《密码学》课程设计实验报告

***本次实验源文件和实验报告均已上传至 RSA，ELGamal 以及 SM 文件夹下：**

sterzhang/Cryptology: Jianshu Zhang's cryptological experiments (github.com)

实验序号：05　　实验项目名称：公钥密码 RSA

学　　号	2021302181216	姓　　名	张鉴殊	专业、班	21 信安
实验地点	C202	指导教师	余荣威	时间	2023.12.22

一、　实验目的及要求

实验目的：

(1)　掌握公钥密码的概念和基本工作方式；

(2)　掌握 RSA 密码、ElGamal 密码和椭圆曲线密码的原理与算法；

(3)　了解 RSA 密码、ElGamal 密码和椭圆曲线密码的安全性；

(4)　了解 RSA 密码、ElGamal 密码和椭圆曲线密码的应用。

实验要求：

(1)　掌握 RSA 密码的实现方案；

(2)　掌握 ElGamal 密码的实现方案；

(3)　掌握椭圆曲线密码的实现方案；

(4)　了解公钥算法实现中的相关优化算法。

二、实验设备（环境）及要求

Windows 操作系统，高级语言开发环境

三、实验内容与步骤

1. RSA 密码

主函数代码如下：

```
if __name__ == '__main__':
    p = int(input('prime p: '))
    q = int(input('prime q: '))
    e = int(input('e (if do not know, enter 0): '))
    d = int(input('d (if do not know, enter 0): '))
    print("generate keys...")
    pub, pri = generate_keys(p, q, e, d)  # 随机选择两个质数 p 和 q 来生成公钥和私
```

```
钥
    M = int(input('message M: '))

    C = encrypt(M, pub)
    D = decrypt(M, pri)
    print("keys (pub,pri):", pub, pri)  # 打印公钥和私钥
    print("message M:", M)  # 打印原始消息
    print("encrypt message C:", C)  # 打印加密后的消息
    print("decrypt message D:", D)  # 打印解密后的消息
```

①随机地选择两个大素数 p 和 q，而且保密：

```
    p = int(input('prime p: '))
    q = int(input('prime q: '))
```

②计算 n=pq，将 n 公开：

```
# 定义生成公钥和私钥的函数
def generate_keys(p, q, e, d):
    numbers = range_prime(10, 100)  # 在给定范围内生成素数列表
    N = p * q  # 计算 N，N 是公钥和私钥的一部分,随意选择两个大的质数 p 和 q，p 不等于 q，计算 N=pq。
```

③计算φ(n)=(p-1)(q-1)，对φ(n)保密：

```
    phi = (p-1) * (q-1)  # 根据欧拉函数计算(p-1)(q-1)。根据欧拉函数，不大于 N 且与 N 互质的整数个数为(p-1)(q-1)
    print(f"phi:{phi}")
```

④随机地选取一个正整数 e，1<e<φ(n)且（e，φ(n)）=1，将 e 公开：

```
    if e==0:
        e = 0
        # 选择一个与 phi 互质的 e，且 e 小于 phi；选择一个整数 e 与(p-1)(q-1)互质，并且 e 小于(p-1)(q-1)
        for n in numbers:
            if n < phi and phi % n > 0:
                e = n
                break
        print(f"e:{e}")
        if e == 0:
            raise Exception("e not found") # 如果没有找到合适的 e，则抛出异常
```

⑤根据 ed＝1　mod　φ(n)，求出 d，并对 d 保密：

```
    if d==0:
        d = 0
        # 计算 d，d 是 e 的模逆，即(d * e) % phi == 1；用以下这个公式计算 d：d × e ≡ 1 (mod (p-1)(q-1))
```

```
        for n in range(2, phi):
            if (e * n) % phi == 1:
                d = n
                break
        print(f"d:{d}")
        if d == 0:
            raise Exception("d not found") # 如果没有找到合适的 d，则抛出异常
    return ((N, e), (N, d))  # 返回公钥(N,e)和私钥(N,d)
```

⑥加密运算：

$$C = M^e \bmod n \qquad (7\text{-}4)$$

```
# 定义加密函数
def encrypt(m, key):
    C, x = key
    return (m ** x) % C  # 使用公钥 key 来加密消息 m
```

