上一篇在曲速未来安全区中说到的拜占庭协定中，如果10个将军中的几个同时发起消息，势必会造成系统的混乱，造成各说各的攻击时间方案，行动难以一致。谁都可以发起进攻的信息，但由谁来发出呢？其实这只要加入一个成本就可以了，即：一段时间内只有一个节点可以传播信息。当某个节点发出统一进攻的消息后，各个节点收到发起者的消息必须签名盖章，确认各自的身份。

如今，非对称加密技术完全可以解决这个签名问题。

非对称加密技术，在现在网络中，有非常广泛应用。加密技术更是数字货币的基础。

所谓非对称，就是指该算法需要一对密钥，使用其中一个（公钥）加密，则需要用另一个（私钥）才能解密。非对称加密，加密与解密使用的密钥不是同一密钥，对中一个对外公开，称为公钥，另一个只有所有者知道，称为私钥。

用公钥加密的信息只有私钥才能解开，反之，用私钥加密的信息只有公钥才能解开（签名验签）。

代表：RSA算法。速度慢，适合少量数据加密。对称加密算法不能实现签名，因此签名只能非对称算法。

RSA算法原理

RSA算法的基于这样的数学事实：两个大质数相乘得到的大数难以被因式分解。如：有很大质数p跟q，很容易算出N，使得 N = p * q，但给出N, 比较难找p q（没有很好的方式， 只有不停的尝试）

这其实也是单向函数的概念。

下面来看看数学演算过程：

选取两个大质数p，q，计算N = p * q 及 φ（N）= φ (p) * φ (q) = (p-1) * (q-1)

质数(prime numbe)：又称素数，为在大于1的自然数中，除了1和它本身以外不再有其他因数。

互质关系：如果两个正整数，除了1以外，没有其他公因子，我们就称这两个数是互质关系（coprime）。

φ(N)：叫做欧拉函数，是指任意给定正整数N，在小于等于N的正整数之中，有多少个与N构成互质关系。

1. 欧拉函数

定义：对于一个正整数 n ，小于 n 且和 n 互质的正整数（包括 1）的个数，记作 φ(n).则

证明：

如果n是一个质数，那么φ(n) = n-1

如果n是一个质数p的幂，即n = p^k，则φ(n) = p^k-p^(k-1) = (p-1)*p^(k-1)

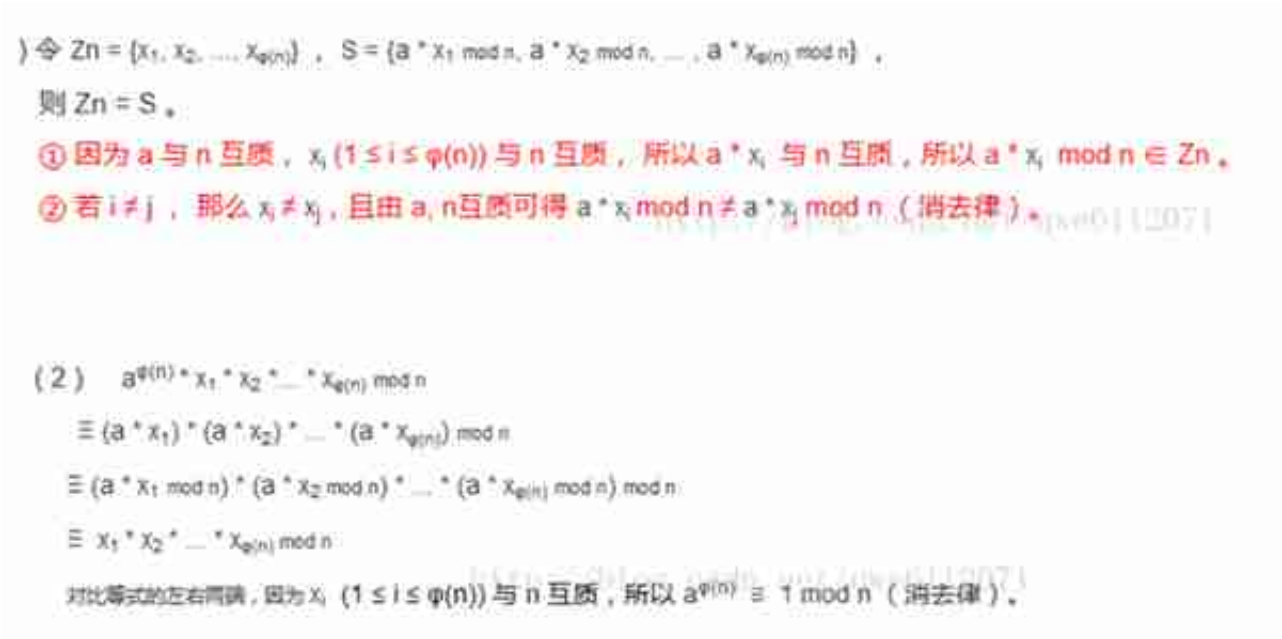
欧拉函数是一个积性函数，当n,m互质的时候，φ(n*m) = φ(n)*φ(m)

