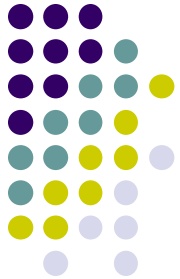


第2章
数据加密算法

第2章 内容概要

- 2.1 数据加密算法设计标准
- 2.2 数据加密标准DES
- 2.3 多重DES
- 2.4 高级加密标准AES
- 2.5 标准分租密码运算
- 2.6 流密码
- 2.7 密钥生成

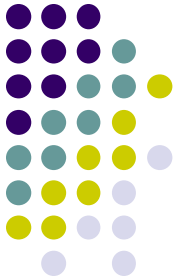
预备知识：

- 任何由一个固定符号集合构成的消息都可以用一个*二进制串*来表示 (0和1构成的序列)
- 二进制0和1称为*比特*
- 为了减少计算开销，加密算法应该只用那些便于实现的运算
- 对于一个二进制串X:
  - $|X|$表示X的长度, 数值上为X包含的bit个数
  - 若$|X| = l$, X 是个l-比特的二进制串
  - 令a 为一个二进制比特，k 为一个非负整数.
  - 用a^k 表示一个由a的k个拷贝组成的二进制串

$$a^k = \underbrace{aa \ldots a}_{k\ a's}$$

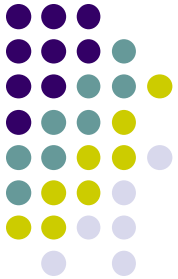
  - *用*XY 或$X||Y$*表示*X 和 Y 连接

密码是什么?



- 两种常见的网络安全方法
  - 基于密码的: 基于密码学算法和安全协议
  - 基于系统的: 非加密的方法
  - 两者结合形成一个标准的安全结构
- 加密
  - 使明文变换为不可理解的文本
  - 不可理解的文本可被复原成原始的明文
- 通常的加密方法使用密钥和密码算法
  - 传统加密 (也称为：对称加密): 加密和解密使用相同的密钥
  - 公钥加密 (也称为：非对称加密):加密和解密使用不同的密钥

例: 替换

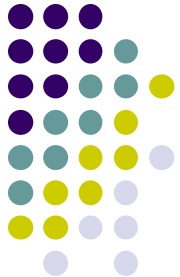


- 基于字符的一一映射; 例如：

  用*d 替换a*, *z 替换b*, *t 替换c*, 等

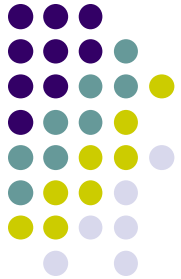
- 对于未经训练的人不可读，但这种方法具有基于语言的统计规律 (例如，字符频率)
- 在英语里，字母 “*e*” 出现的频率最高
- 在不可理解的文本中出现频率最高的字母很有可能代表“*e*”
- 可以用这个方法找出其它字母之间的替换关系

ASCII 码



- 7-位 二进制串
  - 从第一个到第32个是控制编码
  - 32 到126 包含了大小写英文字母，数字，标点符号和算术运算符
- 我们通常在前面添加一位，使得每个字符构成一个字节
  - 增加的这一位可以用来整体代表额外的128个字符，或者作为奇偶校验用于差错检测
- 因此任意ASC II码的二进制串都是可以被8整除的
- 对于其它编码集的长度，例如Unicode，可被16整除
- 不失一般性，假定任意明文的长度其二进制串都可被8整除

XOR 加密



- ⊕ 或XOR表示异或运算, 是用于加密的最简单的二进制运算
- XOR 加密: 将一个串分为等长的块，然后用等长的密钥加密每个块
- 例如, 我们用一个8bit的块（一个字节）, 加密两个字符（两个字节）的串M，我们用一个8-bit的密钥 (例如: 1100 1010) 加密M 两次:

$$\begin{array}{lll} M: & & 1111\ 1111\ 0000\ 0000 \\ K: & \oplus & 1100\ 1010\ 1100\ 1010 \\ C: & & 0011\ 0101\ 1100\ 1010 \end{array}$$

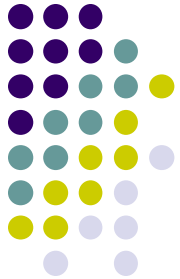
我们可以用相同的密钥解密; 即, 我们将C 和 K进行异或运算就可得到M:

$$\begin{array}{lll} C: & & 0011\ 0101\ 1100\ 1010 \\ K: & \oplus & 1100\ 1010\ 1100\ 1010 \\ M: & & 1111\ 1111\ 0000\ 0000 \end{array}$$

- 上述运算实现起来非常简单
- 但它不安全，知道 (M_i, C_i)中的任意一个将泄漏 K:

$$M_i \oplus C_i = M_i \oplus (M_i \oplus K) = K$$

数据加密标准



- 异或加密是安全的，只要一个密码只用一次，但这在实践中是不可行的
- 如果不公开加密算法呢？
- 为了研究加密算法的安全性，我们假定除了加密密钥以外的一切都是公开的，而且密钥可以重用
- 好的加密算法必须满足下列标准：

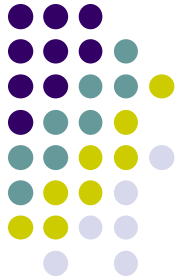
  -高效

  -抵御统计分析攻击

  -抵御暴力攻击

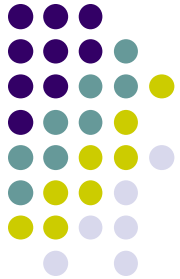
  -抵御数学分析攻击

高效



- 用于算法的运算必须在软硬件上便于实现
- 算法运行时应该适度消耗系统资源
- 时空复杂度必须保持在输入规模的小的常数倍数以内
- 基本运算:
  - 异或
  - 置换:一对一映射
  - 替换:多对一映射
  - 循环位移: 置换的一种特殊形式
  - 有限域的运算

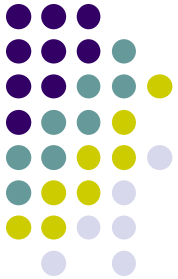
抵御统计分析



- 分析*C*中字符出现的频率, 可以得到其在明文*M*中所对应的原始字符
- *扩散*和*混淆*是平滑统计结构的标准方法
  - *扩散*: *C* 中的每个bit应该依赖于*M*中的多个bit，且尽可能的均匀产生扩散性可以对明文段执行某些特定的运算，如替换运算，并对新产生的二元字符串如法重复数次
  - *混淆*: *C*中的每个bit应该依赖于秘密钥*K*的多个bit，且尽可能的均匀

    混淆性可通过生成K的子密钥并且在不同的轮次使用不同的子密钥和获得

抵御暴力攻击



- 密码算法的长度依赖于运算和密钥的长度
- 假定密钥长度为l-bit, 则密钥空间为2^l
- 若窃听者Eve获得一个密文C，而且知道其机密算法, 她可以一次尝试一个密钥来破解密文
- 因此，暴力攻击的时间复杂度是$O(2^l)$
- 基于当前技术，普遍认为 $l = 128$即足够
- 暴力攻击的时间复杂度通常用于衡量其它密码分析攻击的基准:若一个攻击算法的时间复杂度大大地小于 2^l , 则改攻击被认为是有效的

抵御其它攻击

- 其它常见攻击：*选择明文攻击*和*数学攻击*
- *选择明文攻击*:
  - 获得一个特定的明文M及其密文C
  - 用二元组(M, C) 来找出密钥
  - 例如: 异或加密

    若Eve 知道(M, C) 则她可轻松获取K :

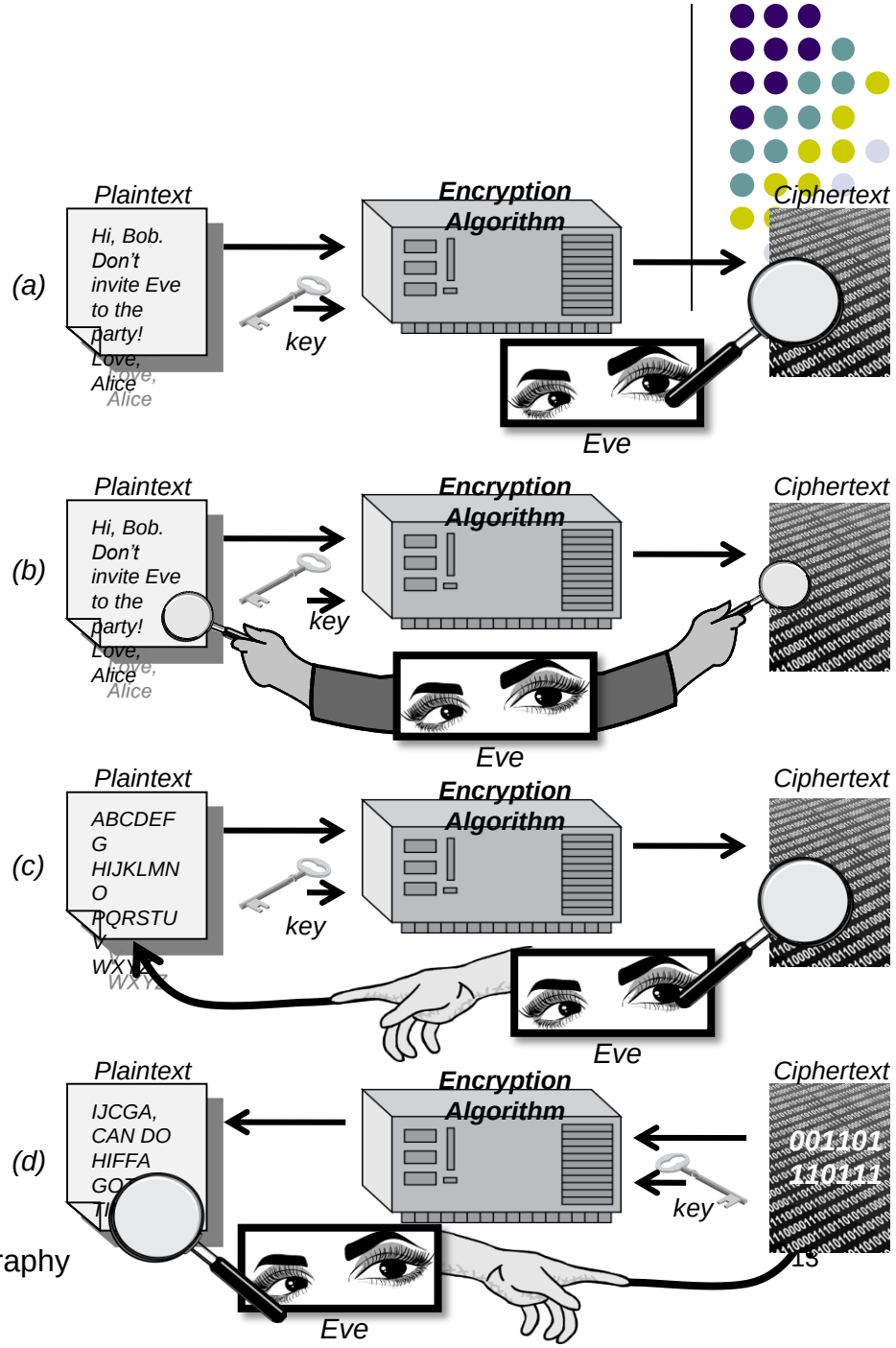
    $$C = (M \oplus K)$$

    $$M \oplus C = M \oplus (M \oplus K)$$

    $$M \oplus C = K$$
- *数学攻击*:
  - 用数学方法破译加密消息
    - 差分分析, 线性分析, 代数分析.
    - 需要很深的数学知识

Attacks

- Attacker may have
  a) collection of ciphertexts (ciphertext only attack)
  b) collection of plaintext/ciphertext pairs (known plaintext attack)
  c) collection of plaintext/ciphertext pairs for plaintexts selected by the attacker (chosen plaintext attack)
  d) collection of plaintext/ciphertext pairs for ciphertexts selected by the attacker (chosen ciphertext attack)



密码分析学

穷举攻击：又称作蛮力攻击，是指密码分析者用试遍所有密钥的方法来破译密码对可能的密钥或明文的穷举

统计分析攻击 ：指密码分析者通过分析密文和明文的统计规律来破译密码。

数学分析攻击：指密码分析者针对加密算法的数学依据，通过数学求解的方法来破译密码。

根据密码分析者掌握明、密文的程度密码分析可分类为：

唯密文攻击：仅根据密文进行的密码攻击；

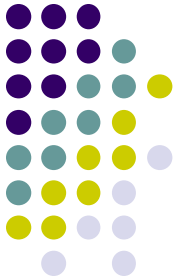
已知明文攻击：根据一些相应的明、密文对进行的密码攻击。

选择明文攻击：可以选择一些明文，并获取相应的密文，这是密码分析者最理想的情形。例如，在公钥体制中。

选择密文攻击：密码分析者能选择不同的被加密的密文，并可得到对应的解密的明文，密码分析者的任务是推出密钥。

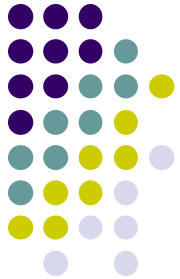
选择密钥攻击：这种攻击并不表示密码分析者能够选择密钥，它只表示密码分析者具有不同密钥之间关系的有关知识。

抵御旁道攻击



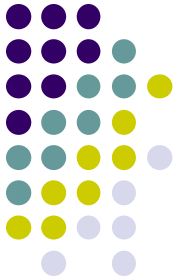
- 密码算法的实现必须能够抵御侧信道攻击*side channel attacks* (SCA)
- SCA 探测实现环境的漏洞
  - 计时攻击: 攻击者分析特定运算的计算时间
  - 如果当密钥具有不同的bit值时，特定运算的运算时间变化，则计时攻击有效
- 抵御计时攻击:
  - 通过在执行指令时假如冗余运算来平滑计算时间的区别

第2章 内容概要



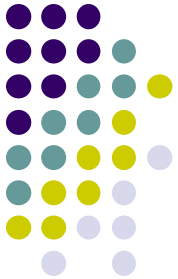
- 2.1 数据加密算法设计标准
- 2.2 数据加密标准DES
- 2.3 多重DES
- 2.4 高级加密标准AES
- 2.5 标准分组密码运算
- 2.6 流密码
- 2.7 密钥生成

数据加密标准(DES)



- 美国国家标准化局(NBS)于1977年公布
- 是Feistel 密码体系(FCS，由Horst Feistel发明)的一个具体的实现
- 对称的加密和解密结构
- 使用四个基本运算: 异或, 置换, 替换和循环位移
- 从70年代中期到2000年早期被广泛使用
- 逐步被AES 和其他更好的算法所取代

Feistel 密码体系(FCS)



- 将M 分成2l-bits长的分组 (必要时需要对最后一个分组填充)
- 仅使用异或和替换运算
- 从密钥K中生成n个固定长度子密钥: $K_1,\ldots,K_n$
- 将一个2l-bit 的输入分组分为两个部分: L_0 和 R_0, 长度均为l (分别表示分组的前缀和后缀)
- 对一个l-bit的输入串执行一个替换函数F 和一个子密钥得到一个l-bit的输出
- 加密和解密均执行n轮相同的序列的运算

Feistel密码结构

- Feistel结构的具体实现依赖于以下参数和特征。
  - 分组长度：分组长度越长意味着安全性越高（其他数据不变），但是会降低加密和解密的速度。这种安全性的增加来自更好的扩散性。传统上，64位的分组长度比较合理，在分组密码设计里很常用。然而，高级加密标准使用的是128位的分组长度。
  - 密钥长度：密钥长度较长同样意味着安全性较高，但会降低加密和解密的速度。这种安全性的增加来自更好的抗穷尽攻击能力和更好的混淆性。现在一般认为64位的密钥还不够，通常使用的密钥长度是128位。
  - 迭代轮数：Feistel密码的本质在于单轮不能提供足够的安全性，而多轮加密可取得很高的安全性。迭代轮数的典型值是16。
  - 子密钥产生算法：子密钥产生越复杂，密码分析攻击就越困难。
  - 轮函数：轮函数越复杂，抗攻击能力就越强。

FCS 加密和解密

FCS 加密

- 令$M = L_0R_0$; 在第i轮执行下列运算, $i = 1, \ldots, n$:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

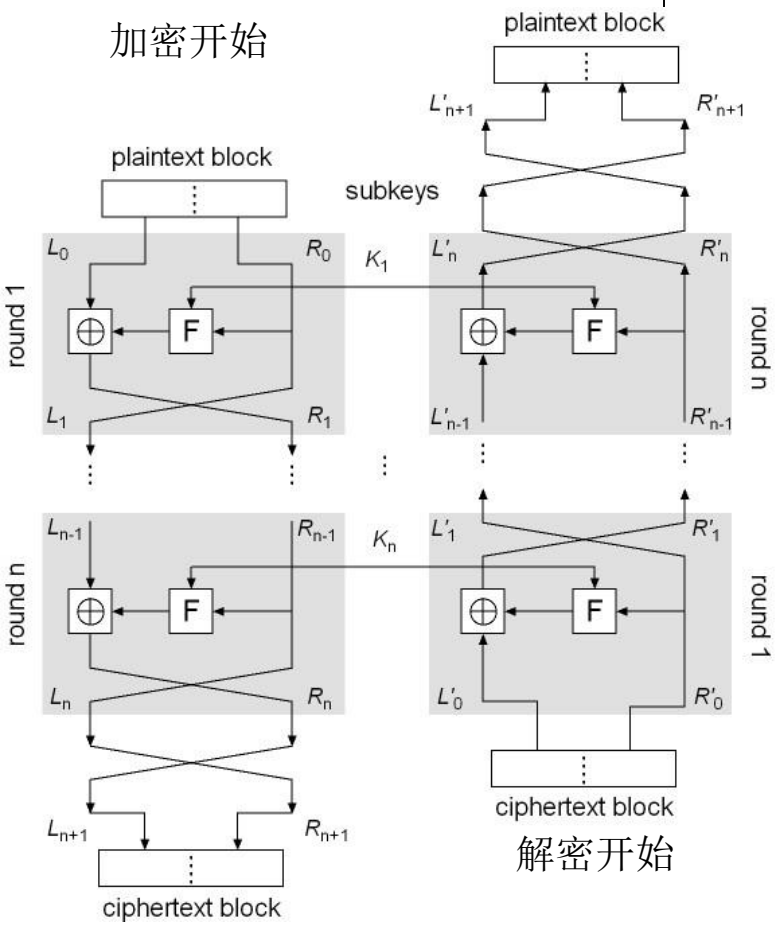
- 令$L_{n+1} = R_n$, $R_{n+1} = L_n$ 且$C = L_{n+1}R_{n+1}$

FCS 解密

- 与加密对称, 以相反的顺序使用子密钥
- 将C 重写为$C = L'_0R'_0$
- 在第i轮执行下列运算 ($i = 1, \ldots, n$):

$$L'_i = R'_{i-1}$$
$$R'_i = L'_{i-1} \oplus F(R'_{i-1}, K'_{n-i+1})$$

- 另$L'_{n+1} = R'_n$, $R'_{n+1} = L'_n$
- 我们将得到$M = L'_{n+1}R'_{n+1}$



FCS解密的证明

- 欲证明密文$C = L_{n+1}R_{n+1} = L'_0R'_0$ 通过FCS解密算法可被还原为明文$M = L_0R_0$, 通过引入下列等式来证明:

  (1) $L'_i = R_{n-i}$ (2) $R'_i = L_{n-i}$

- 基础: $L_0' = L_{n+1} = R_n$, $R_0' = R_{n+1} = L_n$; (1) 和(2) 满足
- 假设: 假定当 $i \le n$时:

  $L_{i-1}' = R_{n-(i-1)}$ $R_{i-1}' = L_{n-(i-1)}$

- 归纳：：

  $L'_i = R'_{i-1}$ (基于解密算法可得) = L_{n-i+1} (基于假设) = R_{n-i} (基于加密算法)

  因此(1)满足

- $R'_i = L'_{i-1} \oplus F(R'_{i-1}, K_{n-i+1})$

  $= R_{n-(i+1)} \oplus F(L_{n-(i+1)}, K_{n-i+1})$

  $= [L_{n-i} \oplus F(R_{n-i}, K_{n-i+1})] \oplus F(R_{n-i}, K_{n-i+1})$

  $= L_{n-i}$

  因此(2) 也满足

DES算法原理

- 1973年5月13日美国国家标准局（NBS），即现在的国家标准和技术协会（NIST）在认识到建立数据保护标准既明显又急迫需要的情况下，开始征求国家密码标准方案。这一举措最终导致了数据加密标准DES的出现。NBS提出密码算法标准包括：
  - 算法必须是安全的；
  - 算法必须是公开的；
  - 能够经济、有效地用硬件实现；
  - 能够得到批准；
  - 可出口。

DES算法原理

- 在众多算法中，IBM公司Tuchman Meyer提出的算法Lucifer被选中。1975年3月17日，NBS 公布了IBM 公司提供的密码算法，以标准建议的形式在全国范围内征求意见。1977年7月15日，NBS 宣布接受这个建议，即DES（Data Encryption Standard）正式颁布，DES被NBS确定为联邦信息处理标准（FIPS PUB 46），供商业界和非国防性政府部门使用。几十年来，DES得到了广泛应用，尤其在金融领域发挥了重要的作用。

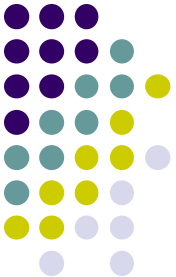
DES算法原理

DES采用了64位的分组长度和56位的密钥长度。除了初始置换和逆初始置换，DES结构与Feistel结构完全相同。DES首先把明文分成以64 bit为单位的块m，对于每个m，　执行如下操作:

$$DES(m)=IP^{-1} \cdot T_{16} \cdot T_{15} \cdot \ldots\ldots T_2 \cdot T_1 \cdot IP(m)$$

- 初始置换，　IP
- 16轮迭代，T_i ，　　i=1,2,...16
- 逆置换，IP^{-1}

DES 子密钥生成



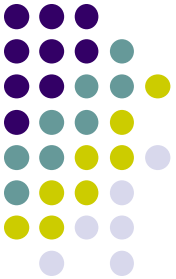
- DES的分组大小为64 bits且加密密钥是 56 bits（由一个64-bit的串 $K = k_1\ k_2\ \ldots\ k_{64}$表示）
- DES用16个子密钥迭代16轮
- 子密钥生成:
  1. 从K中移除第$8i$-th 个bit ($i = 1, 2, \ldots, 8$)
  2. 对K的剩余56个bits中执行一个初始置换,记为$\mathrm{IP}_{\mathrm{key}}(K)$
  3. 将这个56-bit密钥分为两部分: U_0V_0, 均为28 bits
  4. 对U_0 和V_0 执行一个规定次数的左循环移位，得到U_iV_i:

$$U_i = \mathrm{LS}_{z(i)}\ (U_{i-1}), \qquad V_i = \mathrm{LS}_{z(i)}\ (V_{i-1})$$

  5. 用一个规定的压缩置换置换得到的U_iV_i, 生成一个48-bit的串作为子密钥, 记为 K_i

$$K_i = \mathrm{P}_{\mathrm{key}}\ (U_i\ V_i)$$

DES 替换盒



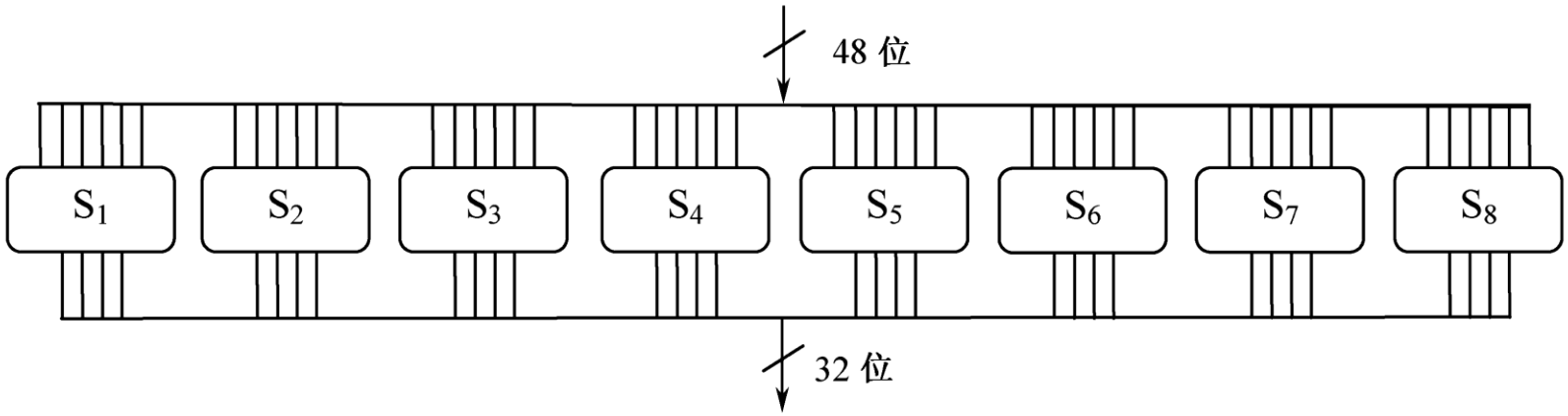
- DES的替换函数F定义如下:

$$F(R_{i-1}, K_i) = P(S(EP(R_{i-1}) \oplus K_i)), i = 1,\ldots,16$$

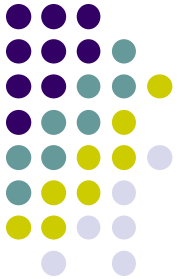
- 首先, 用$EP(R_i)$ 置换R_i 得到一个48-bit的串x
- 接着, 用48-bit子密钥K_i 异或x 得到一个48-bit串y
- 函数S 将y 转换为一个32-bits的串z, 利用8个4x16的特殊矩阵,即S-boxes
  - S-box中的每一项是一个4-bit串
  - 将y 分成8个分组,每个6-bits
  - 用第i^{th}个分组$b_1b_2b_3b_4b_5b_6$的第i^{th} 个矩阵
  - 令b_1b_6 为行号, $b_2b_3b_4b_5$ 为列号, 且返回对应的项
  - 每6-bit分组变换成一个4-bit串, 最终构成一个32-bit串z
- 最后, 用P置换z得到DES的F函数的结果
- 该结果, 与L_{i-1}异或, 就是R_i

S盒的设计

- S盒代换是DES算法中最重要的部分，也是最关键的步骤，因为其他的运算都是线性的，易于分析，只有S盒代换是非线性的，它比DES中任何一步都提供了更好的安全性。下图解释了S盒在函数F中的作用。代换函数由8个S盒组成，每个S盒输入6位，输出4位。



DES 加密步骤



- 重写IP(M) = L_0R_0, 这里$|L_0| = |R_0| =32$
- 对$i = 1, 2, \ldots, 16$, 依次执行下列运算:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

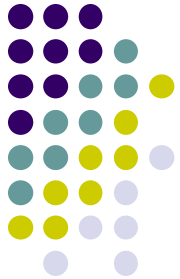
- 令$C = \text{IP}^{-1}(R_{16}L_{16})$

初始置换表IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

逆初始置换表

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25



- 这两个置换是互逆的。对于64位的二进制分组数据M，经过置换X=IP(M)后，再对它进行逆置换IP^{-1}，得到$Y = IP^{-1}(IP(M))$，就会恢复出M。例如，经过IP置换后，M的第1位被置换到第40位，再经过逆置换IP^{-1}，第40位又回到第1位的位置。

DES的半公开性

- DES的核心是S盒，除此之外的计算是属线性的。S盒作为该密码体制的非线性组件对安全性至关重要。
- S盒的设计准则：

1. S盒不是它输入变量的线性函数
2. 改变S盒的一个输入位至少要引起两位的输出改变
3. 对任何一个S盒，如果固定一个输入比特，其它输入变化时，输出数字中0和1的总数近于相等。

DES的半公开性

- 在DES里的所有计算中，除S盒之外全是线性的，作为唯一的非线性部件，S盒对DES的安全性至关重要，是DES的核心。而S盒的设计标准，实际上包括整个算法的设计标准是不公开的，在DES刚提出时，就有人怀疑S盒里隐藏了“陷门”，使得美国国家安全局能够轻易地解密消息。后来表明DES里的S盒被设计成能够防止某些类型的攻击。在20世纪90年代初，Biham与Shamir发现差分密码分析时，美国国家安全局承认某些未公布的S盒设计准则正是为了使得差分密码分析变得不可行。事实上，差分密码分析在DES最初被研发时就已被IBM的研究者所知，但这种方法却被保密了近30年，直到Biham与Slhalnir又独立地发现了这种攻击。

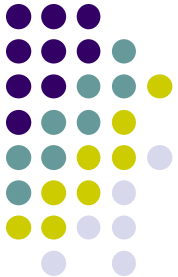
DES的半公开性

- 多年来人们的确发现了S盒的许多规律和一些缺点，尽管如此，还没有人发现S盒存在致命的弱点。
- 1998年美国政府终于决定不再继续延用DES作为联邦加密标准，DES将退出加密标准的舞台，寻找DES的替代者已刻不容缓，新的标准AES粉墨登场。

DES的安全性

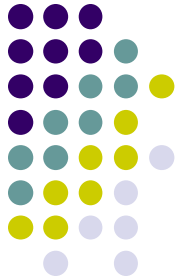
- 随着计算机速度的提高和硬件价格的下降，最终会导致DES毫无价值。幸运的是，有大量DES的代换算法，最重要的有高级加密标准（AES）和3DES。

DES够好吗？



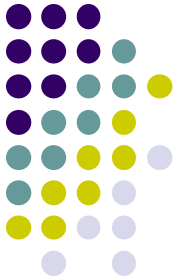
- DES的安全强度
  - 轮次数
  - 加密密钥的长度
  - 替换函数的构成
- DES 被使用到1990’s.
- 人们开始进行以攻破DES为目的的DES挑战赛
- 仅使用56-bit密钥即密钥空间为$2^{56} \sim 7.2 \times 10^{16}$
- 以目前技术暴力破解已经可行
  - 1997年Internet线上破解耗时数月
  - 1998年专门的破解机(美国电子前沿协会)只耗时数天
  - 1999年两者结合只耗时 22个小时
- 我们怎么办？重新开始?
- 新标准开始着手指定
- 同时，我们能利用现有的DES的软硬件对DES增强吗？

第2章 内容概要



- 2.1 数据加密算法设计标准
- 2.2 数据加密标准DES
- 2.3 多重DES
- 2.4 高级加密标准AES
- 2.5 标准分租密码运算
- 2.6 流密码
- 2.7 密钥生成

3DES/2, 2DES 和3DES/3



- DES的一个性质：
  - 没有两次加密与一次加密相同的情况: $E_K(M) \neq E_{K1}(E_{K2}(M))$
- 我们可以用多重DES
  - 用X 个密钥应用DES Y 次得到：YDES/X
  - 例如, 2DES/2, 3DES/2, 3DES/3
  - 我们可以用现有的DES有效的扩展加秘密钥的长度
  - 可抵御暴力攻击
- 例如, 3DES/2:

$$C = E_{K1}(D_{K2}(E_{K1}(M)))$$
$$M = D_{K1}(E_{K2}(D_{K1}(C)))$$

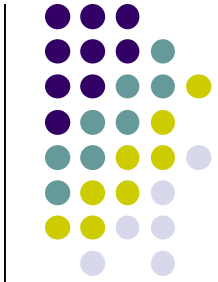
  - 注意: 其他组合 (如EEE和DDD)也是安全的,但利用现有的DES密文解密变得困难
  - 使用两个密钥将密钥扩展为112 bits,可使DES更安全从而抵御暴力攻击
- 注意2DES/2:
  - 2DES/2 使用与3DES/2一样多的密钥, 将密钥长度扩展为112
  - 然而, 2DES/2 无法抵御中途相遇攻击（*meet-in-the-middle* attack）

针对DES的中途相遇攻击

- 针对2DES/2的暴力攻击需要测试K_1 和K_2 的每种组合来找到合适的密钥(= 2^{56} x 2^{56} = 2^{112})
- 若攻击者得到两个明文密文对(M_1, C_1) 和(M_2, C_2) ， 其中$C_i = E_{K2}(E_{K1}(M_i))$
- 意味着$D_{K2}(C_i) = X_i = E_{K1}(M_i)$ 对两个明文密文对都成立
- 做两张表，一个用来基于56-bit密钥解密C，另一个用来加密M，结果匹配的项意味着一个潜在的K_1 and K_2匹配. (中途相遇)
- 对同一个明文密文对(M, C),有可能在两次返回相同结果的密钥对(K_1, K_2)的个数为$2^{112}/2^{64} = 2^{48}$.
- 对两个明文密文对(M, C),有可能在两次返回相同结果的密钥对的个数为$2^{48}/2^{64} = 2^{-16}$. （小概率事件）
- 因此, 找出(K_1, K_2) 的概率是$1-2^{-16}$。非常高！
- 时间复杂度逼近$2(2^{56} + 2^{48}) < 2^{58}$. 远小于$2^{112}$

第2章 内容概要

- 2.1 数据加密算法设计标准
- 2.2 数据加密标准DES
- 2.3 多重DES
- 2.4 高级加密标准AES
- 2.5 标准分组密码运算
- 2.6 流密码
- 2.7 密钥生成



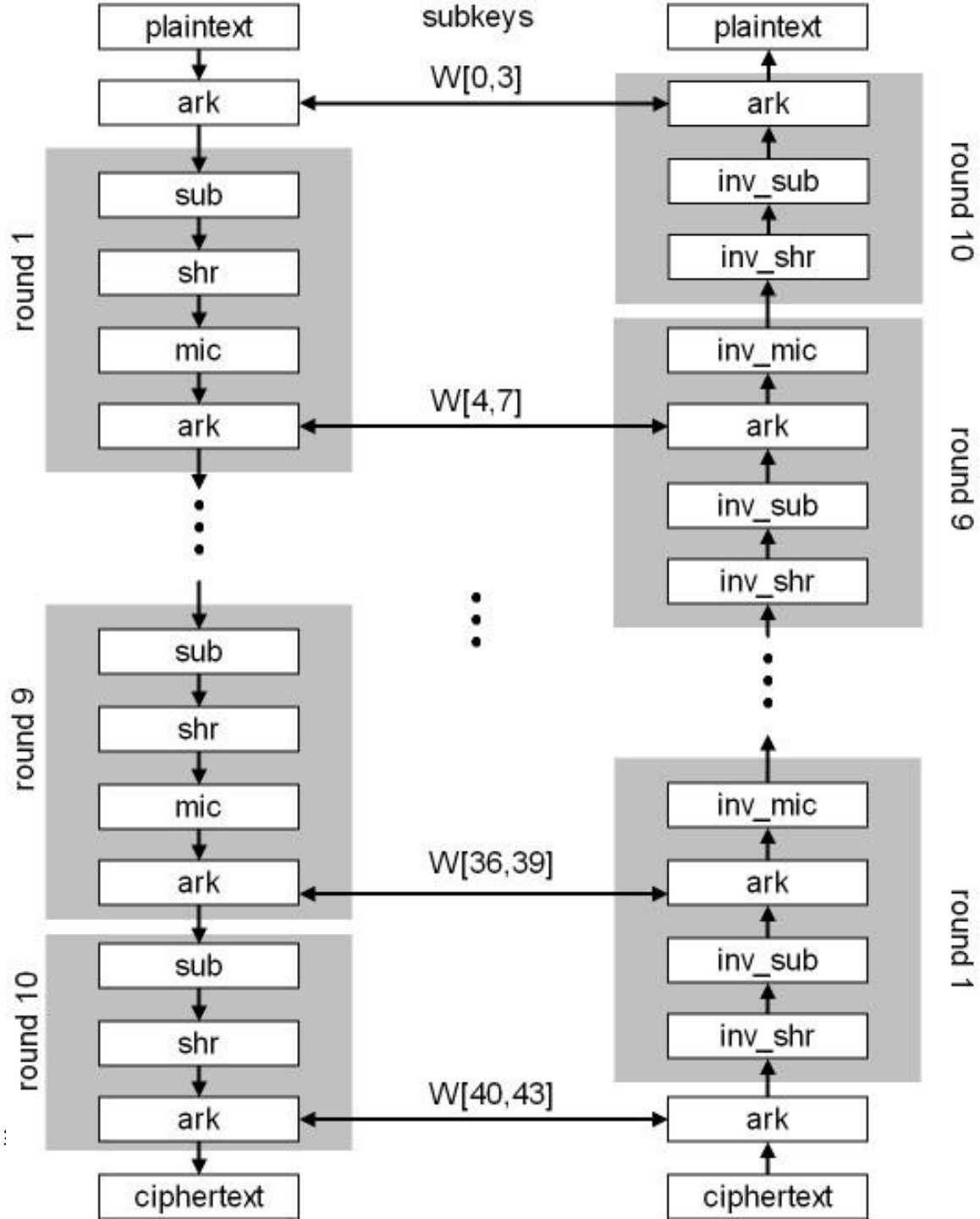
- 高级加密标准征集开始于1997年
- 2001年Rijndael被选出来作为新的AES
- AES的基本构造:
  - 分组密码，但不是Feistel密码
  - 加密和解密类似，但不对称
  - 基本的单元: 字节,不是比特
  - 分组大小: 16-字节(128 bits)
  - 三种不同的密钥长度: 128, 192, 256 bits
    - AES-128, AES-192, AES-256
  - 每16-字节分组由一个4X4的正方阵表示，叫做*状态矩阵*
  - 轮次数依赖于密钥长度
  - 每一轮在*状态矩阵*上执行4个简单的运算 (除了最后一轮)

四个简单运算:

- 字节替代 (sub)
  - 基于规定替换盒上进行非线性替换运算
  - 用于抵御密码学分析和数学分析攻击
- 行位移(shr)
  - 矩阵初等运算
  - 用于产生扩散性的线性运算
- 列混淆(mic)
  - 也用于产生扩散性的线基本运算
- 轮密钥加(ark)
  - 矩阵上进行异或运算
  - 线性运算
  - 产生混淆性

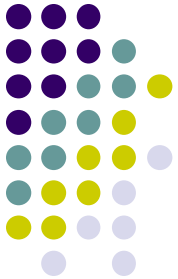
《计算机网络安全的理论与实践（第2版）》.【美】王杰，高等教育出版社，2011年.

AES-128


plaintext
ark
sub
shr
mic
ark
round 1
sub
shr
mic
ark
round 9
sub
shr
ark
round 10
ciphertext
subkeys
W[0,3]
W[4,7]
W[36,39]
W[40,43]
plaintext
ark
inv_sub
inv_shr
round 10
inv_mic
ark
inv_sub
inv_shr
round 9
inv_mic
ark
inv_sub
inv_shr
round 1
ark
ciphertext

《计算机网

AES S-Box



- S-box:在有限域GF(2^8)上构造的一个16x16矩阵
  - 置换GF(2^8)中的所有256个元素
  - 每个元素和其索引由两个十六进制数字表示
- 令$w = b_0b_1b_2b_3\,b_4b_5b_6b_7$ 为一个字节. 定义一个字节替换函数S 如下:
  令$i = b_0b_1b_2b_3$, 行索引的二进制形式
  令$j = b_4b_5b_6b_7$, 列索引的二进制形式
  令$S(w) = s_{ij,}\ S^{-1}(w) = s'_{ij}$
- 我们有$S(S^{-1}(w)) = w$ 且$S^{-1}(S(w)) = w$

AES-128 轮密钥

- 令$K = K[0,31]K[32,63]K[64,95]K[96,127]$ 表示一个4-字的加秘密钥
- AES 扩展K 到一个44-元数组$W[0,43]$，其中每个元素为4字节长
- 定义一个字节变换函数M 如下:

$$M(b_7b_6b_5b_4b_3b_2b_1b_0) = \begin{cases} b_6b_5b_4b_3b_2b_1b_00, & \text{if } b_7 = 0, \\ b_6b_5b_4b_3b_2b_1b_00 \oplus 00011011, & \text{if } b_7 = 1 \end{cases}$$

- 接着, 令j 为一个非负数. 定义$m(j)$如下:

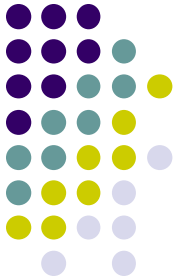
$$m(j) = \begin{cases} 00000001, & \text{if } j = 0 \\ 00000010, & \text{if } j = 1 \\ M(m(j-1)), & \text{if } j > 1 \end{cases}$$

- 最后, 定义一个字-替换函数 T 如下, 将一个32-bit串变换为另一个32-bit串, 用参数j 和AES S-Box:

$$T(w, j) = [(S(w_2) \oplus m(j-1)]S(w_3)\ S(w_4)\ S(w_1),$$

其中$w = w_1w_2w_3w_4$ 且每个w_i 是一个字节

《计算机网络安全的理论与实践（第2版）》. 【美】王杰, 高等教育出版社, 2011年.



- 用所有这些函数构造4-字大小的轮密钥 (AES-128需要11轮密钥;即44个字)

$$W[0] = K[0, 31]$$

$$W[1] = K[32, 63]$$

$$W[2] = K[64, 95]$$

$$W[3] = K[96, 127]$$

$$W[i] = \begin{cases} W[i-4] \oplus T(W[i-1], i/4), \text{若 } i \text{ 可被4整除} \\ W[i-4] \oplus W[i-1], \text{否则} \end{cases}$$

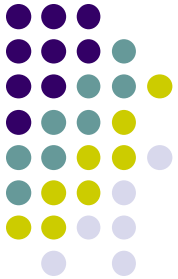
$$i = 4, \ldots, 43$$

- 11轮密钥: 对于$i = 0, \ldots, 10$:

$$K_i = W[4\text{i}, 4i + 3] = W[4i + 0]\ W[4i + 1]\ W[4i + 2]\ W[4i + 3]$$

《计算机网络安全的理论与实践（第2版）》.【美】王杰，高等教育出版社，2011年.

轮密钥加(*ark*)



- 将K_i重写为一个4 x 4 的字节矩阵:

$$K_i = \begin{pmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{pmatrix}$$

其中每个元素是一个字节且$W[4i + j] = k_{0,j}k_{1,j}k_{2,j}k_{3,j}$, $j = 0, 1 , 2, 3$

- 初始化，令 $A = M$

$$ark(A, K_i) = A \oplus K_i = \begin{pmatrix} k_{0,0} \oplus a_{0,0} & k_{0,1} \oplus a_{0,1} & k_{0,3} \oplus a_{0,3} & k_{0,4} \oplus a_{0,4} \\ k_{1,0} \oplus a_{1,0} & k_{1,1} \oplus a_{1,1} & k_{1,2} \oplus a_{1,2} & k_{1,3} \oplus a_{1,3} \\ k_{2,0} \oplus a_{2,0} & k_{2,1} \oplus a_{2,1} & k_{2,2} \oplus a_{2,2} & k_{2,3} \oplus a_{2,3} \\ k_{3,0} \oplus a_{3,0} & k_{3,1} \oplus a_{3,1} & k_{3,2} \oplus a_{3,2} & k_{3,3} \oplus a_{3,3} \end{pmatrix}$$

- 由于这是异或运算, ark^{-1} 与*ark*相同. 我们有

$$ark(ark^{-1}(A, K_i), K_i) = ark^{-1}(ark(A, K_i), K_i) = A$$

《计算机网络安全的理论与实践（第2版）》. 【美】王杰，高等教育出版社，2011年.

字节替换(*sub*)

- 回顾一下，*S* 是一个替换函数，将一个字节作为输入, 用其前4个比特作为行索引，后4个比特作为列索引,且输出一个字节用作S-box中的查找表
- 令*A* 是一个状态矩阵. 则：
-

$$sub(A) = \begin{pmatrix} S(a_{0,0}) & S(a_{0,1}) & S(a_{0,2}) & S(a_{0,3}) \\ S(a_{1,0}) & S(a_{1,1}) & S(a_{1,2}) & S(a_{1,3}) \\ S(a_{2,0}) & S(a_{2,1}) & S(a_{2,2}) & S(a_{2,3}) \\ S(a_{3,0}) & S(a_{3,1}) & S(a_{3,2}) & S(a_{3,3}) \end{pmatrix}$$

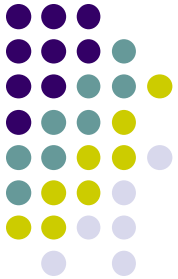
- $sub^{-1}(A)$ 将仅作为应用于矩阵的逆替换运算

$$sub^{-1}(A) = \begin{pmatrix} S^{-1}(a_{0,0}) & S^{-1}(a_{0,1}) & S^{-1}(a_{0,2}) & S^{-1}(a_{0,3}) \\ S^{-1}(a_{1,0}) & S^{-1}(a_{1,1}) & S^{-1}(a_{1,2}) & S^{-1}(a_{1,3}) \\ S^{-1}(a_{2,0}) & S^{-1}(a_{2,1}) & S^{-1}(a_{2,2}) & S^{-1}(a_{2,3}) \\ S^{-1}(a_{3,0}) & S^{-1}(a_{3,1}) & S^{-1}(a_{3,2}) & S^{-1}(a_{3,3}) \end{pmatrix}$$

- 我们有 $sub(sub^{-1}(A)) = sub^{-1}(sub(A)) = A$

《计算机网络安全的理论与实践（第2版）》. 【美】王杰, 高等教育出版社, 2011年.

行位移 (*shr*)



- $shr(A)$ 在矩阵A的第i-th行执行一个左循环移位$i-1$次

$$shr(A) = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,1} & a_{1,2} & a_{1,3} & a_{1,0} \\ a_{2,2} & a_{2,3} & a_{2,0} & a_{2,1} \\ a_{3,3} & a_{3,0} & a_{3,1} & a_{3,2} \end{pmatrix}$$

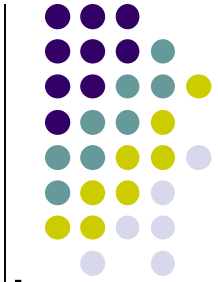
- $shr^{-1}(A)$在矩阵A的第i-th行执行一个右循环移位$i-1$次

$$shr^{-1}(A) = \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,3} & a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,2} & a_{2,3} & a_{2,0} & a_{2,1} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,0} \end{pmatrix}$$

- 我们有$shr(shr^{-1}(A)) = shr^{-1}(shr(A)) = A$

《计算机网络安全的理论与实践（第2版）》. 【美】王杰，高等教育出版社，2011年.

列混淆 (*mic*)



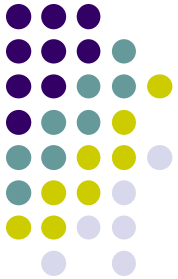
- $mic(A) = [a'_{ij}]_{4\times4}$，每一列由如下运算确定 ($j$ = 0, 1, 2, 3):

$$a'_{0,j} = M(a_{0,j}) \oplus [M(a_{1,j}) \oplus a_{1,j}] \oplus a_{2,j} \oplus a_{3,j}$$
$$a'_{1,j} = a_{0,j} \oplus M(a_{1,j}) \oplus [M(a_{2,j}) \oplus a_{2,j}] \oplus a_{3,j}$$
$$a'_{2,j} = a_{0,j} \oplus a_{1,j} \oplus M(a_{2,j}) \oplus [M(a_{3,j}) \oplus a_{3,j}]$$
$$a'_{3,j} = [M(a_{0,j}) \oplus a_{0,j}] \oplus a_{1,j} \oplus a_{2,j} \oplus M(a_{3,j})$$

- $mic^{-1}(A)$ 定义如下:
  - 令w 为一个字节，i 为一个正整数:

$$M^i(w) = M(M^{i-1}(w)) \ (i > 1), \ M^1(w) = M(w)$$

列混淆(*mic*)—续.



- 令

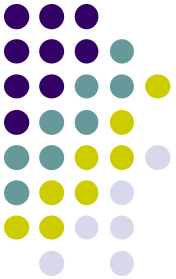
$$M_1(w) = M^3(w) \oplus M^2(w) \oplus M(w)$$
$$M_2(w) = M^3(w) \oplus M(w) \oplus w$$
$$M_3(w) = M^3(w) \oplus M^2(w) \oplus w$$
$$M_4(w) = M^3(w) \oplus w$$

- $mic^{-1}(A) = [a''_{ij}]_{4\times 4}$:

$$a''_{0,j} = M_1(a_{0,j}) \oplus M_2(a_{1,j}) \oplus M_3(a_{2,j}) \oplus M_4(a_{3,j})$$
$$a''_{1,j} = M_4(a_{0,j}) \oplus M_1(a_{1,j}) \oplus M_2(a_{2,j}) \oplus M_3(a_{3,j})$$
$$a''_{2,j} = M_3(a_{0,j}) \oplus M_4(a_{1,j}) \oplus M_1(a_{2,j}) \oplus M_2(a_{3,j})$$
$$a''_{3,j} = M_2(a_{0,j}) \oplus M_3(a_{1,j}) \oplus M_4(a_{2,j}) \oplus M_1(a_{3,j})$$

- 我们有$mic(mic^{-1}(A)) = mic^{-1}(mic(A)) = A$

AES-128 加密



- AES-128 加密:
- 令A_i (i = 0, …, 11) 是状态矩阵的一个序列, 其中A_0 is 初始状态矩阵M, 且A_i (i = 1, …, 10) 代表第i轮的输入状态矩阵
- A_{11} 是密码本分组C, 由如下运算获得:

$$A_1 = ark(A_0, K_0)$$

$$A_{i+1} = ark(mic(shr(sub(A_i))), K_i),\ i = 1,\ldots,9$$

$$A_{11} = arc(shr(sub(A_{10})), K_{10}))$$

《计算机网络安全的理论与实践（第2版）》.【美】王杰，高等教育出版社，2011年.

AES-128 解密

- AES-128 解密:
- 令$C_0 = C = A_{11}$, 其中C_i 是第i轮的输入状态矩阵

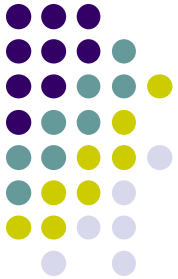
$$C_1 = ark(C_0, K_{10})$$

$$C_{i+1} = mic^{-1}(ark(sub^{-1}(shr^{-1}(C_i)), K_{10-i}))$$

$$i = 1,\ldots,9$$

$$C_{11} = ark(sub^{-1}(\mathrm{shr}^{-1}(C_{10})), K_0)$$

解密的正确性证明



- 为证明$C_{11} = A_0$
- 我们先利用数学归纳证明下述等式:

$$C_i = shr(sub(A_{11-i})),\ i = 1, \ldots, 10$$

对$i = 1$ 我们有

$$\begin{aligned} C_1 &= ark(A_{11}, K_{10}) \\ &= A_{11} \oplus K_{10} \\ &= ark(shr(sub(A_{10})), K_{10}) \oplus K_{10} \\ &= (shr(sub(A_{10})) \oplus K_{10}) \oplus K_{10} \\ &= shr(sub(A_{10})) \end{aligned}$$

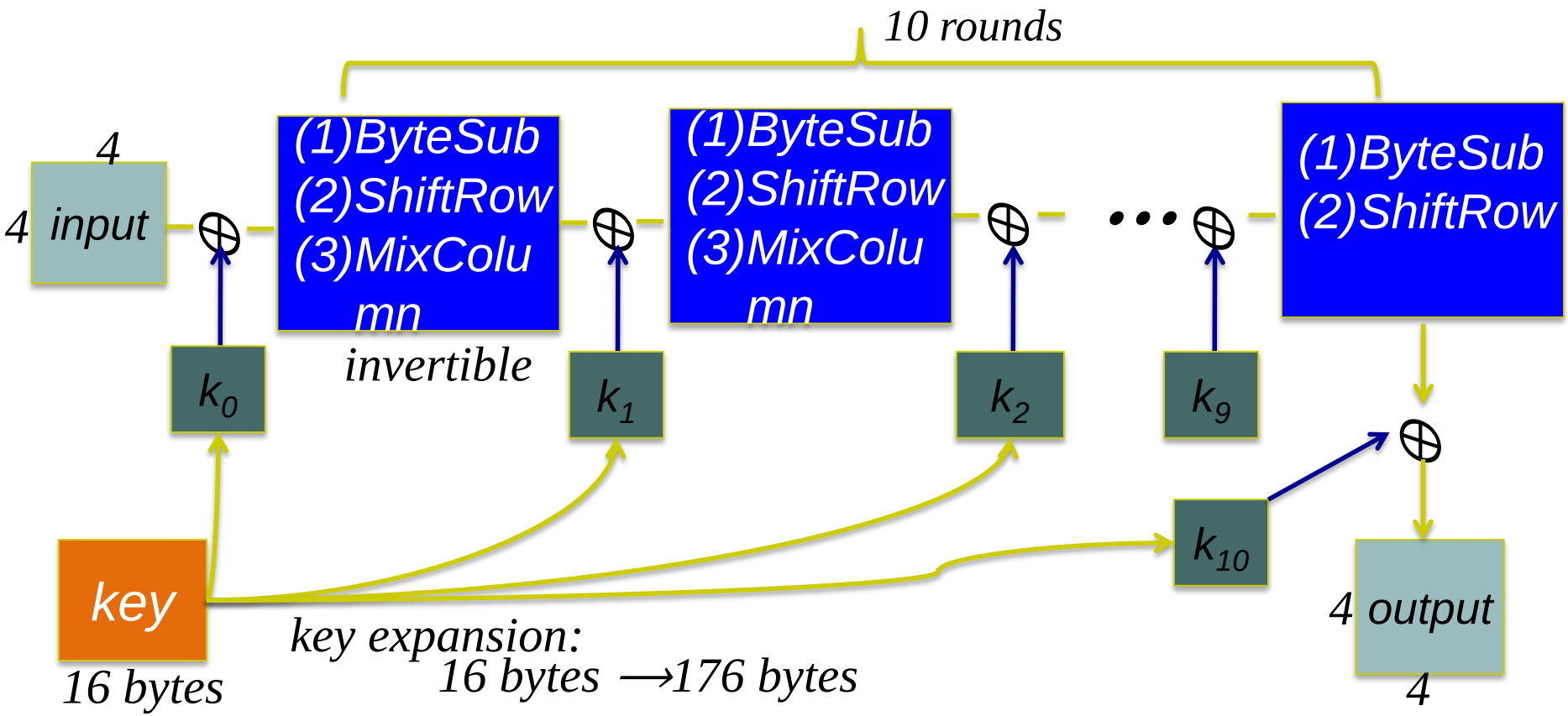
- 假定对于$1 \le i \le 10$,等式成立. 我们有
- $$C_{i+1} = mic^{-1}(ark(sub^{-1}(shr^{-1}(\underline{C_i})), K_{10-i}))$$
  $$= mic^{-1}(ark(sub^{-1}(shr^{-1}(shr(sub(A_{11-i}))))) \oplus K_{10-i}))$$
  $$= mic^{-1}(A_{11-i} \oplus K_{10-i})$$
  $$= mic^{-1}(ark(mic(shr(sub(A_{10-i}))), K_{10-i}) \oplus K_{10-i})$$
  $$= mic^{-1}([mic(shr(sub(A_{10-i}))) \oplus K_{10-i}] \oplus K_{10-i})$$
  $$= shr(sub(A_{10-i})$$
  $$= shr(sub(A_{11-(i+1)}))$$
- 这就完成了归纳证明

- 最后，我们有

$$
\begin{aligned}
C_{11} &= ark(sub^{-1}(shr^{-1}(C_{10})), K_0) \\
&= sub^{-1}(shr^{-1}(shr(sub(A_1)))) \oplus K_0 \\
&= A_1 \oplus K_0 \\
&= (A_0 \oplus K_0) \oplus K_0 \\
&= A_0
\end{aligned}
$$

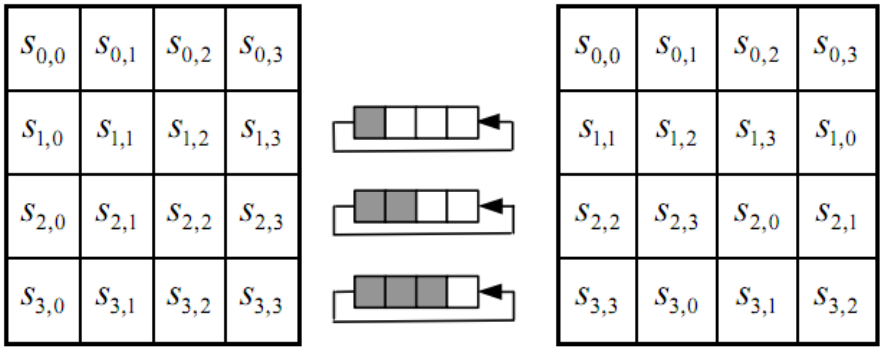
AES-128 解密的正确性证毕。

AES-128 图解


10 rounds
4
4 input
(1)ByteSub
(2)ShiftRow
(3)MixColu
mn
invertible
(1)ByteSub
(2)ShiftRow
(3)MixColu
mn
(1)ByteSub
(2)ShiftRow
k0
k1
k2
k9
k10
key
16 bytes
key expansion:
16 bytes ⟶176 bytes
4 output
4

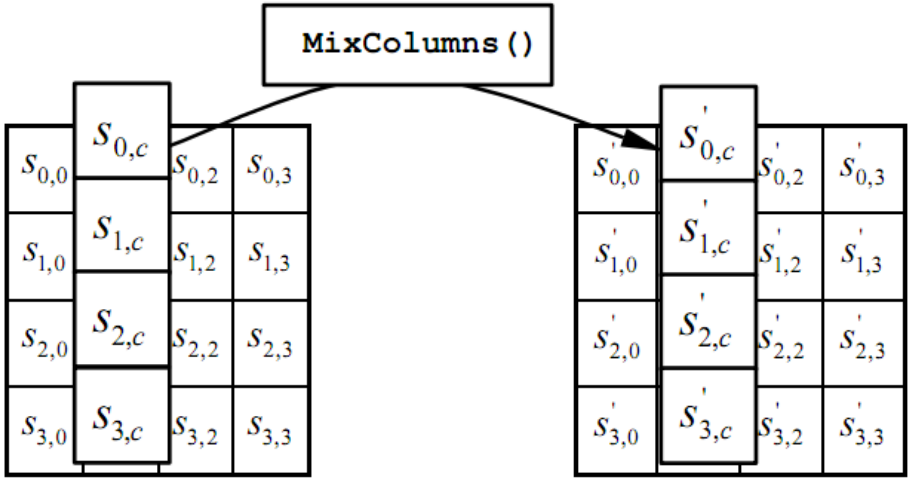
每轮变换函数

- **ByteSub**: a 1 byte S-box. 256 byte table
  (easily computable)

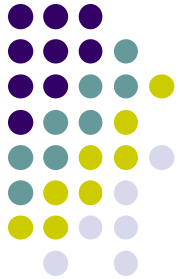


- **ShiftRows**:

- **MixColum**



Code size/performance tradeoff



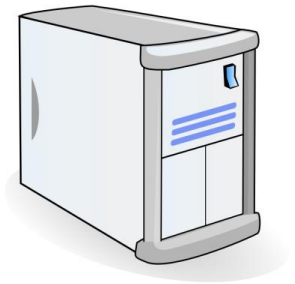
	Code size	Performance
Pre-compute round functions (24KB or 4KB)	largest	fastest: table lookups and xors
Pre-compute S-box only (256 bytes)	smaller	slower
No pre-computation	smallest	slowest

Example: Javascript AES

AES in the browser:

AES library (6.4KB)

no pre-computed tables



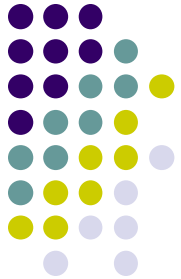
Prior to encryption:

pre-compute tables

Then encrypt using tables

http://crypto.stanford.edu/sjcl/

AES in hardware



AES instructions in Intel Westmere:

- **aesenc, aesenclast**: do one round of AES

  128-bit registers: xmm1=state, xmm2=round key

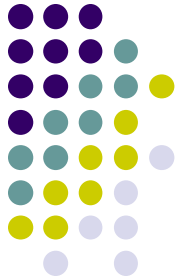
  aesenc xmm1, xmm2 ; puts result in xmm1

- **aeskeygenassist**: performs AES key expansion

- Claim 14 x speed-up over OpenSSL on same hardware

Similar instructions on AMD Bulldozer

Attacks on AES



Best key recovery attack:

four times better than ex. search [BKR'11]

Related key attack on AES-256: [BK'09]

Given 2^{99} inp/out pairs from **four related keys** in AES-256

can recover keys in time $\approx 2^{99}$

第2章 内容概要

- 2.1 数据加密算法设计标准
- 2.2 数据加密标准DES
- 2.3 多重DES
- 2.4 高级加密标准AES
- 2.5 标准分组密码运算 (分组密码的工作模式)
- 2.6 流密码
- 2.7 密钥生成

分组密码的工作模式

- 令 l 为给定分组密码的分组大小
  ($l = 64$ in DES, $l = 128$ in AES).
- 令M 为一个明文串. 将M 分为一个分组的序列:
  $$M = M_1 M_2 \ldots M_k,$$
  满足 $|M_i| = l$ (必要时对最后一块进行填充)

- 有几种加密M的方法, 被称为分组密码的工作模式

- 标准的分组密码的工作模式:
  - 电子密码本 (ECB)
  - 密码块链接 (CBC)
  - 密文反馈(CFB)
  - 输出反馈 (OFB)
  - 计数器模式(CTR)

电子密码本 (ECB)

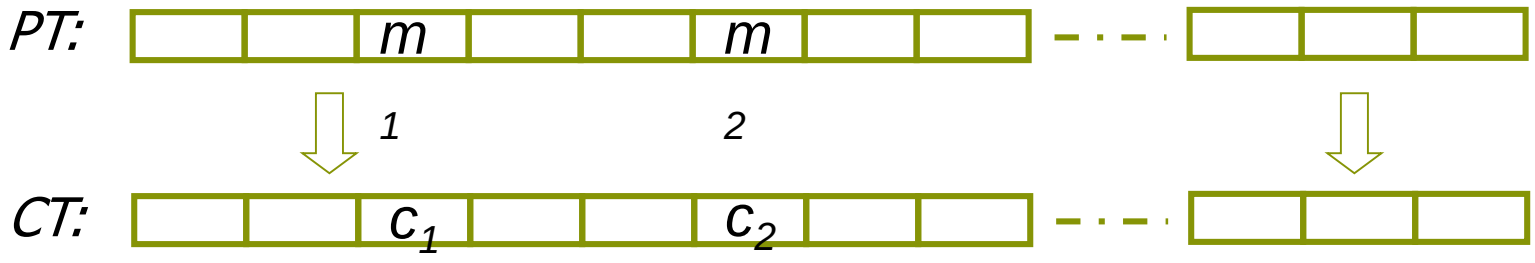
- ECB 独立的加密每个明文分组. 令C_i 为第i-th 密码本分组:

ECB 加密步骤	ECB 解密步骤
$C_i = E_k(M_i),$ $i = 1,2,\cdots,k$	$M_i = D_k(C_i),$ $i = 1,2,\cdots,k$

- 简单易懂. ECB常用于加密短的明文消息
- 然而,如果我们将我们的串进行分组, 两个分组相同是有可能的:
  $M_i = M_j$ $(i \neq j)$
- 这使得攻击者能够得到关于加密的某些信息
- 其他工作模式以不同的方式处理这个问题

Incorrect use of block ciphers

Electronic Code Book (ECB):



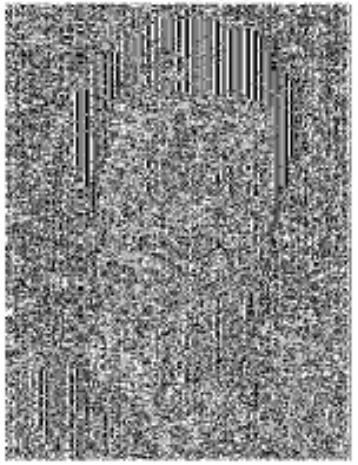
Problem:

- if $m_1=m_2$ then $c_1=c_2$

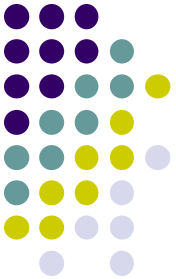
In pictures

An example plaintext

Encrypted with AES in ECB mode



密码分组链接 (CBC)



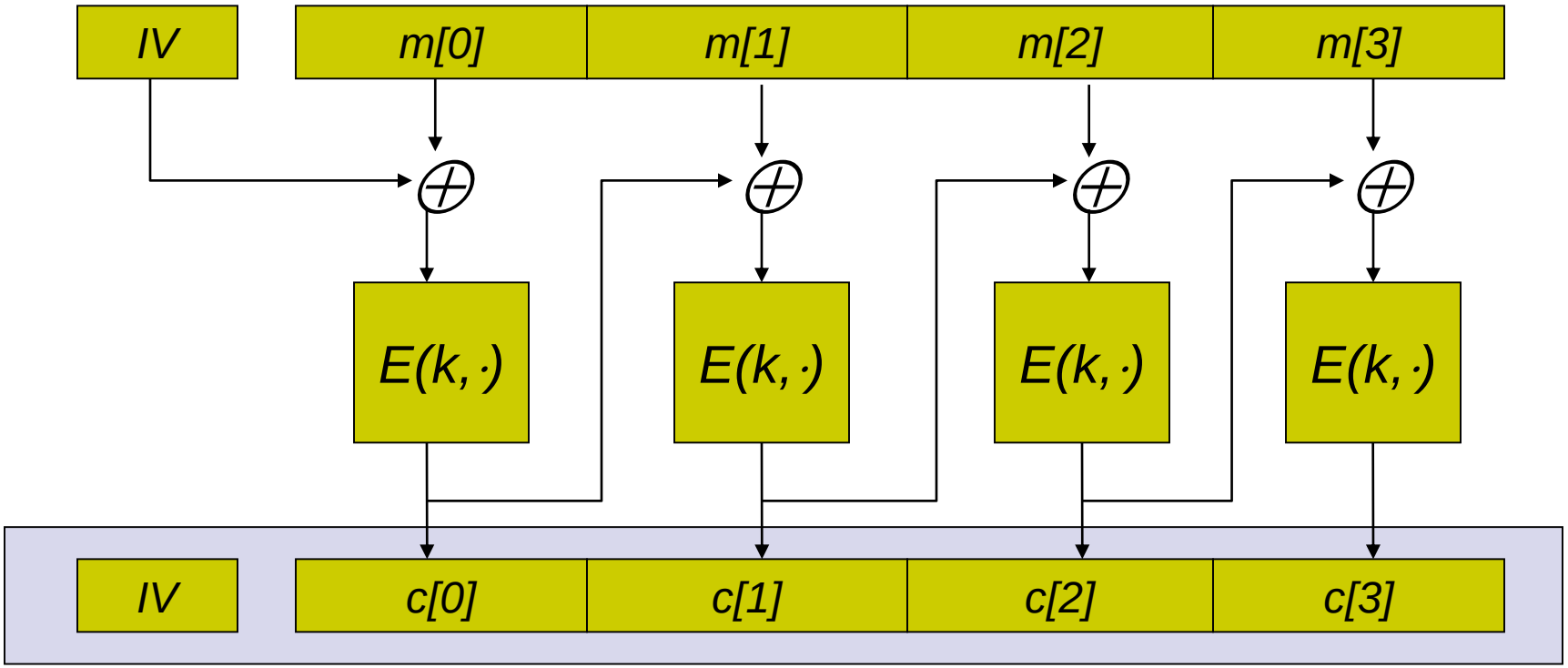
- 在ECB模式下，当明文消息 *M* 较长, 对于每个$i \neq j$, $M_i = M_j$ *的概率 将增加*
- CBC 克服了ECB的缺点
- CBC模式中, 前一个密码本分组用于加密当前的明文分组
- CBC用一个初始的*l*-bit 分组C_0, 称为*初始向量*

CBC 加密步骤	*CBC* 解密步骤
$C_i = E_k(C_{i-1} \oplus M_i),$ $i = 1,2,\cdots,k$	$M_i = D_k(C_i) \oplus C_{i-1},$ $i = 1,2,\cdots,k$

- 如果在传输的时候，在密码本分组中出现一个比特错误怎么办？(扩散)
- 在C_i 中的一个比特的变化影响接下来的分组

Correct use of block ciphers I: CBC mode

E a secure PRP. *Cipher Block Chaining with random IV:*



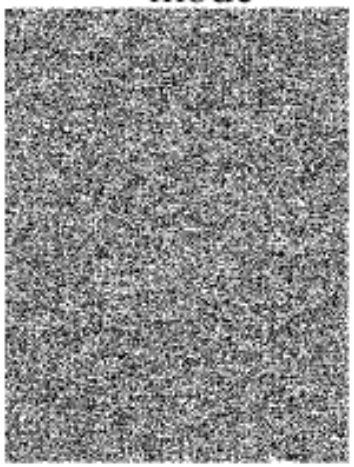
ciphertext

Q: how to do decryption?

In pictures

An example plaintext

Encrypted with AES in CBC mode



密文反馈 (CFB)

- *CFB 把分组密码变为流密码*
- *M = $w_1 w_2 \ldots w_m$, 其中w_i的长度为s-bit*
- *一次加密一个s-bit的分组（以明文字符为单位）:*
  - *s=8:* *ASCII码的流密码*
  - *s=16:* *unicode 流密码*
- *Also has an l-bit initial vector V_0*

$$pfx_s(U) = \text{S bits prefix of U}$$

$$sfx_s(U) = \text{S bits subfix of U}$$

CFB 加密步骤	CFB 解密步骤
$U_i = E_k(V_{i-1})$ $C_i = w_i \oplus pfx_s(U_i)$ $V_i = sfx_{l-s}(V_{i-1})C_i$ $i = 1,2,\cdots,m-1$ $U_m = E_k(V_{m-1})$ $C_m = w_m \oplus pfx_s(U_m)$	$U_i = E_k(V_{i-1})$ $w_i = C_i \oplus pfx_s(U_i)$ $V_i = sfx_{l-s}(V_{i-1})C_i$ $i = 1,2,\cdots,m-1$ $U_m = E_k(V_{m-1})$ $w_m = C_m \oplus pfx_s(U_m)$

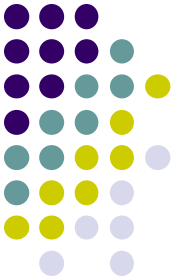
输出反馈 (OFB)

- OFB也把分组密码变为流密码
- CFB 和OFB仅有的区别是OFB 不在V_i设置C_i .
- 消息的反馈是独立的
- 在易出错的环境中使用

OFB 加密步骤	OFB 揭秘步骤
$U_i = E_k(V_{i-1})$ $C_i = w_i \oplus pfx_s(U_i)$ $V_i = sfx_{l-s}(V_{i-1})pfx_s(U_i)$ $i = 1,2,\cdots,m-1$ $U_m = E_k(V_{m-1})$ $C_m = w_m \oplus pfx_s(U_m)$	$U_i = E_k(V_{i-1})$ $w_i = C_i \oplus pfx_s(U_i)$ $V_i = sfx_{l-s}(V_{i-1})pfx_s(U_i)$ $i = 1,2,\cdots,m-1$ $U_m = E_k(V_{m-1})$ $w_m = C_m \oplus pfx_s(U_m)$

《计算机网络安全的理论与实践（第2版）》. 【美】王杰，高等教育出版社，2011年.

计数器模式(CTR)



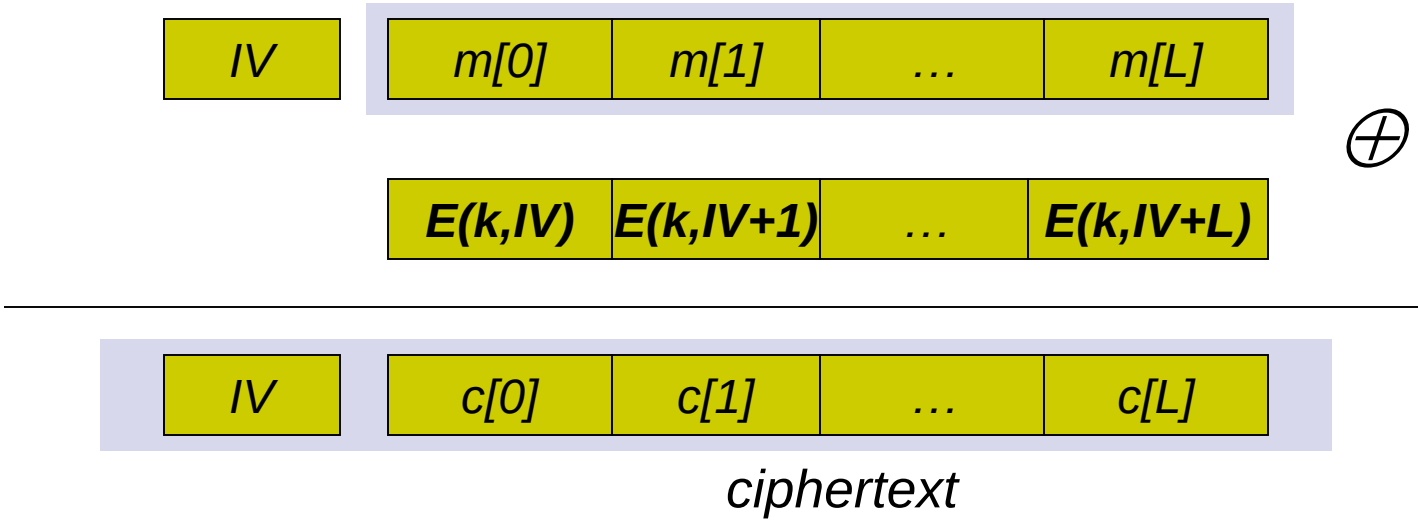
- CTR是分组模式.
- 一个l-bit 的计数器Ctr, 从一个初始值开始，每次增加1
- 用于需要高速加密的应用

CTR 加密步骤	CTR 解密步骤
$Ctr = Ctr_0$ $C_i = E_k(Ctr^{++}) \oplus M_i$ $i = 1,2,\cdots,k$	$Ctr = Ctr_0$ $M_i = E_k(Ctr^{++}) \oplus C_i$ $i = 1,2,\cdots,k$

《计算机网络安全的理论与实践（第2版）》.【美】王杰，高等教育出版社，2011年.

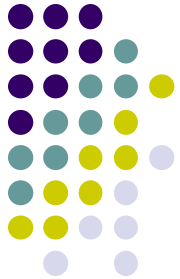
Correct use of block ciphers II: CTR mode

Counter mode with a random IV: (parallel encryption)



- *Why are these modes secure?* *not today.*

Performance: Crypto++ 5.6.0 [Wei Dai]



Intel Core 2 (on Windows Vista)

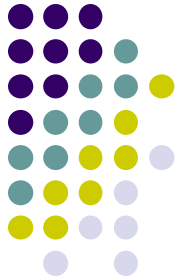
Cipher	Block/key size	Speed (MB/sec)
RC4		126
Salsa20/12		643
3DES	64/168	10
AES/GCM	128/128	102

AES is about 8x faster with AES-NI : Intel Westmere and onwards

第2章 内容概要

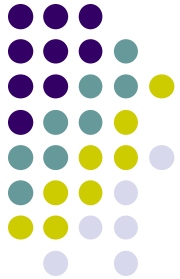
- 2.1 数据加密算法设计标准
- 2.2 数据加密标准DES
- 2.3 多重DES
- 2.4 高级加密标准AES
- 2.5 标准分组密码运算
- 2.6 流密码
- 2.7 密钥生成

流密码



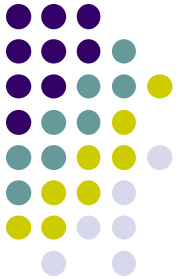
- 流密码一次加密一个比特，一个字节或几个比特的消息
- 采用CFB或OFB等方法，任意分组密码算法能够转换为流密码；但计算繁琐
- 如果构造轻量级的流密码?

