Министерство образования Республики Беларусь

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Отчет к лабораторной работе № 10

«ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ АЛГОРИТМОВ НА ОСНОВЕ ЭЛЛИПТИЧЕСКИХ КРИВЫХ»

Выполнила:

студентка 3 курса 2 группы

Шастовская М. С.

Вариант: 12

Преподаватель:

Сазонова Д.В.

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:**

* 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;

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

у2 = х3 + aх + b,

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

4a3 + 27b2 ≠ 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.

На основе последнего определения мы можем определить группу для ЭК.

*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 (рис. 11.1), 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.

*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 Fp. Здесь операции сложения и умножения работают как в модулярной арифметике.

Например, поле F13 (р = 13) состоит из чисел: 0, 1, …, 12.

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

у2 ≡ х3 + aх + b (mod p).

*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 qР = 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).

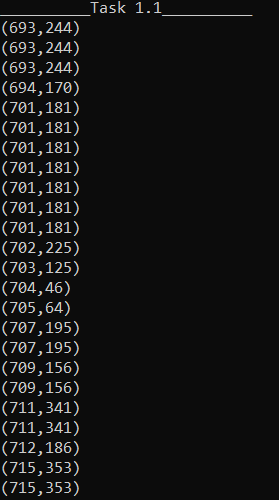
Порядок точки Р связан с порядком m ЭК теоремой Лагранжа, согласно которой порядок подгруппы – это делитель порядка исходной группы. Иными словами, если ЭК содержит m точек, а одна из подгрупп содержит q, то q является делителем m.

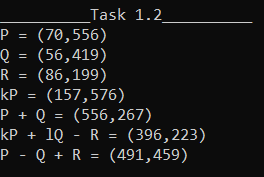
Рассмотрим наиболее общий случай. Предположим, что Eр – это ЭК над Fр, а Q – заранее определенная и согласованная сторонами А и В точка на E.

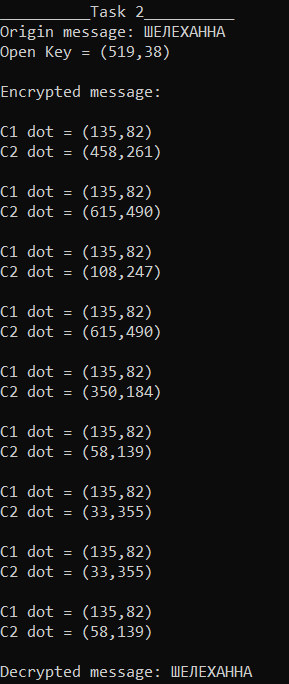
Отправитель A выбирает тайное случайное число kA, вычисляет точку РА = kA Q и отправляет ее получателю B. Bдействует аналогично: он случайным образом выбирает число kB, вычисляет случайное число kA, вычисляет точку РВ = kВQ и отправляет результат стороне A.

Общий ключ P = kAkBQ. Отправитель A вычисляет P путем умножения числа РВ, полученного от получателя B, на его секретное число kA. Похожим образом действует другая строна.

2 Практическая часть







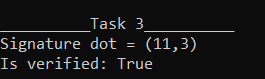


Рисунок 2.1 – Результат генерации алгоритмов на основе ЭК

На данных скриншотах отображены задания 1-3. В задание 1.1 были найдены точки ЭК для значений x, далее в 1.2 было разработано приложение для выполнения операций на точками кривой в соответствии с вариантом. В следующем задании было разработано приложение для зашифрования/расшиврования, вычисление открытого ключа. В третьем задании было создано приложение для генерации/верификации ЭЦП на основе ECDSA.

Основные функции программы представлены на рисунке 2.2.

public static Dott CreateSignatureWithKandOrder(BigInteger data, BigInteger privateKey, Dott baseDott, EllipticCurve curve, BigInteger k, BigInteger order)

{

stopWatch.Restart();

Dott dott;

BigInteger r = 0;

BigInteger s = 0;

while (s == 0)

{

r = 0;

while (r == 0 || k == 0)

{

dott = Dott.DottMultiplication(baseDott, k, curve);

r = dott.x % order;

}

if ((data + r \* privateKey) % k == 0)

s = ((data + r \* privateKey) / k) % order;

else if (AreNumbersCoprime(k, order))

{

s = ((data + r \* privateKey) \* ModInverse(k, order)) % order;

}

else

s = 0;

}

stopWatch.Stop();

createSignatureTicks.Add(stopWatch.ElapsedTicks);

return new Dott(r, s, curve);

}

public static Dott CreateSignatureWithK(BigInteger data, BigInteger privateKey, Dott baseDott, EllipticCurve curve, BigInteger k)

{

stopWatch.Restart();

Dott dott;

BigInteger r = 0;

BigInteger s = 0;

while (s == 0)

{

r = 0;

while (r == 0 || k == 0)

{

dott = Dott.DottMultiplication(baseDott, k, curve);

r = dott.x % curve.order;

}

if ((data + r \* privateKey) % k == 0)

s = ((data + r \* privateKey) / k) % curve.order;

else if (AreNumbersCoprime(k, curve.order))

{

s = ((data + r \* privateKey) \* ModInverse(k, curve.order)) % curve.order;

}

else

s = 0;

}

stopWatch.Stop();

createSignatureTicks.Add(stopWatch.ElapsedTicks);

return new Dott(r, s, curve);

}

public static Dott Addition(Dott left, Dott right, EllipticCurve curve)

{

Dott result = new Dott();

BigInteger lambda;

if (left.x == -1 && left.y == -1)

{

result.x = right.x;

result.y = curve.modulo - right.y;

}

else if (right.x == -1 && right.y == -1)

{

result.x = left.x;

result.y = curve.modulo - left.y;

}

else if (left.x == right.x && left.y != right.y)

{

result.x = -1;

result.y = -1;

}

else

{

if (left == right)

{

if ((3 \* (BigInteger.Pow(left.x, 2)) + curve.a) % (2 \* left.y) == 0)

lambda = (3 \* (BigInteger.Pow(left.x, 2)) + curve.a) / (2 \* left.y);

else

lambda = (3 \* (BigInteger.Pow(left.x, 2)) + curve.a) \* BigInteger.ModPow(2 \* left.y, curve.modulo - 2, curve.modulo);

}

else

{

if ((right.y - left.y) % (right.x - left.x) == 0)

lambda = (right.y - left.y) / (right.x - left.x);

else

lambda = ((right.y - left.y) > 0 ? (right.y - left.y) : (right.y - left.y) + curve.modulo)

\* (BigInteger.ModPow(right.x - left.x, curve.modulo - 2, curve.modulo) % curve.modulo >= 0

? BigInteger.ModPow(right.x - left.x, curve.modulo - 2, curve.modulo) % curve.modulo

: BigInteger.ModPow(right.x - left.x, curve.modulo - 2, curve.modulo) % curve.modulo + curve.modulo)

% curve.modulo;

}

result.x = ((lambda \* lambda - left.x - right.x) % curve.modulo) >= 0

? ((lambda \* lambda - left.x - right.x) % curve.modulo)

: ((lambda \* lambda - left.x - right.x) % curve.modulo) + curve.modulo;

result.y = ((lambda \* (left.x - result.x) - left.y) % curve.modulo) >= 0

? ((lambda \* (left.x - result.x) - left.y) % curve.modulo)

: ((lambda \* (left.x - result.x) - left.y) % curve.modulo) + curve.modulo;

}

return result;

}

Вывод

В ходе выполнения лабораторной работы были изучены и приобретены практические навыки разработки и использования приложений для реализации криптографических алгоритмов на основе эллиптических кривых.