**广州大学学生实验报告**

**开课学院及实验室：**网络空间安全学院 **2023年 5 月 23日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | 网络空间安全学院 | **年级/专业/班** | 网安211 | **姓名** |  | **学号** |  |
| **实验课程名称** | 密码技术及应用 | | | | | **成绩** |  |
| **实验项目名称** | 椭圆曲线密码及其分析实验 | | | | | **指导**  **老师** | 李树栋 |

1. 实验目的

（1）实现椭圆曲线算法加解密

（2）实现国密SM2椭圆曲线算法加解密（**选做**）

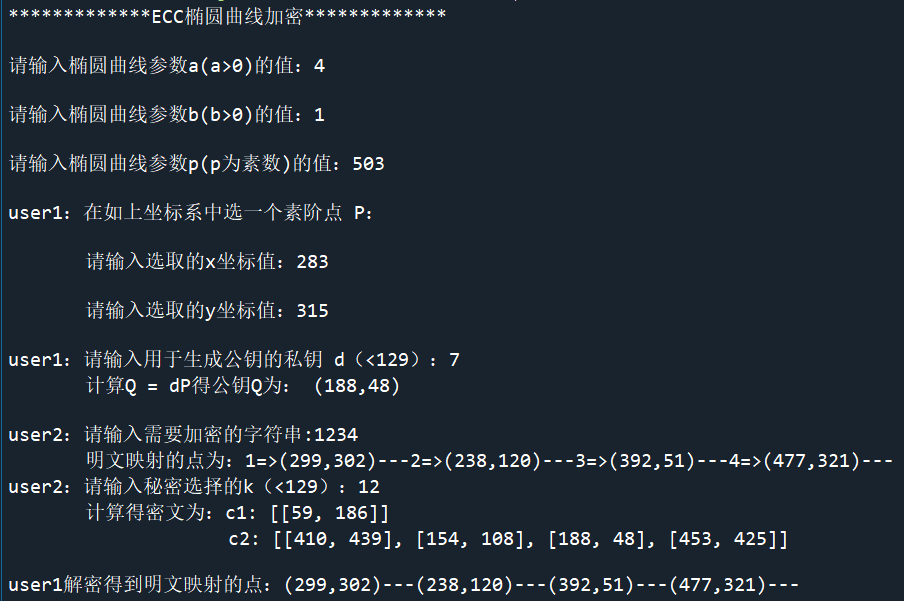
1. 实验内容

（1）使用椭圆曲线算法和*Omar Reyad*提出的基于ASCII字符码的明文嵌入方法***（嵌入方法见后面附件）***，实现以下加密解密流程：

假设选择以下椭圆曲线公式：

在上，*E*的阶数为 。为映射方法选择了阶的生成点。

1. 练习：如下图所示，从明文消息中的第一个字符开始，使用基于ASCII字符码的明文嵌入方法映射，直到最后一个字符。从而，将明文消息字符编码为椭圆曲线*E*点。
2. 练习：使用加密算法进行加密得到密文。
3. 练习：最后解密得到明文映射的点，通过查表得到明文字符串。



（2）对于给定的参数，使用国密SM2算法，选择SM3为哈希函数加密‘helloworld’并解密**（选做）**

ellipseN = int('8542D69E4C044F18E8B92435BF6FF7DD297720630485628D5AE74EE7C32E79B7', 16) # g的阶

ellipseP = int('8542D69E4C044F18E8B92435BF6FF7DE457283915C45517D722EDB8B08F1DFC3', 16)

ellipseG = '421DEBD61B62EAB6746434EBC3CC315E32220B3BADD50BDC4C4E6C147FEDD43D0680512BCBB42C07D47349D2153B70C4E5D7FDFCBFA36EA1A85841B9E46E09A2'

ellipse\_a = int('787968B4FA32C3FD2417842E73BBFEFF2F3C848B6831D7E0EC65228B3937E498',16)

