3.4 密码攻击方法

- 3.4.2差分密码分析法
  - 差分密码分析是采用选择明文统计分析来攻击迭代分组密码的。
  - 它主要分析明文差分和密文差分之间的统计相关性。

- 对于给定的r轮迭代密码，对已知分组长为n的明文对B和B‘，其差分$\Delta B=B\otimes B'^{-1}$，其中$\otimes$是所有n-bit构成的集合中所定义的群运算，$B'^{-1}$是B’在群中的逆元。
- 在密钥k控制下，各轮迭代所产生的中间密文差分为：
- $\Delta X(i)=X(i)\otimes(X'(i))^{-1}$　　$0\le i\le r$
- 当i=0时，X(0)=B，X'(0)=B'，$\Delta X(0)=\Delta B$，而最终的密文差分$\Delta X=\Delta X(r)$。
- 由于明文对$B\neq B'$，故$\Delta X(i)\neq e$(群的单位元)，即$\Delta X(i)\in\{0,1\}^n-\{e\}$。
- 每轮迭代所用子密钥k_i与明文统计独立，并认为是服从均匀分布的。

- 定义3.1：设a是两个不同明文B和B'的差分，b是密码第i轮输出X(i)和X'(i)之间的差分，称(a,b)为第i轮差分。在给定明文的差分a=ΔB条件下，第i轮出现一对输出的差分是b的概率称为第i轮差分概率，记为P(b=ΔX(i)|a=ΔB)。
- 以DES加密1轮为例了解差分分析攻击的过程。

- DES的关键是S盒，设S_j是一个特定的S-盒$(1\le j\le 8)$，(B_j, B_j^*)是一对长度为6bit的串。$B_j\oplus B_j^*$作为S_j的输入异或，$S_j(B_j)\oplus S_j(B_j^*)$则为$S_j$的输出异或。
- 首先考虑对给定的$a_j\in Z_2^6$，把所有异或值为a_j的(B_j,B_j^*)拿来构成集合，即$\Delta(a_j)=\{(B_j,B_j^*)|B_j\oplus B_j^*=a_j\}$。
- $|\Delta(a_j)|=2^6=64$，
- 因为$B_j\oplus B_j^*=a_j$，所以$B_j^*=B_j\oplus a_j$，即$\Delta(a_j)=\{(B_j,B_j\oplus a_j)|B_j\in Z_2^6\}$。
- 对$\Delta(a_j)$中的每一对，计算出S_j的一个输出异或，
- 共可计算出64个输出异或，
- S_j盒输出是4bit，只有$2^4=16$个可能的输出值，
- 即64个输出异或分布在16个可能值上，
- 将这些分布列成表，分布的不均匀性就是差分攻击的基础。

- 例3.1：S0的一个输入异或为110100，则Δ(110100)={(000000,110100),(000001, 110101),…, (111111,001011)}。
- 对Δ(110100)中的所有元素计算输出异或，并由此统计各输出的分布，得表：

0000	0001	0010	0011	0100	0101	0110	0111
0	8	16	6	2	0	0	12
1000	1001	1010	1011	1100	1101	1110	1111
6	0	0	0	0	8	0	6

16个可能的输出异或中实际仅出现了8个。实验表明，所有可能的输出异或中实际上出现约75%～80%。

- 对$1\le j\le 8$，长度为6bit的串aj和长度为4bit的串bj(aj和bj为给定的差分，分别是输入差分和输出差分)
- 定义3.2：$INj(aj,bj)=\{Bj\in Z_2^6 | Sj(Bj)\oplus Sj(Bj\oplus aj)=bj\}$，
- $Nj(aj,bj)=|INj(aj,bj)|$
- $Nj(aj,bj)$是S-盒Sj具有输入异或aj和输出异或bj的明文对数量。书上给出的是$IN1(110100,bj)(bj\in Z_2^4)$
- 对8个S盒中的每一个，都有64个可能的输入异或，所以共需计算512个分布，这可通过利用计算机得到。
- 在DES中，第i轮S-盒的输入可表示为$B=E\oplus J$，其中$E=E(R_{i-1})$是R_{i-1}的扩展，$J=K_i$为第i轮的工作密钥。输入异或$B\oplus B^*=(E\oplus J)\oplus(E^*\oplus J)=E\oplus E^*$。输入异或与工作密钥无关。但输出异或则与工作密钥有关。
- $B=B_1B_2B_3B_4B_5B_6B_7B_8$(48bit,每个$B_i$为6bit)

- $E=E_1E_2E_3E_4E_5E_6E_7E_8$
- $J=J_1J_2J_3J_4J_5J_6J_7J_8$
- B^*, E^*也类似表示
- 攻击的目的是获得工作密钥，由此导出主密钥。
- 假定对某个$j(1\le j\le 8)$知道Ej和Ej^*的值，以及Sj的输出异或$bj=Sj(Bj)\oplus Sj(Bj^*)(Bj^*=Bj\oplus aj)$。
- $INj(E'j,bj)=\{Pj\in Z_2^6|Sj(Pj)\oplus Sj(Pj\oplus E'j)=bj\}$，这里$E'j= Ej\oplus Ej^*$。则正确的加密结果$Ej\oplus Jj\in INj(E'j,bj)$。

- testj(Ej,Ej*,bj)={Bj⊕Ej|Bj∈INj(E'j,bj)}。该集合是所有INj(E'j,bj)中元素与Ej异或的值全体，故元素个数为|INj(E'j,bj)|=N1(E'j,bj)。
- 由于$E_j \oplus J_j \in IN_j(E'_j, b_j)$,所以$(E_j \oplus J_j) \oplus E_j \in test_j(E_j, E_j^*, b_j)$，
- 即$J_j \in test_j(E_j, E_j^*, b_j)$。
- 因此testj(E'j,Ej*,bj)中必有正确的工作密钥。
- 例3.2：设E1=000001,E1*=110101, b1=1101。故E1'=E1⊕E1*=110100。
- 因为N1(E1',b1)=8，故test1(E'1,E1*,b1)有8个元素。从表中可知IN1(110100,1101)={000110,010000,010110,011100,100010,100100,101000,110010},所以可以求得test1(E'1,E1*,b1)={000111,010001,010111,011101,100011,100101,101001,110011}。

- 如果还有第2组这样的三元组(E‘1,E1*,b1)，就可获得包含工作密钥J1的第2个test1，自然J1是这两个集合的交的元素。
- 如果拥有一系列三元组，就可确定J1。最直接的方法是：建立一个有64个计数器的矩阵，记录工作密钥J1的64种可能情况。每计算一次test1，如果某些元素(6bit)在test1中，则这些元素所对应的计数器加1，否则不变。给定t个三元组(E'1,E1*,b1)，希望能找到唯一的计数器，使得其值为t，则该计数器所对应的6bit就是工作密钥J1。

- 对一个r轮迭代密码的差分分析攻击基本步骤是：
- (1)寻找第r-1轮差分(a,b)使概率
- P(b=ΔX(r-1)|a=ΔB)的值尽可能为最大。
- (2)随机选择明文B，求另一明文B'，使得B与B'的差分为a。
- (3)在密钥k下对B和B'进行加密得到X(r)和X'(r)，
- 并求出能使ΔX(r-1)=b的所有可能的第r轮密钥$k^{(r)}$，
- 并对各子密钥$k_i^{(r)}$计数，
- 若选定的ΔB=a，明文对(B,B')在$k_i^{(r)}$下产生的(X,X')满足ΔX(r-1)=b，就将相应的$k_i^{(r)}$的计数加1。
- (4)重复(2)、(3)步，直到计数的某个或几个字符串的值，明显大于其他字符串的计数值，这一字符串或几个子密钥就可作为对实际子密钥k_r的分析结果。

- 在差分分析攻击时，所有子密钥是固定的(在给定未知密钥k下)，而明文则是随机选择的。
- 在计算差分概率时，明文和所有子密钥是独立且是均匀随机选择的。
- 通过差分分析，逐步推算出各轮实际所用的子密钥k_i。
- 有结论表明，若(r-1)轮差分(a,b)的差分概率

$$P(\Delta X(r-1)=b \mid \Delta X=a) \quad >> \frac{1}{2^n-1}$$

则这样的r轮迭代分组密码是经受不了差分分析攻击的

- 差分分析攻击计算复杂性主要与所需明文对(B,B')数目有关，即主要取决于数据复杂性。对于分组为n-bit的r轮迭代，差分分析攻击的数据复杂性为：

$$C_d(r) \geq \frac{2}{P_{\max}^{(r-1)} - \frac{1}{2^n - 1}}$$

$$P_{\max}^{(r-1)} = \max_a \max_b P(\Delta X(r-1) = b \mid \Delta B = a)$$

如果一个 r 轮迭代分组密码的 $P_{\max}^{(r-1)} \leq \frac{3}{2^n}$，则认为该密码算法可以抗差分分析攻击。

3.4.3线性攻击

- 线性攻击就是用最佳线性函数逼近S盒输出的非零线性组合。
- 对于已知明文m、密文y和特定密钥k，寻找线性表达式$(a \bullet m) \oplus (b \bullet y) = (d \bullet k)$，其中(a,b,d)为攻击参数。
- 对每一S盒的输入和输出之间构造统计线性路径，并最终扩展到整个算法。

- 定义3.3：对于布尔函数$h:\Sigma^n \rightarrow \Sigma$，如果存在$a_i \in \{0,1\}$(这里i=1,2,…n)，使得$h(s)=a_1s_1 \oplus a_2s_2 \oplus \ldots \oplus a_ns_n$，这里$s=(s_1,s_2,\ldots,s_n) \in \Sigma^n$，则称该布尔函数为线性的。所有n个变量的布尔线性函数全体构成的集合记为
- $L_n=\{h: \Sigma^n \rightarrow \Sigma | h=a_1s_1 \oplus a_2s_2 \oplus \ldots \oplus a_ns_n\}$。

- 如果对于某些$h(s)\in Ln$，布尔函数$h:\Sigma^n\rightarrow\Sigma$满足$f(s)=h(s)$或者$f(s)=h(s)\oplus 1$，则称该布尔函数为仿射函数。所有n个变量的仿射布尔函数的集合记为：
- $A_n=L_n\cup\{h\oplus 1|h\in L_n\}=L_n\cup\ \overline{L}_n$
- 两个布尔函数f,g：$\Sigma^n\rightarrow\Sigma$之间的汉明距离d(f,g)是向量$(f(\alpha_0)\oplus g(\alpha_0),f(\alpha_1)\oplus g(\alpha_1),\ldots,f(\alpha_{2n-1})\oplus g(\alpha_{2n-1}))$中1的个数，即f和g不一致的个数。这里$\alpha_0=(0,0,\ldots 0)$，$\alpha_1=(0,0,\ldots 1),\ldots\alpha_{2n-1}=(1,1,\ldots 1)$。

- 定义3.4：布尔函数$f:\Sigma^n\rightarrow\Sigma$的非线性度$N(f)$定义为：
- $N(f)=\min_{h\in An}d(h,f)$是函数f和它的最佳线性逼近之间的最小距离。
- 现代加密算法的密码强度主要依靠近似选择加密轮的密码结构。总可以将这些结构看作布尔函数的集合，即结束每轮输入输出位的S盒。更准确地说，一个$(n\times m)$的S盒$S:\Sigma^n\rightarrow\Sigma^m$是m个函数$f_i:\Sigma^n\rightarrow\Sigma$，$i=1,2,\ldots,m$的集合，函数$f_i$有n个变量$s=(s_1,s_2,\ldots,s_n)$，且
- $S(s)=(f_1(s), f_2(s),\ldots, f_m(s))$，

- 定义3.5：一个(n×m)的S盒S(f_1, f_2,…, f_m)的非线性度定义为：
- N(S)= $\min_{w=(w_1,\ldots,w_m)\in\Sigma^m;v\in\Sigma} N(w_1 f_1 \oplus \cdots \oplus w_m f_m \oplus v)$
- 是函数f和它的最佳线性逼近之间的最小距离。

- 在DES算法中，有8个S盒S_i：$\Sigma^6 \rightarrow \Sigma^4 (i=1,\ldots 8)$。每个S盒可以作为4个布尔函数的集合来处理。对于某个的S盒，可以建立如下距离表：该表的行由线性函数$l \in L_6$来索引，共有$2^6=64$种可能的线性函数。行的下标是表示线性函数的16进制数。因此下标31=110001 对应的线性函数是$l(s)=s_6 \oplus s_5 \oplus s_1 (s=(s_6,s_5,s_4,s_3,s_2,s_1))$。表的列由S盒输出的线性组合来索引。因而，如果S盒函数$S=(f_4,f_3,f_2,f_1)$，则其线性组合$f=(\alpha_4 f_4 \oplus \alpha_3 f_3 \oplus \alpha_2 f_2 \oplus \alpha_1 f_1)$，其中$\alpha_4\alpha_3\alpha_2\alpha_1$是16进制表示的列的下标值。例如，下标9=1001对应的输出线性组合$f_4 \oplus f_1$,，该表称为S盒的线性简档。
- 查找对应列f的最大和最小项可以找出输出的线性组合的非线性度。

- 可以根据条件列出最佳线性逼近，从而导出密钥。
- 对每一S盒的输入和输出之间构造统计线性路径，并最终扩展到整个算法。
- 如果迭代函数具有高阶非线性性，则这种攻击较难奏效。
- 一种更有效的方法是把差分攻击与线性攻击结合起来。

3.5 双钥密码体制

- 1976年，Diffe和Hellman在一篇名叫“New direction in cryptography”一文中提出了双钥密码的新型密码体制，把加密密钥和解密密钥分割开来，且无法由一个推出另一个，使得不仅可以公开密码算法，而且加密密钥也可公开，开创了公开密钥密码体制，为密码学研究提出了新的方向。

- **3.5.1双钥密码概述**
- 在公钥密码体制中，加密密钥PK和解密密钥SK分离，
- 前者PK可以象发公告牌和号码簿一样公布，供发方加密原始信息用；
- 后者SK则由接收方用户私人秘密保存，作为收到密文后的解密用。
- 密码体制则采用一个适当的以PK为参数的加密函数E_{PK}和以SK为参数的解密函数D_{SK}。

- E_{PK}和D_{SK}应满足下述要求：
- (1)对明文m的加密变换可以表示为$E_{PK}(m)=c$，对密文c的解密变换可以表示为$D_{SK}(c)=D_{SK}(E_{PK}(m))=m$，因此加密变换$E_{PK}(m)$和解密变换信息$D_{SK}(c)$必须是一对互逆变换。
- (2)加密变换函数E_{PK}必须是一个性能很好的陷门单向函数。即已知PK、m求c是十分容易的，而已知c和PK则在实际上是无法计算出m的。但若掌握了解密密钥SK则可容易地由密文c求出明文m来。

- (3)易于产生加密——解密密钥对PK和SK，但从m、c 和PK计算出Sk在计算上是不可行的。
- (4)*交换加密和解密的先后秩序，其效果不变。即PK和SK符合下述意义上的交换律：
- $D_{SK}(E_{PK}(m))=E_{PK}(D_{SK}(m))=m$
- 在公钥密码体制中，每个用户都把自己的加密密钥公开，如果E_{PK}和D_{SK}满足条件(1)—(3)，就可实现可靠的密码通信：

- 如果所选用的E_{PK}和D_{SK}还满足条件(4)，则该体制不仅能提供保密通信，而且还能成功地提供数字签名。其原理简述如下：
- 设用户A需要发送附有他签名的机密信息m(明文)给用户B。
- A先用自己的秘密密钥SK_A按解密函数D_{SKA}对m进行变换：$D_{SKA}(m)=s$。这一变换称为签名变换，并称s为签名文本。
- 只有A掌握SK_A，这就相当于A的印鉴。
- A的公钥任何人都可以用，因此任何人都可由签名文本s用公式$E_{PKA}(s)$求得m，
- 为使签名文本s及明文m保密，用户A再利用收方B的公钥PK_B按加密函数E_{PKB}对签名文本s进行加密变换：$E_{PKB}(s)=c$。
- c是与明文m和签名文本s相对应的密文。

