SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

INFORMATION SECURITY ANALYSIS AND AUDIT

REVIEW 3

ENCRYPTING AND DECRYPTING TEXT FILE USING HYBRID CRYPTOSYSTEM

TEAM MEMBERS:

18MIS0163- D HARISH

18MIS0193-A R HARIHARAN

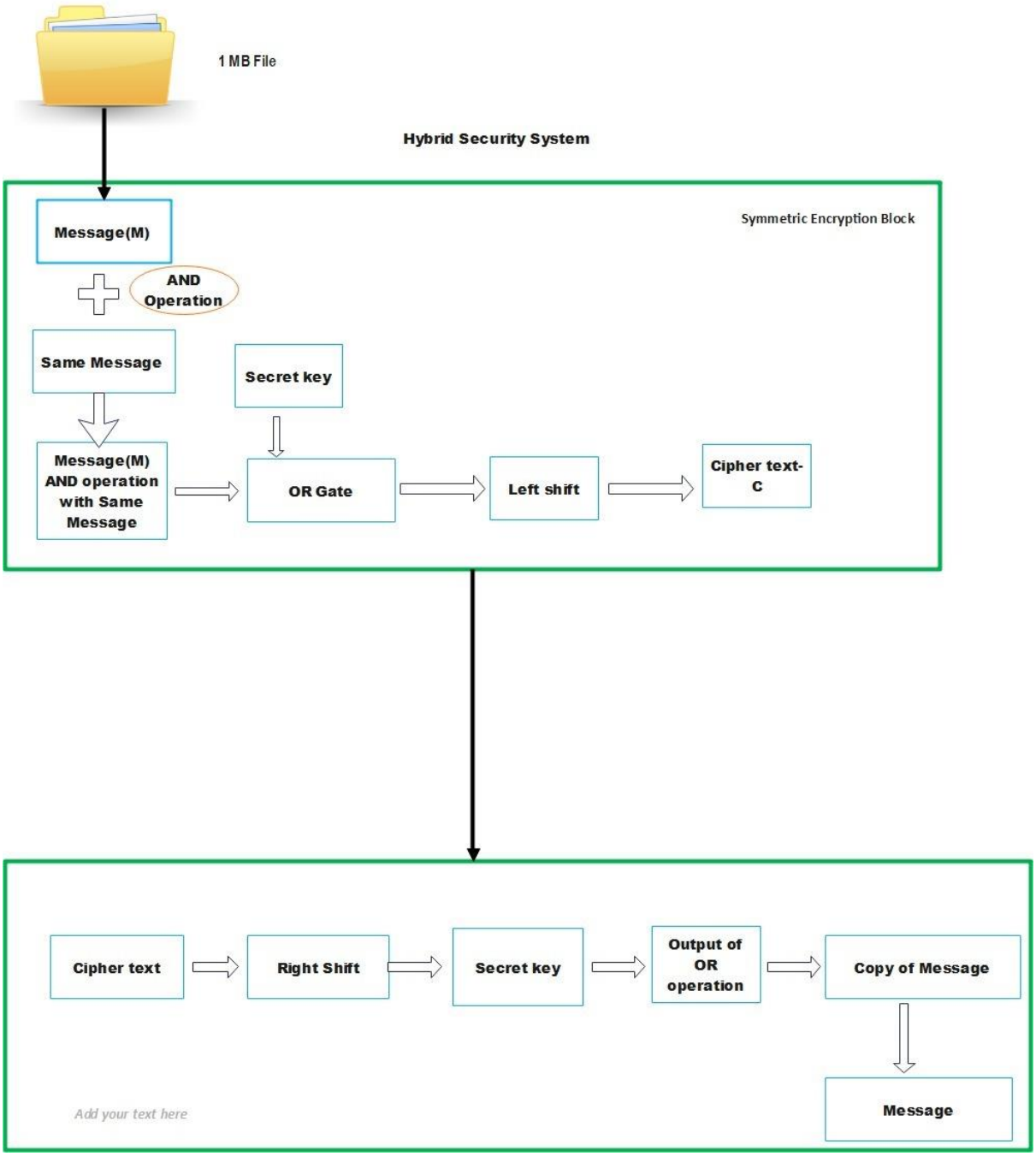
18MIS0260-R SUDHARSHAN

SLOT : F2

CODE : CSE3501

FACULTY:Prof.CHANDRASEGAR.T

ARCHITECTURE:



ALGORITHM:

CREATING A TEXT FILE LESS THAN 1 MB

Then creating the path to encrypt the data from the text file using previously created Hybrid Cryptosystem.

