- 公开钥密码诞生之前, 所有的传统密码都属于单钥密码体制, 及加密钥和解密钥相同
- 单密钥算法可以分为两大类:
  - 一次一位地对明文进行操作和运算, 称为序列密码或流密码.
  - 一次若干位一组地对明文进行操作和运算, 称为分组密码.
  - 在计算机上实现的分组密码算法, 以往通常选 64 位 (足以应付密码分析, 又便于操作和运算); 现在往往选 128 位.
- 1977 年美国国家标准局 (NIST) 公布了 IBM 公司研制的一种加密算法, 作为非机要部门使用的数据加密标准 (DES): Data Encryption Standard.
- DES 自公布以来, 一直是国际商用通信和计算机通信最常用的加密算法, 原定使用期位 10 年, 超期服役到 2000 年.
- 1990's, 以色列密码学家 Shamir 等人提出一种" 差分分析法", 日本人也提出类似方法, 对 DES 构成威胁.
  - DES 提出不久, 有人提出借助硬件设备实现对所有密钥的遍历搜索. 电子技术的发展, 是这种设备的造价大大降低, 速度大大提高.
  - DES 的弱点是: 密钥太短, 实际为 56 位.

0.1 Feistel 加密算法

- Feistel 网络是一种乘积型加密算法, 已成为一种有效的构造密码方法. DES 即是 Feistel 网络的一种实现.

  Shannon 曾建议交替使用搅乱和扩散方法设计密码.
  - 搅乱: 即打乱明文, 使密文和明文的统计关系尽可能复杂化.
  - 扩散: 使密文和密钥的关系变得毫无统计规律, 扩大密钥对明文的影响.
- Feistel 网络是将明文分成 L_0 和 R_0 两部分, 各 b*bit*, 经过 n 轮迭代, 最后归并成密文块.

- 第 i 轮的输入为 L_{i-1} 和 R_{i-1}(前一轮的输出), 输出为 L_i 和 R_i. 流程图见 4.1
- 各轮结构相同, 单子密钥 k_i 不同, 由密钥 k 产生.
- 若密文为 C, 加密过程可表示为

$$C = F(m)$$

或者

$$F(m) = C = \underbrace{(F_n)}_{\text{第 n 轮}} \underbrace{(IF_{n-1})}_{\text{第 n-1 轮}} \cdots \underbrace{(IF_2)}_{\text{第 2 轮}} \underbrace{(IF_1)}_{\text{第 1 轮}} (m)$$

其中

$$\begin{cases} F_i(L_i, R_i) = [L_{i-1} \oplus f(R_{i-1}, k_i), R_i], i = 1, 2, \cdots, n \\ I(L, R) = (R, L) \end{cases}$$

$f(R_{i-1}, k_i)$ 称为轮函数, k_i 是子密钥.

- 解密过程与加密过程类似, 只是要将密钥的顺序颠倒过来, 即

$$m = F^{-1}(C) = (F_1)(IF_2)(IF_3)\cdots(IF_{n-1})(IF_n)(C)$$

0.2 数据加密标准 DES

- DES 的完整描述: 利用长度为 56 的密钥比特串加密长度加密长度为 64 位的明文比特串, 从而得到长度为 64 位的密文比特串.
- 加密时, 先进行初始置换 IP 的处理, 在进行一系列运算, 然后进行逆初始置换 IP^{-1} 给出加密结果.
- 与密钥有关的算法包括: 一个加密函数 f 和密钥编排函数 KS.
- DES 加密的框架为:

(1) 初始置换 IP: 把明文顺序打乱重排, 置换输出为 64 位, 数据置换后的第一位, 第二位分别为原来的 58 位, 50 位.

(2) 将置换输出的 64 位数据分成左右两半, 左一半称为 L_0, 右一半称为 R_0, 各 32 位.

(3) 计算函数的 16 轮迭代.

- 第 1 轮加密迭代: 左半边输入 L_0, 右半边输入 R_0; 由加密函数 f 实现子密钥 K_1 对 R_0 的加密, 结果为 32 为数据组 $f(R_0, K_1), f(R_0, K_1)$ 与 L_0 模 2 相加 (异或), 得到一个 32 位数据组 $L_0 \oplus f(R_0, K_1)$.
- 第 2 轮加密迭代: 左半边输入 $L_1 = R_0$, 右半边输入 $R_2 = L_0 \oplus f(R_0, K_1)$, 有加密函数 f 实现子密钥 K_2 对 R_1 的加密, 结果为 32 位数据组 $f(R_1, K_2)$, $f(R_1, K_1)$ 与 L_1 模 2 相加, 得到一个 32 位数据组 $L_1 \oplus f(R_1, K_2)$.
- 第 3 轮加密迭代: 第三论加密迭代值第 16 轮加密迭代分别用子密钥 $K_3, \cdots, K_16$ 进行, 其过程与第 1 次, 第 2 次加密迭代相同.
- 加密过程用数学公式描述如下

$$\begin{cases} L_1 = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \\ i = 1, 2, \cdots, 16 \end{cases}$$

