

密码学实验实验报告十一

18374480-黄翔

2021 年 7 月 1 日

1 实验目的

1. 了解并掌握常用的数字签名方法并编程实现
2. 感受不同数字签名方案的优缺点及适用场景

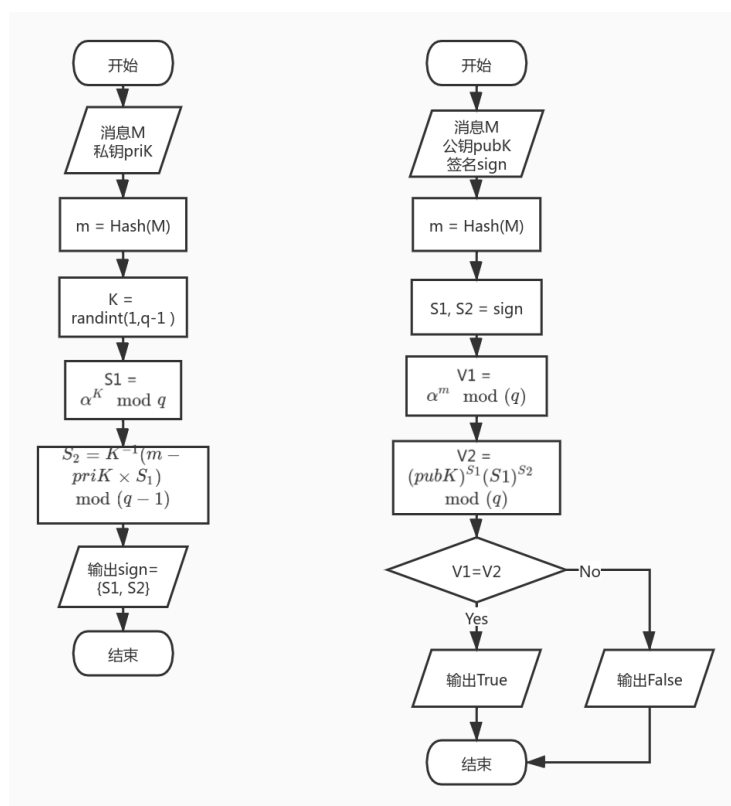
2 实验环境

1. python 3.9

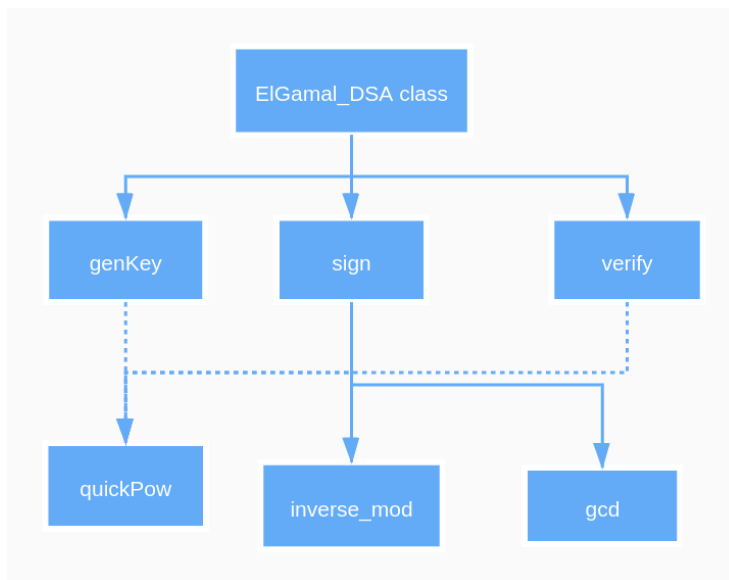
3 实验内容

3.1 Elgamal 数字签名

3.1.1 算法流程图



3.1.2 函数调用关系



3.1.3 算法伪代码

Algorithm 1 Elgamal 签名生成

Input: 消息 M , 私钥 $priK$

Output: 签名 $sign$

```

1:  $m \leftarrow Hash(M)$ 
2: while  $True$  do
3:    $K \leftarrow$  随机选择  $1 \leq K \leq q - 1$ 
4:   if  $gcd(K, q - 1)$  is 1 then
5:     break
6:   end if
7: end while
8:  $S_1 \leftarrow \alpha^K \bmod q$ 
9:  $S_2 \leftarrow K^{-1}(m - priK \times S_1) \bmod (q - 1)$ 
10:  $sign \leftarrow (S_1, S_2)$ 
11: return  $sign$ 