- 密文c由A送到收方B。B在接收密文c后，先用自己的秘密密钥PK_B对密文c进行解密变换，
  $D_{SKB}(c)=D_{SKB}(E_{PKA}(s))=s$然后再用发方A的公开密钥$PK_A$对s进行加密变换就可恢复明文m：
  $E_{PKA}(s)=E_{PKA}(D_{PKB}(c))=m$。称上述变换为译名变换。
- 收方B同时保存了发方A送来的明文和签名文本对(m,s)，它完全等效于签字的书面文件。
- 因为收方B不知道SK_A，故无法剪接、篡改和伪造(m,s)，故达到了对收方的证实。
- 发方也无法抵赖(m,s)是由他发送的
- 因为收方B是利用A的公钥PK_A从签名文本s中得到明文m的，而与PK_A相对应的解密密钥SK_A只有用户A才知道，这又达到了对发方的证实，
- 成功地解决了发、收方就传送的信息内容可能引起的争端。

- 单向函数
- 陷门单向函数。
- 对于一个双射函数$f:A\to B$，如果对所有$x\in A$都容易计算$f(x)$，而几乎对所有的$x\in A$，要想由$f(x)$求出x则是极为困难的，就称$f(x)$为单向函数。
- “极为困难”是相对于现有计算资源和算法而言，随着计算技术的发展，一些原来符合要求的函数可能变为容易计算。

- 一个函数要能作为公钥密码算法，不仅要具有单向性，而且还要保证合法接收者能容易由$f(x)$求出x，即$f(x)$应是陷门单向函数。
- 陷门单向函数实际上并不是真正的单向函数，因为单向函数要求的是在任何情况下求逆是困难的。
- 设A_z,B_z分别是集合A和B的子集，它们的构成与陷门信息集Z中的元素z有关。

- 对于单向函数$f:A\to B$的限制$f_z:A_z\to B_z$，如果满足如下条件，就可认为f_z是陷门单向函数。
- (1)对所有$z\in Z$，容易找到一对算法E_x和D_x，使得对任意$x\in A_z$，容易计算f_z和它的逆，即$f_z(x)=E_z(x)$，$D_z(f_z(x))=x$。
- (2)对所有$z\in Z$，容易找到算法F_z，对所有$x\in A$都能方便地检验x是否为A_z的元素，并且容易实现明文集到A_z的映射。
- (3)几乎对所有$z\in Z$，当只给定E_z和F_z时，几乎对所有$x\in A_z$，很难从$y=E_z(x)$求出x。

- 一个公钥系统，实际上就是所有用户共同选定一个陷门单向函数、加密运算E以及函数F。
- 用户I从陷门信息集Z中选定z_I，并公开E_{z_I}和F_{z_I}。
- 任何用户要向用户发送信息x，可用函数F_{z_I}检验x是否在允许范围内，然后将$y=E_{z_I}(x)$送给用户I。

- 现在所用的一些密码算法主要利用如下可能的单向函数进行构造的。
- 离散对数:给定一个大素数p，满足p-1包含有大素因子q，由此构造p-1阶循环群，其生成元为整数a。

由x求$y=a^x \bmod p$是容易的，只要$\lfloor \lg 2x \rfloor -1$次乘法。

由y,a,p求$x=\log_a y$就是离散对数问题，其最快算法为

$L(p)=O(e^{1+o(1)\sqrt{\ln p \ln \ln p}})$，当 $p=512$ 时，$L(p)=2^{256}=10^{77}$。

在一般的n阶循环群G上定义离散对数问题就称为广义离散对数问题

- 整数分解问题：判断一个大奇数是否为素数的有效算法，大约需要$(\log n)^4$。由两个大素数 p 和 q，求 n=p•q 只需一次乘法，而由 n 去分解得到 p 和 q 则是数论专家的攻关对象。一般认为，当 n=p 时，解离散对数问题要更难些。

背包问题、二次剩余问题和模n的平方根问题。

- 一个密钥分配系统
- 第一个具有公钥思想的是Diffe和Hellman建立的密钥分配系统。
- 该系统是基于求离散对数难解问题。
- 设p为一个大素数，a为p-1阶循环群的本原元。
- 每个用户I从集合{1,2,…,p-1}中随机选择数x_I作为自己的秘密密钥，并计算
- $y_I \equiv a^{x_I} \bmod p$
- 把y_I作为公开密钥予以公开。

