Элиптические кривые. ECDSA

Лабораторная работа №3

Доскоч Роман 4 курс 13 группа

In [19]:

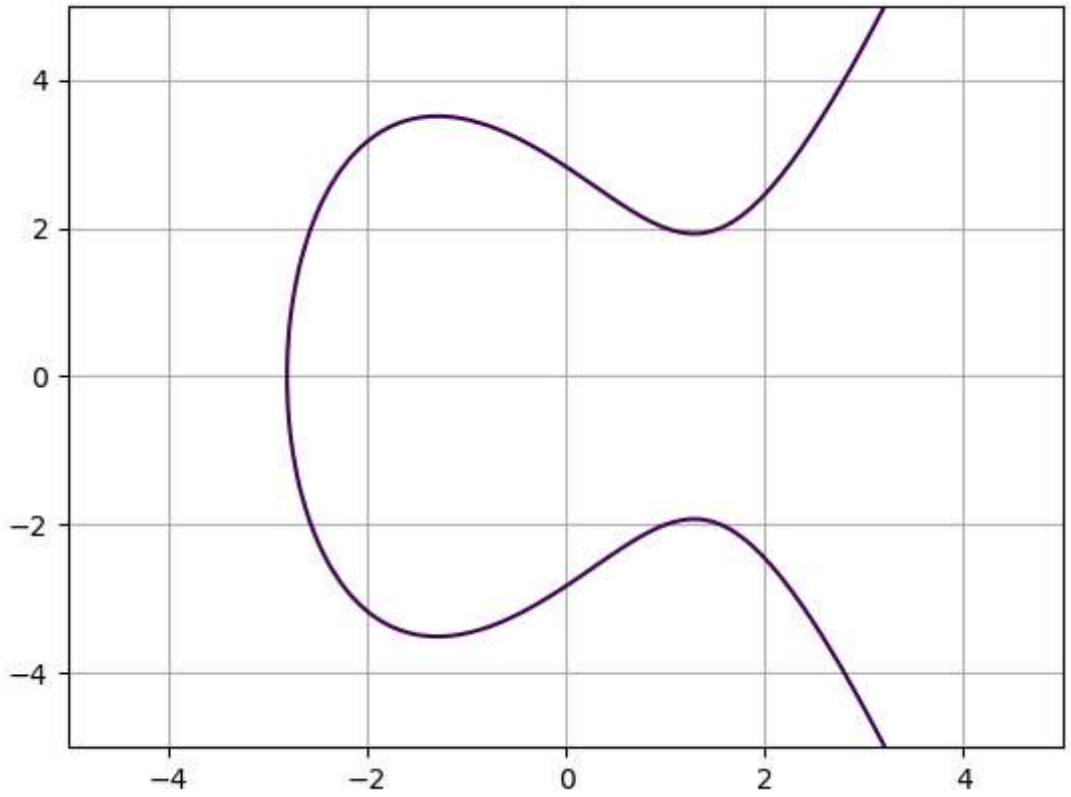
```
import numpy as np
import matplotlib.pyplot as plt
import hashlib
```

In [3]:

```
A,B = -5,8
M = 37
ECurve = lambda x,y: y ** 2 - (x ** 3 + x * A + B)
```

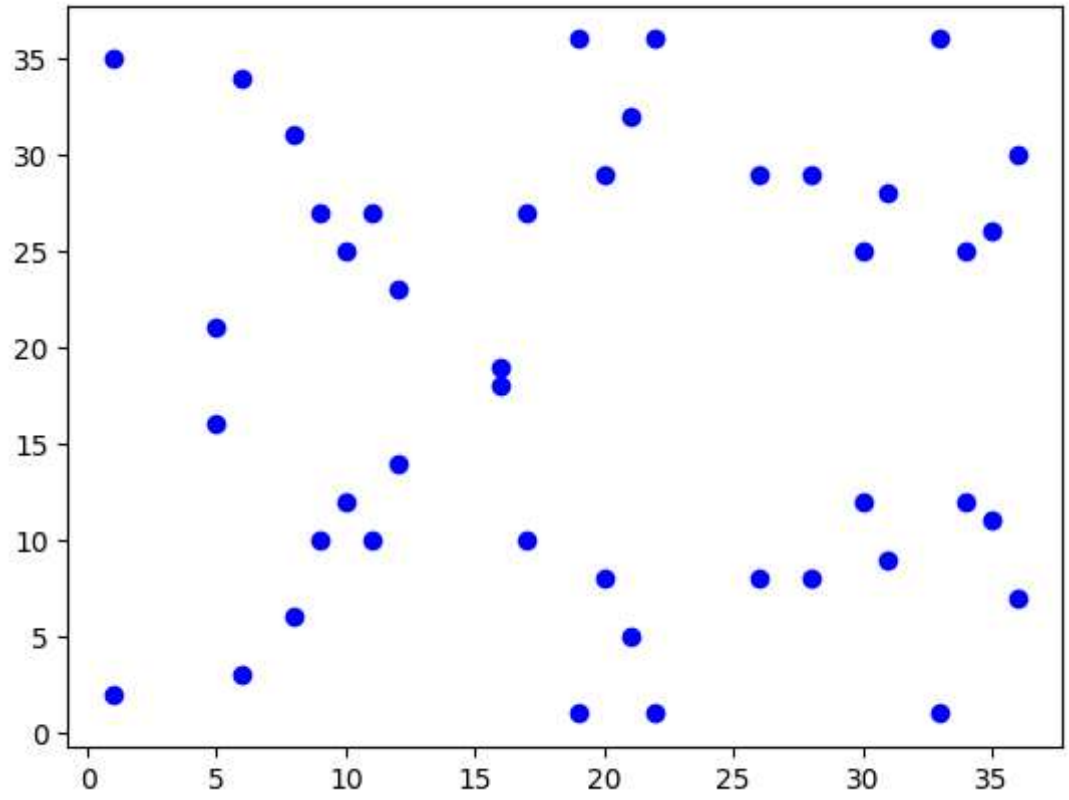
In [4]:

```
y, x = np.ogrid[-5:5:100j, -5:5:100j]
plt.contour(x.ravel(), y.ravel(), ECurve(x,y) , [0])
plt.grid()
```