RC4 流密码



- 由Rivest 设计, 是一种轻量级的流密码
  - 是IEEE 802.11b无线通信安全标准 WEP的关键部分
  - 具有可变的密钥长度: 从1 字节到256字节
  - 采用了三种运算：替代，模加和异或

RC4 轮密钥生成



设 K 是一个加密密钥:

$K = K[0]K[1] \ldots K[l-1]$,

其中 $|K|=8l$, $1 \le l \le 256$

- RC4 用一个256 字节的数组S[0, 255] 生成轮密钥
- 对数组中的字节进行置换

密钥扩展算法(KSA)

Initialization :

for each i = 0, . . . , 255

set S[i] ← i

Initial permutation :

set j ← 0

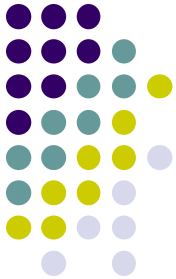
for each i = 0, 1, . . . , 255

set j ← (j + S[j] + K[j mod l]) mod 256

swap S[i] with S[j]

《计算机网络安全的理论与实践（第2版）》. 【美】王杰, 高等教育出版社, 2011年.

轮密钥生成算法(SGA)



Initialization :

set i $\leftarrow 0$

set j $\leftarrow 0$

set u $\leftarrow 0$

Permutation and generation loop :

set u $\leftarrow$ u + 1

set i $\leftarrow$ (i + 1) mod 256

set j $\leftarrow$ (j + S[i]) mod 256

swap S[i] with S[j]

set $K_u \leftarrow S[(S[i] + S[j]) \bmod 256]$

repeat

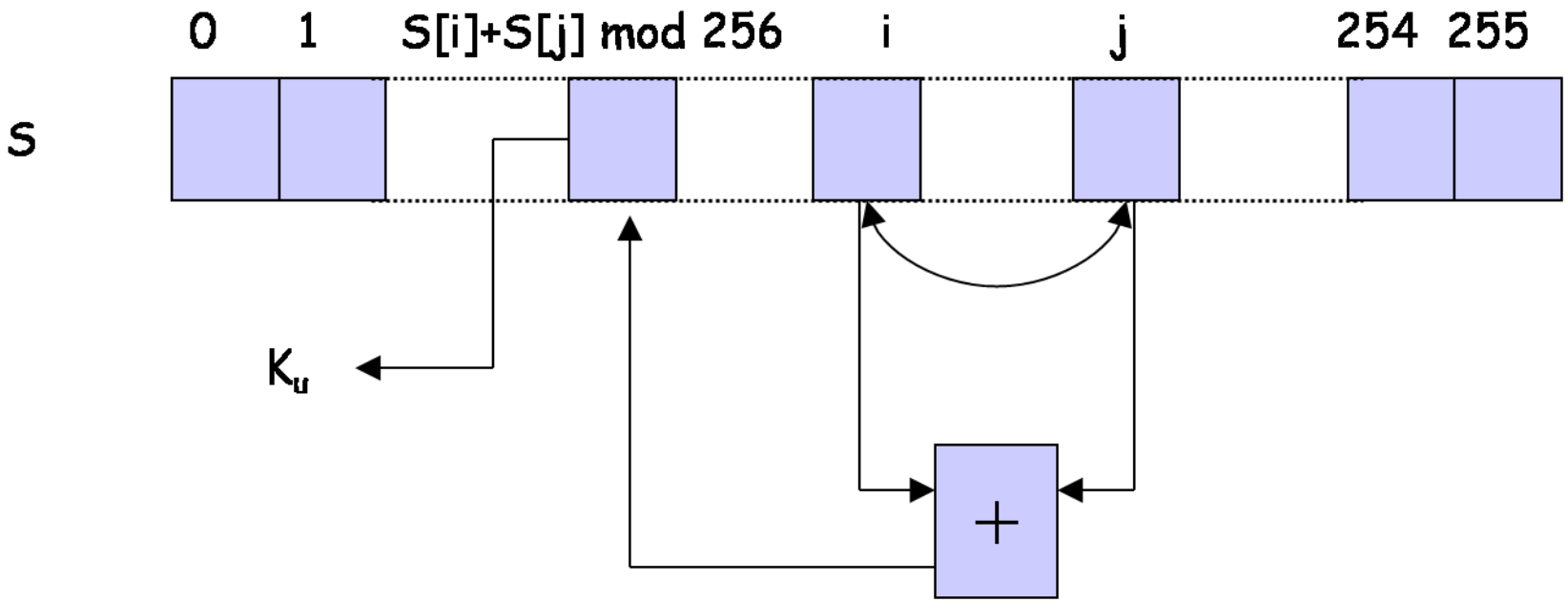
《计算机网络安全的理论与实践（第2版）》.【美】王杰，高等教育出版社，2011年.

RC4 加密和解密

Let $M = M_1 M_2 \cdots M_k$, M_i is an 8-bit binary string.

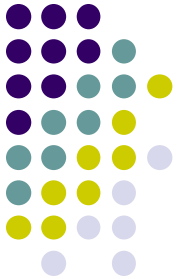
RC4 encryption: $C_i = M_i \oplus K_i, i = 1, \cdots, k$

RC4 decryption: $M_i = C_i \oplus K_i, i = 1, \cdots, k$



《计算机网络安全的理论与实践（第2版）》.【美】王杰，高等教育出版社，2011年.

RC4 的安全弱点



- 知道KSA中生成的S的初始置换等价于破解 RC4 加密
- 弱密钥: 密钥K里的一小部分比特串能够确定初始置换中的大量比特, 这有助于恢复秘密密钥（[Fluhrer,Mantin,and Shamir,2001]）
  - 表明 选择一个合适的密钥K，使之能产生一个安全的初始置换是很困难的。

RC4 的安全弱点

- 密钥重用情况下（要求自密钥流只能用一次，否则）：
  - 已知明文攻击: 获取加密的轮密钥流
  - 相关明文攻击（以下示例）

$$M_1 = m_{11} m_{12} \cdots m_{1n}$$

$$M_2 = m_{21} m_{22} \cdots m_{2n}$$

each m_{ij} is a binary bit.

$$c_{ij} = m_{ij} \oplus k_j$$

suppose the attacker intercepts C_1 and C_2

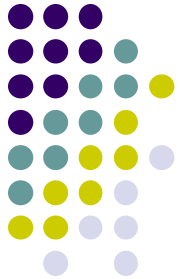
$$c_{1j} \oplus c_{2j} = (m_{1j} \oplus k_j) \oplus (m_{2j} \oplus k_j) = m_{1j} \oplus m_{2j}$$

$$j = 1, 2, \cdots, n$$

第2章 内容概要

2.1 数据加密算法设计标准
2.2 数据加密标准DES
2.3 多重DES
2.4 高级加密标准AES
2.5 标准分组密码运算
2.6 流密码
2.7 密钥生成

密钥生成



- 秘密密钥是加密算法中最关键的数据
- 最好的方式: 随机生成
  - 用确定算法生成伪随机数 (称为伪随机数生成器 “PRNG”);.
    - ANSI X9.17 PRNG
    - BBS 伪随机数生成

ANSI X9.17 PRNG

- 1985由美国国家标准技术研究所American National Standard Institute (ANSI) 发布
- 基于3DES/2，具有两个密钥K_1 and K_2, 和初始化向量V_0
- 两个特殊的64比特二进制串T_i 和V_i:
  - T_i 表示当前的日期和时间, 在每一轮之前更新
  - V_i 是一个种子，用以下方法计算:

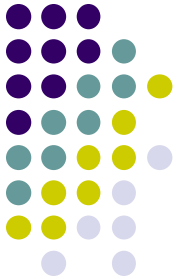
$$R_i = EDE_{K_1,K_2}(V_i \oplus EDE_{K_1,K_2}(T_i)),$$

$$V_{i+1} = EDE_{K_1,K_2}(R_i \oplus EDE_{K_1,K_2}(T_i)),$$

$$i = 0,1,\ldots$$

《计算机网络安全的理论与实践（第2版）》.【美】王杰，高等教育出版社，2011年.

BBS 伪随机数生成器



- 每一轮计算产生一个伪随机比特
- 设 p 和 q 是两个大素数，满足

$$p \bmod 4 = q \bmod 4 = 3$$

- 令 $n = p \times q$，s 是正整数, 其中
  - s 和 p 是互素的，即$\gcd(s,p) = 1$
  - s 和 q 是互素的，即$\gcd(s,q) = 1$
- BBS 为随机数生成:

$$x_0 = s^2 \bmod n,$$

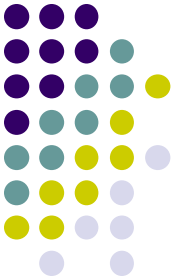
$$x_i = x_{i-1}^2 \bmod n,$$

$$b_i = x_i \bmod 2,$$

$$i = 1, 2, \ldots$$

《计算机网络安全的理论与实践（第2版）》.【美】王杰，高等教育出版社，2011年.

BBS伪随机数生成器的评价?



- 通过前面的 k 比特$b_1, \ldots, b_k$， 预测第(k+1)比特b_{k+1} 的困难性依赖于整数分解问题
- 整数分解问题: 对一个给定的非素数的整数n, 求 n的*素因子*
  - 目前已知的最好的算法的时间复杂性

$$e^{\sqrt[3]{\ln n (\ln \ln n)^2}}$$

- *如果整数分解问题不能在多项式时间内求解，则 BBS 生成的随机数比特与真随机数是不能在多项式时间内区分的*
- 在理论的量子计算机模型上，整数分解问题能够多项式时间内求解

《计算机网络安全的理论与实践（第2版）》.【美】王杰，高等教育出版社，2011年.

2016年英特尔杯大学生电子设计竞赛嵌入式系统专题邀请赛组委会将向各参赛队同时提供一块基于英特尔凌动处理器的Minnow Board Turbot嵌入式平台和一块英特尔和Arduino公司合作开发的基于英特尔居里平台的Genuino 101（美国地区为Arduino 101）开发板。参赛队必须基于英特尔凌动处理器的Minnow Board Turbot嵌入式平台，自主命题、自主设计，独立完成一个有一定功能的应用系统（竞赛作品），Genuino 101（美国地区为Arduino 101）平台可根据项目需要，由参赛队自主决定是否使用。

上报参赛学生名单截止时间：2016年5月31日
上报参赛选题截止时间：2016年5月31日
上报参赛作品设计报告截止时间：2016年7月1日

http://nuedc.sjtu.edu.cn/CN/Default.aspx