密码学课程设计

RSA加解密

申恒恒

公钥密码

概述

公钥密码

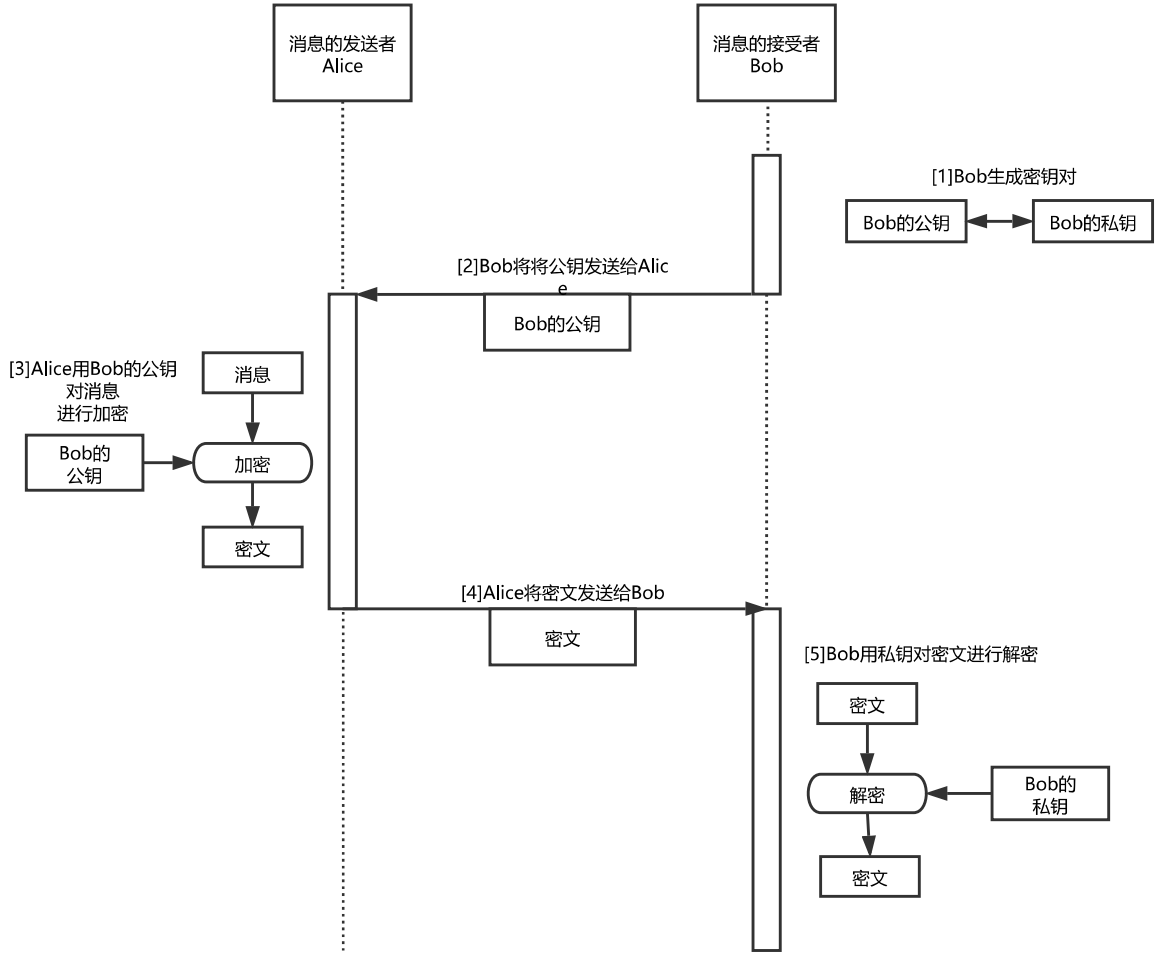
concept

公开密钥加密（英语：public-key cryptography，又译为公开密钥加密），也称为非对 称加密（asymmetric cryptography），一种密码学算法类型，在这种密码学方法中，需要一 对密钥，一个是私人密钥，另一个则是公开密钥。这两个密钥是数学相关，用某用户密钥 加密后所得的信息，只能用该用户的解密密钥才能解密。如果知道了其中一个，并不能计 算出另外一个。因此如果公开了一对密钥中的一个，并不会危害到另外一个的秘密性质。 称公开的密钥为**公钥**；不公开的密钥为**私钥**

公钥通信的流程

- Bob生成一个包含公钥和私钥的密钥对私钥由Bob自行妥善保管
- Bob将自己的公钥发送给Alice
- Alice使用Bob的公钥对消息加密
- Alice将密文将密文发送给Bob

公钥通信的流程


消息的发送者
Alice
消息的接受者
Bob
[1]Bob生成密钥对
Bob的公钥
Bob的私钥
[2]Bob将将公钥发送给Alic
e
Bob的公钥
[3]Alice用Bob的公钥
对消息
进行加密
消息
Bob的
公钥
加密
密文
[4]Alice将密文发送给Bob
密文
[5]Bob用私钥对密文进行解密
密文
解密
Bob的
私钥
密文

RSA算法

概述

RSA历史来源

RSA加密算法是一种非对称加密算法。在公开密钥加密和电子商业中RSA被广泛使 用。RSA是1977年由Ron Rivest、阿迪 萨莫尔Adi Shamir和伦纳 德 阿德曼Leonard Adleman一起提出的。当时他们三人都在麻省理工学院工作。RSA就 是他们三人姓氏开头字母拼在一起组成的

RSA 加密

假设Bob想给Alice送一个消息m，他知道Alice产生的N和E。他使用起先与Alice约好的格式将m转换为一个小于N，且与N互质的整数n，比如他可以将每一个字转换为这个字的Unicode码，然后将这些数字连在一起组成一个数字。假如他的信息非常长的话，他可以将这个信息分为几段，然后将每一段转换为n。用下面这个公式他可以将n加密为c：

$$c \equiv n^E \pmod{N}$$

计算c并不复杂。Bob算出c后就可以将它传递给Alice。

在RSA中，明文、密钥和密文都是数字，在上面的公式里,E和N的组合就是公钥，即，任何人只要拿到这两个数字，就可以完成加密的操作。

RSA解密

Alice得到Bob的消息c后就可以利用她的密钥D来解码。她可以用以下这个公式来将c转换为n：

$$n \equiv c^D \pmod{N}$$

得到n后，她可以将原来的信息m重新复原。 解码的原理是

$$c^D \equiv n^{E \cdot D} \pmod{N}$$

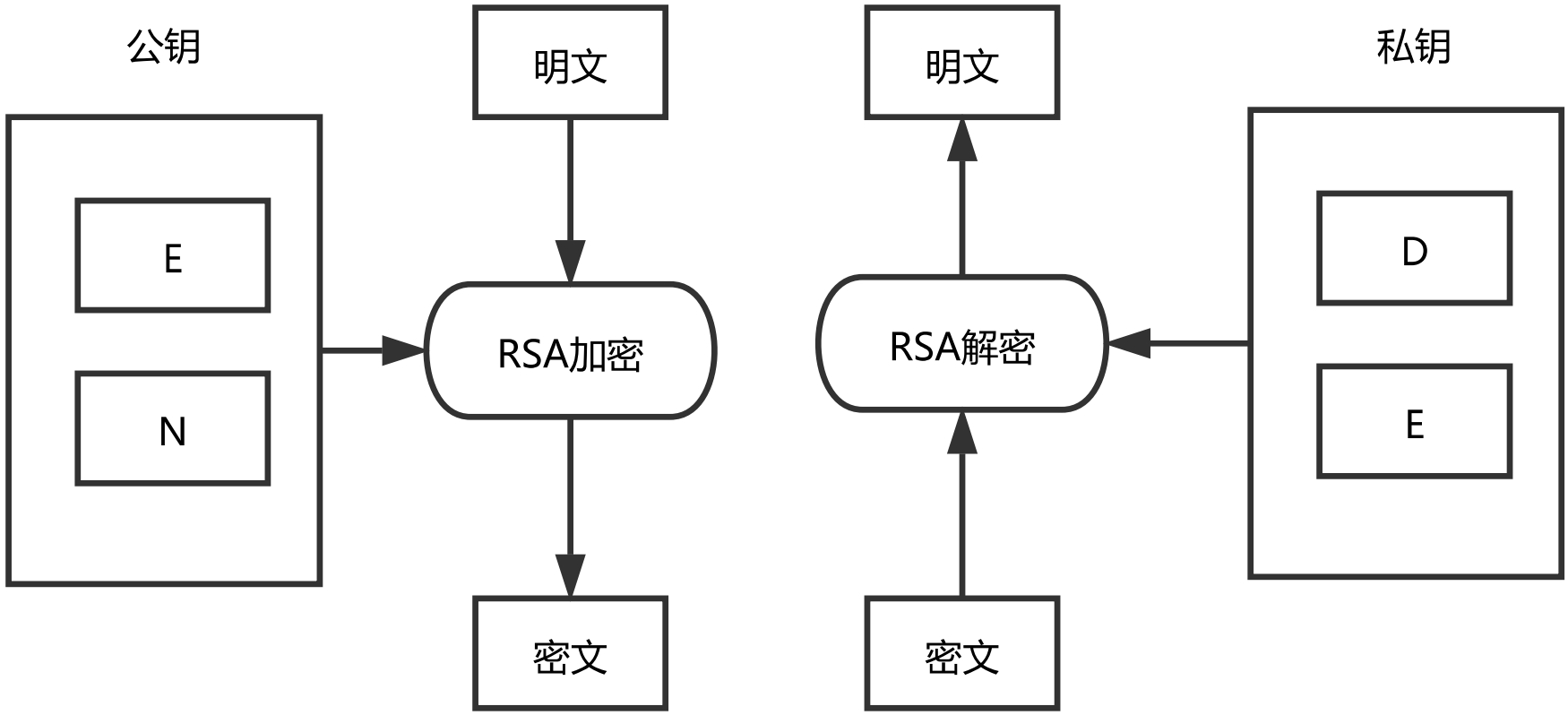
已知$ED \equiv 1 \pmod{r}$，即 $ED = 1 + h\varphi(N)$。 由欧拉定理得：

$$n^{ED} = n^{1+h\varphi(N)} = n\left(n^{\varphi(N)}\right)^h \equiv n(1)^h \pmod{N} \equiv n \pmod{N}$$

注意在这里使用的N和RSA加密是的N是一样的，那么数字D和N的组合就是RSA的私钥.

RSA的加密和解密

workflow chart



密钥对的生成

由于E和N是公钥，D和N是私钥，因此求E、D和N这三个数就是生成密钥对，RSA生成密钥对的生成步骤如下：

- 求N
- 求L
- 求E
- 求D

计算 N

首先准备两个很大的质数 p 和 q 它们的乘积就是 N

$$N = p \cdot q$$

这里选择 $p = 29, q = 31$ 则 $N = p * q =$ 899。

计算 L

L是在RSA的加密和解密过程中都不出现，只出现在生成密钥对的过程中。L是$p-1$和$q-1$的最小公倍数，用$lcm(X,Y)$来表示“X和Y的最小公倍数”，则L可以写作如下式子：

$$L = lcm(p-1, q-1)$$

在例子中，计算 $L = (p-1)\cdot(q-1) = 840$。

计算 E

E是一个比1大比L小的数，另外，E和L的最大公约数(gcd)必须为1，若用gcd表示X和Y的最大公约数，则E和L之间存在如下关系：

$$1 < E < L$$

$$gcd(E, L) = 1$$

对于例子来说，这里选择 E = 37。

计算 D

数D是由E计算得到的，D、E和L之间必须具备如下关系：

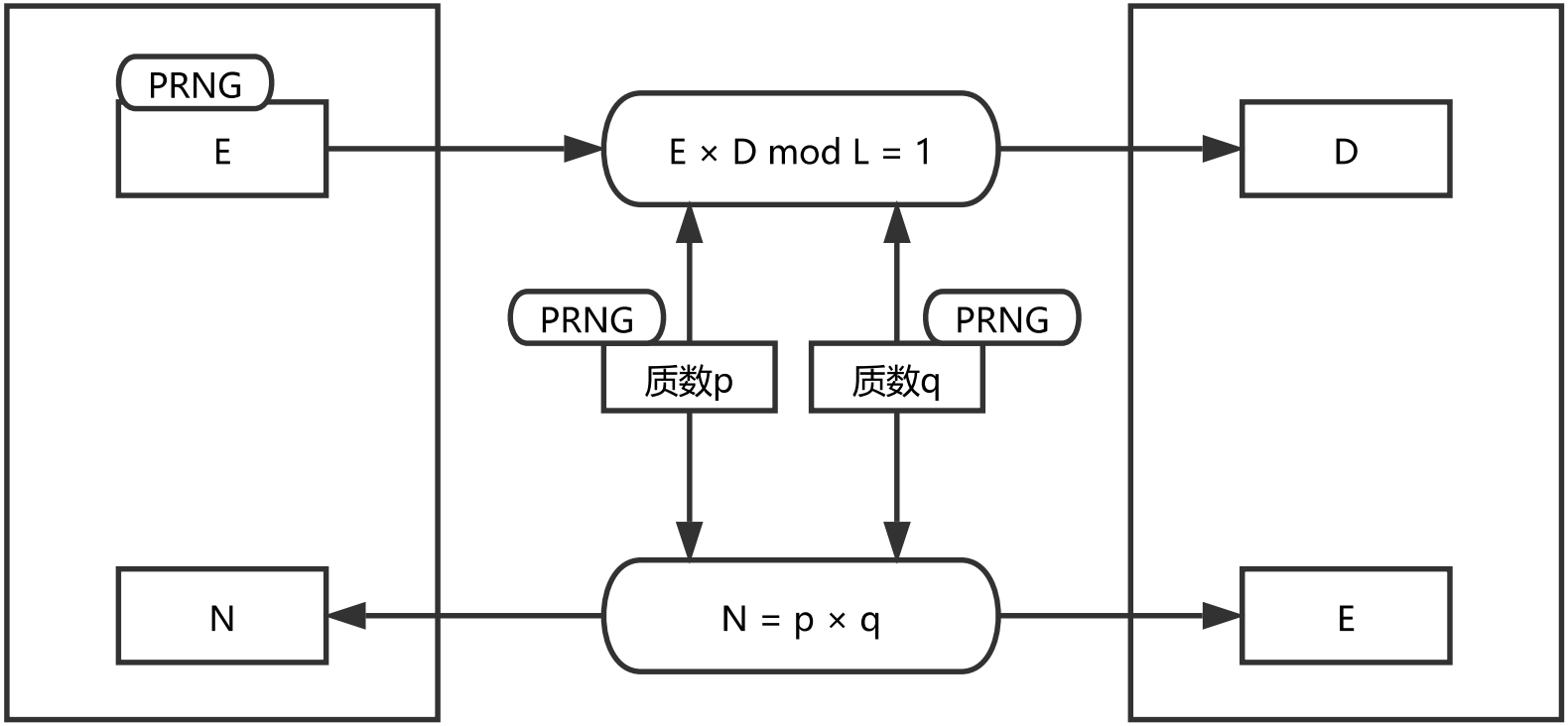
$$1 < D < L$$

$$E \times D \bmod L = 1$$

在例子中用程序算的，算出来最小的 D 就是 613

现在，得到了 N,D,E ，把 p 和 q 扔掉，(n, e)作公钥，(n, d)作私钥，就可以执行 RSA 加密运算了。

RSA的加密和解密—详细工作流程



PRNG 表示伪随机数生成器

RSA算法

Python实现

总体概述

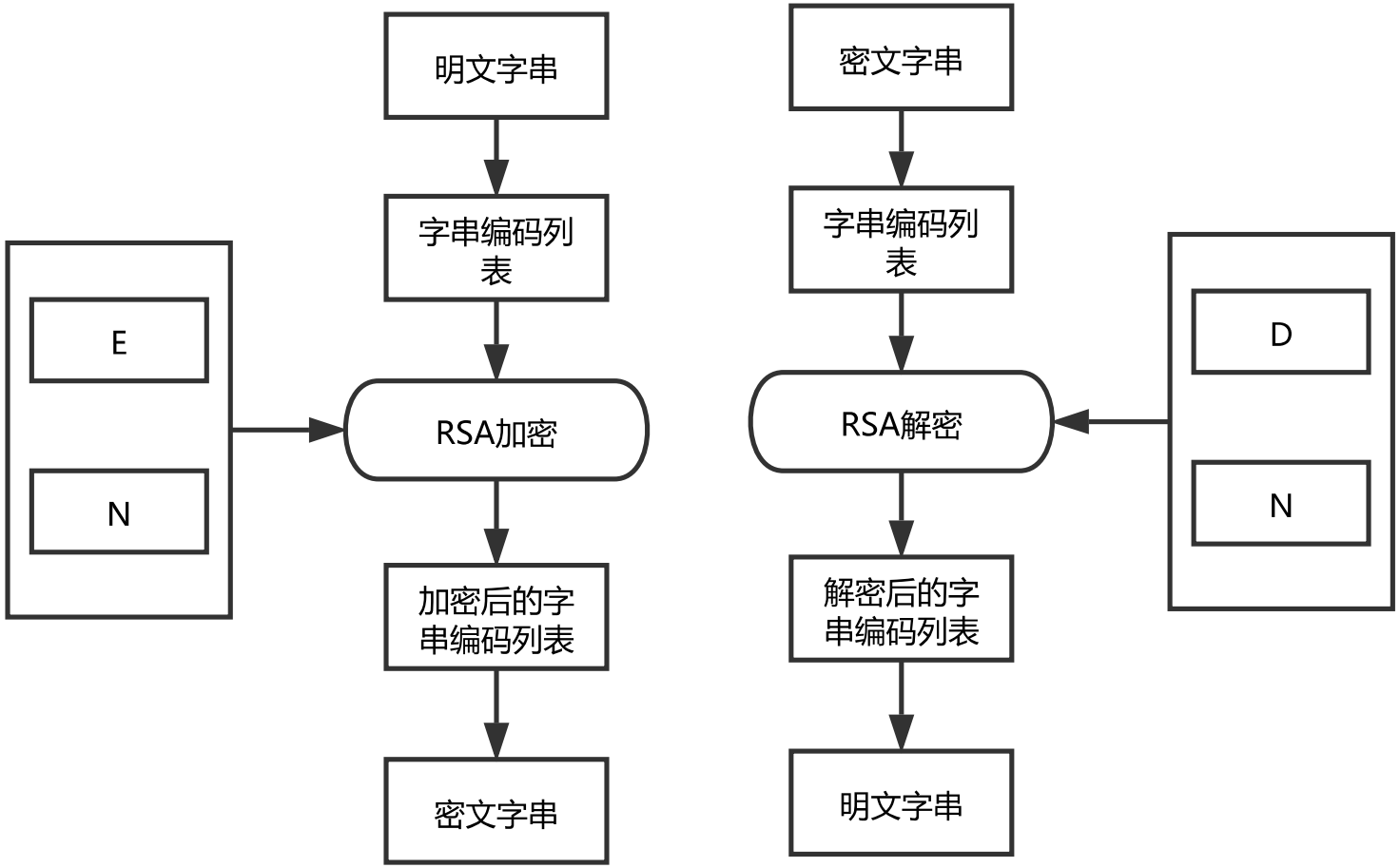
阐述了RSA的基本原理和算法后，利用Python设计和实现一个RSA的加解密程序，在这里我将程序部署在Web上，利用Python的Flask框架结合Python程序作为后台加解密算法的引擎，以极为友好的方式展示出来。