In [5]:

```
def ecc(x):
    assert (4*A**3 + 27*B**2) % M!= 0
    return (x**3 + A*x + B) % M

def sqrt_f(x, y2):
    x2 = x**2 % M
    y = [(i, *y_i) for i, y_i in enumerate([np.where(y2_i == x2)[0] for y2_i in y2]) if y_i
    return y

x = np.array(range(0, M))
y = sqrt_f(x, ecc(x))

fig = plt.figure(dpi=100)
for y_p in y:
    [plt.scatter(y_p[0], i, c='b') for i in y_p[1:]]
```



Количество точек в группе $E_M(a, b)$

In [6]:

```
len(sum([y_p[1:] for y_p in y], ()))
```

Out[6]:

44

Расширенный Алгоритм Евклида

In [7]:

```
def eea(x, p):
    q = int(p / x)

    r = r_old = p % x

    q_s = [q]
    a_s = [0, 1]
    i = 0
    while True:
        if i > 1:
            a_i = (a_s[i-2] - a_s[i-1] * q_s[i-2]) % p
            a_s.append(a_i)
        if r == 0:
            if r_old != 1:
                assert "No inverse"
            i = i + 1
            break
        r_old = r
        q = int(x / r)
        q_s.append(q)
        r = x % r
        x = r_old
        i = i + 1

    a_i = (a_s[i-2] - a_s[i-1] * q_s[i-2]) % p
    return a_i
```

Алгебра над полем конечных элиптических кривых

In [8]:

```
def get_2P(P):
    x,y = P

    lam = (3 * x**2 + A) * eea(2 * y, M) % M
    nu = (-x**3 + A*x + 2*B) * eea(2 * y, M) % M

    x3 = (lam**2 - 2*x) % M
    y3 = (-lam**3 + 2*lam*x - nu) % M

    return x3, y3

def PQ_Helper(P, Q):
    x1,y1 = P
    x2,y2 = Q

    lam = (y2 - y1) * eea((x2 - x1) % M, M) % M
    nu = (y1*x2 - y2*x1) * eea((x2 - x1) % M, M) % M

    x3 = (lam**2 - x1 - x2) % M
    y3 = (-lam**3 + lam*(x1 + x2) - nu) % M

    return x3, y3

def mull_PQ(P, Q):
    if (P == 0):
        return Q
    if (Q == 0):
        return P
    if (P == Q):
        if P[0] == 0:
            return 0
        val = get_2P(P)
        return val
    else:
        if P[0] == Q[0]:
            return 0

        val = PQ_Helper(P, Q)
        return val
```