- $\oplus$ 表示按位做不进位加法运算, 即 $1 \oplus 0 = 0 \oplus 1 = 1, 0 \oplus 0 = 1 \oplus 1 = 0$
- L_{i-1} 和 R_{i-1} 分别是第 $i-1$ 次迭代结果的左右两部分, 各 32 比特.
- K_i 是由 64 比特的密钥产生的子密钥, 长度为 48 比特.
- f 是将 $32bit$ 的输入转换为 $32bit$ 的输出. $f(R_{i-1}, K_i)$ 是 DES 加密算法的关键功能.

(4) 最后一次的迭代 L_{16} 放在右边, R_{16} 放在左边.

(5) 再经过逆初始置换IP^{-1}, 把数据打乱重排, 产生 64 位密文.

0.3 DES 加密标准

■ 假定明文 m 是 0 和 1 组成的长度为 $64bit$ 的符号串, 密钥 k 也是 0 和 1 组成的 $64bit$ 的符号串.

设

$$\begin{cases} m = m_1 m_2 m_3 \cdots m_{64} \\ k = k_1 k_2 k_3 \cdots k_{64} \end{cases}, m_i, k_i = 0, 1; i = 1, 2, 3, \cdots, 64$$

- 密钥 k 只有 $56bit$ 有效, 其中 $k_8, k_{16}, k_{24}, k_{32}, k_{40}, k_{56}, k_{64}$ 这八位数是奇偶校验位, 在算法中不起作用. DES 加密过程如下:

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

其中, IP 为初始置换, IP^{-1} 为逆初始置换.

  - 初始置换 IP 可表示为: 设 $m = m_1 m_2 \cdots m_{64}$, $\widetilde{m} = \widetilde{m_1}\widetilde{m_2} \cdots \widetilde{m_{64}}$ 经 IP 初始置换得到 $\widetilde{m} = \widetilde{m_{58}}\widetilde{m_{50}}\widetilde{m_{42}} \cdots \widetilde{m_{23}}\widetilde{m_{15}}\widetilde{m_7}$, 即 $\widetilde{m_1} = m_{58}, \widetilde{m_2} = m_{50}, \cdots, \widetilde{m_{63}} = m_{15}, \widetilde{m_{64}} = m_7$

    m 经过 IP 初始置换 $\widetilde{m}$, 其 $\widetilde{m_i}(i = 1, 2, 3, \cdots, 64)$ 的下标依次如下表所示

表 1: IP 置换

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

  - 逆初始置换 IP^{-1} 可表示为:

    设 $\widetilde{m} = \widetilde{m_1}\widetilde{m_2} \cdots \widetilde{m_{64}}, m = m_1 m_2 \cdots m_{64}$ 经过逆初始置换 IP^{-1} 得到

$$m = \widetilde{m_{40}}\widetilde{m_8}\widetilde{m_{48}} \cdots \widetilde{m_{17}}\widetilde{m_{57}}\widetilde{m_{25}}$$

    即

$$m_1 = \widetilde{m_{40}}, m_2 = \widetilde{m_8}, m_3 = \widetilde{m_{48}}, \cdots, m_{63} = \widetilde{m_{57}}, m_{64} = \widetilde{m_{25}}$$

    $\widetilde{m}$ 经过 IP^{-1} 逆初始置换为 m, 其 $m_i(i = 1, 2, \cdots, 64)$ 的下标依次如下表所示

表 2: IP^{-1} 置换

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

0.4 子密钥的生成 (密钥编排函数 KS 算法)

- K 的长度为 64 位的比特串, 其中 56 位是密钥, 8 位是奇偶校验位, 分别在 $8, 16, \cdots, 64$ 上
- 奇偶校验位的比特由每个字节包含奇数个 1 定义, 在密钥编排的计算中这些奇偶校验位可以忽略.
- 64 位密钥经过置换选择 1、循环左移、置换选择 2 等变换, 产生 16 个子密钥.

  (1) 置换选择 1(PC-1)
  - 从 64 位密钥中去掉 8 个奇偶校验位.
  - 将其余 56 位数据打乱重排, 且将前 28 位作为 C_0, 后 28 位作为 D_0.
  - 置换选择 1 列于下表

  (2) 循环左移
  - 对于 $1 \leq j \leq 16$, 需要计算

$$\begin{cases} C_i = LS_i(C_{i-1}) \\ D_i = LS_i(D_{i-1}) \end{cases}$$

  - LS_i 表示循环左移一个或两个位置, 这取决于 i 值, 如果 $i = 1, 2, 9, 16$, 则移动一个位置, 否则移动两个位置.

表 3: 置换选择 1(PC-1)[56 位]

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

表 4: 循环左移位数对照表

迭代顺序	1	2	3	4	5	6	7	8	9	10	10	11	12	13	14	15	16
左移顺序	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	2	1

- 循环左移

  设 $C_1 = c_1c_2\cdots c_{28}, D_i = d_1d_2\cdots d_{28}$

  第二轮迭代, 应该循环左移 1 位, 则有

  $$C_2 = c_2c_3\cdots c_{28}c_1, D_2 = d_2d_3\cdots d_{28}d_1$$

  第三轮迭代, 因该循环左移 2 位, 则有

  $$C_3 = c_4c_5\cdots c_{28}c_1c_2c_3, D_3 = d_4d_5\cdots d_{28}d_1d_2d_3$$

(3) 置换选择 2(PC-2)

- 置换选择 2 是从 C_i 和 D_i(共 56 位) 中选择出一个 48 位的子密钥K_i,
- 置换选择 2 列于如下表

(4) 加密函数 f

- 加密函数 f 的作用是在第 i 次加密迭代中用于子密钥K_i 对 R_{i-1} 进行加密

  过程如下:

  * 在第 i 次加密迭代中选择运算 E 对 32 位的 R_{i-1} 的各位进行选择和排列, 产生一个 48 位的结果 (1);

表 5: 置换选择 2(PC-2)[48 位]

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

* 结果 (1) 与子密钥 K_i 模 2 相加, 送入选择函数 S, 得到一个 32 位的数据组 (结果 (2));
* 结果 (2) 经过置换 P 运算, 将其各位打乱重排, 最后的输出为加密函数的输出 $f(R_{i-1}, K_i)$.

[1] 选择运算 E

* E 的作用是将 32 位的 R_{i-1}输入扩展为 $48bit$ 的输出. 输出 $48bit$ 的下标如下表所示.
* 扩展: $E(R_{i-1})$ 由 R_{i-1} 中的 $32bit$ 以特定方式排列产生, 其中 16 个比特出现两次

表 6: E 的选位表

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

[2] 选择函数数组 S

– 选择函数组由 8 个选择函数, 分别记为 $S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8$.

- 选择函数组的输入是一个 48 位的数据组, 从第 1 位到第 48 位依次加到 $S_1 \sim S_8$ 的输入端.
- 每个选择函数有 6 个输入, 4 个输出.
- 每个选择函数都由一个选择矩阵 (见上表) 规定了其输出和输入的选择原则.
- 选择矩阵为 4 行 16 列, 每行都由 $0 \sim 15$ 组成, 但每行的数字排列都不同
- 8 个选择矩阵各不相同
- 每个选择函数 6 位输入组成一个 6 位二进制数字组.
- 选择规则:
  - 行号: 输入二进制数字组的第 1 和第 6 两位组成的二进制数值
  - 列号: 其余 4 位组成的二进制数值
  - 处在被选中的行和列交点处的数字便是选择函数的输出 (以二进制形式输出).

例如: 对于 S_1, 设输入为 101011 求 S_1 的输出

- 行号: 第 1 位和第 6 位组成 $11 = (3)_{10}$, 表示选中 S_1 的标号为 3 的那一行
- 列号: 其余 4 为组成 $0101 = (5)_{10}$, 表示选中 S_1 的标号为 5 的那一列
- 得到交点处的数字: 查询选择矩阵得到输出数字为 9, 即 $(1001)_2$.

[3] 置换 P

- 把选择函数输出的 32 位结果在打乱重排, 得到 32 位的加密函数 f.
- 经过置换 P 的顺序改变

表 7: 置换 P 矩阵

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

0.5 DES 加密举例

加密明文 $M = 00000001\ 00100011\ 01000101\ 01100111$
$= 10001001\ 10101011\ 11001101\ 11101111$

密钥 $K = 00010011\ 00110100\ 01010111\ 01111001$
$= 10011011\ 10111100\ 11011111\ 11110001$

解: 明文 M

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	1	0	0	0	1	0	1	0	1	1	0	0	1	1	1
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
1	0	0	0	1	0	0	1	1	0	1	0	1	0	1	1
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
1	1	0	0	1	1	0	1	1	1	1	0	1	1	1	1

密钥 K

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0	1	0	0	1	1	0	0	1	1	0	1	0	0
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
0	1	0	1	0	1	1	1	0	1	1	1	1	0	0	1
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
1	0	0	1	1	0	1	1	1	0	1	1	1	1	0	0
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
1	1	0	1	1	1	1	1	1	1	1	1	0	0	0	1

[1] IP 置换 (64 位)

$\widetilde{m} = 11001100000000000110011001111111\ 11110000101010101111000010101010$

$L_0 = 11001100000000000110011001111111$

$R_0 = 11110000101010101111000010101010$

[2] PC-1 置换 (56 位)