⑦解密运算：

$$M = C^d \bmod n \qquad (7\text{-}5)$$

```
# 解密函数与加密函数相同，只是使用私钥进行解密
decrypt = encrypt
```

实验（1）令 p=3,q=11,d=7,m=5,手工或编程计算密文 C 。

```
(py38) root@6dceeba50b41:~/project/test/RSA# python RSA.py
prime p: 3
prime q: 11
e (if do not know, enter 0): 0
d (if do not know, enter 0): 7
generate keys...
phi:20
e:3
message M: 5
keys (pub,pri): (33, 3) (33, 7)
message M: 5
encrypt message C: 26
decrypt message D: 5
```

实验（2）设 RSA 密码的 e=3,n=33,C=9, 手工或编程计算明文 M 。

```
(py38) root@6dceeba50b41:~/project/test/RSA# python RSA.py
prime p: 3
prime q: 11
e (if do not know, enter 0): 3
d (if do not know, enter 0): 0
generate keys...
phi:20
d:7
message M: 9
keys (pub,pri): (33, 3) (33, 7)
message M: 9
encrypt message C: 3
decrypt message D: 15
```

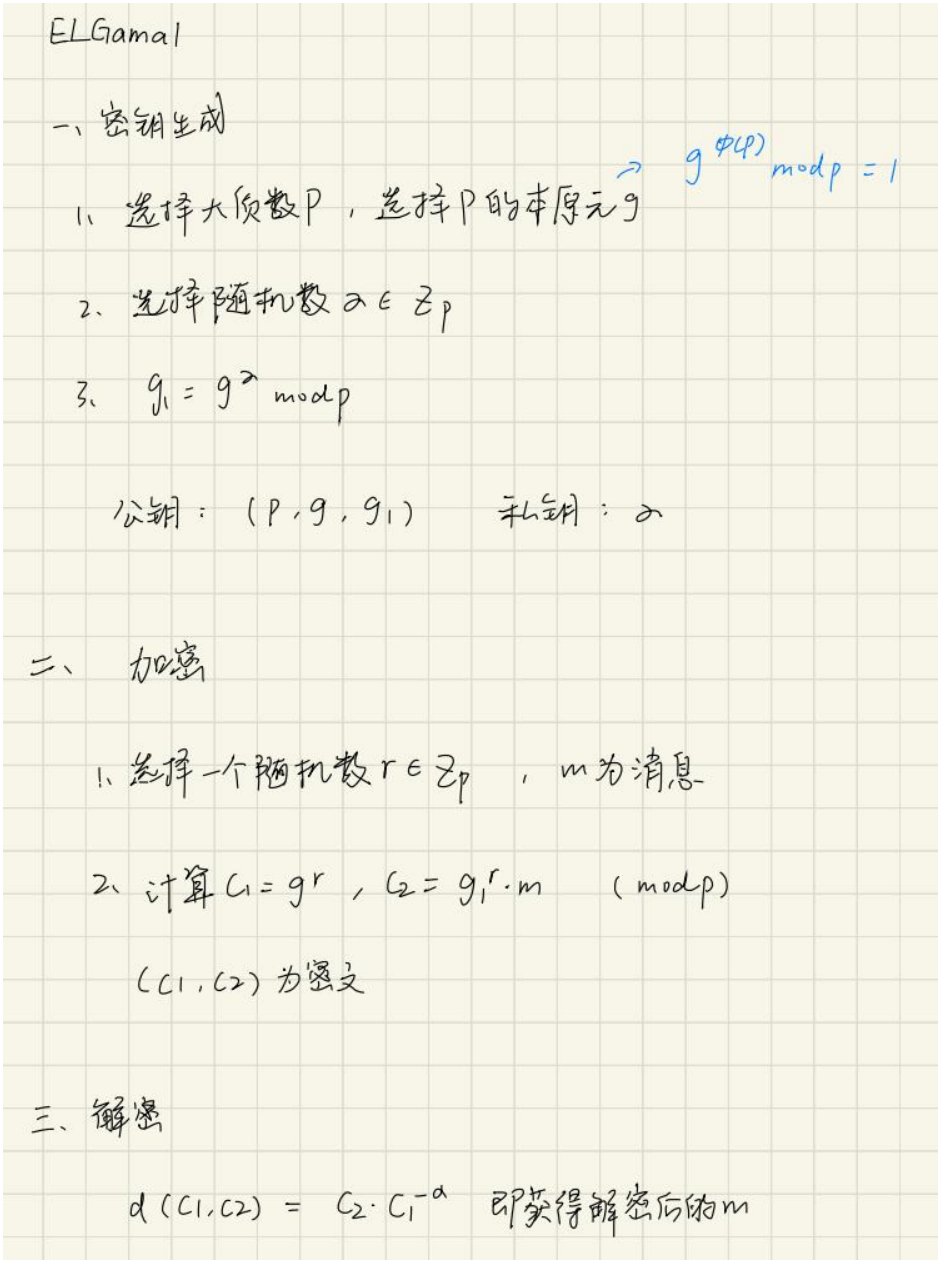
实验（3）令 p=17,q=11, e=7,试计算 RSA 密码其余参数 。

进一步对于 m=88, 计算密文 C 。

```
(py38) root@6dceeba50b41:~/project/test/RSA# python RSA.py
prime p: 17
prime q: 11
e (if do not know, enter 0): 7
d (if do not know, enter 0): 0
generate keys...
phi:160
d:23
message M: 88
keys (pub,pri): (187, 7) (187, 23)
message M: 88
encrypt message C: 11
decrypt message D: 88
```

2. ELGamal 密码（参见教材 p219）

整体过程如下：



ElGamal 算法及相应代码实现如下：

第一步是 ElGamal 密钥生成

密钥生成的总体代码如下：

```
# the owner side 生成公私钥
prime_number = int(input('Enter any prime number（eg.1237): '))
print('Generating keys...')
keys, generator = generate_keys(prime_number)
```

```
public_key, private_key = keys
print(f'public key: {public_key}, private_key: {private_key}')
print('sending public_key, prime_number, generator publicly.....\n')
