# CSE467 Crypto2 Lab.

Do all this work on ceclnx01.

Create a directory called crypto2 and put all work in there.

Copy over all files from /groups/cse467/crypto2 to your directory crypto2

### **1) Decrypt the file napoleon.txt.enc**

aes-256-cbc, uses the algorithm pbkdf2 for creating the key., No base64, password is "class" (do not use any salt or nosalt flag)

Nothing to record

### **2) What is the md5 hash of the decrypted text? It should match:**

MD5(napoleon.txt)= a2a0fcefc4ed5df172817bfb984eb0e1

Nothing to record.

## 3) Compute and record the sha512 hash of the decrypted text

26c8070b7e7ff8399a35350b935b5efe7840802b7105880444e212e2c1eaaeb72cc92d0cfcf1bd11c25b7a10594e0f59bcad82698473e6e9a75a31d0ae67def6 napoleon.txt

### **4) Encrypt the file cse467 with the aes algorithm. Pick some OTHER block method other than CBC. use the base64 option. Use the key "class".**

The hash for this file is: MD5(cse467.txt)= 4b6168df20343cbf006888b9fdcf4346

a) record the command you used to encrypt the message.

openssl -aes-256-ecb -e -a -pbkdf2 -in cse467 -out cse467.enc

b) Record the sha512 of the ciphered message.

1c0f25e48a1d1a335b6d3b7798bb0541fbb565f566d67afaaaff22071a96fdc7099e3af2814e1b2b6ea0655841344cd8abb7da4579f8227b5e539fe58d52e1cf -

### **5) Check integrity:**

Here are a group of hashes, all these files are found in /usr/bin. Verify which files match their hash.

SHA256(2to3-2.7)= 3ea002bead53f6bdf7fade8568285eb14146a7244f631570af55ee08ecef78f3

SHA256(411toppm)= 20475ec2902acda90f19edd9c9d9ba07ecd3350007f2b875fa14f57e64db477f

SHA256(a2ping)= 8df06502dbf434bbcfb7b105065529d3fa31df46f3875417491d7e7c6e797e79

SHA256(a5booklet)= f70ce449d55b6dc5bbc2f4c86439a6ff3c69f9671b6ce1e6a7f223987fc7735a

SHA256(a5toa4)= 20475ec2902acda90f19edd9c9d9ba07ecd3350007f2b875fa14f57e64db477f

SHA256(aa-enabled)= 183157256fece3f21eacad2851d93ca0182e49ffa6442bb7810ee38844b999d4

SHA256(aa-exec)= 2475009134343adefa43c0f12c6ce356e1b6fc2cbcef2a99978f24dcc605189e

SHA256(ab)= 4115fbbec65d29357e7208468bfabab42408e2a2eb83b7729a472b51f125b651

SHA256(aclocal)= d7582671b10a0517a4fa3f7dd59a6864b4a87656ca29f3f83a10a9cbe758e925

SHA256(aclocal-1.16)= d7582671b10a0517a4fa3f7dd59a6864b4a87656ca29f3f83a10a9cbe758e925

### **6) There is a program called brute.sh**

a) Write a brief description of what each line does.

Line 4 begins a for loop that spits out the content of each file in american-english directory

Line 6 runs the openssl decryption algorithm on each word in the american-english file to see if the passphrase hash matches

Line 7 checks if the previous command was successful (had an exitcode of 0)

Line 9 prints the word and the exitcode (0)

b) Use this program to find the key for the file brute.enc.

(hint: the key is one of the first 10 possible hits returned)

c) Why do multiple key's return a file that openssl says was successfully decrypted?

Because the hashes match

d) Describe how you found the correct key.

I tested each key on decrypting the file until I found one that made the file make sense

e) Record the key here.

Aristarchus

### **7) Calculate and record the sha512 hash's for the following files:**

/etc/passwd

c49bba06888a819279f8d7c97cc9c535ab0f1240b2f5496ddc46ebeeafd826e1eb9b69ec5b9fc107c7dbe89125a96a6bb79f160e8da8b194181b4604e07e2817 /etc/passwd

/usr/local/bin/ansible

02d948097e1dfef155038bfa213a56661c4c4aaa6ca9fbcb8b68f53d6cf22b687df48f990b3b2e3b5f3b4b6e8c025fbe34450256e313fafcc7076c4b3c0c812d /usr/local/bin/ansible

/usr/bin/gcc

aec701b31992668d416b16a94161c160543f9718923fed7fee3b081a64e8bb64740c0156c33d0658129a95b8a994442e80dbb26083b968bfbc09d188a4061a52 /usr/bin/gcc

## 8) Recreate your own tux ecb image.

There is a package at <https://github.com/pakesson/diy-ecb-penguin> that has a python program to create an encrypted image using either ecb or cbc.

However this program had two issues that I fixed. The fixed version is in /groups/cse467/crypto2/encrypt\_image.py. Use that version.

Use creative commons images from wikipedia to copy over an image to your public\_html directory on ceclnx01 and create an encrypted ecb version that displays some elements of the original image.

1. list the url to your encrypted image here.
   1. https://ceclnx01.cec.miamioh.edu/~dolljm/usain\_ecb.jpeg
2. List the url to the plain text version here
   1. https://ceclnx01.cec.miamioh.edu/~dolljm/usain.jpeg
3. What is the source of your image.
   1. https://commons.wikimedia.org/wiki/File:Usain\_Bolt\_Olympics\_Celebration.jpg

Hint: High contrast images work best. I ended up using a clipart image.