SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

INFORMATION SECURITY ANALYSIS AND AUDIT

REVIEW 1

CREATION OF HYBRID SYSTEM USING SYMMETRIC AND ASYMMETRIC CRYPTOGRAPHY

TEAM MEMBERS:

18MIS0163- D HARISH

18MIS0193-A R HARIHARAN

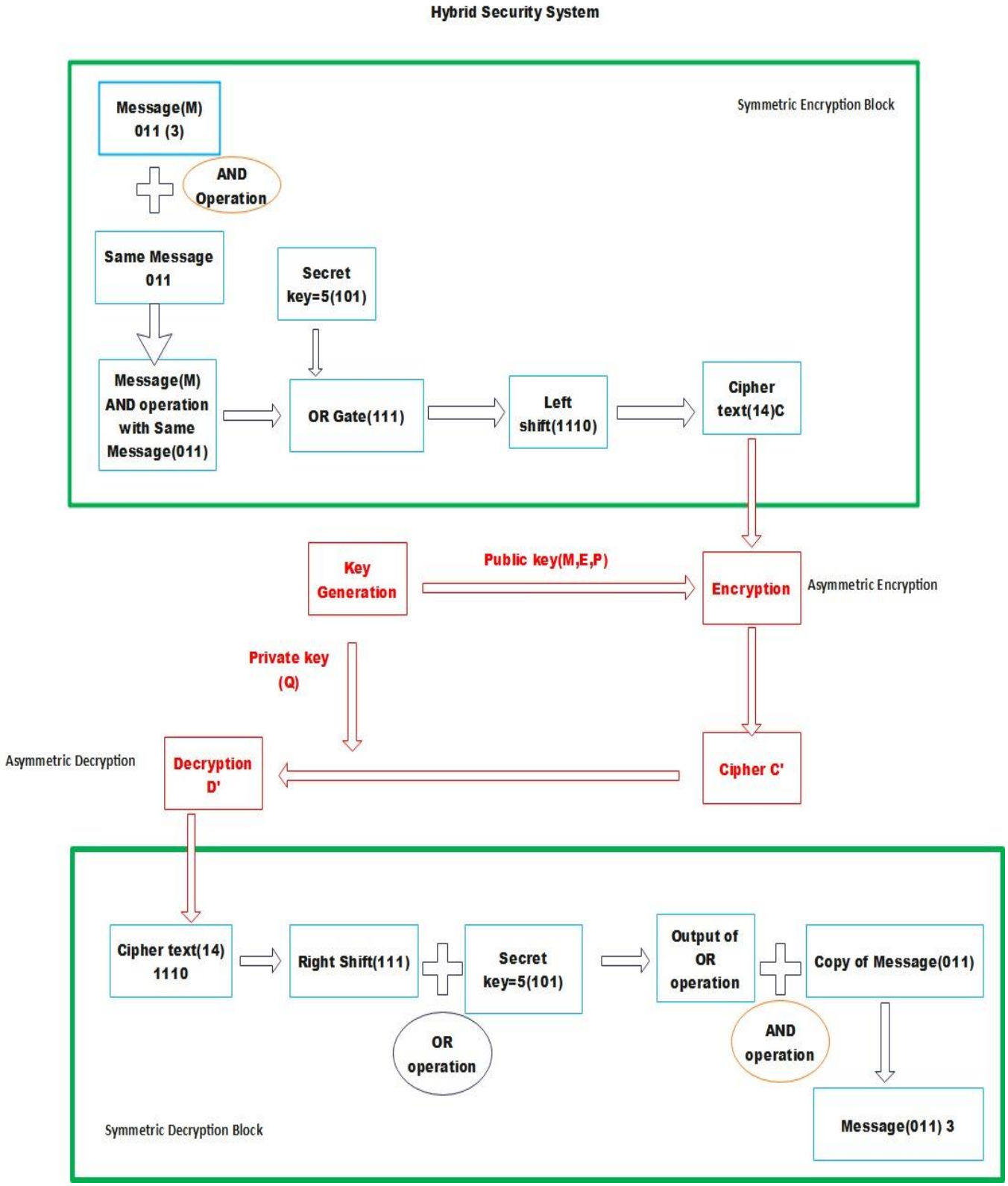
18MIS0260-R SUDHARSHAN

SLOT : F2

CODE : CSE3501

FACULTY:Prof.CHANDRASEGAR.T

ARCHITECTURE:



ALGORITHM:

STEP 1: IMPLEMENT SYMMETRIC ENCRYPTION

STEP 2: ASYMMETRIC KEY GENERATION

STEP 3: IMPLEMENT ASYMMETRIC ENCRYPTION

STEP 4: IMPLEMENT ASYMMETRIC DECRYPTION

STEP 5: IMPLEMENT SYMMETRIC DECRYPTION

EXPLANATION:

STEP 1: 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.

