Учреждение образования

«Белорусский государственный технологический университет»

**Отчет по лабораторной работе №11**

по дисциплине: «Защита информации и надежность информационных систем»

Исполнитель:

Воликов Д. А., ФИТ 4-7

Руководитель:

Асс. Савельева М. Г.

Минск 2024

# Лабораторная работа №11. Исследование криптографических алгоритмов на основе эллиптических кривых

**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 криптографии и ЭК;
* алгоритмам генерации и верификации электронной цифровой подписи на основе асимметричной криптографии и ЭК;
* оценке криптостойкости систем на основе ЭК.

1. Разработать приложение для реализации указанных преподавателем методов криптопреобразования на основе ЭК.
2. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

## Теоретические сведения

**Определение 1**. Эллиптические кривые – математический объект, который может быть определен над любым полем.

**Определение 2**. Эллиптическая кривая над вещественными числами – это множество точек, описываемых уравнением

*y*2 = *х*3 + *a* ⋅ *х* + *b*,

при этом константы (*а* и *b* – вещественные числа) должны удовлетворять условию:

4 ⋅ *a*3 + 27 ⋅ *b*2 ≠ 0.

**Определение 3**. Частью ЭК является бесконечно удаленная точка (также известная как идеальная точка), которую мы обозначим символом *О*.

**Определение 4**. Группа – непустое множество с определенной на нем бинарной операцией, называемой сложением и удовлетворяющей нескольким аксиомам.

**Определение 5**. Группа для ЭК есть непустое множество, элементы которого являются точками ЭК, обладающими следующими свойствами:

* + единичный элемент – это бесконечно удаленная точка *О*;
  + обратная величина точки *R* – это точка, симметричная относительно оси *Х*;
  + сложение задается следующим правилом: сумма трех ненулевых точек *P*, *Q* и –*R*, лежащих на одной прямой, будет равна *P* + *Q* + (–*R*) = *О*.

В соответствии с этим можем сформулировать законы сложения точек эллиптической кривой:

* прямая, проходящая через точки *R* и –*R*, является вертикальной прямой, которая не пересекает ЭК ни в какой третьей точке; если *R* = (*х*, –*у*), то *R* + (*х*, *у*) = *О*. Точка (*х*, *у*) является отрицательным значением точки *R* и обозначается –*R*. Таким образом, по определению *R* + (–*R*) = *О*;
* *P* + *Q* = *R*: пусть *P* и *Q* – две различные точки ЭК, и *Р* не равно *Q*; если проведем через *P* и *Q* прямую, то она пересечет ЭК еще только в одной точке, называемой –*R*; точка –*R* отображается относительно оси *Х* в точку *R*, равную сумме точек *P* и *Q*: *P* + *Q* = *R*.

Что будет, если *P* = *Q*? В этом случае мы можем говорить об операции удвоения точки: *P* + *Р* = 2 ⋅ *Р*. Обобщив (к точке 2 ⋅ *Р* можно прибавить еще раз точку *Р*: 2 ⋅ *Р* + *Р*), сформулируем принцип умножения точки *Р* на целое положительное число *n* – это сумма *n* точек *Р*:

*n* ⋅ *P* = *P* + *P* + *P* + … + *P*.

Понятно, что каждая точка на плоскости задается парой координат: *х* и *у*.

Числа *х* и *у* являются рациональными, а точки *P*, *Q*, *R* и –*R* (как и любые точки ЭК) – рациональными точками.

Если *Р* = (*х*1, *у*1) и *Q* = (*х*2, *у*2), то *Р* + *Q* = (*х*3, *у*3) определяется в соответствии с правилами:

*x*3= *λ*2 – *х*1 – *х*2;

у3= *λ* ⋅ (*х*1 – *х*3) – *у*1,

где

*λ* = (*у*2 – *у*1) / (*х*2 – *х*1)

при *Р* ≠ *Q*

*λ* = (3х12 + *а*) / 2 ⋅ *у*1

при *Р* = *Q*.

ЭК над конечными полями. Именно этот тип ЭК используется в практике криптографии на основе ЭК.

**Определение 6**. Конечное поле – это множество конечного числа элементов. Примером конечного поля является множество целых чисел по модулю *p*, где *p* – простое число.

Поле обозначается как GF(*p*) или *F*p. Здесь операции сложения и умножения работают как в модулярной арифметике.

**Определение 7**. Эллиптическая кривая над полем *F*p задается теми же уравнениями, что и ЭК над действительными числами, только все вычисления производятся по модулю *р* (mod *p*):

*у*2 ≡ *х*3 + *a* ⋅ *х* + *b* (mod *p*),

далее для упрощения используем знак простого неравенства:

4 ⋅ *a*3 + 27 ⋅ *b*2 ≠ 0 (mod *p*).

Формально ЭК над полем задается так: *Е*р(*а*, *b*).

Важно отметить, что, как и ранее, существует точка (бесконечно удаленная) *О*; *а* и *b* – вещественные числа.

Операции над точками производятся аналогичным образом, как и в группе.

**Определение 8**. Если мы складываем два значения, кратных *Р*, то получаем значение, кратное *Р* (т. е. значения, кратные *Р*, замкнуты относительно операции сложения). Это означает, что множество кратных *Р* значений – это циклическая подгруппа группы, образованной эллиптической кривой.

**Определение 9**. Наименьшее значение числа *q*, для которого выполняется равенство *q* ⋅ *Р* = *О*, называется порядком точки *Р*.

**Определение 10**. Порядок группы точек эллиптической кривой равен числу различных точек ЭК, включая точку *О*.

**Определение 11**. Точка *Р* называется генератором или базовой точкой циклической подгруппы (такую точку во многих документах обозначают символом *G*).

Основные этапы генерации ключевой информации на основе ЭК. Первый этап: выбор (генерация) ЭК. Обычно он основан на выполнении следующих условий и операций.

1. Входными параметрами являются: число *l*, число *р*, удовлетворяющее условию 22 ⋅ *l* – 1 < *р* < 22 ⋅ *l*, *р* = 3 mod 4, 0 < *a* < *p*. Можно использовать некоторое простое число *р* = 22 ⋅ *l* – *с*, где *с* – небольшое натуральное число.
2. Выбирается число *b* такое, что 0 < *b* < *p*. Таким образом, задана ЭК:  
   *Е*р(*а*, *b*).
3. Выбираются порядок *q* (простое число) и генерирующая точка *G*, которая задается двумя координатами, например *G* = (0, *у*G).

Второй этап: генерация ключевой информации.

1. Входными параметрами являются: *р*, *а*, *b*, *q* и *G*.
2. Генерируется тайный ключ – число *d*, выбранное из множества {1, 2, …, *q* – 1}.
3. Вычисляется открытый ключ – точка *Q*:

*Q* = *d* ⋅ *G*,

к открытому ключу также относятся *р*, *а*, *b*, *q*.

Реализация алгоритма зашифрования/расшифрования на основе ЭК. Вспомним, что процедура предусматривает использование ключей получателя (стороны В). Рассмотрим это на примере алгоритма Эль-Гамаля.

При использовании ЭК зашифрование предполагает представление сообщения в виде точки *Р* (или представления каждого блока сообщения в виде разных точек *Р*i) ЭК с известной точкой *G* и известным *Q*. Соответственно шифртекст – это две точки на той же ЭК: *С*1 и *C*2 или *С*i1 и *C*i2.

Предположим, что шифруемое сообщение *М* – это точка *Р* на ЭК. Сторона *А* выбирает некоторое случайное число *k* и далее выполняет вычисления с использованием открытого ключа стороны *В*:

*С*1 = *k ⋅ G*, *С*2 = *P* + *k* ⋅ *Q*.

Получатель для расшифрования сообщения вычисляет:

*P* = *С*2 – *d* ⋅ *C*1.

Знак «–» означает сложение с инверсией: инверсией по отношению к точке (*х*, *у*) является точка (*х*, –*у*) на ЭК.

Реализация ЭЦП на основе ЭК. Рассмотрим генерацию и верификацию ЭЦП на основе алгоритма DSA и ЭК (EC) – ЕСDSA. Стоит обратить внимание на то, что используется ключевая информация отправителя (стороны *А*). Генерация ключей происходит так же, как описано выше. Однако во внимание должен приниматься еще один известный параметр ЭК: порядок точки *G*, т. е. число *q*.

