Лабораторная работа № 4.

Реализация протокола Диффи-Хеллмана на эллиптических

Карасев Илья Алексеевич, М23-505

Цель работы, постановка задачи

Цель: изучение особенностей реализации криптографических протоколов распределения ключей, асимметричной криптографии на эллиптических кривых, разработка системы распределения криптографических ключей.

Задача: разработать программную реализацию метода Диффи-Хеллмана. Предусмотреть проверку эллиптической кривой по формуле (8.2). Исходными данными являются параметры кривой, координаты точки и секретные значения каждого участника обмена. Результат работы программы – координаты произведения точки G на число, которые должны совпасть у каждого из участников

Описание исходных данных

Выбран вариант № 3

a=2, b=3, p=97, G(x,y)=(3,6), k1=6, k2=10

Алгоритм работы программы

Текст программы

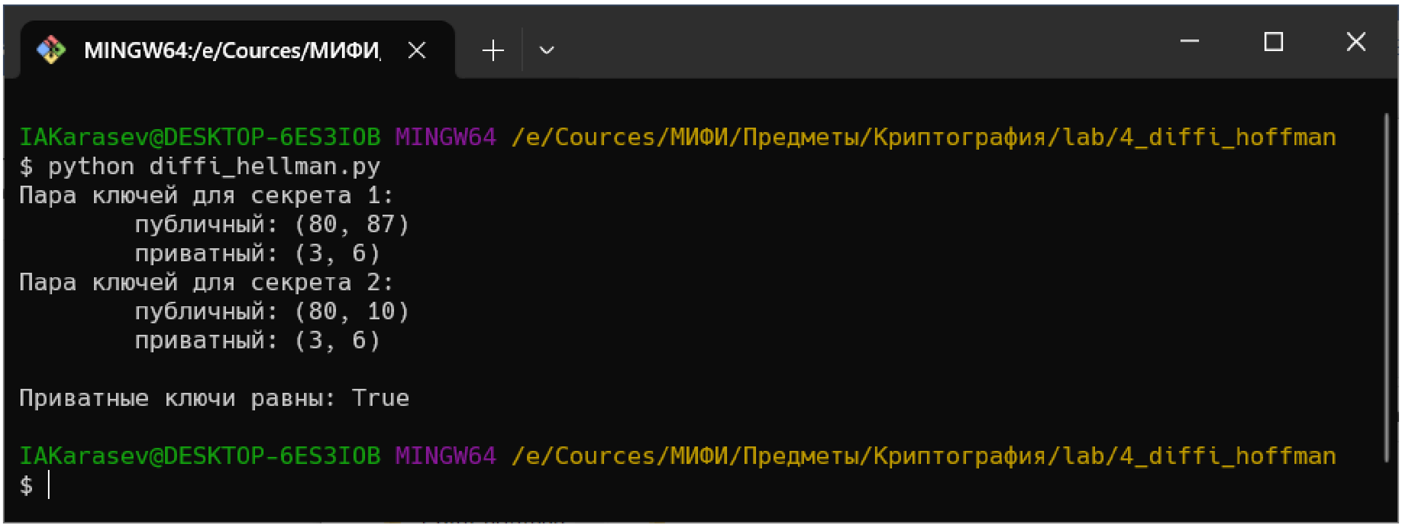
```
import argparse
import math
from typing import List


class Point:
    def __init__(self, x=0, y=0) -> None:
        self.x = x
        self.y = y

    @classmethod
    def zero(cls):
        return Point(0, 0)

    def __eq__(self, p1) -> bool:
        return self.x == p1.x and self.y == p1.y

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


class DiffiHellman:
    def __init__(self, a=2, b=3, p=97, gx=3, gy=6) -> None:
        # Проверка, что p - простое
        if p != 97 and not self.is_prime(p):
            raise ValueError(f"p must be prime number (given p = {p})")

        # Проверка a и b - могут быть параметрами эл. кривой
        if (a != 2 or b != 3) and not self.check_ab(a, b, p):
            raise ValueError(f"a and b are incorrect (a={a}, b={b})")

        self.a = a
        self.b = b
        self.p = p

        if (gx != 3 or gy != 6) and not self.check_point_on_curve(gx, gy):
            raise ValueError(
                f"Point G(x,y) is not on set curve (a={a}, b={b}, p={p}):G(x,y)=({gx},{gy})"
```

```
        )

    self.g = Point(gx, gy)

@classmethod
def check_ab(cls, a: int, b: int, p: int) -> bool:
    """Проверка параметров a и b для элиптической кривой

    Args:
        a (int): параметр a
        b (int): параметр b
        p (int):

    Returns:
        (bool): True - параметры подходят, иначе False
    """
    return 4 * pow(a, 3) + (27 * b * b) % p != 0

@classmethod
def is_prime(cls, n: int):
    """Проверка, что число простое

    Args:
        n (int): число

    Returns:
        (bool): True если простое, иначе False
    """

    if n <= 1:
        return False
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            return False
    return True

@classmethod
def get_bits(cls, n: int) -> List[int]:
    bits = []
    while n:
        bits.append(n & 1)
        n >>= 1
    bits.reverse()
    return bits

def check_point_on_curve(self, x: int, y: int) -> bool:
    """Проверка, что заданные координаты лежат на заданной элептической кривой

    Args:
        x (int): Координата x
        y (int): Координата y

    Returns:
        (bool): True - координаты на кривой, иначе False
    """

    y1 = (pow(x, 3) + self.a * x + self.b) % self.p
    return y * y == y1

def sum_points(self, p1: Point, p2: Point) -> Point:
    """Считает сумму двух точек

    Args:
        p1 (Point): точка 10
        p2 (Point): точка 2

    Returns:
        (Point): Результат суммы
    """
    if p1 == Point.zero():
        return p2

    if p2 == Point.zero():
        return p1

    if p1.x == p2.x:
        m = ((3 * p1.x * p1.x + self.a) * pow(2 * p1.y, -1, self.p)) % self.p
```

```python
        else:
            m = ((p1.y - p2.y) * pow(p1.x - p2.x, -1, self.p)) % self.p

        r = Point()
        r.x = (m * m - p1.x - p2.x) % self.p
        r.y = (m * (p1.x - r.x) - p1.y) % self.p
        return r

    def multiply_point(self, n: int, p1: Point):
        """Умножение точки на число методом сложения и удвоения

        Args:
            n (int): число-множитель
            p1 (Point): точка

        Returns:
            (Point) : результат умножения
        """
        q = Point.zero()

        for bit in self.get_bits(n):
            if bit == 1:
                q = self.sum_points(q, p1)
            p1 = self.sum_points(p1, p1)

        return q

    def gen_pub_key(self, secret: int):
        """Генерация серкретного ключа

        Args:
            secret (int): секрет

        Returns:
            (Point): публичный ключ
        """
        return self.multiply_point(secret, self.g)

    def gen_priv_key(self, secret: int, pub_key: Point):
        """Генерация закрытого ключа

        Args:
            secret (int):
            pub_key (point): публичный ключ

        Returns:
            (Point): закрытый ключ
        """
        return self.multiply_point(secret, pub_key)


parser = argparse.ArgumentParser()

parser.add_argument(
    "-k1",
    type=int,
    default=6,
    action="store",
    help="Секретный ключ 1",
)

parser.add_argument(
    "-k2",
    type=int,
    default=10,
    action="store",
    help="Секретный ключ 2",
)

args = parser.parse_args()

dh = DiffiHellman()

pubkey_1 = dh.gen_pub_key(args.k1)
pubkey_2 = dh.gen_pub_key(args.k2)

privkey_1 = dh.gen_priv_key(args.k1, pubkey_2)
```

```
privkey_2 = dh.gen_priv_key(args.k2, pubkey_1)

print(f"Пара ключей для секрета 1:\n\tпубличный: {pubkey_1}\n\tприватный: {privkey_1}")
print(f"Пара ключей для секрета 2:\n\tпубличный: {pubkey_2}\n\tприватный: {privkey_2}")

print(f"\nПриватные ключи равны: {privkey_1 == privkey_2}")
```

