Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы защиты информации

ОТЧЁТ
к лабораторной работе №7
на тему
на тему «Реализация схемы шифрования (дешифрования) для аналога алгоритма Эль-Гамаля на основе эллиптических кривых»

Выполнил: Е.А. Киселева

Проверил: А. В. Герчик

Минск 2024

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Целью выполнения данной лабораторной работы является реализация схемы шифрования (дешифрования) для аналога алгоритма Эль-Гамаля на основе эллиптических кривых.

2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Преимущество подхода на основе эллиптических кривых заключается в том, что в данном случае обеспечивается эквивалентная защита при меньшей длине ключа.

В общем случае уравнение эллиптической кривой E имеет вид:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

В качестве примера рассмотрим эллиптическую кривую E, уравнение которой имеет вид:

$$y^2 + y = x^3 - x^2$$

На этой кривой лежат только четыре точки, координаты которых являются целыми числами. Это точки A(0, 0), B(1, -1), C(1, 0) и D(0, -1).

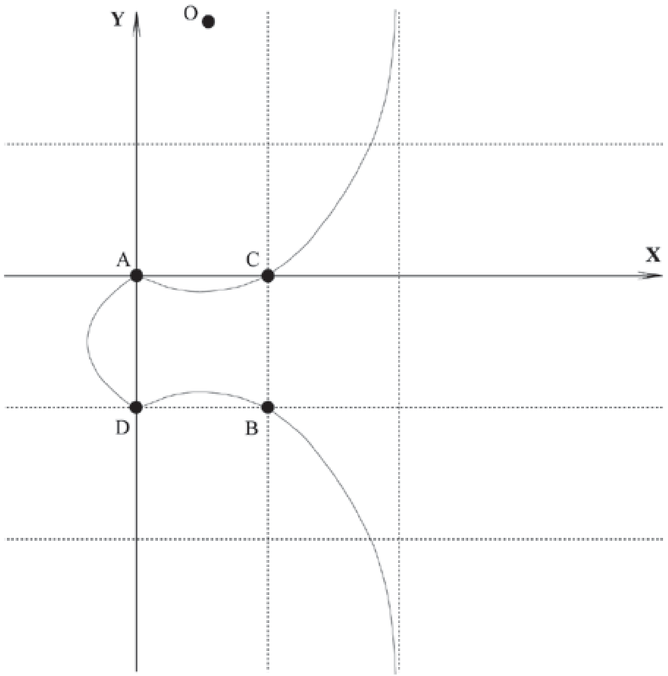

Рисунок 2.1 – Пример эллиптической кривой с четырьмя точками

Для определения операции сложения для точек на эллиптической кривой сделаем следующие предположения:

1 На плоскости существует бесконечно удаленная точка 0 E, в которой сходятся все вертикальные прямые.

2 Будем считать, что касательная к кривой пересекает точку касания два раза.

3 Если три точки эллиптической кривой лежат на прямой линии, то их сумма есть 0.

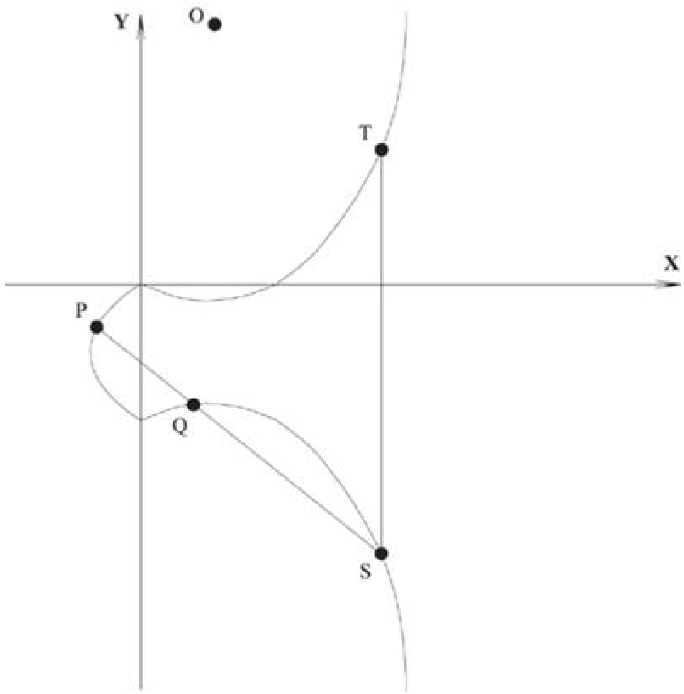

Рисунок 2.2 – Сложение точек на эллиптической кривой

Введем следующие правила сложения точек на эллиптической кривой:

1 Точка 0 выступает в роли нулевого элемента. Так, 0 = -0 и для любой точки P на эллиптической кривой P + 0 = P.

2 Вертикальная линия пересекает кривую в двух точках с одной и той же координатой x - скажем, S = (x, y) и T = (x, -y). Эта прямая пересекает кривую и в бесконечно удаленной точке. Поэтому $P_1 + P_2 + 0 = 0$ и $P_1 = -P_2$.