```

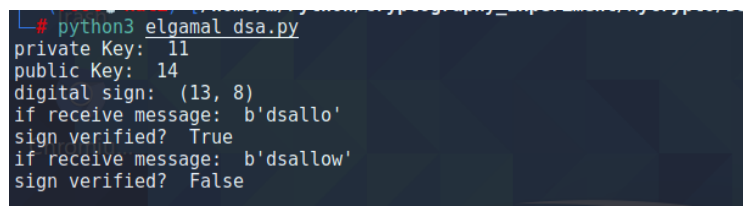
Algorithm 2 Elgamal 签名验证

Input: 消息 M , 公钥 $pubK$, 签名 $sign$

Output: 验证结果

```
1:  $m \leftarrow \text{Hash}(M)$ 
2:  $V_1 \leftarrow \alpha^m \bmod q$ 
3:  $V_2 \leftarrow (\text{pub}K)^{S_1} (S_1)^{S_2} \bmod q$ 
4: if  $V_1 == V_2$  then
5:   return True
6: end if
7: return False
```

3.1.4 测试样例及结果截图



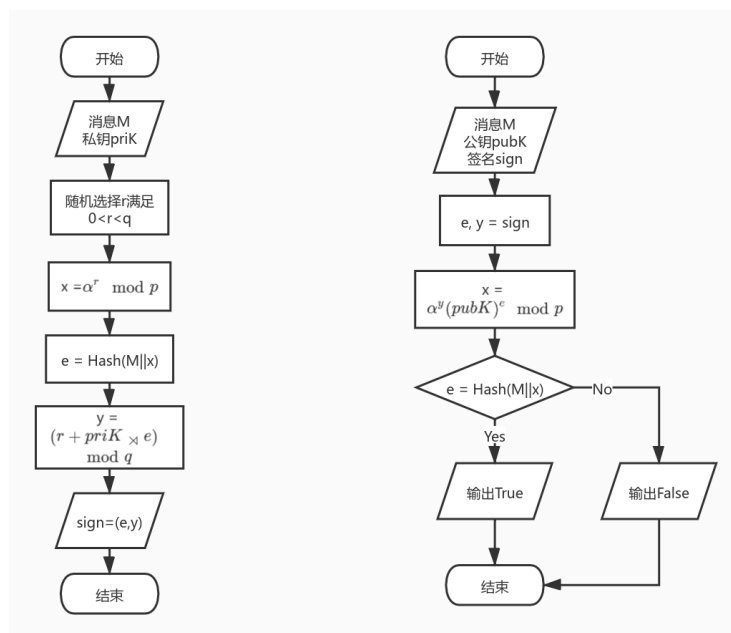
```
# python3 elgamal_dsa.py
private Key: 11
public Key: 14
digital sign: (13, 8)
if receive message: b'dsallo'
sign verified? True
if receive message: b'dsallow'
sign verified? False
```

