SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA ENGINEERING AND TECHNOLOGY

E	0	2	1	9	0	0	7

UNIQUE ID

B. Tech. DEGREE

CONTINUOUS ASSESSMENT 4

OCT – 2020: II YEAR – QUARTER 6

CSE 270: Design and Analysis of Algorithm

(Common to B.Tech CSE - AI & ML, CyS & IoT)

Time: 1 hour – 15 minutes **Maximum Mark: 15** **Date: 31/10/2020**

Problem Statement :

Design and implement algorithm for **public key cryptosystem** to encrypt the banking transaction for a e-commerce application. Analyse the complexity of the algorithm with baseline methods.

Programming Language :

Python is used as a programming language for writing code on encrypting algorithms.

Platform :

Windows (OS) is used as base operating system for implementing this Problem task.

Description of Task :

The Banking Transaction are consider as most important transaction for the Country, People and for Bank. So the Encryption Algorithm must be highly secure , makes attackers like hackers and anonymous forces not able to break those encryption .

Therefore, for Encryption I Choose **DES (Data Encryption System)**, On specific I am using **Triple Key Data Encryption Standard (TKDES)** for encryption.

WHY I CHOOSE TRIPLE KEY DATA ENCRYPTION FOR THIS PROBLEM ? :

Triple Data Encryption Standard (DES) is a type of computerized cryptography where block cipher algorithms are applied three times to each data block. The key size is increased in Triple DES to ensure additional security through encryption capabilities. Each block contains 64 bits of data. Three keys are referred to as bundle keys with 56 bits per key. There are three keying options in data encryption standards:

- **All keys being independent**
- **Key 1 and Key 2 being Independent**
- **All three are being identical**

Triple DES is advantageous because it has a significantly sized key length, which is longer than most key lengths affiliated with other encryption modes.

Triple DES transfers key over the network and the permutation of keys generated while the transaction happens through the network, which makes DES algorithm more safe algorithm at symmetric list.

When user do transaction , the algorithm invokes and messages transmitted over network in form of cypher text and so on.

The complexity of this algorithm to break is exponential , makes this algorithm more secure and safe for banking transaction. Eventhough it takes more time comparing DES (Data encryption Standard) , It three times secure than DES .

Algorithm Development :

There are Three things to deal with this Triple DES algorithm :

- Generating Keys
- Steps for Encryption
- Steps for Decryption

Generating Keys :

There are 16 rounds of encryption in the algorithm, and a different key is used for each round. How keys are generated is listed below.

1. Compress and transpose the given 64-bit key into a 48-bit key using the following table:

```
// The array elements denote the bit numbers
int pc1[56] = {
  57,49,41,33,25,17,9,
  1,58,50,42,34,26,18,
  10,2,59,51,43,35,27,
  19,11,3,60,52,44,36,
  63,55,47,39,31,23,15,
  7,62,54,46,38,30,22,
  14,6,61,53,45,37,29,
  21,13,5,28,20,12,4
};
```

2. Divide the result into two equal parts: C and D.
3. C and D are left-shifted circularly. For encryption rounds 1, 2, 9, and 16 they are left shifted circularly by 1 bit; for all of the other rounds, they are left-circularly shifted by 2.
4. The result is compressed to 48 bits in accordance with the following rule:

```
int pc2[48] = {
  14,17,11,24,1,5,
  3,28,15,6,21,10,
  23,19,12,4,26,8,
  16,7,27,20,13,2,
  41,52,31,37,47,55,
  30,40,51,45,33,48,
  44,49,39,56,34,53,
  46,42,50,36,29,32
};
```

5. The result of step 3 is the input for the next round of key generation.

Steps for Encryption :

1. Transpose the bits in the 64-block according to the following:

```
// 58 means that the 58th bit should be considered
// the first bit, 50th bit the second bit and so on.
int initial_permutation_table[64] = {
    58,50,42,34,26,18,10,2,
    60,52,44,36,28,20,12,4,
    62,54,46,38,30,22,14,6,
    64,56,48,40,32,24,16,8,
    57,49,41,33,25,17,9,1,
    59,51,43,35,27,19,11,3,
    61,53,45,37,29,21,13,5,
    63,55,47,39,31,23,15,7
    };
```

2. Divide the result into equal parts: left plain text (1-32 bits) and right plain text (33-64 bits).
3. The resulting parts undergo 16 rounds of encryption in each round.