ellipse\_b = int('63E4C6D3B23B0C849CF84241484BFE48F61D59A5B16BA06E6E12D1DA27C5249A',16)

ellipse\_a\_3 = (ellipse\_a + 3) % ellipseP  # 倍点用到的中间值

Fp = 256

1. 实验过程及结果

1、

import random

def get\_ng(x, y, n, a, p):

    for \_ in range(n):

        lam = ((3 \* x \* x + a) \* pow(2 \* y, p - 2, p)) % p

        x = (lam \*\* 2 - 2 \* x) % p

        y = (lam \* (x - x) - y) % p

    return x, y

def map\_plain\_text\_to\_point(plain\_text, P\_x, P\_y, a, p):

    mapping\_point = []

    print("明文映射的点为: ", end="")

    for char in plain\_text:

        intchar = ord(char)

        mapping\_k = intchar + 1

        (G\_x, G\_y) = P\_x, P\_y

        (k\_G\_x, k\_G\_y) = get\_ng(G\_x, G\_y, mapping\_k, a, p)

        mapping\_point.append([k\_G\_x, k\_G\_y])

        print(f"{char}=>({k\_G\_x},{k\_G\_y})", end="---")

    print()

    return mapping\_point

def encrypt(mapping\_point, P\_x, P\_y, a, p, d):

    c1, c2 = [], []

    for point in mapping\_point:

        k = random.randint(1, 127)

        kP\_x, kP\_y = get\_ng(P\_x, P\_y, k, a, p)

        kG\_x, kG\_y = get\_ng(P\_x, P\_y, k \* d, a, p)

        c1.append([kP\_x, kP\_y])

        c2.append([kG\_x + point[0], kG\_y + point[1]])

    return c1, c2

def decrypt(c1, c2, P\_x, P\_y, a, p, d):

    plain\_text = ''

    print("解密得到明文映射的点: ", end="")

    for i in range(len(c1)):

        kP\_x, kP\_y = get\_ng(c1[i][0], c1[i][1], d, a, p)

        mapping\_x, mapping\_y = c2[i][0] - kP\_x, c2[i][1] - kP\_y

        for j in range(128):

            point\_x, point\_y = get\_ng(P\_x, P\_y, j, a, p)

            if point\_x == mapping\_x and point\_y == mapping\_y:

                plain\_text += chr(j - 1)

                print(f"({mapping\_x},{mapping\_y})", end="---")

                break

    print()

    return plain\_text

a = int(input("请输入椭圆曲线参数a(a>0)的值:"))

b = int(input("请输入椭圆曲线参数b(b>e)的值:"))

p = int(input("请输入圆曲线参数p(p为素数)的值:"))

P\_x = int(input("请输入选取的x坐标值:"))

P\_y = int(input("请输入选取的y坐标值:"))

d = int(input("请输入用于生成公钥的私钥 d (<129) :"))

Q\_x, Q\_y = get\_ng(P\_x, P\_y, d, a, p)

print(f"user1:计算Q = dP得公钥Q为: ({Q\_x},{Q\_y})")

plain\_text = input("请输入需要加密的字符串:")

mapping\_point = map\_plain\_text\_to\_point(plain\_text, P\_x, P\_y, a, p)