3.1.5 总结

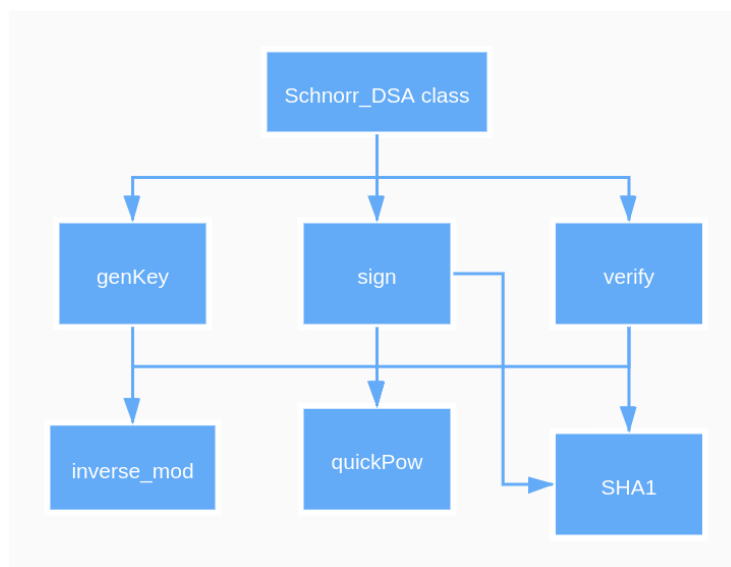
安全性 Elgamal 数字签名在实际使用中存在一些限制，如会话密钥 K 的值必须随机选取，且必须确保在签不同的信息时会话密钥没有重复使用过，必须避免选到弱随机数 2 或者 3

3.2 Schnorr 数字签名

3.2.1 算法流程图



3.2.2 函数调用关系



3.2.3 算法伪代码

Algorithm 3 Schnorr 签名生成

Input: 消息 M , 私钥 $priK$

Output: 数字签名 $sign$

```
1:  $r \leftarrow$  随机选择  $0 < r < q$ 
2:  $x \leftarrow \alpha^r \bmod p$ 
3:  $e \leftarrow Hash(M||x)$ 
4:  $y \leftarrow (r + priK \times e) \bmod q$ 
5:  $sign \leftarrow (e, y)$ 
6: return  $sign$ 
```

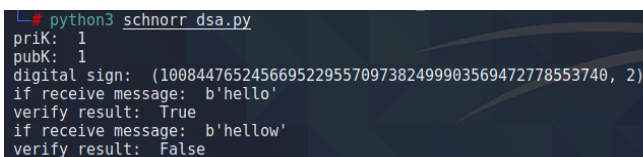
Algorithm 4 Schnorr 签名验证

Input: 消息 M , 公钥 $pubK$, 签名 $sign$

Output: 验证结果

```
1:  $e, y \leftarrow sign$ 
2:  $x \leftarrow \alpha^y (pubK)^e \bmod p$ 
3: if  $e == Hash(M||x)$  then
4:   return  $True$ 
5: end if
6: return  $False$ 
```

3.2.4 测试样例及结果截图



```
python3 schnorr_dsa.py
priK: 1
pubK: 1
digital sign: (1008447652456695229557097382499903569472778553740, 2)
if receive message: b'hello'
verify result: True
if receive message: b'hellow'
verify result: False
```

