Параметры кривой:

curve = EllipticCurve(

# Field characteristic.

p=59,

# Curve coefficients.

a=131,

b=3,

# Base point.

g=(38, 26),

# Subgroup order.

n=37,

# Subgroup cofactor.

h=2,

)

Генерация элементов эллиптической группы:

def get\_point\_from\_eleptic\_curve():

curve = ElepticalCurve(59, 131, 3)

points = []

for x in range(curve.m):

y = curve.res(x)

points.extend(zip([x, x], y))

group\_order = len(points) + 1

selected\_point = random.randint(0, len(points) - 1)

print(f'All points: {points}')

print(f'Group order: {group\_order}')

Обмен ключами:

def make\_keypair():

private\_key = random.randrange(1, curve.n)

public\_key = scalar\_mult(private\_key, curve.g)

return private\_key, public\_key

alice\_private\_key, alice\_public\_key = make\_keypair()

print("Alice's private key:", hex(alice\_private\_key))

print("Alice's public key: (0x{:x}, 0x{:x})".format(\*alice\_public\_key))

bob\_private\_key, bob\_public\_key = make\_keypair()

print("Bob's private key:", hex(bob\_private\_key))

print("Bob's public key: (0x{:x}, 0x{:x})".format(\*bob\_public\_key))

s1 = scalar\_mult(alice\_private\_key, bob\_public\_key)

s2 = scalar\_mult(bob\_private\_key, alice\_public\_key)

assert s1 == s2

print('Shared secret: (0x{:x}, 0x{:x})'.format(\*s1))

Цифровая подпись:

def sign\_message(private\_key, message):

z = hash\_message(message)

r = 0

s = 0

while not r or not s:

k = random.randrange(1, curve.n)

x, y = scalar\_mult(k, curve.g)

r = x % curve.n

s = ((z + r \* private\_key) \* inverse\_mod(k, curve.n)) % curve.n

return r, s

def verify\_signature(public\_key, message, signature):

z = hash\_message(message)

r, s = signature

w = inverse\_mod(s, curve.n)

u1 = (z \* w) % curve.n

u2 = (r \* w) % curve.n

x, y = point\_add(scalar\_mult(u1, curve.g),

scalar\_mult(u2, public\_key))

if (r % curve.n) == (x % curve.n):

return 'signature matches'

else:

return 'invalid signature'

private, public = make\_keypair()

print("Private key:", hex(private))

print("Public key: (0x{:x}, 0x{:x})".format(\*public))

msg = b'Hello!'

signature = sign\_message(private, msg)

print()

print('Message:', msg)

print('Signature: (0x{:x}, 0x{:x})'.format(\*signature))

print('Verification:', verify\_signature(public, msg, signature))

msg = b'Hi!'

print()

print('Message:', msg)

print('Verification:', verify\_signature(public, msg, signature))

private, public = make\_keypair()

msg = b'Hello!'

print()

print('Message:', msg)

print("Public key: (0x{:x}, 0x{:x})".format(\*public))

print('Verification:', verify\_signature(public, msg, signature))