3 Чтобы сложить две точки P и Q (см. рисунок 11.2) с разными координатами x, следует провести через эти точки прямую и найти точку пересечения ее с эллиптической кривой. Если прямая не является касательной к кривой в точках P или Q, то существует только одна такая точка, обозначим ее S. Согласно нашему предположению P + Q + S = O. Следовательно, P + Q = -S или P + Q = T.

Если прямая является касательной к кривой в какой-либо из точек P или Q, то в этом случае следует положить S = P или S = Q соответственно.

Чтобы удвоить точку Q, следует провести касательную в точке Q и найти другую точку пересечения S с эллиптической кривой. Тогда Q + Q = 2 × Q = -S.

Введенная таким образом операция сложения подчиняется всем обычным правилам сложения, в частности коммутативному и ассоциативному законам. Умножение точки P эллиптической кривой на положительное число k определяется как сумма k точек P.

В криптографии с использованием эллиптических кривых все значения вычисляются по модулю p, где p является простым числом. Элементами данной эллиптической кривой являются пары неотрицательных целых чисел, которые меньше p и удовлетворяют частному виду эллиптической кривой:

$$y^2 \equiv x^3 + ax + b \pmod p$$

Такую кривую будем обозначать $E_p(a,b)$. При этом числа a и b должны быть меньше p и должны удовлетворять условию $4a^3 + 27b^2 \pmod p \neq 0$. Множество точек на эллиптической кривой вычисляется следующим образом.

Для каждого такого значения x, что $0 \le x \le p$, вычисляется $x^3 + ax + b \pmod p$.

Для каждого из полученных на предыдущем шаге значений выясняется, имеет ли это значение квадратный корень по модулю p. Если нет, то в $E_p(a,b)$ нет точек с этим значением x. Если корень существует, имеется два значения y, соответствующих операции извлечения квадратного корня (исключением является случай, когда единственным значением оказывается y = 0). Эти значения (x,y) и будут точками $E_p(a,b)$.

Множество точек $E_p(a,b)$ обладает следующими свойствами:

1. P + 0 = P.
2. Если P = (x,y), то P + (x,-y) = 0. Точка (x,-y) является отрицательным значением точки P и обозначается -P. Заметим, что (x,-y) лежит на эллиптической кривой и принадлежит $E_p(a,b)$.
3. Если $P = (x_1,y_1)$ и $Q = (x_2,y_2)$, где $P \neq Q$, то $P + Q = (x_3,y_3)$ определяется по следующим формулам:

$$x_3 \equiv \lambda^2 - x_1 - x_2 \pmod p$$
$$y_3 \equiv \lambda (x_1 - x_3) - y_1 \pmod p$$

где $(y_2 - y_1)/(x_2 - x_1)$, если $P \neq Q$, $\lambda = (3x_1^2 + a)/2y_1$, если P = Q

Число λ есть угловой коэффициент секущей, проведенной через точки $P = (x_1, y_1)$ и $Q = (x_2, y_2)$. При P = Q секущая превращается в касательную, чем и объясняется наличие двух формул для вычисления λ.

Задача, которую должен решить в этом случае атакующий, есть своего рода задача **"дискретного логарифмирования на эллиптической кривой"**, и формулируется она следующим образом. Даны точки P и Q на эллиптической кривой $E_p(a,b)$. Необходимо найти коэффициент k < p такой, что

$$P = k \times Q.$$

Относительно легко вычислить P по данным k и Q, но довольно трудно вычислить k, зная P и Q.

3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной была реализована схема шифрования (дешифрования) для аналога алгоритма Эль-Гамаля на основе эллиптических кривых.

Начальный текст находится в файле input.txt. Программа считывает необходимую информацию, а именно значение начальной точки P, такие как x, y, a, b, p. При шифровании проводится расчет двух точек кривой C1 и C2. После некоторого преобразования над этими точками и начальной точкой P выводится расшифрованное сообщение.

Результат выполнения лабораторной работы представлен на рисунке 3.1.

```
C:\projects\mzi\Lab7\lab7\venv\Scripts\python.exe C:\projects\mzi\Lab7\lab7\main.py
C values:
C1(X: 134247519101210251343400634075301034080183528193652242162126502785259764 30757, Y: 46883626845316720140799379852890090342037496639877965728593282180686861473606)
C2(X: 44683945822979517998040752995756727597170528105267953672337856312088319545174, Y: 36769355488810654987105915190205072122850432923896990358945948850978949936132)

Decrypted message:
Hello, мир!

Process finished with exit code 0
```

Рисунок 3.1 – Результат выполнения лабораторной работы

Таким образом результатом лабораторной работы является реализованная схема шифрования (дешифрования) для аналога алгоритма Эль-Гамаля на основе эллиптических кривых.

ВЫВОДЫ

В ходе данной лабораторной работы была разработана схема шифрования (дешифрования) для аналога алгоритма Эль-Гамаля на основе эллиптических кривых.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Схема шифрования Эль Гамаля [Электронный ресурс]. – Режим доступа: https://crypto-r.narod.ru/glava4/glava4_5.html/. – Дата доступа: 26.10.2024.