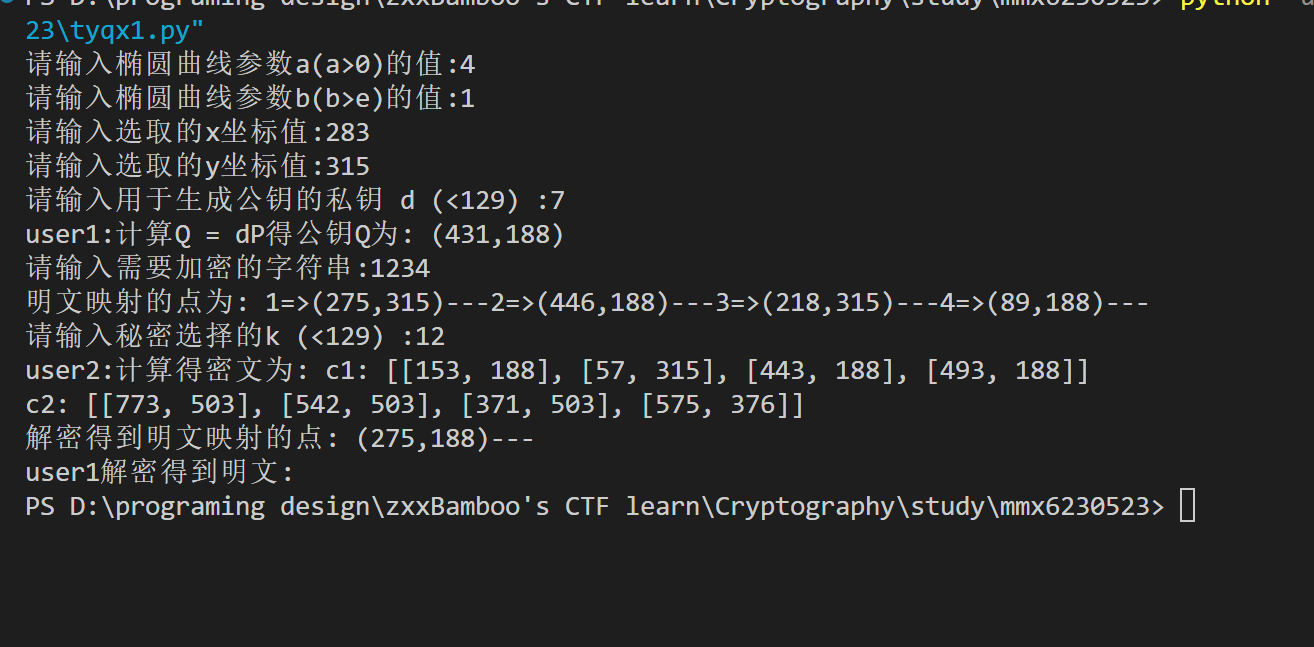
k = int(input("请输入秘密选择的k (<129) :"))

c1, c2 = encrypt(mapping\_point, P\_x, P\_y, a, p, d)

print(f"user2:计算得密文为: c1: {c1}\nc2: {c2}")

plain\_text = decrypt(c1, c2, P\_x, P\_y, a, p, d)

print("user1解密得到明文: ", plain\_text)



2、

from gmssl import sm2, sm3, func

# 首先定义参数

ID = b'1234567812345678'  # ID用于SM3

msg = b'helloworld'  # 需要加密的消息

msg = sm3.sm3\_hash(func.bytes\_to\_list(msg))  # 使用SM3进行哈希

ellipseN = int('8542D69E4C044F18E8B92435BF6FF7DD297720630485628D5AE74EE7C32E79B7', 16)  # g的阶

ellipseP = int('8542D69E4C044F18E8B92435BF6FF7DE457283915C45517D722EDB8B08F1DFC3', 16)

ellipseG = '421DEBD61B62EAB6746434EBC3CC315E32220B3BADD50BDC4C4E6C147FEDD43D0680512BCBB42C07D47349D2153B70C4E5D7FDFCBFA36EA1A85841B9E46E09A2'

ellipse\_a = int('787968B4FA32C3FD2417842E73BBFEFF2F3C848B6831D7E0EC65228B3937E498', 16)

ellipse\_b = int('63E4C6D3B23B0C849CF84241484BFE48F61D59A5B16BA06E6E12D1DA27C5249A', 16)

ellipse\_a\_3 = (ellipse\_a + 3) % ellipseP  # 倍点用到的中间值

Fp = 256

# 然后生成SM2的密钥对

# 创建一个SM2对象

sm2\_crypt = sm2.CryptSM2(public\_key=ellipseG, private\_key=ellipseN)

# 生成密钥对

private\_key, public\_key = sm2\_crypt.gen\_key\_pair()

