# NS\_Lab2 Report

Part 1:

1. Try to print to your screen the content of the input files, i.e., the plaintexts, using System.out.println().What do you see? Are the files printable?

Files are printable and we can see the file printed out.

1. Store the output ciphertext (in byte[] format) to a variable, say cipherBytes. Try to print the ciphertext of the smaller file using System.out.println(new String(cipherBytes)). What do you see? Is it printable?

Ciphertexts are not human-readable. They are a jumble of letters.

1. Now convert the ciphertext in Question 2 into Base64 format and print it to the screen. Is the Base64 encoded data generally printable?

It becomes printable with letters and digits but not in grammar.

1. Is Base64 encoding a cryptographic operation? Why or why not?

A cryptographic operation ensures there is an encrypted method for data exchange and most likely it is running with a “key” which encoding method doesn’t have. This suggest that Base64 is not a cryptographic operation.

1. Print out the decrypted ciphertext for the small file. Is the output the same as the output for question 1?

Yes.

1. Compare the lengths of the encryption result (in byte[] format) for smallFile.txt and largeFile.txt. Does a larger file give a larger encrypted byte array? Why?

SmallSize: Length of output encrypted byte[] 1480

LargeSize: Length of output encrypted byte[] 17360

Yes. Since more data are encrypted and stored in the cipher byte array.

Part 2:

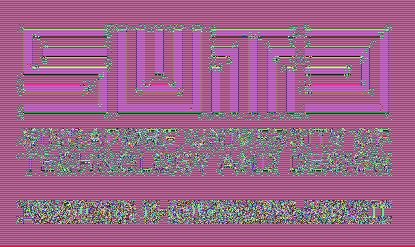
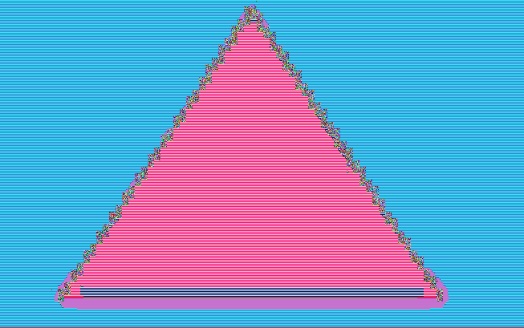
1. Compare the original image with the encrypted image. What similarities between them do you observe? Can you identify the original image from the encrypted one?

It shows that the shape of the triangle is similar between these two pictures which suggest that the vector shape are saved during the encryption.

To identify, Encrypted image is much more blurred than the original one. There are bar-like texture across on the encrypted image which is not on the original one. Some colors are entirely changed, for instance, the background color has changed from white into pink which suggest the encrypted image has a lower variety of the color than original one.

1. Why do those similarities exist? Explain the reason based on what you find out about how the ECB mode works.

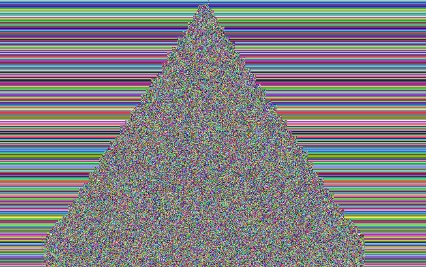
ECB: encrypt each block (64 bits) independently. Hence, repetitive areas in the input image result in repetitive patterns in the encrypted output which suggest that ECB has encrypt identical message blocks and in this way the texture shapes remains in the encrypted image.

ECB:

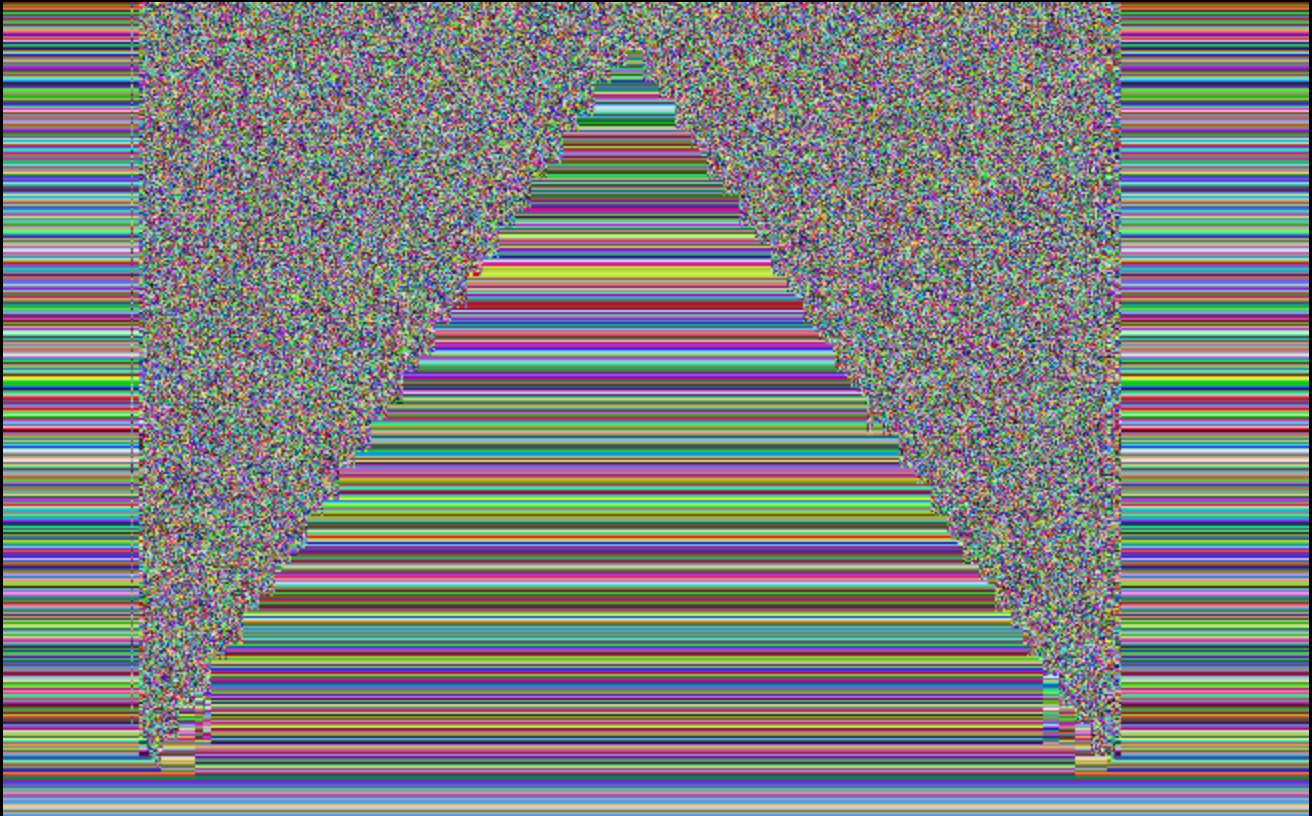
1. Now try to encrypt the image using the CBC mode instead (i.e., by specifying "DES/CBC/PKCS5Padding"). Compare the result with that obtained using ECB mode). What differences do you observe? Explain the differences based on what you find out about how CBC mode works.

CBC: use result of encrypting the previous block to encrypt each current block

CBC use a sequential method to change color in this case. Every encrypted point from below will use the same column reference point from above row and this will result an image with noise in column direction if the reference point has come across the earth.

CBC:

1. Do you observe any issue with image obtained from CBC mode encryption of “SUTD.bmp”? What is the reason of such observation? Can you explain and try on what would be result if data were taken from bottom to top along the columns of image? Can you try your new approach on “triangle.bmp” and comment on the observation?

There appears to be a reversal of colours (red to blue, green to pink and what not). The output of the encryption of SUTD.bmp appears to be less clear as opposed to the triangle. This might be because CBC ensures that every "#000000" hexadecimal color will have a different output, thus causing the file to appear as random. As CBC uses the column reference point from the bottom rather than from the top in question 3, the new approach on traingle.bmp shows a laterally inverted triangle (upside down) such that the outlines are clearly demarcated but the vector is harder to make out. 

Part 3:

1. What are the sizes of the message digests that you created for the two different files? Are they the same or different?

They are the same as 16 bytes since they both have went through a MD5 hashing method and MD5 algorithm used hash function produce a 128-bit(byte) hash value.

1. Compare the sizes of the signed message digests (in byte[ ] format) for smallFile.txt and largeFile.txt. Does a larger file size give a longer signed message digest? Why or why not?

Both signed message digests for smallFile.txt and largeFile.txt are same size of 128 bytes. Since signed with the key of 1024 bit in a RSA method will go through the 1024-bit RSA modulus which result a 128-byte result.