Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

Факультет программной инженерии и компьютерных технологий

Отчет по лабораторной работе № 8

“Расчет точки nP на эллиптической кривой”

по дисциплине Информационная безопасность

Вариант 10

|  |  |
| --- | --- |
| Студент группы № P34151 | Шипулин Павел Андреевич |
| Преподаватель | Маркина Татьяна Анатольевна |

Санкт-Петербург

2024

# Цель работы

Дана точка на эллиптической кривой и натуральное число . Найти точку .

# Вариант задания

|  |  |  |
| --- | --- | --- |
| **№**  **варианта** |  |  |
| 10 | (78, 480) | 147 |

# Ход работы

1. Ознакомиться с теорией.
2. Получить вариант у преподавателя.
3. Найти точку .
4. Результаты и промежуточные вычисления оформить в виде отчета

# Листинг программ

Ссылка на репозиторий:

<https://github.com/PashcalE2/IS/tree/main/cryptography/second_part>

## Файл utils.py

def extended\_euclidean\_algorithm(a, b):s, old\_s = 0, 1  
 t, old\_t = 1, 0  
 r, old\_r = b, a  
  
 while r != 0:  
 quotient = old\_r // r  
 old\_r, r = r, old\_r - quotient \* r  
 old\_s, s = s, old\_s - quotient \* s  
 old\_t, t = t, old\_t - quotient \* t  
  
 return old\_r, old\_s, old\_t  
  
  
def inverse(n, p):gcd, x, y = extended\_euclidean\_algorithm(n, p)  
 assert (n \* x + p \* y) % p == gcd  
  
 if gcd != 1:  
 raise ValueError(  
 '{} has no multiplicative inverse '  
 'modulo {}'.format(n, p))  
 else:  
 return x % p  
  
  
\_p = 751  
  
  
class Point:  
 def \_\_init\_\_(self, x: int, y: int):  
 self.x = x  
 self.y = y  
  
 def \_\_add\_\_(self, other):  
 if self == other:  
 return self.double()  
  
 m = ((other.y - self.y) \* inverse(other.x - self.x, \_p)) % \_p  
 x = (m \* m - self.x - other.x) % \_p  
 y = (-self.y - m \* (x - self.x)) % \_p  
  
 return Point(x, y)  
  
 def double(self):  
 m = ((3 \* (self.x \*\* 2) - 1) \* inverse(2 \* self.y, \_p)) % \_p  
 x = (m \* m - 2 \* self.x) % \_p  
 y = (-self.y - m \* (x - self.x)) % \_p  
  
 return Point(x, y)  
  
 def mul(self, n: int):  
 P = self  
 nP = None  
  
 while n > 0:  
 if n % 2 == 1:  
 if nP is None:  
 nP = P  
 else:  
 nP = nP + P  
  
 P = P.double()  
  
 n >>= 1  
  
 return nP  
  
 def \_\_eq\_\_(self, other):  
 return (self.x == other.x) and (self.y == other.y)  
  
 def \_\_str\_\_(self):  
 return f"({self.x}, {self.y})"  
  
 def \_\_repr\_\_(self):  
 return self.\_\_str\_\_()

## Файл lab8.py

from utils import Point  
  
  
def lab8(Px: int, Py: int, n: int) -> Point:  
 P = Point(Px, Py)  
 print(f"P = {P}")  
  
 Ps = []  
 nP = None  
 nPs = []  
  
 N = 0  
 base = 1  
 i = 0  
 while n > 0:  
 if n % 2 == 1:  
 if nP is None:  
 nP = P  
 else:  
 nP = nP + P  
  
 N += base  
 nPs.append({"n": N, "nP": nP})  
 Ps.append({"pow": i, "P": P})  
  
 P = P.double()  
  
 n >>= 1  
 base <<= 1  
 i += 1  
  
 print(f"{nPs[-1]["n"]}P = {" + ".join([f"{2 \*\* v["pow"]}P" for v in Ps])} = {nP}")  
 for v in Ps:  
 print(f"{2 \*\* v["pow"]}P = {v["P"]}")  
  
 return nP  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 result = lab8(78, 480, 147)

# Выполнение

## Результат выполнения программы

P = (78, 480)

147P = 1P + 2P + 16P + 128P = (463, 15)

1P = (78, 480)

2P = (440, 212)

16P = (406, 354)

128P = (16, 416)

# Вывод

Ознакомился с идеей метода шифрования на основе эллиптических кривых. Реализовал алгоритм нахождения точки на эллиптической кривой.