Лабораторна робота 1 із Еліптичних Кривих

Виконали: Бондар Петро, Кістаєв Матвій (ФІ-42мн)

Мета: Отримання практичних навичок програмної реалізації арфиметики на еліптичних кривих, закріплення теоретичного матеріалу, отриманого на лекційних заняттях.

In [1]:
```
from epileptic import *
import random
import time
```

Деталі щодо програмної реалізації

Загальні відомості щодо інтерфейсу класів

У файлі epileptic.py наведено два класи для роботи із еліптичними кривими: EllipticCurve та EllipticCurvePoint.

Для створення об'єкту еліптичної кривої використовується стандартний конструктор, параметрами якого є значення простого модуля p та значень a, b кривої у формі Веєрштрасса ($y^2 \equiv x^3 + ax + b$).

Для створення об'єкту точки кривої використовується один із двох методів:

- Стандартний конструктор EllipticCurvePoint(): для створення точки за допомогою проективних кординат;
- Статичний метод EllipticCurvePoint.from_affine(): для створення точки за допомогою афінних координат.

Додаткові методи:

- EllipticCurve.rand_point(): створення випадкової точки на кривій шляхом розв'язку рівняння кривої у формі Веєрштрасса для випадково обраного x;
- EllipticCurve.inf(): отримання точки на нескінченності для певної кривої.

Деталі імплементації

Всі операції виконуються у проективних координатах для покращення ефективності.

Додавання та подвоєння: стандартний алгоритм додавання точок у проективних координатах.

Множення: використано алгоритм "сходи Монтгомері" (https://eprint.iacr.org/2011/338.pdf).

Обрана крива: P-224

Параметри кривої взято у відповідності зі стандартом на сайті https://neuromancer.sk/std/nist/P-224.

In [2]:
```
p = 0xffffffffffffffffffffffffffffffff000000000000000000000001
a = 0xfffffffffffffffffffffffffffffffefffffffffffffffffffffffe
b = 0xb4050a850c04b3abf54132565044b0b7d7bfd8ba270b39432355ffb4
n = 0xffffffffffffffffffffffffffff16a2e0b8f03e13dd29455c5c2a3d

EC = EllipticCurve(a, b, p)

G = EllipticCurvePoint.from_affine(0xb70e0cbd6bb4bf7f321390b94a03c1d356c21122343280d6115c1d21, 0xbd376388b5f723fb4c22dfe6cd437
```

Тести на загальні властивості

Наведено тести для наступного:

- Перевірка загальних властивостей точки на нескінченності.
- Перевірка властивостей додавання.
- Перевірка властивостей скалярного добутку.
- Перевірка групових властивостей точок на еліптичній кривій.
- Перевірка властивостей генератора.

In [3]:
```
def test_inf(P):
    O = P.curve.inf()
    assert P.curve.is_on_curve(O), "inf not on curve"
```

```
    assert P + O == P, "point + inf != point"
    assert O + P == P, "inf + point != point"
    assert P - O == P, "point - inf != point"
    assert O - P == -P, "inf - point != -point"
    assert O + O == O, "inf + inf != inf"
    assert O - O == O, "inf - inf != inf"


def test_peklo(P):
    s1 = random.randint(1, n)
    s2 = random.randint(1, n)
    sum = (s1 + s2) % n
    diff = (s1 - s2) % n
    point1 = s1 * P
    point2 = s2 * P
    result_sum = sum * P
    result_diff = diff * P

    assert (point1 + point2) == result_sum, f"[s1]P + [s2]P != [s1+s2]P"
    assert (point1 - point2) == result_diff, "[s1]P - [s2]P != [s1-s2]P"
    assert (point1 + point2) == (point2 + point1), "[s1]P + [s2]P != [s2]P + [s1]P"


def test_scalaro_producto(P):
    s = random.randint(2, 1000)
    point1 = s * P

    point2 = P
    for i in range(s - 1):
        point2 = point2 + P

    assert point1 == point2, "[s]P != P + P + ... + P"
    assert n * P == EC.inf(), "[n]P != inf"


def test_propitis(P):
    s1 = random.randint(1, n)
    s2 = random.randint(1, n)
    s3 = random.randint(1, n)

    point1 = s1 * P
```

```
    point2 = s2 * P
    point3 = s3 * P

    assert (point1 + point2) + point3 == point1 + (point2 + point3), "accociativity property failed"
    assert point1 + point2 == point2 + point1, "commutativity property failed"
    assert (s1 + s2) * P == s1 * P + s2 * P, "distributivity property failed"


def test_gen(gen, n):
    assert n * gen  == EC.inf(), "[n]G != inf"
    assert (n + 1) * gen == gen, "[n+1]G != G"
```