[2] Эллиптические кривые [Электронный ресурс]. – Режим доступа: https://homepage.mi-ras.ru/. – Дата доступа: 27.10.2024.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг программного кода

Листинг 1 – Программный код файла main.py

```
from EllipticCurvePoint import EllipticCurvePoint
from ElGamal import ElGamal


with open("./input.txt", "rb") as f:
    message = f.read()

# Инициализация эллиптической кривой и точек
P = EllipticCurvePoint(
    x=2,

y=4018974056539037503335449422937059775635739389905545080690979365213431566280,
    a=90,

b=4330887654676727690576590459565093199594211179445103958325296884203384958041
4,

p=5789604461865809771178549250434395392663499233282028201972879200395656482104
1
)

d =
47296044618658097711785492524343953912234992332820282019728792003956564821041
Q = P.multiply(d)

#Шифровка, вывод C1, C2 и соотв У
CValues = ElGamal.encrypt(message, P, Q)
print(f"C values:\nC1(X: {CValues[0].x}, Y: {CValues[0].y})\nC2(X:
{CValues[1].x}, Y: {CValues[1].y})\n")

# Дешифровка и вывод сообщения
decrypted_message = ElGamal.decrypt(CValues, d)
print(f"Decrypted message:\n{decrypted_message.decode('utf-8')}")
```

Листинг 2 – Программный код файла ElGamal.py

```
from EllipticCurvePoint import EllipticCurvePoint
from Crypto.Random import random
from Crypto.Util.number import long_to_bytes, bytes_to_long


class ElGamal:
    @staticmethod
    def generate_random_big_integer(N):
        # Генерация случайного числа, меньшего N
        bytes_len = (N.bit_length() + 7) // 8
        while True:
            r = random.getrandbits(bytes_len * 8) % N
            if r < N:
                return r

    @staticmethod
    def get_point_from_bytes(message_bytes, P):
        # Преобразование байтового сообщения в точку эллиптической кривой
        p_length = (P.p.bit_length() + 7) // 8
```

```
        if len(message_bytes) >= p_length - 2:
            raise Exception(f"M({len(message_bytes)}) should be less than p
(Max M Length = {p_length - 2} symbols)")

        # Дополнение сообщения байтами для преобразования в координату x
        message = message_bytes + bytes([0xff]) + b'\x00' * (p_length -
len(message_bytes) - 1)
        return EllipticCurvePoint(
            x=bytes_to_long(message),
            y=0,
            a=P.a,
            b=P.b,
            p=P.p
        )

    @staticmethod
    def get_bytes_from_point(P):
        # Извлечение сообщения из координаты x точки эллиптической кривой
        message_bytes = long_to_bytes(P.x)
        if 0xff in message_bytes:
            return message_bytes[:message_bytes.index(0xff)]
        return message_bytes

    @staticmethod
    def encrypt(message_bytes, P, Q):
        M = ElGamal.get_point_from_bytes(message_bytes, P)
        k = ElGamal.generate_random_big_integer(P.p)
        C1 = P.multiply(k)
        C2 = M + Q.multiply(k)
        return C1, C2

    @staticmethod
    def decrypt(CValues, d):
        temp = CValues[0].multiply(d)
        temp.y = -temp.y % temp.p
        P = temp + CValues[1]
        return ElGamal.get_bytes_from_point(P)
```

Листинг 3 – Программный код файла EllipticCurvePoint.py

```
class EllipticCurvePoint:
    def __init__(self, x, y, a, b, p):
        self.x = x
        self.y = y
        self.a = a
        self.b = b
        self.p = p

    def __add__(self, other):
        # Операция сложения точек
        if self.x == other.x and self.y == other.y:
            return self.double()

        dy = (other.y - self.y) % self.p
        dx = (other.x - self.x) % self.p

        m = (dy * pow(dx, -1, self.p)) % self.p

        x3 = (m * m - self.x - other.x) % self.p
        y3 = (m * (self.x - x3) - self.y) % self.p

        return EllipticCurvePoint(x3, y3, self.a, self.b, self.p)
```

```
    def double(self):
        # Удвоение точки
        dy = (3 * self.x * self.x + self.a) % self.p
        dx = (2 * self.y) % self.p

        m = (dy * pow(dx, -1, self.p)) % self.p

        x2 = (m * m - 2 * self.x) % self.p
        y2 = (m * (self.x - x2) - self.y) % self.p

        return EllipticCurvePoint(x2, y2, self.a, self.b, self.p)

    def multiply(self, k):
        # Умножение точки на скаляр
        result = None
        addend = self

        while k:
            if k & 1:
                result = addend if result is None else result + addend
            addend = addend.double()
            k >>= 1

        return result

    def __str__(self):
        return f"({self.x}, {self.y})"

    def __eq__(self, other):
        return self.x == other.x and self.y == other.y and self.a == other.a
and self.b == other.b and self.p == other.p
```