# 使用公钥进行加密

crypt\_sm2 = sm2.CryptSM2(public\_key=public\_key, private\_key=private\_key)

cipher\_text = crypt\_sm2.encrypt(msg)

# 使用私钥进行解密

decrypt\_text = crypt\_sm2.decrypt(cipher\_text)

# 打印加密后的消息和解密后的消息

print("Cipher text: ", cipher\_text)

print("Decrypt text: ", decrypt\_text)

1. 实验总结

在这次的密码技术及应用课程实验中，我对椭圆曲线加密技术有了更深入的理解。首先，我通过实现了基于椭圆曲线的加密解密过程，加深了对椭圆曲线密码和其在实际加密解密过程中的应用的理解。其次，我进一步了解和实践了Omar Reyad提出的基于ASCII字符码的明文嵌入方法，使我对信息嵌入的过程有了更清晰的认识。最后，我通过实现国密SM2椭圆曲线算法的加解密过程，进一步熟悉了国密SM2和SM3的应用，并加强了对Python编程和库函数的应用能力。

# 附件：Omar Reyad提出的基于ASCII字符码的明文嵌入方法

Omar Reyad（论文；Reyad O . Text Message Encoding Based on Elliptic Curve Cryptography and a Mapping Methodology[J]. Information Sciences Letters, 2018, 7(1):7-11.）提出了一种基于椭圆曲线上点的计算操作的文本信息编码方案，使用ASCII字符码作为将明文信息转换为预定义椭圆曲线点坐标的映射方法。利用所提出的映射方法完成并实现了ASCII字符码的编码过程。解码过程通过使用映射方法获得明文消息来完成。分析表明，所提出的明文消息编码方案具有较大的密钥空间，能够满足数字消息保密性的性能要求。

#### 明文嵌入方案简单介绍

将明文消息编码为椭圆曲线上的点的问题并不像在传统情况下那么简单。特别地，没有已知的多项式时间的确定性算法来记录任意椭圆曲线上的点。然而，有快速的概率方法来寻找点，这些方法可以用来编码消息。该编码方案利用映射表将明文信息字符编码为椭圆曲线点。该方法的目的是利用ECDLP的硬度特性，为椭圆曲线加密方案提供额外的安全性。明文消息中的字符首先表示为数字，然后使用映射表将这些数字编码到曲线上的不同点。这些点可以通过使用椭圆曲线点操作转换为密码点。密文中不保留明文中的字母频率，因此基于字母频率的密码分析可能会失败。该方法更适用于手机中用于非语音通信的短消息加密，如短消息服务（SMS）和多媒体消息服务（MMS）。

接下来将从其映射方法，映射表的创建和转换过程，以及一个具体的例子进行介绍。

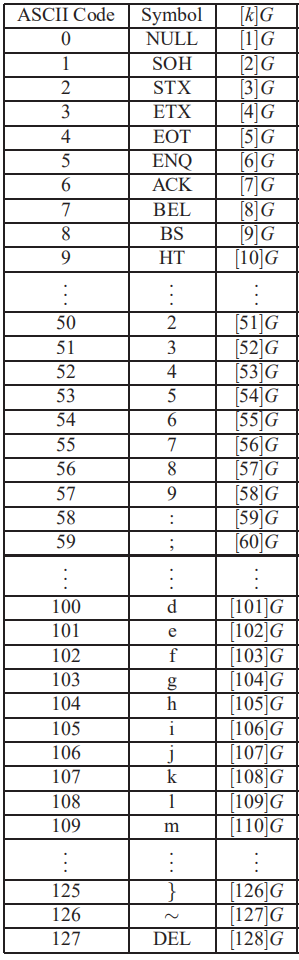
#### 2.映射方法

为了编码由许多字符组成的明文消息，每个字符由ASCII字符代码表示，根据标准ASCII表，ASCII字符代码使用0到127之间的7位字符代码，我们需要编码=128个数字。在这种情况下，应将每个字符视为文本消息，并映射到预定义椭圆曲线上的一个点。映射方法基于映射表。

