Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе №11**

“Исследование криптографических алгоритмов на основе эллиптических кривых”

**Выполнил:** студент 4 курса

1 группы специальности ПОИТ

Халалеенко Андрей Николаевич

**Проверил:** преподаватель

Сазонова Дарья Владимировна

Минск 2024

**Найти точки ЭК для значений х, указанных в таблице.**

**Разработать приложение для выполнения операций над точками кривой: а) kР; б) Р + Q; в) kР + lQ – R; г) Р – Q + R.**

**Создать оконное приложение для зашифрования/расшифрования собственной фамилии (или имени – по выбору) на основе ЭК, указанной в задании 1, для генерирующей точки G = (0, 1).**

**Создать оконное приложение для генерации/верификации ЭЦП на основе алгоритма ЕСDSA: ЭК Е751(–1, 1) c генерирующей точкой G = (416, 55); порядок точки q = 13. Дополнительные параметры – в соответствии с вариантом из табл. 11.8 и 11.10.**

Реализация приложения:

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Numerics;  using System.Text;  using System.Threading.Tasks;  namespace \_22  {  public static class Show  {  public static void Dott(Dott dott)  {  Console.WriteLine("({0},{1})",dott.x, dott.y);  }  }  class Program  {  static void Main(string[] args)  {  ////1  EllipticCurve ellipticCurve = new EllipticCurve(-1,1,751,728);  //1.1  Dott dott = new Dott();  for (int x = 201; x < 235; x++)  {  dott = new Dott(x, ellipticCurve);  Show.Dott(dott);  }  //1.2  BigInteger k = 11;  BigInteger l = 4;  Dott P = new Dott(74, 170);  Dott Q = new Dott(53, 277);  Dott R = new Dott(86, 25);  Console.WriteLine("P");  Show.Dott(P);  Console.WriteLine("Q");  Show.Dott(Q);  Console.WriteLine("R");  Show.Dott(R);  Dott kP = Dott.DottMultiplication(P, k, ellipticCurve);  Console.WriteLine("kP");  Show.Dott(kP);  Dott PQ = Dott.Addition(P, Q, ellipticCurve);  Console.WriteLine("P + Q");  Show.Dott(PQ);  Dott kPlQ\_R = Dott.Addition(Dott.Addition(Dott.DottMultiplication(P, k, ellipticCurve),  Dott.DottMultiplication(Q, l, ellipticCurve),  ellipticCurve),  Dott.GetDottNegativeByY(R, ellipticCurve),  ellipticCurve);  Console.WriteLine("kP + lQ - R");  Show.Dott(kPlQ\_R);  Dott P\_QR = Dott.Addition(Dott.Addition(P,  Dott.GetDottNegativeByY(Q, ellipticCurve),  ellipticCurve),  R,  ellipticCurve);  Console.WriteLine("P - Q + R");  Show.Dott(P\_QR);  ////2  List<char> alphabet = new List<char>()  {  'А' ,  'Б' ,  'В' ,  'Г' ,  'Д' ,  'Е' ,  'Ж' ,  'З' ,  'И' ,  'Й' ,  'К' ,  'Л' ,  'М' ,  'Н' ,  'О' ,  'П' ,  'Р' ,  'С' ,  'Т' ,  'У' ,  'Ф' ,  'Х' ,  'Ц' ,  'Ч' ,  'Ш' ,  'Щ' ,  'Ъ' ,  'Ы' ,  'Ь' ,  'Э' ,  'Ю' ,  'Я'  };  List<Dott> alphabetOfDotts = new List<Dott>() {  new Dott (189, 297), //А  new Dott (189, 454), //Б  new Dott (192, 32), //В  new Dott (192, 719), //Г  new Dott (194, 205), //Д  new Dott (194, 546), //Е  new Dott (197, 145), //Ж  new Dott (197, 606), //З  new Dott (198, 224), //И  new Dott (198, 527), //Й  new Dott (200, 30), //К  new Dott (200, 721), //Л  new Dott (203, 324), //М  new Dott (203, 427), //Н  new Dott (205, 372), //О  new Dott (205, 379), //П  new Dott (206, 106), //Р  new Dott (206, 645), //С  new Dott (209, 82), //Т  new Dott (209, 669), //У  new Dott (210, 31), //Ф  new Dott (210, 720), //Х  new Dott (215, 247), //Ц  new Dott (215, 504), //Ч  new Dott (218, 150), //Ш  new Dott (218, 601), //Щ  new Dott (221, 138), //Ъ  new Dott (221, 613), //Ы  new Dott (226, 9), //Ь  new Dott (226, 742), //Э  new Dott (227, 299), //Ю  new Dott (227, 452) //Я  };  string message = "ХАЛАЛЕЕНКО";  Dott G = new Dott(0, 1);  BigInteger d = 31;  Q = Dott.DottMultiplication(G,d,ellipticCurve);  Console.WriteLine("Open Key");  Show.Dott(Q);  Random rand = new Random();  Dott C1 = new Dott();  Dott C2 = new Dott();  string messageNew = "";  for (int i = 0; i < message.Length; i++)  {  P = alphabetOfDotts[alphabet.IndexOf(message[i])];  Console.WriteLine(Dott.IsDottBelongsToCurve(P, ellipticCurve));  k = 5;  C1 = Dott.DottMultiplication(G, k, ellipticCurve);  C2 = Dott.DottMultiplication(Q, k, ellipticCurve);  C2 = Dott.Addition(P, C2, ellipticCurve);  Show.Dott(C1);  Show.Dott(C2);  P = Dott.Addition(C2,  Dott.GetDottNegativeByY(Dott.DottMultiplication(C1,  d,  ellipticCurve),  ellipticCurve),  ellipticCurve);  messageNew += alphabet[  alphabetOfDotts.IndexOf(  alphabetOfDotts.Find(dd => dd.x == P.x && dd.y == P.y)  )  ];  }  Console.WriteLine(messageNew);  //3  Dott baseDott = new Dott(384, 475);    BigInteger q = 13;  k = 3;  BigInteger data = 8;  BigInteger privateKey = 11;  Dott openKey = Dott.DottMultiplication(baseDott, privateKey,ellipticCurve);  Dott signature = ECDSA.CreateSignatureWithKandOrder(data, privateKey, baseDott, ellipticCurve, k, q);  Show.Dott(signature);  bool isVerifired = ECDSA.VerifySignatureWithOrder(data, openKey, signature, baseDott, ellipticCurve, q);  Console.WriteLine(isVerifired);  Console.ReadLine();  }  }  } |

Результат выполнения:





