首先, 找出三个数, p, q, r,  
　　其中 p, q 是两个相异的质数, r 是与 (p-1)(q-1) 互质的数......  
　　p, q, r 这三个数便是 private key

接著, 找出 m, 使得 rm == 1 mod (p-1)(q-1).....  
　　这个 m 一定存在, 因为 r 与 (p-1)(q-1) 互质, 用辗转相除法就可以得到了.....  
　　再来, 计算 n = pq.......  
　　m, n 这两个数便是 public key

1978年就出现了这种算法，它是第一个既能用于[数据加密](http://zhidao.baidu.com/search?word=%E6%95%B0%E6%8D%AE%E5%8A%A0%E5%AF%86&fr=qb_search_exp&ie=utf8)也能用于[数字签名](http://zhidao.baidu.com/search?word=%E6%95%B0%E5%AD%97%E7%AD%BE%E5%90%8D&fr=qb_search_exp&ie=utf8)的算法。  
它易于理解和操作，也很流行。算法的名字以发明者的名字命名：Ron Rivest, Adi  
Shamir 和Leonard Adleman。但RSA的安全性一直未能得到理论上的证明。  
  
RSA的安全性依赖于大数分解。[公钥](http://zhidao.baidu.com/search?word=%E5%85%AC%E9%92%A5&fr=qb_search_exp&ie=utf8)和[私钥](http://zhidao.baidu.com/search?word=%E7%A7%81%E9%92%A5&fr=qb_search_exp&ie=utf8)都是两个大[素数](http://zhidao.baidu.com/search?word=%E7%B4%A0%E6%95%B0&fr=qb_search_exp&ie=utf8)(质数又称素数。指在一个大于1的自然数中，除了1和此整数自身外，不能被其他自然数(不包括0)整除的数。)（ 大于 100  
个[十进制](http://zhidao.baidu.com/search?word=%E5%8D%81%E8%BF%9B%E5%88%B6&fr=qb_search_exp&ie=utf8)位）的函数。据猜测，从一个密钥和密文推断出明文的难度等同于分解两个  
大[素数](http://zhidao.baidu.com/search?word=%E7%B4%A0%E6%95%B0&fr=qb_search_exp&ie=utf8)的积。   
  
密钥对的产生。选择两个大[素数](http://zhidao.baidu.com/search?word=%E7%B4%A0%E6%95%B0&fr=qb_search_exp&ie=utf8)，p 和q 。计算：   
  
n = p \* q  
  
然后随机选择加密密钥e，要求 e 和 ( p - 1 ) \* ( q - 1 ) [互质](http://zhidao.baidu.com/search?word=%E4%BA%92%E8%B4%A8&fr=qb_search_exp&ie=utf8)(公约数只有1和-1的两个整数，叫做互质整数·公约数只有1的两个自然数，叫做互质自然数·后者是前者的特殊情形·10,13的最大公因数是1，因此这是整数互质。)。最后，利用  
Euclid 算法计算解密密钥d, 满足   
  
e \* d = 1 ( mod ( p - 1 ) \* ( q - 1 ) )  
  
其中n和d也要[互质](http://zhidao.baidu.com/search?word=%E4%BA%92%E8%B4%A8&fr=qb_search_exp&ie=utf8)。数e和  
n是[公钥](http://zhidao.baidu.com/search?word=%E5%85%AC%E9%92%A5&fr=qb_search_exp&ie=utf8)，d是[私钥](http://zhidao.baidu.com/search?word=%E7%A7%81%E9%92%A5&fr=qb_search_exp&ie=utf8)。两个素数p和q不再需要，应该丢弃，不要让任何人知道。   
  
加密信息 m（[二进制](http://zhidao.baidu.com/search?word=%E4%BA%8C%E8%BF%9B%E5%88%B6&fr=qb_search_exp&ie=utf8)表示）时，首先把m分成等长[数据块](http://zhidao.baidu.com/search?word=%E6%95%B0%E6%8D%AE%E5%9D%97&fr=qb_search_exp&ie=utf8) m1 ,m2,..., mi ，块长s  
，其中 2^s <= n, s 尽可能的大。对应的密文是：  
  
ci = mi^e ( mod n ) ( a )  
  
解密时作如下计算：  
  
mi = ci^d ( mod n ) ( b )  
  
RSA 可用于[数字签名](http://zhidao.baidu.com/search?word=%E6%95%B0%E5%AD%97%E7%AD%BE%E5%90%8D&fr=qb_search_exp&ie=utf8)，方案是用 ( a ) 式签名， ( b )  
式验证。具体操作时考虑到安全性和 m[信息量](http://zhidao.baidu.com/search?word=%E4%BF%A1%E6%81%AF%E9%87%8F&fr=qb_search_exp&ie=utf8)较大等因素，一般是先作 HASH 运算。  
  
RSA 的安全性。  
RSA的安全性依赖于大数分解，但是否等同于大数分解一直未能得到理论上的证明，因  
为没有证明破解  
RSA就一定需要作大数分解。假设存在一种无须分解大数的算法，那它肯定可以修改成  
为大数分解算法。目前， RSA  
的一些变种算法已被证明等价于大数分解。不管怎样，分解n是最显然的攻击方法。现  
在，人们已能分解140多个[十进制](http://zhidao.baidu.com/search?word=%E5%8D%81%E8%BF%9B%E5%88%B6&fr=qb_search_exp&ie=utf8)位的大素数。因此，[模数](http://zhidao.baidu.com/search?word=%E6%A8%A1%E6%95%B0&fr=qb_search_exp&ie=utf8)n  
必须选大一些，因具体适用情况而定。   
  
RSA的速度。  
由于进行的都是大数计算，使得RSA最快的情况也比DES慢上100倍，无论是软件还是硬  
件实现。速度一直是RSA的缺陷。一般来说只用于少量[数据加密](http://zhidao.baidu.com/search?word=%E6%95%B0%E6%8D%AE%E5%8A%A0%E5%AF%86&fr=qb_search_exp&ie=utf8)。   
  
RSA的[选择密文攻击](http://zhidao.baidu.com/search?word=%E9%80%89%E6%8B%A9%E5%AF%86%E6%96%87%E6%94%BB%E5%87%BB&fr=qb_search_exp&ie=utf8)。  
RSA在[选择密文攻击](http://zhidao.baidu.com/search?word=%E9%80%89%E6%8B%A9%E5%AF%86%E6%96%87%E6%94%BB%E5%87%BB&fr=qb_search_exp&ie=utf8)面前很脆弱。一般攻击者是将某一信息作一下伪装(  
Blind)，让拥有[私钥](http://zhidao.baidu.com/search?word=%E7%A7%81%E9%92%A5&fr=qb_search_exp&ie=utf8)的实体签署。然后，经过计算就可得到它所想要的信息。实际上  
，攻击利用的都是同一个弱点，即存在这样一个事实：[乘幂](http://zhidao.baidu.com/search?word=%E4%B9%98%E5%B9%82&fr=qb_search_exp&ie=utf8)保留了输入的乘法结构：  
  
( XM )^d = X^d \*M^d mod n  
  
前面已经提到，这个固有的问题来自于[公钥密码](http://zhidao.baidu.com/search?word=%E5%85%AC%E9%92%A5%E5%AF%86%E7%A0%81&fr=qb_search_exp&ie=utf8)系统的最有用的特征--每个人都能使  
用[公钥](http://zhidao.baidu.com/search?word=%E5%85%AC%E9%92%A5&fr=qb_search_exp&ie=utf8)。但从算法上无法解决这一问题，主要措施有两条：一条是采用好的公钥协议  
，保证工作过程中实体不对其他实体任意产生的信息解密，不对自己一无所知的信息  
签名；另一条是决不对陌生人送来的随机文档签名，签名时首先使用One-Way Hash  
Function  
对文档作HASH处理，或同时使用不同的签名算法。在中提到了几种不同类型的攻击方  
法。   
  
RSA的公[共模](http://zhidao.baidu.com/search?word=%E5%85%B1%E6%A8%A1&fr=qb_search_exp&ie=utf8)数攻击。  
若系统中共有一个[模数](http://zhidao.baidu.com/search?word=%E6%A8%A1%E6%95%B0&fr=qb_search_exp&ie=utf8)，只是不同的人拥有不同的e和d，系统将是危险的。最普遍的  
情况是同一信息用不同的[公钥加密](http://zhidao.baidu.com/search?word=%E5%85%AC%E9%92%A5%E5%8A%A0%E5%AF%86&fr=qb_search_exp&ie=utf8)，这些公钥[共模](http://zhidao.baidu.com/search?word=%E5%85%B1%E6%A8%A1&fr=qb_search_exp&ie=utf8)而且[互质](http://zhidao.baidu.com/search?word=%E4%BA%92%E8%B4%A8&fr=qb_search_exp&ie=utf8)，那末该信息无需私钥就  
可得到恢复。设P为信息明文，两个加密密钥为e1和e2，公[共模](http://zhidao.baidu.com/search?word=%E5%85%B1%E6%A8%A1&fr=qb_search_exp&ie=utf8)数是n，则：  
  
C1 = P^e1 mod n  
  
C2 = P^e2 mod n  
  
密码分析者知道n、e1、e2、C1和C2，就能得到P。  
  
因为e1和e2互质，故用Euclidean算法能找到r和s，满足：  
  
r \* e1 + s \* e2 = 1  
  
假设r为负数，需再用Euclidean算法计算C1^(-1)，则  
  
( C1^(-1) )^(-r) \* C2^s = P mod n  
  
另外，还有其它几种利用公共[模数](http://zhidao.baidu.com/search?word=%E6%A8%A1%E6%95%B0&fr=qb_search_exp&ie=utf8)攻击的方法。总之，如果知道给定模数的一对e和d  
，一是有利于攻击者分解模数，一是有利于攻击者计算出其它成对的e’和d’，而无  
需分解模数。解决办法只有一个，那就是不要共享模数n。  
  
