МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**ОТЧЁТ**

**по лабораторной работе № 1**

**« Введение в ООП »**

**по дисциплине: «***Программирование***»**

Выполнил:Проверил:

Студент гр. «АБ-121», «АВТФ» *доцент кафедры ЗИ*

*Новиков Ян Валерьевич Архипова А. Б.*

«21» сентября 2022г«\_\_\_» \_\_\_\_\_\_ 2022 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2022

**Цели и задачи работы**: изучение основных принципов объектно-ориентированного программирования.

**Задание к работе**: Самостоятельно решить задачи в соответствии с индивидуальным вариантом на двух языках программирования высокого уровня (С++ и другой по выбору).

Реализовать криптографические протоколы Шамира, Диффи-Хеллмана, Эль-Гамаля, RSA с использованием классов для хранения данных.

**Код на С++**

#include <iostream>

#include <vector>

#include <random>

#include <ctime>

using namespace std;

struct RGEN {

int genrsa() {

minstd\_rand0 gen(time(NULL));

uniform\_int\_distribution<int> dis1(50, 200);

int num1 = dis1(gen);

return num1;

}

int gensh() {

knuth\_b gen(time(NULL));

uniform\_int\_distribution<int> dis2(50, 200);

int num2 = dis2(gen);

return num2;

}

int gendh() {

mt19937 gen(time(NULL));

uniform\_int\_distribution<int> dis3(50, 200);

int num3 = dis3(gen);

return num3;

}

int genalg() {

ranlux24 gen(time(NULL));

uniform\_int\_distribution<int> dis4(100, 200);

int num4 = dis4(gen);

return num4;

}

};

struct DH {

public:

int p;

int g;

int ya;

int yb;

int Getxa() {

return xa;

}

int Getxb() {

return xb;

}

void Setxa(int t) {

xa = t;

}

void Setxb(int t) {

xb = t;

}

private:

int xa;

int xb;

};

struct SH {

public:

int p;

int m;

int x1;

int x2;

int x3;

int x4;

int Getca() {

return ca;

}

int Getda() {

return da;

}

int Getcb() {

return cb;

}

int Getdb() {

return db;

}

void Setca(int t) {

ca = t;

}

void Setda(int t) {

da = t;

}

void Setcb(int t) {

cb = t;

}

void Setdb(int t) {

db = t;

}

private:

int ca;

int da;

int cb;

int db;

};

struct AlGam {

public:

int p;

int g;

int Y;

int A;

int B;

int k;

int M;

int GetX() {

return X;

}

void SetX(int inpX) {

X = inpX;

}

private:

int X;

};

struct DeAlGam {

public:

int p;

int r;

int e;

int Y;

int M1;

int GetX() {

return X;

}

void SetX(int inpX) {

X = inpX;

}

private:

int X;

};

struct RSA {

public:

int Na;

int Ya;

int m;

int e;

int Getpa()

{

return pa;

}

int Getqa()

{

return qa;

}

int GetXa()

{

return Xa;

}

void Setpa(int ipa)

{

pa = ipa;

}

void Setqa(int iqa)

{

qa = iqa;

}

void SetXa(int iXa)

{

Xa = iXa;

}

private:

int pa;

int qa;

int Xa;

};

struct DeRSA {

public:

int N;

int GetX()

{

return X;

}

void SetX(int iX)

{

X = iX;

}

int e;

int m1;

private:

int X;

};

bool neprostnum(int num)

{

if (num < 3)

return true;

int count = 0;

for (int i = 2; i < sqrt(num); i++)

{

if (num % i == 0)

{

return true;

}

}

return false;

}

int ost(int g, int x, int p) {

int res = 1;

for (int i = x; i != 0; i--)

{

res = res \* g;

res %= p;

}

return res;

}

void vdh()

{

DH dh;

RGEN rgen1;

neprost:

dh.p = rgen1.gendh();

if (neprostnum(dh.p) == false)

{

goto neprost;

}

dh.g = 3;

dh.Setxa(15);

dh.Setxb(13);

dh.ya = ost(dh.g, dh.Getxa(), dh.p);

dh.yb = ost(dh.g, dh.Getxb(), dh.p);

int zab = ost(dh.yb, dh.Getxa(), dh.p);

int zba = ost(dh.ya, dh.Getxb(), dh.p);

cout << "=================== Diffie–Hellman" << endl;

cout << "Ключ у первого равен " << zab << endl;

cout << "Ключ у второго равен " << zba << endl;

}

void swapAtoB(vector<int>& a, vector<int>& b) {

for (int i = 0; i < a.size(); i++) {

a[i] = b[i];

}

}

int genEva(int m, int c) { // cd mod m = 1

vector <int> U = { m, 1, 0 };

vector <int> V = { c, 0, 1 };

vector <int> T = { 0, 0, 0 };

int q;

while (V[0] != 0) {

q = U[0] / V[0];

T[0] = U[0] % V[0];

