南昌大学实验报告

学生姓名：丁俊　　学　　号：8003119100　专业班级：信息安全 193 班
实验类型：□ 验证 □ 综合 □ 设计 □ 创新　实验日期：2021.11.5　实验成绩：

一、实验项目名称

椭圆加密算法 ECC 的实现

二、实验目的

1、掌握密码学中常用的公钥密码算法 ECC 的算法原理；

2、掌握 ECC 的算法流程和实现方法。

三、实验基本原理

椭圆加密算法（ECC）是一种公钥加密体制，最初由 Koblitz 和 Miller 两人于 1985 年提出，其数学基础是利用椭圆曲线上的有理点构成 Abel 加法群上椭圆离散对数的计算困难性。

ECC 的主要优势是在某些情况下它比其他的方法使用更小的密钥，比如 RSA 加密算法，提供相当的或更高等级的安全。ECC 的另一个优势是可以定义群之间的双线性映射，基于 Weil 对或是 Tate 对；双线性映射已经在密码学中发现了大量的应用，例如基于身份的加密。不过一个缺点是加密和解密操作的实现比其他机制花费的时间长。

四、主要仪器设备及耗材

Window10、pycharm

五、实验步骤

```
"""
  考虑 K=kG ，其中 K、G 为椭圆曲线 Ep(a,b)上的点，n 为 G 的阶（nG=O∞ ），k 为小于 n 的整数。
    则给定 k 和 G，根据加法法则，计算 K 很容易但反过来，给定 K 和 G，求 k 就非常困难。
    因为实际使用中的 ECC 原则上把 p 取得相当大，n 也相当大，要把 n 个解点逐一算出来列成上表是不可能的。
    这就是椭圆曲线加密算法的数学依据

    点 G 称为基点（base point）

    k（k<n）为私有密钥（privte key）

    K 为公开密钥（public key)
"""


def get_gcd(a, b):
    if b:
        return get_gcd(b, a % b)
    else:
        return a


def get_inverse(a, p):
    """
    获取 a 的逆元
    :param a: 求 a 的逆元
    :param p: 素数域的范围(0, p-1)
    :return:
    """
    for i in range(1 ,p):
        if (i * a) % p == 1:
            return i
    return -1

```

```python
def get_np(x1, y1, x2, y2, a, p):
    flag = 1
    # 求 n*p
    # 如果 P = Q
    if x1 == x2 and y1 == y2:
        fenzi = 3 * (x1 ** 2) + a
        fenmu = 2 * y1
    else:
        fenzi = y2 - y1
        fenmu = x2 - x1
        if fenmu * fenzi < 0:
            flag = 0 # 标记负数
            fenzi = abs(fenzi)
            fenmu = abs(fenmu)
    gcd_value = get_gcd(fenzi, fenmu)
    zi = fenzi // gcd_value
    mu = fenmu // gcd_value
    # 求分母的逆元  逆元： ∀a ∈G ， ∃ b∈G 使得 ab = ba = e(1)
    # P(x,y)的负元是 (x,-y mod p)= (x,p-y) ，有 P+(-P)= O∞
    inverse_value = get_inverse(mu,p)   # 求分母的逆元
    k = (zi * inverse_value)

    if flag == 0:
        k = -k
    k = k % p

    x3 = (k ** 2 - x1 -x2) % p
    y3 = (k * (x1 - x3) -y1) % p
    return x3, y3  # 记得求得是-R,关于 x 轴对称

# 获取椭圆曲线的阶
def get_rank(x, y, a, b, p):
    '''
    椭圆曲线的阶是点的个数
    '''
    # -p 的坐标
    x1 = x
    y1 = (-1 * y) % p
    tmpx = x
    tmpy = y
    n = 1
    while True:
        n += 1
        # 求 n*p，直到求出阶
```

```
        p_x, p_y = get_np(tmpx, tmpy, x, y, a, p)
        if p_x == x1 and p_y == y1:
            return n+1
        tmpx = p_x
        tmpy = p_y

def get_param(x, a, b, p):
    '''
    计算(x,y)与(x,-y mod p),x是一个点的横坐标
    '''
    y = -1
    for i in range(p):
        if i ** 2 % p == (x ** 3 + a*x + b) % p:
            y = i
            break
    if y == -1:
        return False
    x1 = x
    y1 = (-1*y) % p
    return x, y, x1, y1

def get_graph(a,b,p):
    for i in range(p):
        value = get_param(i,a,b,p)
        if value:
            x, y, x1, y1 = value
            print('(' + str(x) +',' +str(y) + ')',end=' ')
            print('(' + str(x1)+ ','+ str(y1) + ')',end=' ')

def get_nG(Gx, Gy, key, a, p):
    # 获取公钥K,key表示计算次数
    tx = Gx
    ty = Gy
    while key != 1:
        tx, ty = get_np(tx, ty, Gx, Gy, a, p)
        key -= 1
    return tx, ty

def ecc_main():
    while True:
        a = int(input("请输入椭圆曲线a(a>0)的值"))
        b = int(input("请输入椭圆曲线b(b>0)的值"))
        p = int(input("请输入椭圆曲线p的值(p为素数)的值"))