Python作为一门相当高级语言来说，具有简单易用，极为灵活的特点，本次RSA的实现主要根据RSA算法进行设计的，在N,D,E生成，往往会用到伪随机数生成器，但在项目中，只是验证性课题，只是对N,E,D进行了赋值，并没有密钥对的生成环节，换言之，已经将密钥对生成完毕，直接拿过来取加解密就可以，所以该程序主要实现了如下功能:

(1) 在密钥对生成的基础上，如何利用N,E,D对数据(只支持字符串)进行加密和解密

(2) 实现了米勒-拉宾素性测试

工作流程图


明文字串
字串编码列表
E
N
RSA加密
加密后的字串编码列表
密文字串
密文字串
字串编码列表
D
N
RSA解密
解密后的字串编码列表
明文字串

整体设计

```
class RSA(object):
    def __init__(self):...
    def __chr_to_num(self, char):...
    def __num_to_chr(self, num):...
    def __text_to_array(self, text):...
    def __array_to_text(self, array):...
    def __array_to_cipher(self, array):...
    def __cipher_to_array(self, cipher):...
    def RSA_encrypt(self, text):...
    def RSA_decrypt(self, cipher):...
```

核心方法-数据加密的核心算法

```
def calc_mod(x, r, n):
    """Calculate the value of x**r mod n"""
    a, b, c = x, r, 1
    while b != 0:
        while b%2 == 0:
            b = b/2
            a = a*a%n
        else:
            b = b-1
            c = (c*a)%n
    return c
```

计算

$$x^r \mathrm{mod} n$$

核心方法-数据解密的核心算法

```
def calc_reverse(m, n):
    """Calculate the value of m^(-1) mod n"""
    N = n
    m %= n
    m, n = n, m
    q = []
    while True:
        q.append(m//n)
        m, n = n, m%n
        if n == 1:
            break
    P, Q = [1, q[0]], [0, 1]
    for i in range(2, len(q)+1):
        P.append(P[i-2]+P[i-1]*q[i-1])
        Q.append(Q[i-2]+Q[i-1]*q[i-1])
    return (-1)**len(q)*P[-1]%N
```

Miller-Rabin 素性测试算法

```
def is_prime(n, k=5000):
    """Test whether n is a prime, repeat k times."""
    def mill_rab(n):
        b = 0
        while b%2 == 0:
            b = random.randint(2, n-1)      # 产生一个奇数b

        s, m = 0, n-1
        while m%2 == 0:                     # n-1 = (2**s)*m
            m //= 2
            s += 1

        if calc_mod(b, m, n) == 1:
            return True
```

```
    else:
        for r in xrange(s):
            if calc_mod(b, 2**r*m, n) == n-1:
                return True
    return False
if n in [0, 1]:
    return False
elif n == 2:
    return True
for i in range(k):
    if not mill_rab(n):
        return False
return True
```