# 信息安全数学基础大作业：
# RSA 公钥密码系统的实现

徐志鹏 517021910601

April 6, 2019

## 1 作业要求

编程实现 RSA 密码系统：

1. 随机产生大素数 $p, q$（位长 14-bit）以及 $p \cdot q = n$.
2. 随机产生公私钥对 $(e, n)$ 及 $d$.
3. 对消息“m = Mathematical Fundation of Information security + 201904051 + 学号”进行数字化
4. 对消息 m 加密和解密

## 2 文件说明

我一共使用 Python 语言编写了 7 个文件，其相关功能说明如下：

- ✓ **Eratoshenes.py** - 使用平凡除法/厄拉托塞斯筛法计算小于 10000 的 1229 个素数，计算结果放入 Prime.py 文件中以供后续使用。
- ✓ **fast_power.py** - 模重复平方计算法，用以快速计算形如 $b^n \ (mod\ m)$ 的算式。
- ✓ **FermatTest.py** - 费马素性检验，用以检验随机生成的奇数是否是一个伪素数。
- ✓ **Prime.py** - 保存了小于 10000 的 1229 个素数。
- ✓ **RandBigPrime.py** - 随机产生大素数
- ✓ **rsa.py** - 主程序，完成消息输入，参数计算，加密解密的功能。
- ✓ **str_rsa.py** - 完成字符串的十六进制 ASCII 表示与整数之间格式的转换。

## 3 运行

### 3.1 程序依赖

- 需要 Python3.x 版本，Python2.x 版本无法正常运行。
- 需要且只需要 Python 的一个标准库 random 库用以生成随机整数。

### 3.2 样例结果

```
message to encrypt: Mathematical Fundation of Information security + 201904051 +
 517021910601
message  encrypted: 26826431511141003658128711762377292869628229855812575572426 3
071536304812736504448333238233279939716582248895477141202647571923254277729575 01
165777051930650112859637768483639582415585865943446731296773654082575926634141 71
002334321066882056233358097614090020451246544194052273784051842021713593539541 70
172249641745936569895351367353278665035952729105880987344214251728975025147111 85
341150840076631115306222497838230036515809557660625142504053498545773389855003 99
801926872006739624481633408373834834113314290008263171427723097844214348559444 69
187323378698008592897782774966422402141234960643590908708541795434499590736
decrypted  message: Mathematical Fundation of Information security + 2904051 + 5
17021910601
```

Figure 1: 样例输出

# 4 总体思路

## 4.1 产生大素数的思路

首先用系统随机函数产生一个随机整数，若为偶数则加一成为奇数，然后使用 10000 内的素数试除，确定不存在小于 10000 的因子后进行费马素性检验，在多轮费马素性检验且成功之后可以假设该伪素数即为素数。

## 4.2 加密思路

由于每个字符可以转换为其对应的 ascii 码表示，所以将用户输入的一个字符串作为整体看作是一个大整数。

# 5 具体实现步骤

从程序功能从基本到全局的角度来说：

- ➔ 首先，使用厄拉托塞斯筛法打表计算小于 10000 的 1229 个素数，计算结果会存储起来以供后续使用。
- ➔ 实现模重复平方计算功能。
- ➔ 实现大整数随机生成。每一位都随机从 0 到 $F$ 中选择以生成十六进制格式的整数。
- ➔ 实现费马素性检验以判断任何给定的素数是否是伪素数。
- ➔ p，q 全部按照 RSA-1024 标准采用了 1028-bit 长的格式，这样可以加密较长的字符串。
- ➔ 实现计算逆元的功能，用到了 Bezout 定理。
- ➔ 考虑了待加密字符串长度过长的情况，若待加密的数字大于 n，则只截取字符串前面的部分字节加密。
- ➔ 主文件：**rsa.py**，先随机生成 $p, q$，计算 $n, \phi(n)$，再随机生成与 $\phi(n)$ 互素的 $e$，$e^{-1} = d\ (mod\ \phi(n))$。
- ➔ 加密：$ciphertext \equiv plaintext^e \pmod{n}$
- ➔ 解密：$plaintext \equiv ciphertext^d \pmod{n}$

# 6 参考


- RSA 周边——大素数是怎样生成的
  https://bindog.github.io/blog/2014/07/19/how-to-generate-big-primes/
- 使用 python 生成固定长度的随机字符串
  https://www.oschina.net/code/snippet_153443_4752