Результаты работы программы



```
MINGW64:/e/Cources/МИФИ

IAKarasev@DESKTOP-6ES3IOB MINGW64 /e/Cources/МИФИ/Предметы/Криптография/lab/4_diffi_hoffman
$ python diffi_hellman.py
Пара ключей для секрета 1:
        публичный: (80, 87)
        приватный: (3, 6)
Пара ключей для секрета 2:
        публичный: (80, 10)
        приватный: (3, 6)

Приватные ключи равны: True

IAKarasev@DESKTOP-6ES3IOB MINGW64 /e/Cources/МИФИ/Предметы/Криптография/lab/4_diffi_hoffman
$
```

Анализ результатов

Для обоих ключей были сгенерированы пары публичного и закрытого ключей. Закрытые ключи получились одинаковы, как это и требуется от протокола Диффи-Хеллмана на эллиптических ключах.

Контрольные вопросы

1. **Цель применения протокола Диффи-Хеллмана.**
   предназначен для безопасного обмена ключами между двумя сторонами в открытом канале связи. Его основной целью является обеспечение конфиденциальности передаваемой информации путем создания общего секретного ключа между двумя сторонами несмотря на то, что сам обмен данными может происходить в открытом виде
2. **Что представляет собой эллиптическая кривая?**
   Эллиптическая кривая — это математическая структура, определенная уравнением вида: $y^2 = x^3 + a * x + b$, где a и b удовлетворяют условию $4 * a^3 + 27 * b^2 \neq 0$
3. **Какие операции определены на эллиптической кривой при использовании в криптографических приложениях?**
   Сложение координат на эллиптической кривой, удвоение точки, умножение точки кривой на число
4. **Как выполнить умножение точки эллиптической кривой на число?**
   Координаты точки складываются сами с собой заданное количество раз. Для данной операции используют метод последовательного сложения и удвоения точки эллиптической кривой.
5. **Как вычислить число. Обратное к данному по заданному модулю?**
   Есть несколько алгоритмов, как прямого перебора чисел и их проверка, так и расширенный алгоритм Евклида и обобщение малой теоремы Ферма
6. **Что является нулем эллиптической кривой?**
   Это бесконечно удаленная тока, на которой сходятся все вертикальные прямые. Эта точка (O) обладает следущеми свойствами: (x,y) + O = (x,y); O + O = O; (x,y) + (x, -y) = O