Lab-05

Page No.: 1
Date: 20/1/2022

AIM:- Write a program to implement Knapsack Cryptosystem: Key Generation, Encryption, Decryption.

- code:-

```
#include<bits/stdc++.h>
#include "functions.h"   #has function like Extended Euclidian
unsigned long long m,w;

vector<int> generateSuperIncreasingArray(int size)
{
    vector<int> array(size);
    array[0] = 1+ rand()%5;
    srand(0); // to reintialize rand().
    array[1] = array[0] + (1+rand()%5);
    int nextNum = array[0] + array[1]

    for(int i=2; i<size; ++i)
    {
        srand(time(0));
        array[i] = nextNum+1;
        nextNum += array[i];
    }
    return array;
}
```

```
string convertToBinary(int msg)
{
    bitset<16> binaryNum(msg);
    string temp, binary = "";

    temp = binaryNum.to_string();

// remove extra zero from 16 bit binary num.
    int i;
    for(int i = 0; i < temp.length(); ++i)
    {
        if(temp[i] == '1')
            break;
    }

    binary = temp.substr(i, temp.length());
    return binary;
}

int power(int num, int pow)
{
    int (pow == 0)
        return 1;
    else
        return (num * power(num, pow-1));
}
```

```
Pair< vector<int>, vector<int>>

    keyGeneration(vector<int> -a)
{
    srand(time(0));
    m = accumulate(-a.begin(), -a.end(), 0
            1+(rand()%mg);
    Vector<int> a(-a.size());
    for(int i=2; i<m; ++i)
    {
        if( gcdUsingEEA( i,m) == 1)
        {
            w = i;
            break;
        }
    }

    for(int i=0; i<-a.size; ++i)
    {
        a[i] = (w* -a[i]) %m;
    }

    return make_Pair(a, -a);
}
```

```
int encryption (vector<int> key, string msg)
{
    unsigned int encrypted Msg = 0;

    for (int i=0; i< msg.length(); ++i)
    {
        if (msg[i] == '1')
            encrypted Msg += key[i];
    }
    return encrypted Msg;
}

string decryption( vector<int> key, int encryptedMsg)
{
    int sum = (multiplicative Inverse(w, m) *
                encrypted Msg ) %. m;

    cout<<"sum: " << sum << endl;
    char decrypted Binary [key.size() +1];
    decrypted Binary [key.size()] = '\0';

    for(int i = keysize() -1 ; i>=0; --i)
    {
        if (sum >= key [i])
        {
            decrypted Binary [i] = '1';
            sum -= key[i];
        }
        else
            decrypted Binary [i] = '0';
    }
```

```
    return decryptedBinary;
}

void print(vector<int> arr)
{
    for(int i=0; i<arr.size(); ++i)
        cout<<arr[i]<<" ";

    cout<<endl;
}

unsigned int convertToDecimal(string decryptedBinary)
{
    int num=0;
    int j=0;
    for(int i=decryptedBinary.length()-1;
                    i>=0; --i--)
    {
        if(decryptedBinary[i]=='1')
        {
            num += Power(2,j++);
        }
        else if(decryptedBinary[i]=='0')
            j++;
    }
    return num;
}
```

```
int main()
{
    int msg;
    cout << "Enter your msg: ";
    cin >> msg;

    if( msg > 65535 )
    {
        cout << "Number limit is exicided " << endl;
        exit(0);
    }

    String binaryMsg = convertToBinary(msg);

    int size = binaryMsg.length();
    cout << "Binary: " << binaryMsg << endl;
    vector<int> a = generateSuperIncreasingArray
                                        (size);
    pair <vector<int>, vector<int>> keys;

    keys = keyGeneration(a);
    cout << "m: " << m << "\tw: " << w << endl;
    vector<int> publicKey, privateKey;

    publicKey = keys.first;
    privateKey = keys.second;
    cout << privateKey: ";
    print(privateKey);
    cout << "Public key: ";
    print(publicKey);
```

```
int encryptedMsg = encryption(PublicKey, binaryMsg);
cout << "Encrypted msg: " << encryptedMsg << endl;

String decryptedBinary = decryption(PrivateKey,
                                    encryptedMsg);

cout << "Decrypted Binary: " << decryptedBinary
                             << endl;
cout << "Decrypted Msg: "
     << convertDecimalToBinary(decryptedBinary)
     << endl;
return 0;
}
```

running code:-

In: Enter your msg: 3425

out: Binary: 110101100001

m: 6144 w: 5

Private key: 2 3 6 12 24 48 96 192 384 768 1536 3072

Public Key: 10 15 30 60 120 240 480 960 1920 3840 1536 3072

Encrypted msg: 3877

Sum: 3233

Decrypted Binary: 1101 0110 0001

Decrypted msg: 3425: