分组密码算法及其工作模式

清华大密码理论与技术研究中心
于红波
2015年3月22日

推荐文献

- 密码学原理与实践， 第三章
- Wikipedia
  - 分组密码(Block Cipher)
    http://en.wikipedia.org/wiki/Block_cipher
  - Feistel structure
    http://en.wikipedia.org/wiki/Feistel_cipher
  - DES
    http://en.wikipedia.org/wiki/Data_Encryption_Standard

密码体制的安全性

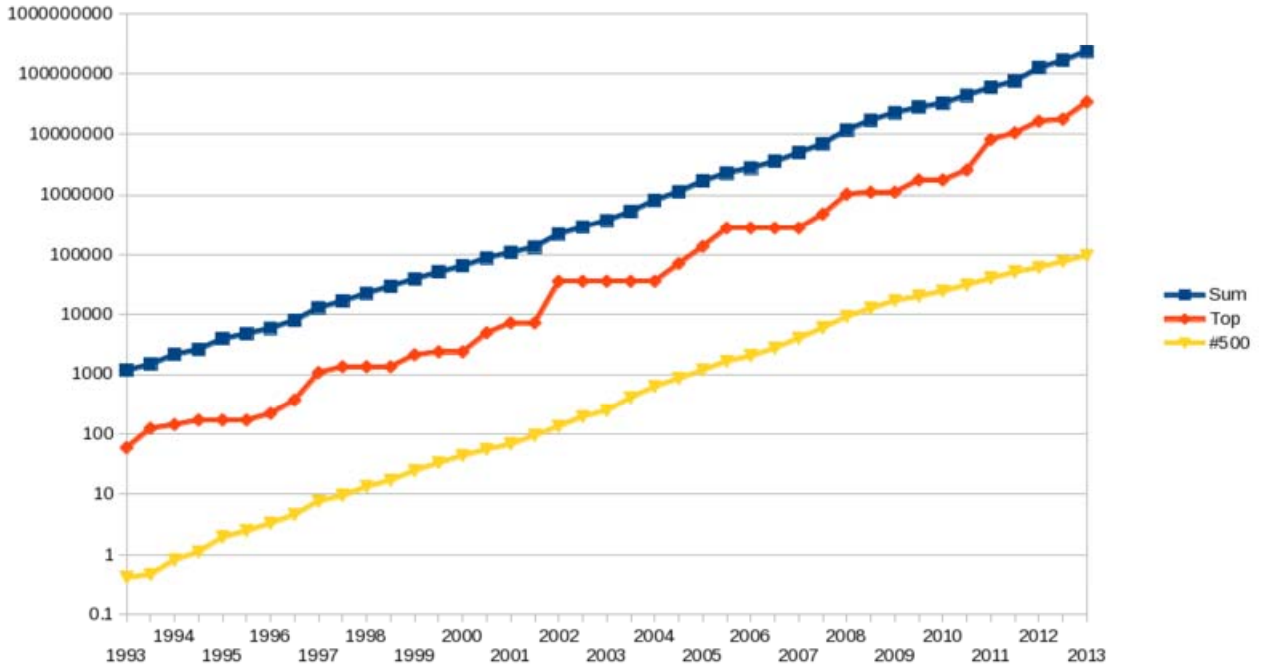
- 计算安全性(computational security)
  - 破译一个密码体制所做的计算上的努力
  - 如果使用最好的算法破译一个密码体制至少需要N次操作（N是一个特定的非常大的数字），定义该密码体制是计算安全的
- 可证明安全性(provable security)
  - 通过规约的方式为安全性提供证据
  - 如果可以破译密码体制A，则就可以解决一个数学难题B（分解因子问题，离散对数问题）
- 无条件安全性(unconditional security)即完善保密性(perfect security)
  - 即使攻击者有无限的计算资源也不可能攻破密码体制，则该密码体制是无条件安全的
  - OTP是无条件安全的

现有的计算资源

- Intel Core-i3, i5, i7 CPUs
  - Widely used CPUs in 2011 (notebook, desktop computers)
  - Normally each costs RMB 4000-10000
  - Each has two or four cores, runs at the speed about 2.5GHz
  - Each core performs about 2^{32} 64-bit integer (or floating point) operations per second
- The most powerful supercomputers in the world in 2015
  - 天河二号（TH-2），33.86 pflops, China，2013,
  - Cray Titan, 17.59 pflops, USA, 2012 （1P=1024T 1T=1024G 1G=1024M 1M=1024K）
  - IBM Sequoia（红杉）, 16.32 pflops, USA, 2012
  - K computer（京）, Japan, 10.51 pflops , 2011, (FLoating Point Operatios Per Second)
  - Tianhe-1A（天河一号）, China, 2.6 pflops (fastest in 2010)
  - Jaguar（美洲豹）, USA, about 1.7 6 pflops (fastest in 2009)
  - IBM Roadrunner, USA, 1.105pflops (2008)
- The combined computing power of top 500 supercomputers：84 pflops in 2011(2^{81} operation per year)
- Computer is getting faster and faster (for the same price)
  - 128-bit key is needed today

超级计算机

- http://en.wikipedia.org/wiki/Supercomputer



Growth of supercomputers performance.

唯密文攻击&已知明文攻击

- 唯密文攻击（Cipher-test only attack）
  - 攻击者仅知道密文
  - 攻击者知道明文的一些统计信息
- 已知明文攻击
  - 攻击者
    - 仅知道当前密钥下一些明密文对
    - 恢复密钥，从而恢复用该密钥加密过的其他消息
  - 已知明文攻击在许多应用现实的应用环境下都是可行的
    - QQ
    - 电话、数字电视

Kerckhoff's principle

- Kerchhoff 假设
  - 攻击者知道所使用的密码系统（除密钥外）
  - 该密码体制仍然是安全的
- 根据Kerchhoff假设，密码算法是否应该公开？

实际的对称密码算法

- 实际的对称加密算法需要满足两个条件
  - 计算安全
    - 攻击者知道所使用的密码体制（Kerckhoff假设）
    - 能够抵抗已知明文攻击
  - 容易使用
    - 相对短的密钥能够用来加密很多消息
- 实际的对称加密体制
  - 分组密码（Block Cipher)
  - 序列密码 （Stream Cipher)

分组密码

- 代换密码（substitution cipher）
  - 代换表太小， 26个字母
  - 不能够抵抗唯密文攻击和已知明文攻击
- 如果把代换表增加到2^{128}个元素？
  - The resulting cipher is strong!
    - 能够抵抗两种攻击
  - 密钥太长，没法存储
    - Internation Data Corporation （IDT）估计：到2009年，全世界产生的数据是 1.2 millision Petabyte (2^{70} byte=10^{21})

分组密码算法

- 分组密码可以看成是一个巨大的“代换密码”
  - DES：2^{64}个元素的代换表
  - AES：2^{128}个元素的代换表
- 表中每个元素只被计算一次
  - 无需存储巨大的表
  - 如何计算每个元素？
    - 等价于： 如何加密一个消息分组？
      如何设计一个分组密码算法？

分组密码研究进展

- DES时代
  - 1973年：NBS(NIST)征集加密标准，IBM提交
  - 1974年：NBS第二次征集
    - IBM提交64比特密钥的算法
    - NSA加入设计，建议缩减密钥长度
  - 1976年：DES被NBS采纳为数据加密标准(FIPS 46)
  - 1983年：DES被再次确认为标准
  - 1988年：1993年：第二次，第三次确认标准
  - 1998年：DES被破译 （55小时，暴力攻击）
  - 1999年：DES第四次被确认（FIPS 46-3， Triple-DES）

分组密码研究进展（续）

- AES时代
  - 1997年，NIST公开征数据加密算法以取代DES
  - 1998年，共收到15个算法
  - 1999年，从15个中选中5个算法：MARS、RC6、Rijndael、Serpent和Twofish
  - 2000年10月：Rijndael获胜，Vincent Rijmen 和Joan Daemen
  - 2001年11月：NIST公布新的数据加密标准AES

分组密码研究进展（续）

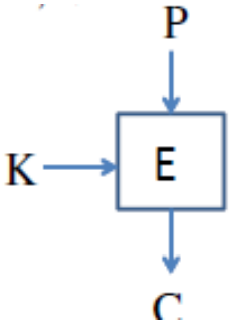
- 欧洲NESSIE工程
  - 2000年启动，共征集到17个分组密码算法
  - 2001年，进入第二轮的7个分组密码算法
  - 2003年，MSITY1、Camellia和SHACAL-2获胜
- 韩国标准：SEED和ARIA
- 我国：SMS4（SM4），用于无线局域网产品
- ISO/IET标准：2006年
  - 64位：TDEA、MISTY1和CAST-128
  - 128位：AES、Camellia、SEED

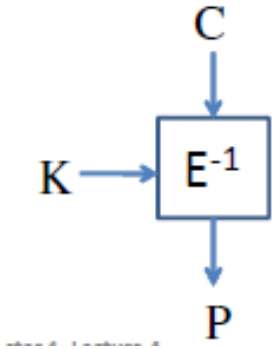
分组密码算法

- n-比特的明文分组加密成n-比特的密文分组
  - 分组长度n比特
    - DES： 64比特
    - AES： 128比特
  - 密钥长度
    - DES：56比特（不安全）
    - AES：128、192和256比特

加密：

$C = E_K(P)$



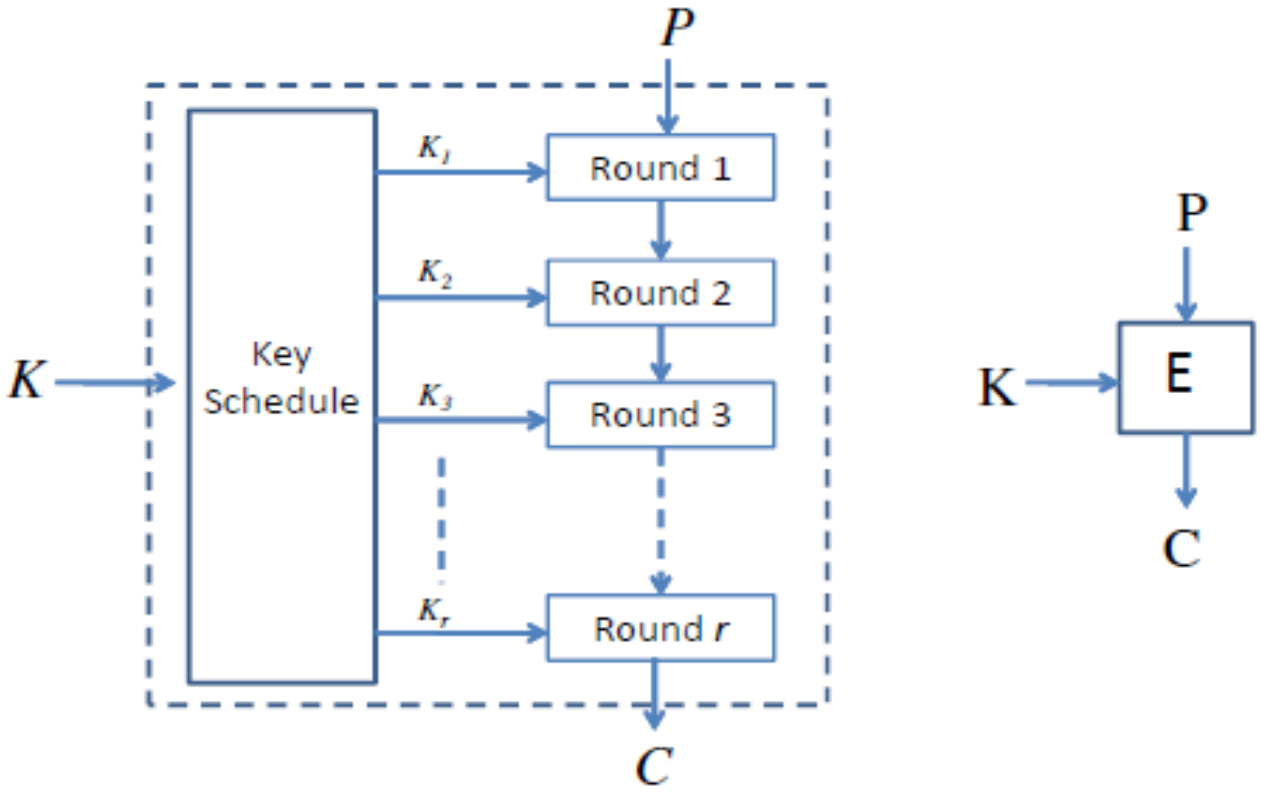


解密：

$P = E^{-1}_K(C)$

迭代结构

- 基于轮函数的迭代结构分组密码算法：便于设计、安全性评估和实现



轮函数

- 轮函数的设计策略
  - 混淆(Confusion): 非线性部件
    - 小的代换表 S-box
    - 乘法与异或
    - 加与异或
    - ……
  - 扩散(Diffusion)：让所有的比特互相影响
    - 线性变换
    - 置换
    - 移位、循环移位
    - ……..

轮函数（续）

- 设计轮函数的两种方案
  - Feistel Network
    - DES
  - Substitution-Permutation Network(SPN)
    - AES
  - 其他方案

密钥方案

- 密钥生成方案（Key Schedule)
  - 多种方法
  - 目前为止没有完美的方法
  - 基本准则：密钥的每个比特应该影响不同位置的很多轮的轮子密钥

DES

- 分组加密算法
- 64比特消息分组
- 56比特密钥
  - 另加 8 个冗余比特用于奇偶检验
- Feistel 结构
- 共16轮　＋　起始置换　＋　末置换

DES: Feistel Network

- Feistel Network
  - 由Horst Feistel发明
  - DES的设计者之一

Horst Feistel
1915-1990

DES: Feistel Network

- 加密过程

$(L_0, R_0) = P$ (L_i & R_i are the same size)

$L_1 = R_0$

$R_1 = L_0 \oplus F(K_1, R_0)$

…

$L_i = R_{i-1}$

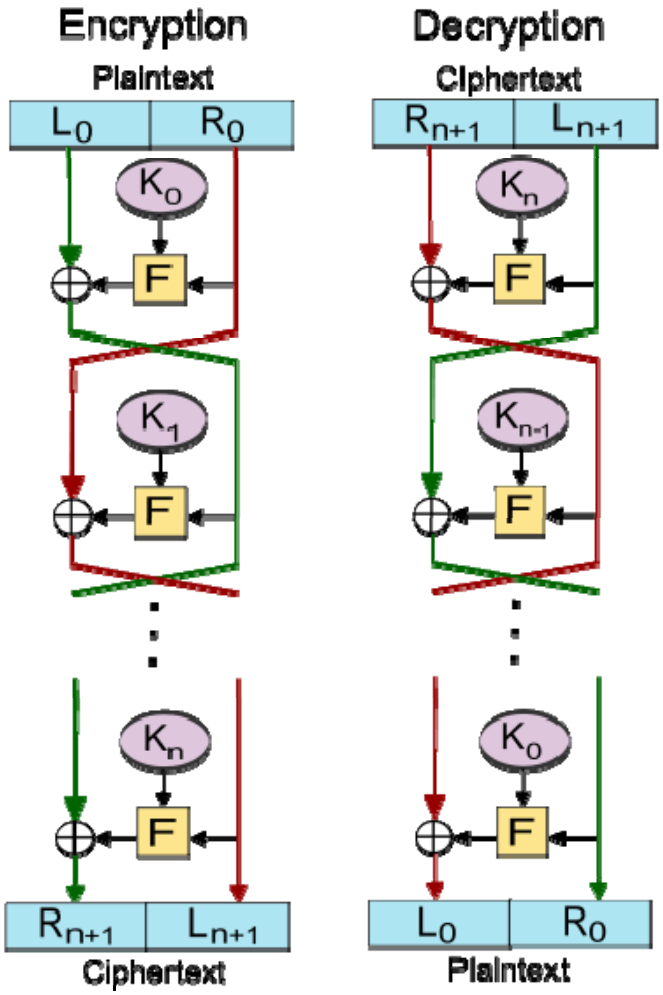
$R_i = L_{i-1} \oplus F(K_i, R_{i-1})$

…

$L_r = R_{r-1}$

$R_r = L_{r-1} \oplus F(K_r, R_{r-1})$

$C = (R_r, L_r)$



DES: Feistel Network

- 加解密过程

Encryption:

$(L_0, R_0) = P$

$L_1 = R_0$

$R_1 = F(K_1, L_0)$

…

$L_i = R_{i-1}$

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

…

$L_r = R_{r-1}$

$R_r = L_{r-1} \oplus F(K_r, R_{r-1})$

$C = (R_r, L_r)$

Decryption:

$(L_0, R_0) = C$

$L_1 = R_0$

$R_1 = F(K_r, L_0)$

…

$L_i = R_{i-1}$

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

…

$L_r = R_{r-1}$

$R_r = L_{r-1} \oplus F(K_1, R_{r-1})$

$P = (R_r, L_r)$

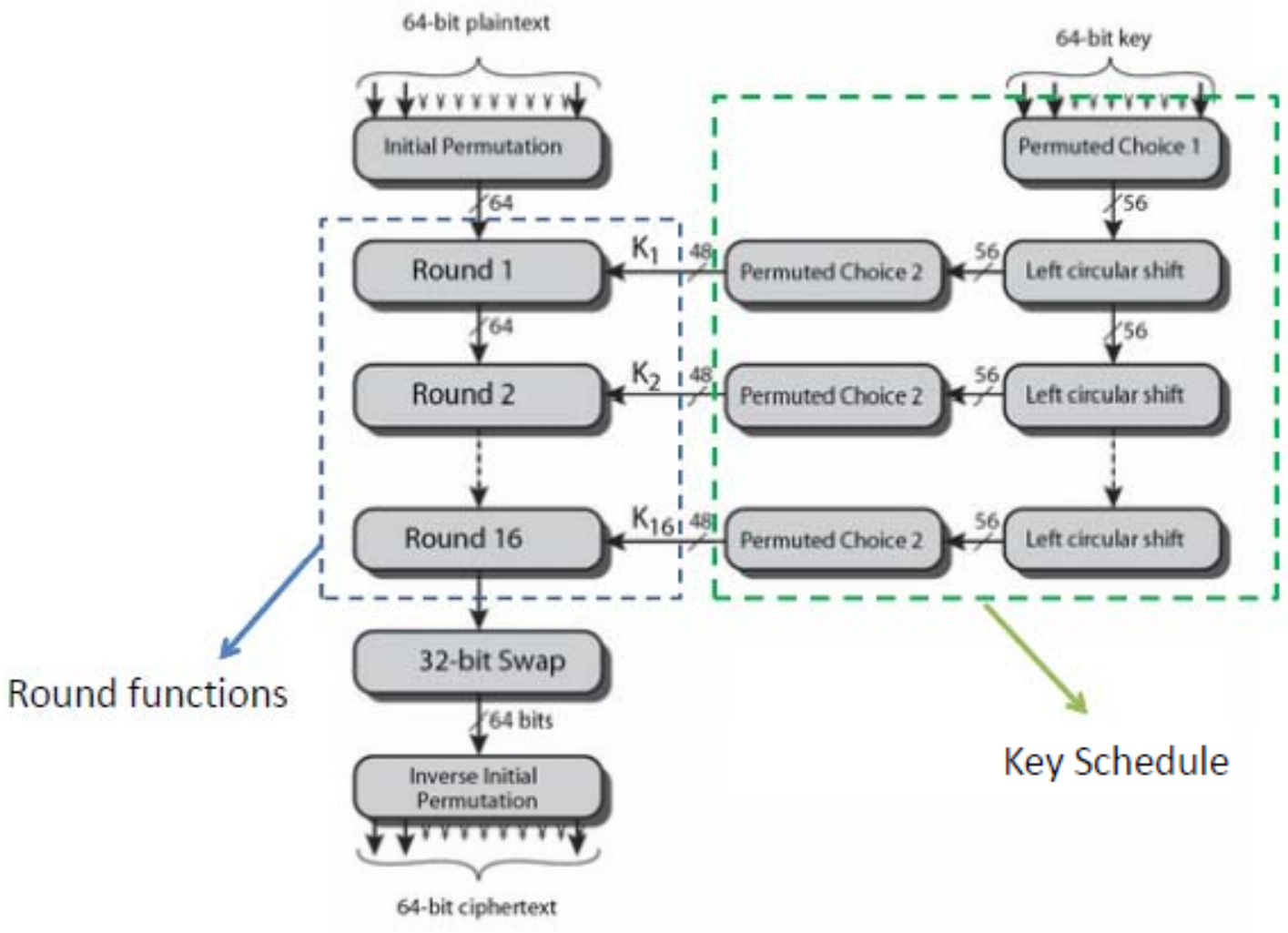
DES: Feistel Network

- Feistel结构的属性
  - 可逆性

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

  - 加解密使用通一套逻辑：除了密钥使用顺序不用
    - 加密：$K_1, K_2, K_3, \ldots, K_r$
    - 解密：$K_r, K_{r-1}, K_{r-2}, \ldots, K_1$

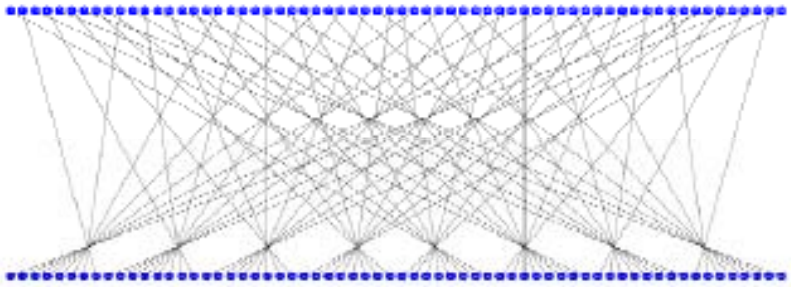
DES 总体结构

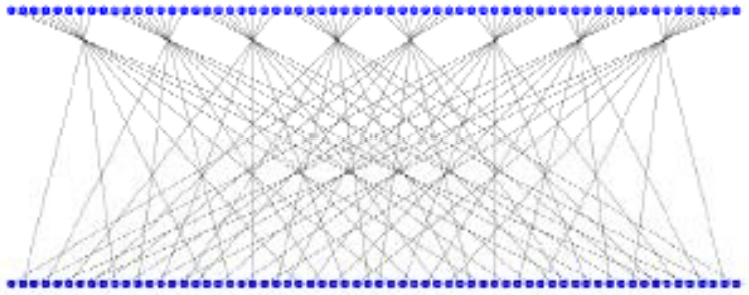

64-bit plaintext
64-bit key
Initial Permutation
Permuted Choice 1
64
56
Round 1
K1
48
Permuted Choice 2
56
Left circular shift
64
56
Round 2
K2
48
Permuted Choice 2
56
Left circular shift
Round 16
K16
48
Permuted Choice 2
56
Left circular shift
32-bit Swap
64 bits
Inverse Initial Permutation
64-bit ciphertext
Round functions
Key Schedule

起始置换和末置换

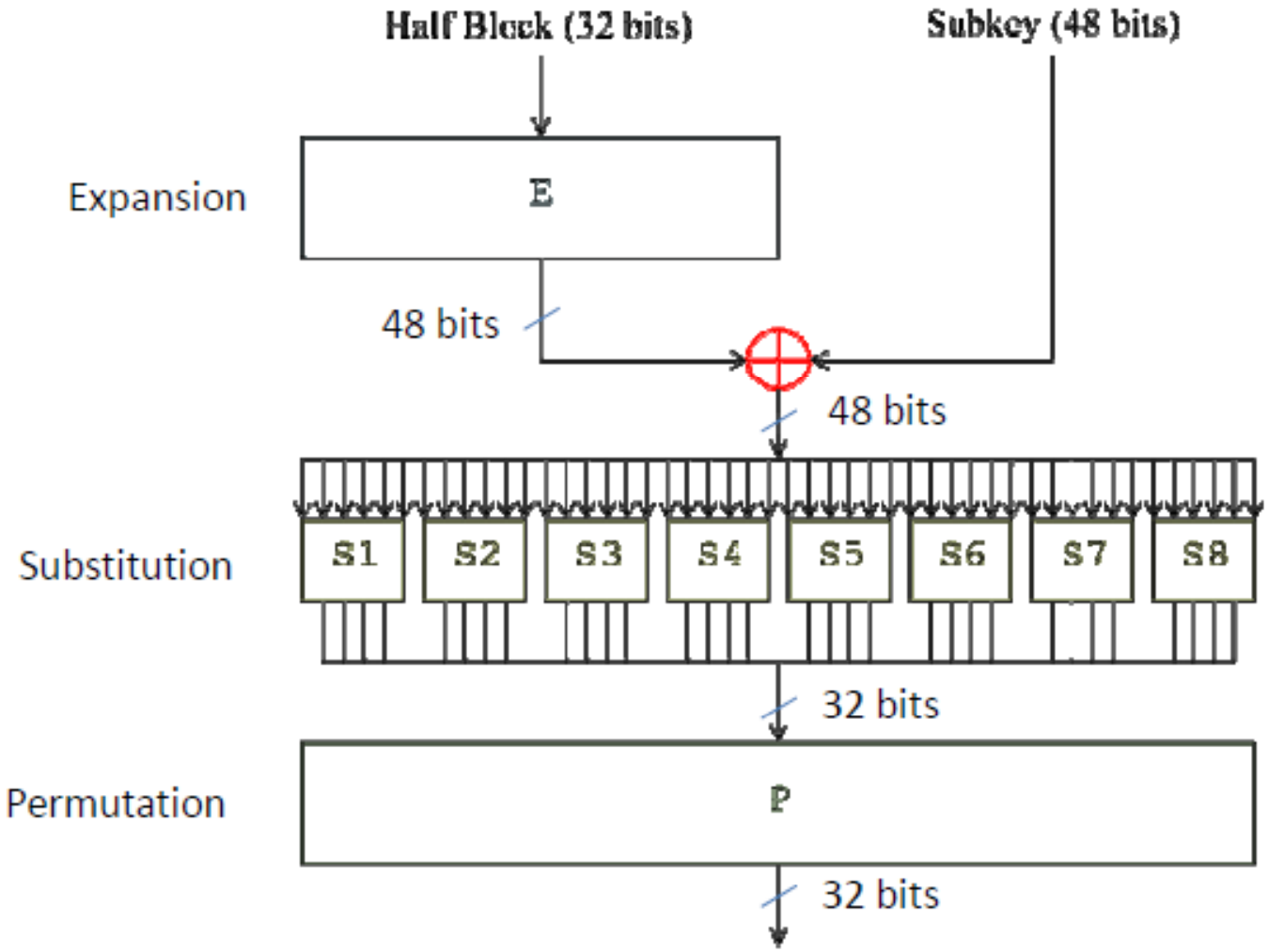
IP							
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

IP^{-1}							
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





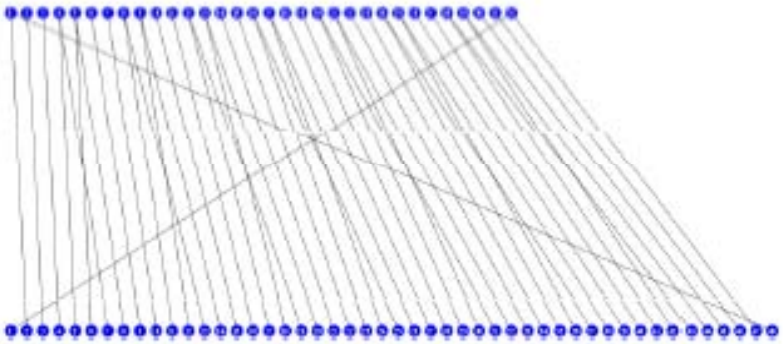
DES: F 函数

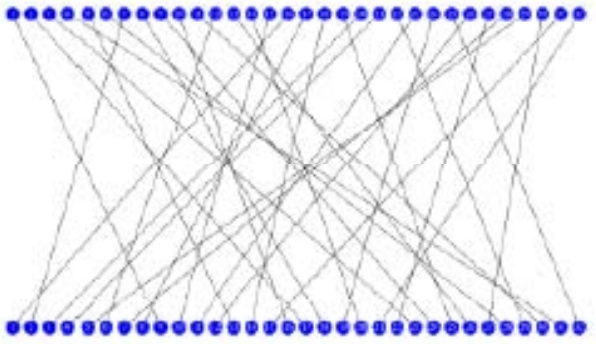

Half Block (32 bits)
Subkey (48 bits)
Expansion
E
48 bits
48 bits
Substitution
S1
S2
S3
S4
S5
S6
S7
S8
32 bits
Permutation
P
32 bits

DES: F函数中的扩展和置换

E					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25





DES: S-box

- 使用8个 S 盒： $S_1S_2\ldots\ldots S_8$

  每一个S盒S_i :$\{0,1\}^6$->$\{0,1\}^4$

  用4乘16的矩阵描述

- 6比特的串$B=b_1b_2b_3b_4b_5b_6$
  - b_1b_6决定矩阵的行，$b_2b_3b_4b_5$决定矩阵的列
  - $C_j=S_j(B_j)$

S-box

S1=
{14, 4, 13, 1, 2, 15, 11, 8, 3, 10, 6, 12, 5, 9, 0, 7,
0, 15, 7, 4, 14, 2, 13, 1, 10, 6, 12, 11, 9, 5, 3, 8,
4, 1, 14, 8, 13, 6, 2, 11, 15, 12, 9, 7, 3, 10, 5, 0,
15, 12, 8, 2, 4, 9, 1, 7, 5, 11, 3, 14, 10, 0, 6, 13};

S2={
15, 1, 8, 14, 6, 11, 3, 4, 9, 7, 2, 13, 12, 0, 5, 10,
3, 13, 4, 7, 15, 2, 8, 14, 12, 0, 1, 10, 6, 9, 11, 5,
0, 14, 7, 11, 10, 4, 13, 1, 5, 8, 12, 6, 9, 3, 2, 15,
13, 8, 10, 1, 3, 15, 4, 2, 11, 6, 7, 12, 0, 5, 14, 9
};

S3={
10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,
13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,
13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,
1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12
};

S4={
7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,
13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,
10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,
3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14
};

S5={
2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,
14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6,
4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14,
11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3

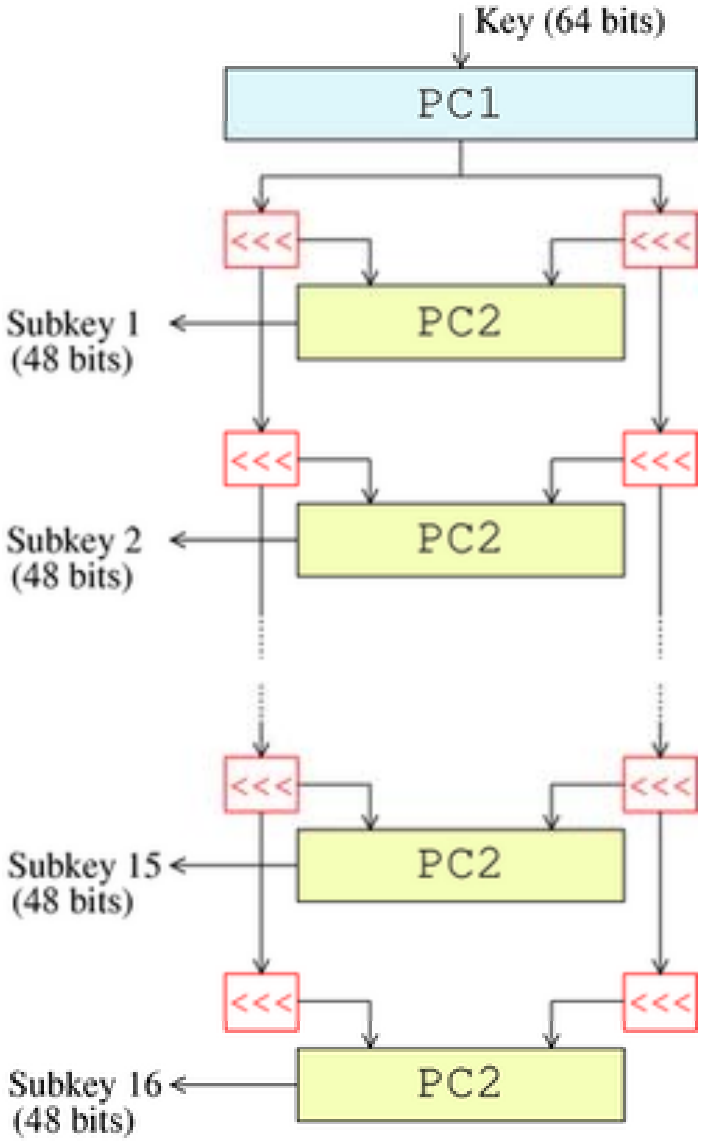
};
S6={
12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11,
10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8,
9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6,
4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13
};

S7={
4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1,
13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6,
1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,
6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12
};

S8={
13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,
1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,
7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,
2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11
};

DES密钥生成算法

- 当i=1,2,9,16时，移位数位1
- 当i=3,4,5,6,7,8,10,11,12,13,14,15时，移位数为2
- PC1为一56比特的置换
- PC2从56比特中选择48比特

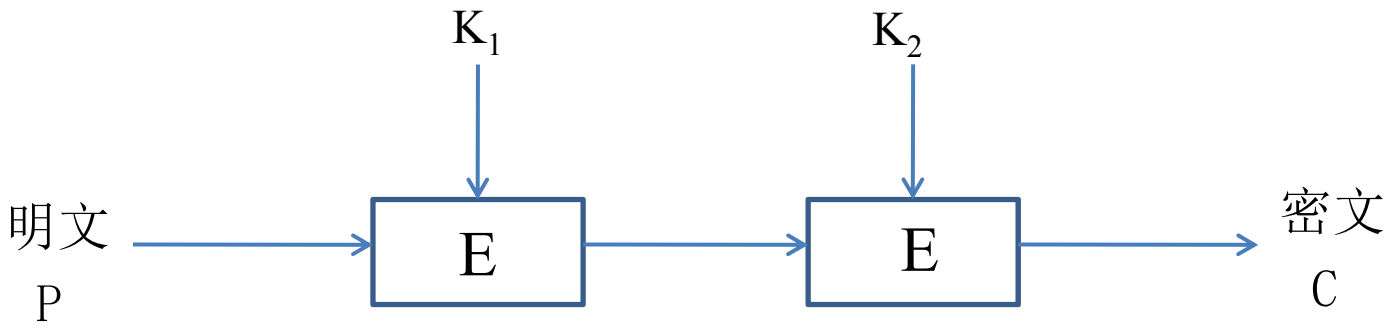


多重加密

- DES：56比特密钥长度
  - 在今天的计算能力下，不再安全
  - 如何增加密钥长度?
- 多重加密
  - Double-DES
  - Triple-DES

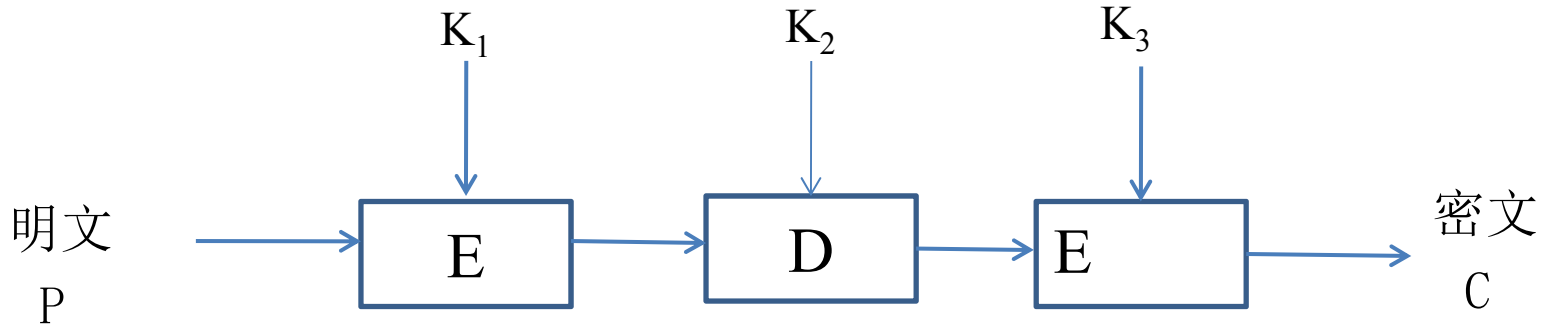
多重加密

- Double-DES
  - 密钥长度112比特
  - 加密： $C=E_{k2}(E_{k1}(P))$
  - 不能够抵抗中间相遇攻击：时间2^{57}, 存储2^{56}
    - 构造两个集合$I=\{E_{k1}(P\}\}$, $J=\{D_{k2}(C)\}$, 利用生日攻击寻找碰撞， 时间、空间复杂度$2^{n/2}$



多重加密

- Triple-DES
  - 加密： $C=E_{k3}(D_{k2}(E_{k1}(P)))$
  - 解密： $P=D_{k1}(E_{k2}(D_{k3}(C)))$
- 密钥选择
  - 3-key: k1,k2,k3 互相独立，168比特
  - 2-key: k1,k2独立，k3=k1, 112比特



分组密码的工作模式

- 工作模式

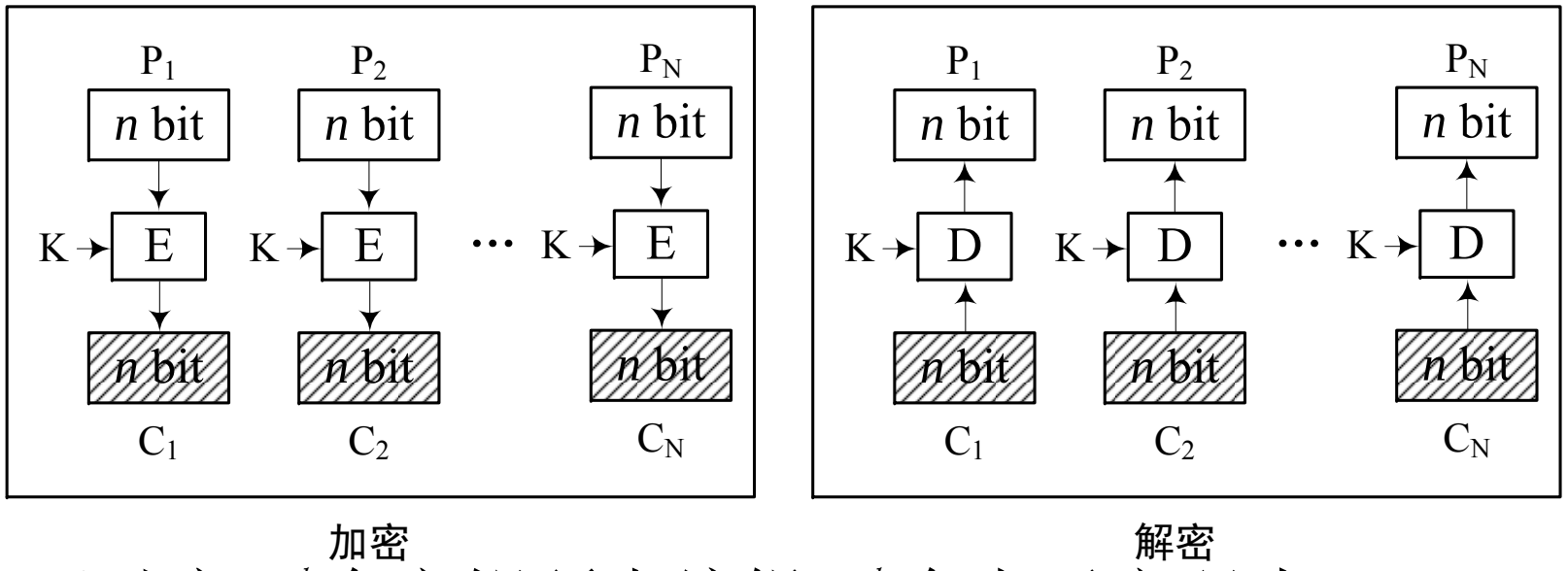
– 分组密码算法在实际中的使用方式称为工作模式。
– NIST在FIPS-PUB-81中定义了4种工作模式, 1980。
– 后又在NIST Special Publication 800－38A中增加一个，现有5个工作模式, 2001。

- 五种工作模式

– 电子密码本模式(Electronic Codebook Mode: ECB)
– 密码分组链接模式(Cipher Block Chaining Mode, CBC)
– 密码反馈模式(Cipher Feedback Block (Mode, CFB)
– 输出反馈模式(Output Feedback Mode, OFB)
– 计数器模式(Counter Mode, CTR)

电子密码本(ECB)模式

- ECB模式是将明文的N个分组独立地使用同一密钥K加密和解密，如下图所示。



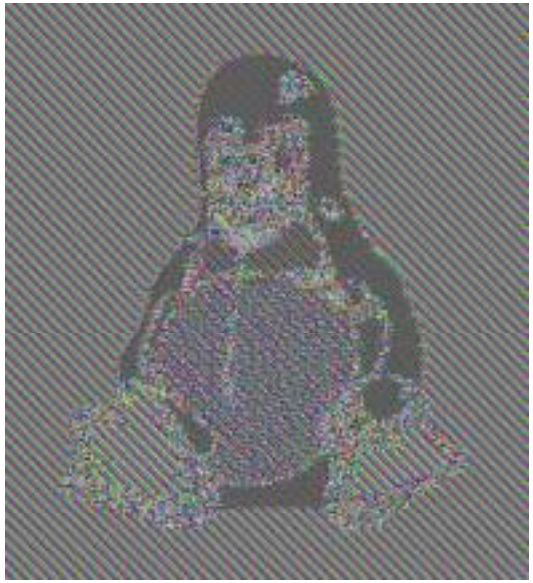
- 可对应 2^k个密钥预先编辑 2^k个电子密码本;
- 每个密码本有 2^n个条目——（明文分组,密文分组);
- 所以称为电子密码本。不过当k和n很大时，密码本会过于庞大而无法预先编辑和保存。

ECB模式的优、缺点和应用

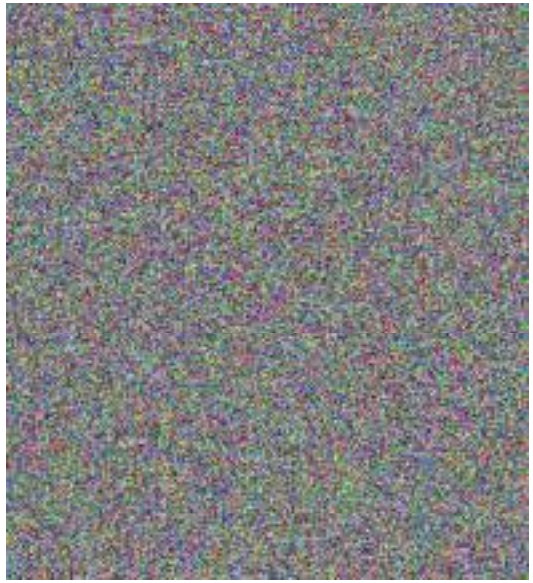
- 优点
  - 实现简单;
  - 不同明文分组的加密可并行实施,尤其是硬件实现时速度很快。
- 缺点
  - 不同明文分组之间的加密独立进行,故保留了单表代替缺点,造成相同明文分组对应相同密文分组,因而不能隐藏明文分组的统计规律和结构规律。
- 典型应用
  - 用于随机数的加密保护;
  - 用于单分组明文的加密。

ECB模式缺点

原始图片
bmp格式



使用ECB
模式加密

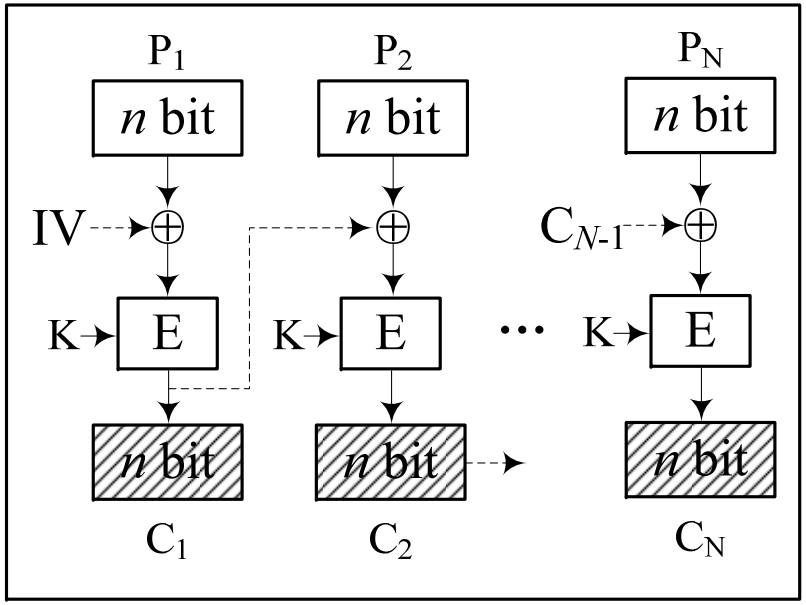


使用其他四
种模式加密

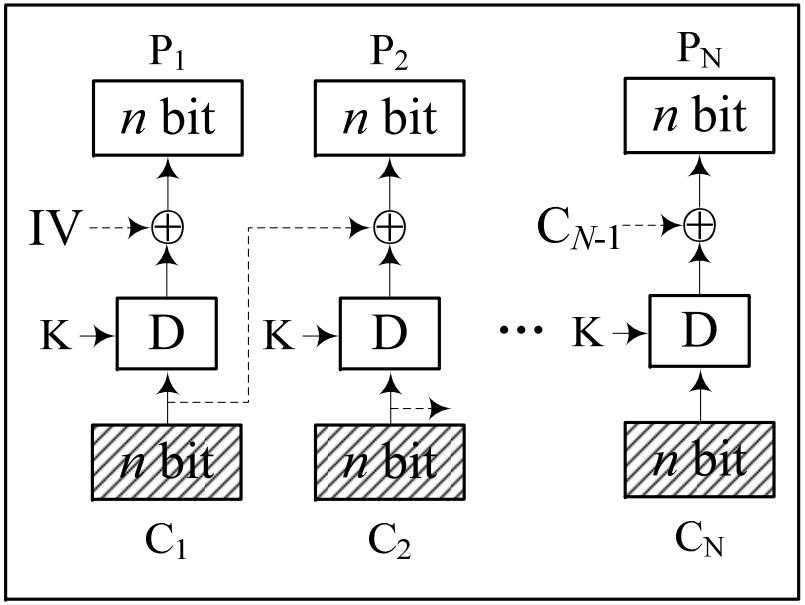
密码分组链接(CBC)模式

- 克服ECB的安全性缺陷：同一个明文分组重复出现时产生不同的密文分组。
- 简单的想法：使输出不仅与当前输入有关，而且与以前输入和输出有关——密码分组链接模式。
- CBC模式中每个明文分组在加密之前都要与以前的密文分组进行异或。第一个分组之前没有密文，故要用到一个伪分组IV。
  - IV不要求保密
  - IV必须是不可预测的，而且要保证完整性
- 在发送方，异或要在加密之前完成；在接收方，解密要在异或之前完成。

密码分组链接(CBC)模式



加密



解密

加密：

$$C_0 = IV$$
$$C_i = E_k(P_i \oplus C_{i\text{-}1})$$

解密：

$$C_0 = IV$$
$$P_i = D_k(C_i) \oplus C_{i\text{-}1}$$

CBC模式的特点

- 优点
  - 明文块的统计特性得到了隐蔽
    - CBC模式中各密文块不仅与当前明文块有关，而且还与以前的明文块及初始化向量有关，从而使明文的统计规律在密文中得到了较好的隐蔽。
  - 具有有限的(两步)错误传播特性；
  - 具有自同步功能
    - 密文出现丢块和错块不影响后续密文块的解密。若从第t块起密文块正确，则第t+1个明文块就能正确求出。
- 缺点
  - 加密不能并行处理（解密可以）
  - 消息是分组长度的整数倍
    - 可用ciphertext stealing 技术解决这个问题

CBC模式的应用

CBC模式是应用最广，影响最大的一个工作模式

(1) 数据加密；

(2) 完整性认证和身份认证；

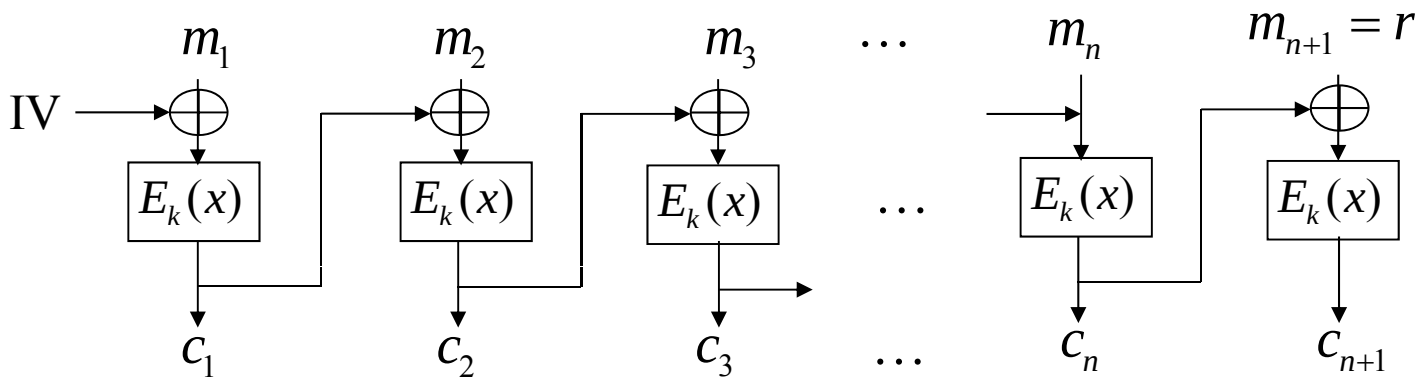
完整性认证是一个“用户”检验它收到的文件是否遭到第三方有意或无意的篡改。因此完整性认证：

- 不需检验文件的制造者是否造假；
- 文件的制造者和检验者利益一致，不需互相欺骗和抵赖；
- 目的是检查文件在传输和存储中是否遭到有意或无意的篡改。

CBC模式的应用——完整性认证

具体实现：

1. 文件的制造者和检验者共享一个密钥；
2. 文件的明文m产生一个奇偶校验码分组 $r = m_{n+1} = m_1 \oplus \cdots \oplus m_n$
3. 采用分组密码的CBC模式，对附带校验码的明文(m, r)加密，得到的最后一个密文分组C_{n+1}就是认证码；
4. 仅需认证而不需加密时，传送(m, C_{n+1})，此时也可仅保留C_{n+1}的 t 个比特作为认证码；既需认证又需加密时，传送(C, C_{n+1})。



CBC模式的应用——完整性认证

检验方法：

- 仅需认证不需加密时，验证者仅收到明文m和认证码C_{n+1}：
  - 首先产生明文m的校验码；
  - 然后利用共享密钥使用CBC模式对(m, r)加密，将得到的最后一个密文分组与接收到的认证码C_{n+1}比较，二者一致时判定接收的明文无错；二者不一致时判定明文出错。
- 既需认证又需加密时，验证者收到密文C和认证码C_{n+1}：
  - 首先利用共享密钥使用CBC模式对C解密得到明文m，之后同上。
  - …

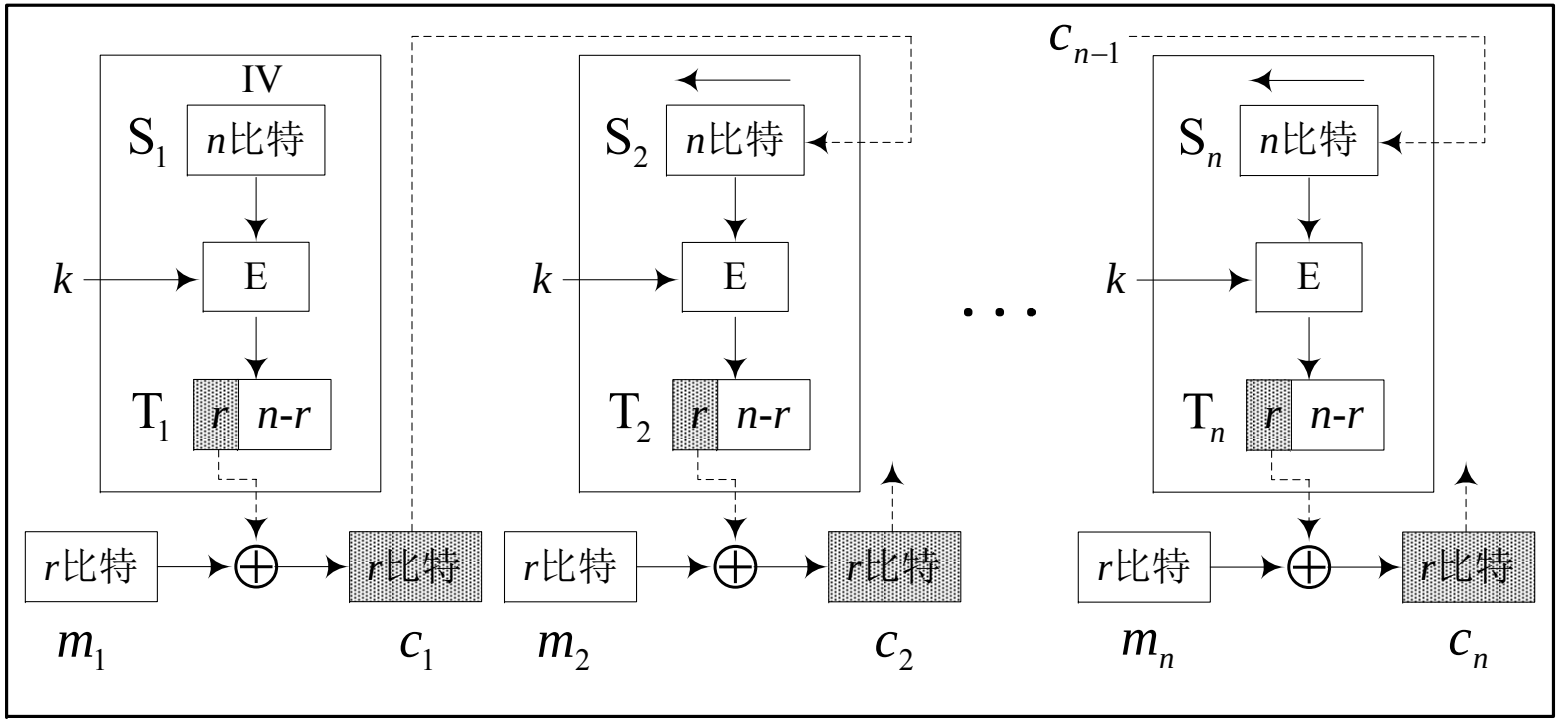
密码反馈(CFB)模式

- EBC和CBC加密，分组大小n由基本密码确定，如若用DES则n=64，用AES则n=128。
- 若待加密消息需按字符、字节或比特处理时，可采用CFB模式。其中基本密码分组大小为n，但是明文或密文分组大小为r，且$1 \leq r \leq n$，称为 r 比特CFB模式。
  - 如：1比特 CFB模式，8比特CFB模式，64比特CFB模式，128比特CFB模式
- 需要IV作为第一个输入分组
  - IV：不需要保密，但要不可预测
- 适用于每次处理 r比特明文块的特定需求的加密情形，能灵活适应数据各格式的需要。如数据库加密要求加密时不能改变明文的字节长度，这时就要以明文字节为单位进行加密。

密码反馈(CFB)模式

- r比特CFB模式中的加密如下图

IV：初始向量（S_1）　　S_i：移位寄存器（左移r比特）　　T_i：临时寄存器

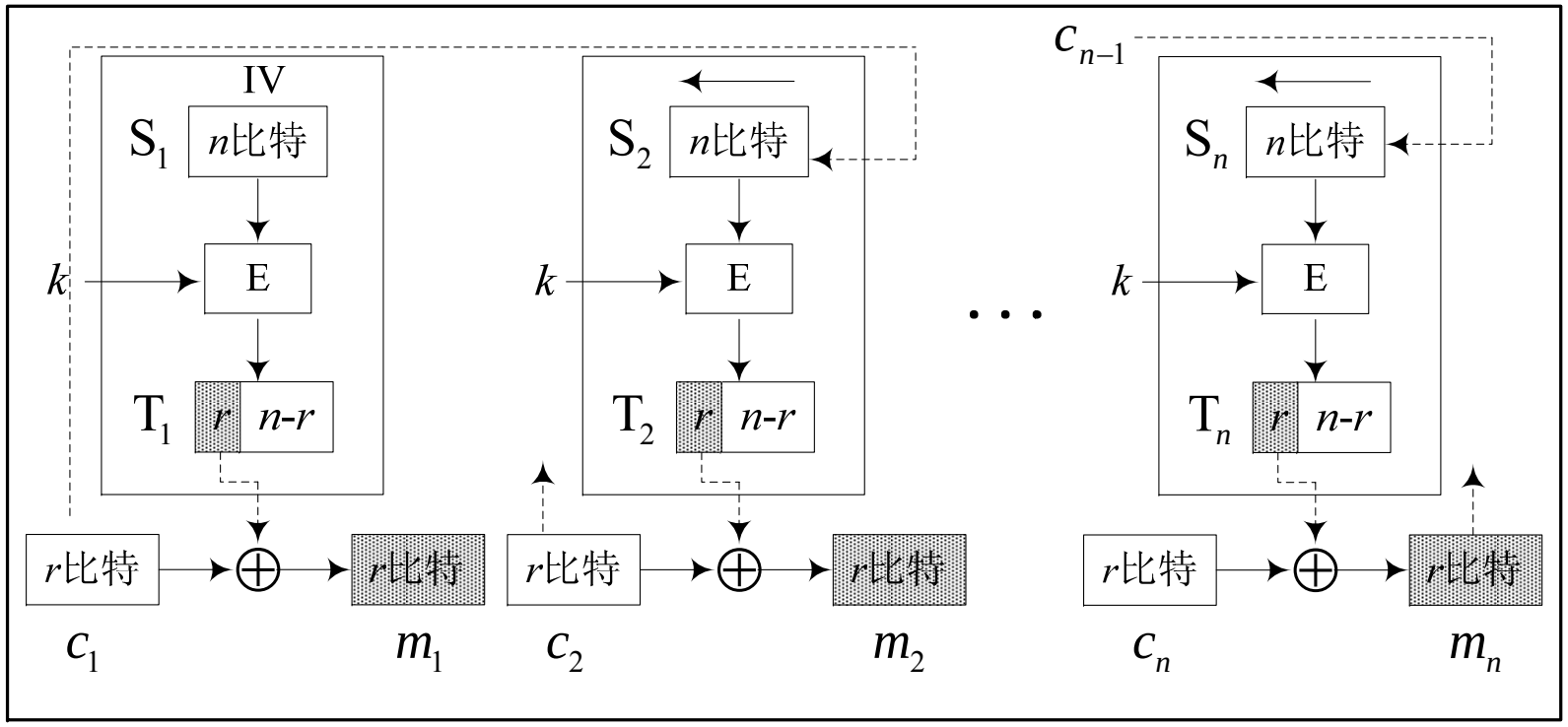


加密

密码反馈(CFB)模式

r比特CFB模式中的解密如下图

IV：初始向量（S_1）　　S_i：移位寄存器（左移r比特）　　T_i：临时寄存器



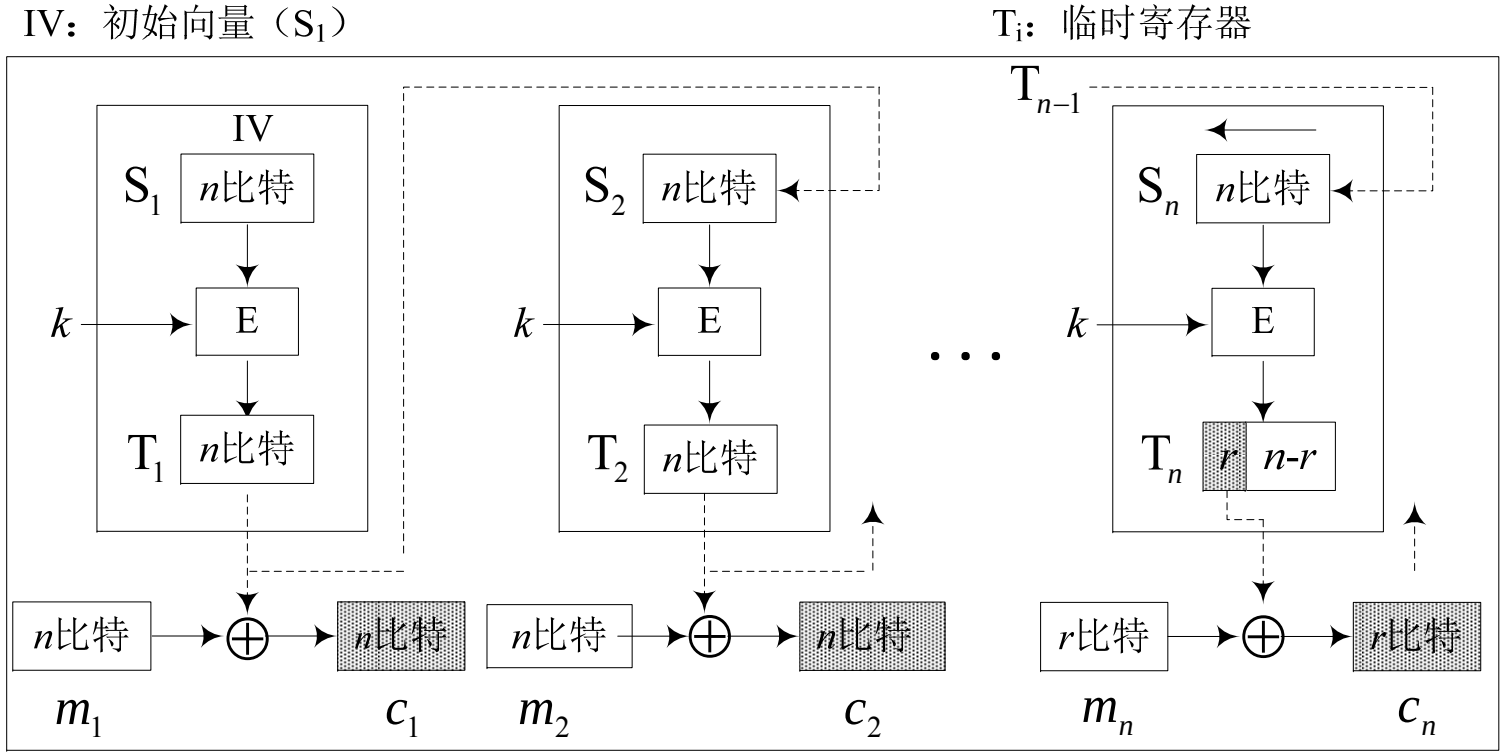
解密

CFB模式的优、缺点及应用

- 优点：
  - 适应用户不同的数据格式的需求；
  - 具有有限步的错误传播(n/r步)，可用于认证；
  - 具有自同步功能
- 缺点：
  - 加密效率低
  - 加密不能并行(解密可以并行）
  - 明文或IV的一个比特的变化影响密文所有比特
- 应用：
  - 该模式适应于数据库加密、无线通信加密等对数据格式有特殊要求

输出反馈(OFB)模式

- OFB模式在结构上类似于CFB模式，但反馈内容是分组密码输出的密钥流而不是密文。
- 因此，OFB模式的密文每一个分组都独立于先前的分组，避免了错误传播。



加密

OFB模式的优、缺点及应用

优点：

- 这是将分组密码当作同步序列密码使用的一种方式
- 不具有错误传播特性。密文的1比特错误只导致明文的1比特错误。只要密文在传输过程中不丢信号，即使信道不好，也能将明文的大部分信号正常恢复。

缺点：

- 不能实现报文的完整性认证
- IV无需保密，但是对每个消息必须选择不同的IV
- 不具有自同步能力
- 密钥序列的周期可能有短周期现象

适用范围：

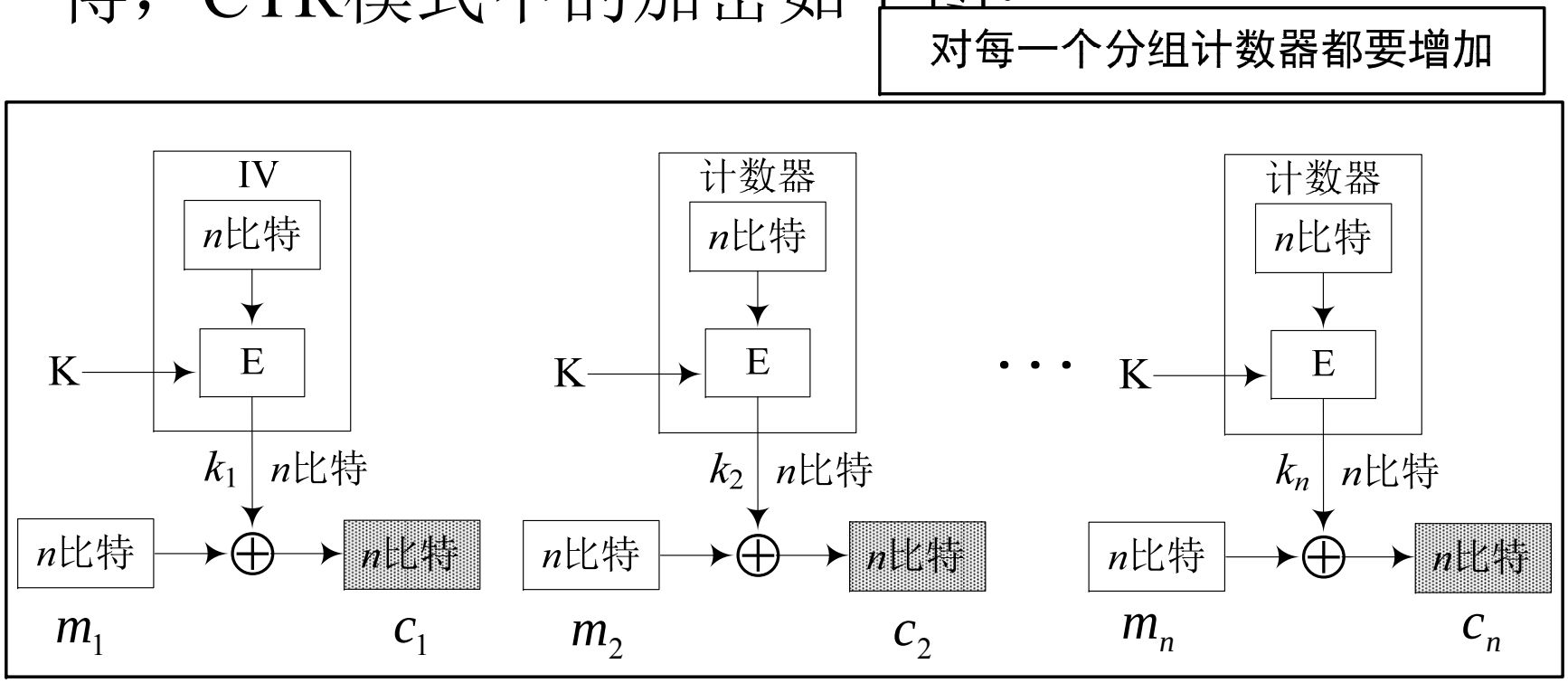
- 明文的冗余度特别大，信道不好但不易丢信号，明文信号出些错误也不影响效果的情形。如图象加密、语音加密等。

IV的产生

- CBC，CFB
  - IV不可预测，NIST推荐两种方法
    - 用加密算法在相同的密钥之下加密一个nonce
    - 用FIPS支持的随机数发生器产生一个随机的明文分组
- OFB：对IV敏感，每个明文必须对应不同的IV
  - 可以使用一个计数器
  - 一定不能使用另外的一个消息在给定的密钥下加密

计数器（CTR）模式

- Diffe和Hellman设计：明密文分组与基本密码大小相同，密钥流中的伪随机数运用计数器获得，CTR模式中的加密如下图.



加密

计数器（CTR）模式

- 明文分组和密文分组的关系：

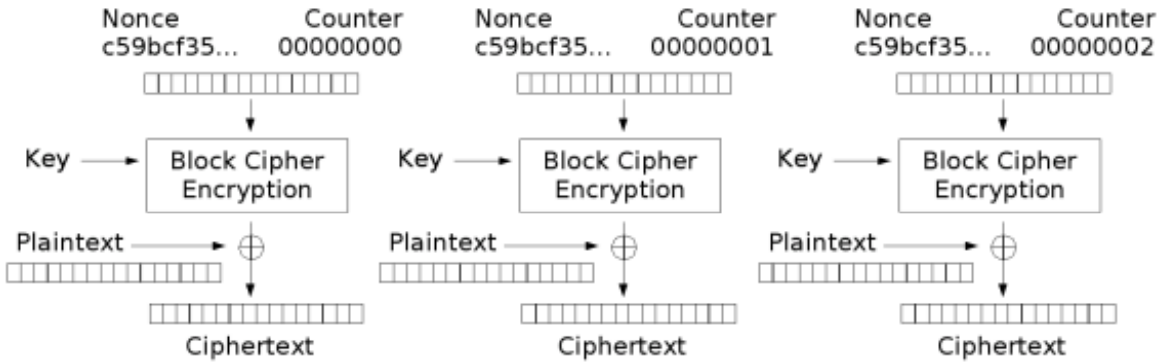
加密：$C_i = P_i \oplus E_{k_i}$（计数器）　　解密：$P_i = C_i \oplus E_{k_i}$（计数器）

- CTR与OFB、ECB比较：
  - CTR模式与OFB模式一样，创建了独立于以前密文分组的密钥流，但是不运用反馈；
  - CTR模式与ECB模式一样，创建了彼此相互独立的n比特的密文分组，他们只依赖于计数器的值。

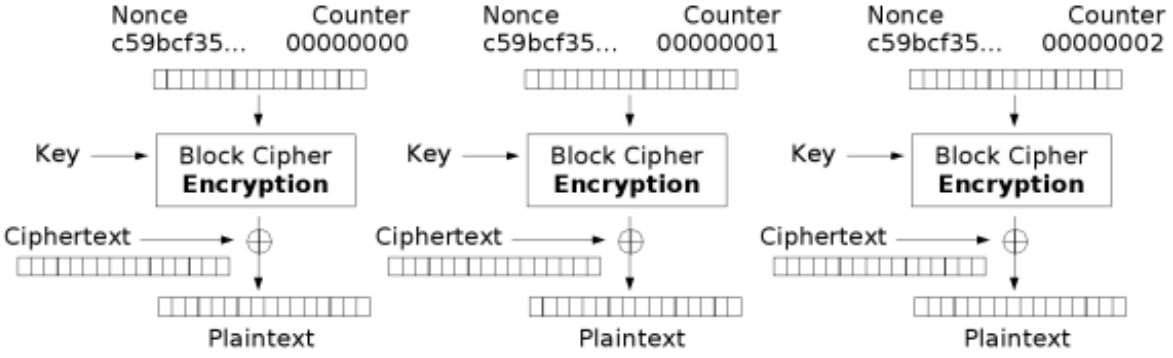
计数器分组的产生

- 方法一：所有的明文依次加密
  - 假设明文分组长度为*n*比特，则任意选择一个长度为n比特的计数器ctr, 则第*i*个分组所用的计数器为 T_i=ctr+*i*-1 mod 2^n
- 方法二：不同的明文间可以并行加密
  - 由两部分够成：m比特的IV和m比特计数器ctr
  - IV：不同的明文IV不同，相同的明文IV相同
  - ctr: 从0计数，每个明文分组增加1

AES-CTR



Counter (CTR) mode encryption



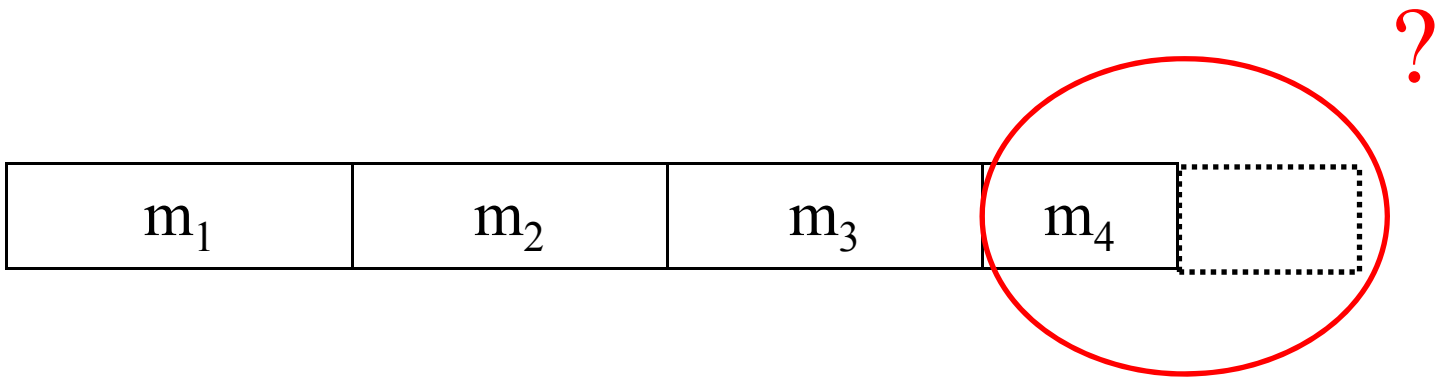
Counter (CTR) mode decryption

五种操作模式对比

操作模式	加密	结果类型	数据单位大小
ECB	每一个n比特的分组都要用相同的密码密钥独立进行加密	分组密码	n
CBC	与EBC相同，不过每一个分组都要用前面的密文进行异或	分组密码	n
CFB	每一个r比特的分组都要用一个r比特的密钥进行异或，这个密钥是前面密文的一部分	序列密码	$r \leq n$
OFB	与CFB相同，不过移位寄存器要用前面的r比特密钥更新	序列密码	$r \leq n$
CTR	与OFB相同，不过用的是计数器而不是移位寄存器	序列密码	n

如何加密一个部分分组？

- 若一个明文长度不是一个分组密码分组长度的倍数
  - 最后一个分组称为partial block



明文填充（padding）

- ECB，CBC模式需要填充
  - 填充后密文长度大于明文长度
- CFB，OFB，CTR模式不需要填充
- 填充方法
  - 比特填充
    - 在消息后面先填充一比特的“1”，然后填充若干比特的“0”，使其成为一个完整的消息分组
    - 为了使填充没有二义性，针对于这种填充方法，即使明文是一个完整的分组，也要填充；或者是使用一个明文长度指示器

... | 1011 1001 1101 0100 0010 011**1 0000 0000** |

明文填充(padding)

- 字节填充
  - PKCS7: 填充“需要填充部分”的字节数，如

... | DD DD DD DD DD DD DD DD | DD DD DD DD **04 04 04 04**

  - ANSI X.923：填充“0”字节，最后一个字节填充padding的长度,如

... | DD DD DD DD DD DD DD DD | DD DD DD DD 00 00 00 04 |

  - ISO 10126：填充若干个随机字节，最后一个字节填充padding的长度

... | DD DD DD DD DD DD DD DD | DD DD DD DD 81 A6 23 04 |

  - ISO/IEC 7816-4：与比特填充相同

... | DD DD DD DD DD DD DD DD | DD DD DD DD **80 00 00 00** |

  - Zero填充，空格填充

Ciphertext stealing

- Ciphertext stealing技术
  - 取得
    - 密文长度=明文长度
  - ECB ciphertext stealing
    - 明文长度必须大于1个分组
    - 否则，只能使用填充方法
  - CBC ciphertext stealing
    - 明文分组不必要大于1
    - 当明文分组小于1时，从C_0(IV) stealing

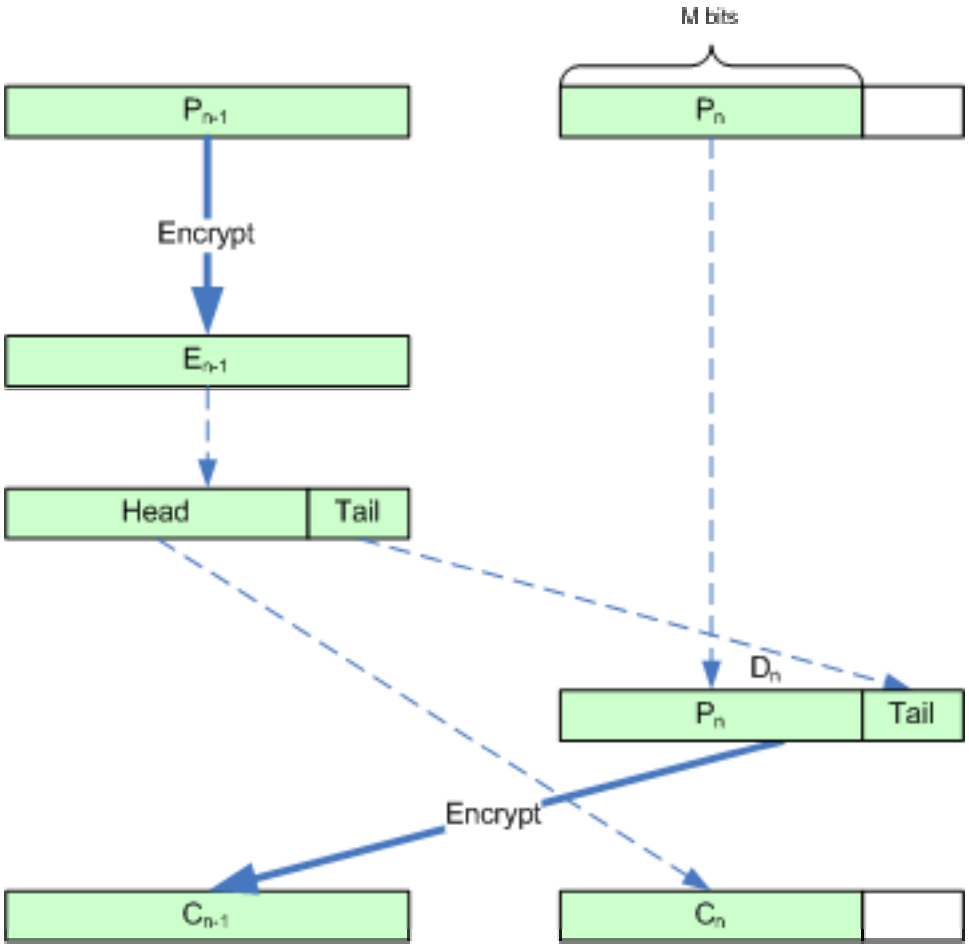
ECB ciphertext stealing

加密

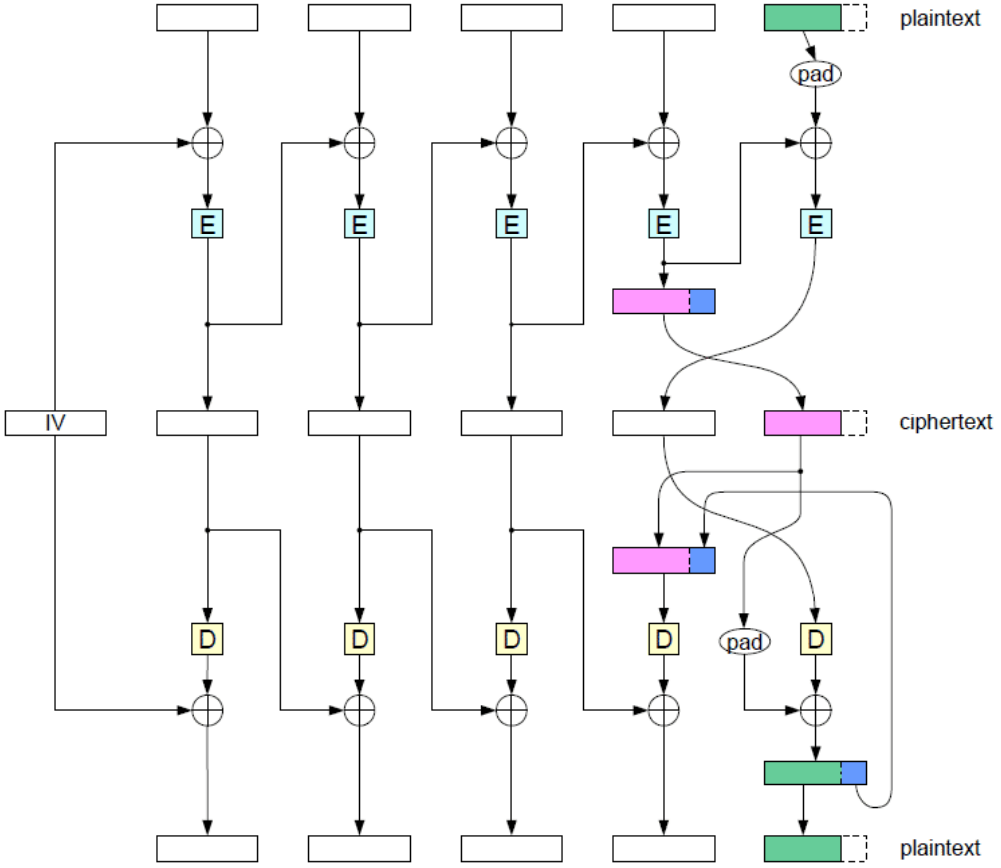
1. $E_{n-1} = Encrypt(k, P_{n-1})$
2. $C_n = head(E_{n-1}, M)$
3. $D_n = P_n \| Tail(E_{n-1}, B-M)$
4. $C_{n-1} = Encrypt(k, D_n)$

解密

1. $D_n = Decrypt(k, C_{n-1})$
2. $\mathrm{E}_{n-1} = C_n \| Tail(D_n, B-M)$
3. $P_n = Head(D_n, M)$
4. $P_{n-1} = Decrypt(k, E_{n-1})$



CBC ciphertext stealing

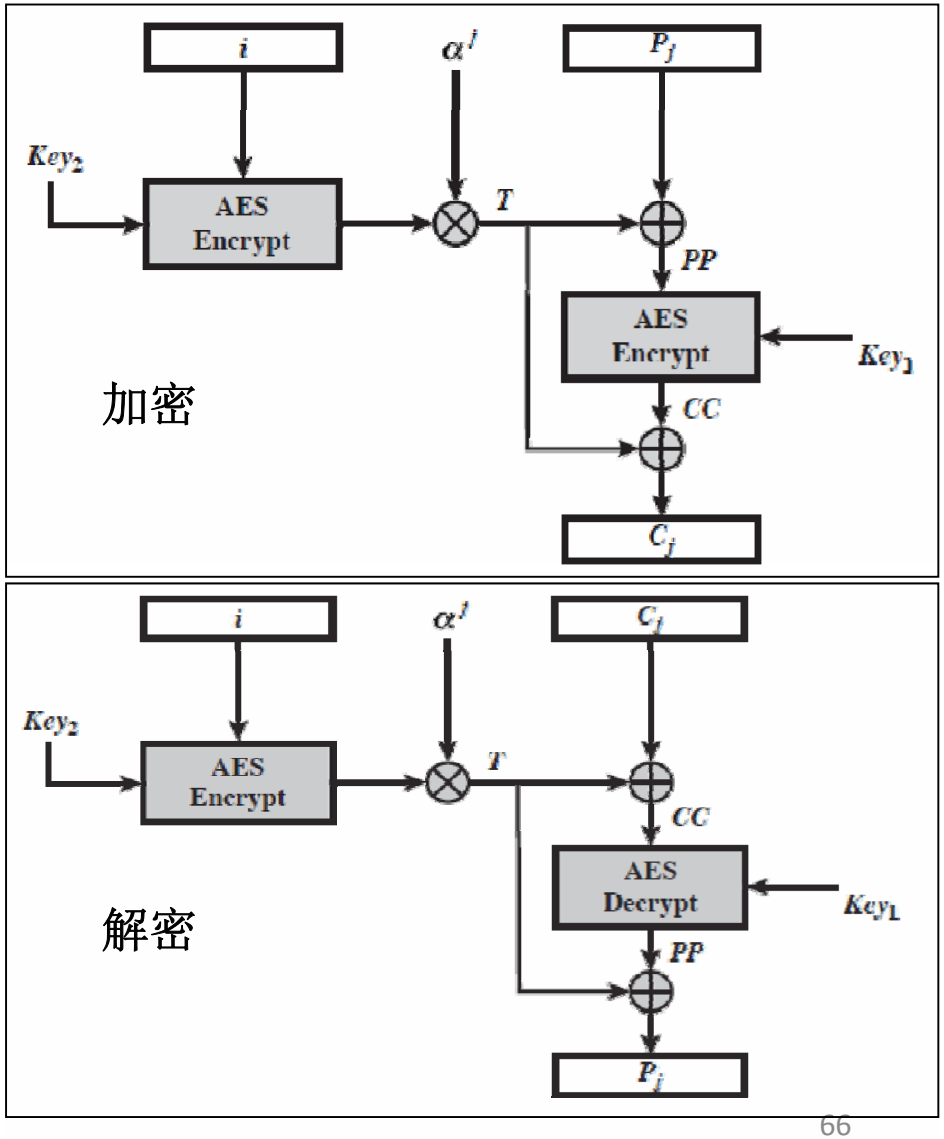


存储加密

- 对基于扇区的磁盘数据进行加密要求
  - 加密后密文长度与明文长度相同
  - 数据访问和加密为独立的、固定长度的数据块(扇区）
  - 明文分组的位置是仅有的可用的元数据（描述数据的数据）
  - 不同位置的相同的数据被加密成不同的密文
- XTS-AES被推荐为存储加密标准
  - IEEE std 1619-2007； NIST SP 800-38E, 2010
- 该标准被广泛应用
  - BestCrypt, dm-crypt, FreeOTFE, TrueCrypt, DiskCryptor, FreeBSD's geli, OpenBSD softraid disk encryption software, Mac OS X Lion's FileVault， in hardware-based media encryption devices by the SPYRUS Hydra PC Digital Attache and The Kingston Data Traveler 5000

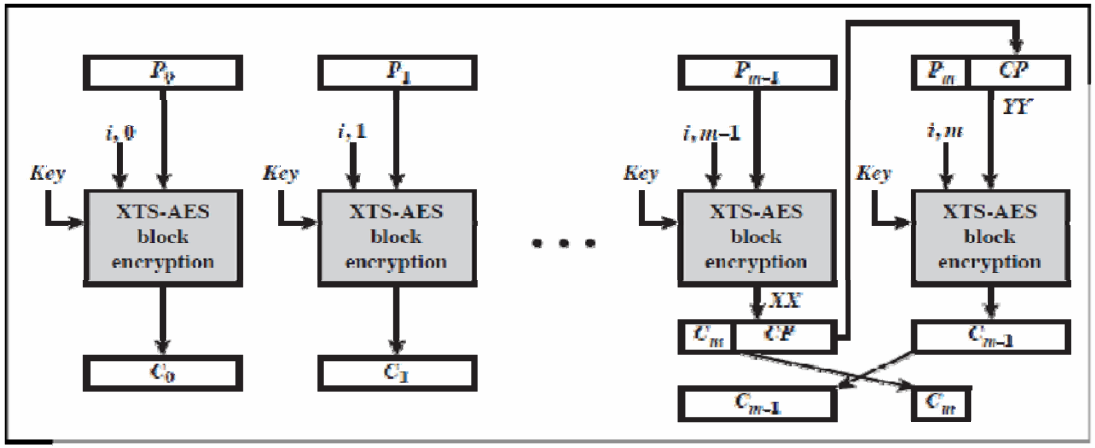
XTS-AES磁盘加密模式

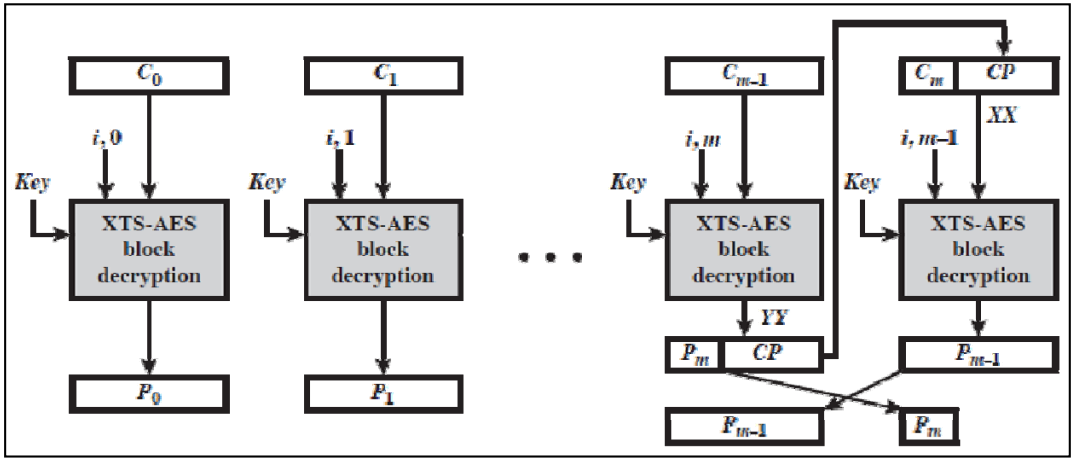
- 单一明文分组的加密
- 参数
  - key=(key_1||key_2), 256比特或512比特
  - P_j:明文的第j个分组
  - j: 在一个数据单元中明文分组的序号
  - i: 128比特的tweak值(数据块的磁盘地址）
  - α：GF(2128)中的一个生成元, 对应多项式 *x*



XTS-AES磁盘加密模式

- 对一个扇区数据的加密
  - 每个明文分组单独加密
  - 若最后一个明文分组小于128比特，则使用ciphertext stealing 技术





XTS-AES磁盘加密

- $T \bullet \alpha$的计算： GF(2^{128})中模多项式 $x^{128}+x^7+x^2+x+1$ 乘法
- 128比特的T表示为T[15],T[14]….,T[0],则 $T \bullet \alpha$ 计算如下

$$cin = 0;$$
$$for(j = 0; j < 16; j++)$$
$$\{$$
$$Cout = (T[j] >> 7) \,\&\, 1; \text{ //取一个字节的最高比特}$$
$$T[j] = ((T[j] << 1) + cin) \,\&\, 0xFF; \text{ //循环左移1比特}$$
$$Cin = Cout; \text{ //低位字节的最高位}$$
$$\}$$
$$if\,(Cout) \text{ //若最高比特是1}$$
$$T[0] = T[0] \wedge 0x87$$


谢谢！