Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**«Отчёт по лабораторной работе 13»**

“ ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ

АЛГОРИТМОВ НА ОСНОВЕ ЭЛЛИПТИЧЕСКИХ

КРИВЫХ ”

**Выполнила:** студентка 3 курса

4 группы специальности ПОИТ

Сапегина Екатерина Игоревна,

Вариант 11 (1)

Минск 2023

***Цель:*** изучение и приобретение практических навыков разработки и использования приложений для реализации криптографических алгоритмов на основе эллиптических кривых (содержит 3 самостоятельных задания, каждое из которых рассчитано на 2 часа аудиторных занятий).

***Задачи:***

1. Закрепить теоретические знания по алгебраическому описанию и геометрическому представлению операций над эллиптическими кривыми (ЭК):

• по алгоритмам согласования ключевой информации на основе ЭК;

• алгоритмам зашифрования/расшифрования информации на основе асимметричной криптонафии и ЭК;

• алгоритмам генерации и верификации электронной цифровой подписи на основе асимметричной криптографии и ЭК;

• оценке криптостойкости систем на основе ЭК.

2. Разработать приложение для реализации указанных преподавателем методов криптопреобразования на основе ЭК.

3. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

***Практическая часть:***

В основе задания – ЭК вида у2 = х3 – х + 1 (mod 751): а = –1, b = 1, р = 751, т. е. Е751(–1, 1).

Задание 1 (рассчитано на 2 часа аудиторных занятий).

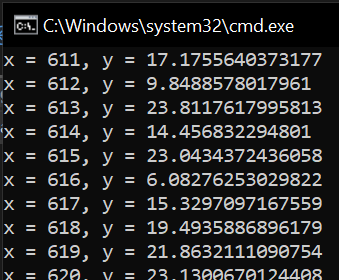
* 1. Найти точки ЭК для значений х, указанных в табл. 11.6



***Реализация:***

|  |
| --- |
| int xmin = 586, xmax = 620, a = -1, b = 1, p = 751;  for (int x = xmin; x <= xmax; x++)  { Console.WriteLine($"x = {x}, y = {Math.Sqrt((x \* x \* x - x + b) % p)}"); } |

***Результат:***

******

1.2. Разработать приложение для выполнения операций над

точками кривой:

а) kР; б) Р + Q; в) kР + lQ – R; г) Р – Q + R.

Варианты коэффициентов приведены в табл. 11.7.



В табл. 11.6 указаны координаты точек, над которыми выполняются операции.