3.2.5 总结

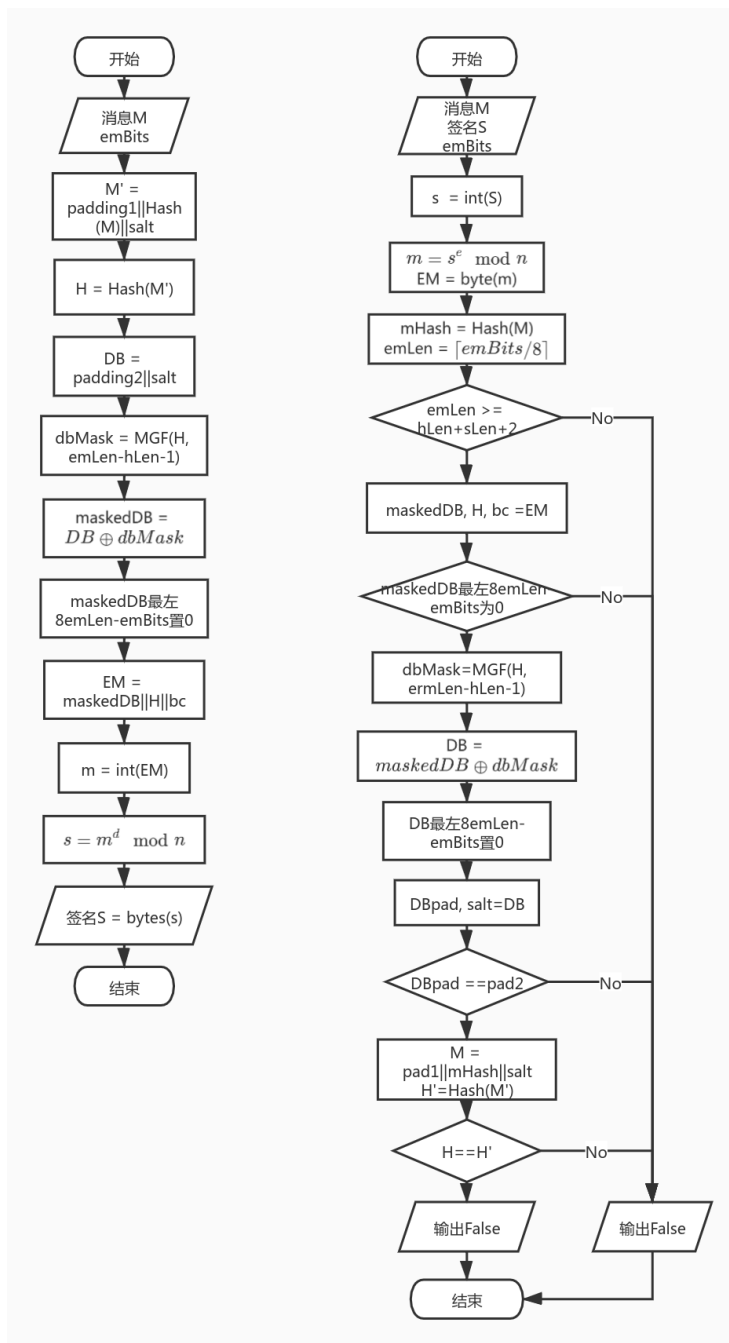
签名算法优点

- 1 Schnorr 方案将生成签名所需的消息计算量最小化。生成签名的主要工作不依赖于消息，可以在处理器空闲时执行
- 2 Schnorr 签名算法有可证明安全性

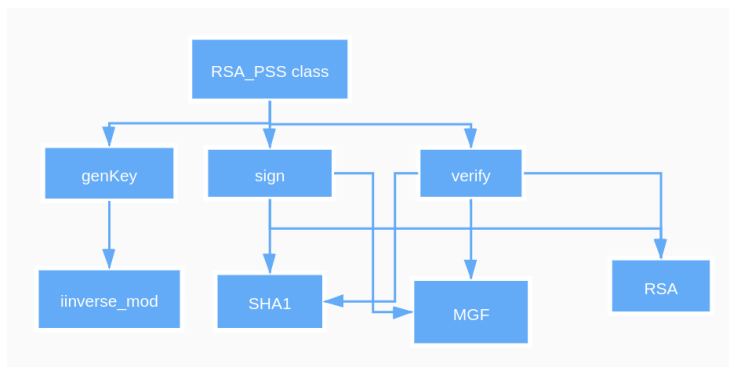
- 3 Schnorr 签名算法具有不可延展性，第三方无法在不知道私钥的情况下，针对某一公钥和消息的有效签名，改造成针对该公钥和信息的另一有效签名
- 4 Schnorr 签名算法是线性的，使得多个合作方能生成对他们的公钥之和也有效的签名

3.3 RSA-PSS 数字签名

3.3.1 算法流程图



3.3.2 函数调用关系



3.3.3 算法伪代码

Algorithm 5 RSA-PSS 签名生成

Input: 信息 M , $emBits$

Output: 签名 S

```

1:  $emLen \leftarrow \lceil \frac{emBits}{8} \rceil$ 
2:  $mHash \leftarrow Hash(M)$ 
3:  $salt \leftarrow$  随机生成  $hLen$  长度的字节串
4:  $M' \leftarrow padding_1 || mHash || salt$ 
5:  $H \leftarrow Hash(M')$ 
6:  $DB \leftarrow padding_2 || salt$ 
7:  $dbMask \leftarrow MGF(H, emLen - hLen - 1)$ 
8:  $maskedDB \leftarrow DB \oplus dbMask$ 
9:  $maskedDB \leftarrow$  将  $maskedDB$  最左字节的最左  $8emLen - emBits$  置 0
10:  $EM \leftarrow maskedDB || H || bc$ 
11:  $m = int.from\_bytes(EM, 'big')$ 
12:  $s \leftarrow m^d \bmod n$ 
13:  $k \leftarrow (n.bit\_length() + 7) >> 3$ 
14:  $S \leftarrow s.to\_bytes(k, 'big')$ 
15: return  $S$ 
  
