# 实 验 报 告


课程名称 密码学基础

学　　院 计算机学院

专　　业 保密管理

姓　　名 王君可

学　　号 **17300240009**

开 课 时 间 **2019** 至 **2020** 学年第 二 学期

| 实验项目名　　称 | RSA | 成绩 | |
|---|---|---|---|

## 一、实验目的

1. implement the RSA encryption 实现 RSA 加密
2. understand factoring the public key to decrypt the RSA 理解模数分解解密 RSA
3. understand the common modulus attack to decrypt the RSA 理解公共模数攻击

## 二、实验内容

1. implement the RSA encryption

   1.1 key generation

   1.2 encryption & decryption

2. breaking RSA

   2.1 factoring the public key

   2.2 common modulus attack

## 三、实验步骤

1. 实现 RSA 算法

   实现$multiplicative_inverse(e, phi)$

   实现$key_generation(p, q)$

   实现$encrypt(pk, plaintext)$

   实现$decrypt(sk, ciphertext)$

2. 分解模数破解 RSA

   使用$openssl$*解析*$pubkey.pem$中的参数

   将 n 由十六进制转为十进制

   对大整数 n 分解为 p 和 q

   求出 d

   读入$secret.enc -> bytes2num ->$ *私钥d解密* $-> num2str ->$ *输出明文*

3. 公共模数攻击

   在 common_moduls.py 中实现攻击脚本并破解

The detailed implementation process is in the PPT.

## 四、　实验结果及分析

1. Use '*iamjuke*' as plaintext, the ciphertext is:

```
08be1ae53f2d8b7dc21677b2197878310865c9a101acea34
iamjuke
time for random generation:  19.851417064666748
time for key generation:  0.000225067138671875
time for encryption:  0.00020074844360351562
time for decryption:  0.00017189979553222656
```

and by implementing the decryption function, we obtain the *'iamjuke'*, which is the same with plaintext.
备注：我在写加密和解密代码的时候并不是把明文当作数来加密的，二是一串字符串，先转为 16 进制的字符串然后进行的加密

2.

| Experiment | Time(s) |
|---|---|
| Random value generation | 141.886209 (average of 10 times) |
| Key generation | 0.00030398 |
| encryption | 0.00018000 |
| decryption | 0.00015997 |

The random value generation is time-costly, because judging whether a big number (10^9~10^11) costs much time.

3.

从文件中提取到 e 是 65537，n 是 'C2636AE5C3D8E43FFB97AB09028F1AAC6C0BF6CD3D70EBCA281BFFE97FBE30DD'，十进制表示为 '87924348264132406875276140514499937145050893665602592992418171647042491658461'，其可以被分解为：

```
p = 275127860351348928173285174381581152299
q = 319576316814478949870590164193048041239
```

然后通过计算得到，d 的值是 10866948760844599168252082612378495977388271279679231539839049698621994994673。

```
7/bin/python3 /Users/macbook/.vscode/extensions/ms-python.python-2020.3.71659/pythonFiles/lib/python
s/debugpy/launcher /Users/macbook/Desktop/tmp.py
87924348264132406875276140514499937145050893665602592992418171647042491658461
84905763485077958193641410886067237517896375906905759069003959545809446517649
private key:
10866948760844599168252082612378495977388271279679231539839049698621994994673
plaintext:
re: vFQQvManyQuestionMarks???

macbook@macbookdeMacBook-Pro ~ %
```

在获得了 n 并进行分解得到 p、q 之后，调用 invert 函数得到 d,然后我首先自己实现了解密函数，得到的结果如上；

```
macbook@macbookdeMacBook-Pro ~ % /Users/macbook/Desktop/test.py
zsh: permission denied: /Users/macbook/Desktop/test.py
macbook@macbookdeMacBook-Pro ~ % python3 /Users/macbook/Desktop/test.py
10866948760844599168252082612378495977388271279679231539839049698621994994673
macbook@macbookdeMacBook-Pro ~ % cd /Users/macbook/Desktop/密码学/实验/hw3
macbook@macbookdeMacBook-Pro hw3 % openssl
OpenSSL> rsautl -decrypt -in secret.enc -inkey private.pem
ManyQuestionMarks???
OpenSSL>
```

然后我尝试了通过$openssl$提供的解密接口进行解密，把得到的 p、q、d 写入$private.pem$文件进行解密结果如上。“ManyQuestionMarks???”

4.

首先计算 e1*s1+e2*s2=1 解出 s1 和 s2.

s1 和 s2 首先解出为-6 和 7，

然后 s1<0,s1 转为 6,c1 转为 gmpy2.invert(c1,n)

```
macbook@macbookdeMacBook-Pro ~ %  env DEBUGPY_LAUNCHER_PORT=58721 /Library/Frameworks/Python.framewo
/macbook/.vscode/extensions/ms-python.python-2020.3.71659/pythonFiles/lib/python/debugpy/wheels/debu
p/密码学/实验/hw3/common_modulus_scaffold.py
[+] Started attack...
message:  327031397008878782014591296266113673698907130010957 96
[+] Attack finished!

Plaintext:
 When does school start
macbook@macbookdeMacBook-Pro ~ %
```

The decryption result using common modular is “When does school start”.

## 五、 实验总结

1. RSA 加密的时候通过扩展欧几里得算法（辗转相除法）来计算得到$\boldsymbol{e*d==1 mod \varphi(n)}$的解，通过这个侧面说明了如果$\boldsymbol{gcd}$(e1,e1)=1,总可以得到 s1,s2 使得 e1*s1+e2*s2=1
2. 利用幂指数模运算的特性对求指数运算优化，可以极大的提高 RSA 加密的效率
3. Python 的 gmqy2 库提供了丰富而强大的高精度计算的函数
4. 判断一个大整数是否为素数，如果利用（1～$\sqrt{n}$）逐个比较的方法时间复杂度比较高，利用 gmqy2 的$isprime$（）可以提高性能
5. 总体来说 RSA 加密解密算法利用了数论中的大整数质因数分解的难问题，实现起来并不复杂
6. 关于利用分解模数破解 RSA 的方法，需要借助一些外部的工具分解大的整数，对解密需要的信息少，但是对计算能力要求高
7. 关于公共模数破解的方法，要求 e1\e2 互质、使用相同的模数，这一点要求比较高，需要的信息也多，但是需要的计算能力较弱。
8. 总的来说破解 RSA 的难度比较高，所以 RSA 相对来说是一种较为安全的加密方案。