Запуск тестів для кривої P-224

Тести виконуються із випадковими точками 50 разів.

In [4]:
```
W = 4
N = 50
for i in range(1, N):
    print(f"Test({i}){" "*(W - len(str(i)))}: ".ljust(16), end="")
    k = random.randint(1, n)
    kP = k * G

    test_inf(kP)
    test_peklo(kP)
    test_scalaro_producto(kP)
    test_propitis(kP)

    print("Passed✅")

print()
print("Test [de]Generator", end=":\t")
test_gen(G, n)
print("Passed✅")
```

Test(1) : Passed ✅
Test(2) : Passed ✅
Test(3) : Passed ✅
Test(4) : Passed ✅
Test(5) : Passed ✅
Test(6) : Passed ✅
Test(7) : Passed ✅
Test(8) : Passed ✅
Test(9) : Passed ✅
Test(10) : Passed ✅
Test(11) : Passed ✅
Test(12) : Passed ✅
Test(13) : Passed ✅
Test(14) : Passed ✅
Test(15) : Passed ✅
Test(16) : Passed ✅
Test(17) : Passed ✅
Test(18) : Passed ✅
Test(19) : Passed ✅
Test(20) : Passed ✅
Test(21) : Passed ✅
Test(22) : Passed ✅
Test(23) : Passed ✅
Test(24) : Passed ✅
Test(25) : Passed ✅
Test(26) : Passed ✅
Test(27) : Passed ✅
Test(28) : Passed ✅
Test(29) : Passed ✅
Test(30) : Passed ✅
Test(31) : Passed ✅
Test(32) : Passed ✅
Test(33) : Passed ✅
Test(34) : Passed ✅
Test(35) : Passed ✅
Test(36) : Passed ✅
Test(37) : Passed ✅
Test(38) : Passed ✅
Test(39) : Passed ✅
Test(40) : Passed ✅
Test(41) : Passed ✅

```
Test(42)  :     Passed✅
Test(43)  :     Passed✅
Test(44)  :     Passed✅
Test(45)  :     Passed✅
Test(46)  :     Passed✅
Test(47)  :     Passed✅
Test(48)  :     Passed✅
Test(49)  :     Passed✅

Test [de]Generator:     Passed✅
```

Заміри часу виконання операцій додавання та множення на кривій P-224

In [5]:

```python
def benchmark_scalar_mult(N):
    general_time = 0
    for _ in range(N):
        P = EC.rand_point()
        k = random.randint(1, n)
        start = time.time()
        k * P
        end = time.time()
        general_time += (end - start)


    print(f"Time taken for {N} scalar multiplications: {general_time:.4f} seconds (~{general_time / N:.4f} seconds per multipl

def benchmark_addition(N):
    general_time = 0
    for i in range(N):
        P = EC.rand_point()
        Q = EC.rand_point()
        start = time.time()
        P + Q
        end = time.time()
        general_time += (end - start)


    print(f"Time taken for {N} addition: {general_time*1000:.6f} ms (~{general_time*1000 / N:.6f} ms per multiplication)")
```

In [6]:
```
benchmark_scalar_mult(1000)
benchmark_addition(1000)
```

```
Time taken for 1000 scalar multiplications: 3.7748 seconds (~0.0038 seconds per multiplication)
Time taken for 1000 addition: 8.907318 ms (~0.008907 ms per multiplication)
```