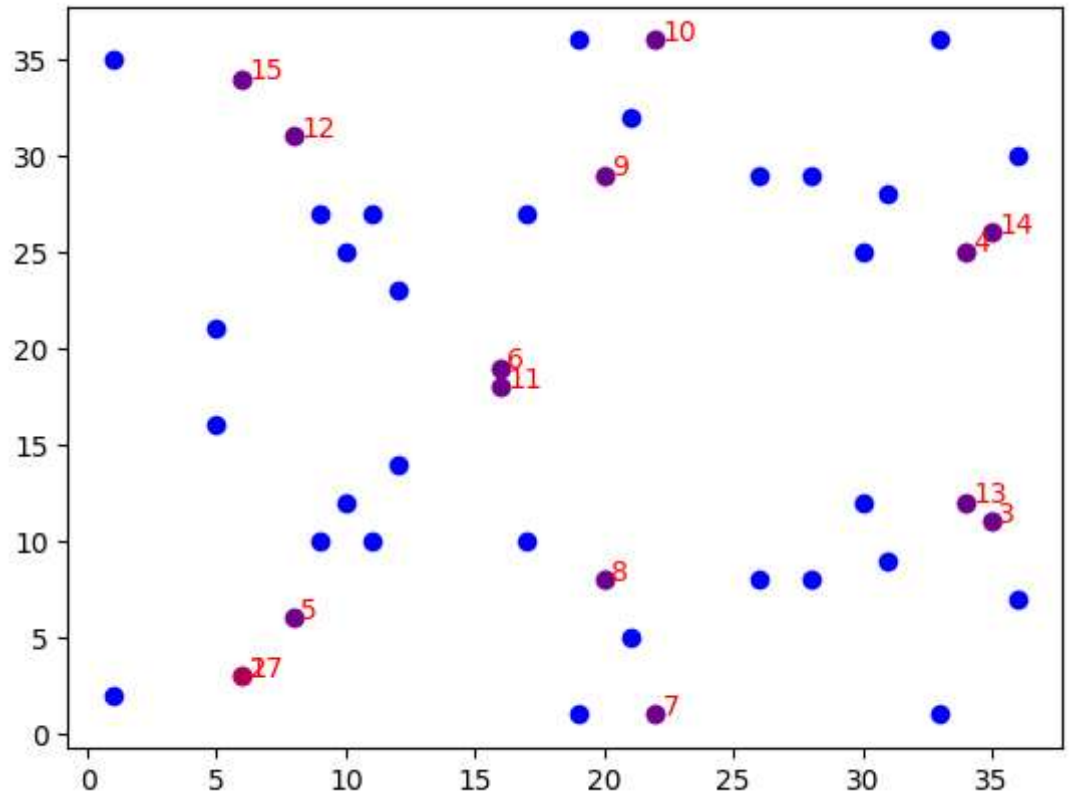
Вычисление точки kP

In [9]:

```
def kP(k, P):
    m_binary = "{:b}".format(k)

    doubling_P = list(range(len(m_binary)))
    doubling_P[0] = P
    for i in doubling_P[1:]:
        doubling_P[i] = mull_PQ(doubling_P[i-1], doubling_P[i-1])

    # Get all non-zero m_i * 2^i P
    to_compute = [doubling_P[i] for i in range(len(m_binary)) if m_binary[::-1][i] == '1']

    # Get the sum
    result = to_compute[0]
    for P_2i in to_compute[1:]:
        result = mull_PQ(result, P_2i)
    return result
```

In [10]:

```
fig = plt.figure(dpi=100)

for y_p in y:
    [plt.scatter(y_p[0], i, c='b') for i in y_p[1:]]

P = (6,3)
for k in range(1,17):
    R = kP(k, P)
    if R == 0:
        continue
    plt.scatter(*R, c='r', alpha=0.4)
    plt.annotate(f'{k+1}', (R[0]+0.25, R[1]), c='r')
```