```

Algorithm 6 RSA-PSS 签名验证

Input: 消息 M , $emBits$, 签名 S

Output: 验证结果

```

1:  $s \leftarrow \text{int.from\_bytes}(S, 'big')$ 
2:  $m \leftarrow s^e \bmod n$ 
3:  $emLen \leftarrow \lceil \frac{emBits}{8} \rceil$ 
4:  $EM \leftarrow m.to\_bytes(emLen, 'big')$ 
5:  $mHash \leftarrow \text{Hash}(M)$ 
6: if  $emLen < hLen + sLen + 2$  then
7:   return False
8: end if
9: if  $EM[-1]! = bc$  then
10:  return False
11: end if
12:  $maskedDB, H, bc \leftarrow EM$ 
13: if  $maskedDB$  最左  $8emLen - emBits$  位不全为 0 then
14:  return False
15: end if
16:  $dbMask \leftarrow MGF(H, emLen - hLen - 1)$ 
17:  $DB \leftarrow maskedDB \oplus dbMask$ 
18:  $DB \leftarrow$  设置  $DB$  最左  $8emLen - emBits$  位为 0
19: if  $DB[: emLen - hLen - sLen - 1] \neq padding_2$  then
20:  return False
21: end if
22:  $salt \leftarrow DB[-sLen :]$ 
23:  $M' \leftarrow padding_1 || mHash || salt$ 
24:  $H' \leftarrow \text{Hash}(M')$ 
25: if  $H' == H$  then
26:  return True
27: end if
28: return False

