Phang Teng Fone 1003296 CSE: LAB 5

Tasks for Part 1

Complete the file DesTextStartingCode.java so that it can encrypt an input text file using DES. Use your program to encrypt the provided files (smallFile.txt and largeFile.txt) and answer the following questions:

Question 1 (2pt): 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?

Yes both files are printable.

For shorttext.txt, it is the lyrics of Lana Del Rey - Young and Beautiful.

```
Original content:
I've seen the world
Done it all
Had my cake now
Diamonds, brilliant
And Bel Air now
Hot summer nights, mid July
When you and I were forever wild
The crazy days, city lights
The way you'd play with me like a child
Will you still love me
When I'm no longer young and beautiful?
Will you still love me
When I got nothing but my aching soul?
I know you will, I know you will
I know that you will
Will you still love me when I'm no longer beautiful?
I've seen the world, lit it up
As my stage now
Channeling angels in the new age now
Hot summer days, rock 'n' roll
The way you play for me at your show
And all the ways I got to know
```

For longtext.txt, it is an eBook of “The Project Gutenberg EBook of The Sign of the Four, by Arthur Conan Doyle”

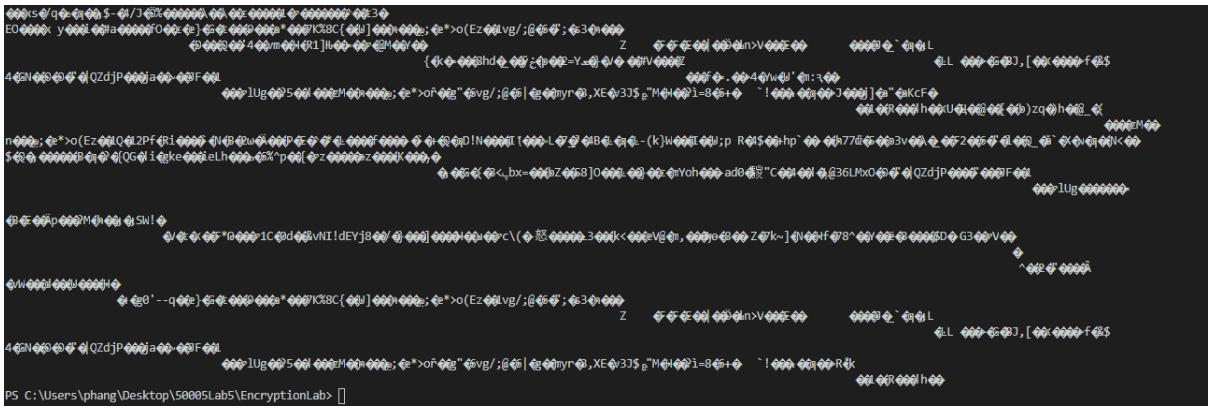
```
Original content:
The Project Gutenberg EBook of The Sign of the Four, by Arthur Conan Doyle

This eBook is for the use of anyone anywhere at no cost and with
almost no restrictions whatsoever.  You may copy it, give it away or
re-use it under the terms of the Project Gutenberg License included
with this eBook or online at www.gutenberg.net


Title: The Sign of the Four

Author: Arthur Conan Doyle

Posting Date: November 19, 2008 [EBook #2097]
Release Date: March, 2000
[This file last updated March 2, 2011]

Language: English


*** START OF THIS PROJECT GUTENBERG EBOOK THE SIGN OF THE FOUR ***
```

Question 2 (3pt): 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?

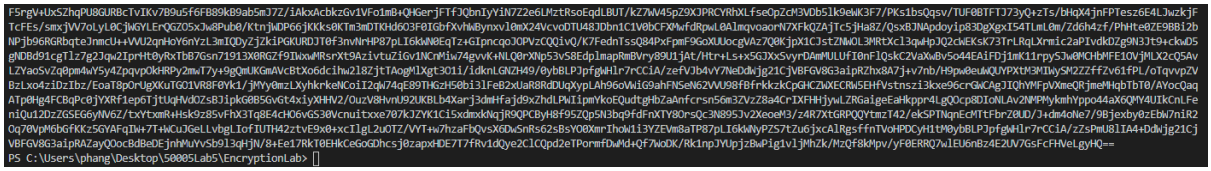
I see this:



It is unprintable.

