

# 密码学引论实验三——RSA 加密

学号: 202200460104

姓名:密语

班级: 网安1班

### 密码学引论实验三



# 目录

1	任务一	2
	1.1 任务内容	2
	1.2 正确性分析	2
	1.3 代码验证	2
<b>2</b>	任务二	3
	2.1 任务内容	3
	2.2 任务分析	3
	2.3 代码实现	3
	2.4 解密时间对比	4
3	实验总结	5
4	附录	5



# 1 任务一

#### 1.1 任务内容

对 RSA 加密算法做如下修改:  $ed\equiv 1 \mod \lambda(n)$ ,其中  $\lambda(n)=\frac{(p-1)(q-1)}{\gcd(p-1,q-1)}$ ,证明修改后方案的正确性。

#### 1.2 正确性分析

根据 RSA 加解密的原理, 我们知道:

$$\begin{cases} c \equiv m^e \pmod{n}, \\ m \equiv c^d \pmod{n}. \end{cases}$$
 (1)

```
c^d \pmod n \to (m^e)^d \pmod n \to m^{ed} \pmod n
又因为 ed \equiv 1 \pmod {\lambda(n)},所以 ed = k\lambda(n) + 1
所以 m^{ed} = m^{k\lambda(n)+1} = m \cdot (m^{\lambda(n)})^k = m \cdot (m^{k'\varphi(n)})^k = m \cdot (m^{\varphi(n)})^{k \cdot k'} \equiv m \pmod n
所以 m \equiv c^d \pmod n,所以修改后的方案是正确的。
```

#### 1.3 代码验证

```
import math
1
2
   from Crypto.Util.number import *
3
4 \mid p = getPrime(1024)
   q = getPrime(1024)
6 e = 65537
  n = p * q
7
8 | phi = (p - 1) * (q - 1)
  lambda_n = (p - 1) * (q - 1)
10 | d = inverse(e, int(lambda_n))
  message = bytes_to_long(b'This is right')
11
12 \mid c = pow(message, e, n)
13 \mid m = pow(c, d, n)
   print("加密后的密文为: ", c)
14
   print("解密后的明文为: ",long_to_bytes(m))
15
```

运行结果:

```
D:\python\venv\Scripts\python.exe D:\python\RSA_test.py
加密后的密文为: 849202216794869156426626710968285894931546975573185242090526
解密后的明文为: b'This is right'
进程已结束,退出代码0
```

图 1: 输出界面



## 2 任务二

#### 2.1 任务内容

已知 n=3026533,利用中国剩余定理加速解密算法,设加密指数 e=3,解密指数 d=2015347,密文 c=152702,求明文 m。

#### 2.2 任务分析

根据中国剩余定理,我们知道:

$$\begin{cases}
 m_1 = c^d \pmod{p}, \\
 m_2 = c^d \pmod{q}.
\end{cases}$$
(2)

这样我们就把模数的位数减小了,从而加速了解密的过程,但是 d 的位数依然很大,我们可以通过欧拉定理来降低 d 的位数。

$$\begin{cases}
 m_1 = c^{d \pmod{p-1}} \pmod{p}, \\
 m_2 = c^{d \pmod{q-1}} \pmod{q}.
\end{cases}$$
(3)

上式的证明如下:

$$m_1 = c^{k \cdot (p-1) + dp} \pmod{p}$$

$$= (c^{p-1})^k \cdot c^{dp} \pmod{p}$$

$$= c^{dp} \pmod{p}$$

$$(4)$$

同理可得  $m_2 = c^{dq} \pmod{q}$ 

故同余方程可以转化为:

$$\begin{cases}
 m_1 = c^{dp} \pmod{p}, \\
 m_2 = c^{dq} \pmod{q}.
\end{cases}$$
(5)

利用中国剩余定理求解这个方程即可得到明文 m。

#### 2.3 代码实现

```
1 #sagemath
2 e = 3
3 n =3026533
4 p=1511
5 q=2003
6 d = 2015347
7 c=152702
8 dp = d % (p - 1)
9 dq = d % (q - 1)
10 mp = pow(c, dp, p)
11 mq = pow(c, dq, q)
12 m= crt([int(mp),int(mq)],[int(p),int(q)])#使用中国剩余定理解密
13 print("解密后的明文为: ",m )
```



上面的代码使用了 sagemath 的 crt 函数,这个函数可以直接求解中国剩余定理的解,从而得到明文 m。crt 函数的参数是两个列表,第一个列表是余数,第二个列表是模数,函数返回的是解。