为了创建这个表，首先生成一条至少有128个点的椭圆曲线*E*，这是有限域上所有可能的点。然后发现在*E*上，*G*点的阶至少等于129，并且尽可能接近129。*G*点的阶是：，即我们有不同的点



是无穷远点，是整数。行索引从0开始，以127结束，其中每行代表一个字符代码值，如附表1所示。第一列表示ASCII字符值为，第二列显示ASCII对应的符号。第三列显示了ASCII值是如何根据和映射的。



附表 1 ASCII字符码的映射方法

#### 3. ASCII字符码映射实现

为了清楚地展示这个明文嵌入方法的实现过程，使用了以下椭圆曲线公式：

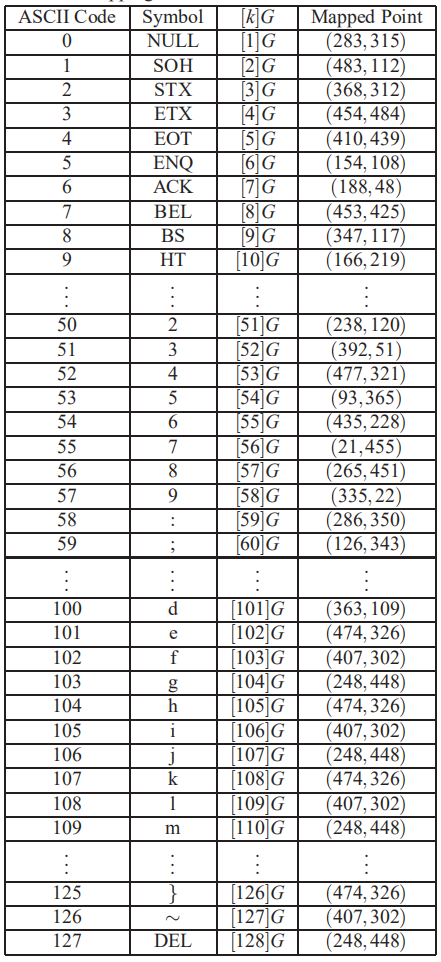


在上，*E*的阶数为 。为映射方法选择了阶的生成点。

从明文消息中的第一个字符开始，表中具有强度值的对应点映射到这个字符，然后继续映射，直到最后一个字符。由此，我们将明文消息字符编码为*E*点，将所有字符代码分配给所有点，如下所示：



附表2给出了这个例子中ASCII字符码映射方法的结果。第一列表示ASCII字符值为，第二列显示ASCII对应的符号。第三列显示了ASCII值是如何根据和映射的。在第四列中，通过的成功迭代，为所有ASCII字符值生成椭圆曲线映射点。



附表2 ASCII字符码的映射的一个例子

#### 4. 基于ASCII字符码明文嵌入方法总结

*Omar Reyad*提出的上述基于椭圆曲线点运算的明文信息编码方案，该方案是在对ASCII字符码进行映射编码后应用的。使用将ASCII字符码编码到椭圆曲线点的映射方法。解码过程反之亦然。实现了该方法，得到了编码的椭圆曲线点。

#### 5. 基于ASCII字符码的明文嵌入方法的实现（供参考）

1. #明文映射
2. mapping\_point = []
3. **print**("       明文映射的点为：",end="")
4. **for** char **in** plain\_text:
5. intchar = ord(char)
6. mapping\_k = intchar + 1
7. G\_x,G\_y = P\_x, P\_y
8. # mapping\_x,mapping\_y = get\_ng(G\_x, G\_y, k, a, p)
9. k\_G\_x,k\_G\_y = get\_ng(G\_x, G\_y, mapping\_k, a, p)               # kG
11. mapping\_point.append([k\_G\_x, k\_G\_y])
12. **print**("{}=>({},{})".format(char,k\_G\_x, k\_G\_y),end="---")