```

关键是 generate_keys 函数的实现。

（1）随机选择一个大素数 p，且要求 p-1 有大素数因子。再选择一个模 p 的本原元 g。将 p 和 g 公开：

```
def generate_keys(prime):
    '''generates public_key, private_key pair'''

    # 1.选择大质数 p，并选择 p 的本原元 g
    p = prime
    g = find_primitive_root(p)
```

（2）随机选择一个整数 x 作为密钥，2≤x≤p-2 ：

```
    # 2.选择随机数 x，注意这里的 x 不能超过 p
    x = random.randint(1, (p - 1) // 2)
```

（3）计算 g1=g ^ x mod p，取 g1 为公钥：

```
# 3.得到 g1
    g1 = pow(g, x, p)
    private_key = x
    public_key = g1
    # 将（g1，x）作为密钥，g1 是公钥，这里随机选取的 x 是私钥，g 是本原元
    return (public_key, private_key), g
```

2、ElGamal 加密

这是调用 encrypt 的地方

```
# C1 为短钥 ephemeral_key，C2 为 cipher
cipher, ephemeral_key = encrypt(public_key, prime_number, generator)
# what we recieve is a cipher and temporary session key called "ephermeral_key"
print('Encrypting...')
print('ciphertext:', cipher, 'ephemeral_key:', ephemeral_key)
```

重要的是 encrypt 函数的实现。

（1）对于明文 M 加密，随机地选取一个整数 r，2≤r≤p-2

```
def encrypt(public_key, prime, g):
    '''返回 encrypted_msg 和 ephemeral_key'''

    print("on other side...")
    secret_message = int(input('Enter any message to encrypt: '))
    # 1.随机选取随机数 r，原算法中取值应该是 2≤y≤p-2，这里为了简化
    r = random.randint(1, (prime - 1) // 2)
```

（2）计算计算与密文计算相关的 C1（ephemeral_key）和 C2（(secret_message * masking_key) % prime），C1＝α^k mod p；C2＝MY^k mod p，密文为（C1,C2）

```
    # 2.计算与密文计算相关的 C1（ephemeral_key）和 C2（(secret_message *
masking_key) % prime）
    ephemeral_key = pow(g, r, prime)
    masking_key = pow(public_key, r, prime)
    # 返回 C2，C1
    return (secret_message * masking_key) % prime, ephemeral_key
```