#### 2.4 解密时间对比

测试解密时间的代码如下,我们使用 ns 为单位来计算时间。

```
#sagemath
1
2 | import time
3 | e = 3
4 | n = 3026533
5 p=1511
6 q=2003
7 d = 2015347
8
   c = 152702
9
10
  | start_time1 = time.perf_counter_ns()
11
   m1 = pow(c, d, n)
   end_time1 = time.perf_counter_ns()
   print("未使用CRT加速,解密后的明文为:",m1)
13
   print(f"未使用CRT加速,解密时间:{end_time1-start_time1} ns")
14
15
16
   start_time2 = time.perf_counter_ns()
17 | dp = d \% (p - 1)
  dq = d \% (q - 1)
18
   mp = pow(c, dp, p)
19
   mq = pow(c, dq, q)
20
   m2= crt([int(mp),int(mq)],[int(p),int(q)])
21
22
   end_time2 = time.perf_counter_ns()
23
  print("使用 CRT 加速, 解密后的明文为: ", m2 )
24
   print(f"使用CRT加速,解密时间:{end time2-start time2} ns")
25
   print("加速比: ",(end_time1-start_time1)/(end_time2-start_time2))
26
```

运行结果:

```
miyu@miyu-virtual-machine:~/Desktop/Study$ sage RSA_CRT.sage
未使用CRT加速,解密后的明文为: 1186745
未使用CRT加速,解密时间:1171557 ns
使用CRT加速,解密后的明文为: 1186745
使用CRT加速,解密时间:165867 ns
加速比: 7.063231384181302
```

图 2: 输出界面



解密方式	解密时间	解密结果
普通解密	1171557 ns	1186745
中国剩余定理解密	165867  ns	1186745

从上面的结果可以看出,使用中国剩余定理加速解密的时间比普通解密的时间快了很多,实现了 7 倍的加速。

# 3 实验总结

本次实验主要学习了 RSA 加密算法的原理,以及中国剩余定理的应用,通过实验验证了 RSA 加密算法的正确性,以及中国剩余定理加速解密的效果。实验中,我使用了 sagemath 库中的 crt 函数来求解中国剩余定理的解,从而加速解密的过程。

# 4 附录

修改后的 RSA 加解密代码:

```
#sagemath
1
2 | import math
3 | from Crypto.Util.number import *
4
5 p = getPrime(1024)
6 \mid q = getPrime(1024)
7 e = 65537
8 | n = p * q
9 | phi = (p - 1) * (q - 1)
10 lambda n = (p - 1) * (q - 1)
11 | d = inverse(e, int(lambda_n))
12 | message = bytes_to_long(b'This is right')
13 \mid c = pow(message, e, n)
14 \mid m = pow(c, d, n)
15 | print("加密后的密文为: ", c)
16 | print("解密后的明文为: ",long_to_bytes(m))
```

利用中国剩余定理加速 RSA 解密的代码:

```
1 #sagemath
2 e = 3
3 n = 3026533
4 p=1511
5 q=2003
6 d = 2015347
7 c=152702
8 dp = d % (p - 1)
```



```
9 dq = d % (q - 1)
10 mp = pow(c, dp, p)
11 mq = pow(c, dq, q)
12 m= crt([int(mp),int(mq)],[int(p),int(q)])#使用中国剩余定理解密
13 print("解密后的明文为: ",m)
```

#### 测试解密时间的代码:

```
1 #sagemath
2 import time
3 | e = 3
4 n = 3026533
5 p=1511
6 q=2003
7 d = 2015347
   c = 152702
8
9
10 | start_time1 = time.perf_counter_ns()
11 | m1 = pow(c, d, n)
12 | end_time1 = time.perf_counter_ns()
13 | print("未使用CRT加速, 解密后的明文为: ", m1)
  print(f"未使用CRT加速,解密时间:{end_time1-start_time1} ns")
14
15
16 | start_time2 = time.perf_counter_ns()
17 | dp = d \% (p - 1)
18 | dq = d \% (q - 1)
19 \mid mp = pow(c, dp, p)
20
   mq = pow(c, dq, q)
21
   m2= crt([int(mp),int(mq)],[int(p),int(q)])
22
   end_time2 = time.perf_counter_ns()
23
24 | print("使用CRT加速, 解密后的明文为: ", m2 )
  |print(f"使用CRT加速, 解密时间:{end_time2-start_time2} ns")
25
26 | print("加速比: ",(end_time1-start_time1)/(end_time2-start_time2))
```