МОЛДАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАТИКИ

ДЕПАРТАМЕНТ ИНФОРМАТИКИ

“Криптография и безопасность информации”

**Индивидуальная работа**

Тема: **Эллиптические кривые,** **Эль-Гамаль**

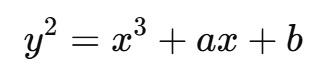
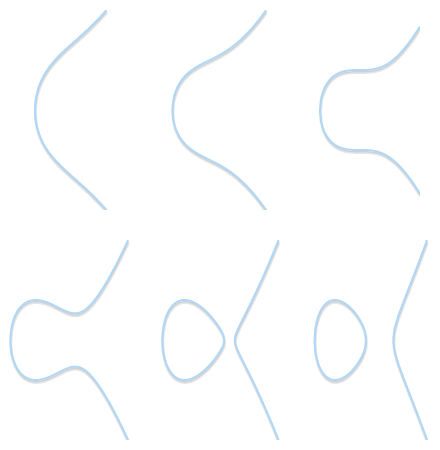
**Автор:** Калинкова С.

группа I2302

**Проверила:** Чербу О.

Кишинев, 2024

**Задание 1: Эллиптические кривые**

**Эллиптические кривые** — это кривые, описываемые уравнением:  
где a и b — параметры, а x и y — переменные. Эти кривые имеют важное значение в теории чисел и криптографии.  


### **1. Основные свойства**

* **Группа точек**: Все точки на кривой с операцией сложения образуют группу. Это значит, что можно "складывать" точки на кривой и получить другие точки на ней.
* **Операция сложения**: Если у нас есть две точки на кривой, мы можем сложить их. Это делается так, как если бы мы проводили прямую через эти точки и искали пересечение этой прямой с кривой.
* **Точка на бесконечности**: Это специальная точка, которая работает как "ноль" в сложении точек на кривой.

### **2. Применение в криптографии**

Эллиптические кривые используются в криптографии благодаря высокой безопасности при малых размерах ключей. Они применяются в таких алгоритмах, как **ECDSA** для цифровых подписей и **ECDH** для обмена ключами.

* **Устойчивость к вычислениям**: Операции на эллиптических кривых трудны для обращения, что делает их подходящими для криптографических схем.
* **Криптосистемы**: Включают алгоритмы на базе эллиптических кривых, которые обеспечивают безопасность при меньших размерах ключей по сравнению с традиционными методами, такими как RSA.

### **3. Эллиптические кривые в конечных полях**

Когда поля конечны (например, Zp​), такие кривые применяются в криптографии. Операции выполняются по модулю простого числа p.

### **4. Преимущества эллиптических кривых**

* **Меньшие размеры ключей**: Для того чтобы обеспечить такую же безопасность, как у других методов, например, RSA, для эллиптических кривых требуется меньший размер ключа.
* **Высокая безопасность**: Задачи, такие как вычисление логарифмов на эллиптических кривых, сложны, что повышает устойчивость криптографических систем.  
    
  **Ход работы:**

Строим эллиптическую кривую посредством следующего уравнения **y2=ax3 + bx + 1**, где

*a = 1 и b = 1*.

Находим область определения эллиптической кривой (ЭК):

Пусть p = 5, значит поле Z5,

Проверим: 4 \* a3 + 27 \* b (mod p)= 4 + 27 (mod 5) = 31 mod 5= 1 ≠ 0.

Значит уравнение ЭК — **y2 = x3 + x + 1** на поле **Z5.**

Вычисляем *квадратичные вычеты* для нахождения точек на эллиптической прямой:

*for a:=0 to p-1 do*

*a2:= a2 mod p*

| a | a2 |
| --- | --- |
| 0 | 02 mod 5=0 |
| 1 | 12 mod 5=1 |
| 2 | 22 mod 5=4 |
| 3 | 32 mod 5=4 |
| 4 | 42 mod 5=1 |

Вычисляем точки E:

*for x:=0 to p-1 do*

*z:=(x3+x+1) mod 5*

x = 0 => z = x3 + x + 1 (mod 5) = 03 + 0 + 1 (mod 5) = 1 mod 5 = 1;

x = 1 => z = x3 + x + 1 (mod 5) = 13 + 1 + 1 (mod 5) = 3 mod 5 = 3;

x = 2 => z = x3 + x + 1 (mod 5) = 23 + 2 + 1 (mod 5) = 11 mod 5 = 1;

x = 3 => z = x3 + x+ 1 (mod 5) = 33 + 3 + 1 (mod 5) = 31 mod 5 = 1;

x = 4 => z = x3 + x+ 1 (mod 5) = 43 + 4 + 1 (mod 5) = 69 mod 5 = 4;

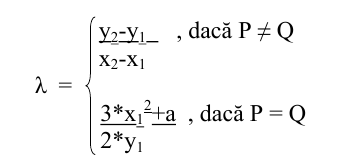
Составляем таблицу и заполняем её: поля **x** и **z** наполняем, в соответствии с выше произведенными вычислениями. В случае с **y** — ищем каждое значение **z** среди значений **a2** и, в случае, если они равны, заполняем в поле **y** все значения **a**:

| x | z=x3+x+1 | y |
| --- | --- | --- |
| 0 | 1 | 1,4 |
| 1 | 3 | - |
| 2 | 1 | 1, 4 |
| 3 | 1 | 1, 4 |
| 4 | 4 | 2,3 |

По данным из таблицы у нас имеется 8 точки, а также точка бесконечности O. Ниже приведены эти 8 точки:

(0, 1), (0, 4), (2, 1), (2, 4), (3, 1), (3, 4), (4, 2), (4, 3)

Давайте начнем с точки (3,1) и вычислим степени этой точки для эллиптической кривой y2=x3+x+1 в поле Z5



Отсюда имеем:

xi = λ2 - xi-2 - xi-1

yi = λ (xi-2 - xi)-yi-1, где i= 3 ,4 ,…, количество точек-1

**Выполняем расчёты:**

# Начальная точка: P = (3, 1) **Шаг 1: Вычисление 2α для точки (3, 1)**

Поскольку P = Q, то вычисление λ по второй формуле:

λ = (3 \* 3^2 + 1) / 2 \* 1 (mod 5) = (3 \* 9 + 1) / 2 (mod 5) = (27 + 1) / 2 (mod 5) = 28 / 2 (mod 5)= 14 (mod 5) = 14 mod 5 = 4

Теперь вычислим x и y для 2α:

x2 = λ^2 - x1 - x1 = 4^2 - 3 - 3 (mod 5) = 16 - 6 (mod 5) = 10 (mod 5) = 0

y2 = λ(x1 - x2) - y1 = 4 \* (3 - 0) - 1 = 4 \* 3 - 1 = 12 - 1 = 11 (mod 5) = 11 mod 5 = 1

Ответ: 2α = (0, 1)

# **Шаг 2: Вычисление 3α для точки (0, 1)**

Теперь вычислим 3α = 2α + α = (0, 1) + (3, 1)

Поскольку P ≠ Q, то λ вычисляется по первой формуле:

λ = (y2 - y1) / (x2 - x1) = (1 - 1) / (0 - 3) = 0 / -3 mod 5 = 0

Теперь вычислим x и y для 3α:

x3 = λ^2 - x1 - x2 = 0^2 - 0 - 3 = -3 (mod 5) = 2

y3 = λ(x1 - x3) - y1 = 0 \* (0 - 2) - 1 = 0 - 1 = -1 (mod 5) = 4

Ответ: 3α = (2, 4)

# **Шаг 3: Вычисление 4α для точки (2, 4)**

Теперь вычислим 4α = 3α + α = (2, 4) + (3, 1)

Поскольку P ≠ Q, то λ вычисляется по первой формуле:

λ = (y2 - y1) / (x2 - x1) = (4 - 1) / (2 - 3) = 3 / -1 (mod 5)= 2

Теперь вычислим x и y для 4α:

x4 = λ^2 - x1 - x2 = 2^2 - 2 - 3 = -1 (mod 5) = 4

y4 = λ(x1 - x4) - y1 = 2 \* (3 - 4) - 1 = 2 \* (-1) - 1 = -2 - 1 = -3 (mod 5) = 2

