Kpi-best

МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМ. І.СІКОРСЬКОГО»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

**Лабораторна робота №3**

з дисципліни «Безпека програм і даних»

Виконав:

студент 4-го курсу

факультету ІОТ

групи ІП-54

Макаренко Антон

Перевірив:

Марковський О.П.

Київ 2018

**Завдання:**

**Програма генерації 16-ти розрядних ключів для алгоритму Ель-Гамаля**

**Лістинг:**

**ElGamalAlgorithm.cs:**

using System;

namespace Lab\_3

{

public static class ElGamalAlgorithm

{

public static ElGamalKey GenerateKeys(ushort p = 0)

{

var random = new Random();

if (p < 2) {

p = MathUtils.GetFirstPrimeNumber((ushort)random.Next(ushort.MaxValue));

}

ushort x = (ushort)random.Next(2, p);

ushort g = MathUtils.GetRandomPremetiveRoot(p);

ushort y = (ushort)MathUtils.ModularExponentiation(g, x, p);

return new ElGamalKey

{

X = x,

Y = y,

G = g,

P = p

};

}

}

}

**ElGamalKey.cs:**

using System;

namespace Lab\_3

{

public class ElGamalKey

{

public ushort X { get; set; }

public ushort Y { get; set; }

public ushort P { get; set; }

public ushort G { get; set; }

}

}

**MathUtils.cs:**

using System;

using System.Collections.Generic;

namespace Lab\_3

{

public static class MathUtils

{

public static ushort GetRandomPremetiveRoot(ushort p)

{

var list = new List<ushort>();

int phi = p - 1;

var primeFactors = FindPrimeFactors(phi);

for (var r = 2; r <= phi; r++) {

bool flag = false;

for (var i = 0; i < primeFactors.Count; i++) {

if (ModularExponentiation(r, phi / primeFactors[i], p) == 1) {

flag = true;

break;

}

}

if (flag == false)

list.Add((ushort) r);

}

var random = new Random();

var randomIndex = random.Next(list.Count);

return list[randomIndex];

}

public static List<int> FindPrimeFactors(int p)

{

var result = new List<int>();

while (p % 2 == 0) {

result.Add(2);

p /= 2;

}

for (ushort i = 3; i <= Math.Sqrt(p); i += 2) {

while (p % i == 0) {

result.Add(i);

p /= i;

}

}

if (p > 2)

result.Add(p);

return result;

}

private static bool IsPrime(int number)

{

if (number <= 1) return false;

if (number == 2) return true;

if (number % 2 == 0) return false;

var boundary = (int) Math.Floor(Math.Sqrt(number));

for (int i = 3; i <= boundary; i += 2) {

if (number % i == 0) return false;

}

return true;

}

public static ushort GetFirstPrimeNumber(ushort value = 2, int maxNumber = ushort.MaxValue)

{

ushort number = value;

while (!IsPrime(number)) {

number++;

}

return number;

}

public static int ModularExponentiation(int x, int y, int p)

{

int res = 1;

x = x % p;

while (y > 0) {

if (y % 2 == 1)

res = (res \* x) % p;

y = y >> 1;

x = (x \* x) % p;

}

return res;

}

}

}

**Program.cs:**

using System;

namespace Lab\_3

{

class Program

{

static void Main(string[] args)

{

var key = ElGamalAlgorithm.GenerateKeys();

Console.WriteLine("p: " + key.P);

Console.WriteLine("g: " + key.G);

Console.WriteLine("x: " + key.X);

Console.WriteLine("y: " + key.Y);

}

}

}