3、ElGamal 解密

由密文可得明文 M，M=C2/C1^d mod p，这里有个等价的细节 pow 用于计算 C1^-a，根据费马小定理，对任何整数 a 和素数 p，如果 a 不是 p 的倍数，那么 a^p-1 mod p = 1，所以这里 a^p-1-d 实际上是 a^d 的模逆：

```
# decrypt the encrypted message on our side
# computing the masking key from ephermeral_key
print('Decrypting...')
print('Computing Masking key (use ephemeral_key)...')
# pow 用于计算 C1^-a，根据费马小定理，对任何整数 a 和素数 p，如果 a 不是 p 的倍数，那么
a^p-1 mod p = 1，所以这里 a^p-1-d 实际上是 a^d 的模逆
masking_key = pow(ephemeral_key, prime_number - 1 - keys[1])

# decipher
decipher = (cipher * masking_key) % prime_number
print('decrypted:', decipher)
```

例：设 p=19，m=17,构造一个 ELGamal 密码，并用它对 m 加密。

```
(py38) root@6dceeba50b41:~/project/test/ELGamal# python ELGamal-main.py
Enter any prime number (eg.1237): 19
Generating keys...
public key: 18, private_key: 9
sending public_key, prime_number, generator publicly.....

on other side...
Enter any message to encrypt: 17
Encrypting...
ciphertext: 2 ephemeral_key: 14

Decrypting...
Computing Masking key (use ephemeral_key)...
decrypted: 17
```

实验（4）设 p=5，m=3,构造一个 ELGamal 密码，并用它对 m 加密。

```
(py38) root@6dceeba50b41:~/project/test/ELGamal# python ELGamal-main.py
Enter any prime number (eg.1237): 5
Generating keys...
public key: 2, private_key: 1
sending public_key, prime_number, generator publicly.....

on other side...
Enter any message to encrypt: 3
Encrypting...
ciphertext: 1 ephemeral_key: 2

Decrypting...
Computing Masking key (use ephemeral_key)...
decrypted: 3
```

3.椭圆曲线密码（选作）

（1）GF(p)上的椭圆曲线

实验（5）取 p=23,求出椭圆曲线 $y^2=x^3+x+1$ 的全部解点。（选作）

x	x^3+x+1	y	x	x^3+x+1	y
0	1	1,22	12	16	4,19
1	3	7,16	13	3	7,16
2	11	无	14	22	无
3	8	10,13	15	10	无
4	0	0	16	19	无
5	16	4,19	17	9	3,20
6	16	4,19	18	9	3,20
7	6	11,12	19	2	5,18
8	15	无	20	17	无
9	3	7,16	21	14	无
10	22	无	22	22	无
11	9	3,20			

因此全部节点为:

(0,1) (0,22) (1,7) (1,16) (3,10) (3,13) (4,0) (5,4) (5,19) (6,4) (6,19) (7,11) (7,12) (9,7) (9,16) (11,3) (11,20) (12,4) (12,19) (13,7) (13,16) (17,3) (17,20) (18,3) (18,20) (19,5) (19,18)

