Computer Security Coding Assignment

Ghoradkar Prajwal Pandurang B160497CS

Nikunj Prasad B160024CS

Shirswar Malhar Hemantkumar B160511CS

Problem Statement:

To study the Avalanche effect in single DES and Triple DES.

- To demonstrate and analyse the ciphertext with single bit change in Plaintext keeping the key constant. (case 1)
- To demonstrate and analyse the ciphertext with single bit change in Key keeping the Plaintext constant. (case 2)
- To analyse and compare the time taken for encryption and decryption in single DES and Triple DES. (case 3)

What is Avalanche Effect?

The **avalanche effect** is the desirable property of cryptographic algorithms, typically block ciphers, wherein if an input is changed slightly (for example, flipping a single bit), the output changes significantly (e.g., half the output bits flip).[1]

Implementation (Case 1)

To demonstrate and analyse the Avalanche effect the **DES** and **DES3** modules from python's **Crypto.Cipher** library is used.[3][4]

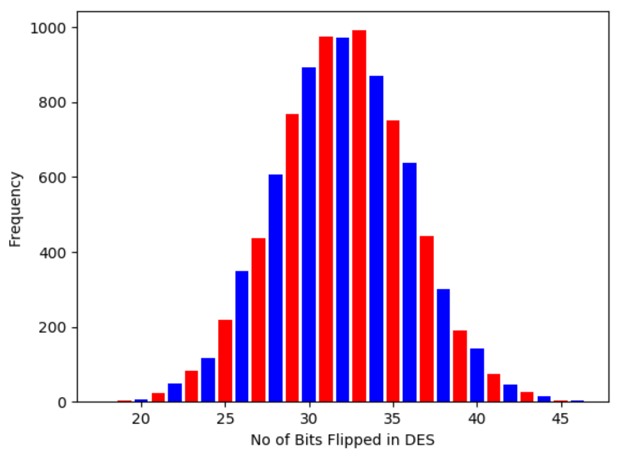
Initially a huge number of plaintexts (10000) is created such that the consecutive plaintext differs in a single bit. For this, the python library **random** is used. Initially a random number of 64 bit is generated an appended to an empty list of plaintexts. A **randomBitFlip**() function is created which takes in a 64-bit plaintext and gives output as another 64-bit number differing in single bit [2]. This number generated is appended to the list of plaintexts. This number generated is again fed to the function to create another plaintext. In such a way 10000 plaintexts are generated.

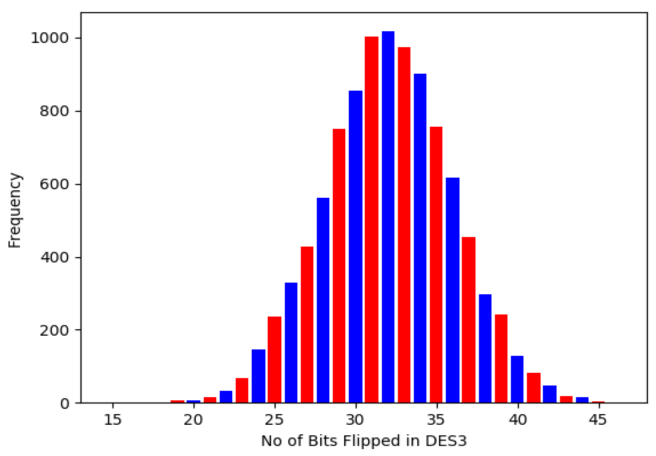
To evaluate the bits flipped in the ciphertext when a single bit is changed in plaintext we loop over our array of plaintext. We first create an object of DES with fixed key. Take two consecutive plaintext, encrypt them and send the two generated ciphertext to a function **HammingDistance()**. This function takes two 64 bit numbers as input and calculates the number of bit position in which the two bits are different.(Basically in our case it calculates the number of bits flipped in ciphertext with one bit change in plaintext). The hamming distance calculated is then stored in a python list. Similarly, we do it for the rest of 10000 plaintexts.

