Paul Garewal
V00803658

Title

Lab 3:

The *Key* to Learning Encryption

Abstract:

Lab 3 focused on becoming familiar and utilizing encryption methods. We learned how to encrypt text, images as well as decrypt text using python code.
Overall the lab was able to provide a good introduction to encryption with its involved methods.

Aim:

This lab focused on encryption and getting familiar with its inner workings. Specifically the lab focused on getting familiar with encryption algorithms, encryption modes, paddings, and initial vectors (IV).

Introduction and Background:

Encryption is everywhere in society and is the backbone for many of our processes online. Essentially, all private information, banking, transactions, etc utilize encryption and its loss would be destructive, especially to businesses and secret information [1]. Therefore, it is of great importance to understand how it functions, to protect it and make encryption as secure as possible.

Method:

Using different ciphers (encryption algorithms) and the openssl library we were tasked with encrypting an image and plain text file. Additionally we witnessed the encryption with and without padding to analyze the hex information present before the file contents. Additionally, we implemented an IV experiment, where we used the same IV on an encrypted file, and reverse the contents to perform a *known-plaintext attack.* This allowed us to reveal the secret information presented in the lab in the text file.

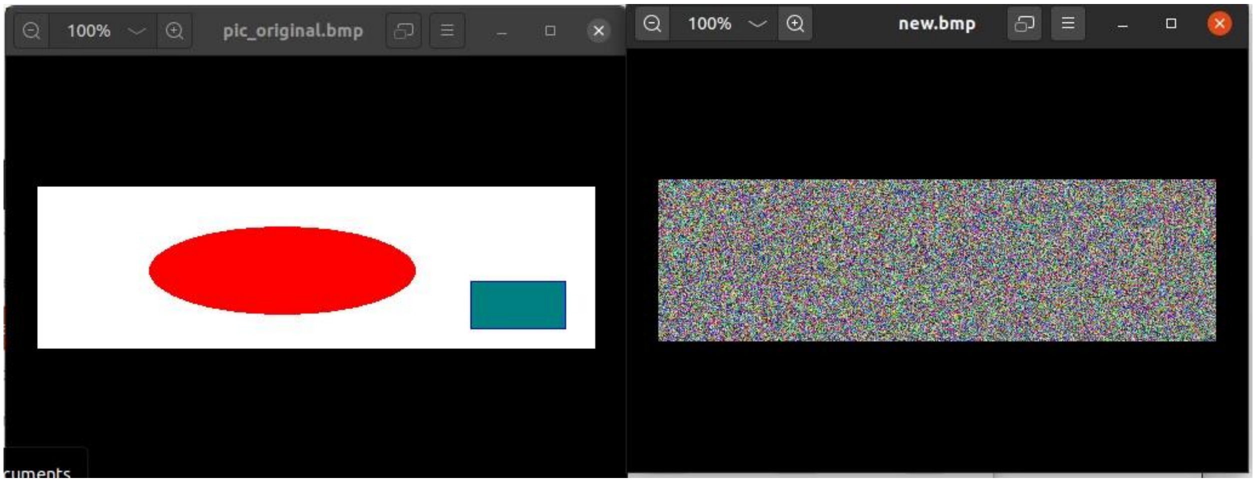
Results and Discussion:

Q1

Padding refers to a number of distinct practices. Furthermore, it refers to adding data to the beginning, middle or end of messages so they do not follow a predictable routine.

Q2

The encrypted image provided was still acknowledged as a normal picture and also still had a similar size to the original picture. Although we could not know the shape of the original picture, having the dimensions could be a useful resource, even if it was all static replaced.

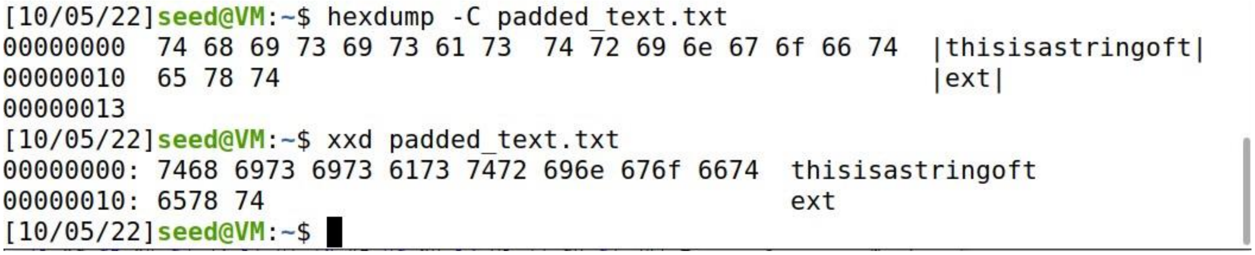


Q3

Similar to Q2 the encrypted image still had the same space dimensions although it was replaced by the static image. This would still provide us with *some* information.

Q4

The padded data was inputted before the message in our encryption, similar to before with "12345".



```
[10/05/22]seed@VM:~$ hexdump -C padded_text.txt
00000000  74 68 69 73 69 73 61 73  74 72 69 6e 67 6f 66 74  |thisisastringoft|
00000010  65 78 74                                          |ext|
00000013
[10/05/22]seed@VM:~$ xxd padded_text.txt
00000000: 7468 6973 6973 6173 7472 696e 676f 6674  thisisastringoft
00000010: 6578 74                                  ext
[10/05/22]seed@VM:~$ █
```

Q5

They must be unique since the file would be predictable with the IV's being un-unique. When the text file repeats, it is not advisable to use the same IV, which is what occurred here.

Q6

Order: Launch Missile!

We obtained this by using the decode.("utf-8") command in our python file

Q7

At most half of the data would be able to be revealed. Using the cfb cipher will cause the decryption to use the next block of data rather than the current block of data. Therefore, we could only get half of the message using this method of decryption.

References:

[1] "Why cloud data encryption is crucial for small businesses," *the Guardian*, Jul. 04, 2014. https://www.theguardian.com/small-business-network/2014/jul/04/cloud-data-encryption-small-businesses (accessed Oct. 07, 2022).