial peace of information regarding the encrypted files will allow you to attack the key generation function? Outline an attack on the key generation function.

d. How could Dr. Nefario improve the encryption software?

Describe at least two ways that Dr. Nefario could improve the security of the encryption software. Your answers must improve the overall security of the system and, specifically, address the weakness that lead to the attack in (2.c).

Extra Credit

Implement the attack you outlined in (2.c) and recover the plaintext for `msg2.enc` and `msg3.enc`. Turn in the decrypted messages *and* the code you used to recover the plaintext.