Results (case 1)

The mean of all the hamming distance over 10000 plaintexts differing in single bit is approximately **32** for both DES and Triple DES.

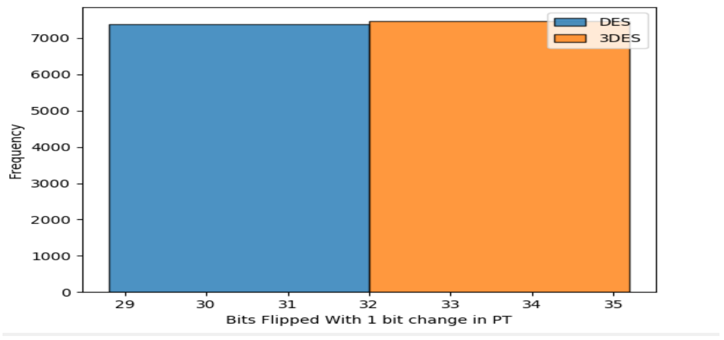
The graph of no of bits flipped (x-axis) vs frequency of the bits flipped (y-axis) for DES (left image) and Triple DES (right image) is shown below:





The above graph shows us that the most of the bits flipped are in the range 30-35.

Another graph of comparing the results of DES and Triple DES is plotted. This graph shows us the sum of the frequency of the bits flipped in the range 28-36. We selected this range because the standard deviation is approximately 4 and hence the deviation from half that is 28 = 32 -4 and 36 = 32 + 4



As we can see that there is a very slight difference in their performance.

Conclusion (case 1)

We can conclude by saying that both DES and Triple DES shows a good avalanche effect as 1 bit change in plaintext changes approximately half of the bits in the cipher text. But when it comes to comparison between DES and Triple DES, we can say that neither of it gives a significant difference than the other. The above statement can be backed up with this paper [5] which stated that 3DES is more efficient than DES in avalanche effect. While another paper [6] compares various algorithms in which the avalanche effect of DES is more than that of 3DES. Also, when we try running our program for different random values, we sometimes get the performance of DES better than 3DES.

Implementation (Case 2)

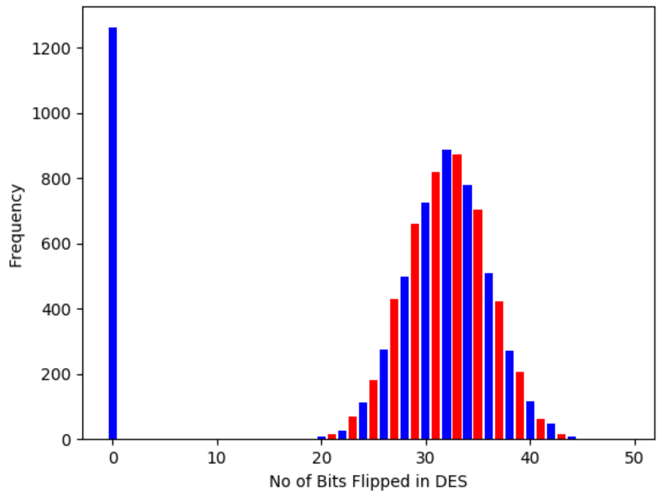
The code implementation for case 2 is similar to that of case 1. We have the same randomBitFlip() function for generating the keys this time, than plaintext in earlier case. The same HammingDistance() function is used for calculating the bits flipped in the ciphertext. This time the plaintext is kept constant and the DES or DES3 object is initialized inside the loop with different key every time.