T[1] = U[1] - q \* V[1];

T[2] = U[2] - q \* V[2];

swapAtoB(U, V);

swapAtoB(V, T);

}

int NOD = U[0], xx = U[1], yy = U[2];

if (yy < 0)

yy += m;

return yy;

}

void vsh()

{

SH sh;

RGEN rgen2;

neprost:

sh.p = rgen2.gensh();

if (neprostnum(sh.p))

{

goto neprost;

}

sh.m = 10;

sh.Setca(7);

sh.Setda(genEva((sh.p - 1), sh.Getca()));

sh.Setcb(19);

sh.Setdb(genEva((sh.p - 1), sh.Getcb()));

sh.x1 = ost(sh.m, sh.Getca(), sh.p);

sh.x2 = ost(sh.x1, sh.Getcb(), sh.p);

sh.x3 = ost(sh.x2, sh.Getda(), sh.p);

sh.x4 = ost(sh.x3, sh.Getdb(), sh.p);

cout << "=================== Shamir" << endl;

cout << "Изначальное сообщение: " << sh.m << endl;

cout << "Полученное сообщение: " << sh.x4 << endl;

}

int specost(int g, int x, int p, int M)

{

int res = M;

for (int i = x; i != 0; i--)

{

res = res \* g;

res %= p;

}

return res;

}

void valg()

{

AlGam ag;

DeAlGam deag;

RGEN rgen3;

neprost:

ag.p = rgen3.genalg();

if (neprostnum(ag.p))

{

goto neprost;

}

ag.g = 342;

ag.SetX(124);

ag.Y = ost(ag.g, ag.GetX(), ag.p);

ag.M = 23;

ag.k = 43;

ag.A = ost(ag.g, ag.k, ag.p);

deag.r = ag.A;

ag.B = specost(ag.Y, ag.k, ag.p, ag.M);

deag.e = ag.B;

deag.p = ag.p;

deag.SetX(ag.GetX());

deag.Y = ag.Y;

deag.M1 = specost(deag.r, (deag.p - 1 - deag.GetX()), deag.p, deag.e);

cout << "=================== El Gamal" << endl;

cout << "Исходное сообщение: " << ag.M << endl;

cout << "Полученное сообщение: " << deag.M1 << endl;

}

void vrsa()

{

RSA rsa;

DeRSA dersa;

RGEN rgen4;

neprost1:

rsa.Setpa(rgen4.genrsa());

if (neprostnum(rsa.Getpa()))

{

goto neprost1;

}

neprost2:

rsa.Setqa(rgen4.genalg());

if (neprostnum(rsa.Getqa()))

{

goto neprost2;

}

rsa.Na = rsa.Getpa() \* rsa.Getqa();

rsa.Ya = 121;

if (((rsa.Getpa() - 1) \* (rsa.Getqa() - 1)) % rsa.Ya == 0)

{

rsa.Ya = rsa.Ya + 1;

}

rsa.m = 88;

rsa.e = ost(rsa.m, rsa.Ya, rsa.Na);

dersa.e = rsa.e;

dersa.N = rsa.Na;

rsa.SetXa(genEva((rsa.Getpa() - 1) \* (rsa.Getqa() - 1), rsa.Ya));

dersa.SetX(rsa.GetXa());

dersa.m1 = ost(dersa.e, dersa.GetX(), dersa.N);

cout << "=================== RSA" << endl;

cout << "Исходное сообщение: " << rsa.m << endl;

cout << "Полученное сообщение: " << dersa.m1 << endl;

}

int main()

{

setlocale(LC\_ALL, "Rus");

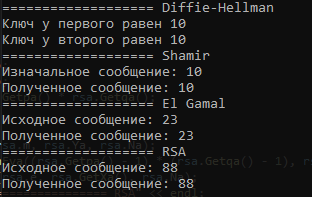
//vdh();

//vsh();

//valg();

vrsa();

}

****

**Код на С#**

using System;

namespace ConsoleApp1

{

class DH

{

public int p;

public int g;

public int ya;

public int yb;

private int xa;

private int xb;

public int Getxa()

{

return xa;

}

public int Getxb()

{

return xb;

}

public void Setxa(int t)

{

xa = t;

}

public void Setxb(int t)

{

xb = t;

}

}

class AlGam

{

public int p;

public int g;

public int Y;

public int A;

public int B;

public int k;

public int M;

public int M1;

public int GetX()

{

return X;

}

public void SetX(int inpX)

{

X = inpX;

}

private int X;

}

class SH

{

public int p;

public int m;

public int x1;

public int x2;

public int x3;

public int x4;

public int Getca()

{

return ca;

}

public int Getda()

{

return da;

}

public int Getcb()

{

return cb;

}

public int Getdb()

{

return db;

}

public void Setca(int t)

{

ca = t;

}

public void Setda(int t)

{

da = t;

}

public void Setcb(int t)

{

cb = t;

}

public void Setdb(int t)