Краткая характеристика алгоритма генерации и верификации ЭЦП состоит в следующем. Полагаем, что отправитель подписывает хеш *Н*(*М*) сообщения *М*.

Генерация ЭЦП.

1. Выбрать число *k* (1 < *k* < *q*), *q* – порядок точки *G*.
2. Вычислить точку *k* ⋅ *G* = (*х*, *у*), вычислить *r* ≡ *x* mod *q*; при *r* = 0 изменить *k* и повторить шаг 2.
3. Вычислить *t* ≡ *k*–1 mod *q* (например, на основе расширенного алгоритма Евклида).
4. Вычислить *s* = *t* ⋅ (*H*(*M*) + *d* ⋅ *r*)) mod *q*; при *s* = 0 изменить *k* и повторить алгоритм.

Стороне *В* отсылаются сообщение *М* и ЭЦП (числа *r* и *s*).

Верификация ЭЦП. Получатель знает алгоритм хеширования, который использовался отправителем, открытый ключ отправителя, с помощью чего выполняет следующие операции над *М* и полученной ЭЦП (обозначения чисел оставим без изменений).

1. Проверить выполнение условия: 1< *r*, *s* < *q*; если условие не выполняется, то легитимность подписи не подтверждается, в противном случае – выполняются дальнейшие шаги.
2. Вычисляются *Н*(*М*) и *w* ≡ *s*–1 mod *q*.
3. Вычисляются *u*1 ≡ *w* ⋅ *Н*(*М*) (mod *q*), *u*2 ≡ *w* ⋅ *r* (mod *q*).
4. Вычисляются *G* ⋅ *u*1 + *Q* ⋅ *u*2 = (*x*', *y*'), *v* ≡ *x*' mod *q*.
5. Сравниваются *v* и *r*; если равенство выполняется, подтверждается легитимность подписи и целостность полученного сообщения.

## Практическое задание

Задание 1. Найти точки ЭК для значений *х*min = 36 и *x*max = 70. Разработать приложение для выполнения операций над точками кривой, где *k* = 6 и *l* = 10:

а) *k* ⋅ *P*; б) *P* + *Q*; в) *k* ⋅ *P* + *k* ⋅ *P* – *Q*; г) *P* – *Q* + *R*.

Точки *P*, *Q* и *R* принимают следующие значения: *P* = (61, 129), *Q* = (59, 365), *R* = (105, 369).

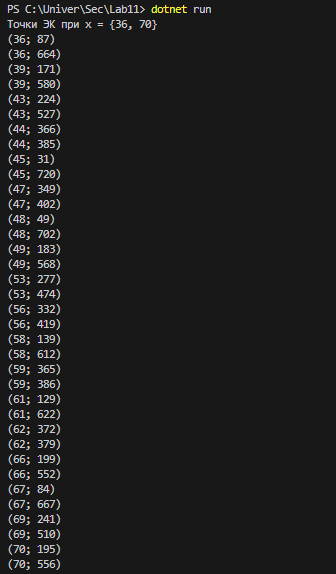


Рисунок 1 – Точки ЭК при данных *x*min и *x*max

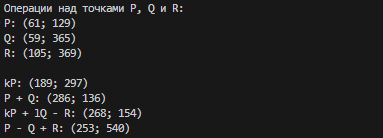


Рисунок 2 – Значения операций над точками

Задание 2. Создать 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 (*d* = 27).

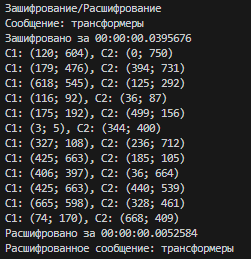


Рисунок 3 – Зашифрование/расшифрование сообщения

Как можно заметить, зашифрованное сообщение представляет собой массив пар точек, т.е. каждый символ сообщения – это 2 точки ЭК (ибо операции основаны на базе алгоритма ElGamal).

Задание 3. Создать приложение для генерации/верификации ЭЦП на основе алгоритма ЕСDSA: ЭК *Е*751(–1, 1) c генерирующей точкой *G* = (416, 55); порядок точки *q* = 13. Закрытый ключ *d* = 12.