```

```
122.             if (4 * (a**3) + 27*(b**2)) % p ==0:
123.                 print("你输入的参数有误，请重新输入")
124.             else:
125.                 break
126.         get_graph(a,b,p)
127.
128.         # 选点作为 G 点
129.         print("user1:在如上坐标系中选一个值为 G 的坐标")
130.         G_x = int(input("user1:请输入选取的 x 坐标值："))
131.         G_y = int(input("user1:请输入选取的 y 坐标值："))
132.
133.         # 获取椭圆曲线的阶
134.         n = get_rank(G_x, G_y, a, b, p)
135.         # 生成私钥 k
136.         key = int(input("user1:请输入私钥小 key(<{})".format(n)))
137.         Kx,Ky = get_nG(G_x,G_y,key,a,p)
138.
139.         # user2 阶段，拿到 user1 的公钥 KEY，Ep(a,b)的阶 n,求 rK,rG
140.         r = int(input("user2:请输入一个整数 k(<{})用于求 rK 和
    rG".format(n)))
141.         rGx, rGy = get_nG(G_x,G_y,r,a,p)
142.         rKx, rKy = get_nG(Kx,Ky,r,a,p)
143.         # 加密
144.         plain_text = input("user2:请输入要加密的字符串:")
145.         plain_text = plain_text.strip()
146.         c = []
147.         print("密文为:",end=" ")
148.         for ch in plain_text:
149.             intchar = ord(ch)
150.             cipher_text = intchar * rKx
151.             c.append([rGx,rGy,cipher_text])
152.             print("({},{}),{}".format(rGx, rGy, cipher_text),end="-")
153.         print(rGx,rGy)
154.         print("\nuser1 解密得到明文：", end="")
155.         for charArr in c:
156.             decrypto_text_x, decrypto_text_y = get_nG(charArr[0], char
    Arr[1], key, a, p)
157.             print(chr(charArr[2] // decrypto_text_x), end="")
158.
159.
160.     if __name__ == "__main__":
161.         print("*************ECC 椭圆曲线加密*************")
162.         ecc_main()
```

各函数功能：
1、get_gcd(a,b)：求 a、b 的最大公约数
2、get_inverse(a,p)：求 amodp 的逆元(在 1~p-1 范围内)
3、get_np(x1,y1,x2,y2,a,p)：求 P(x1,y1)、Q(x2,y2)两点 P+Q 运算，即 PQ 两点连线与椭圆曲线交点 R(xr,yr)的逆元-R(xr,-yr mod p)。
4、get_rank(x,y,a,b,p)：获取椭圆曲线的阶数，即曲线上的点的个数。
5、get_param(x,a,b,p)：求椭圆曲线上的点的坐标，已知 x 横坐标求出 y，以及(x,y)的逆元。
6、get_graph(a,b,p)：输入椭圆曲线上点的坐标。
7、get_nG(Gx,Gy,a,b)：由于 K=kG,求 K 公钥的坐标值。

六、实验数据及处理结果

```
D:\Python\python.exe F:/PythonFile/Algo/ECC.py
*************ECC椭圆曲线加密*************
请输入椭圆曲线a(a>0)的值7
请输入椭圆曲线b(b>0)的值11
请输入椭圆曲线p的值(p为素数)的值29
(2,2) (2,27) (3,1) (3,28) (4,4) (4,25) (8,12) (8,17) (9,7) (9,22) (12,5) (12,24) (18,13) (18,16) (21,9) (21,20) (22,5) (22,24) (24,5) (24,24) (25,8) (25,21) user1:在
user1:请输入选取的x坐标值：3
user1:请输入选取的y坐标值：1
user1:请输入私钥小key(<23)5
user2:请输入一个整数k(<23)用于求rK和rG6
user2:请输入要加密的字符串:BOY I Love you
密文为：(2,2),1650-(2,2),1975-(2,2),2225-(2,2),800-(2,2),1825-(2,2),800-(2,2),1900-(2,2),2775-(2,2),2950-(2,2),2525-(2,2),800-(2,2),3025-(2,2),2775-(2,2),2925-2 2

user1解密得到明文：BOY I Love you
Process finished with exit code 0
```

七、思考讨论题或体会或对改进实验的建议

八、参考资料

现代密码学第 4 版