Министерство образования Республики Беларусь  
Учреждение образования  
«Брестский государственный технический университет»  
Кафедра ИИТ

Лабораторная работа №4  
По дисциплине: «Криптографические методы защиты информации»

Тема: «Эллиптические кривые»

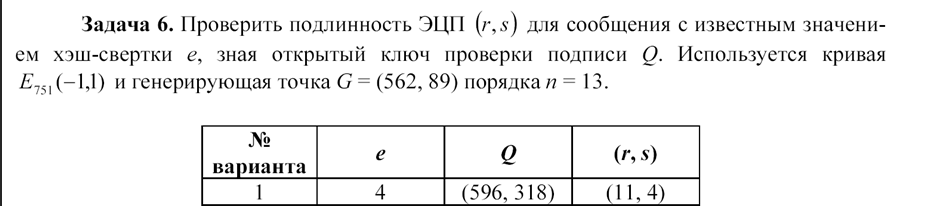
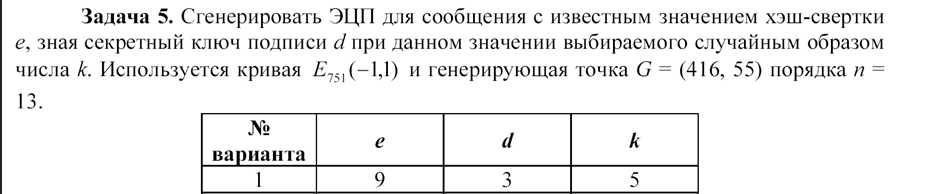
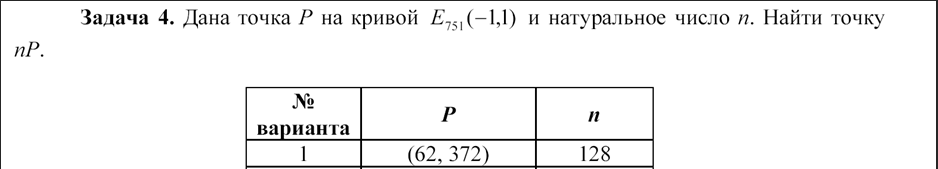
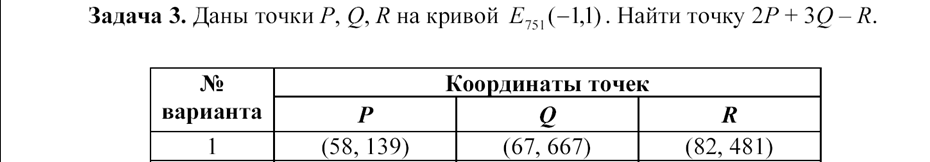
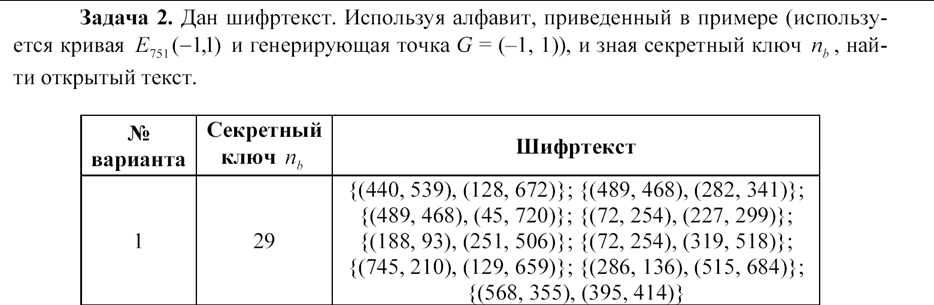
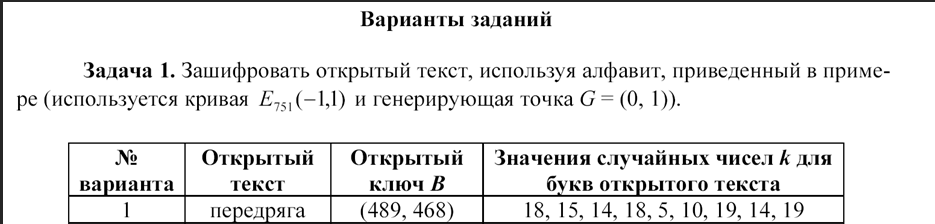
Выполнил:  
студент 3 курса  
группы ИИ-23  
Романюк А. П.

Проверил:  
Хацкевич А. С.

Брест 2024

Цель работы: разработать программу для упражнений на эллиптических кривых.