Ответ: 4α = (4, 2)

# **Шаг 4: Вычисление 5α для точки (4, 2)**

Теперь вычислим 5α = 4α + α = (4, 2) + (3, 1)

Поскольку P ≠ Q, то λ вычисляется по первой формуле:

λ = (y2 - y1) / (x2 - x1) = (2 - 1) / (4 - 3) = 1 / 1 = 1 (mod 5) = 1

Теперь вычислим x и y для 5α:

x5 = λ^2 - x1 - x2 = 1^2 - 3 - 4 = 1 - 7 = -6 (mod 5) = 4

y5 = λ(x1 - x5) - y1 = 1 \* (3 - 4) - 1 = -2 (mod 5) = 3

Ответ: 5α = (4, 3)

# **Шаг 5: Вычисление 6α для точки (4, 3)**

Теперь вычислим 6α = 5α + α = (4, 3) + (3, 1)

Поскольку P ≠ Q, то λ вычисляется по первой формуле:

λ = (y2 - y1) / (x2 - x1) = (3 - 1) / (4 - 3) = 2 / 1 = 2 (mod 5) = 2

Теперь вычислим x и y для 6α:

x5 = λ^2 - x1 - x2 = 2^2 - 3 - 4 = 4 - 7 = -3 (mod 5) = 2

y5 = λ(x1 - x5) - y1 = 2 \* (3 - 2) - 1 = 2 -1 (mod 5) = 1

Ответ: 6α = (2, 1)  
  
**Шаг 6: Вычисление 7α для точки (2, 1)**

Теперь вычислим 7α = 6α + α = (2, 1) + (3, 1)

Поскольку P ≠ Q, то λ вычисляется по первой формуле:

λ = (y2 - y1) / (x2 - x1) = (1 - 1) / (2 - 3) = 0 / -1 = 0 (mod 5) = 0

Теперь вычислим x и y для 7α:

x5 = λ^2 - x1 - x2 = 0^2 - 3 - 2 = -5 (mod 5) = 0

y5 = λ(x1 - x5) - y1 = 0 \* (3 - 0) - 1 = 0 -1 (mod 5) = 4

Ответ: 7α = (0, 4)  
  
**Шаг 7: Вычисление 8α для точки (0, 4)**

Теперь вычислим 8α = 7α + α = (0, 4) + (3, 1)

Поскольку P ≠ Q, то λ вычисляется по первой формуле:

λ = (y2 - y1) / (x2 - x1) = (4 - 1) / (0 - 3) = 3 / -3 (mod 5) = 4

Теперь вычислим x и y для 8α:

x5 = λ^2 - x1 - x2 = 4^2 - 3 - 0 = 16 - 3 = 13 (mod 5) = 3

y5 = λ(x1 - x5) - y1 = 4 \* (3 - 3) - 1 = 0 -1 (mod 5) = 4

Ответ: 8α = (3, 4)

**Шаг 8: Вычисление 9α для точки (3, 4)**

Теперь вычислим 8α = 7α + α = (3, 4) + (3, 1)

Поскольку P ≠ Q, то λ вычисляется по первой формуле:

λ = (y2 - y1) / (x2 - x1) = (4 - 1) / (3 - 3) = 3 / 0   
Это деление на ноль, что означает, что вычисление завершено, и мы пришли к точке бесконечности O. Таким образом, после 8 шагов мы возвращаемся к точке бесконечности.  
  