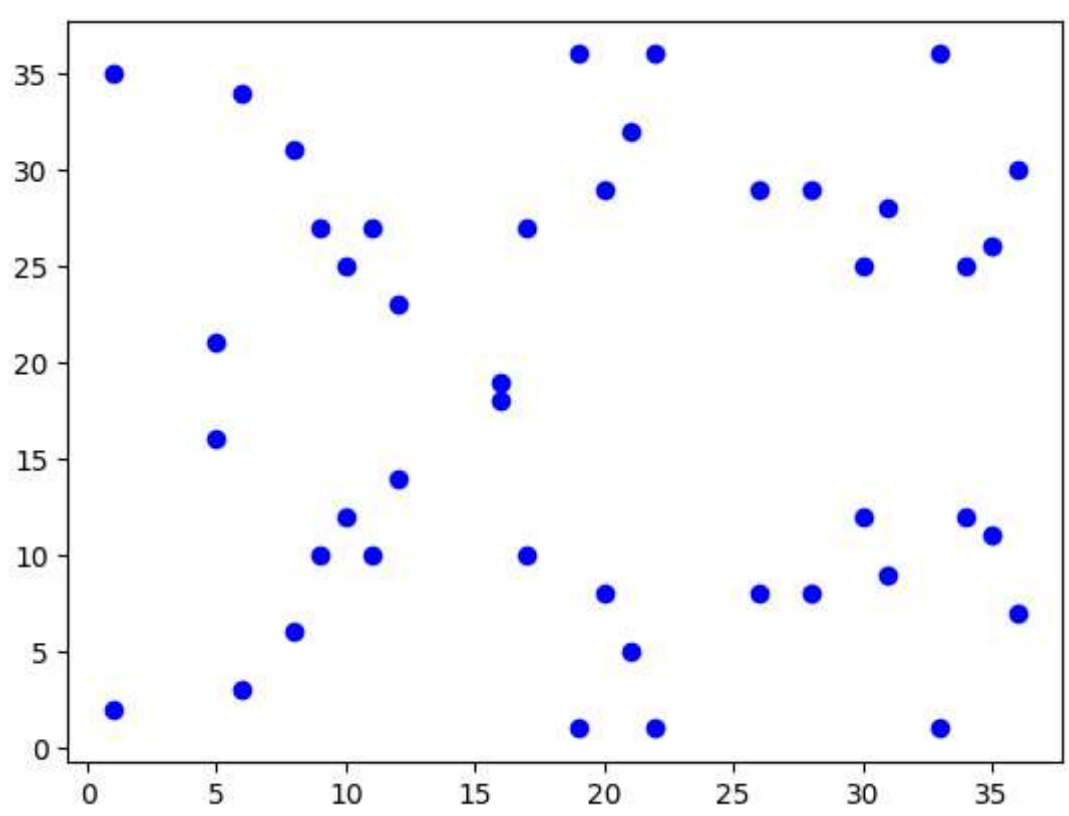


Открытые данные

In [11]:

```
G = (6, 3)
A, B = -5, 8
M = 37


x = np.array(range(0, M))
x2 = x**2 % M
y2 = ecc(x)
y = sqrt_f(x, y2)

fig = plt.figure(dpi=100)
for y_p in y:
    [plt.scatter(y_p[0], i, c='b') for i in y_p[1:]]

print(*['[x:{}, y:{}]\n'.format(y_p[0], y_p[1:]) for y_p in y])
```

```
[x:1, y:(2, 35)]
 [x:5, y:(16, 21)]
 [x:6, y:(3, 34)]
 [x:8, y:(6, 31)]
 [x:9, y:(10, 27)]
 [x:10, y:(12, 25)]
 [x:11, y:(10, 27)]
 [x:12, y:(14, 23)]
 [x:16, y:(18, 19)]
 [x:17, y:(10, 27)]
 [x:19, y:(1, 36)]
 [x:20, y:(8, 29)]
 [x:21, y:(5, 32)]
 [x:22, y:(1, 36)]
 [x:26, y:(8, 29)]
 [x:28, y:(8, 29)]
 [x:30, y:(12, 25)]
 [x:31, y:(9, 28)]
 [x:33, y:(1, 36)]
 [x:34, y:(12, 25)]
 [x:35, y:(11, 26)]
 [x:36, y:(7, 30)]
```



Алиса

In [12]:

```
n_a = 13
P_a = kP(n_a, G)
print(f'Alise\n{G=}, kP={P_a}')
```

```
Alise
G=(6, 3), kP=(35, 26)
```

Боб

In [13]:

```
n_b = 19
P_b = kP(n_b, G)
print(f'Bob\n{G=}, kP={P_b}')
```

```
Bob
G=(6, 3), kP=(8, 6)
```

In [14]:

```
k1 = kP(n_a, P_b)
k2 = kP(n_b, P_a)
print(f'{k1} == {k2}')
```

```
(20, 8) == (20, 8)
```

ECDSA

In [42]:

```
def SHA1(m):
    sha1 = hashlib.sha256(m)
    digest = sha1.hexdigest()
    hm = int(digest,16)
    return hm
```

Алиса

In [43]:

```
m=b"my secret message"
k = 13
n_a = 7
P_a = kP(k, G)
kG = kP(n_a, G)
x1,y1=P_a
r = x1 % M
hm = SHA1(m)
s = (hm + n_a*r)*eea(k, M) % M
print(f'Alise\n{G=} \n{kG=} \n{P_a=} \n{k=} \n{hm=} \n{r=} \n{s=}')
```

```
Alise
G=(6, 3)
kG=(20, 8)
P_a=(35, 26)
k=13
hm=70002835156374348357262373952667989648497450572443541419551376701264978910331
r=35
s=5
```

Боб

In [44]:

```
hm2 = SHA1(m)

w=eea(s,M)
u1, u2 = hm2*w % M, r*w % M

(x2, y2) = mull_PQ(kP(u1,G), kP(u2,P_a))
r2 = x2%M

print(f'{w=} \n{(u1,u2)=} \n{(x2,y2)=} \n{hm2=} \n{r2=}')
```

```
w=15
(u1,u2)=(1, 7)
(x2,y2)=(35, 11)
hm2=70002835156374348357262373952667989648497450572443541419551376701264978910331
r2=35
```

In [ ]:

```
```