{

db = t;

}

private int ca;

private int da;

private int cb;

private int db;

};

class RSA

{

public int Na;

public int Ya;

public int m;

public int e;

public int m1;

public int Getpa()

{

return pa;

}

public int Getqa()

{

return qa;

}

public int GetXa()

{

return Xa;

}

public void Setpa(int ipa)

{

pa = ipa;

}

public void Setqa(int iqa)

{

qa = iqa;

}

public void SetXa(int iXa)

{

Xa = iXa;

}

private int pa;

private int qa;

private int Xa;

};

class Program

{

static int ost(int g, int x, int p)

{

int res = 1;

for (int i = x; i != 0; i--)

{

res \*= g;

res %= p;

}

return res;

}

static void vdh()

{

DH dh = new DH();

dh.p = 17;

dh.g = 3;

dh.Setxa(15);

dh.Setxb(13);

dh.ya = ost(dh.g, dh.Getxa(), dh.p);

dh.yb = ost(dh.g, dh.Getxb(), dh.p);

int zab = ost(dh.yb, dh.Getxa(), dh.p);

int zba = ost(dh.ya, dh.Getxb(), dh.p);

Console.WriteLine("=================== Diffie–Hellman");

Console.WriteLine($"First key = {zab}");

Console.WriteLine($"Second key = {zba}");

}

static int specost(int g, int x, int p, int M)

{

int res = M;

for (int i = x; i != 0; i--)

{

res \*= g;

res %= p;

}

return res;

}

static void valg()

{

AlGam ag = new AlGam();

ag.p = 997;

ag.g = 342;

ag.SetX(124);

ag.Y = ost(ag.g, ag.GetX(), ag.p);

ag.M = 23;

ag.k = 43;

ag.A = ost(ag.g, ag.k, ag.p);

ag.B = specost(ag.Y, ag.k, ag.p, ag.M);

ag.M1 = specost(ag.A, (ag.p - 1 - ag.GetX()), ag.p, ag.B);

Console.WriteLine("=================== El Gamal");

Console.WriteLine($"Input = {ag.M}");

Console.WriteLine($"Output = {ag.M1}");

}

static void swapAtoB(ref int[] a, ref int[] b)

{

for (int i = 0; i < 3; i++)

{

a[i] = b[i];

}

}

static int genEva(int m, int c)

{ // cd mod m = 1

int[] U = { m, 1, 0 };

int[] V = { c, 0, 1 };

int[] T = { 0, 0, 0 };

int q;

while (V[0] != 0)

{

q = U[0] / V[0];

T[0] = U[0] % V[0];

T[1] = U[1] - q \* V[1];

T[2] = U[2] - q \* V[2];

swapAtoB(ref U, ref V);

swapAtoB(ref V, ref T);

}

int NOD = U[0], xx = U[1], yy = U[2];

if (yy < 0)

yy += m;

return yy;

}

static void vsh()

{

SH sh = new SH();

sh.p = 23;

sh.m = 10;

sh.Setca(7);

sh.Setda(genEva((sh.p - 1), sh.Getca()));

sh.Setcb(5);

sh.Setdb(genEva((sh.p - 1), sh.Getcb()));

sh.x1 = ost(sh.m, sh.Getca(), sh.p);

sh.x2 = ost(sh.x1, sh.Getcb(), sh.p);

sh.x3 = ost(sh.x2, sh.Getda(), sh.p);

sh.x4 = ost(sh.x3, sh.Getdb(), sh.p);

Console.WriteLine("=================== Shamir");

Console.WriteLine($"Input = {sh.m}");

Console.WriteLine($"Output = {sh.x4}");

}

static void vrsa()

{

RSA rsa = new RSA();

rsa.Setpa(17);

rsa.Setqa(11);

rsa.Na = rsa.Getpa() \* rsa.Getqa();

rsa.Ya = 7;

rsa.m = 88;

rsa.e = ost(rsa.m, rsa.Ya, rsa.Na);

rsa.SetXa(genEva((rsa.Getpa() - 1) \* (rsa.Getqa() - 1), rsa.Ya));

rsa.m1 = ost(rsa.e, rsa.GetXa(), rsa.Na);

Console.WriteLine("=================== RSA");

Console.WriteLine($"Input = {rsa.m}");

Console.WriteLine($"Output = {rsa.m1}");

}

static void Main(string[] args)

{

vdh();

valg();

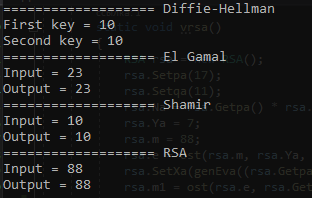
vsh();

vrsa();

}

}

}



**Вывод**

В ходе проделанной работы были реализованы криптографические протоколы Шамира, Диффи-Хеллмана, Эль-Гамаля, RSA с использованием классов для хранения данных на двух языках программирования: С++ и C#. Также была изучена и освоена теоретическая часть.