|  |  |  |  |
| --- | --- | --- | --- |
| **Ex. No. 04** | **RSA** | | |
| Date of Exercise | 03 – 02 - 2015 | Date of Output Verification | 17 – 02 - 2015 |

**Question**

For a message size of 8-bits and a key size of 10-bits, write a java program to execute a

simplified DES algorithm for two rounds of iteration for given values:

Straight P box table = 3 5 2 7 4 10 1 9 8 6

Compression table P-box = 6 3 7 4 8 5 10 9

Straight P box permutation table = 2 4 3 1

Initial Permutation table = 2 6 3 1 4 8 5 7

Final permutation table = 4 1 3 5 7 2 8 6

Expansion P-box = 4 1 2 3 2 3 4 1

S0:

1 0 3 2

3 2 1 0

0 2 1 3

3 1 3 2

S1:

0 1 2 3

2 0 1 3

3 0 1 0

2 1 0 3

**Procedure**

1. Generate two large random primes, *p* and *q*, of approximately equal size such that their product n = pq is of the required bit length, e.g. 1024 bits.
2. Compute n = pq and (phi) φ = (p-1)(q-1).
3. Choose an integer *e*, 1 < e < phi, such that gcd(e, phi) = 1.
4. Compute the secret exponent *d*, 1 < d < phi, such that ed ≡ 1 (mod phi).
5. The public key is (n, e) and the private key (d, p, q). Keep all the values d, p, q and phi secret. [We prefer sometimes to write the private key as (n, d) because you need the value of n when using d. Other times we might write the key pair as ((N, e), d).]

* n is known as the *modulus*.
* e is known as the *public exponent* or *encryption exponent* or just the *exponent*.
* d is known as the *secret exponent* or *decryption exponent*.

**Program**

import java.math.BigInteger;

import java.util.Random;

import java.util.Scanner;

public class NetworkSecurityLabExp4RSA {

public static Scanner in = new Scanner(System.in);

public static BigInteger m, c, n, d, e;

public static int p = 17, q = 11, pin;

public static void main(String[] args) {

boolean r = true;

int choice;

System.out.println("AES - UR12CS135");

setkey();

do {

System.out.println("\n\n---------Main Menu---------\n1.Change Key \n2.Encode \n3.Decode \n4.Exit");

choice = in.nextInt();

in.nextLine();

switch (choice) {

case 1:

setkey();

break;

case 2:

callencode();

break;

case 3:

decode();

break;

case 4:

r = false;

break;

default:

System.out.println("Invalid Option!, Try Again...");

break;

}

} while (r == true);

}

public static void setkey() {

System.out.println("\n--------Set Key--------");

int y;

Random r = new Random();

BigInteger t = BigInteger.ONE;

getprimes();

n = BigInteger.valueOf(p \* q);

pin = (p - 1) \* (q - 1);

System.out.println("n : " + n);

System.out.println("p(n): " + pin);

do {

y = 2 + (int) (Math.random() \* (pin - 2));

System.out.println("e: " + y);

} while (1 != gcd(y, pin));

e = BigInteger.valueOf(y);

do {

t = t.add(BigInteger.valueOf(pin));

System.out.println("\nt : " + t);

System.out.println("d : " + t.divide(e));

System.out.println("mod: " + t.mod(e));

} while ((t.mod(e)) != BigInteger.ZERO);

d = t.divide(e);

System.out.println("d final: " + d);

System.out.println("e final: " + e);

}

public static int gcd(int a, int b) {

while (b > 0) {

int temp = b;

b = a % b;

a = temp;

}

return a;

}

public static void getprimes() {

do {

System.out.print("Enter Prime Number 1 (p):");

p = in.nextInt();

in.nextLine();

} while (!checkprime(p));

do {

System.out.print("Enter Prime Number 2 (q):");

q = in.nextInt();

in.nextLine();

} while (!checkprime(q));

}

public static boolean checkprime(int a) {

int r = 0;

if (a <= 1) {

return false;

}

for (int i = 2; i < a; i++) {

if (a % i == 0) {

return false;

}

}

return true;

}

public static void callencode() {

int a;

System.out.println("\n---------Encode--------\nEnter Plain Text");

String b = in.next();

for (int i = 0; i < b.length(); i++) {

a = (int) b.charAt(i);

encode(a);

}

}

public static void encode(int a) {

m = BigInteger.valueOf(a);

System.out.println("\n\*" + (char) a);

System.out.println("m: " + m);

System.out.println("e: " + e);

System.out.println("n: " + n);

c = m.modPow(e, n);

int i = c.intValue();

System.out.println("Cipher Text: " + c);

}

public static void decode() {

System.out.println("\n---------Decode--------\nEnter Cipher Text ASCII Value");

/\*

String b = in.next();

int a = (int) b.charAt(0);

c = BigInteger.valueOf(a);

\*/

c = in.nextBigInteger();

System.out.println("\nc: " + c);

System.out.println("d: " + d);

System.out.println("n: " + n);

m = c.modPow(d, n);

int i = m.intValue();

System.out.println("Plain Text(ASCII): " + m);

System.out.println("Plain Text(Char) : " + (char) i);

}

}

**Input**

P: 317

Q: 139

Plain Text: ‘H’

**Output**

AES - UR12CS135

--------Set Key--------

Enter Prime Number 1 (p):317

Enter Prime Number 2 (q):139

n : 44063

p(n): 43608

e: 13096

e: 17048

e: 35780

e: 11293

e: 20407

d final: 40783

e final: 20407

---------Main Menu---------

1.Change Key

2.Encode

3.Decode

4.Exit

2

---------Encode--------

Enter Plain Text

H

\*H

m: 72

e: 20407

n: 44063

Cipher Text: 12633

---------Main Menu---------

1.Change Key

2.Encode

3.Decode

4.Exit

3

---------Decode--------

Enter Cipher Text ASCII Value

12633

c: 12633

d: 40783

n: 44063

Plain Text(ASCII): 72

Plain Text(Char) : H

---------Main Menu---------

1.Change Key

2.Encode

3.Decode

4.Exit

4

BUILD SUCCESSFUL (total time: 43 seconds)

**Result**

RSA is successfully implemented in java.

[Signature of the Staff In-charge]

Name of the Staff In – charge: Mr. Manoj Kumar

Date: 17 – 02 - 2015