***Реализация:***

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Numerics;  using System.Text;  using System.Threading.Tasks;  namespace lab\_13  {  class EllipticCurves  {  private static Random random = new Random();  private static string alphabeth = "абвгдежзийклмнопрстуфхцчшщъыьэюя";  private static int[,] points =  { { 189, 297 }, { 189, 454 }, { 192, 32 }, { 192, 719 }, { 194, 205 }, { 194, 546 }, { 197, 145 }, { 197, 606 },  { 198, 224 }, { 198, 527 }, { 200, 30 }, { 200, 721 }, { 203, 324 }, { 203, 427 }, { 205, 372 }, { 205, 379 },  { 206, 106 }, { 206, 645 }, { 209, 82 }, { 209, 669 }, { 210, 31 }, { 210, 720 }, { 215, 247 }, { 215, 504 },  { 218, 150 }, { 218, 601 }, { 221, 138 }, { 221, 613 }, { 226, 9 }, { 226, 742 }, { 227, 299 }, { 227, 542 } };  public static int[] InversePoint(int[] P)  {  return new int[2] { P[0], (-1) \* P[1] };  }  private static int CalculateLambda(int[] P, int a, int p)  {  return GCD.Mod(GCD.Mod(3 \* (P[0] \* P[0]) + a, p) \* GCD.ModInverse(2 \* P[1], p), p);  }  private static int CalculateLambda(int[] P, int[] Q, int p)  {  return GCD.Mod(GCD.Mod(Q[1] - P[1], p) \* GCD.Mod(GCD.ModInverse(Q[0] + GCD.Mod(-P[0], p), p), p), p);  }  public static int[] CalculateSum(int[] P, int[] Q, int p)  {  int lambda = CalculateLambda(P, Q, p);  int x = GCD.Mod(lambda \* lambda - P[0] - Q[0], p);  int y = GCD.Mod(lambda \* (P[0] - x) - P[1], p);  return new int[] { x, y };  }  public static int[] CalculateSum(int[] P, int a, int p)  {  int lambda = CalculateLambda(P, a, p);  int x = GCD.Mod(lambda \* lambda - P[0] - P[0], p);  int y = GCD.Mod(lambda \* (P[0] - x) - P[1], p);  return new int[] { x, y };  }  public static int[] kP(int k , int[] P, int a, int p)  {  int[] kP = P;  for (int i = 0; i < (int)Math.Log(k, 2); i++)  kP = CalculateSum(kP, a, p);  k = k - (int)Math.Pow(2, (int)Math.Log(k, 2));  while (k > 1)  {  for (int i = 0; i < (int)Math.Log(k, 2); i++)  kP = CalculateSum(kP, CalculateSum(P, a, p), p);  k = k - (int)Math.Pow(2, (int)Math.Log(k, 2));  }  if (k == 1) kP = CalculateSum(kP, P, p);  return kP;  }  public static int[,] Encrypt(string text, int[] G, int a, int p, int d)  {  int[] Q = kP(d, G, a, p), P;  int[,] encryptedText = new int[text.Length, 4];  int k;  Console.WriteLine($"G = ({G[0]}, {G[1]}), d = {d}, Q = ({Q[0]}, {Q[1]})");  for (int i = 0; i < text.Length; i++)  {  k = random.Next(2, d);  P = Enumerable.Range(0, points.GetLength(1)).Select(x => points[alphabeth.IndexOf(text[i]), x]).ToArray();  int[] C1 = kP(k, G, a, p), kQ = kP(k, Q, a, p), C2;  C2 = CalculateSum(P, kQ, p);  encryptedText[i, 0] = C1[0]; encryptedText[i, 1] = C1[1];  encryptedText[i, 2] = C2[0]; encryptedText[i, 3] = C2[1];  }  return encryptedText;  }  public static string Decrypt(int[,] encryptedText, int a, int p, int d)  {  string decryptedText = "";  for (int i = 0; i < encryptedText.GetUpperBound(0) + 1; i++)  {  int[] C1 = kP(d, new int[] { encryptedText[i, 0], encryptedText[i, 1] }, a, p), C2 = { encryptedText[i, 2], encryptedText[i, 3] };  int[] P = CalculateSum(C2, InversePoint(C1), p);  for (int k = 0; k < points.GetUpperBound(0) + 1; k++)  {  if (points[k, 0] == P[0] && points[k, 1] == P[1])  {  decryptedText += alphabeth[k];  break;  }  }  }  return decryptedText;  }  public static int[] CreateDigitalSign(int[] G, int q, int d, int a, int p)  {  Console.WriteLine($"G = ({G[0]}, {G[1]}), d = {d}, q = {q}");  int[] digitalSign = new int[2];  int[] kG;  int k, t;  do  {  do  {  k = random.Next(2, q);  kG = kP(k, G, a, p);  digitalSign[0] = kG[0] % q;  } while (digitalSign[0] <= 1);  t = GCD.ModInverse(k, q);  int H = points[alphabeth.IndexOf("м"), 0] % 13;  digitalSign[1] = (t \* (H + d \* digitalSign[0])) % q;  } while (digitalSign[1] <= 0);  return digitalSign;  }  public static bool VerifyDigitalSign(int[] digitalSign, int[] G, int q, int d, int a, int p)  {  if (digitalSign[0] <= 1 || digitalSign[1] >= q)  return false;  int H = points[alphabeth.IndexOf("м"), 0] % 13;  int w = GCD.ModInverse(digitalSign[1], q);  int u1 = (w \* H) % q;  int u2 = (w \* digitalSign[0]) % q;  int[] Q = kP(d, G, a, p), u1G = kP(u1, G, a, p), u2Q = kP(u2, Q, a, p);  int v = CalculateSum(u1G, u2Q, p)[0] % q;  return digitalSign[0] == v;  }  }  } |

***Результат:***

Задание 2 (рассчитано на 2 часа аудиторных занятий).

