



第 2 章：分组密码体制

赵俊舟

junzhou.zhao@xjtu.edu.cn

2025 年 3 月 18 日

目录

- 1 分组密码的基本原理
- 2 数据加密标准
- 3 DES 的安全性
- 4 数论基础
- 5 高级数据加密标准
- 6 多重加密
- 7 分组密码的工作模式

目录

1 分组密码的基本原理

- 流密码与分组密码
- 理想分组密码
- Feistel 分组密码

2 数据加密标准

3 DES 的安全性

4 数论基础

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

目录

1 分组密码的基本原理

- 流密码与分组密码
- 理想分组密码
- Feistel 分组密码

2 数据加密标准

3 DES 的安全性

4 数论基础

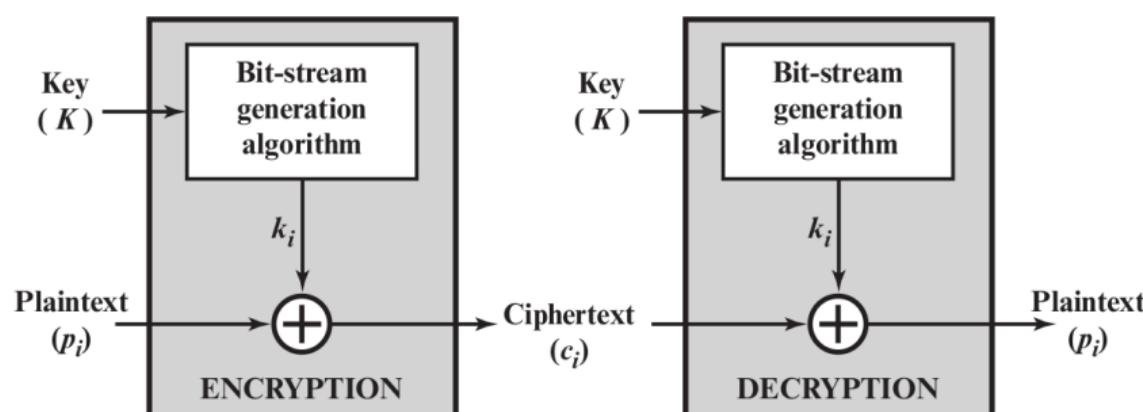
5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

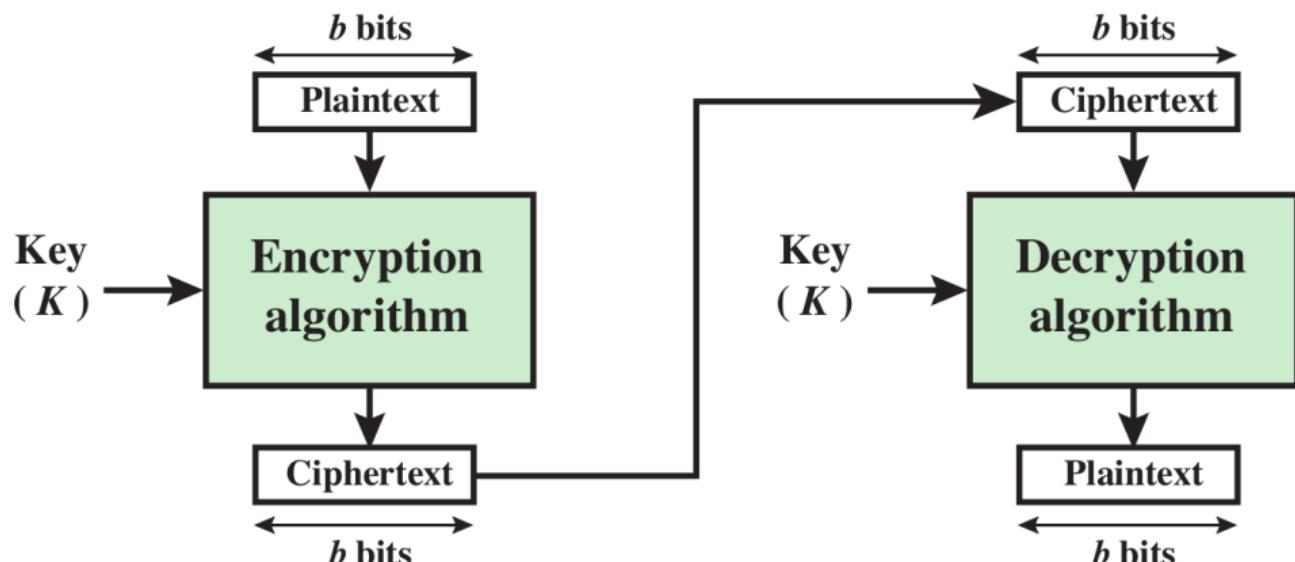
流密码 (Stream Cipher)

- **流密码**每次加密数据流的一个比特位或一个字节，得到与明文序列同样长度的密文序列。
- **加密**：以比特/字节为单位，让明文序列与密钥流按比特/字节异或运算后，作为密文序列。
- **解密**：以比特/字节为单位，让密文序列与相同的密钥流按比特/字节异或运算后，得到明文文序列。



分组密码 (Block Cipher)

- 分组密码将一个明文分组作为整体进行加密，得到与明文等长的密文分组。
- 通常以大于等于 64 位的数据块为分组单位，加密得到相同长度的密文分组。



目录

1 分组密码的基本原理

- 流密码与分组密码
- **理想分组密码**
- Feistel 分组密码

2 数据加密标准

3 DES 的安全性

4 数论基础

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

理想分组密码：可逆映射

- 分组密码作用于 n 位明文分组上，产生 n 位密文分组。
- n 位明文分组有 2^n 种输入，每一种都必须产生一个唯一密文分组，这种变换称为可逆的或非奇异的。

明文	密文
00	11
01	10
10	00
11	01

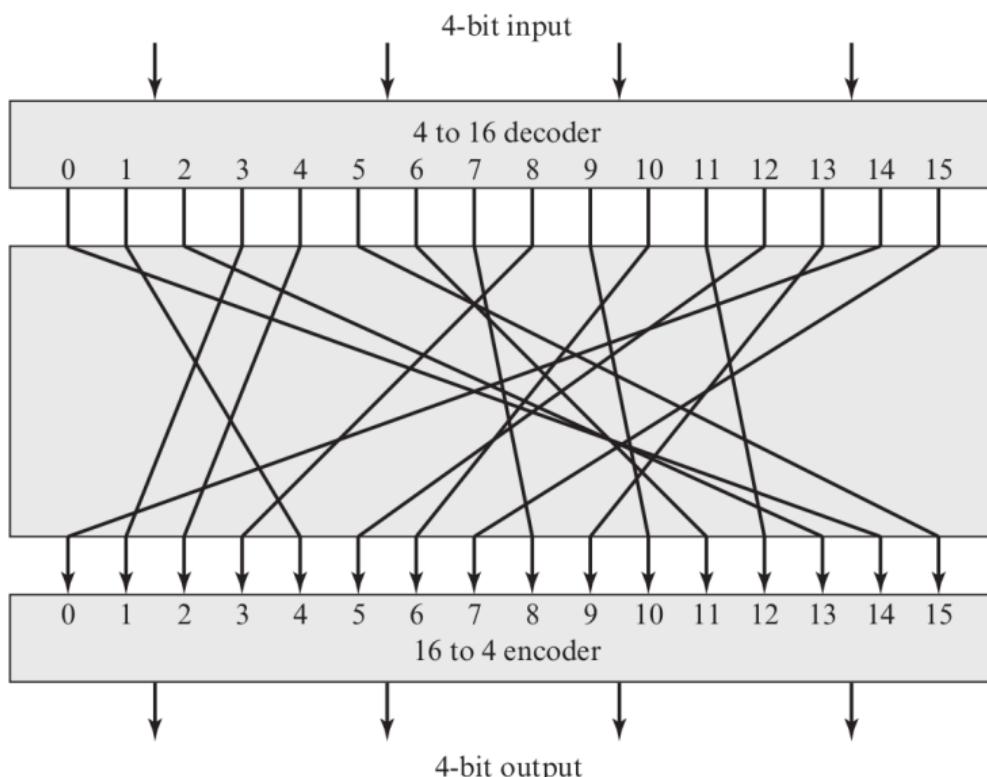
可逆映射

明文	密文
00	11
01	10
10	01
11	01

不可逆映射

理想分组密码：一个 4 位到 4 位的分组密码

分组密码本质上可以看作是一个巨大的代换密码。



理想分组密码的密钥

- 一共有 $2^n!$ 种可逆映射。
- 表的第二列定义了 $2^n!$ 个映射中的某个特定映射，即为理想分组密码的密钥。
- Feistel 称这种密码为**理想分组密码**，因为它允许生成最大数量的映射。
- 理想分组密码拥有最大的密钥空间 $2^n!$
- 密钥大小为 $n2^n$ 比特，因为只需保存表的第二列。

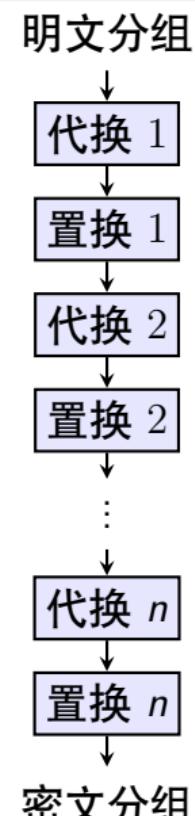
Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

理想分组密码存在的问题

- 分组长度 n 比较小时, 例如 $n = 8$, 密码系统等价于传统代换密码, 容易利用明文的统计信息攻击它。
- 如果 n 充分大并且允许明密文之间采用任意可逆变换, 那么明文的统计特征将被掩盖, 从而不能利用明文的统计信息攻击这种密码系统。
- 对于 n 位分组, 密钥大小为 $n2^n$ 比特。
 - 例如一个 64 位理想分组密码, 密钥大小为 $64 \times 2^{64} = 2^{70}$ bit = 1Zb
 - 1ZB: global yearly Internet traffic in 2016.
- 1973 年, Horst Feistel 指出: 我们所需要的分组密码是对理想分组密码的一种近似, 更容易实现, 提出了基于可逆乘积密码概念的 Feistel 分组密码结构。

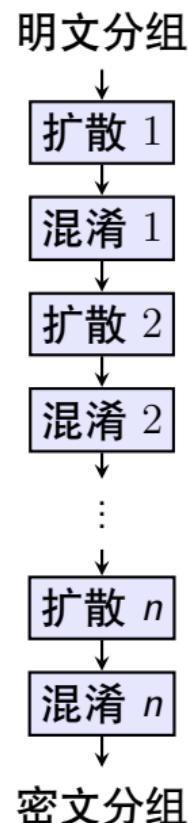
乘积密码的设计思想

- 乘积密码指依次使用两个或两个以上的基本密码，增强密码的强度。
- Feistel 建议交替使用代换和置换设计分组密码：
 - 代换**：每个明文字母被唯一地替换为相应的密文字母
 - 置换**：明文字母序列被替换为该序列的一个置换
- 实际上这个方案是 Shannon 1949 年提出的 Substitution-Permutation Networks (SPN) 的实现。
- Shannon 认为，为了应对基于统计分析的密码分析，必须对明文做**扩散**和**混淆**，以减少密文的统计特性，为密码分析制造障碍。



乘积密码的设计思想：扩散和混淆

- **扩散 (Diffusion)**：使输入（包括明文和密钥）的统计特征消散在输出（例如密文或哈希值）中，让每个输入比特影响尽可能多的输出比特。其目的是**隐藏输出和输入的统计关系**，使输入的统计特征扩散到输出中去，从而无法根据输出的统计特征分析输入的统计特征。
- **混淆 (Confusion)**：输出结果的每一个比特位都应该依赖于输入的大部分内容，即输入和输出之间没有直接的映射关系。其目的是**隐藏输入与输出之间的映射关系**，使之变得尽可能复杂而难以分析。
- 扩散-混淆原则的目的是为了增强密码算法的安全强度，评判扩散-混淆效果的标准是看能否发生**雪崩效应**：输入的微小改变导致输出的大幅改变。



目录

1 分组密码的基本原理

- 流密码与分组密码
- 理想分组密码
- Feistel 分组密码

2 数据加密标准

3 DES 的安全性

4 数论基础

5 高级数据加密标准

6 多重加密

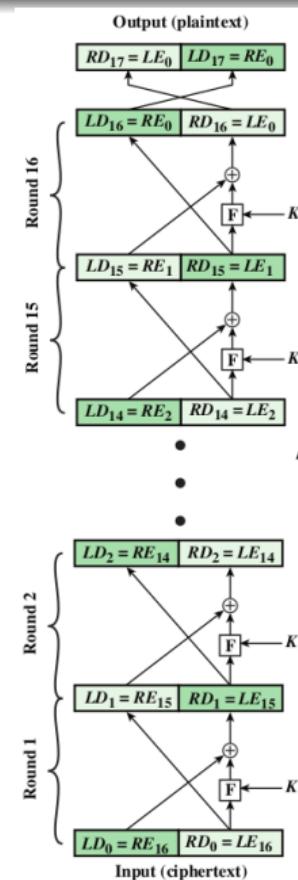
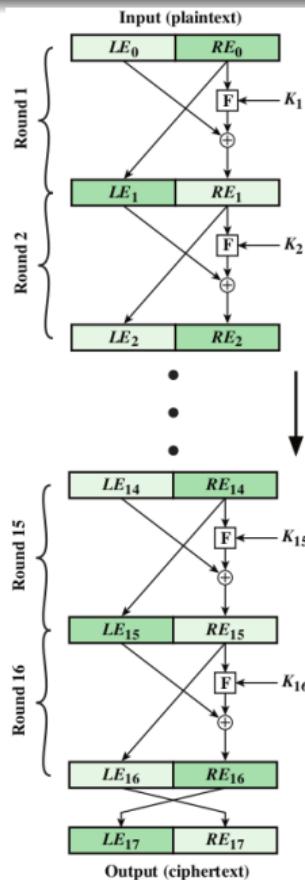
7 分组密码的工作模式

Feistel 密码结构

- Feistel 密码结构（也叫 Feistel Networks），由德裔美国人 Horst Feistel（物理学家和密码学家）在 1973 年提出。
- Feistel 在美国 IBM 工作期间完成此项开拓性研究，目前大部分分组密码都使用该方案，包括数据加密标准（DES）。
- Feistel 密码结构的优点在于加密和解密操作非常相似，在某些情况下甚至是相同的，只需逆转密钥编排，因此能够使代码或电路规模减半。
- 密码学家已经深入研究了 Feistel 密码结构的安全性。

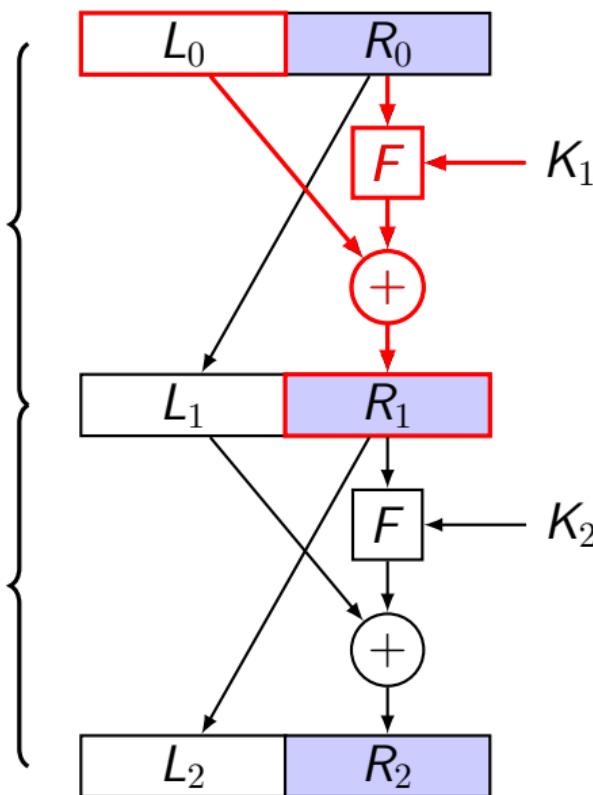
Feistel 密码结构

加密过程 →



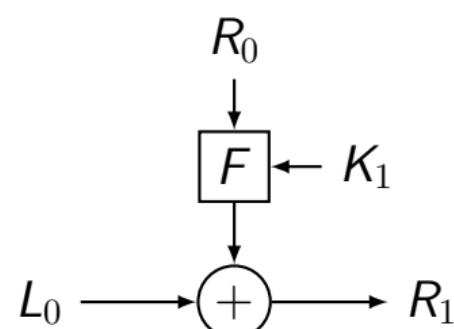
Feistel 密码结构

第 1 轮



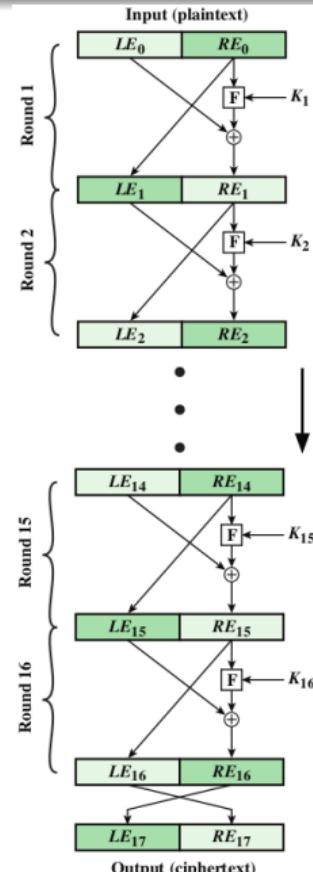
第 2 轮

L_0, R_0	本轮输入分组
K_1	轮密钥
F	轮函数
L_1, R_1	本轮输出分组



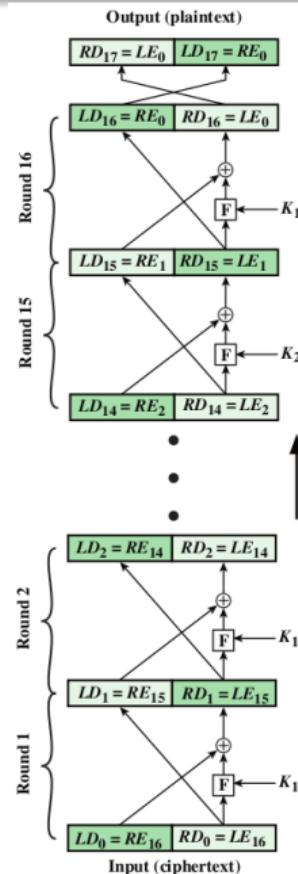
Feistel 密码：加密过程

- 将输入分组分成左右两部分，实施 Shannon's 的 SPN 概念；
- 对左半部数据实施多回合的代替操作；
- 将右半部数据和子密钥输入到轮函数 F ，其输出与左半部分数据异或；
- 最后一轮操作结束后，将两部分数据进行互换，得到输出密文分组。



Feistel 密码：解密过程

- 解密过程本质上与加密过程一致；
- 将密文作为算法输入，逆序使用子密钥；
- 解密的过程不要求轮函数 F 是可逆的；
- 由于加密与解密对称，Feistel 结构的电路实现可以减少硬件元器件。



Feistel 密码设计原则

- **分组长度**：分组越长则安全性越高，但加/解密速度越低，分组长度为 64 位是一个合理的折衷；
- **密钥长度**：密钥越长越安全，但加/解密速度越低，64 位长的密钥已被证明是不安全的，128 位是常用的长度；
- **迭代次数**：迭代越多越安全，通常为 16 次迭代；
- **子密钥产生算法**：越复杂则密码分析越困难；
- **轮函数 F** ：越复杂则抗密码分析的能力越强；
- **快速的软件加密/解密**：算法的执行速度很重要；
- **简化分析难度**：算法简洁清楚，易于分析弱点，发现问题。

目录

1 分组密码的基本原理

2 **数据加密标准**

3 DES 的安全性

4 数论基础

5 高级数据加密标准

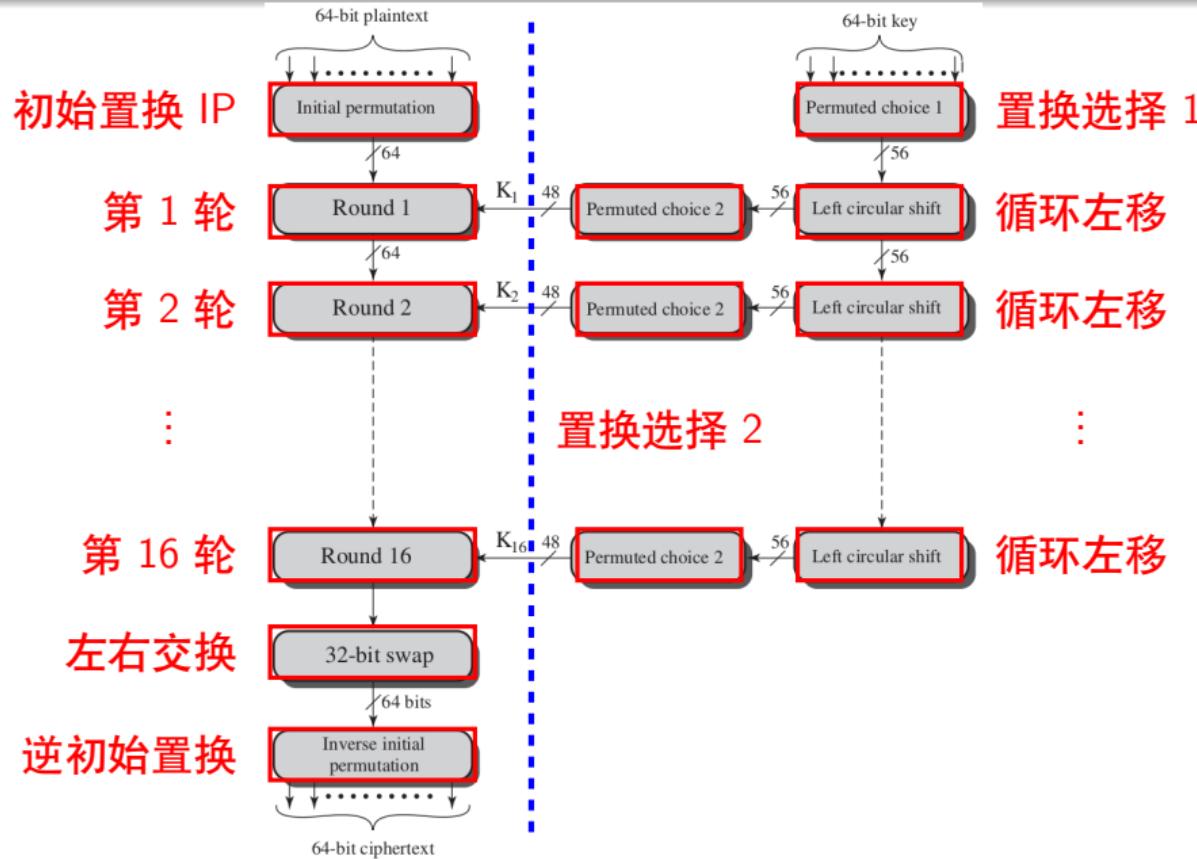
6 多重加密

7 分组密码的工作模式

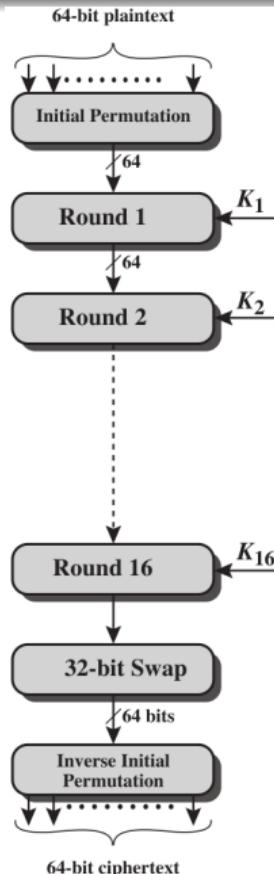
DES 的历史

- IBM 公司在 1971 年由 Horst Feistel 领导开发了 Lucifer Cipher，使用 128 位密钥加密 64 位的分组。
- 1974 年，IBM 与 NSA 合作开发了 Lucifer 的一个修订版，易于在芯片上实现，抗密码分析能力更强，且密钥缩短为 56 位。
- 1977 – 1998 期间，这个加密方案成为美国国家密码标准，称为 DES。
- DES 在密码学领域被深入分析。
- 目前 DES 已经不安全，已经被淘汰，替代算法包括 3DES、AES 等。

DES 加密过程



DES 加密过程



- DES 的明文长 64 位, 密钥长 56 位 (虽然输入 64 位密钥, 但内部仅使用了 56 位);
- 明文处理经过三个阶段:
 - 首先 64 位明文经过初始置换 (IP) 而被重新排列;
 - 然后进行 16 轮相同函数的作用, 每轮都进行代替和置换;
 - 最后一轮输出 64 位分组, 左右互换产生预输出, 经过逆初始置换 (IP^{-1}) 产生 64 位密文。
- 除了初始和末尾的置换操作, DES 的结构与 Feistel 密码结构完全相同。

初始置换 IP 和逆初始置换 IP^{-1}

(a) Initial Permutation (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

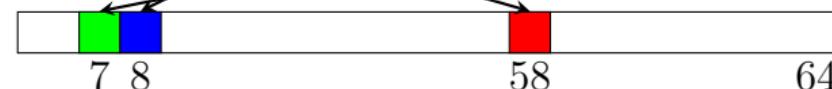
(b) Inverse Initial Permutation (IP^{-1})

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

64 位输入分组



IP 后分组

IP⁻¹ 后分组

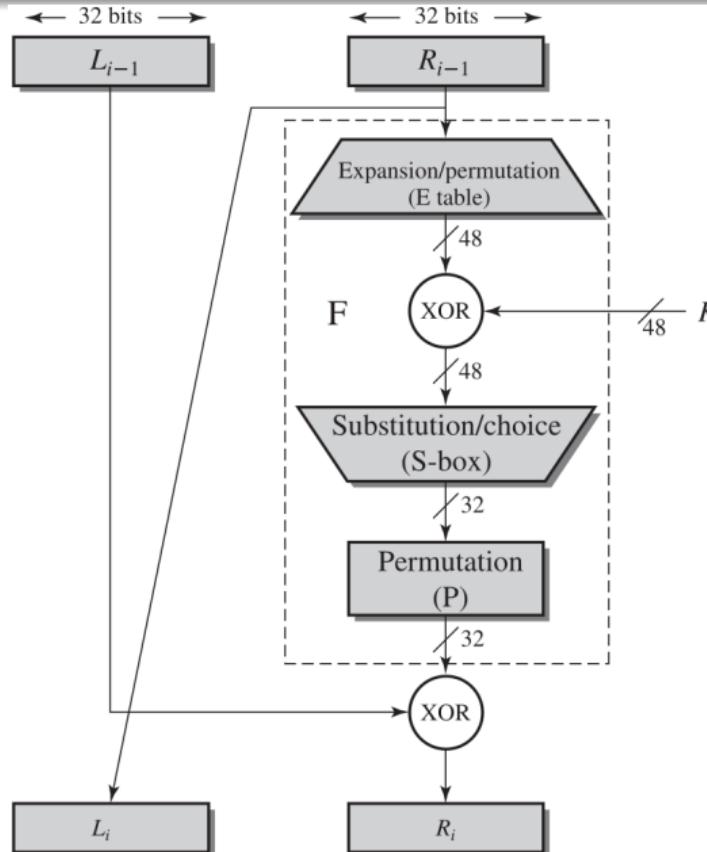
初始置换 IP 和逆初始置换 IP^{-1}

- 初始置换和逆置换并不能增强 DES 的安全性，而是为了方便硬件电路实现¹。
- 当总线宽度为 8 位时，需要配合使用 8 个 8 位移位寄存器，经过 8 个时钟得到 64 位输入分组。
- 如果不使用初始置换，那么从 8 个寄存器取前 32 位和后 32 位时，会在电路连线中产生很多交叉。

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

¹<https://crypto.stackexchange.com/a/6>

每一轮的运算



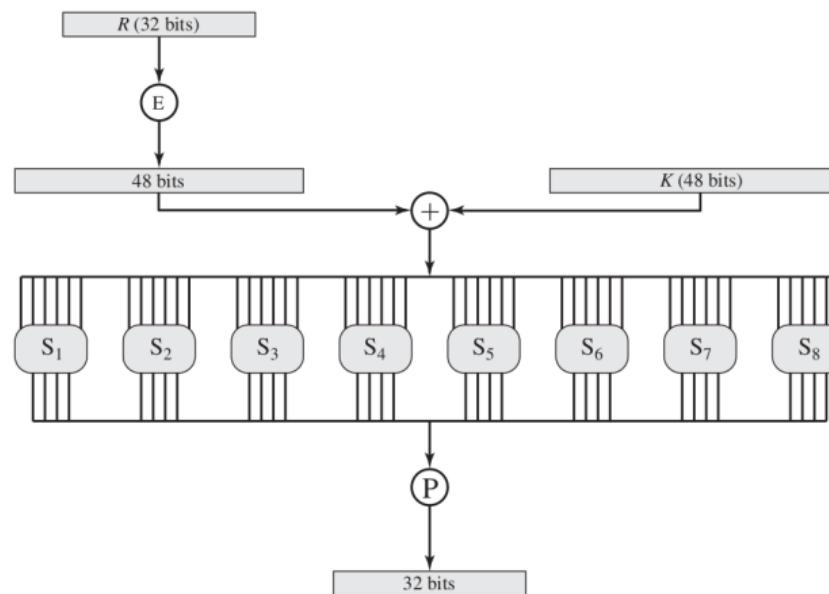
- 第 i 轮的输入分成左右两部分 L_{i-1} 和 R_{i-1} ；
- 做如下运算得到本轮输出的左右两部分 L_i 和 R_i ：

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

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

其中 \oplus 表示异或运算。

轮函数 F

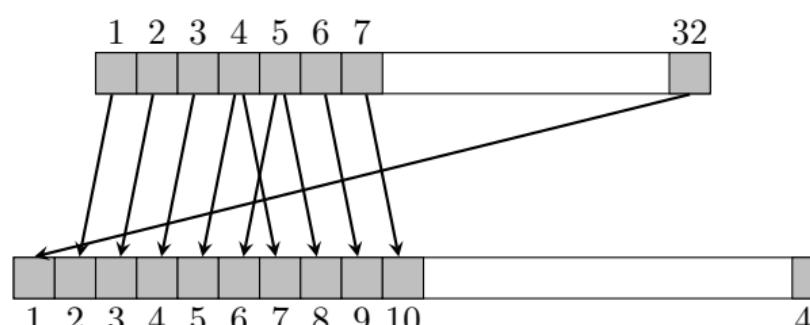


轮函数 F 是 DES 的核心运算函数，包含 4 步运算：扩展置换函数 E 、与子密钥异或、 S 盒替换和置换函数 P 。

扩展置换函数 E

使用置换表 E 将 32 位 R 扩展成 48 位，起扩散作用。

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



S 盒替换

- 48 位结果送给 8 个替换盒 S_1, \dots, S_8 , 得到 32 位结果;

S_1	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
S_2	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
S_3	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
S_4	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

S 盒替换

S_5	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
S_6	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
S_7	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
S_8	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

S 盒替换

- S 盒是轮函数 F 的核心，每个 S 盒输入 6 位，输出 4 位。
- 作用是混淆，即通过 S 盒替换使密文和密钥之间的关系尽可能复杂。
- 每个 S 盒输入的第一位和最后一位组成一个 2 位二进制数用来选择 S 盒 4 行中的某一行，中间 4 位用来选择 16 列中的某一列。
- 行列对应的十进制数转换为二进制后可得到输出的 4 位二进制数。

例

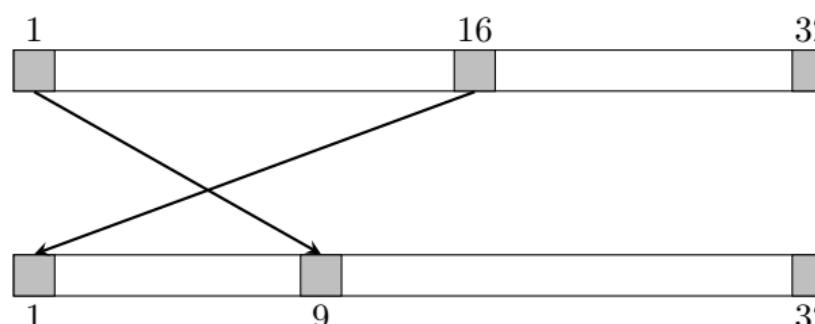
例如，在 S_1 中，若输入为 011001，则行是 1(01)，列是 12(1100)，该处的值为 9，所以输出 1001。

置换函数 P

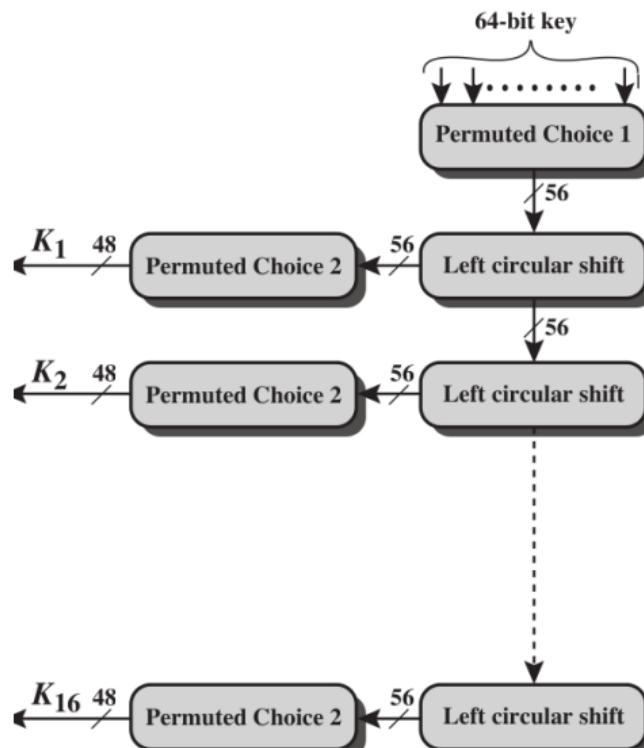
- 最后使用 32 位置换表 P ，把 32 位结果再进行一次置换处理。

(d) Permutation Function (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

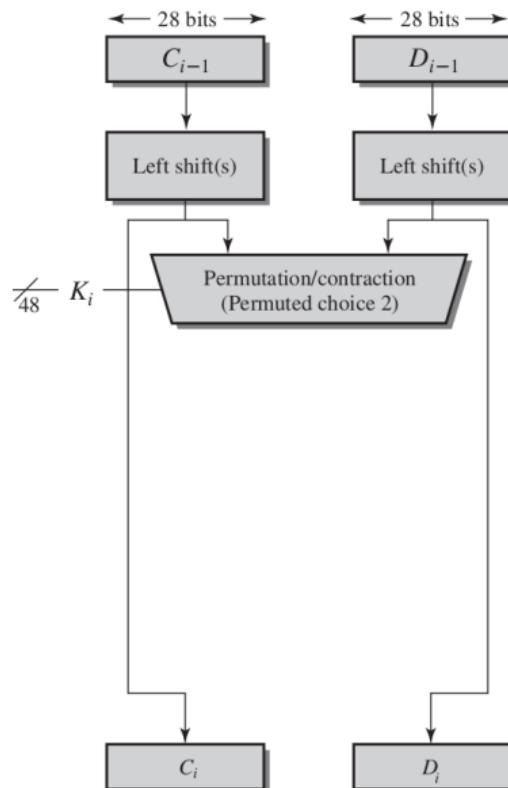


DES 子密钥生成过程



- 密钥经过一个置换，然后循环左移，再经过另一个置换，得到各轮的子密钥 K_i ；
- 每轮的置换函数都一样，由于循环左移，使得各轮子密钥各不相同。

子密钥生成算法



- 每一轮都要依据输入密钥生成一个子密钥以供加密使用；
- 输入密钥为 64 位，DES 只使用其中 56 位，其余位可用于奇偶校验；
- 使用**置换选择 1 (PC-1)**，将 56 位密钥分成两半 C 和 D ，每部分 28 位；
- 根据**循环左移表**将这两半分别循环左移 1 位或 2 位；
- 使用**置换选择 2 (PC-2)**，形成 48 位子密钥，用在轮函数 F 中。

DES 只使用输入密钥中的 56 位

(a) Input Key

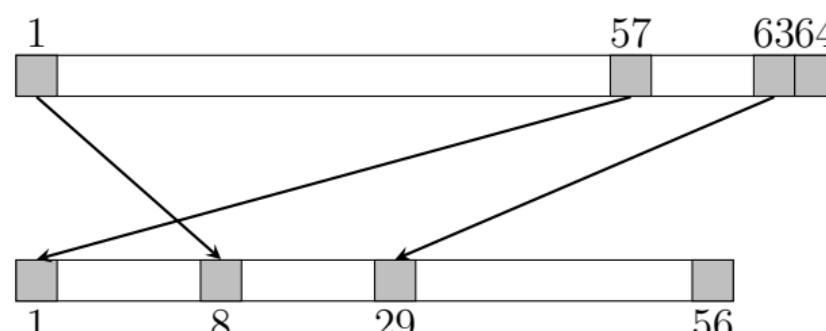
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

- 为了与输入明文分组长度一致，DES 密钥长度为 64 位，实际只使用 56 位，每个字节的最后一位在 DES 算法中没有使用，可用于校验等其他功能。

置换选择 1

(b) Permuted Choice One (PC-1)

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



循环左移表

(d) Schedule of Left Shifts

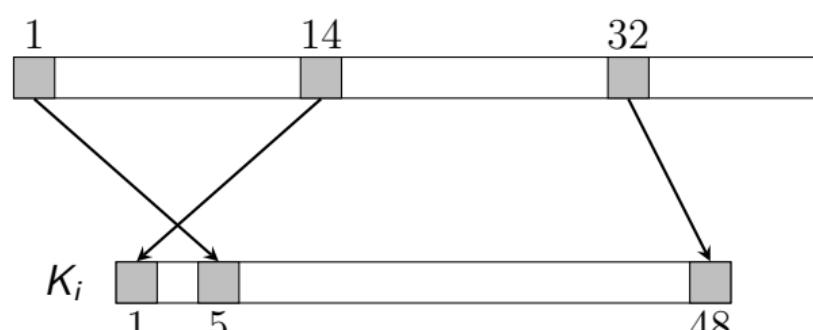
Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

每轮循环左移 1 ~ 2 位。

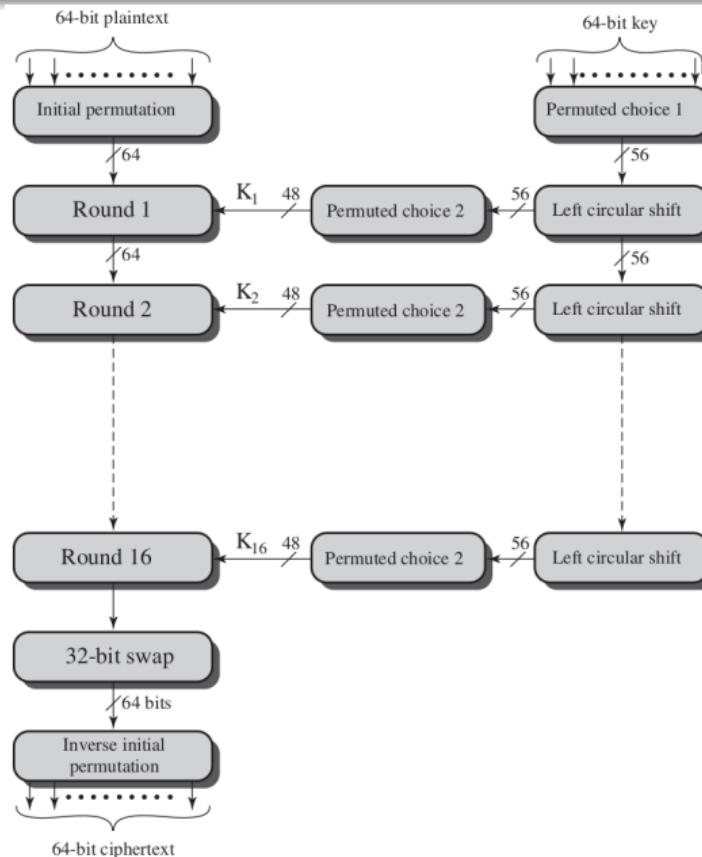
置换选择 2

(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32



DES 解密



- 同 Feistel 密码，DES 解密使用与加密相同的算法，只是子密钥的使用顺序相反。

DES 举例

Plaintext:	02468aceeca86420
Key:	0f1571c947d9e859
Ciphertext:	da02ce3a89ecac3b

Round	K_i	L_i	R_i
IP		5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP⁻¹		da02ce3a	89ecac3b

Note: DES subkeys are shown as eight 6-bit values in hex format

雪崩效应

- 雪崩效应 (Avalanche Effect): 明文或密钥的一比特的变化，引起密文许多比特的改变。如果变化太小，就可能找到一种方法减小有待搜索的明文和密文空间的大小。
- 如果用同样密钥加密只差一比特的两个明文：
0000000000000000.....00000000
1000000000000000.....00000000
3 次循环以后密文有 21 个比特不同，16 次循环后有 34 个比特不同。
- 如果用只差一比特的两个密钥加密同样明文：
3 次循环以后密文有 14 个比特不同，16 次循环后有 35 个比特不同。

DES 的雪崩效应：改变明文

使用相同密钥加密两个只差 1 比特的明文：

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccacaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33

Round		δ
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbcc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP⁻¹	da02ce3a89ecac3b 057cde97d7683f2a	32

DES 的雪崩效应：改变密钥

使用相差 1 比特的两个密钥 (0f1571c947d9e859 与 1f1571c947d9e859) 加密相同明文：

Round		δ
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32

Round		δ
9	c11bfc09887fbcc6c 548f1de471f64dfd	34
10	887fbcc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeea	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	33
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP⁻¹	da02ce3a89ecac3b ee92b50606b62b0b	30

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

- DES 加密强度
- 差分分析和线性分析
- 分组密码的设计原理

4 数论基础

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

- DES 加密强度
- 差分分析和线性分析
- 分组密码的设计原理

4 数论基础

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

密钥长度问题

- 56 位密钥有 $2^{56} \approx 7.2 \times 10^{16} = 7.2$ 亿亿之多。
- 穷举搜索 (brute force search) 似乎很困难, 20 世纪 70 年代估计要 1000 ~ 2000 年。
- 技术进步使穷举搜索成为可能:
- 1997 年 1 月 29 日, RSA 公司发起破译 RC4、RC5、MD2、MD5, 以及 DES 的活动, 破译 DES 奖励 10,000 美金。明文是: Strong cryptography makes the world a safer place。结果仅搜索了 24.6% 的密钥空间便得到结果, 耗时 96 天。
- 1998 年在一台专用机 (“DES 破译机”) 上只要三天时间即可!
- 1999 年在超级计算机上只要 22 小时!
- 现在只需要 10 小时!

DES 的内部结构问题

- 密码分析者有可能利用 DES 算法本身的特征进行攻击。
- S 盒的设计标准被列为官方机密，并没有公开。
- NSA 有可能利用这些内部机密在没有密钥的情况下解密。
- 但是迄今为止并没有发现 S 盒存在致命弱点。

计时攻击

- 通过观察算法对多种密文解密所需的时间，来获取关于密钥或明文的信息。
- 计时攻击所利用的信息是加密或解密算法对于不同输入所花的时间有着细微的差别。
- 例如利用计时攻击分析密钥的汉明权重，即二进制串中 1 的个数。
- 目前为止，计时攻击还不可能成功攻击 DES。

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

- DES 加密强度
- 差分分析和线性分析
- 分组密码的设计原理

4 数论基础

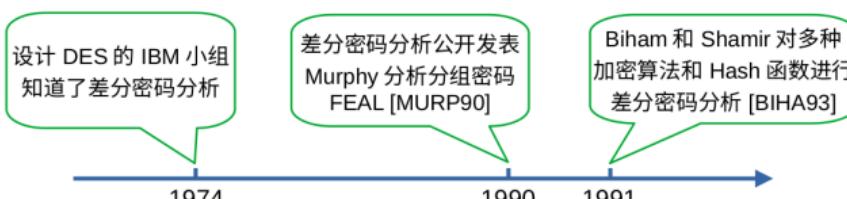
5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

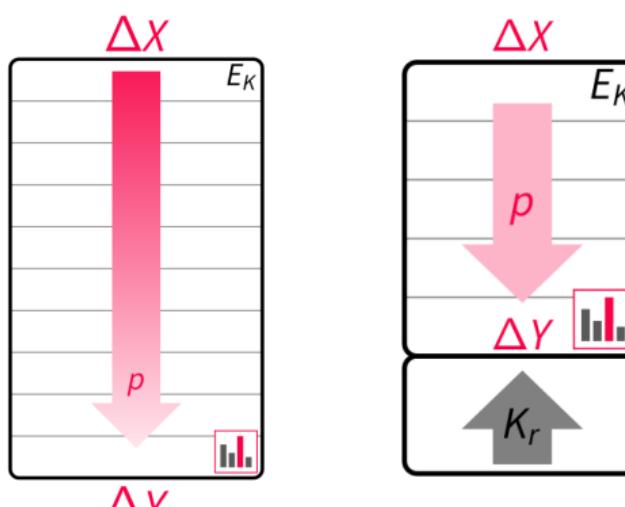
差分密码分析 (Differential Cryptanalysis)

- **差分密码分析**属于选择明文攻击。通过分析明文对的差分（即异或）对结果密文对的差分的影响，确定最有可能的密钥。
 - 1990 年，Murphy、Biham 和 Shamir 首次提出用差分密码分析攻击分组密码和散列函数。
 - 研究表明，若有 2^{47} 个选择明文，用差分分析就可以在 2^{47} 次加密运算内成功攻击 DES。但是要拥有 2^{47} 个选择明文的条件使得这种方法只具有理论上的意义。
 - DES 在设计之初已经考虑了抵抗差分密码分析。在设计 S 盒和置换 P 时已经做了充分考虑。

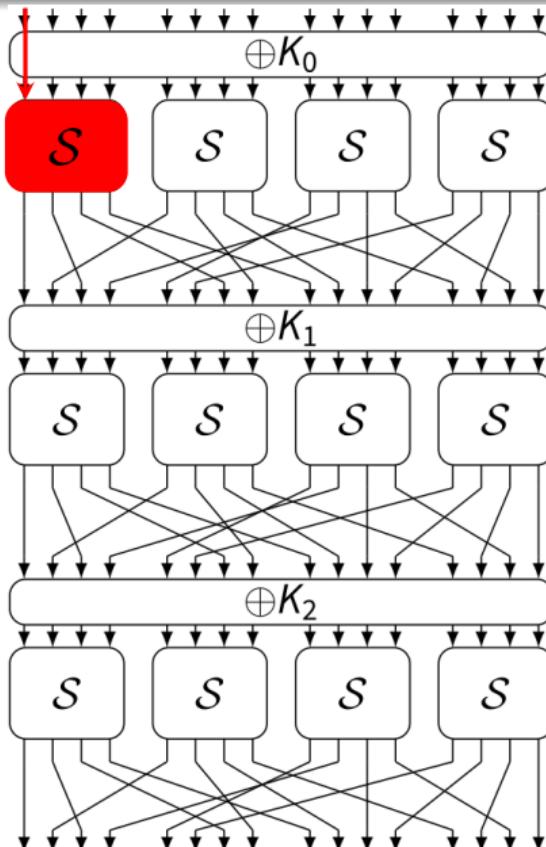


差分密码分析的主要思想

- 考虑输入差分为 ΔX 的一对明文；
- 跟踪这对明文经过每轮处理后的变化；
- 根据输出的差分推测每一轮处理所使用的子密钥。



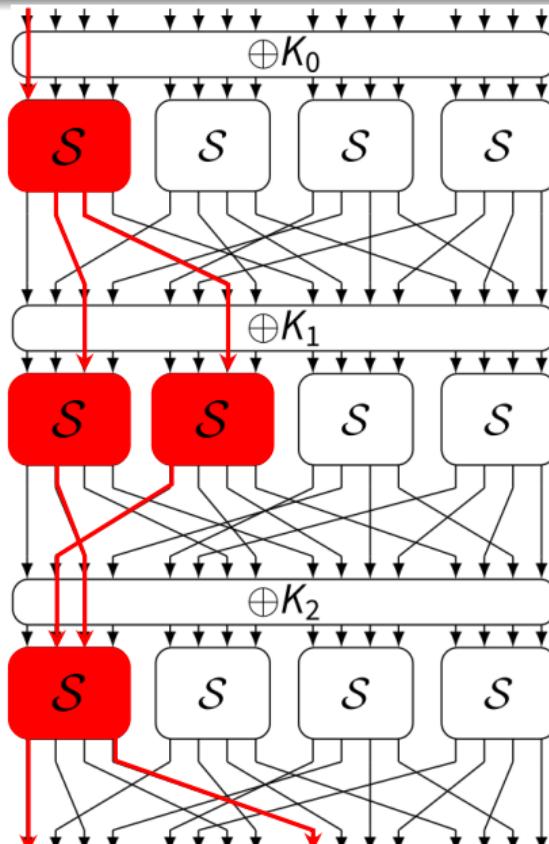
差分密码分析的主要思想



- 将 DES 简化为左图所示的运算，只包含 DES 的关键运算。
- 一次轮函数运算可以看作输入与该轮子密钥异或后再经过 S 盒替换。
- 输入异或不受子密钥影响**：考虑一对输入 x 和 x' ，则

$$(x \oplus k) \oplus (x' \oplus k) = x \oplus x'$$
- 当差分输入到一个 S 盒时，可以利用 S 盒的**差分特性**进行分析。

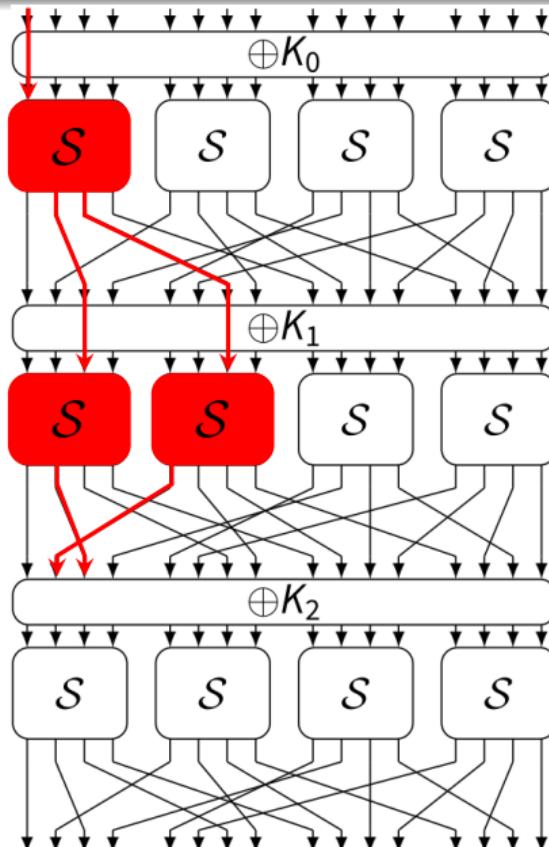
差分密码分析的主要思想



- 考虑具有相同输入差分的所有输入对: $\{(x, x') | x \oplus x' = \delta\}$
- 统计 S 盒相应的输出对的差分, 会发现**输出差分的分布往往不均匀**。
- 例如, 当输入差分为 1011 时, S_1 盒的输出差分分布为

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	
0	0	8	0	0	2	0	2	0	0	0	0	0	0	2	0	2
- 从而可以计算出从某个输入差分产生某个输出差分的概率。

差分密码分析的攻击过程



- 搜集尽可能多的明密文对 $(x, y), (x', y')$ 且满足 $x \oplus x' = \delta$;
- 对于每对输入明文，计算出倒数第二轮的输出差分及其对应的传递概率；
- 枚举最后一轮的所有可能密钥，对于每对输出密文用密钥解密，得到最后一轮的输入对；
- 若该输入对的差分等于上一步计算出的输出差分，则该密钥计数加一；
- 根据每个密钥的计数值，输出计数值最大的密钥为该轮的子密钥。

线性密码分析 I

- 1993 年提出的一种统计攻击方法，通过寻找 DES 变换的线性近似来进行攻击。
- 可以在有 2^{43} 个已知明文的情况下破译 DES 密钥，但仍然只具有理论意义。
- 令明文分组为 $P[1], \dots, P[n]$ ，密文分组为 $C[1], \dots, C[n]$ ，密钥为 $K[1], \dots, K[m]$ 。定义 $A[i, j, \dots, k] \triangleq A[i] \oplus A[j] \oplus \dots \oplus A[k]$ 。
- 线性密码分析的目标是找到如下有效线性方程：

$$P[\alpha_1, \alpha_2, \dots, \alpha_a] \oplus C[\beta_1, \beta_2, \dots, \beta_b] = K[\gamma_1, \gamma_2, \dots, \gamma_c]$$

其中 $1 \leq a, b \leq n, 1 \leq c \leq m, \alpha, \beta$ 和 γ 等表示固定的唯一的比特位置。

线性密码分析 II

- 要求方程以概率 $p \neq 0.5$ 成立, p 离 0.5 越远, 方程越有效。
- 对于大量的明文密文对, 计算方程左边的值, 如果结果中有一半以上为 0, 则假定 $K[\gamma_1, \gamma_2, \dots, \gamma_c] = 0$; 如果大多为 1, 则假定 $K[\gamma_1, \gamma_2, \dots, \gamma_c] = 1$ 。

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

- DES 加密强度
- 差分分析和线性分析
- 分组密码的设计原理

4 数论基础

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

S 盒的设计准则：增加扰乱性

- 输出比特不应太接近输入比特的一个线性函数；
- 每一行应该包括所有 16 种比特组合；
- 两个输入相差一个比特，输出必须相差两个比特；
- 如果两个输入刚好在两个中间比特上不同，输出必须在至少两个比特上不同；
- 两个输入前两位不同而最后两位相同，两个输出必须不同；
- 具有非零 6 比特差值的输入，32 对中有不超过 8 对输出相同。

置换 P 的设计准则：增加扩散性

- 第 i 次循环时每个 S 盒输出的四个比特被分布开，以便其中两个影响下一循环的中间比特，两个影响两端的比特；
- 每个 S 盒输出的四个比特影响下一循环的 6 个不同的 S 盒，并且任何两个都不会影响同一个 S 盒；
- 如果 S_j 的一个输出比特影响下一循环 S_k 的中间比特，则 S_k 的一个输出比特就不能影响 S_j 的一个中间比特。

其他设计准则

- **迭代轮数**: 迭代次数越多则进行密码分析的难度就越大, 选择准则就是要使已知的密码分析工作量大于简单的穷举密钥搜索的工作量。
- **轮函数 F** : 提供扰乱作用, 要求强非线性, 良好的雪崩性质。
- **密钥扩展算法**: 选择子密钥时要使得推测各子密钥和由此推出主密钥难度尽可能大, 保证密钥/密文的严格雪崩效应准则和位独立准则。

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

- 群、环和域
- 模算术
- 欧几里得算法
- 有限域

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

- 群、环和域
- 模算术
- 欧几里得算法
- 有限域

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

群 (Groups)

定义 (群, Groups)

记作 $\{G, \cdot\}$, 定义了一个二元运算 \cdot 的集合 G , G 中每一个序偶 (a, b) 通过运算 \cdot 生成 G 中的元素 $a \cdot b$, 满足下列公理:

- (A1) **封闭性 Closure**: 如果 a 和 b 都属于 G , 则 $a \cdot b$ 也属于 G ;
- (A2) **结合律 Associative**: 对于 G 中任意元素 a, b, c , 都有 $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ 成立;
- (A3) **单位元 Identity element**: G 中存在一个元素 e , 对于 G 中任意元素 a , 都有 $a \cdot e = e \cdot a = a$ 成立;
- (A4) **逆元 Inverse element**: 对于 G 中任意元素 a , G 中都存在一个元素 a' , 使得 $a \cdot a' = a' \cdot a = e$ 成立。

注: 当群中的运算符是加法时, 习惯上记它的单位元为 0, a 的逆元是 $-a$, 并且减法用以下的规则定义: $a - b = a + (-b)$.

有限群、无限群、阶、交换群和循环群

定义 (有限群, 无限群, 阶)

如果群的元素是有限个, 则该群称为**有限群**; 否则, 称为**无限群**。有限群中元素的个数称为**有限群的阶**。

定义 (交换群, 阿贝尔群, Abelian Groups)

还满足以下条件的群称为**交换群**:

(A5) **交换律 Commutative**: 对于 G 中任意的元素 a, b , 都有 $a \cdot b = b \cdot a$ 成立。

定义 (循环群, Cyclic Groups)

如果群中的每一个元素都是一个固定的元素 $g \in G$ 的幂 g^k (k 为整数), 则称群 G 为**循环群**。元素 g 生成了群 G , 或者说 g 是群 G 的**生成元**。

环 (Rings)

定义 (环, Rings)

环 R , 记为 $\{R, +, \times\}$, 是具有加法和乘法两个二元运算的元素的集合, 对于环中的任意元素 a, b, c 满足以下公理:

(A1-A5) R 关于加法是一个交换群, 单位元是 0, a 的逆是 $-a$ 。

(M1) 乘法封闭性: 如果 a 和 b 属于 R , 则 ab 也属于 R 。

(M2) 乘法结合律: 对于 R 中任意 a, b, c 有 $a(bc) = (ab)c$ 。

(M3) 分配律: $a(b + c) = ab + ac$ 或 $(a + b)c = ac + bc$ 。

例 (环)

定义在整数集 \mathbb{Z} 上的加法和乘法运算, 都满足上述公理, 所以 $\{\mathbb{Z}, +, \times\}$ 构成一个环。

交换环和整环

定义 (交换环)

环如果还满足以下条件，则被称为**交换环**：

(M4) 乘法交换律： $ab = ba$ 。

定义 (整环)

交换环如果还满足以下条件，则被称为**整环**：

(M5) 乘法单位元： R 中存在元素 1 使得所有 a 有 $a1 = 1a$ 。

(M6) 无零因子：如果 R 中有 a, b 且 $ab = 0$ ，则 $a = 0$ 或 $b = 0$ 。

注：无零因子指没有非平凡零因子。

例 (整环)

定义在整数集上的环 $\{\mathbb{Z}, +, \times\}$ 是交换环，也是整环。

域 (Fields)

定义 (域, Fields)

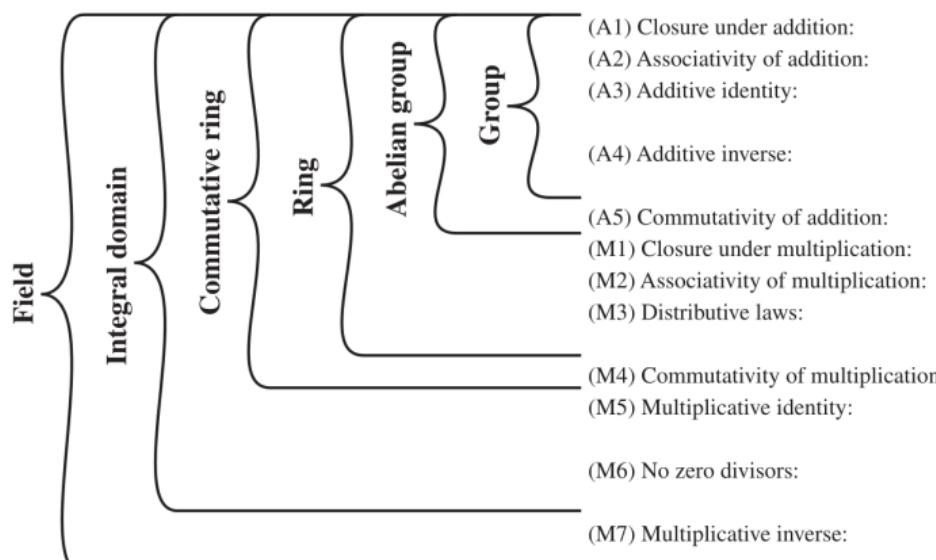
记为 $\{F, +, \times\}$, 是有加法和乘法的两个二元运算的元素的集合, 对于 F 中的任意元素 a, b, c , 满足以下公理:

(A1-M6) F 是一个整环;

(M7) 乘法逆元: 对于 F 中的任意非零元素 a , F 中都存在一个元素 a^{-1} , 使得 $aa^{-1} = a^{-1}a = 1$ 。

- 域就是一个集合, 在其上进行加减乘除而不脱离该集合, 除法按以下规则定义: $a/b = ab^{-1}$ 。
- 有理数集合、实数集合和复数集合都是域;
- 整数集合不是域, 因为除了 1 和 -1 有乘法逆元, 其他元素都无乘法逆元。

群、环和域的关系



If a and b belong to S , then $a + b$ is also in S

$a + (b + c) = (a + b) + c$ for all a, b, c in S

There is an element 0 in R such that

$a + 0 = 0 + a = a$ for all a in S

For each a in S there is an element $-a$ in S such that $a + (-a) = (-a) + a = 0$

$a + b = b + a$ for all a, b in S

If a and b belong to S , then ab is also in S

$a(bc) = (ab)c$ for all a, b, c in S

$a(b + c) = ab + ac$ for all a, b, c in S

$(a + b)c = ac + bc$ for all a, b, c in S

$ab = ba$ for all a, b in S

There is an element 1 in S such that

$a1 = 1a = a$ for all a in S

If a, b in S and $ab = 0$, then either

$a = 0$ or $b = 0$

If a belongs to S and $a \neq 0$, there is an element a^{-1} in S such that $aa^{-1} = a^{-1}a = 1$

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

- 群、环和域
- **模算术**
- 欧几里得算法
- 有限域

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

模运算和同余

定义 (模运算)

如果 a 是整数, n 是正整数, 定义 a 除以 n 所得余数为 a 模 n , 记为 $a \bmod n$ 。对于任意整数 a , 有

$$a = \lfloor a/n \rfloor \times n + (a \bmod n).$$

例如, $11 \bmod 7 = 4$, $-11 \bmod 7 = 3$.