Question 3 (3pt): Now convert the ciphertext in Question 2 into Base64 format and print it to the screen. Is the Base64 encoded data generally printable?

Printable with characters but not readable.



Question 4 (3pt): Is Base64 encoding a cryptographic operation? Why or why not?

No. Encoding means to convert a format, not necessarily secure. Encoding is not an encryption as it is used to transform data into another format using a scheme that is publicly available so that it can easily be reversed. On the other hand, encryption is used to transform data in such a way that only specific individual can reverse and view the actual data. Cryptographic requires an encrypted method for data exchange and have a key which the encoding method does not have thus it is not a cryptographic operation.

Question 5 (3pt): Print out the decrypted ciphertext for the small file. Is the output the same as the output for question 1?

Yes, it is the same.

```
Decrypted String:
I've seen the world
Done it all
Had my cake now
Diamonds, brilliant
And Bel Air now
Hot summer nights, mid July
When you and I were forever wild
The crazy days, city lights
```

```
Original Text:
I've seen the world
Done it all
Had my cake now
Diamonds, brilliant
And Bel Air now
Hot summer nights, mid July
When you and I were forever wild
The crazy days, city lights
```

Question 6 (4pt): 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?

```
The length of shorttext.txt in terms of byte[] format is: 1480
The length of longtext.txt in terms of byte[] format is: 17360
```

Yes. A larger file gives a larger encrypted byte array. This is because the byte size depends on the size of the inputs, the larger the inputs, the more blocks will be divided to be placed into the cipher, as the number of blocks increases, so does the bytes.

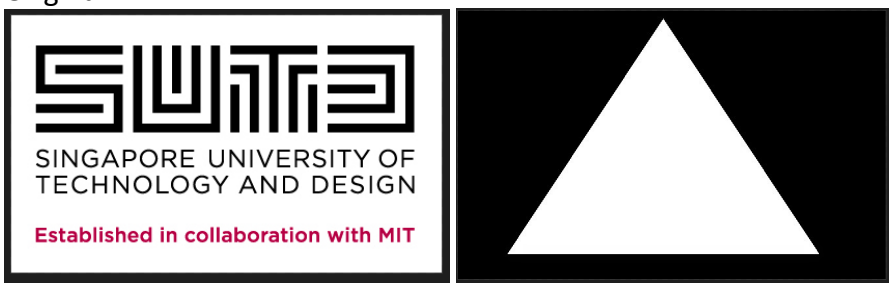
Task for part 2

Complete the file DesImageSolution.java to encrypt the input file, a .bmp image file using DES in ECB mode. You will need to specify the parameter "DES/ECB/PKCS5Padding" for creating your instance of the Cipher object.

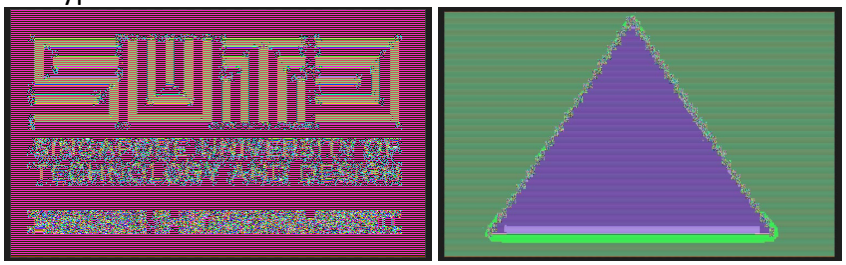
Note: Your encrypted file should also be in .bmp format. **Please ensure that your encrypted .bmp file can be opened using any image viewer you have in your computer.**

Question 1 (4pt): 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?