2.1. 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). Тайный ключ – в соответствии с вариантом из табл. 11.8.

2.2. Вычислить самостоятельно значение открытого ключа Q.

При этом следует воспользоваться основной формулой (11.9),

а также соотношениями (11.3)–(11.5) для случая P = Q; не следует

также забывать, что все вычисления производятся по mod 751;

см. также пример 5 (вычисление 2Р) и пример 7.

Принять, что шифруемым блоком является один символ сообщения, координаты которого на ЭК соответствуют табл. 11.9 (может быть принята за основу и иная таблица).

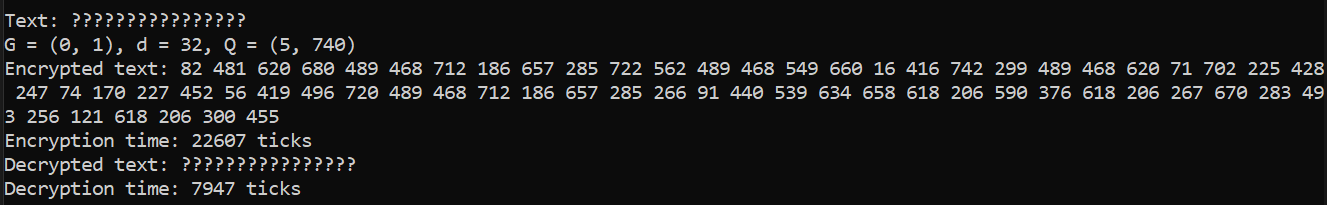
Параметры k – по собственному усмотрению.



***Реализация:***

|  |
| --- |
| string text = "катеринасапегина";  Console.WriteLine($"Text: {text}");  var stopwatch = Stopwatch.StartNew();  int[,] encryptedText = EllipticCurves.Encrypt(text, new int[] { 0, 1 }, a, p, 32);  stopwatch.Stop();  Console.WriteLine($"Encrypted text: {string.Join(" ", encryptedText.Cast<int>())}");  Console.WriteLine($"Encryption time: {stopwatch.ElapsedTicks} ticks");  stopwatch.Restart();  Console.WriteLine($"Decrypted text: {EllipticCurves.Decrypt(encryptedText, a, p, 32)}");  stopwatch.Stop();  Console.WriteLine($"Decryption time: {stopwatch.ElapsedTicks} ticks");  Console.WriteLine(); |

***Результат:***



Задание 3 (рассчитано на 2 часа аудиторных занятий).

3.1. 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.

3.2. Вычислить самостоятельно значение открытого ключа Q.

При этом следует воспользоваться основной формулой (11.8),

а также соотношениями (11.3)–(11.5) для случая P = Q; не следует

также забывать, что все вычисления производятся по mod 751;

см. также пример 5 (вычисление 2Р) и пример 7.

Параметры k – по собственному усмотрению.

3.3. Хешем подписываемого сообщения (Н(М)) является модуль по основанию 13 координаты х точки ЭК, соответствующей

первому символу собственной фамилии из табл. 11.10. Например,

фамилия начинается на букву «Я»: х = 227, тогда 227 mod 13 = 6,

значит, в данном конкретном случае Н(М) = 6.

***Реализация:***

|  |
| --- |
| stopwatch.Restart();  int[] digitalSign = EllipticCurves.CreateDigitalSign(new int[] { 416, 55 }, 13, 12, a, p);  stopwatch.Stop();  Console.WriteLine($"Digital sign: {digitalSign.Select(el => el.ToString()).Aggregate((prev, current) => prev + " " + current)}");  Console.WriteLine($"Creating digital sign time: {stopwatch.ElapsedTicks} ticks");  stopwatch.Restart();  Console.WriteLine($"Result of checking digital sign: {EllipticCurves.VerifyDigitalSign(digitalSign, new int[] { 416, 55 }, 13, 12, a, p)}");  stopwatch.Stop();  Console.WriteLine($"Checking digital sign time: {stopwatch.ElapsedTicks} ticks");  Console.ReadKey(); |

***Результат:***