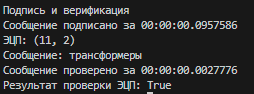


Рисунок 4 – Результат подписи сообщения и проверки подписи

Как можно заметить, ЭЦП представляет собой пару чисел, которые в дальнейшем используются для проверки подписи на основе алгоритма ECDSA.

Вывод: в ходе этой лабораторной работы я изучил основы криптографии на базе эллиптических кривых. Разобрал основные операции над точками ЭК, а также тонкости вычислений. После этого создал приложение, которое генерирует ЭК, зашифровывает и расшифровывает сообщение с использованием точек ЭК, а также реализовал алгоритм ЭЦП ECDSA.

# Листинг «Лабораторная работа №11»

using Lab11.ExtensionMethods;

namespace Lab11.Elliptical

{

public class EllipticalCurve

{

public int A { get; private set; }

public int B { get; private set; }

public int P { get; private set; }

public EllipticalPoint[] Points { get; private set; } // w/o Infinite point

public EllipticalCurve(int a, int b, int p)

{

if (p <= 0) throw new Exception("P param of curve cannot be negative or 0");

A = a;

B = b;

P = p;

Points = GetEllipticalPoints();

}

public EllipticalPoint[] GetPointsRange(int x1, int x2)

{

if (x1 < 0 || x2 < 0) throw new Exception("params cannot be below 0");

x1 = x1 >= P ? 0 : x1;

x2 = x2 >= P ? P : x2;

return [.. Points.Where(point => point.X >= x1 && point.X <= x2)];

}

public IPoint SumPoints(IPoint point1, IPoint point2)

{

var check = CheckPointsForInfinity(point1, point2);

if (check != null) return check;

int lambda = GetLambda(point1, point2);

if (lambda == -1) return new EllipticalInfinitePoint();

int x = (lambda \* lambda - point2.X - point1.X) % P;

int y = (-point1.Y - lambda \* (x - point1.X)) % P;

return new EllipticalPoint(x.RemoveMinusByModule(P), y.RemoveMinusByModule(P));

}

public IPoint MultiplyPoint(IPoint point, int k)

{

if (k == 0 || point is EllipticalInfinitePoint) return new EllipticalInfinitePoint();

IPoint result = new EllipticalInfinitePoint();

var addend = point;

while (k > 0)

{

if ((k & 1) == 1)

{

result = SumPoints(result, addend);

}

addend = SumPoints(addend, addend);

k >>= 1;

}

return result;

}

public int GetPointOrder(IPoint point)

{

int order = 1;

IPoint buf = point;

while (buf is not EllipticalInfinitePoint)

{

buf = SumPoints(buf, point);

order++;

}

return order;

}

public IPoint GetMinusPoint(IPoint point)

{

return point is EllipticalInfinitePoint ?

point :

new EllipticalPoint(point.X, (-point.Y).RemoveMinusByModule(P));

}

private int GetLambda(IPoint point1, IPoint point2)

{

if (point1.X == point2.X && point1.Y != point2.Y) return -1;

int up = point2.Y - point1.Y;

int down = point2.X - point1.X;

if (point1.X == point2.X)

{

up = 3 \* point1.X \* point1.X + A;

down = 2 \* point1.Y;

if (down == 0) return -1;

}

up = up.RemoveMinusByModule(P) % P;

down = down.RemoveMinusByModule(P) % P;

return up % down == 0 ? up / down % P : up \* down.GetReversed(P) % P;

}

private EllipticalPoint[] GetEllipticalPoints()

{

var xRange = Enumerable.Range(0, P).Select(x => KeyValuePair.Create(x, CurveFunction(x)));

var yRange = Enumerable.Range(0, P).Select(y => KeyValuePair.Create(y, y \* y % P));

return [.. xRange.SelectMany(

x => yRange.Where(y => y.Value == x.Value).Select(y => new EllipticalPoint(x.Key, y.Key))

)];

}

private static IPoint? CheckPointsForInfinity(IPoint point1, IPoint point2)