STEP 2: ASYMMETRIC KEY GENERATION

Public keys: M, E, P

1)let us consider, P be prime number p=17 and M ∈ P.

2)Lets, take M=3 where m is a primitive element of mod p

Private key: Q

3)Choose Q secret 'Q' and compute E ≡M^ Q mod p

4)COMPUTING Q AND E

Choose a random Q =14

Compute e =m ^Q mod p

=3^14 mod 17

e=2

5) choose a random key k=8

STEP 3: IMPLEMENT ASYMMETRIC ENCRYPTION

CIPHERTEXT X:

Formulae: Y1=m ^k mod p

Y2 = X. e ^k mod p

STEP 4: IMPLEMENT ASYMMETRIC DECRYPTION

Formula: D (x)=y2 x (y1^Q) ^-1 mod p

STEP 5: IMPLEMENT SYMMETRIC DECRYPTION

1) Now perform right shift to the Asymmetric Decrypted 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.

NUMERICAL INSTANCE:

STEP 1: IMPLEMENT SYMMETRIC ENCRYPTION

1)*Enter the given message: 18*

Duplicate the same message and perform AND operation

Converting 18 into binary = 10010

Duplicate message same as original is 18= 10010

10010

AND

10010

10010

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

Output of previous step is 10010.

Let us consider symmetric key value :14 = 1110

10010

OR

01110

11110

3)*Now perform left shift of the previous output.*

Left shift performing: 11110

We get 11110 as the output (11110=30)

We consider this as the CIPHER TEXT.

CIPHER TEXT=30

STEP 2: ASYMMETRIC KEY GENERATION:

Public keys: M, E, P

Let us consider P be prime number p=19 and M ∈ P.

Let us take M=2 where m is a primitive element of mod P

Private key: Q

Choose Q secret 'Q' and compute E ≡M^ Q mod p

COMPUTING Q AND E

Choose a random Q =14

Compute e =m^ Q mod p

=3^14 mod 17

e =2

choose a random key k=4

STEP 3: IMPLEMENT ASYMMETRIC ENCRYPTION

ASYMMETRIC ENCRYPTION

ENTER THE CIPHER TEXT (SYMMETRIC ENCRYPTION) , X=30

Y1 =m ^k mod p

= 2^4mod 19

=16

Y2 = X. e ^k mod p

= 30 x 2^4 mod 19

=30 x 16

=480

STEP 4: IMPLEMENT ASYMMETRIC DECRYPTION

D (x)=y2 x (y1^Q) ^-1 mod p

= 480 x (16^14) ^-1 mod 19

= 480 x 5

D(X) =2400----------DECRYPTED MESSAGE

We get the cipher text of symmetric encryption.

STEP 5: IMPLEMENT SYMMETRIC DECRYPTION

1)Now perform right shift to the Asymmetric Decrypted text.

VALUE OF 2400 IN BINARY = 100101100000

RIGHT SHIFT OF 100101100000= 10010110000

After performing right shift we getting =10010110000

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

Output of previous step=10010110000

Symmetric private key=1110

10010110000

OR

00000001110

10010111110

We get 10010111110 as output after performing OR gate.

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

Output of previous step =10010111110

Duplicate message (same as original message) =10010

10010111110

AND

00000010010

00000010010

Now, we get the entered original message as decrypted value.

Decrypted message --------10010=18

Entered original message in symmetric encryption is 18.