定义 (同余)

如果 $a \bmod n = b \bmod n$, 则称整数 a 和 b 是模 n 同余, 表示为 $a \equiv b \pmod{n}$ 或 $a \equiv_n b$.

例如, $73 \equiv 4 \pmod{23}$, $21 \equiv -9 \pmod{10}$

同余的性质

性质

- $n|(a - b) \Leftrightarrow a \equiv b \pmod{n}$.
- **对称性**: $a \equiv b \pmod{n} \Leftrightarrow b \equiv a \pmod{n}$.
- **传递性**: $a \equiv b \pmod{n}$ 且 $b \equiv c \pmod{n} \Rightarrow a \equiv c \pmod{n}$.

证明.

- (\Rightarrow) 如果 $n|(a - b)$, 则有 $(a - b) = kn$, k 为某个整数, 所以 $a = b + kn$ 。故 $a \bmod n = (b + kn) \bmod n = b \bmod n$ 。
- (\Leftarrow) 如果 $a \equiv b \pmod{n}$, 那么 $a = k_1n + r, b = k_2n + r$, 进而 $n|(a - b)$ 。



模算术运算

性质 (模运算的分配率)

$$(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$$

$$(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$$

性质 (模运算的加性和乘性)

如果 $a \equiv b \pmod{n}$ 且 $c \equiv d \pmod{n}$, 则

$$(a + c) \equiv (b + d) \pmod{n}$$

$$(a \times c) \equiv (b \times d) \pmod{n}$$

- $n|(a - b) \wedge n|(c - d) \Rightarrow n|(a - b + c - d) \Rightarrow n|[(a + c) - (b + d)] \Rightarrow (a + c) \equiv (b + d) \pmod{n}.$
- $n|(a - b) \wedge n|(c - d) \Rightarrow n|[c(a - b) + b(c - d)] \Rightarrow n|(ac - bd) \Rightarrow ac \equiv bd \pmod{n}.$

模算术运算

性质

如果 $ac \equiv bd \pmod{n}$ 且 $c \equiv d \pmod{n}$, $\gcd(c, n) = 1$, 则 $a \equiv b \pmod{n}$ 。

例如: $3 \times 2 \equiv 1 \times 2 \pmod{4}$ 且 $2 \equiv 2 \pmod{4}$, 但 $3 \neq 1 \pmod{4}$, 因为 $\gcd(2, 4) \neq 1$ 。

证明.

$$ac \equiv bd \pmod{n} \Rightarrow n | (ac - bd) \quad (1)$$

$$c \equiv d \pmod{n} \Rightarrow n | (c - d) \Rightarrow c - d = kn \text{ for some } k. \quad (2)$$

So we have $d = c - kn$. Continuing the argument of Eq. (1), we have that

$$n | [ac - b(c - kn)] \Rightarrow n | (ac - bc - kbn) \Rightarrow n | (a - b)c$$

Because $\gcd(c, n) = 1$, then c does not contain divisor n . Hence $a - b$ must have divisor n , i.e., $n | (a - b)$. We thus obtain $a \equiv b \pmod{n}$.



模算术运算

推论

如果 $ai \equiv aj \pmod{n}$ 且 $\gcd(a, n) = 1$ ，则 $i \equiv j \pmod{n}$ 。

令 $\mathbb{Z}_n \triangleq \{0, \dots, n-1\}$ 为小于 n 的非负整数集合。

引理

如果 $\gcd(a, n) = 1$ ，则对于每个 $i, j \in \mathbb{Z}_n$ 且 $i \neq j$ ，那么

$$ai \bmod n \neq aj \bmod n$$

证明

假设 $ai \bmod n = aj \bmod n$ ，即 $ai \equiv aj \pmod{n}$ 。由于 $\gcd(a, n) = 1$ ，所以 $i \equiv j \pmod{n}$ 。又因为 $i, j \in \mathbb{Z}_n$ ，所以只能 $i = j$ ，这与条件 $i \neq j$ 相矛盾，所以假设不成立。 □

加法逆元和乘法逆元

- **加法逆元**: 对于给定的 $a \in \mathbb{Z}_n$, 如果存在 $z \in \mathbb{Z}_n$, 使得 $a + z \equiv 0 \pmod{n}$, 则称 z 为 a 的加法逆元, 即 $-a = z$ 。
- **乘法逆元**: 对于给定的 $a \in \mathbb{Z}_n \setminus \{0\}$, 如果存在 $z \in \mathbb{Z}_n \setminus \{0\}$, 使得 $az \equiv 1 \pmod{n}$, 则称 z 为 a 的乘法逆元, 即 $a^{-1} = z$ 。
- $\mathbb{Z}_n \setminus \{0\}$ 中的所有元素都有加法逆元, 但不一定都有乘法逆元

+ 0	1	2	3	4	5	6	7	
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	0
2	2	3	4	5	6	7	0	1
3	3	4	5	6	7	0	1	2
4	4	5	6	7	0	1	2	3
5	5	6	7	0	1	2	3	4
6	6	7	0	1	2	3	4	5
7	7	0	1	2	3	4	5	6

模 8 加法

× 0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	0	2	4
3	0	3	6	1	4	7	2
4	0	4	0	4	0	4	0
5	0	5	2	7	4	1	6
6	0	6	4	2	0	6	4
7	0	7	6	5	4	3	2

模 8 乘法

乘法逆元存在的条件

\times	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

模 7 乘法

定理 (乘法逆元存在的条件)

如果 $\gcd(a, n) = 1$, 则 $\mathbb{Z}_n \setminus \{0\}$ 中存在 a 的模 n 乘法逆元 $z \in \mathbb{Z}_n \setminus \{0\}$, 使得 $az \equiv 1 \pmod{n}$, 即 $a^{-1} \pmod{n} = z$ 。

因为 a 与 n 互素, 由前面的引理知, 如果用 a 乘以 $\mathbb{Z}_n \setminus \{0\}$ 中的所有数 z 模 n , 得到的余数将以不同次序涵盖 $\mathbb{Z}_n \setminus \{0\}$ 中的所有数, 那么至少有一个余数为 1, 这时的 z 即为 a 的乘法逆元。

乘法逆元存在的条件

$i \in \mathbb{Z}_n$	$ai \bmod n \in \mathbb{Z}_n$
0	0
1	$a \bmod n$
2	$2a \bmod n$
\vdots	\vdots
$n - 1$	$a(n - 1) \bmod n$

- 由引理知, 当 $\gcd(a, n) = 1$ 时, 第二列的 n 个元素互不相等。
- 又这 n 个元素都取值于 \mathbb{Z}_n , 因此它们构成的集合就是 \mathbb{Z}_n 。
- 那么这个集合必然存在元素 1, 记 $ax \bmod n = 1$, 这个 x 就是 a 的乘法逆元。

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

- 群、环和域

- 模算术

- 欧几里得算法

- 有限域

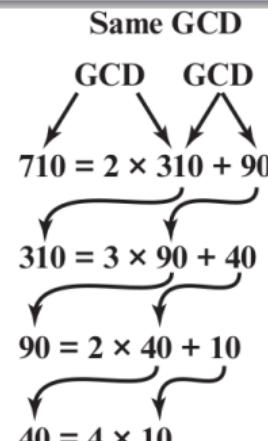
5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

欧几里得算法 (Euclidean Algorithm)

- 欧几里得算法是数论中的一个基本技巧，可以求两个正整数的最大公约数。
- 欧几里得算法的原理：对任意整数 a, b ，且 $a \geq b > 0$ ，则 $\gcd(a, b) = \gcd(b, a \bmod b)$ 。
- 也就是说，求 a, b 的最大公约数可以转化为求 b 和 a 模 b 的最大公约数，即**辗转相除法**。



```

Euclid(a, b){
    if (b==0) then return a;
    else return Euclid(b, a mod b);
}

```

欧几里得算法的原理

- 假设要求整数 a 和 b 的最大公因子，不妨令 $a \geq b > 0$ 。
- b 除 a 可以表示为 $a = qb + r$ ，其中 $0 \leq r < b$ 为余数。
- 如果 $r = 0$ ，则 $\gcd(a, b) = b$ ；
- 如果 $r \neq 0$ ，考虑 $\gcd(a, b)$ 和 $\gcd(b, r)$ 之间的关系：
 - 令 $d = \gcd(a, b)$ 。因为 $d|a$ 且 $d|b$ ，所以 $d|(a - qb)$ ，即 $d|r$ 。也就是说， d 是 b, r 的公因子。那么， $d \leq \gcd(b, r)$ 。
 - 令 $c = \gcd(b, r)$ 。因为 $c|b$ 且 $c|r$ ，所以 $c|(qb + r)$ ，即 $c|a$ 。也就是说， c 是 a, b 的公因子。因为 a, b 的最大公因子是 d ，所以 $c = \gcd(b, r) \leq d$ 。
- 所以 $\gcd(a, b) = \gcd(b, r)$ ，即求 a 和 b 的最大公因子可以转化为求 b 和 r 的最大公因子。

扩展欧几里得算法

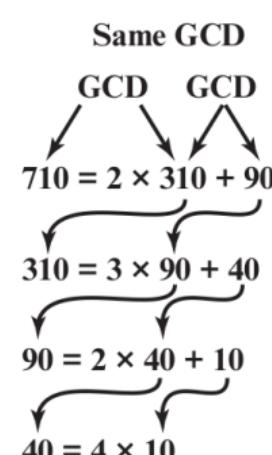
- 给定两个整数 a 和 b , 扩展欧几里得算法不仅可以求出最大公因子 d , 而且可以得到两个整数 x 和 y , 满足

$$ax + by = d = \gcd(a, b)$$

- 利用欧几里得算法, 并且假设每步 i 都可得到 x_i 和 y_i 满足 $r_i = ax_i + by_i$ 。则有以下关系式:

$$\begin{aligned} a &= q_1b + r_1 & r_1 &= ax_1 + by_1 \\ b &= q_2r_1 + r_2 & r_2 &= ax_2 + by_2 \\ r_1 &= q_3r_2 + r_3 & r_3 &= ax_3 + by_3 \\ &\vdots & &\vdots \\ r_{n-2} &= q_nr_{n-1} + r_n & r_n &= ax_n + by_n \\ r_{n-1} &= q_{n+1}r_n + 0 & & \end{aligned}$$

- 从而得到 $d = r_n = ax_n + by_n = ax + by$, 即 $x = x_n, y = y_n$.



扩展欧几里得算法

$r_0 = b$		$x_0 = 0; y_0 = 1$	$b = ax_0 + by_0$
$r_1 = a \bmod b$ $q_1 = \lfloor a/b \rfloor$	$a = q_1b + r_1$	$x_1 = x_{-1} - q_1x_0 = 1$ $y_1 = y_{-1} - q_1y_0 = -q_1$	$r_1 = ax_1 + by_1$
$r_2 = b \bmod r_1$ $q_2 = \lfloor b/r_1 \rfloor$	$b = q_2r_1 + r_2$	$x_2 = x_0 - q_2x_1$ $y_2 = y_0 - q_2y_1$	$r_2 = ax_2 + by_2$
$r_3 = r_1 \bmod r_2$ $q_3 = \lfloor r_1/r_2 \rfloor$	$r_1 = q_3r_2 + r_3$	$x_3 = x_1 - q_3x_2$ $y_3 = y_1 - q_3y_2$	$r_3 = ax_3 + by_3$
• • •	• • •	• • •	• • •
$r_n = r_{n-2} \bmod r_{n-1}$ $q_n = \lfloor r_{n-2}/r_{n-1} \rfloor$	$r_{n-2} = q_n r_{n-1} + r_n$	$x_n = x_{n-2} - q_n x_{n-1}$ $y_n = y_{n-2} - q_n y_{n-1}$	$r_n = ax_n + by_n$
$r_{n+1} = r_{n-1} \bmod r_n = 0$ $q_{n+1} = \lfloor r_{n-1}/r_n \rfloor$	$r_{n-1} = q_{n+1} r_n + 0$		$d = \gcd(a, b) = r_n$ $x = x_n; y = y_n$

用扩展欧几里得算法求乘法逆元

- 如果 a 和 n 互素，那么 a 有模 n 的乘法逆元，即 a^{-1} 存在。
- 问题：**如何确定 a 的乘法逆元 a^{-1} ？
- 利用扩展欧几里得算法，存在整数 x 和 y ，满足

$$ax + ny = \gcd(a, n) = 1$$

两边同时模 n ，得到

$$(ax + ny) \bmod n = 1$$

进而得到

$$ax \bmod n = 1$$

所以 $a^{-1} = x$.

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

- 群、环和域
- 模算术
- 欧几里得算法
- 有限域

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

有限域 (Galois Fields)

- 有限域（也称伽罗瓦域）是包含有限个元素的域，用 $GF(q)$ 或 \mathbb{F}_q 表示包含 q 个元素的有限域。
- 有限域的阶（即元素个数）只能是素数 p 或素数的幂次 p^n 。
- 包含 p 个元素的有限域称为**素域**，记为 $GF(p)$ 。
- 包含 p^n 个元素的域称为**扩域**，记为 $GF(p^n)$ 。
- 关注两种有限域：有限域 $GF(p)$ 和有限域 $GF(2^n)$ 。

有限域 $GF(p)$

- 给定素数 p , 有限域 $GF(p)$ 的集合为 \mathbb{Z}_p , 运算为模 p 算术运算。
- 由于 \mathbb{Z}_p 中的所有非零整数都与 p 互素, 因此 \mathbb{Z}_p 中所有非零整数都有乘法逆元。
- 最简单的有限域是 $GF(2)$, 它的代数运算简述如下:

$+$	$0 \quad 1$	\times	$0 \quad 1$	w	$-w \quad w^{-1}$
0	$0 \quad 1$	0	$0 \quad 0$	0	$0 \quad -$
1	$1 \quad 0$	1	$0 \quad 1$	1	$1 \quad 1$
Addition			Multiplication		
			Inverses		

有限域 $GF(p)$ 的问题

- 所有加密算法都涉及整数集上的算术运算。
- 假如使用 8 比特来表示一个数，那么整数集为 \mathbb{Z}_{256} ；
- 由于 256 不是一个素数，这个集合不是一个域；
- 小于 256 的最大素数为 251，所以可以在域 \mathbb{Z}_{251} 上运算，但 251 ~ 255 范围内的数就不能使用，造成存储空间浪费。
- 所以希望寻找一个包含 2^n 个元素的集合，其上定义了加法和乘法使之成为一个域，给集合的每个元素赋值为 0 到 $2^n - 1$ 之间的唯一整数。
- $GF(2^n)$ 是一种含有 2^n 个元素的有限域。

有限域 $GF(2^n)$ 的定义

- **集合**: 所有次数小于 n 且系数为 0 或 1 的多项式, 其中每个多项式有如下形式:

$$f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0 = \sum_{i=0}^{n-1} a_i x^i$$

其中 $a_i \in \{0, 1\}$ 。

- 多项式 $m(x)$ 是系数为 0 或 1 且次数为 n 的**不可约多项式**, 也称为**素多项式**, 扮演模数的角色。
- **加法**: 多项式系数对应相加, 按照 \mathbb{Z}_2 上的模 2 加法, 等价于异或。
- **乘法**: 按照多项式乘法进行, 如果运算结果的次数大于 $n - 1$, 需要除以不可约多项式 $m(x)$, 得到的余式为乘法计算结果。

举例：有限域 $GF(2^3)$

- 有限域 $GF(2^3)$ 所在的集合包含 8 个多项式：

$$\{0, 1, x, x^2, x^2 + x, x + 1, x^2 + 1, x^2 + x + 1\}$$

- 需要选择次数为 3 的不可约多项式，仅有两个这样的多项式 $x^3 + x^2 + 1$ 和 $x^3 + x + 1$ 。
- 加法和乘法分别为多项式加法和多项式乘法，对应系数模 2。
- 考虑 $f(x) = x + 1$, $g(x) = x^2 + x + 1$, 取不可约多项式 $m(x) = x^3 + x + 1$ 。
- $f(x) + g(x) = x^2$ 。
- $f(x)g(x) = x^3 + 1$, 次数超过 3, 需模不可约多项式。
- $f(x)g(x) \bmod m(x) = x$ 。

举例：有限域 $GF(2^8)$

有限域 $GF(2^8)$, $m(x) = x^8 + x^4 + x^3 + x + 1$, 考虑两个多项式 $f(x) = x^6 + x^4 + x^2 + x + 1$ 和 $g(x) = x^7 + x + 1$ 。

$$\begin{aligned} f(x) + g(x) &= x^6 + x^4 + x^2 + x + 1 + x^7 + x + 1 \\ &= x^7 + x^6 + x^4 + x^2 \end{aligned}$$

$$\begin{aligned} f(x) \times g(x) &= x^{13} + x^{11} + x^9 + x^8 + x^7 \\ &\quad + x^7 + x^5 + x^3 + x^2 + x \\ &\quad + x^6 + x^4 + x^2 + x + 1 \\ &= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \end{aligned}$$

$$\begin{array}{r} x^5 + x^3 \\ \hline x^8 + x^4 + x^3 + x + 1 \end{array} \begin{array}{r} x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \\ \hline x^{13} + x^9 + x^8 + x^6 + x^5 \end{array} \begin{array}{r} + x^4 + x^3 \\ \hline x^{11} + x^7 + x^6 + x^4 + x^3 \\ \hline x^7 + x^6 + 1 \end{array}$$

Therefore, $f(x) \times g(x) \bmod m(x) = x^7 + x^6 + 1$.

有限域 $GF(2^n)$ 上乘法的另一种计算方式

- $GF(2^n)$ 内的一个多项式可以由它的二元系数 $(a_{n-1} \dots a_1 a_0)$ 唯一表示，因此 $GF(2^n)$ 内的每个元素可以用 n 位数来表示。
- 加法等价于按位异或，乘法通过左移及按位异或计算。
- 考虑有限域 $GF(2^8)$ ，使用不可约多项式 $x^8 + x^4 + x^3 + x + 1$
- 考虑两个元素 $A = (a_7 \dots a_0)$ 和 $B = (b_7 \dots b_0)$
- 则 $A + B = (c_7 \dots c_0)$ 其中 $c_i = a_i \oplus b_i$
- 考虑乘法 $\{02\} \cdot A$ ，即用 x 乘 A 对应的多项式：
 - 当 $a_7 = 0$ 时， $\{02\} \cdot A = (a_6 \dots a_0 0)$
 - 当 $a_7 = 1$ 时， $\{02\} \cdot A = (a_6 \dots a_0 0) \oplus (00011011)$
- 这样可以通过反复运用上面的规则计算 $A \cdot B$ 。

举例：有限域 $GF(2^8)$ 上的乘法

- 有限域 $GF(2^8)$, $m(x) = x^8 + x^4 + x^3 + x + 1$, 考虑两个多项式 $f(x) = x^6 + x^4 + x^2 + x + 1$ 和 $g(x) = x^7 + x + 1$ 。
- 计算 $f(x) \times g(x)$, 即 $(01010111) \times (10000011)$:

Redoing this in binary arithmetic, we need to compute $(01010111) \times (10000011)$. First, we determine the results of multiplication by powers of x :

$$\begin{aligned}
 (01010111) \times (00000010) &= (10101110) \\
 (01010111) \times (00000100) &= (01011100) \oplus (00011011) = (01000111) \\
 (01010111) \times (00001000) &= (10001110) \\
 (01010111) \times (00010000) &= (00011100) \oplus (00011011) = (00000111) \\
 (01010111) \times (00100000) &= (00001110) \\
 (01010111) \times (01000000) &= (00011100) \\
 (01010111) \times (10000000) &= (00111000)
 \end{aligned}$$

So,

$$\begin{aligned}
 (01010111) \times (10000011) &= (01010111) \times [(00000001) \oplus (00000010) \oplus (10000000)] \\
 &= (01010111) \oplus (10101110) \oplus (00111000) = (11000001)
 \end{aligned}$$

which is equivalent to $x^7 + x^6 + 1$.

GF(2³) 中的运算, $m(x) = x^3 + x + 1$

		000	001	010	011	100	101	110	111
+		0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
000	0	0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + 1$	$x^2 + x + 1$
001	1	1	0	$x + 1$	x	$x^2 + 1$	x^2	$x^2 + x + 1$	$x^2 + x$
010	x	x	$x + 1$	0	1	$x^2 + x$	$x^2 + x + 1$	x^2	$x^2 + 1$
011	$x + 1$	$x + 1$	x	1	0	$x^2 + x + 1$	$x^2 + x$	$x^2 + 1$	x^2
100	x^2	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$	0	1	x	$x + 1$
101	$x^2 + 1$	$x^2 + 1$	x^2	$x^2 + x + 1$	$x^2 + x$	1	0	$x + 1$	x
110	$x^2 + x$	$x^2 + x$	$x^2 + x + 1$	x^2	$x^2 + 1$	x	$x + 1$	0	1
111	$x^2 + x + 1$	$x^2 + x + 1$	$x^2 + x$	$x^2 + 1$	x^2	$x + 1$	x	1	0

		000	001	010	011	100	101	110	111
×		0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
000	0	0	0	0	0	0	0	0	0
001	1	0	1	x	$x + 1$	x^2	$x^2 + 1$	$x^2 + x$	$x^2 + x + 1$
010	x	0	x	x^2	$x^2 + x$	$x + 1$	1	$x^2 + x + 1$	$x^2 + 1$
011	$x + 1$	0	$x + 1$	$x^2 + x$	$x^2 + 1$	$x^2 + x + 1$	x^2	1	x
100	x^2	0	x^2	$x + 1$	$x^2 + x + 1$	$x^2 + x$	x	$x^2 + 1$	1
101	$x^2 + 1$	0	$x^2 + 1$	1	x^2	x	$x^2 + x + 1$	$x + 1$	$x^2 + x$
110	$x^2 + x$	0	$x^2 + x$	$x^2 + x + 1$	1	$x^2 + 1$	$x + 1$	x	x^2
111	$x^2 + x + 1$	0	$x^2 + x + 1$	$x^2 + 1$	x	1	$x^2 + 1$	x^2	$x + 1$

使用生成元定义 $GF(2^n)$

- 阶为 q 的有限域 \mathbb{F} 的生成元是域的一个元素，记为 g ，该元素的前 $q - 1$ 个幂构成了 \mathbb{F} 的所有非零元素，即域 \mathbb{F} 的元素为 $\{0, g^0, g^1, \dots, g^{q-2}\}$ 。
- 考虑由多项式 $m(x)$ 定义的域 \mathbb{F} ，如果 \mathbb{F} 内的一个元素 b 满足 $m(b) = 0$ ，则称 b 为多项式 $m(x)$ 的根。
- 可以证明一个不可约多项式的根 g 是这个不可约多项式定义的有限域的生成元。
- 通常，由不可约多项式 $m(x)$ 生成的域 $GF(2^n)$ ，有 $g^n = m(g) - g^n$ 。计算 g^{n+1} 到 g^{2^n-2} 。域的元素对应 g^0 到 g^{2^n-2} ，外加 0。域元素的乘法用等式 $g^k = g^{k \bmod (2^n-1)}$ 计算。

由 $m(x) = x^3 + x + 1$ 生成的域 $GF(2^3)$

- $m(g) = 0 \Rightarrow g^3 + g + 1 = 0$, 所以 $g^3 = g + 1$ 。
- $g^4 = g \cdot g^3 = g^2 + g$ 。
- $g^5 = g \cdot g^4 = g^3 + g^2 = g^2 + g + 1$ 。
- $g^6 = g \cdot g^5 = g^3 + g^2 + g = g^2 + 1$ 。
- $g^7 = g^3 + g = 1 = g^0$, 开始循环, 一般地 $g^k = g^{k \bmod (2^n-1)}$ 。

Power Representation	Polynomial Representation	Binary Representation	Decimal (Hex) Representation
0	0	000	0
$g^0 (= g^7)$	1	001	1
g^1	g	010	2
g^2	g^2	100	4
g^3	$g + 1$	011	3
g^4	$g^2 + g$	110	6
g^5	$g^2 + g + 1$	111	7
g^6	$g^2 + 1$	101	5

使用生成元的 $GF(2^3)$ 算术

		000	001	010	100	011	110	111	101
		0	1	G	g^2	g^3	g^4	g^5	g^6
+		0	1	G	g^2	$g + 1$	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$
000	0	0	1	G	g^2	$g + 1$	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$
001	1	1	0	$g + 1$	$g^2 + 1$	g	$g^2 + g + 1$	$g^2 + g$	g^2
010	g	g	$g + 1$	0	$g^2 + g$	1	g^2	$g^2 + 1$	$g^2 + g + 1$
100	g^2	g^2	$g^2 + 1$	$g^2 + g$	0	$g^2 + g + 1$	g	$g + 1$	1
011	g^3	$g + 1$	g	1	$g^2 + g + 1$	0	$g^2 + 1$	g^2	$g^2 + g$
110	g^4	$g^2 + g$	$g^2 + g + 1$	g^2	g	$g^2 + 1$	0	1	$g + 1$
111	g^5	$g^2 + g + 1$	$g^2 + g$	$g^2 + 1$	$g + 1$	g^2	1	0	g
101	g^6	$g^2 + 1$	g^2	$g^2 + g + 1$	1	$g^2 + g$	$g + 1$	g	0

		000	001	010	100	011	110	111	101
		0	1	G	g^2	g^3	g^4	g^5	g^6
×		0	1	G	g^2	g^3	g^4	g^5	g^6
000	0	0	0	0	0	0	0	0	0
001	1	0	1	G	g^2	$g + 1$	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$
010	g	0	g	g^2	$g + 1$	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$	1
100	g^2	0	g^2	$g + 1$	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$	1	g
011	g^3	0	$g + 1$	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$	1	g	g^2
110	g^4	0	$g^2 + g$	$g^2 + g + 1$	$g^2 + 1$	1	g	g^2	$g + 1$
111	g^5	0	$g^2 + g + 1$	$g^2 + 1$	1	g	g^2	$g + 1$	$g^2 + g$
101	g^6	0	$g^2 + 1$	1	g	g^2	$g + 1$	$g^2 + g$	$g^2 + g + 1$

多项式欧几里得算法

- 类似于计算两个整数最大公因子的欧几里得算法

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

- 计算两个多项式 $a(x)$ 和 $b(x)$ 最大公因式的欧几里得算法为

$$\gcd(a(x), b(x)) = \gcd(b(x), a(x) \bmod b(x))$$

Euclidean Algorithm for Polynomials	
Calculate	Which satisfies
$r_1(x) = a(x) \bmod b(x)$	$a(x) = q_1(x)b(x) + r_1(x)$
$r_2(x) = b(x) \bmod r_1(x)$	$b(x) = q_2(x)r_1(x) + r_2(x)$
$r_3(x) = r_1(x) \bmod r_2(x)$	$r_1(x) = q_3(x)r_2(x) + r_3(x)$
•	•
•	•
•	•
$r_n(x) = r_{n-2}(x) \bmod r_{n-1}(x)$	$r_{n-2}(x) = q_n(x)r_{n-1}(x) + r_n(x)$
$r_{n+1}(x) = r_{n-1}(x) \bmod r_n(x) = 0$	$r_{n-1}(x) = q_{n+1}(x)r_n(x) + 0$ $d(x) = \gcd(a(x), b(x)) = r_n(x)$

多项式欧几里得算法

- $a(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1, b(x) = x^4 + x^2 + x + 1$,
计算 $\gcd(a(x), b(x))$.
- 首先用 $a(x)$ 除以 $b(x)$, 得余式 $r_1(x) = x^3 + x^2 + 1$

$$\begin{array}{r}
 \frac{x^2 + x}{x^4 + x^2 + x + 1} \\
 \hline
 x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 \\
 \frac{x^6 + x^4 + x^3 + x^2}{x^5} \\
 \hline
 x^5 + x^3 + x^2 + x \\
 \frac{x^5 + x^3 + x^2 + x}{x^3 + x^2} \\
 \hline
 x^3 + x^2 + 1
 \end{array}$$

- 再用 $b(x)$ 除以 $r_1(x)$, 整除, 所以 $\gcd(a(x), b(x)) = r_1(x)$.

$$\begin{array}{r}
 \frac{x + 1}{x^3 + x^2 + 1} \\
 \hline
 x^4 + x^2 + x + 1 \\
 \frac{x^4 + x^3}{x^3 + x^2} \\
 \hline
 x^3 + x^2 + 1 \\
 \frac{x^3 + x^2}{x^3 + x^2} \\
 \hline
 1
 \end{array}$$

多项式扩展欧几里得算法

- 类似的，扩展欧几里得算法在计算两个多项式 $a(x)$ 和 $b(x)$ 最大公因式的同时能够得到两个多项式 $v(x)$ 和 $w(x)$ ，满足

$$a(x)v(x) + b(x)w(x) = \gcd(a(x), b(x))$$

- 当 $\gcd(a(x), b(x)) = 1$ 时，存在 $a(x) \bmod b(x)$ 或 $b(x) \bmod a(x)$ 的乘法逆元，即为：

$$a(x)^{-1} \bmod b(x) = v(x)$$

或

$$b(x)^{-1} \bmod a(x) = w(x)$$

多项式扩展欧几里得算法

例

- 已知 $a(x) = x^8 + x^4 + x^3 + x + 1$, $b(x) = x^7 + x + 1$, 计算 $b(x)^{-1} \bmod a(x) = ?$

Initialization	$a(x) = x^8 + x^4 + x^3 + x + 1; v_{-1}(x) = 1; w_{-1}(x) = 0$ $b(x) = x^7 + x + 1; v_0(x) = 0; w_0(x) = 1$
Iteration 1	$q_1(x) = x; r_1(x) = x^4 + x^3 + x^2 + 1$ $v_1(x) = 1; w_1(x) = x$
Iteration 2	$q_2(x) = x^3 + x^2 + 1; r_2(x) = x$ $v_2(x) = x^3 + x^2 + 1; w_2(x) = x^4 + x^3 + x + 1$
Iteration 3	$q_3(x) = x^3 + x^2 + x; r_3(x) = 1$ $v_3(x) = x^6 + x^2 + x + 1; w_3(x) = x^7$
Iteration 4	$q_4(x) = x; r_4(x) = 0$ $v_4(x) = x^7 + x + 1; w_4(x) = x^8 + x^4 + x^3 + x + 1$
Result	$d(x) = r_3(x) = \gcd(a(x), b(x)) = 1$ $w(x) = w_3(x) = (x^7 + x + 1)^{-1} \bmod (x^8 + x^4 + x^3 + x + 1) = x^7$

目录

- 1 分组密码的基本原理
- 2 数据加密标准
- 3 DES 的安全性
- 4 数论基础
- 5 高级数据加密标准
- 6 多重加密
- 7 分组密码的工作模式

AES 的起源

- DES 不安全, 建议用 3DES, 密钥 168 位, 抵御密码分析攻击, 但是 3DES 用软件实现速度较慢, 分组仅 64 位。
- 美国国家标准技术协会 NIST 在 1997 年征集新标准, 要求明文分组长度 128 位, 密钥长 128、192 或 256 位。
- 1998 年 6 月, 15 种候选算法通过了第一轮评估; 1999 年 8 月, 仅有 5 个候选算法通过了第二轮评估。
- 2000 年 10 月, NIST 选择 Rijndael 算法作为 AES 算法, Rijndael 的作者是比利时的密码学家 Joan Daemen 博士和 Vincent Rijmen 博士。
- 2001 年 11 月 NIST 发布了 AES 的最终标准 FIPS PUB 197。

AES 的评估准则

- **安全性**：指密码分析方法分析一个算法所需的代价；
- **成本**：期望 AES 能够广泛应用于各种实际应用，计算效率要高；
- **算法和执行特征**：算法灵活性、适合于多种硬件和软件方式的实现、简洁性，便于分析安全性。

Rijndael 的评估结果

- 一般安全性：依赖于密码学界的公共安全分析
- 软件实现：软件执行速度，跨平台执行能力及密钥长度改变时速度变化
- 受限空间环境：在诸如智能卡中的应用
- 硬件实现：硬件执行提高执行速度或缩短代码长度
- 对执行的攻击：抵御密码分析攻击
- 加密与解密
- 密钥灵活性：快速改变密钥长度的能力
- 其他的多功能性和灵活性
- 指令级并行执行的潜力

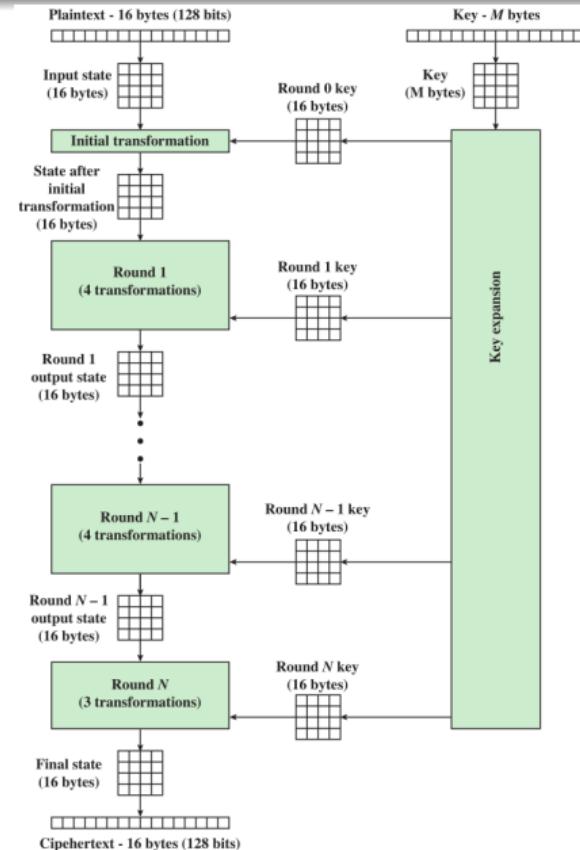
AES 参数选择

- AES 的分组长度为 128 位。
- 密钥长度可以是 128/192/256 的任意一种，根据使用的密钥长度，分为 AES-128，AES-192 或 AES-256。
- 接下来主要以 AES-128 为例进行讲述。

Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

AES 加密过程总体结构

- AES 采用非 Feistel 结构。
- 输入 128 位分组，表示为 4×4 字节方阵，称为状态数组。
- 密钥被表示为 4×4 密钥方阵，扩展为密钥字阵列，共包含 $N + 1$ 个密钥方阵。
- 加密由 N 轮构成，前 $N - 1$ 轮由 4 种不同变换组成，最后一轮仅包含 3 种变换。
- 每一轮修改状态数组，最后一轮输出密文。

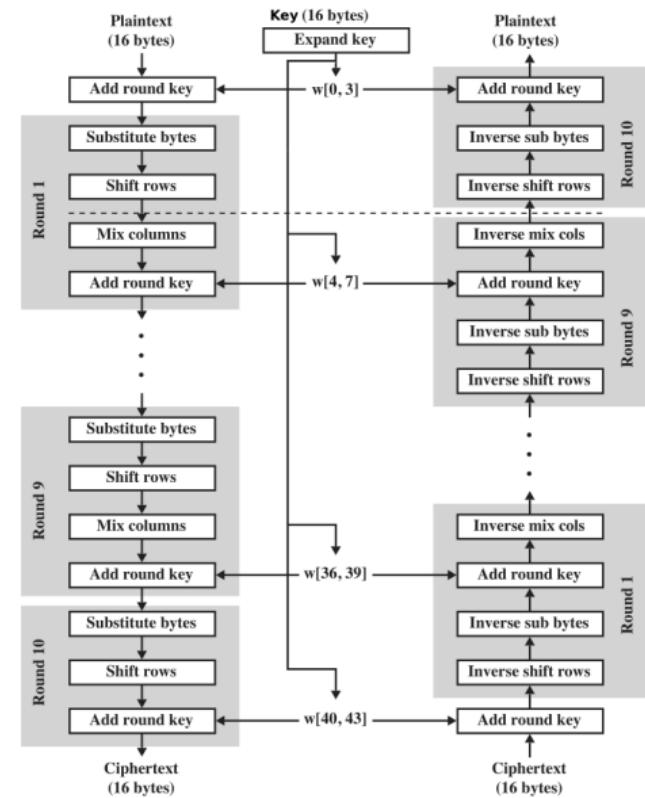


状态数组的修改及密钥扩展



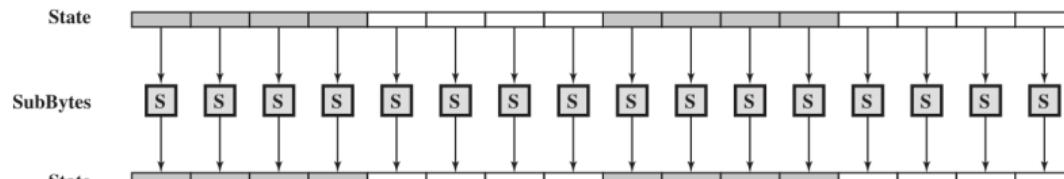
AES 详细结构

- 输入密钥被扩展为 44 字节数组 $w[i]$ 。
- 每轮运算包含 4 种操作：
 - 字节代替：S 盒完成字节代替
 - 行移位：一个简单置换操作
 - 列混淆：域 $GF(2^8)$ 上的算术
 - 轮密钥加：当前分组与扩展密钥一部分按位异或
- 每种操作均可逆，仅仅在轮密钥加阶段使用密钥。
- AES 的加解密算法不同，加解密的最后一轮均只包含 3 种操作。

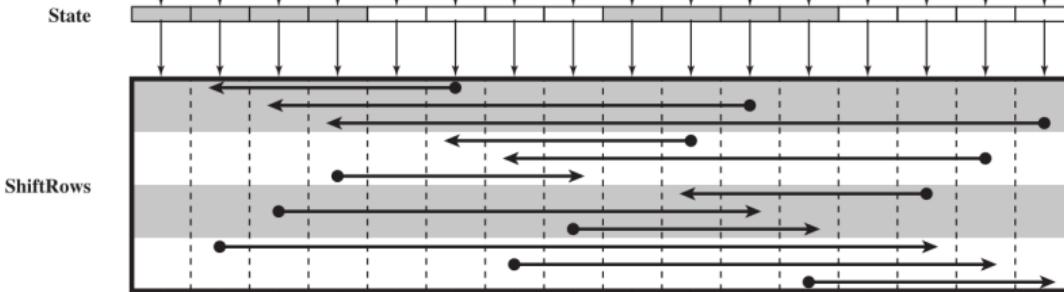


AES 的一轮加密过程

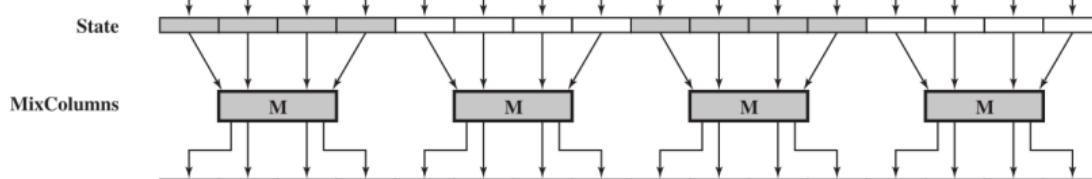
字节代替



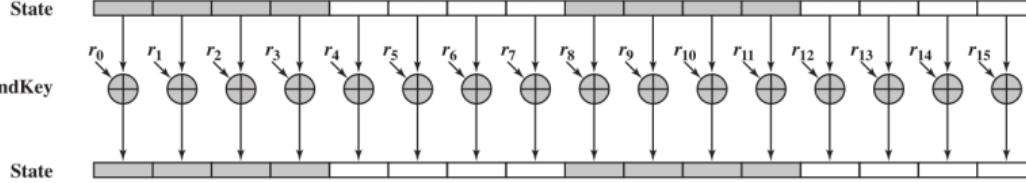
行移位



列混淆

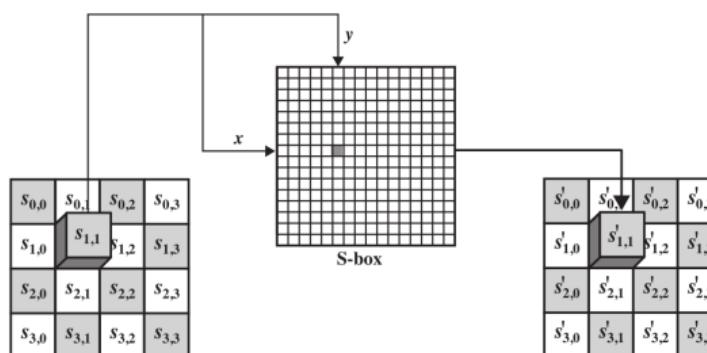


轮密钥加 AddRoundKey



字节代替变换

- 字节代替变换是一个查表操作，实现分组字节到字节的代替。
- S 盒是一个 16×16 字节表，包含了所有 8 位的置换值。
- 每个字节的左 4 位选择行，右 4 位选择列来查表替换。



EA	04	65	85
83	45	5D	96
5C	33	98	B0
F0	2D	AD	C5

→

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

S 盒

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

逆 S 盒

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

S 盒的构造方法

- 按字节值的升序逐行初始化 S 盒。第 x 行 y 列的值为 $\{xy\}$ 。
- 把 S 盒中的每个字节映射为它在域 $GF(2^8)$ 中的逆，使用素多项式 $m(x) = x^8 + x^4 + x^3 + x + 1$ ，其中 $\{00\}$ 映射为 $\{00\}$ 。
- 把 S 盒中的每个字节表示为 $b_7b_6 \cdots b_0$ ，对每个字节的每位做如下变换
 $b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$
 其中 c_i 指常数 $\{63\}$ 的第 i 位，即 $c = \{63\} = (01100011)$ 。
- 可以表示为下式，注意行和列相乘后再进行按位异或。

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

逆 S 盒的构造方法

- 求上一变换的逆变换，然后再求其在 $GF(2^8)$ 上的乘法逆：

$$b'_i = b_{(i+2) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus d_i$$

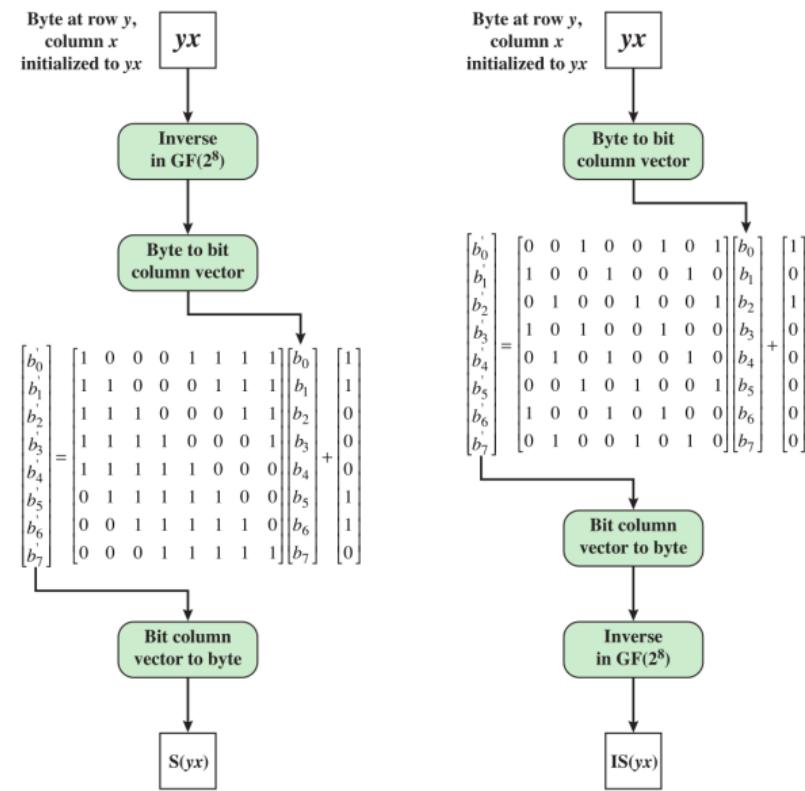
其中字节 $d = \{05\} = (00000101)$ 。表示为矩阵形式：

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

- 令字节代替变换和逆变换中的矩阵分别为 X 和 Y ，常量 c 、 d 的向量表示为 C 和 D 。
- $B' = XB \oplus C \Rightarrow YB' \oplus D = Y(XB \oplus C) \oplus D = YXB \oplus YC \oplus D = B$
- 可以发现 $YX = I$, $YC = D$, 故 $YB' \oplus D = B$

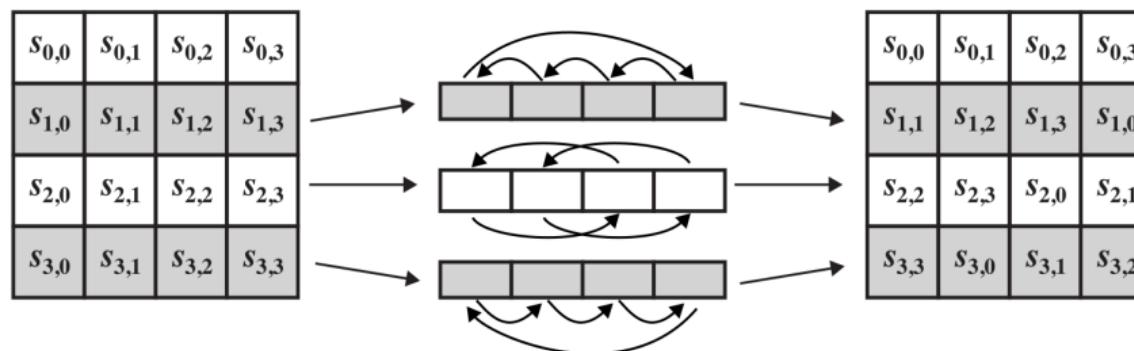
S 盒和逆 S 盒

- S 盒被设计成能防止已知的各种密码分析攻击。
 - 输入和输出的相关性很低，输出不是输入的线性函数，非线性度的产生是由于使用了乘法逆。
 - 常量的选择使得 S 盒中没有不动点，即 $S[a] = a$ ，也没有“反不动点”，即 $S[a] = \bar{a}$ ， \bar{a} 与 a 逐位取反。
 - S 盒可逆，即 $S^{-1}[S[a]] = a$ ，但不自逆，即 $S[a] \neq S^{-1}[a]$ 。



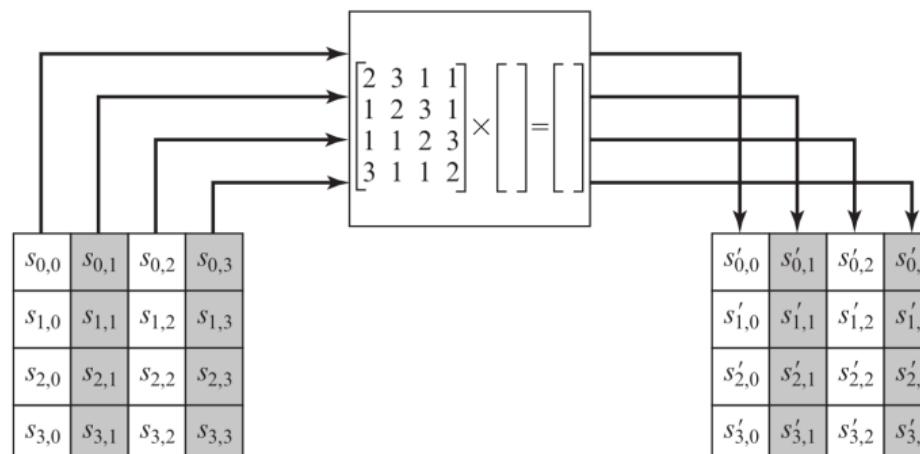
行移位变换

- **正向行移位变换**：第 1 行保持不变。第 2 行循环左移 1 个字节。第 3 行循环左移 2 个字节。第 4 行循环左移 3 个字节。
- **逆向行移位变换**进行相反的移位以实现解密。
- 因为状态矩阵是按列处理的，行移位变换就把字节在列之间进行了置换。



列混淆变换

- 正向列混淆变换对每列独立进行操作。
- 每列中的每个字节被映射为一个新值，由该列中的 4 个字节通过函数变换得到。
- 可以用状态矩阵乘法表示，使用素多项式
 $m(x) = x^8 + x^4 + x^3 + x + 1$ ，加法和乘法都定义在 $GF(2^8)$ 上。



列混淆变换

- 写成矩阵形式为

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix} = \begin{bmatrix} s'_{00} & s'_{01} & s'_{02} & s'_{03} \\ s'_{10} & s'_{11} & s'_{12} & s'_{13} \\ s'_{20} & s'_{21} & s'_{22} & s'_{23} \\ s'_{30} & s'_{31} & s'_{32} & s'_{33} \end{bmatrix}$$

例如 $s'_{0j} = 2s_{0j} \oplus 3s_{1j} \oplus s_{2j} \oplus s_{3j}$

- 逆向列混淆变换也由矩阵乘法定义：

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \cdot \begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix} = \begin{bmatrix} s'_{00} & s'_{01} & s'_{02} & s'_{03} \\ s'_{10} & s'_{11} & s'_{12} & s'_{13} \\ s'_{20} & s'_{21} & s'_{22} & s'_{23} \\ s'_{30} & s'_{31} & s'_{32} & s'_{33} \end{bmatrix}$$

轮密钥加变换

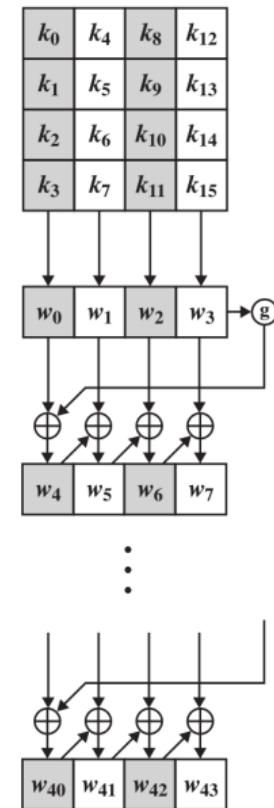
- 正向轮密钥加变换中 128 位的状态矩阵按位与 128 位的密钥异或。
- 都是基于状态矩阵列的操作，即把状态矩阵的一列中的 4 个字节与轮密钥的 1 个字进行异或。
- 逆向轮密钥加变换与正向轮密钥加变换相同，因为异或操作是其本身的逆。
- 轮密钥加变换非常简单，却能影响状态矩阵中的每一位。

密钥扩展算法

- AES 密钥扩展算法的输入是 4 字，输出 44 字。
- 输入密钥直接被复制到扩展密钥数组的前 4 个字。
- 然后每次用 4 字填充扩展密钥数组余下的部分：

$$w[i] = \begin{cases} w[i-4] \oplus w[i-1] & i \bmod 4 \neq 0 \\ w[i-4] \oplus g(w[i-1]) & i \bmod 4 = 0 \end{cases}$$

其中 g 函数对一个字进行复杂变换，输出另外一个字。

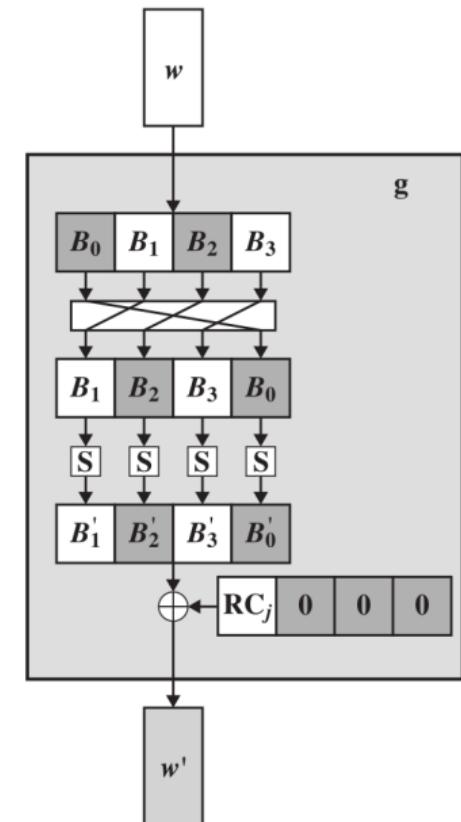


密钥扩展算法: g 函数

- 首先, **字循环**使一个字中的 4 个字节循环左移一个字节, 即将输入字 $[B_0, B_1, B_2, B_3]$ 变为 $[B_1, B_2, B_3, B_0]$ 。
- 然后, **字代替**利用 S 盒对输入字中的每个字节进行字节代替。
- 最后, 将前两步的结果与轮常量 $Rcon_j = [RC_j, 0, 0, 0]$ 相**异或**。
 - 轮常量是一个字, 字的最右边 3 个字节总为 0
 - 轮常量第一个字节定义为

$$RC_1 = 1, \quad RC_j = 2 \cdot RC_{j-1}$$

j	1	2	3	4	5	6	7	8	9	10
RC_j	01	02	04	08	10	20	40	80	1B	36



密钥扩展算法

```
KeyExpansion(byte key[16], word w[44]){
    word tmp;
    for(i=0; i<4; ++i)
        w[i] = (key[4*i], key[4*i+1], key[4*i+2],
                 key[4*i+3]);
    for(i=4; i<44; ++i){
        tmp = w[i-1];
        if(i mod 4 ==0)
            tmp = SubWord(RotWord(tmp))^Rcon[i/4];
        w[i] = w[i-4]^tmp;
    }
}
```

AES 举例

Start of Round	After SubBytes	After ShiftRows	After MixColumns	Round Key
01 89 fe 76 23 ab dc 54 45 cd ba 32 67 ef 98 10				0f 47 0c af 15 d9 b7 7f 71 e8 ad 67 c9 59 d6 98
0e ce f2 d9 36 72 6b 2b 34 25 17 55 ae b6 4e 88	ab 8b 89 35 05 40 7f f1 18 3f f0 fc e4 4e 2f c4	ab 8b 89 35 40 7f f1 05 f0 fc 18 3f c4 e4 4e 2f	b9 94 57 75 e4 8e 16 51 47 20 9a 3f c5 d6 f5 3b	dc 9b 97 38 90 49 fe 81 37 df 72 15 b0 e9 3f a7
65 0f c0 4d 74 c7 e8 d0 70 ff e8 2a 75 3f ca 9c	4d 76 ba e3 92 c6 9b 70 51 16 9b e5 9d 75 74 de	4d 76 ba e3 c6 9b 70 92 9b e5 51 16 de 9d 75 74	8e 22 db 12 b2 f2 dc 92 df 80 f7 c1 2d c5 1e 52	d2 49 de e6 c9 80 7e ff 6b b4 c6 d3 b7 5e 61 c6
5c 6b 05 f4 7b 72 a2 6d b4 34 31 12 9a 9b 7f 94	4a 7f 6b bf 21 40 3a 3c 8d 18 c7 c9 b8 14 d2 22	4a 7f 6b bf 40 3a 3c 21 c7 c9 8d 18 22 b8 14 d2	b1 c1 0b cc ba f3 8b 07 f9 1f 6a c3 1d 19 24 5c	c0 89 57 b1 af 2f 51 ae df 6b ad 7e 39 67 06 c0
71 48 5c 7d 15 dc da a9 26 74 c7 bd 24 7e 22 9c	a3 52 4a ff 59 86 57 d3 f7 92 c6 7a 36 f3 93 de	a3 52 4a ff 86 57 d3 59 c6 7a f7 92 de 36 f3 93	d4 11 fe 0f 3b 44 06 73 cb ab 62 37 19 b7 07 ec	2c a5 f2 43 5c 73 22 8c 65 0e a3 dd f1 96 90 50
f8 b4 0c 4c 67 37 24 ff ae a5 c1 ea e8 21 97 bc	41 8d fe 29 85 9a 36 16 e4 06 78 87 9b fd 88 65	41 8d fe 29 9a 36 16 85 78 87 e4 06 65 9b fd 88	2a 47 c4 48 83 e8 18 ba 84 18 27 23 eb 10 0a f3	58 fd 0f 4c 9d ee cc 40 36 38 9b 46 eb 7d ed bd

AES 举例

72 ba cb 04 1e 06 d4 fa b2 20 bc 65 00 6d e7 4e	40 f4 1f f2 72 6f 48 2d 37 b7 65 4d 63 3c 94 2f	40 f4 1f f2 6f 48 2d 72 65 4d 37 b7 2f 63 3c 94	7b 05 42 4a 1e d0 20 40 94 83 18 52 94 c4 43 fb	71 8c 83 cf c7 29 e5 a5 4c 74 ef a9 c2 bf 52 ef
0a 89 c1 85 d9 f9 c5 e5 d8 f7 f7 fb 56 7b 11 14	67 a7 78 97 35 99 a6 d9 61 68 68 0f b1 21 82 fa	67 a7 78 97 99 a6 d9 35 68 0f 61 68 fa b1 21 82	ec 1a c0 80 0c 50 53 c7 3b d7 00 ef b7 22 72 e0	37 bb 38 f7 14 3d d8 7d 93 e7 08 a1 48 f7 a5 4a
db a1 f8 77 18 6d 8b ba a8 30 08 4e ff d5 d7 aa	b9 32 41 f5 ad 3c 3d f4 c2 04 30 2f 16 03 0e ac	b9 32 41 f5 3c 3d f4 ad 30 2f c2 04 ac 16 03 0e	b1 1a 44 17 3d 2f ec b6 0a 6b 2f 42 9f 68 f3 b1	48 f3 cb 3c 26 1b c3 be 45 a2 aa 0b 20 d7 72 38
f9 e9 8f 2b 1b 34 2f 08 4f c9 85 49 bf bf 81 89	99 1e 73 f1 af 18 15 30 84 dd 97 3b 08 08 0c a7	99 1e 73 f1 18 15 30 af 97 3b 84 dd a7 08 08 0c	31 30 3a c2 ac 71 8c c4 46 65 48 eb 6a 1c 31 62	fd 0e c5 f9 0d 16 d5 6b 42 e0 4a 41 cb 1c 6e 56
cc 3e ff 3b a1 67 59 af 04 85 02 aa a1 00 5f 34	4b b2 16 e2 32 85 cb 79 f2 97 77 ac 32 63 cf 18	4b b2 16 e2 85 cb 79 32 77 ac f2 97 18 32 63 cf		b4 ba 7f 86 8e 98 4d 26 f3 13 59 18 52 4e 20 76
ff 08 69 64 0b 53 34 14 84 bf ab 8f 4a 7c 43 b9				

AES 的实现

- 可以在 8 位处理器上非常有效地实现：
 - 字节代替在字节级别上操作，只要求一个 256 字节的表。
 - 行移位是简单的移字节操作。
 - 轮密钥加是按位异或操作。
 - 列混淆变换要求在域 $GF(2^8)$ 上的乘法，所有的操作都是基于字节的，只要简单地查表即可。
- 可以在 32 位处理器上非常有效地实现：
 - 将操作定义在 32 位的字上。
 - 事先准备好 4 张 256 字的表。
 - 每一轮通过查表并加上 4 次异或即可计算每一列的值。
 - 需要 4KB 空间来存储这 4 张表。

AES 在 32 位处理器上的高效实现

每一轮的四种变换可以归纳为以下运算：

3. 列混淆

1. 字节代替

$$b_{i,j} = S[a_{i,j}]$$

$$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}$$

2. 行移位

$$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{bmatrix}$$

4. 轮密钥加

$$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

AES 在 32 位处理器上的高效实现

可以写成一个式子：

$$\begin{aligned}
 \begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} &= \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} S[a_{0,j}] \\ S[a_{1,j-1}] \\ S[a_{2,j-2}] \\ S[a_{3,j-3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix} \\
 &= \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \cdot S[a_{0,j}] \right) \oplus \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot S[a_{1,j-1}] \right) \\
 &\quad \oplus \left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \cdot S[a_{2,j-2}] \right) \oplus \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \cdot S[a_{3,j-3}] \right) \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}
 \end{aligned}$$

AES 在 32 位处理器上的高效实现

- 定义 4 个 256 字的表 $T_i, i = 0, 1, 2, 3$ 。
- 每个表输入一个 $0 \sim 255$ 范围内的数 x ，输出一个字 $T_i[x]$ 。
- 四个表可以提前计算好，需要存储空间 4KB。

$$\begin{aligned}
 T_0[x] &\triangleq \begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot S[x] & T_1[x] &\triangleq \begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \cdot S[x] \\
 T_2[x] &\triangleq \begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \cdot S[x] & T_3[x] &\triangleq \begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \cdot S[x]
 \end{aligned}$$

AES 在 32 位处理器上的高效实现

- 对状态数组一列进行一轮运算可以简化为 4 次查表操作和 4 次异或运算：

$$\begin{bmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{bmatrix} = T_0[s_{0,j}] \oplus T_1[s_{1,j-1}] \oplus T_2[s_{2,j-2}] \oplus T_3[s_{3,j-3}] \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$$

- 这种紧凑、高效的实现方式是 Rijndael 能被选为 AES 的重要原因之一。

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

5 高级数据加密标准

6 多重加密

- 多重加密提出的背景
- 双重加密及其安全性分析
- 三重加密及其安全性分析

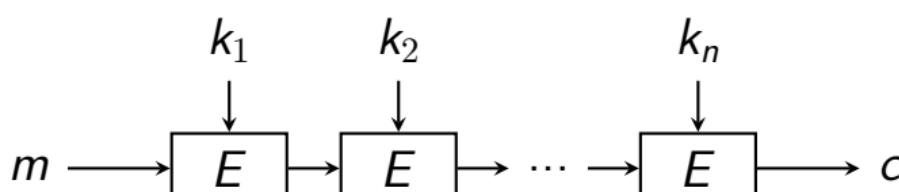
7 分组密码的工作模式

目录

- 1 分组密码的基本原理
- 2 数据加密标准
- 3 DES 的安全性
- 4 数论基础
- 5 高级数据加密标准
- 6 多重加密
 - 多重加密提出的背景
 - 双重加密及其安全性分析
 - 三重加密及其安全性分析
- 7 分组密码的工作模式

多重加密提出的背景

- DES 的密钥长度为 56 位，已经不安全，需要寻找更安全的加密方法。
- 例如，DES 密钥的穷举攻击目前仅需要 10 小时。
- 在高级加密标准 AES 出现之前，**多重加密**是一种增强加密算法安全性的解决方案。
- 即多次使用同一个加密算法，对明文反复加密。
- 多重加密的**优点**：可以利用现有软硬件资源。



目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

5 高级数据加密标准

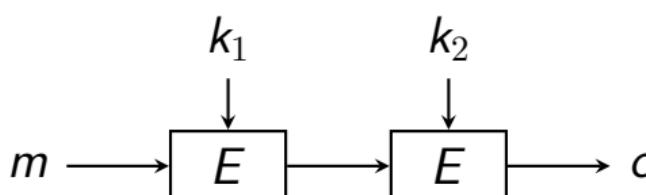
6 多重加密

- 多重加密提出的背景
- **双重加密及其安全性分析**
- 三重加密及其安全性分析

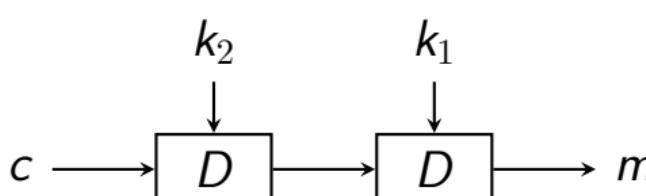
7 分组密码的工作模式

双重加密与 2DES

- 多重加密最简单的形式是双重加密，使用两个密钥加密。
- 加密: $c = E(k_2, E(k_1, m))$

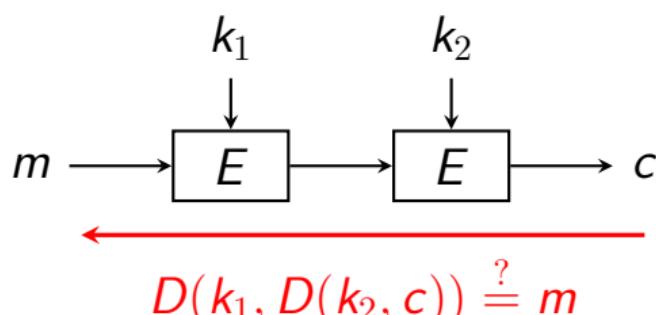


- 解密: $m = D(k_1, D(k_2, c))$



- 对于 DES，使用双重加密的 2DES 密钥长度为 112 位。

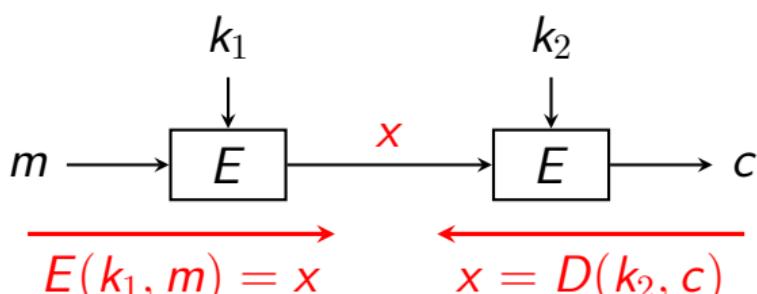
对双重加密的穷举攻击



- 给定密文 c ，依次尝试所有可能的密钥 k_2 和 k_1 ，直到发现明文 m 。
- 需要尝试 $2^{56} \times 2^{56} = 2^{112}$ 次。
- 有没有更有效的攻击手段？

中间相遇攻击 (Meet-in-the-Middle Attack)

- 中间相遇攻击对任何使用双重加密的分组密码都有效。

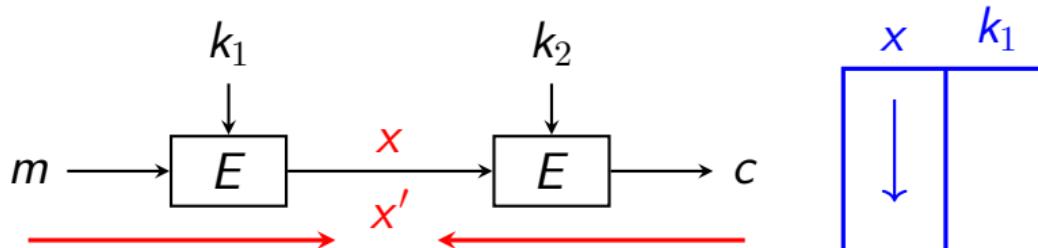


```
for k1 in K
    for k2 in K
        // ...
```

\Rightarrow

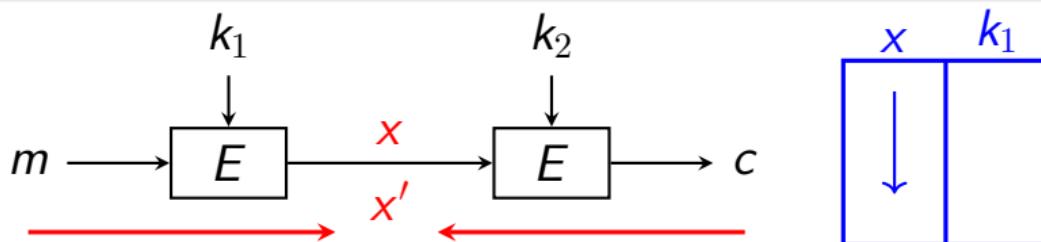
```
for k1 in K
    // ...
    for k2 in K
        // ...
```

中间相遇攻击的攻击步骤



- ① 搜集尽可能多的明密文对 (m, c) ;
- ② 遍历密钥 k_1 对 m 加密, 得到的结果按 \times 排序后保存在表中;
- ③ 遍历密钥 k_2 对 c 解密, 每解一次密, 在表中匹配;
- ④ 如产生匹配, 则找到一对可能密钥, 然后用这两个密钥对一个新的明密文对进行验证, 若通过则说明密钥正确。

中间相遇攻击攻击 2DES 的复杂度分析



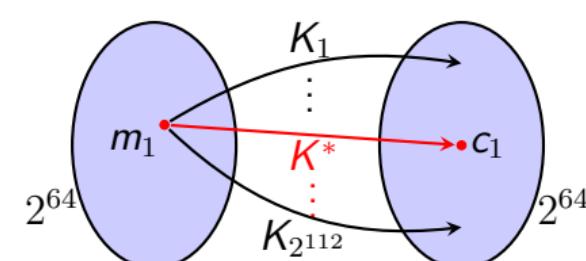
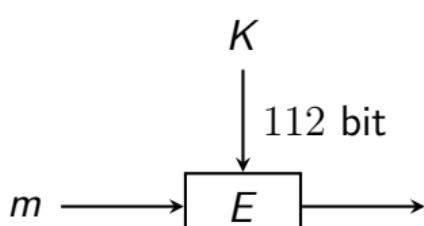
- 第 2 步和第 3 步的时间复杂度: $2^{56} + 2^{56} = 2^{57}$
- 空间复杂度: $(56 + 64) \times 2^{56}$ bit
- 第 4 步分析:
 - 使用一对 (m, c) 找到的错误密钥平均个数为: $2^{112}/2^{64} = 2^{48}$
 - 使用两对 (m, c) 找到的错误密钥平均个数为: $2^{48}/2^{64} = 2^{-16} \approx 0$

2DES 的安全性

？相较于攻击 DES 的最差时间复杂度 2^{56} , 2DES 的加密强度并没有提高很多！

中间相遇攻击第 4 步需要试多少对 (m, c) ?

- 给定一个明密文对 (m_1, c_1) ，分组长度 64 位，2DES 密钥长度 112 位，所以会存在 2^{112} 种映射将 m_1 映射为密文。
- 由于密文空间大小仅仅为 2^{64} ，所以平均会有 $2^{112}/2^{64} = 2^{48}$ 种映射将 m_1 映射为 c_1 。
- 即每个明密文对平均存在 2^{48} 个映射，其中 $2^{48} - 1 \approx 2^{48}$ 个映射是错误的。
- 使用第二个明密文对 (m_2, c_2) 验证时，会从 2^{48} 个映射中找到正确的映射，找到错误映射的平均个数为 $2^{48}/2^{64} = 2^{-16} \approx 0$ 。



目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

5 高级数据加密标准

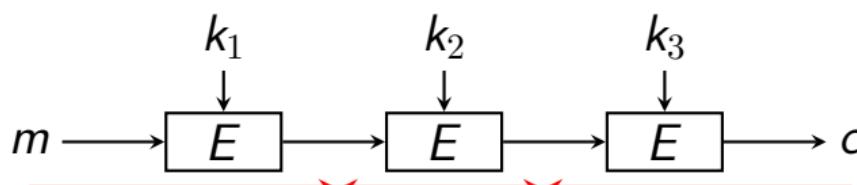
6 多重加密

- 多重加密提出的背景
- 双重加密及其安全性分析
- 三重加密及其安全性分析

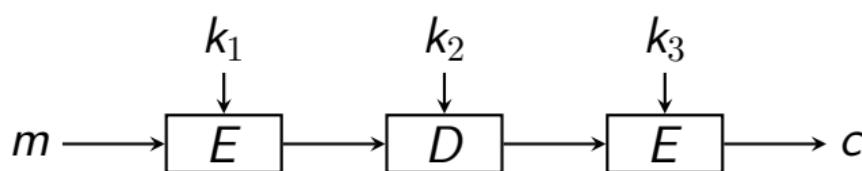
7 分组密码的工作模式

三重加密与 3DES

- 为了进一步抵抗密码分析，可以使用三重加密：



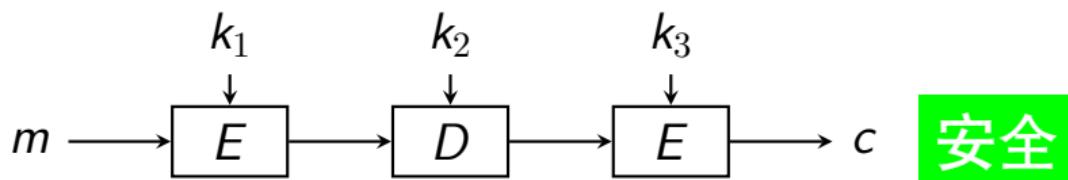
- 中间相遇攻击攻击的时间复杂度变为 $O(2^{112})$ 。
- 实际中会使用如下更灵活的三重加密方案 (RFC 1851)：



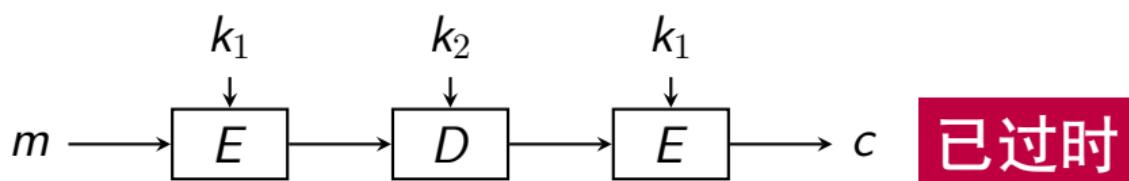
- $k_1 \neq k_2 \neq k_3$ 强三重加密，密钥长度 $3 \times 56 = 168$
- $k_1 = k_3 \neq k_2$ 普通三重加密，密钥长度 $2 \times 56 = 112$
- $k_1 = k_2 = k_3$ 等价于普通分组加密，密钥长度 56

3DES 现状

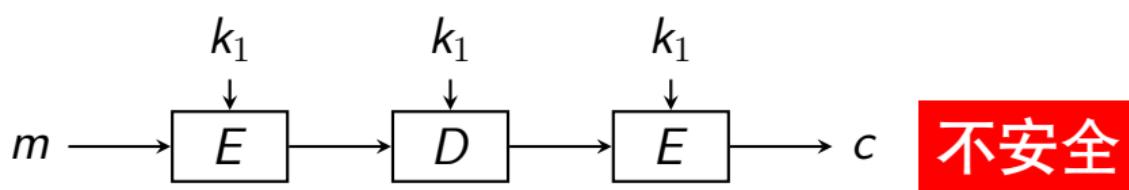
- 使用工作模式 1 的 3DES 目前仍然被广泛应用；



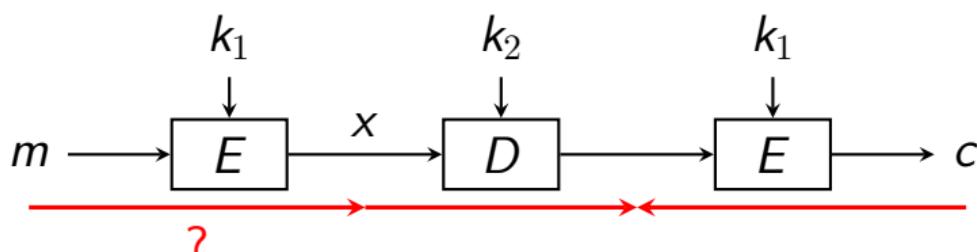
- 使用工作模式 2 的 3DES 于 2017 年被认为已过时；



- 使用工作模式 3 的 3DES 等价于普通 DES，不安全。

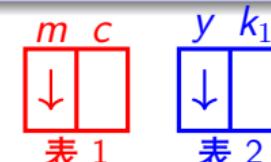
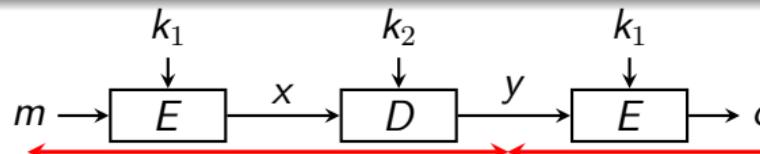


针对 3DES 的已知明文攻击



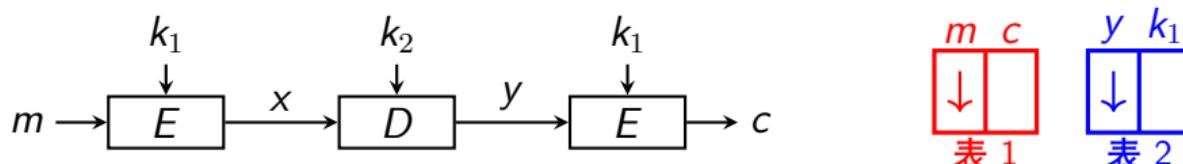
- 如果已知 x 和 c ，那么对三重加密的攻击可以转化为对二重加密的攻击；
- 当然，只要攻击者不知道密钥 k_1 ，即使知道 m 和 c ，还是无法知道 x ；
- 然而，攻击者可以选择 x 的一个可能值，再试着找到一个可产生 x 的 (m, c) 对，从而将对三重加密的攻击转化为对二重加密的攻击。

攻击步骤



- ① 获取尽量多个 (m, c) 对, 存入表 1;
- ② 随意选择值 x , 按以下步骤创建表 2:
 - 对每个可能密钥 k_1 计算可产生 x 的明文 $m = D(k_1, x)$;
 - 在表 1 中匹配 m , 若匹配成功, 则在表 2 中添加一项 (y, k_1) , 其中 $y = D(k_1, c)$ 。
- ③ 搜索 k_2 :
 - 对每个可能密钥 k_2 , 计算 $y = D(k_2, x)$;
 - 在表 2 中匹配 y , 若匹配成功, 则找到一对密钥 (k_1, k_2) 可以产生已知 (m, c) 对。
- ④ 在其他 (m, c) 上验证找到的密钥对, 若验证成功, 则找到正确密钥对 (k_1, k_2) ; 否则, 返回步骤 2。

时间复杂度分析



- 对给定的 (m, c) 对, 选择 x 成功的可能性为 2^{-64} 。
- 给定 n 个 (m, c) 对, 则选择 x 成功的可能性为 $n2^{-64}$ 。
- 所以, 前两步平均需要尝试 $2^{64}/n$ 次才会得到一个正确的 x 。
- 对于每个 x , 第 3 步还需要遍历 k_2 , 因此总的时间复杂度为 $2^{56}2^{64}/n = 2^{120-\log_2 n}$ 。

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

- 电码本
- 密文分组链接
- 密文反馈
- 输出反馈
- 计数器

分组密码的工作模式

- 分组密码输入 b 位明文分组，输出 b 位密文分组。
- 若明文长度大于 b ，则需要将明文分成 b 位一组的块。
- 每次使用相同的密钥对多个分组加密，会引发安全问题。
- 为了将分组密码应用于各种各样的应用，NIST 定义了**五种工作模式**。
- 本质上，工作模式是一项增强密码算法或者使算法适应具体应用的技术。
- 五种工作模式可使用包括 DES 和 AES 在内的任何分组密码。

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

- 电码本

- 密文分组链接

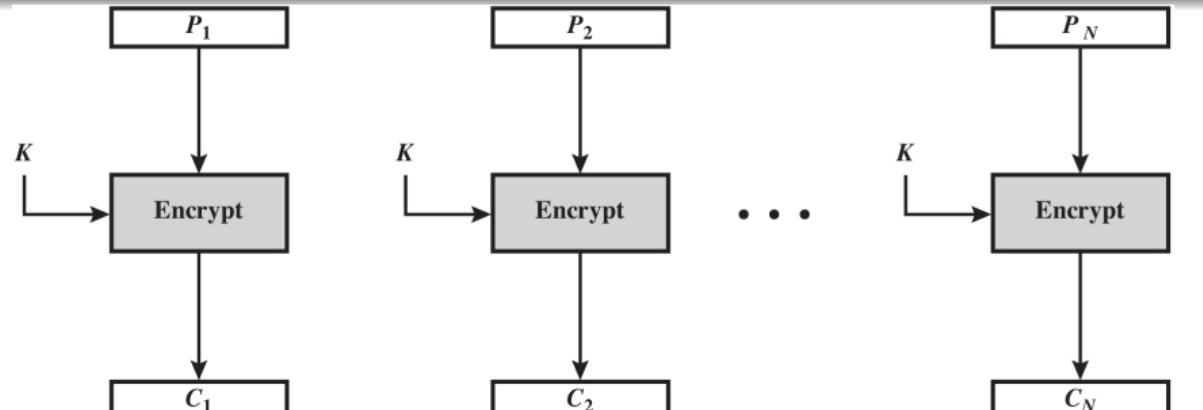
- 密文反馈

- 输出反馈

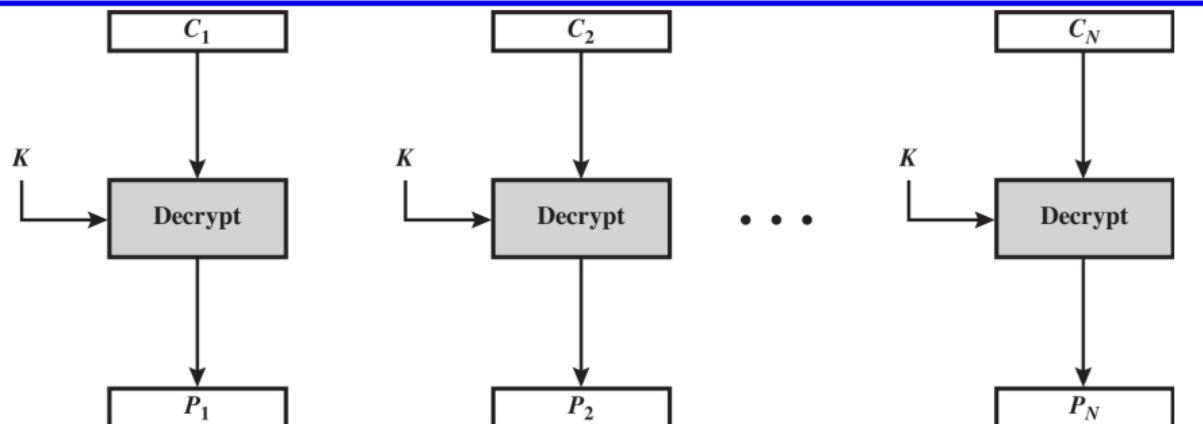
- 计数器

电码本 (Electronic Codebook, ECB)

加密:



解密:



ECB 的局限性

- 给定密钥，任何 b 位明文分组有唯一密文分组，类似于在一个很厚的密码本里查明文的相应密文，所以叫电码本模式。
- ECB 模式特别适合数据较少的情况，例如传输 DES 密钥。
- 一段明文消息中若有几个相同的明文分组，则密文也将出现几个相同的片段。
- 对于很长的消息，ECB 是不安全的，如果消息是非常结构化的，密码分析可能利用其结构特征来破解。
- ECB 的弱点来源于其加密过的密文分组互相独立。

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

- 电码本

- 密文分组链接

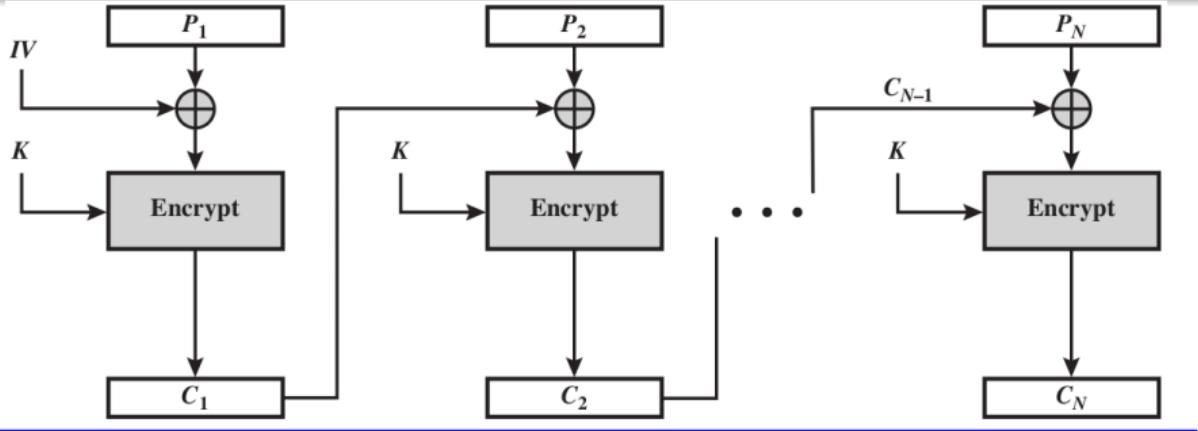
- 密文反馈

- 输出反馈

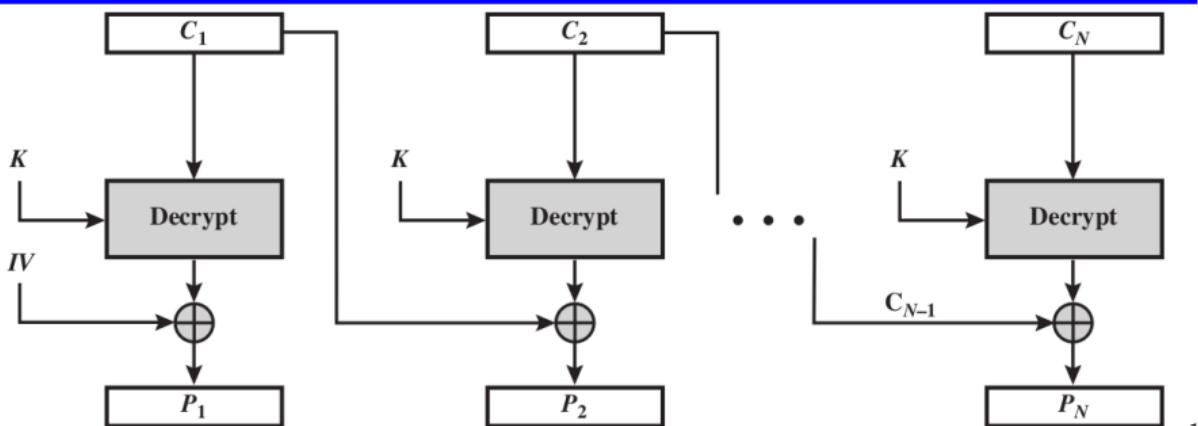
- 计数器

密文分组链接 (Cipher Block Chaining, CBC)

加密：



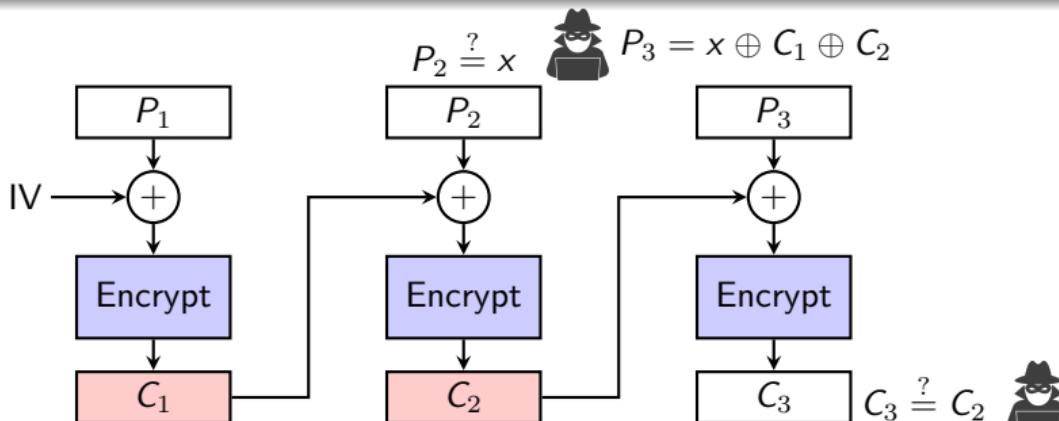
解密：



CBC 的优缺点

- 加密算法的输入是当前明文分组与上一个密文分组的异或。
- 每个密文分组依赖于所有之前的明文分组。
- 明文消息中的任何一点变化都会影响之后所有的密文分组。
- 发送方和接收方需要共享初始向量 (Initial Value, IV)。
 - 如果 IV 被明文传送，则攻击者可以通过改变 IV 进而改变则接收者收到的 P_1 。
 - 因此，IV 必须是定值或者必须用 ECB 方式在消息之前加密传送。
- 如果最后一个分组不是完整的分组，则需要填充：可以填充已知非数据值，或者在最后一块补上填充位长度。
 - eg., [b1 b2 b3 0 0 0 0 5], i.e., 3 data bytes, then 5 bytes pad+count.

CBC 的一个潜在漏洞



- 如果攻击者能观察连续两个密文分组 C_1 和 C_2 ，那么通过选择下一个明文分组 $P_3 = x \oplus C_1 \oplus C_2$ ，便可以判断第二个明文分组 P_2 是否等于 x 。
- 因为

$$C_3 = E(C_2 \oplus P_3) = E(C_2 \oplus x \oplus C_1 \oplus C_2) = E(x \oplus C_1)$$

当 $P_2 = x$ 时，有 $C_3 = C_2$ 。

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

5 高级数据加密标准

6 多重加密

7 分组密码的工作模式

● 电码本

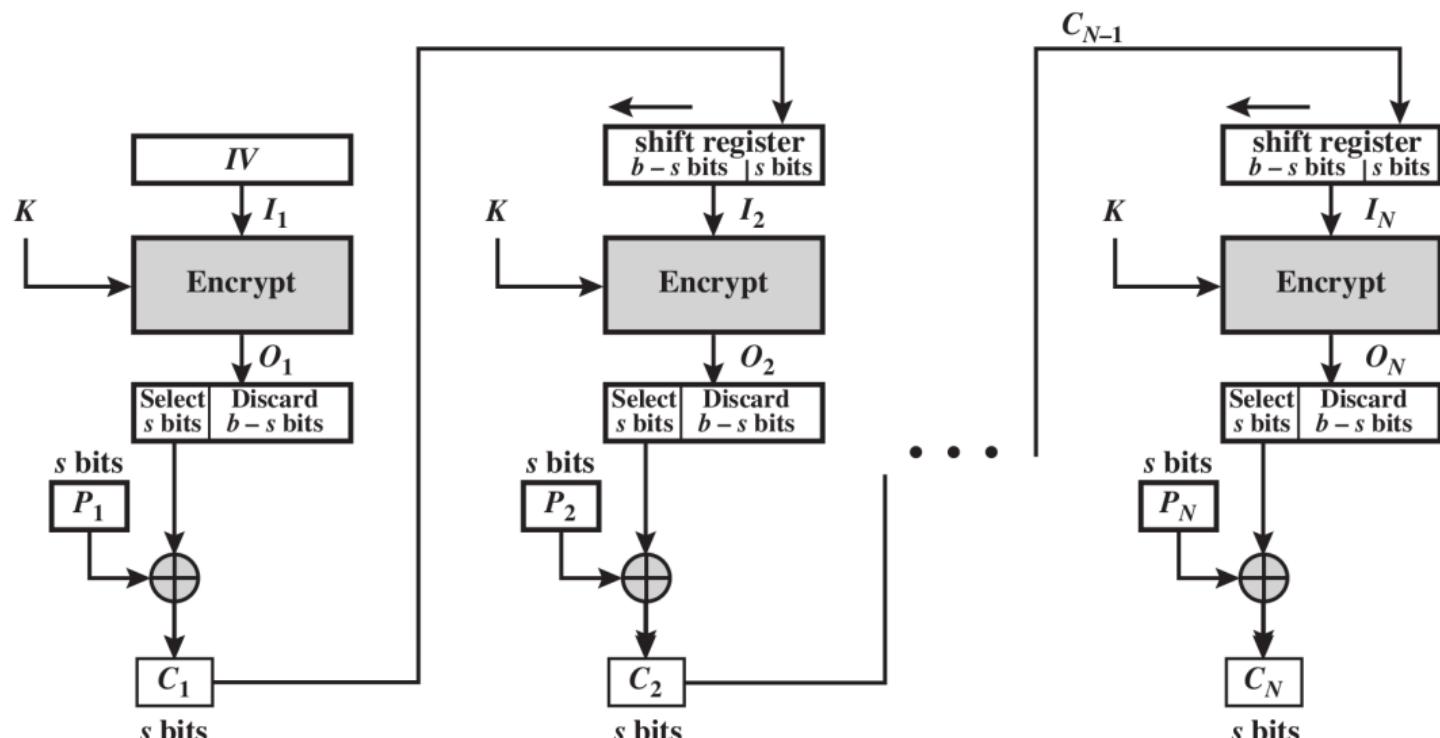
● 密文分组链接

● **密文反馈**

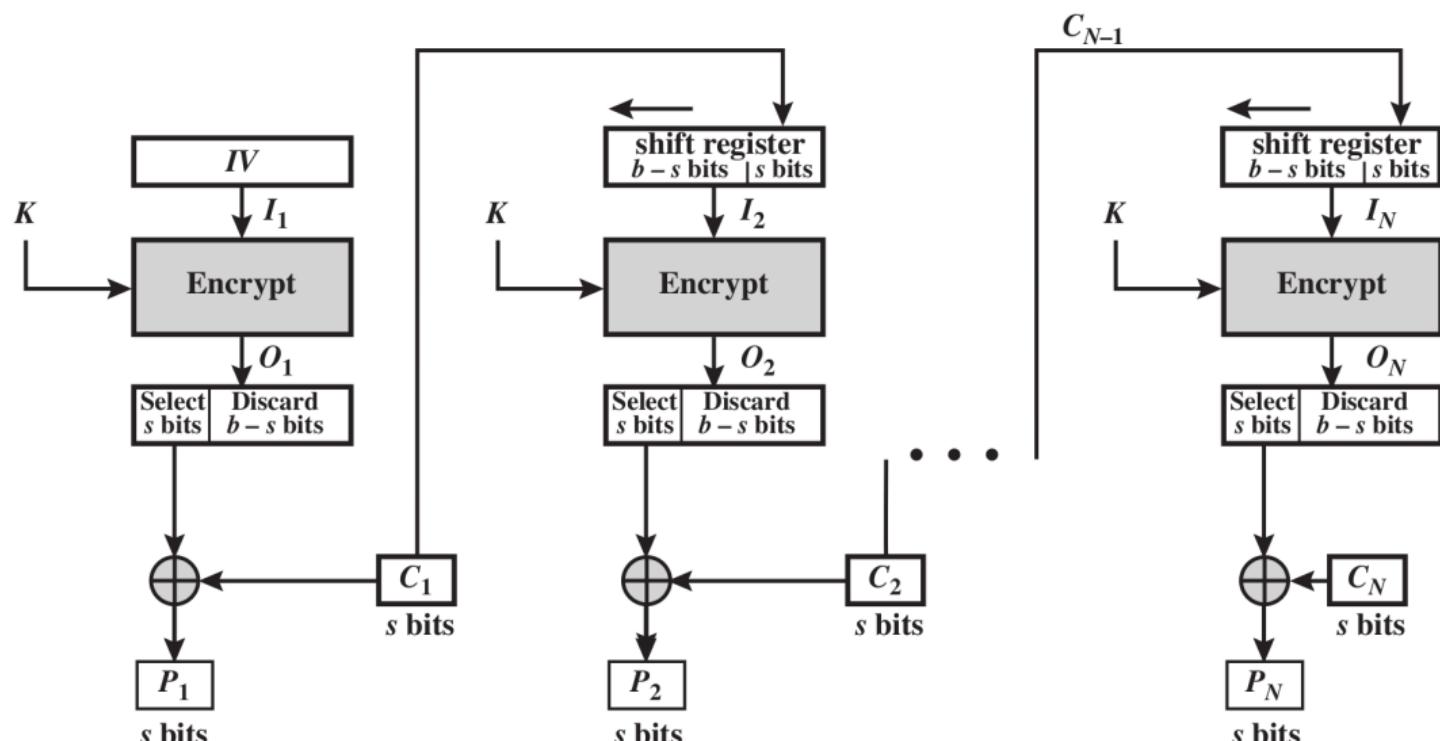
● 输出反馈

● 计数器

密文反馈 (Cipher Feedback, CFB): 加密



密文反馈 (Cipher Feedback, CFB): 解密



CFB 的优缺点

- 可以将分组密码转化成流密码的技术。
- 不再要求报文被填充成整个分组，数据以位或字节形式到达时都适用。
- 加解密使用相同方案，注意解密时仍使用加密函数。
- 密文在传输过程中发生错误时，会得到错误明文，错误会传播几个分组。

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

5 高级数据加密标准

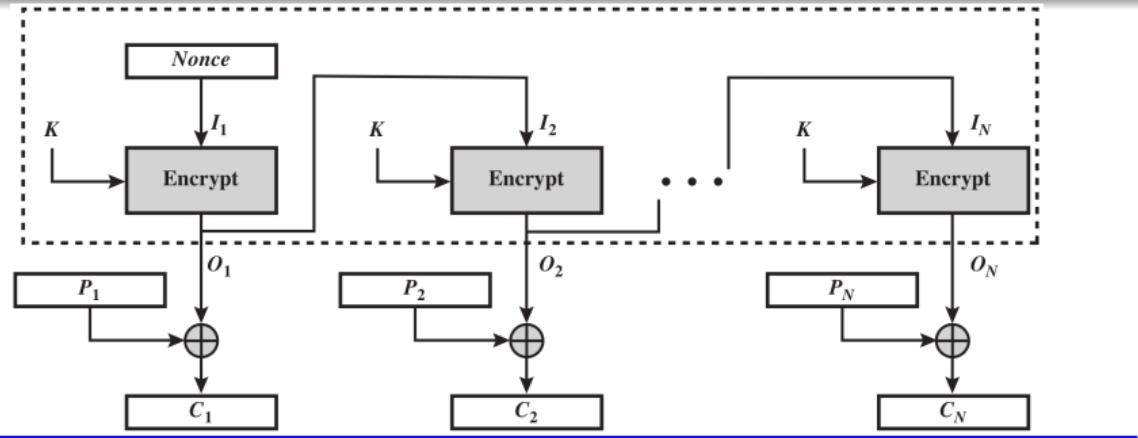
6 多重加密

7 分组密码的工作模式

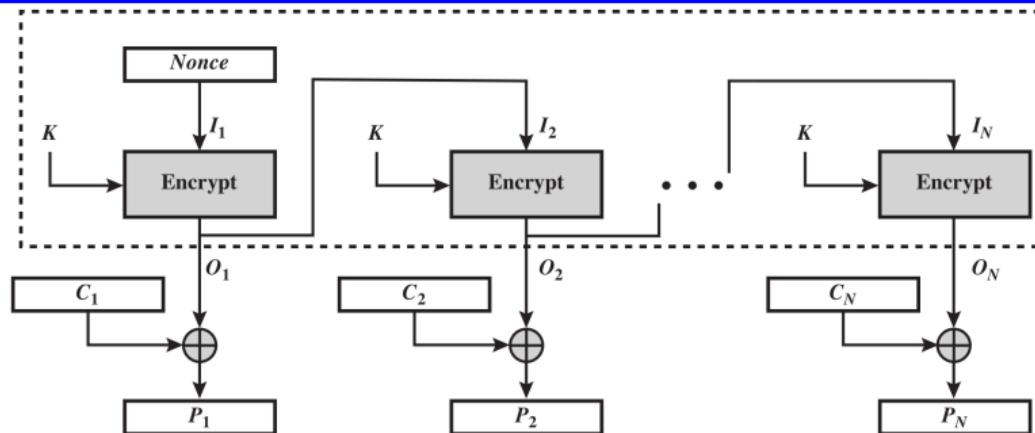
- 电码本
- 密文分组链接
- 密文反馈
- **输出反馈**
- 计数器

输出反馈 (Output Feedback, OFB)

加密:



解密:



OFB 的优缺点

- **优点**：密文传输过程中某位上发生的错误不会影响其他明文的恢复。
- 例如， C_1 中有一位发生了错误，只会影响 P_1 的恢复，不会影响后续明文的恢复。
- **缺点**：抗消息流篡改能力不如 CFB，即如果密文某位取反，则恢复出来的明文相应位也取反。

目录

1 分组密码的基本原理

2 数据加密标准

3 DES 的安全性

4 数论基础

5 高级数据加密标准

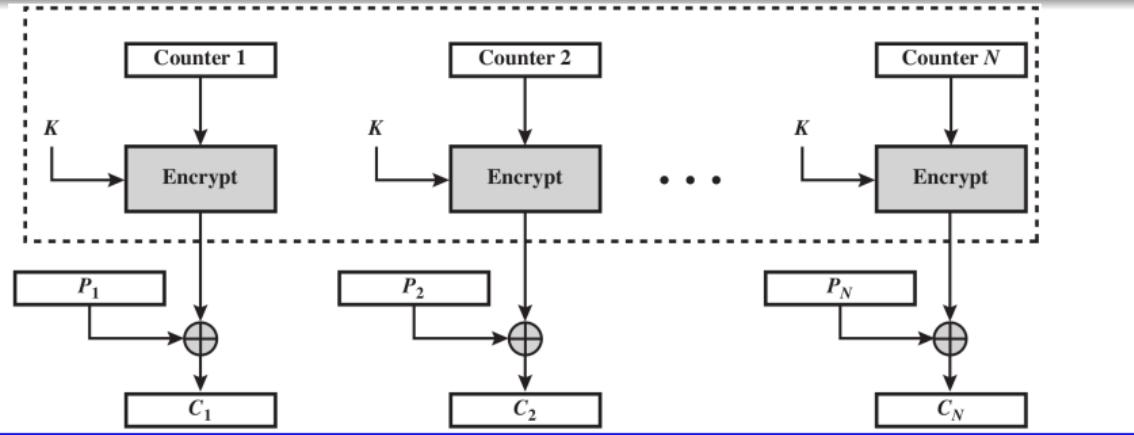
6 多重加密

7 分组密码的工作模式

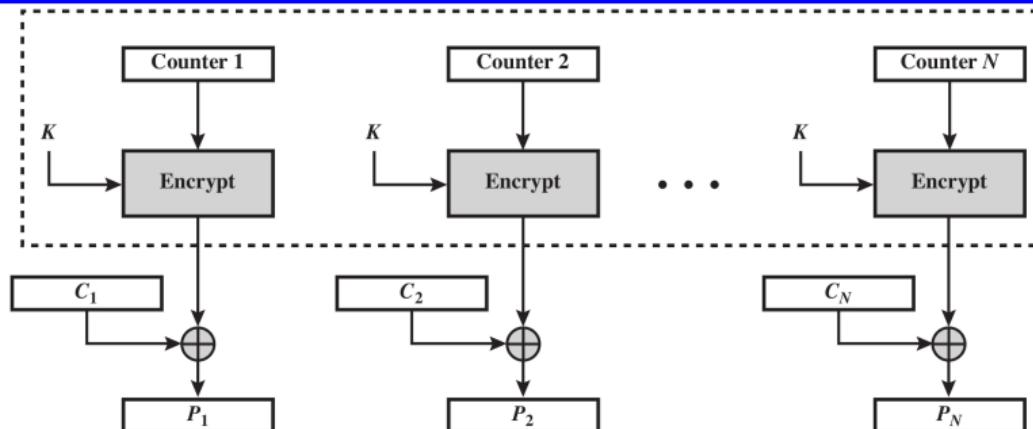
- 电码本
- 密文分组链接
- 密文反馈
- 输出反馈
- 计数器

计数器 (Counter, CTR)

加密：



解密：



CTR 的优缺点

- 高效，可以做并行加密，可以用于高速网络加密中。
- 可以对被加密的分组进行随机存取。
- 相当安全。
- 简洁。
- 必须决不重复使用密钥和计数器值。

小结

- 1 分组密码的基本原理
- 2 数据加密标准
- 3 DES 的安全性
- 4 数论基础
- 5 高级数据加密标准
- 6 多重加密
- 7 分组密码的工作模式