2. 欧拉定理

1.定义：如果两个正整数a和n互质，则n的欧拉函数 φ(n) 可以让下面的等式成立：

a^φ(n)%n=1

证明：



2.选择一个大于1 小于φ(N)的数e，使得 e 和 φ(N)互质

3.计算d，使得de=1 mod φ(N) 等价于方程式 ed-1 = k φ(N) 求一组解。

4. (N, e)封装成公钥，(N, d)封装成私钥。假设m为明文，加密就是算出密文c:m^e mod N = c (明文m用公钥e加密并和随机数N取余得到密文c)解密则是：c^d mod N = m (密文c用密钥解密并和随机数N取余得到明文m)

加解密步骤

具体还是来看看步骤，举个例子，假设Alice和Bob又要相互通信。

Alice 随机取大质数P1=53，P2=59，那N=53*59=3127，φ(N)=3016

取一个e=3，计算出d=2011。

只将N=3127，e=3 作为公钥传给Bob（公钥公开）

假设Bob需要加密的明文m=89，c = 89^3 mod 3127=1394，于是Bob传回c=1394。（公钥加密过程）

Alice使用c^d mod N = 1394^2011 mod 3127，就能得到明文m=89。（私钥解密过程）

安全性分析

那么，有无可能在已知n和e的情况下，推导出d？

如果n可以被因数分解，d就可以算出，因此RSA安全性建立在N的因式分解上。大

整数的因数分解，是一件非常困难的事情。只要密钥长度足够长，用RSA加密的信息实际上是不能被解破的。

RSA核心算法

快速幂取模算法

算法1：连乘算法，时间复杂度O(n)

算法2：快速幂算法，时间复杂度O(logn)

```
    def myPow(self, x, n):
        def rec_pow(x,n):
            if n == 0:
                return 1
            elif n == 1:
                return x
            return rec_pow(x*x,n / 2) * rec_pow(x,n&1)
        if n >= 0:
            return rec_pow(x,n)
        else:
            return 1/rec_pow(x,-n)
```

算法3：快速幂取模算法，时间复杂度O(logn)

```
def superPow(self, a, b,m):
    def myPow(a,b):
        if b == 0:
            return 1
        a = a%m
        if b == 1:
            return a
        return myPow(a*a,b/2) * myPow(a,b&1) % m
    res = myPow(a,b)
    return res
```

素数判定算法：

```python
## 以一定概率判断是否为素数
def primeTest(n):
    q = n - 1
    k = 0
    #Find k, q, satisfied 2^k * q = n - 1
    while q % 2 == 0:
        k += 1;
        q /= 2
    a = random.randint(2, n-2);
    #If a^q mod n= 1, n maybe is a prime number
    if fastExpMod(a, q, n) == 1:
        return True
    #If there exists j satisfy a ^ ((2 ^ j) * q) mod n == n-1,
    #n maybe is a prime number
    for j in range(0, k):
        if fastExpMod(a, (2**j)*q, n) == n - 1:
            return True
    #a is not a prime number
    return False
#多次测试，知道获取基本不可能为素数的"奇数"
def findPrime(halfkeyLength):
    while True:
        #Select a random number n
        n = random.randint(0, 1<<halfkeyLength)
        if n % 2 != 0:
            found = True
            #If n satisfy primeTest 10 times, then n should be a prime number
            for _ in range(0, 10):
                if not primeTest(n):
                    found = False
                    break
            if found:
                return n
```

由于非对称加密算法的运行速度比对称加密算法的速度慢很多，当我们需要加密大量的数据时，建议采用对称加密算法，提高加解密速度。

对称加密算法不能实现签名，因此签名只能非对称算法。

由于对称加密算法的密钥管理是一个复杂的过程，密钥的管理直接决定着他的安全性，因此当数据量很小时，我们可以考虑采用非对称加密算法。

在实际的操作过程中，我们通常采用的方式是：采用非对称加密算法管理对称算法的密钥，然后用对称加密算法加密数据，这样我们就集成了两类加密算法的优点，既实现了加密速度快的优点，又实现了安全方便管理密钥的优点。