SCHOOL OF INFORMATION TECHNOLOGY AND ENGINEERING

INFORMATION SECURITY ANALYSIS AND AUDIT

REVIEW 2

LIGHTWEIGHT CRYPOGRAPHY USING BLIND SIGNATURE AND AUTHENTICATION

TEAM MEMBERS:

18MIS0163- D HARISH

18MIS0193-A R HARIHARAN

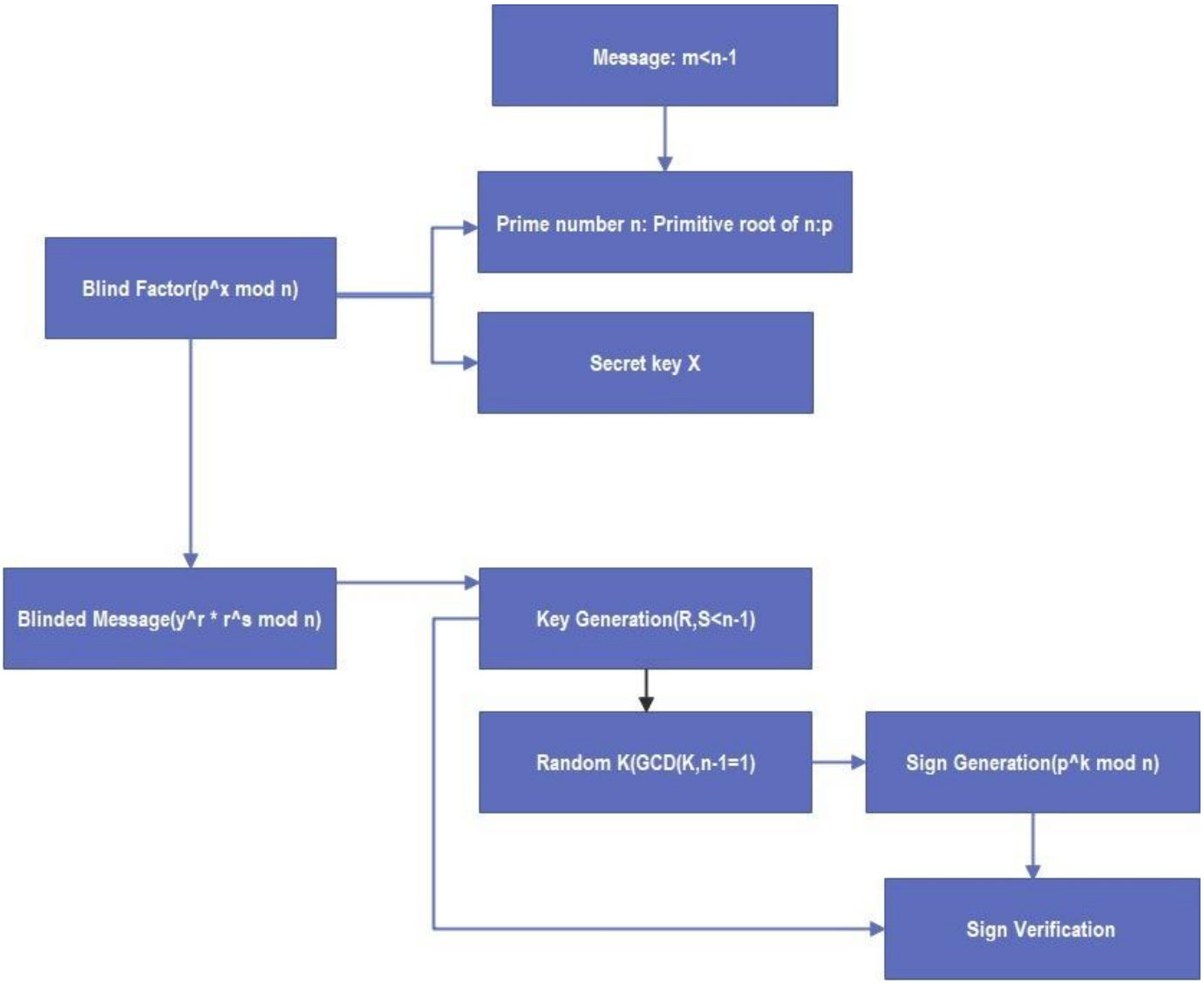
18MIS0260-R SUDHARSHAN

SLOT : F2

CODE : CSE3501

FACULTY:Prof.CHANDRASEGAR.T

ARCHITECTURE:



ALGORITHM:

- For each user, there is a key pair, which consists of a secret key x, and a public key y where,

  P is primitive root of the prime number;

  BLIND FACTOR: y = p^ x mod n.

- The public key y is published in a public file and known to everybody while the secret key x is kept secret.

- Let m, be a document to be signed, where: 0< m < n -1 and p is a prime.

- The public file consists of the public key y = p^ x mod n for each user.

- To sign a document, a user A uses the secret key X to compute a signature for, m so that any user can verify that this message has been signed by A, using the public key p together with n and p.