Вариант 9



Реализация:

#include <iostream>

#include <vector>

#include <map>

#include <cmath>

#include <random>

#include <windows.h>

#include <locale>

#include <string>

using namespace std;

class ElipticCurves {

int p = 751;

int a = -1;

int b = 1;

pair<int, int> G = { 0,1 };

pair<int, int> Gq23 = { -1,1 };

pair<int, int> G5 = { 416,55 };

int n = 13;

pair<int, int> G6 = { 562,89 };

pair<int, int> GTest = { 384,475 };

map<int, pair<int, int>> alphabet = {

{' ', {33, 355}},

{'!', {33, 396}},

……………………

};

public:

int positiveMod(int a, int m) {

return (a % m + m) % m;

}

int extendEucled(int a, int b, int &x, int &y) {

if (a == 0) {

x = 0;

y = 1;

return b;

}

int x1, y1;

int gcd = extendEucled(positiveMod(b,a), a, x1, y1);

x = y1 - (b / a) \* x1;

y = x1;

return gcd;

}

int gcd(int a, int m) {

if (a == 0) {

return 0;

}

int gcd, x, y;

gcd = extendEucled(a, m, x, y);

if (gcd != 1) {

cout << "Error gcd\n";

return 0;

}

return positiveMod(x, m);

}

pair<int, int> doubleAdditing(pair<int, int> P, pair<int, int> Q) {

if (P.first == -1) return Q;//maybe error

if (Q.first == -1) return P;

int x1 = P.first, y1 = P.second;

int x2 = Q.first, y2 = Q.second;

int lm;

if (P != Q) {

int denom1 = positiveMod((y2 - y1), p);

int denom2 = positiveMod((x2 - x1), p);

denom2 = gcd(denom2, p);

lm = positiveMod((denom1 \* denom2), p);

}

else {

int denom1 = positiveMod((3 \* x1 \* x1 + a), p);

int denom2 = positiveMod((2 \* y1), p);

denom2 = gcd(denom2, p);

lm = positiveMod((denom1 \* denom2), p);

}

int x3 = positiveMod((lm \* lm - x1 - x2), p);

int y3 = positiveMod((lm \* (x1 - x3) - y1), p);

return { x3,y3 };

}

int bit\_length(unsigned int a) {

return a > 0 ? static\_cast<int>(std::log2(a)) + 1 : 0;

}

pair<int, int> multiplyPoint(int k, pair<int, int> P) {

pair<int, int> Q = {-1,-1};

for (int i = 0; i < bit\_length(k); i++) {

if ((k >> i) & 1) {

Q = doubleAdditing(P, Q);

}

P = doubleAdditing(P, P);

}

return Q;

}

pair<pair<int, int>, pair<int, int>> encryptChar(int k, pair<int, int> Pb, char \_char) {

pair<int, int> C1 = multiplyPoint(k, G);

pair<int, int> C2 = multiplyPoint(k, Pb);

C2 = doubleAdditing(C2, alphabet[\_char]);

return { C1,C2 };

}

void encryptText(vector<int> k, pair<int,int> Pb, string plaintext) {

vector<pair<pair<int, int>, pair<int, int>>> answer;

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

answer.push\_back(encryptChar(k[i], Pb, plaintext[i]));

cout <<"x: (" << answer[i].first.first << "," << answer[i].first.second << ")k \* G, y: (" << answer[i].second.first << "," << answer[i].second.second << ")\n";

}

cout << "\n";

}

pair<int, int> reverseSign(pair<int, int> C) {

int x = C.first;

int y = -1 \* C.second;

return {x,y};

}

int findKeyByValue(const std::pair<int, int>& value) {

for (const auto& pair : alphabet) {

if (pair.second == value) {

return pair.first;

}

}

return '\0';

}

wchar\_t decryptChar(pair<pair<int, int>, pair<int, int>> C, int nb) {

pair<int, int> C\_ = multiplyPoint(nb, C.first);

pair<int, int> C\_m = reverseSign(C\_);

pair<int, int> C\_a = doubleAdditing(C.second, C\_m);

int char\_ = findKeyByValue(C\_a);

if (!char\_) {

wcout << L"Error char index\n";

return 0;

}

return static\_cast<wchar\_t>(char\_);

}

wstring decryptText(int nb, vector<pair<pair<int, int>, pair<int, int>>> ciphertext) {

wstring decryptedText = L"";

for (auto C : ciphertext) {

wchar\_t char\_ = decryptChar(C, nb);

if (!char\_) {

wcout << L"error find char";

return L"";

}

decryptedText += char\_;

}

wcout << decryptedText;

return decryptedText;

}

pair<int, int> signatureGenerate(int k, int e, int d) {

pair<int, int> kG = multiplyPoint(k, G5);

int r = positiveMod(kG.first, n);

int z = gcd(k, n);

int s = positiveMod(z \* (e + d \* r), n);

cout << "r: " << r << " s: " << s << endl;

return { r,s };