   The Right plain text expanded using the following expansion table:

```
// The array elements denote the bit numbers
int expansion_table[48] = {
  32,1,2,3,4,5,4,5,
  6,7,8,9,8,9,10,11,
  12,13,12,13,14,15,16,17,
  16,17,18,19,20,21,20,21,
  22,23,24,25,24,25,26,27,
  28,29,28,29,30,31,32,1
};
```

4. The expanded right plain text now consists of 48 bits and is XORed with the 48-bit key.

5. The result of the previous step is divided into 8 boxes. Each box contains 6 bits. After going through the eight substitution boxes, each box is reduced from 8 bits to 6 bits. The first and last bit of each box provides the row index, and the remaining bits provide the column index. These indices are used to look-up values in a substitution box. A substitution box has 4 rows, 16 columns, and contains numbers from 0 to 15.

6. The result will be in accordance of following rule :

```
// The array elements denote the bit numbers
int permutation_table[32] = {
    16,7,20,21,29,12,28,17,
    1,15,23,26,5,18,31,10,
    2,8,24,14,32,27,3,9,
    19,13,30,6,22,11,4,25
};
```

7. XOR the left half with the result from the above step. Store this in the right plain text.
8. Store the initial right plain text in the left plain text.
9. These halves are inputs for the next round. Remember that there are different keys for each round.
10. After the 16 rounds of encryption, swap the left plain text and the right plain text.
11. Finally, apply the inverse permutation (inverse of the initial permutation), and the ciphertext will be generated.

Steps for Decryption :

The order of the 16 48-bit keys is reversed such that key 16 becomes key 1, and so on. Then, the steps for encryption are applied to the ciphertext.

This are the steps for Encryption and Decryption of the DES algorithm , where Triple keys data encryption System highly depends on DES. So the next is what is triple key DES..

Implementation of Triple DES using Python :

For this task I am using inbuilt Triple DES Extension developed by pip .

At Command line :

pip install pycrypto // which install all cryptographic algorithm developed code .

Transaction.py (python file) :

```
from Crypto.Cipher import DES3     //Triple DES

from Crypto import Random

key =  'Example key     '          //16 byte key

nsize = Random.new().read(DES3.block_size)      //block size is 8

cipher_encrypt = DES3.new(key, DES3.MODE_CTR, nsize)    //inbuilt encryption function

plaintext = 'This is for checking tripleDES  '  // plain text assigning.

cipher_text = cipher_encrypt.encrypt(plaintext)


//three key repetition

cypher_decrypt = DES3.new(key, DES3.MODE_CTR, nsize)     //we cannot use same key

cipher_decrypt.decrypt(cipher_text)

cipher_decrypt.decrypt(cipher_text)
```

//There are different modes are available DES3

//MODE_ECB, MODE_CBC, MODE_OFB must have plain text with multiple of block size

//MODE_CTR plaintext can be any length

Above code access Triple DES to encrypt our key and message , which functions with the help of pycrypto and decrypts a message (over transaction) .

Possible attacks on Transaction :

Triple DES goes through multiple encryption exercise to build a composite cipher that is stronger than Single DES. Because of **Meet-in-the-middle attacks**, Triple DES has 112 bits of Strength.

There is argument on the power of Triple DES over **Double DES**, but there is large difference between theoretical attack vs real one. Lets take an example, if there is attack with 2^80 cipher blocks , which would reduce the strength of three keys, so no stronger than 112 bits. This attack may be worthy of publication, but in real life it is not possible for such a attack. A tera-block is **2^40 blocks**. With this attack, we would need eight tera-terabytes of memory and a CPU that could address that much.

So Therefore Triple DES is stronger alternative to any of the standard 128 bit ciphers. And it is safe to held those kind of transaction .

Comparison Of Cryptographic Algorithms :

- DES
- 3DES (selected one)
- AES (advanced Encryption System)
- RSA Algorithm
- Blowfish

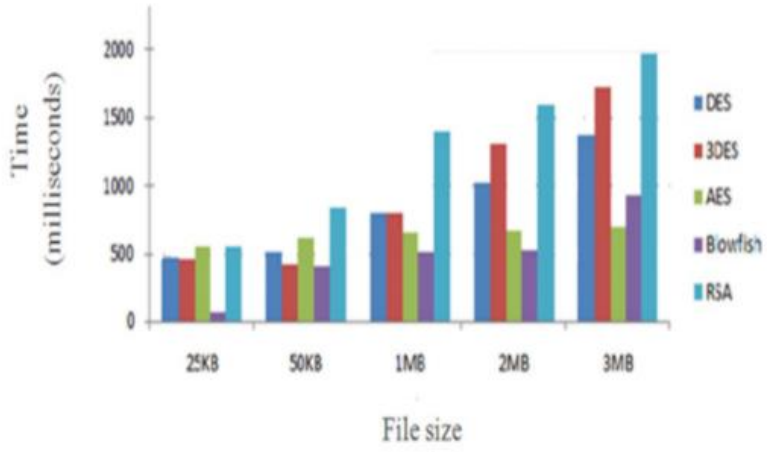
Even though there are lots of cryptographic methods , selection depends on **response time , bandwidth, confidentiality and integrity .** However each cryptographic methods has their own strength and weakness. Lets compare them with analysis and arrive the conclusion .