Реализация на Python  
# Функция для проверки принадлежности точки кривой

def is\_point\_on\_curve(P, a, b, p):

x, y = P

return (y\*\*2) % p == (x\*\*3 + a \* x + b) % p

# Функция для удвоения точки P на эллиптической кривой

def point\_double(P, a, p):

x1, y1 = P

if y1 == 0:

return None

λ = (3 \* x1\*\*2 + a) \* pow(2 \* y1, p - 2, p) % p

x3 = (λ\*\*2 - 2 \* x1) % p

y3 = (λ \* (x1 - x3) - y1) % p

return (x3, y3)

# Функция для сложения двух точек P и Q на эллиптической кривой

def point\_add(P, Q, p):

x1, y1 = P

x2, y2 = Q

if P == Q:

return point\_double(P, a, p)

if x1 == x2:

return None

λ = (y2 - y1) \* pow(x2 - x1, p - 2, p) % p

x3 = (λ\*\*2 - x1 - x2) % p

y3 = (λ \* (x1 - x3) - y1) % p

return (x3, y3)

# Функция для умножения точки P на скаляр k (kP)

def point\_multiply(P, k, a, p):

R = None

Q = P

while k:

if k % 2 == 1:

if R is None:

R = Q

else:

R = point\_add(R, Q, p)

Q = point\_double(Q, a, p)

k //= 2

return R

# Пример использования

a = 1

b = 1

p = 5 # Поле Z5

# Точки на кривой

points = [(0, 1), (0, 4), (2, 1), (2, 4), (3, 1), (3, 4), (4, 2), (4, 3)]

# Вывод точек на кривой

print(f"Точки на кривой: {points}")

# Проверяем принадлежность каждой точки кривой

for point in points:

print(f"Принадлежит ли точка {point} кривой? {is\_point\_on\_curve(point, a, b, p)}")

# Для каждой точки вычисляем 2P, 3P, ..., 8P

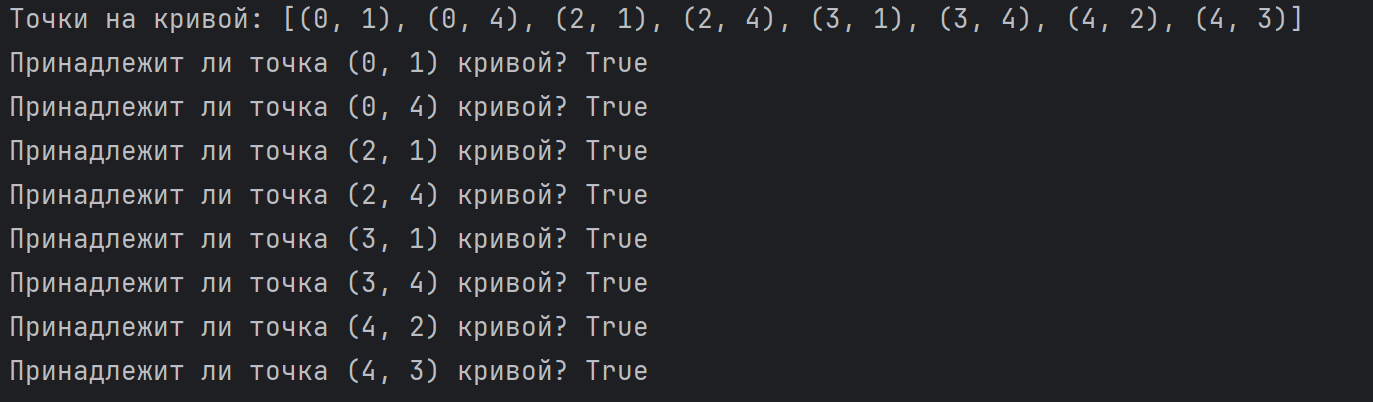
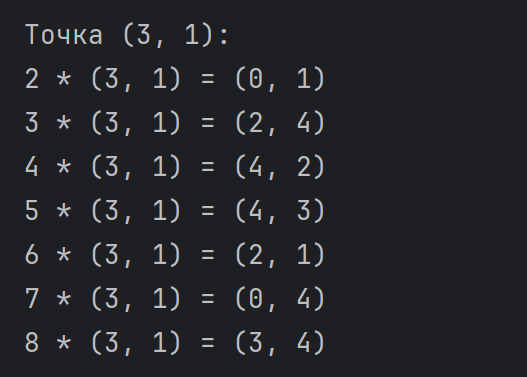
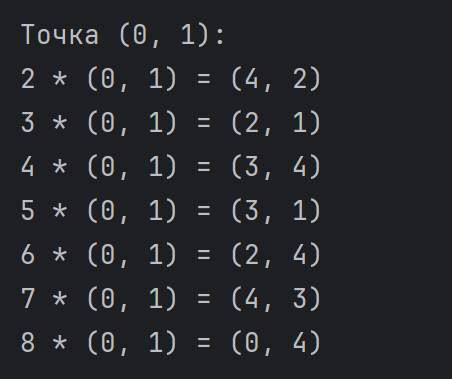
for point in points:

print(f"\nТочка {point}:")

for k in range(2, 9):

result = point\_multiply(point, k, a, p)

print(f"{k} \* {point} = {result}")

**Вывод программы  
  
  
   
Мы видим, что вывод совпадает с расчетами для точки (3,1)**

**Задание 2: Шифрование «Эль-Гамаль»**  
Шифрование Эль-Гамаля — это алгоритм асимметричного шифрования, основанный на задаче дискретного логарифма.

1. **Генерация ключей**: публичный ключ — пара чисел (g,y)(g, y)(g,y), где y=gxmod  py = g^x \mod py=gxmodp, а секретный ключ — число xxx.
2. **Шифрование**: для сообщения mmm выбирается случайное kkk, затем вычисляются c1=gkmod  pc\_1 = g^k \mod pc1​=gkmodp и c2=m⋅ykmod  pc\_2 = m \cdot y^k \mod pc2​=m⋅ykmodp.
3. **Дешифрование**: получатель вычисляет s=c1xmod  ps = c\_1^x \mod ps=c1x​modp и восстанавливает m=c2⋅s−1mod  pm = c\_2 \cdot s^{-1} \mod pm=c2​⋅s−1modp.

Это обеспечивает конфиденциальность, так как задача нахождения дискретного логарифма сложна.

**Ход работы:**

В процессе шифрования и дешифрования все коэффициенты вычисляются посредством модуля, равному кол-ву точек — **mod 9.**

р = 5, α=(3,1) Точка.

Берём число a =7 (Закрытый ключ).

Вычисляем β = a \* α = 7α.

**Шифрование:**

eK(M, k) = (k \* α , M + k \* β), где M принадлежит кривой E, 0 ≤ k ≤ число точек минус 1 => **0 ≤ k ≤ 8**

Выбираем сообщение для шифрования, которое должно быть точкой на кривой E:

Пусть M = (2, 1) = 6α.

Выбираем **k = 3**. Вычисляем:

y1 = k \* α = 3 \* α = 3α

y2 = M + k \* β = 6α + 3 \* 7α = 6α + 21α = 27α = (27 mod 9) α = 0α

Получили:

у = (y1, y2) = (3α, 0α).

*Отправляем результат user\_2.*

**Дешифрация**

dK (y1, y2) = y2 – a \* y1, a = 7.

M = y2 – 7 \* y1 = 0α – 7\*3α = - 21α = (-21 mod 9)α = 6a

Дешифрация выполнена.

**Генерация ключей для цифровой подписи**

Рассмотрим аналогичную ЭК с уравнением **y2 = x3 + x + 1** на поле **Z5,** n = 9.

Выбираем точку P принадлежащую кривой: P = 6α = (2, 1).

Выбираем следующее число d, 1≤d≤n-1: пусть d = 5.

Вычисляем точку Q = d \* P = 5 \* 6α = 30α = (30 mod 9)α = 3α = (2,4).

Закрытый ключ d = 3, открытый ключ (E, P, n, Q) = (y2 = x3 + x + 1, (2, 1), 9, (2,4)).

**Генерация цифровой подписи**

Выберем случайное число k, 1≤k≤n-1, пусть k = 5.

Вычислим k \* P = 5 \* 6α = 30α = (30 mod 9)α = 3α = (2, 4) = (x1, y1).

Вычисляем r = x1 mod n = 2 mod 9 = 2 (если r = 0 выбираетсяs другое k).