IMPLEMENT SYMMETRIC ENCRYPTION

1) After giving the input(message), duplicate the same message and perform AND GATE operation.

2) Now perform the output of previous step with the symmetric private key in OR GATE operation.

3) Now perform left shift of the previous output and consider it as CIPHER TEXT

IMPLEMENT SYMMETRIC DECRYPTION

1) Now perform right shift to the Symmetric Encrypted text.

2) Perform OR GATE for the output of the previous step with the symmetric private key.

3) Perform AND GATE with the output of the previous step and duplicate message.

4) Now, we get the entered original message as decrypted value

READING THE FILE USING:

```
try {
        File myObj = new File("demo.txt");
        Scanner myReader = new Scanner(myObj);
        while (myReader.hasNextLine()) {
          BigInteger data = myReader.nextBigInteger();
          System.out.println(data);
}
}
```

Calculating the time using:

```
long start=System.currentTimeMillis();

long enc_time =System.currentTimeMillis();

long dec_time =System.currentTimeMillis();
```

Printing the time:

```
System.out.println("Time taken to Encrypt the text file : " + (enc_time - start) +
" ms ");
        System.out.println("Time taken to Decrypt the text file : " + (dec_time -
enc_time) + " ms ")
```

CODING:

```
import java.util.*;
import java.math.*;
import java.io.File;  // Import the File class
import java.io.FileNotFoundException;  // Import this class to handle errors
import java.util.Scanner; // Import the Scanner class to read text files

import javax.xml.crypto.Data;
public class Testing
{

    // Returns true if n is prime
    static boolean isPrime(int n)
    {
        // Corner cases
        if (n <= 1)
        {
            return false;
        }
        if (n <= 3)
        {
            return true;
        }

        // This is checked so that we can skip
        // middle five numbers in below loop
        if (n % 2 == 0 || n % 3 == 0)
        {
            return false;
        }

        for (int i = 5; i * i <= n; i = i + 6)
        {
            if (n % i == 0 || n % (i + 2) == 0)
            {
                return false;
            }
        }

        return true;
    }


    // Utility function to store prime factors of a number
    static void findPrimefactors(HashSet<Integer> s, int n)
```

```
    {
        // Print the number of 2s that divide n
        while (n % 2 == 0)
        {
            s.add(2);
            n = n / 2;
        }

        // n must be odd at this point. So we can skip
        // one element (Note i = i +2)
        for (int i = 3; i <= Math.sqrt(n); i = i + 2)
        {
            // While i divides n, print i and divide n
            while (n % i == 0)
            {
                s.add(i);
                n = n / i;
            }
        }

        // This condition is to handle the case when
        // n is a prime number greater than 2
        if (n > 2)
        {
            s.add(n);
        }
    }

    // Function to find smallest primitive root of n
    static int findPrimitive(int n)
    {
        HashSet<Integer> s = new HashSet<Integer>();

        // Check if n is prime or not
        if (isPrime(n) == false)
        {
            return -1;
        }

        // Find value of Euler Totient function of n
        // Since n is a prime number, the value of Euler
        // Totient function is n-1 as there are n-1
        // relatively prime numbers.
        int phi = n - 1;

        // Find prime factors of phi and store in a set
        findPrimefactors(s, phi);
```

```
        // Check for every number from 2 to phi
        for (int r = 2; r <= phi; r++)
        {
            // Iterate through all prime factors of phi.
            // and check if we found a power with value 1
            boolean flag = false;
            for (Integer a : s)
            {

                // Check if r^((phi)/primefactors) mod n
                // is 1 or not
                if (power(r, phi / (a), n) == 1)
                {
                    flag = true;
                    break;
                }
            }

            // If there was no power with value 1.
            if (flag == false)
            {
                return r;
            }
        }

        // If no primitive root found
        return -1;
    }
    static int calmodInv(int a, int b)
    {
      a = a % b;
      for (int x = 1; x < b; x++)
      if ((a * x) % b ==1)
        return x;
      return 1;
    }
    static int power(int at, int bt, int ct)
  {
    int res = 1; // Initialize result

    at = at % ct; // Update x if it is more than or
    // equal to p

    if (at == 0)
      return 0; // In case x is divisible by p;