（2）椭圆曲线密码

理解并实现 SM2 算法加解密过程。（教材 p239）

```
# bit str M
def SM2_encrypt(M,n,Gx,Gy,a,b,p,Px,Py):
    #print('SM2 ENCRYPTION')
    klen = len(M)
```

```
    # A1. 用随机数发生器产生随机数 k ∈ [1, n-1]
    k = random.randint(1,n-1)

    # A2. 计算椭圆曲线点 C_1=[k]G=(x1, y1),按 GB/T 2918.1-2016 中 4.2.9 和 4.2.5
给出的方法，将 C_1 的数据类型转换为比特串;
    x1,y1 = multiplyk_point(Gx,Gy,k,a,p)
    C1 = point2bit(x1,y1,p)

    # A3. 计算椭圆曲线点 S=[h]P_B,若 S 是无穷远点,则报错并退出
    h = math.floor(((math.sqrt(p)+1)**2)/n)
    Sx,Sy = multiplyk_point(Px,Py,h,a,p)
    if(Sx=='O' or Sy=='0'):
        return False

    # A4. 计算椭圆曲线点[k]P_B=(x2, y2),按 GB/T 32918.1-2016 中 4.2.6 和 4.2.5 给
出的方法,将坐标 x2、y2 的数据类型转换为比特串
    x2,y2 = multiplyk_point(Px,Py,k,a,p)
    x2_bit = Fq2bit(x2,p)
    y2_bit = Fq2bit(y2,p)

    # A5. 计算 t=KDF(x_2 || y_2,klen),若 t 为全 0 比特串,则返回 A_1
    t = KDF(x2_bit+y2_bit,klen)
    if(int(t,base=2)==0):
        return False

    # A6. 计算 C_2
    C2 = Xor(M,t)

    # A7. 计算 C_3
    C3 = SM3.SM3_digest(x2_bit+M+y2_bit)

    # A8. 输出密文 C
    C = [C1,C2,C3]
    return C
```

```
def SM2_decrypt(C,n,Gx,Gy,a,b,p,d):
    #print('SM2 DECRYPTION')
    C1 = C[0]
    C2 = C[1]
    C3 = C[2]
    klen = len(C2)
    # B1
    PC = C1[:8] # PC=04
```

```
    bit_len = int((len(C1)-8)/2)
    x1 = bit2Fq(C1[8:8+bit_len])
    y1 = bit2Fq(C1[8+bit_len:])
    left = (y1**2)%p
    right = (x1**3+a*x1+b)%p
    if(left!=right):
        return False
    # B2
    h = math.floor(((math.sqrt(p)+1)**2)/n)
    Sx,Sy = multiplyk_point(Px,Py,h,a,p)
    if(Sx=='O' or Sy=='0'):
        return False
    # B3
    x2,y2 = multiplyk_point(x1,y1,d,a,p)
    x2_bit = Fq2bit(x2,p)
    y2_bit = Fq2bit(y2,p)
    # B4
    t = KDF(x2_bit+y2_bit,klen)
    if(int(t,base=2)==0):
        return False
    # B5
    MM = Xor(C2,t)
    # B6
    u = SM3.SM3_digest(x2_bit+MM+y2_bit)
    if(u!=C3):
        return False
    # B7
    return MM
```

四、分析与讨论

1. 在 RSA 加密中，如果两个不同的用户碰巧选择了相同的大素数，假设是 p，作为他们公钥的一部分，这会不会影响他们的加密系统的安全性？

在 RSA 中，公钥是由两个大素数（记为 p 和 q）的乘积构成的。私钥是根据 p 和

q 计算出来的，但是 p 和 q 本身是保密的。假设两个用户分别拥有公钥(n1,e)和(n2,e)，其中 n1=p*q1，n2=p*q2。这种情况下，攻击者应该是可以很容易通过计算 n1，n2 的最大公约数来找到那个共同的素数 p，这是十分危险的，因为一旦知道了 p，就能反推得到另外一个大素数，这样私钥就泄露了。所以原来这个大素数不撞车是 RSA 安全的前提。

2. 在 ElGamal 加密中，如果加密时使用的随机数 k 不是真正的随机，而是可预测的或重复使用了，这会如何影响 ElGamal 加密系统的安全性？

我们知道在 ElGamal 加密中，k 是用于生成每个消息的唯一密文的临时密钥。如果攻击者能够预测出用于加密的 k 值，他们可以直接计算出 ElGamal 中的一部分密文，从而减轻了破解整个密文所需的工作量。这种情况下，即使攻击者不知道私钥，他们也有可能通过已知 k 值来获取有关明文的信息。另外，如果在不同的消息加密中重复使用相同的 k 值，攻击者可以在分析多个由相同 k 生成的密文时利用这一点。比如，攻击者可以尝试比较不同的密文来找出共同的模式或者运用已知明文攻击，这可以帮助他们破解密文或推断出私钥。因此，确保 k 的高质量随机性和每次加密都使用新的 k 值是维护 ElGamal 加密安全性的关键。

六、教师评语 签名： 日期：	成绩