Вычисляем k-1 mod n = 5-1 mod 9 = 2.

Вычисляем s = k-1 (hash (M) + d \* r) (mod n). Пусть hash(M) = 4 =>

s = 6-1 (4 + 3 \* 5) (mod 9) = 4 mod 9 = 4 (если s = 0 выбирается другое k).

***Подпись сообщения M — (r, s) = (2, 4).***

**Проверка подписи**

Проверим, что [r и s] входят в интервал [1, n-1], иначе *подпись не верна.*

Вычисляем w = s-1 mod n = 4-1 mod 9 = 7.

Пусть hash(M) = 4

u1 = hash(M) \* w (mod n) = 4 \* 7 mod 9 = 28 mod 9 = 1.

u2 = r \* w mod n = 5 \* 7 mod 9 = 8.

u1P + u2Q = 1 \* 6α + 8 \* 9α = 78α = (78 mod 9)α = 6α = (2,1) = (x0,y0).

v = x0 = **r** = 2 —> **подпись валидна.**

**Обмен ключами при помощи Диффи-Хелман**

Рассмотрим аналогичную ЭК с уравнением **y2 = x3 + x + 1** при **Z5** и **α = (3,1).**

E и α — открыты.

User A выбирает закрытый ключ nA, который < количества точек — nA = 3.

Открытый ключ вычисляется по формуле PA= nA\* α = 3α.

User B выбирает закрытый ключ nB, который < количества точек — nB = 2.

Открытый ключ вычисляется по формуле PB= nB\* α = 2α.

Пользователи A и B обмениваются открытыми ключами и вычисляют общий ключ K следующим образом:

A: K = nA \* PB = 3 \* 2α = 6α = (6 mod 9)α = α.

B: K = nB \* PA = 2 \* 3α = 6α = (6 mod 9)α = α.

***Получаем общий ключ K = α.***

Реализация на Python  
  
# Функция для умножения точки на число по модулю p

def multiply\_point(point, n, p):

x, y = point

x\_n = (x \* n) % p # Умножение компоненты на число по модулю p

y\_n = (y \* n) % p # Умножение компоненты на число по модулю p

return (x\_n, y\_n)

# Функция для сложения точек на эллиптической кривой по модулю p

def add\_points(P1, P2, p):

x1, y1 = P1

x2, y2 = P2

# Простое сложение точек для примера

x = (x1 + x2) % p

y = (y1 + y2) % p

return x, y

# Шифрование

def encrypt(alpha, beta, M, k, p):

y1 = multiply\_point(alpha, k, p) # y1 = k \* α

y2 = add\_points(M, multiply\_point(beta, k, p), p) # y2 = M + k \* β

return y1, y2

# Дешифрование

def decrypt(private\_key, y1, y2, p):

a\_y1 = multiply\_point(y1, private\_key, p) # a \* y1

M = subtract\_points(y2, a\_y1, p) # y2 - a \* y1

return M

# Функция для вычитания точек по модулю p

def subtract\_points(P1, P2, p):

x1, y1 = P1

x2, y2 = P2

x = (x1 - x2) % p

y = (y1 - y2) % p

return x, y

# Параметры

p = 9 # Модуль

alpha = (3, 1) # Точка α

a = 7 # Закрытый ключ

# Генерация публичного ключа

beta = multiply\_point(alpha, a, p) # β = a \* α

# Сообщение M = (2, 1) - вы даете это конкретное значение

M = (2, 1) # Сообщение для шифрования

# Шифрование

k = 3 # Выбираем случайное k

y1, y2 = encrypt(alpha, beta, M, k, p)

# Дешифрование

decrypted\_message = decrypt(a, y1, y2, p)

# Вывод

print(f"Точка α: {alpha}")

print(f"Публичный ключ (β): {beta}")

print(f"Сообщение M: {M}")

print(f"Шифрованное сообщение (y1, y2): {y1}, {y2}")

print(f"Расшифрованное сообщение: {decrypted\_message}")

**вывод программы**   