{

return (point1, point2) switch {

(EllipticalPoint point, EllipticalInfinitePoint) => point,

(EllipticalInfinitePoint, EllipticalPoint point) => point,

(EllipticalInfinitePoint, EllipticalInfinitePoint) => new EllipticalInfinitePoint(),

\_ => null

};

}

private int CurveFunction(int x) => (x \* x \* x + A \* x + B) % P;}}

Листинг 1 – Класс «EllipticCurve»

using Lab11.Elliptical;

using Lab11.Generators.PrimeGenerator;

namespace Lab11.ElGamal

{

public class ElGamal

{

public EllipticalCurve Curve;

public IPoint G { get; } = new EllipticalPoint(0, 1); // point generator encryption

public IPoint Q { get; private set; } // public key

private readonly int D; // private key

private readonly string RussianAlphabet = "абвгдеёжзийклмнопрстуфхцчшщъыьэюя";

public ElGamal(EllipticalCurve curve)

{

if (curve.Points.Length <= 33)

{

throw new Exception("Curve has not much points for russian alphabet (at least 34 points)");

}

Curve = curve;

D = new Random().Next(2, Curve.P - 1);

Q = curve.MultiplyPoint(G, D);

}

public EncryptedChar[] Encrypt(string message)

{

return message

.Select(c => {

var k = PrimeGenerator.GetRandomPrime();

var p = Curve.Points[RussianAlphabet.IndexOf(c)];

return new EncryptedChar

{

C1 = Curve.MultiplyPoint(G, k),

C2 = Curve.SumPoints(p, Curve.MultiplyPoint(Q, k))

};

}).ToArray();

}

public string Decrypt(EncryptedChar[] message)

{

return new string(

message.Select(c => {

var point = Curve.SumPoints(c.C2!, Curve.GetMinusPoint(Curve.MultiplyPoint(c.C1!, D)));

return RussianAlphabet[Array.IndexOf(Curve.Points, point)];

}).ToArray()

);

}

public record class EncryptedChar()

{

public IPoint? C1 { get; init; }

public IPoint? C2 { get; init; }

}

}

}

Листинг 2 – Класс «ElGamal»

using Lab11.Elliptical;

using Lab11.ExtensionMethods;

using Lab11.Hash;

namespace Lab11.ECDSA

{

public class ECDSA

{

public EllipticalCurve Curve;

public IPoint G { get; } = new EllipticalPoint(416, 55); // point generator signature

public IPoint Q { get; private set; } // public key

private readonly int D; // private key

private readonly int Order;

public ECDSA(EllipticalCurve curve)

{

if (curve.Points.Length <= 33)

{

throw new Exception("Curve has not much points for russian alphabet (at least 34 points)");

}

Curve = curve;

Order = curve.GetPointOrder(G);

D = new Random().Next(2, Order - 1);

Q = curve.MultiplyPoint(G, D);

}

public SignedMessage Sign(string message)

{

var hM = (int)(SHA1.Hash(message, false) % Order);

var s = 0;

var r = 0;

while (s == 0)

{

var k = new Random().Next(2, Order - 1);

var kG = Curve.MultiplyPoint(G, k);

r = kG.X % Order;

if (r == 0) continue;

s = k.GetReversed(Order) \* (hM + D \* r) % Order;

}

return new SignedMessage

{

Message = message,

R = r,

S = s

};

}

public bool Verify(SignedMessage signedMessage)

{

if (!Enumerable.Range(2, Order - 2).Contains(signedMessage.R)) {

return false;

}

if (!Enumerable.Range(2, Order - 2).Contains(signedMessage.S)) {

return false;

}

var hM = (int)(SHA1.Hash(signedMessage.Message!, false) % Order);

var w = signedMessage.S.GetReversed(Order);

var u1 = w \* hM % Order;

var u2 = w \* signedMessage.R % Order;

var u1G = Curve.MultiplyPoint(G, u1);

var u2Q = Curve.MultiplyPoint(Q, u2);

var resultPoint = Curve.SumPoints(u1G, u2Q);

if (resultPoint is EllipticalInfinitePoint) return false;

return resultPoint.X % Order == signedMessage.R;

}

public record class SignedMessage()

{

public string? Message { get; init; }

public int R { get; init; }

public int S { get; init; }

}

}

}

Листинг 3 – Класс «ECDSA»