- 通信双方A和B可以利用他们的公钥产生通信密钥，其方法是双方分别作计算
- k_{AB}可作为A和B双方通信的密钥。
- 任何第三方因不知道x_A和x_B就不能求出k_{AB}。
- 由y_A求x_A就是在GF(p)上求离散对数问题：$x_A=\log_a y_A \ \mathrm{mod} p$，这在计算上是难的
- 但存在中间人攻击
- 认证技术，如何安全应用上述协议

- CDH问题(计算Diffe-Hellman问题)
- 输入：在有限域F_q中，g为F_q^*的生成元，$g^a, g^b \in F_q^*$，这里$0<a,b<q$
- 输出： g^{ab}.
- 如果CDH问题是容易的，即g^{ab}是容易的
- CDH假设(计算Diffe-Hellman问题假设)
- 不存在计算Diffe-Hellman问题的多项式时间算法

- CDH问题基于离散对数问题(DLP)的困难性
- DLP(离散对数问题)：

  输入：在有限域F_q中，g为F_q^*的生成元，在F_q^*中均匀随机选取h

  输出： 唯一的整数$a<q$，使得$h=g^a$.
- DL假设(离散对数假设)

  不存在解决离散对数的多项式时间算法
- 考虑两个问题之间的关系：
- 如果存在求解离散对数的多项式时间算法。则由g^a,g^b可求得a,b,故可求得g^{ab}.

- 若DLP能解，则CDH可解，因此如果DL假设不成立，则CDH假设不成立。
- 即CDH问题比DL问题弱， 即CDH假设是比DL假设更强的假设。
- 而在安全性研究中，假设应该尽可能的弱
- 如果CDH是容易的，那么DL是否也容易？
- 这是一个研究的问题，一般猜测这2个问题可能是等价的。

3.5.2 RSA密码体制

- RSA加密算法是由美国麻省理工学院(MIT)的研究小组于1978年提出的，
- 该体制的名称是用了三位研究者Rivest,Shamir和Adleman的第一个字母拼合而成。
- 该体制的理论基础是数论中的下述论断：要求得两个大素数的乘积是容易的，但要分解一个合数为两个大素数的乘积，则在计算上几乎是不可能的。

1. RSA的基本方案

- 在RSA 体制中，首先为每个用户产生密钥对：
- 由用户I随机选择两个大素数p_I和q_I，并求$n_I=p_Iq_I$，以及欧拉函数$\varphi(n_I)=(p_I-1)(q_I-1)$
- (欧拉函数$\varphi(n_I)$表示在$\{1,2,\ldots,n_I-1\}$中与n_I 互素的数的个数)，
- 再随机选择整数d_I，使得满足$(d_I,\varphi(n_I))=1$，
- 最后，从p_I,q_I,d_I按下式求出整数e_I：
- $e_Id_I\equiv1 \bmod\varphi(n_I)$
- 随后把n_I和e_I公开，d_I则由用户保存，而其他信息也应保密或毁掉。
- 由于其他人不知道$\varphi(n_I)$以及p_I和q_I，因此无法由n_I和e_I求出d_I。

- 当用户A要向用户B发送信息m时，就利用B的公钥n_B和e_B加密明文m：$c=m^{e_B} \bmod n_B$
- 用户B收到c后，就用自己的解密密钥d_B解密密文c：$m=c^{d_B} \bmod n_B$。下面证明该加解密过程是正确的。即证明解密运算所得到的结果的确是原来的明文。
- 证明：对任意$m<n_B$，
- 分两种情况讨论：
- (1)$(m,n_B)=1$，
- 欧拉定理:a,b互素，则$a^{\varphi(b)} \equiv 1 \bmod b$
- (2) $(m,n_B)>1$。因为$n_B=p_B \cdot q_B$，故(m,n_B)恰含p_B和q_B中的一个，不仿设是p_B。

- **2.RSA体制的安全性分析**
- $c=m^e \bmod n$
- **在理论上，RSA的安全性取决于模n分解的困难性。若n被分解成功，则RSA被攻破。目前最快的分解因子算法其时间复杂性为** $e^{(\sqrt{\ln n \ln\ln n})}$

