Blockchain Technology

Practical 1

EL5

19BCE248

AIM: To implement digital signature to sign and verify authenticated user. Also, show a message when tampering is detected.

Simple Implementation of RSA with Integer message

```
import java.util.*;
import java.io.*;
public class Prac1_Rsa
{
public static void main(String[] args) {
ArrayList<Integer> prime =genPrime(20);
Random random=new Random();
int p=prime.get(random.nextInt(prime.size()));
int q=p;
while(p==q){
   q=prime.get(random.nextInt(prime.size()));
}
int n=p*q;
int phi_n=(p-1)*(q-1);
int e=-1;
for(int i=2;i<phi_n;i++){
   if(gcd(i,phi_n)==1){
       e=i;
       break;
   }
}
// (e*d)%phi_n=1;
int d=1;
while((d==p) || (d==q) || (e*d)%phi_n!=1 ){
   d++;
}
int message=5;
long encrypt=((long)Math.pow((message),e))%n;
long decrypt=((long)Math.pow((encrypt),d))%n;
if(decrypt==message){
   System.out.println("Correct!!");
```

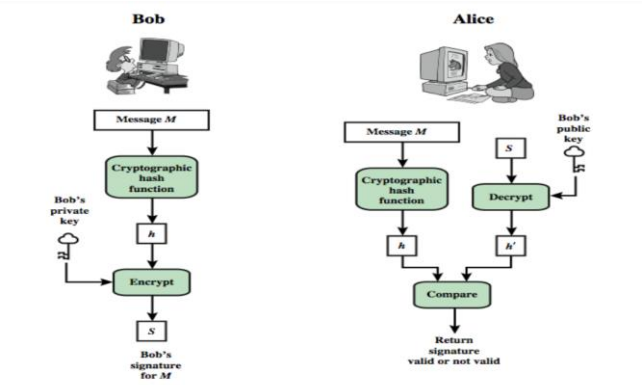
```
}else{
   System.out.println("Sorry Incorrect!!");
}
}
public static int gcd(int a,int b){
   if(b==0){
       return a;
   }
   return gcd(b,a%b);
}
public static ArrayList<Integer> genPrime(int n){
   ArrayList<Integer> primeList =new ArrayList<Integer>();
   boolean prime[] = new boolean[n+1];
        for(int i=0;i<=n;i++)
            prime[i] = true;

        for(int p = 2; p*p <=n; p++)
        {
            if(prime[p] == true)
            {
                for(int i = p*p; i <= n; i += p)
                    prime[i] = false;
            }
        }
        for(int i = 2; i <= n; i++)
        {
            if(prime[i])
            primeList.add(i);
        }
        return primeList;

}
}
```

Output:

```
PS D:\SEM 7\BCT\Lab> cd "d:\SEM 7\BCT\Lab\" ; if ($?) { javac Prac1_Rsa.java } ; if ($?) { java Prac1_Rsa
 }
Correct!!
```

Python Code for Implementation of RSA with inbuilt python library:

Below shows diagrammatic way of how code is implemented:




```
from Crypto.PublicKey import RSA
from Crypto.Signature.pkcs1_15 import PKCS115_SigScheme
from Crypto.Hash import SHA256
import binascii
keyPair = RSA.generate(bits=1024)
pubKey = keyPair.publickey()
# Can't access private key as it would be of no use if can be access by anyone
hasPrivateKey=keyPair.has_private()
print(pubKey)
print(hasPrivateKey)
msg = b'Message to be send to other side without any tampering'
hash = SHA256.new(msg)
signer = PKCS115_SigScheme(keyPair)
signature = signer.sign(hash)
print("Signature:", binascii.hexlify(signature))
msg = b'Message to be send to other side without any tampering'
hash = SHA256.new(msg)
verifier = PKCS115_SigScheme(pubKey)
try:
    verifier.verify(hash, signature)
    print("Signature is valid.")
except:
    print("Signature is invalid.")
```

```
Signature is valid.
```

```
msg = b'A tampered message for veryfying'
hash = SHA256.new(msg)
verifier = PKCS115_SigScheme(pubKey)
try:
```

```
    verifier.verify(hash, signature)
    print("Signature is valid.")
except:
    print("Signature is invalid.")
```

```
Signature is invalid.
```

Learning Outcome:

From these practical we had a clear idea of working of RSA as we implemented it from scratch for a very basic understanding. Apart from that we also had done with few python libraries which makes it fully functional for any datatype to be passed as a message through medium. Here we explored different new libraries which are built on purpose to make complex security application without worrying about primary implementation.