Original:



Encrypted:

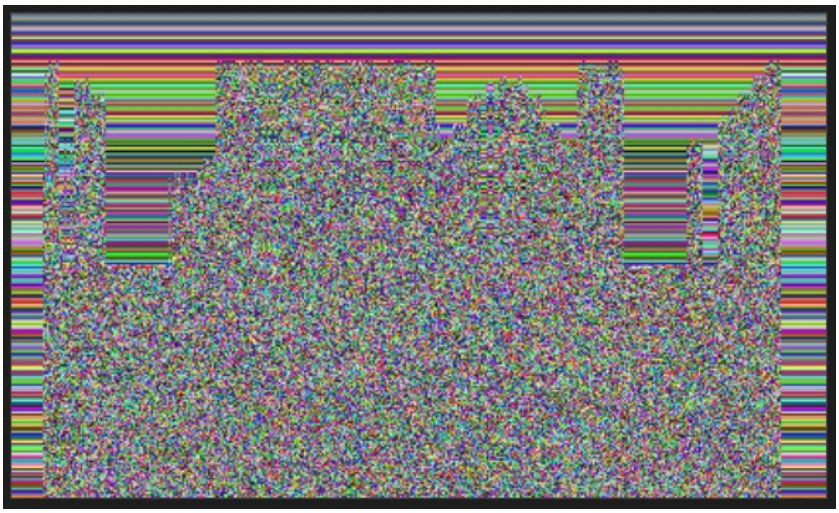


The SUTD word is very well visible while the rest of the texts are not very visible. Yes, the encrypted image has a different colour scheme

Question 2 (5pt): Why do those similarities exist? Explain the reason based on what you find out about how the ECB mode works.

ECB codes each block (64bits) independently. ECB is not semantically secure, by encrypting the same block using ECB, it always yields the same block of ciphertext. This allows an attacker to detect if two ECB-encrypted messages are identical, sharing a common prefix, detecting where and whether a single ECB-encrypted message contains repetitive data. Thus, with the above example, any images encrypted with ECB will always yield about the same pixels with a few disruptions, however it is still able to roughly make out what the image is.

Question 3 (8pt)**:** 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 (Cipher block chaining) uses result of encrypting the previous block to encrypt the next block. It uses a sequential method to change the pixel, every encrypted point from below will use the same column reference point from above which will result an image with noise in the column direction.

Question 4 (15pt): Do you observe any issue with image obtained from CBC mode encryption of "SUTD.bmp"? What is the reason for such observation? Can you explain and try on what would be the result if data were taken from bottom to top along the columns of the image? Can you try your new approach on "triangle.bmp" and comment on observation?

CBC mode takes a longer time to encrypt as compared to ESC. This is because CBC initialize a vector which then is XOR to the first block of a plaintext. Then after encrypting that block, the next block of plaintext is then XOR with the last encrypted block before encrypting the current block. CBC relies on each block to do its encryption thus taking a longer time. Also, for the output of CBC SUTD.bmp it looks random while the triangle.bmp CBC looks clears. This is because CBC ensures that every pixel colour will have a different output, as SUTD.bmp contains more coloured pixels as compared to a plain triangle, SUTD.bmp contains more random distortion. After taking the data from bottom to top, we can see the outline of the triangle, this is because CBC uses column reference point from the bottom, the new image will be an inverted triangle, the vector will still be visible as a triangle.

Task for Part 3

Complete the file DigitalSignatureSolution.java so that it can generate a signed message digest of an input file.

Apply your program to the provided text files (shorttext.txt, longtext.txt) and answer the following questions:

Question 1 (7pt): What are the sizes of the message digests that you created for the two different files? Are they the same or different?

```
The length of output digest for shorttext.txt is: 16
The length of output digest for longtext.txt is: 16
```

16 bytes, they are the same. Both of them went through the MD5 hashing method where no matter the size of the file, it will always produce a 128 bit hash value.

Question 2 (8pt): Compare the sizes of the signed message digests (in `byte[] encryptedBytes = eCipher.doFinal(data.getBytes());`
format) for shorttext.txt and longtext.txt. Does a larger file size give a longer signed message digest? Why or why not?

```
Signed message digest size of shorttext.txt is: 128
Signed message digest size of longest.txt is: 128
```

Same size of 128 bytes as singing with the key of RSA with 1024 bit will causes the result to be in a 128-byte format thus regardless of the size of the input, it will always remain the same at 128 bytes.