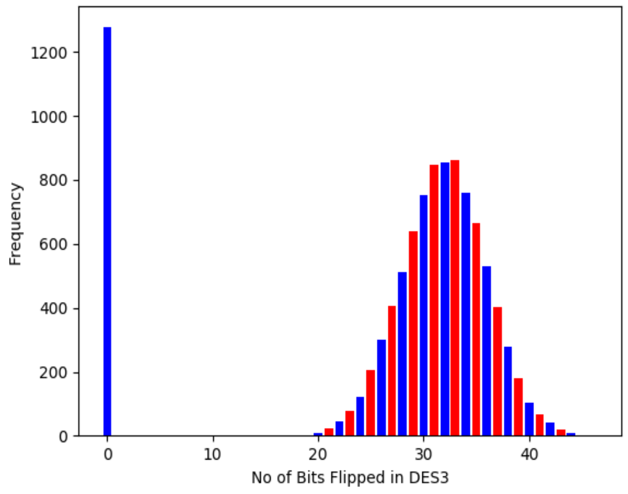
Results (Case 2)

The mean of hamming distance obtained with single bit change in the key is approximately **28** for both DES and Triple DES and the standard deviation is approximately 11.

The graph for both DES and DES3 can be seen below:

The big bar on zero can be seen because the DES has an effective key length of 56-bit. The 8 bits from the 64-bit key are parity bits. If any of these parity bits is flipped when changing the key by 1 bit the ciphertext doesn't change. The parity bit is always the least significant bit of each byte.





Conclusion (case 2)

The performance of both DES and Triple DES is approximately the same. But we see that, if one of the parity bits is flipped, when flipping the key by 1 bit the ciphertext doesn't change in both DES and 3DES.

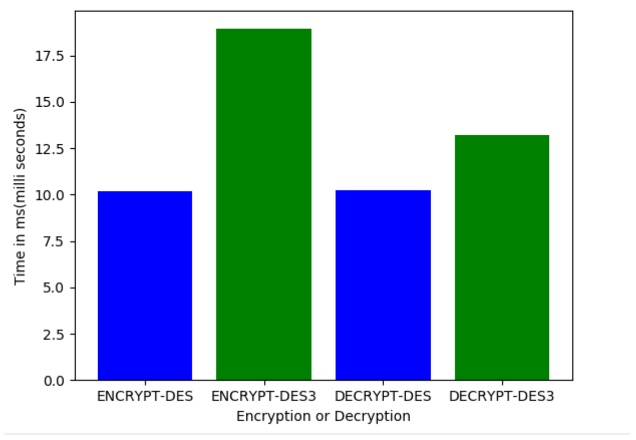
Implementation (case 3)

In this case, we create DES and DES3 object. Create the 10000 plaintexts similar to 1st case. Encrypt all the 10000 plaintext and note the time required using the **time** library in python. Similarly, for decryption the earlier ciphertext stored in a list are decrypted back to plaintext and the time is noted and averaged out for the number of iterations.

Result (Case 3)

A graph for the time required for encryption and decryption for both DES and DES3 is plotted:

The encryption time for DES: 10.17ms	The encryption time for DES3: 18.95ms
The decryption time for DES: 10.25ms	The decryption time for DES3: 13.24ms



Conclusion (case 3)

The encryption time for 3DES is more than that of DES because triple DES goes through encryption-decryption-encryption in its encryption step and vice-versa for decryption [5]. Also, it is observed that the decryption time is less than the encryption time [6].

References

[1] https://en.wikipedia.org/wiki/Avalanche_effect

[2] https://stackoverflow.com/questions/40071311/how-to-change-1-bit-from-a-string-python

[3] https://pycryptodome.readthedocs.io/en/latest/src/cipher/des.html

[4] https://pycryptodome.readthedocs.io/en/latest/src/cipher/des3.html

[5] https://www.ijcaonline.org/research/volume130/number14/rao-2015-ijca-907190.pdf

[6] https://www.sciencedirect.com/science/article/pii/S1877050916001101

For project code please refer this link :

https://github.com/PrajwalG12121998/Avalanche-Effect-in-DES-and-Triple-DES