  KEY GENERATION:

- No one can forge a signature without knowing the secret xA.

- The signature for, m is a pair (r, s), where 0 < r, s c>n- 1, chosen such that

  BLINDED MESSAGE: BM = y^ r x r^ s mod n.

- The following three steps are done to compute the signature:

- Choose a random number k, uniformly distributed between 0 and n - 1, such that: can be written as gcd (k, n - 1) =1.

- **SIGN GENERATION:**

  Compute SG = p^ k mod n,

- **SIGN VERIFICATION:**

  SV=M^ x*sg x M1^k*s mod n

NUMERICAL INSTANCE:

- PRIME NUMBER n=19
- MESSAGE=14
- RANDOM X=13
- R=2(0<r<n-1)
- S=3(0<s<n-1)

Primitive root of the prime number(n) 19= p (2)

BLIND FACTOR:

y = p^ x mod n.

y=2^13 mod 19

y=3

Blinded Message:

BM = y^ r x r^ s mod n.

=3^2 x 2^3 mod 19

= 15

gcd (k, n - 1) =1

We find gcd(k,n-1)=1

We get 17 as k value.

Sign Generation:

Compute SG = p^ k mod n,

=2^17 mod 19

=10

Sign Verification:

SV1=M^ x*sg mod n =14^13*10 mod 19=17
Sv2= M1^k*s mod n=14^17*3 mod 19=12
Sv3= sv1*sv2 mod n
=17 x 12 mod 19
=204 mod 19
=14

We get the original message as the sign verification.

CODING:

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

public class Rev2
{

    // 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;
    }

/* Iterative Function to calculate (x^y) in O(log y) */
static int power(int h1, int h2, int h3)
{
  int res = 1; // Initialize result

  h1 = h1 % h3; // Update x if it is more than or
  // equal to p

  if (h1 == 0)
    return 0; // In case x is divisible by p;

  while (h2 > 0)
  {
```

```
    // If y is odd, multiply x with result
    if ((h2 & 1) != 0)
      res = (res * h1) % h3;

    // y must be even now
    h2 = h2 >> 1; // y = y/2
    h1 = (h1 * h1) % h3;
  }
  return res;
}
 public static void main(String[]args) {
   System.out.println(" ");
   Scanner sc = new Scanner(System.in);
   System.out.println(" KEY GENERATION:");
   System.out.println(" ");
   System.out.println("-----------------------------------------");
   System.out.println(" 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 Message :");
   int sh=sc.nextInt();
   System.out.println("Enter the random number X : ");
   int x=sc.nextInt();
   System.out.println("Random Number is:"+x);
   int powerOfNumber = (int) Math.pow(p, x);
   int e=powerOfNumber % n;
   System.out.println("-----------------------------------------");
   System.out.println("BLIND FACTOR :"+e);
   System.out.println(" ");

   System.out.println("-----------------------------------------");
   System.out.println(" ");
   System.out.println("Random R between 0<r>P-1 : ");
   int r=sc.nextInt();
   System.out.println("Random S between 0<s>P-1 : ");
   int s=sc.nextInt();
   int power1=(int) Math.pow(e,r);
   int power2=(int) Math.pow(r,s);
   int bm=power1*power2 % n;
   System.out.println("-----------------------------------------");
   System.out.println(" ");
```

```
    System.out.print("BLINDED MESSAGE :"+bm);
    System.out.println(" ");

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

    int ok=n-1;
   System.out.println("P-1 :"+ok);
   System.out.print("Enter the random k:");
    int k=ok-1;
    System.out.println("K MUST BE GCD (K,P-1 = 1)");
    System.out.println("Random Number K is:"+k);

    System.out.println(" ");
    System.out.println("-------------------------------------------");
    int power3=(int) Math.pow(p,k);
    int sg=power3 % n;
    System.out.println("SIGN GENERATION :"+sg );
    System.out.println(" ");
    System.out.println("-------------------------------------------");
    int l1=power(sh,x*sg,n);

    int l2=power(sh,k*s,n);

    int l3=(l1*l2) % n;
    System.out.println(" ");

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

   System.out.println("SIGN VERIFICATION :"+l3);
   System.out.println(" ");

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



  }
}
```

SAMPLE OUTPUT:

```
 KEY GENERATION:

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

Enter the prime number:17
Smallest primitive root of 17 is : 3
Enter the Message :3
Enter the random number X :
14
Random Number is:14
-------------------------------------------
BLIND FACTOR :2

-------------------------------------------

Random R between 0<r>P-1 :
2
Random S between 0<s>P-1 :
3
-------------------------------------------

BLINDED MESSAGE :15
-------------------------------------------
```

```
P-1 :16
Enter the random k:K MUST BE GCD (K,P-1 = 1)
Random Number K is:15

-------------------------------------------
SIGN GENERATION :6

-------------------------------------------

-------------------------------------------
SIGN VERIFICATION :3

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