    while (bt > 0)
    {
```

```
      // If y is odd, multiply x with result
      if ((bt & 1) != 0)
        res = (res * at) % ct;

      // y must be even now
      bt = bt>> 1; // y = y/2
      at = (at * at) % ct;
    }
    return res;
  }
    public static void main(String[]args) {
        //symmetric encryption
        long start=System.currentTimeMillis();
        System.out.println(" ");
        System.out.println(" ");
         System.out.println( "IMPLEMENT SYMMETRIC ENCRYPTION:");
         System.out.println("-------------------------------------------");
         System.out.println("Symmetric Encryption:");
         System.out.println("-------------------------------------------");
         System.out.println(" ");
         System.out.print("MESSAGE READ FROM FILE : ");
         Scanner sc = new Scanner(System.in);
         try {
            File myObj = new File("demo.txt");
            Scanner myReader = new Scanner(myObj);
            while (myReader.hasNextLine()) {
              BigInteger data = myReader.nextBigInteger();
              System.out.println(data);

         BigInteger val1 = data;
         System.out.print("Enter the message for secret key:");
         BigInteger val2 = sc.nextBigInteger();
         BigInteger val3 = (val1.and(val1));
         System.out.println("AND operation of message and duplicate message:
"+val3);
         BigInteger val4 = (val3.or(val2));
         System.out.println("OR operation of previous step and key: "+val4);
         BigInteger c=val4.shiftLeft(1);
         System.out.println("Left shift of previous step: "+c);
         System.out.println ("Ciphertext ct:" + c);
         System.out.println(" ");
         long enc_time =System.currentTimeMillis();

         //

          // print result value//Symmetric decryption
          System.out.println(" ");
```

```
          System.out.println("IMPLEMENT SYMMETRIC DECRYPTION");
          System.out.println("-------------------------------------------");
          System.out.println("symmetric  Decryption:");
          System.out.println(" ");
          System.out.println("-------------------------------------------");

          BigInteger z=c.shiftRight(1);
          System.out.println("Right shift of Asymmetric Decrypted text:"+z);
          BigInteger val5 = (z.or(val2));
          System.out.println("OR operation of previous step and key:"+val5);
          BigInteger val6 = (val5.and(val1));
          System.out.println("AND operation of previous step and duplicate
message:"+val6);
          System.out.println(" ");
          System.out.println("-------------------------------------------");
          System.out.println("Decrypted Original Message:"+val6);
          System.out.println(" ");
          System.out.println("-------------------------------------------");

          //
          long dec_time =System.currentTimeMillis();
          System.out.println("-------------------------------------------");
          System.out.println("-------------------------------------------");
            System.out.println("Time taken to Encrypt the text file : " +
(enc_time - start) + " ms ");
            System.out.println("Time taken to Decrypt the text file : " +
(dec_time - enc_time) + " ms ");
            System.out.println("Overall Timing : "+ (dec_time - start) + " ms
");
            System.out.println("-------------------------------------------
");
            System.out.println("-------------------------------------------
");
        }
          sc.close();
          myReader.close();
          } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
          }

     }
 }
```

SAMPLE OUTPUT:

```
IMPLEMENT SYMMETRIC ENCRYPTION:
-------------------------------------------
Symmetric Encryption:
-------------------------------------------

MESSAGE READ FROM FILE : 14
Enter the message for secret key:3
AND operation of message and duplicate message: 14
OR operation of previous step and key: 15
Left shift of previous step: 30
Ciphertext ct:30


IMPLEMENT SYMMETRIC DECRYPTION
-------------------------------------------
symmetric  Decryption:

-------------------------------------------
Right shift of Asymmetric Decrypted text:15
OR operation of previous step and key:15
AND operation of previous step and duplicate message:14

-------------------------------------------
Decrypted Original Message:14

-------------------------------------------
-------------------------------------------
-------------------------------------------
Time taken to Encrypt the Message : 1766 ms
Time taken to Decrypt the Message : 2 ms
Overall Timing : 1768 ms
-------------------------------------------
-------------------------------------------
```