CODING:

```
import java.io.*;
import java.util.*;
import java.math.*;

public class Test
{

    // 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 */
        System.out.println(" ");
        System.out.println(" ");
         System.out.println("STEP 1: IMPLEMENT SYMMETRIC ENCRYPTION:");
         System.out.println("-------------------------------------------");
         System.out.println("Symmetric Encryption:");
         System.out.println("-------------------------------------------");
         System.out.println(" ");
         System.out.print("Enter the message for symmetric encryption:");
         Scanner sc = new Scanner(System.in);
         BigInteger val1 = sc.nextBigInteger();
         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);
         int intValueOfc=c.intValue();
         System.out.println ("Ciphertext ct:" + c);
         System.out.println(" ");
         System.out.println("STEP 2: ASYMMETRIC KEY GENERATION:");
         System.out.println(" ");
         System.out.println("-------------------------------------------");
         System.out.println("Asymmetric Key Generation:");
         System.out.println("-------------------------------------------");
         System.out.println(" ");
         System.out.print("Enter the prime number:");
         int n=sc.nextInt();
         System.out.println("Smallest primitive root of " + n+ " is : " +
findPrimitive(n));
         int p=findPrimitive(n);
         System.out.print("Enter the random Q:");
         int q=sc.nextInt();
         System.out.println("Random Number is:"+q);
```

```
        int powerOfNumber = (int) Math.pow(p, q);
        int e=powerOfNumber % n;
        System.out.println("Public key E:"+e);
        System.out.print("Enter the random k:");
        int k=sc.nextInt();
        System.out.println("Random Number is:"+k);

       /**Asymmetric key Encryption */
       System.out.println(" ");
       System.out.println(" ");
        System.out.println("STEP 3: IMPLEMENT ASYMMETRIC ENCRYPTION:");

        System.out.println("-----------------------------------------");
        System.out.println("Asymmetric Key encryption:");
        System.out.println(" ");
        System.out.println("-----------------------------------------");
        int power=(int) Math.pow(p, k);
        int y1=power % n;
        System.out.println("Asymmetric encryption of Y1:"+y1);
        int power1=(int) Math.pow(e, k);
        int s1=power1 % n;
        int y2=intValueOfc*s1;
        System.out.println("Asymmetric encryption of Y2 :"+y2);

        /**Asymmetric Decryption */
        System.out.println(" ");
        System.out.println("STEP 4: IMPLEMENT ASYMMETRIC DECRYPTION:");
        System.out.println(" ");
        System.out.println("-----------------------------------------");
        System.out.println("Asymmetric Key Decryption:");
        System.out.println(" ");
        System.out.println("-----------------------------------------");


        int yy=(int) Math.pow(y1,q);
        long yu=yy;
        int yo=(int)yu;
        BigInteger biginteger1, biginteger2, result;
        biginteger1 = BigInteger.valueOf(yo);

        // Initialize all BigInteger Objects
        biginteger2 = BigInteger.valueOf(n);
        result =biginteger1.modInverse(biginteger2);
        int result1 = result.intValue();

       String sp = biginteger1 + " ^ -1 % "
                           + biginteger2 + " = " + result;
       // print result value
```

```
          int yi=result1 % n;

          int ym=y2*yi;
         // print result value
         System.out.println("ASYMMETRIC DECRYPTED MESSAGE:"+ym);
         System.out.println(" ");
         System.out.println("STEP 5: IMPLEMENT SYMMETRIC DECRYPTION");
         System.out.println("-----------------------------------------------");
         System.out.println("symmetric  Decryption:");
         System.out.println(" ");
         System.out.println("-----------------------------------------------");
         BigInteger dt1=BigInteger.valueOf(ym);
         BigInteger z=dt1.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("-----------------------------------------------");
         sc.close();

    }
}
```

OUTPUT:

```
STEP 1: IMPLEMENT SYMMETRIC ENCRYPTION:
-----------------------------------------
Symmetric Encryption:
-----------------------------------------

Enter the message for symmetric encryption:7
Enter the message for secret key:3
AND operation of message and duplicate message: 7
OR operation of previous step and key: 7
Left shift of previous step: 14
Ciphertext ct:14

STEP 2: ASYMMETRIC KEY GENERATION:

-----------------------------------------
Asymmetric Key Generation:
-----------------------------------------

Enter the prime number:19
Smallest primitive root of 19 is : 2
Enter the random Q:8
Random Number is:8
Public key E:9
Enter the random k:4
Random Number is:4


STEP 3: IMPLEMENT ASYMMETRIC ENCRYPTION:
-----------------------------------------
Asymmetric Key encryption:

-----------------------------------------
Asymmetric encryption of Y1:16
Asymmetric encryption of Y2 :84

STEP 4: IMPLEMENT ASYMMETRIC DECRYPTION:

-----------------------------------------
Asymmetric Key Decryption:

-----------------------------------------
ASYMMETRIC DECRYPTED MESSAGE:840
```

```
STEP 5: IMPLEMENT SYMMETRIC DECRYPTION
-----------------------------------------------
symmetric  Decryption:

-----------------------------------------------
Right shift of Asymmetric Decrypted text:420
OR operation of previous step and key:423
AND operation of previous step and duplicate message:7

-----------------------------------------------
Decrypted Original Message:7

-----------------------------------------------
PS C:\Users\HARISH>
```