RSA的[小指](http://zhidao.baidu.com/search?word=%E5%B0%8F%E6%8C%87&fr=qb_search_exp&ie=utf8)数攻击。 有一种提高  
RSA速度的建议是使公钥e取较小的值，这样会使加密变得易于实现，速度有所提高。  
但这样作是不安全的，对付办法就是e和d都取较大的值。  
  
[RSA算法](http://zhidao.baidu.com/search?word=RSA%E7%AE%97%E6%B3%95&fr=qb_search_exp&ie=utf8)是第一个能同时用于加密和[数字签名](http://zhidao.baidu.com/search?word=%E6%95%B0%E5%AD%97%E7%AD%BE%E5%90%8D&fr=qb_search_exp&ie=utf8)的算法，也易于理解和操作。RSA是被研  
究得最广泛的[公钥算法](http://zhidao.baidu.com/search?word=%E5%85%AC%E9%92%A5%E7%AE%97%E6%B3%95&fr=qb_search_exp&ie=utf8)，从提出到现在已近二十年，经历了各种攻击的考验，逐渐为  
人们接受，普遍认为是目前最优秀的公钥方案之一。RSA  
的安全性依赖于大数的因子分解，但并没有从理论上证明破译RSA的难度与大数分解难  
度等价。即RSA的重大缺陷是无法从理论上把握它的保密性能如何，而且[密码学](http://zhidao.baidu.com/search?word=%E5%AF%86%E7%A0%81%E5%AD%A6&fr=qb_search_exp&ie=utf8)界多数  
人士倾向于因子分解不是[NPC问题](http://zhidao.baidu.com/search?word=NPC%E9%97%AE%E9%A2%98&fr=qb_search_exp&ie=utf8)。  
RSA的缺点主要有：A)产生密钥很麻烦，受到素数产生技术的限制，因而难以做到一次  
一密。B)分组长度太大，为保证安全性，n 至少也要 600 bits  
以上，使运算代价很高，尤其是速度较慢，较对称[密码算法](http://zhidao.baidu.com/search?word=%E5%AF%86%E7%A0%81%E7%AE%97%E6%B3%95&fr=qb_search_exp&ie=utf8)慢几个[数量级](http://zhidao.baidu.com/search?word=%E6%95%B0%E9%87%8F%E7%BA%A7&fr=qb_search_exp&ie=utf8)；且随着大  
数分解技术的发展，这个长度还在增加，不利于[数据格式](http://zhidao.baidu.com/search?word=%E6%95%B0%E6%8D%AE%E6%A0%BC%E5%BC%8F&fr=qb_search_exp&ie=utf8)的标准化。目前，SET(  
Secure Electronic Transaction  
)协议中要求CA采用2048比特长的密钥，其他实体使用1024比特的密钥。  
  
DSS/[DSA算法](http://zhidao.baidu.com/search?word=DSA%E7%AE%97%E6%B3%95&fr=qb_search_exp&ie=utf8)  
  
Digital Signature Algorithm  
(DSA)是Schnorr和ElGamal签名算法的变种，被美国NIST作为DSS(Digital Signature  
Standard)。算法中应用了下述参数：  
p：L bits长的素数。L是64的倍数，范围是512到1024；  
q：p - 1的160bits的素因子；  
g：g = h^((p-1)/q) mod p，h满足h < p - 1, h^((p-1)/q) mod p > 1；  
x：x < q，x为私钥 ；  
y：y = g^x mod p ，( p, q, g, y )为公钥；  
H( x )：One-Way Hash函数。DSS中选用SHA( Secure Hash Algorithm )。  
p, q,  
g可由一组用户共享，但在实际应用中，使用公共模数可能会带来一定的威胁。签名及  
验证协议如下：  
1. P产生[随机数](http://zhidao.baidu.com/search?word=%E9%9A%8F%E6%9C%BA%E6%95%B0&fr=qb_search_exp&ie=utf8)k，k < q；  
2. P计算 r = ( g^k mod p ) mod q  
s = ( k^(-1) (H(m) + xr)) mod q  
签名结果是( m, r, s )。  
3. 验证时计算 w = s^(-1)mod q  
u1 = ( H( m ) \* w ) mod q  
u2 = ( r \* w ) mod q  
v = (( g^u1 \* y^u2 ) mod p ) mod q   
若v = r，则认为签名有效。   
  
DSA是基于整数[有限域](http://zhidao.baidu.com/search?word=%E6%9C%89%E9%99%90%E5%9F%9F&fr=qb_search_exp&ie=utf8)[离散对数](http://zhidao.baidu.com/search?word=%E7%A6%BB%E6%95%A3%E5%AF%B9%E6%95%B0&fr=qb_search_exp&ie=utf8)难题的，其安全性与RSA相比差不多。DSA的一个重要特  
点是两个素数公开，这样，当使用别人的p和q时，即使不知道私钥，你也能确认它们  
是否是随机产生的，还是作了手脚。[RSA算法](http://zhidao.baidu.com/search?word=RSA%E7%AE%97%E6%B3%95&fr=qb_search_exp&ie=utf8)却作不到。