并没有证明分解大整数就是NP问题，并不能排除存在尚未发现的多项式时间算法。
也没有证明对RSA的攻击的难度与分解n等价
因此对RSA的攻击的困难程度不比大数分解难

- 在实际使用时，应注意下列问题：
- (1)不要使用同一模n。这是因为在同一个模n下，若两个不同的公钥互素，则用任一密钥就可恢复明文。
- 例如对于两个不同的互素公钥e_1和e_2，使用同一个n。如果加密同一个信息x，
- $y_1=x^{e_1} \bmod n$, $y_2=x^{e_2} \bmod n$
- 攻击者由n,e_1,e_2,y_1,y_2容易求出x。
- 因为$(e_1,e_2)=1$，故存在r和s，使得$r\cdot e_1+s\cdot e_2=1$，取r为负数，则由Euclidean算法可计算

$y_1 = m^e \bmod n_1$、 $y_2 = m^e \bmod n_2$和$y_3 = m^e \bmod n_3$在模 $n_1 \cdot n_2 \cdot n_3$ 下的唯一解$y = m^e \bmod (n_1 n_2 n_3)$，然后求 $y^{1/e}$ 即得 m。

通常便用较小的e，但如果e取得太小，容易受

因此 e 应取得足够大，一般，e 可选 16 位素数，可兼顾快速加密和防止这种攻击。另外，还应使 e 在乘法循环群$Z^*_{\varphi(n)}$中的阶达到(p-1)(q-1)/2。顺便指出，d 也不能太小，应大于 $n^{1/4}$。

- **在一个系统中，规定n_1,n_2,n_3互素（否则可求出它们的公因子，从而降低安全性），**
- **利用中国剩余定理，可由y_1,y_2,y_3求出同余方程**
- **$y=m^e \bmod n$，然后求$y^{1/e}$**
- **这就是为何不能取e=3的原因**

- (3)注意不动点问题。
- 对于信息m(0≤m<n)，用RSA加密时，可能出现$m=m^e \bmod n$，导致信息泄露。
- 一般有(1+gcd(e-1,p-1))(1+gcd(d-1,q-1))个不动点
- 只要适当注意e,d的选择，就可使不动点个数很少而可忽略。
- (4)对于n=pq，p和q必须是强素数。如果一个素数满足如下条件，就称为强素数：
- 存在两个大素数q_1和q_2，使得$q_1|(p-1)$，$q_2|(p+1)$；存在四个大素数r_1,r_2，s_1和s_2，使得$r_1|(q_1-1)$，$s_1|(q_1+1)$, $r_2|(q_2-1)$，$s_2|(q_2+1)$。
- 费尔马定理$2^L=1 \bmod p$。

- (5) 要求p与q之差不能太小，否则可由$(p+q)^2=(p-q)^2+4pq\approx 4n$，
- 由此求出p+q，从而与$(p-q)\approx 0$一起估计出p和q的大致值。
- 当然p和q的位数则应大致相同，并且p和q的取值应足够大。
- (6)在RSA实际使用过程中，还要注意短明文加密的问题
- 例如如果知道明文是小于1000000的数字(如支付信息)，则已知密文，攻击者可以用不超过1000000次的尝试加密找出明文

- 随着计算技术的提高，因子分解速度也极大提高。
- 1994年，43个国家的600多人参加，用1600台机器同时产生820条指令数据，通过互联网，耗时8个月，成功分解了一个129位的整数。
- 1996年4月经过几个月的努力成功分解了一个130位的整数。
- 1999年2月一个140位的整数分解成功。
- 1999年8月一个155位的整数也分解成功。
- 512bit为154位整数
- 因此应取n为2048bit及以上

3. RSA的实现

- RSA的硬件实现，其最快速度也是DES的1/1000，512bit的RSA大约为1MB/s，软件实现的速度只有DES的软件实现的1/100，因此在速度上RSA是无法与对称密码体制相比的。
- 目前RSA体制主要用在密钥交换和认证
- 512bit的RSA软件实现可达20KB/s。
- 如果选e=65537时，运算速度可大大加快，这是因为65537的二进制表示中只有两个1，可极大地减少运算量。

- 作业：p110　6, 7