}

bool signatureVerification(int e, pair<int, int> rs, pair<int, int> Q) {

if (1 <= rs.first <= n - 1 && 1 <= rs.second <= n - 1) {

int v = gcd(rs.second, n);

int u1 = positiveMod(e \* v, n);

int u2 = positiveMod(rs.first \* 3, n);

pair<int, int> u1G = multiplyPoint(u1, G6);

pair<int, int> u2Q = multiplyPoint(u2, Q);

pair<int, int> X = doubleAdditing(u1G, u2Q);

int X\_ = positiveMod(X.first, n);

if (rs.first == X\_) {

cout << "r: " << rs.first << " x: " << X\_ << endl;

return 1;

}

else {

cout << "r: " << rs.first << " x: " << X\_ << endl;

return 0;

}

}

else {

cout << "error sign ver";

return -1;

}

}

};

class Task1 {

public:

string plaintext;

pair<int, int> Pb;

vector<int> k;

Task1(string plaintext, pair<int, int> Pb, vector<int> k) {

this->plaintext = plaintext;

this->Pb = Pb;

this->k = k;

}

void activate(ElipticCurves obj) {

obj.encryptText(k, Pb, plaintext);

}

};

class Task2 {

int nb;

vector<pair<pair<int, int>, pair<int, int>>> ciphertext;

public:

Task2(int nb, vector<pair<pair<int, int>, pair<int, int>>> ciphertext) {

this->nb = nb;

this->ciphertext = ciphertext;

}

void activate(ElipticCurves obj) {

obj.decryptText(nb,ciphertext);

}

};

class Task3 {

public:

pair<int,int> P;

pair<int, int> Q;

pair<int, int> R;

int kP = 2;

int kQ = 3;

Task3(pair<int, int> P, pair<int, int> Q, pair<int, int> R) {

this->P = P;

this->Q = Q;

this->R = R;

}

void activate(ElipticCurves obj) {

pair<int, int> p2 = obj.multiplyPoint(kP, P);

pair<int, int> q3 = obj.multiplyPoint(kQ, Q);

pair<int, int> r\_ = obj.reverseSign(R);

pair<int, int> p2q3 = obj.doubleAdditing(p2, q3);

pair<int, int> C = obj.doubleAdditing(p2q3, r\_);

cout << "x: " << C.first << " y: " << C.second << endl;

}

};

class Task4 {

public:

pair<int, int> P;

int k;

Task4(pair<int, int> P, int k) {

this->P = P;

this->k = k;

}

void activate(ElipticCurves obj) {

pair<int, int> C = obj.multiplyPoint(k, P);

cout << "x: " << C.first << " y: " << C.second << endl;

}

};

class Task5 {

int e;

int d;

int k;

public:

Task5(int e, int d, int k) {

this->e = e;

this->d = d;

this->k = k;

}

void activate(ElipticCurves obj) {

obj.signatureGenerate(k, e, d);

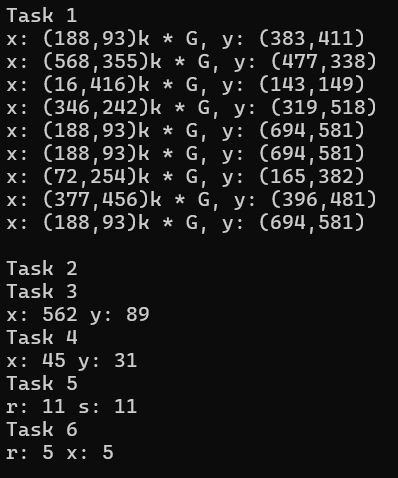
}

};

class Task6 {

int e;

pair<int, int> Q;

 pair<int, int> rs;

public:

Task6(int e,pair<int, int> Q,pair<int, int> rs) {

this->e = e;

this->Q = Q;

this->rs = rs;

}

void activate(ElipticCurves obj) {

obj.signatureVerification(e, rs, Q);

}

};

int main() {

ElipticCurves obj;

cout << "Task 1\n";

Task1 task1("Renessans", { 725,195 }, vector<int>{2, 19, 4, 8, 2, 2, 16, 10, 2});

task1.activate(obj);

cout << "Task 2\n";

vector<pair<pair<int, int>, pair<int, int>>> data = {

{{188, 93}, {623, 166}},

{{725, 195}, {513, 414}},

{{346, 242}, {461, 4}},

{{489, 468}, {739, 574}},

{{725, 195}, {663, 476}},

{{745, 210}, {724, 522}},

{{725, 195}, {663, 476}},

{{618, 206}, {438, 40}},

{{286, 136}, {546, 670}},

{{179, 275}, {73, 72}}

};

Task2 task2(32, data);

task2.activate(obj);

cout << "Task 3\n";

Task3 task3({ 59,386 }, { 61,129 }, { 100,364 });

task3.activate(obj);

cout << "Task 4\n";

Task4 task4({ 45,720 },111);

task4.activate(obj);

cout << "Task 5\n";

Task5 task5(11,5,6);

task5.activate(obj);

cout << "Task 6\n";

Task6 task6(7, {384,475}, { 5,5 });

task6.activate(obj);

}

Вывод: разработал программу для упражнений на эллиптических кривых.