Comparing the following factors with other algorithms :

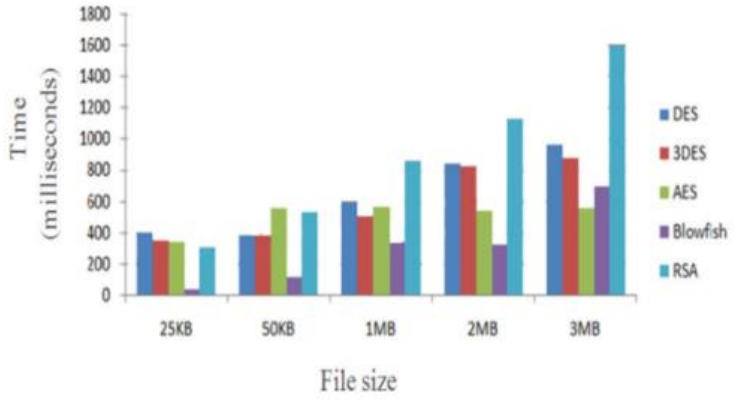
- Encryption time
- Decryption time
- Memory used
- Avalanche effect
- Entropy
- Number of bits needed to encoding optimally.

Results and Discussion :

- As Figure shows below , the blowfish algorithm records the fastest encryption time, and RSA algorithm records the slowest encryption time. Based on the encryption time we will see below



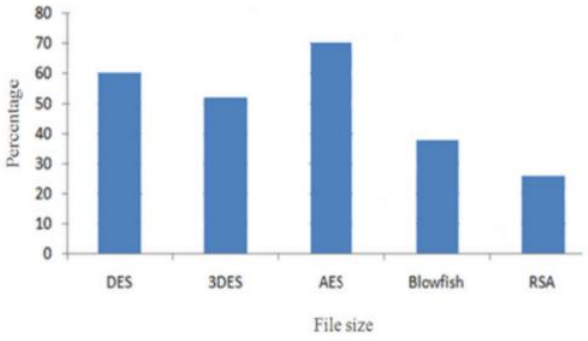
- Next the decryption time for all algorithms is faster than the encryption time. Also Blowfish algorithm records the fastest decryption time and RSA algorithm records the slowest decryption time. Based on the decryption time we can see Triple key DES is far behind DES because of its linear complexity.



- Then the below table presents that memory used for unit operations for all cryptographic techniques that we studied. Blowfish consumed less memory storage than other types, while RSA is Highest. Triple DES is consuming more space, because of its high key secure. But size wise its not good.

Algorithm	Memory used (KB)
DES	18.2
3DES	20.7
AES	14.7
Blowfish	9.38
RSA	31.5

- Next figure below displays about AES manifests the highest **avalanche effect**, where RSA manifest least avalanche effect. This has turned attend to AES . Triple DES places third with decent effect.



- As the entropy test and final experiment. Results shows that blowfish records the highest average entropy per byte of encryption. Where Triple DES is not at this measure.

Algorithm	Average entropy per byte of encryption
DES	2.9477
3DES	2.9477
AES	3.84024
Blowfish	3.93891
RSA	3.0958

- Last table Presents Number of bits to be encoded optimally, where as DES demands lowest. Triple DES places better place in this comparison.

Algorithm	Average number of bits demanded to optimally encode a byte of encrypted data
DES	27
3DES	40
AES	256
Blowfish	128
RSA	44

Conclusion :

All Cryptographic algorithms has **weakness** points and **strength** points. We will select cryptographic algorithm on our basis of problem. As I Selected Triple Key DES which is better for securing banking transaction next to Advanced Data Encryption. Also by bandwidth comparison DES and Triple DES places better position , so this factor is important for banking transaction then it will be good option for transaction .

Advanced Encryption algorithm is better than Triple DES but both has their own perspective of strength and weakness. In Future new kinds of algorithm may arrive and replaces many algorithms.