```

3.3.4 测试样例及结果截图

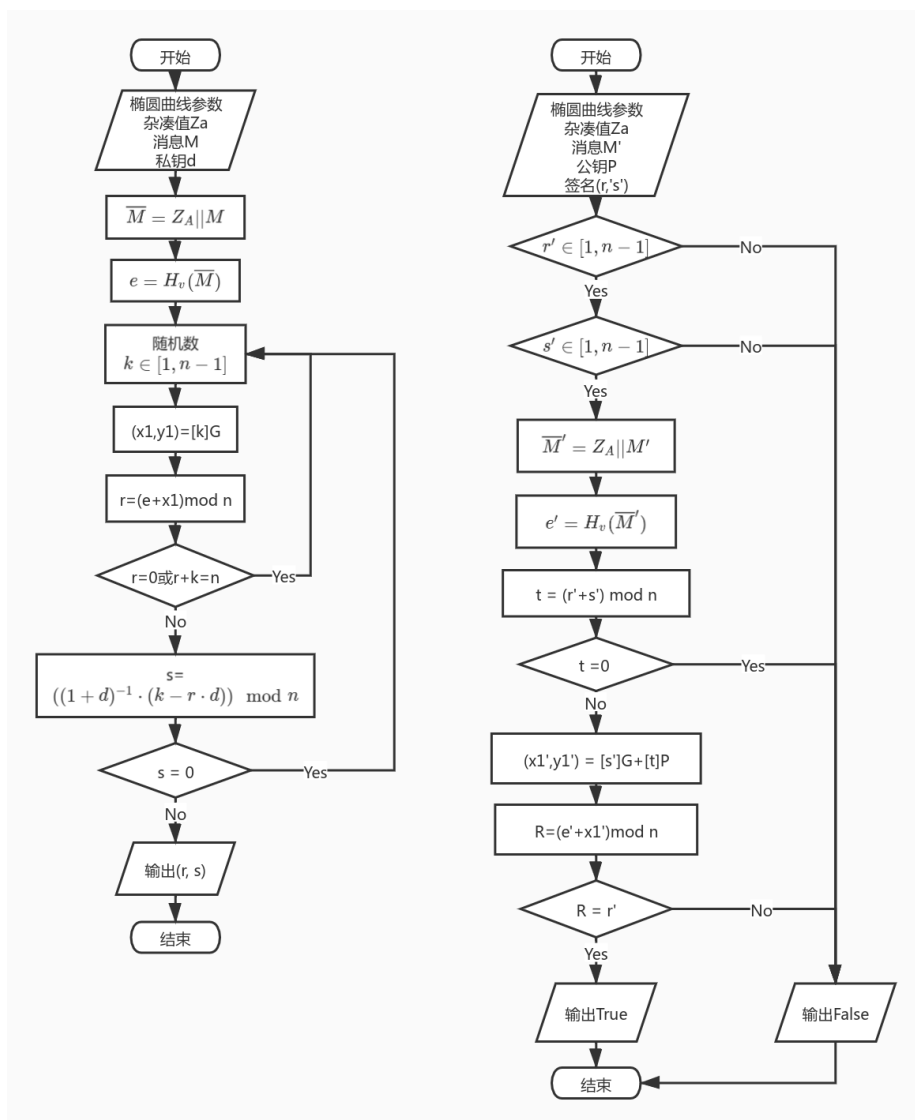


3.3.5 总结

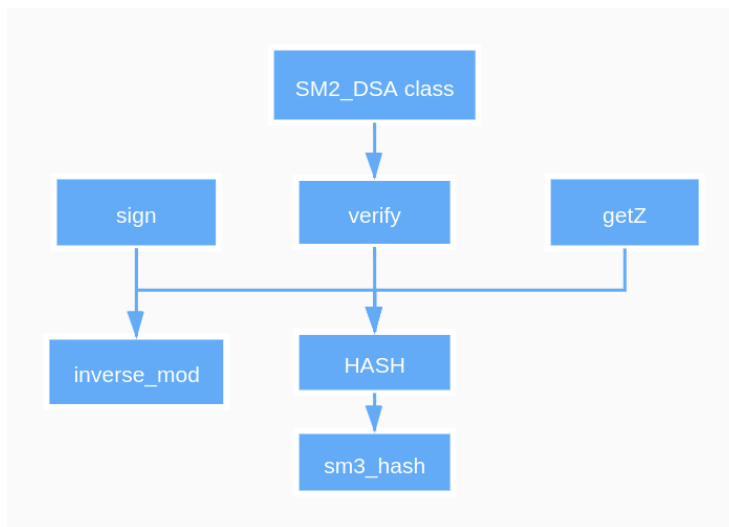
安全性 签名算法中，PSS 方案与其他基于 RSA 的方案，该方案使用了随机化的处理过程，能够证明其安全性与 RSA 算法的安全性相关。因为每次使用时盐的值都不同，所以使用相同的私钥对相同的消息进行两次签名，将得到两个不同的结果，这增加了签名方案的安全度。

3.4 sm2 数字签名

3.4.1 算法流程图



3.4.2 函数调用关系



3.4.3 算法伪代码

Algorithm 7 sm2 签名生成

Input: 信息 M , 私钥 d_A , 杂凑值 Z_A

Output: 签名 $sign$

```

1:  $e \leftarrow Hash_v(Z_A || M)$ 
2: while  $True$  do
3:    $k \leftarrow$  随机生成  $k \in [1, n - 1]$ 
4:    $(x_1, y_1) \leftarrow [k]G$ 
5:    $r \leftarrow (e + x_1) \bmod n$ 
6:   if  $r = 0$  or  $r + k = n$  then
7:     continue
8:   end if
9:    $s \leftarrow ((1 + d_A)^{-1} \cdot (k - r \cdot d_A)) \bmod n$ 
10:  if  $s \neq 0$  then
11:    break
12:  end if
13: end while
14: return  $sign \leftarrow (r, s)$ 

```

写在纸上的普通的物理签名，但是使用了公钥加密领域的技术来实现的，用于鉴别数字信息的方法。一套数字签名通常定义两种互补的运算，一个用于签名，另一个用于验证。数字签名是非对称密钥加密技术与数字摘要技术的应用。