$\widetilde{K} = 11110000110011001011010101111\ 01010101011001100111100011111$

$C_0 = 11110000110011001010101011111$

$D_0 = 01010101011001100111100011111$

[3] 循环左移 (LS)

[4] PC-2 置换 (48 位) 由 $C_1 = 11100001100110000101010111111, D_1 = 10101010110011001111100011110$, 及 PC-2 置换得到 K_1, 由 C_2, D_2 即 PC_2 得到 $K_2, \cdots$, 由 C_{16}, D_{16} 以及 PC-2 置换得到 K_{16}

[5] E 置换 (48 位) 由 R_0 即 E 置换得到 $E(R_0)$, 同理得到 $E(R_1), E(R_2), \cdots, E(R_{15})$

[6] $E(R_i) \oplus K_i$ 运算 (48 位)

[7] S 盒输出 (32 位)

[8] P 置换 (32 位)

[9] $L_i \oplus f(R_{i-1}, K_i)$ 运算 (32 位)

[10] 对 L_{16}, L_{17} 进行 IP^{-1} 置换

0.6 DES 解密过程

- 由于 DES 运算是对合运算, 所以其解密和加密共同用一个算法, 只是要将 16 轮的子密钥的使用顺序倒过来, 即依次使用 $K_{16}, K_{15}, \cdots, K_1$.
- 解密时, 把 64 位的密文当作明文输入, 第 1 轮的解密迭代使用子密钥 K_{16}, 第 2 轮解密迭代使用子密钥 K_{15}, $\cdots\cdots$, 第 16 轮的解密迭代使用子密钥 K_1, 最后的输出便是 64 位明文.
  - 解密过程中使用的数学公式如下

$$\begin{cases} R_{i-1} = L_i \\ L_{i-1} = R_i \oplus f(L_i, K_i) \\ i = 16, 15, \cdots, 1 \end{cases}$$

  - 加密过程用数学公式描述如下:

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \\ i = 1, 2, \cdots, 16 \end{cases}$$

0.7 DES 的若干问题

(1) DES 的特点

- DES 的综合运用了置换、代替、代数等多种密码技术, 是一种乘积密码

表 8: 循环左移

第 1 轮 (左移一位)	$C_1 = 11100001100110000101010111111$ $D_1 = 10101010110011001111100011110$
第 2 轮 (左移一位)	$C_2 = 11000011001100001010101111111$ $D_2 = 01010101100110011111000111101$
第 3 轮 (左移二位)	$C_3 = 00001100110000101010111111111$ $D_3 = 01010110011001111100011110101$
第 4 轮 (左移二位)	$C_4 = 00110011000010101011111111100$ $D_4 = 01011001100111110001111010101$
第 5 轮 (左移二位)	$C_5 = 11001100001010101111111110000$ $D_5 = 01100110011110001111010101010$
第 6 轮 (左移二位)	$C_6 = 00110000101010111111111000011$ $D_6 = 10011001111000111101010101010$
第 7 轮 (左移二位)	$C_7 = 11000010101011111111100001100$ $D_7 = 01100111100011111010101010101$
第 8 轮 (左移二位)	$C_8 = 00001010101111111110000110011$ $D_8 = 10011110001111101010101010101$
第 9 轮 (左移一位)	$C_9 = 00101010111111111000011001100$ $D_9 = 00111100011110101010101011$
第 10 轮 (左移二位)	$C_{10} = 10101011111111100001100110000$ $D_{10} = 11110001111010101010101100$
第 11 轮 (左移二位)	$C_{11} = 10101111111110000110011000010$ $D_{11} = 11000111101010101010110011$
第 12 轮 (左移二位)	$C_{12} = 10111111111000011001100001010$ $D_{12} = 00011110101010101011001111$
第 13 轮 (左移二位)	$C_{13} = 11111111000011001100000101010$ $D_{13} = 01111010101010101100111100$
第 14 轮 (左移二位)	$C_{14} = 11111100001100110000101010111$ $D_{14} = 11101010101010101100111110001$
第 15 轮 (左移二位)	$C_{15} = 11110000110011000010101011111$ $D_{15} = 10101010101010110011111000111$
第 16 轮 (左移一位)	$C_{16} = 11100001100110000101010111111$ $D_{16} = 01010101010101100111110001111$

- DES 算法结构紧凑、条例清楚, 其运算为对合运算, 便于实现
- DES 使用了初始置换 IP 和逆初始置换 IP^{-1} 各 1 次, 置换 $P16$ 次. 安排只用这三种置换的目的是把数据彻底打乱, 重排 (扩展). 但在密码意义上作用不大, 因为这三种置换与密钥无关, 置换关系固定.
- 算法中除选择函数组 S 是非线性变换外, 其余变换均为线性变换, 所以保密性的关键是选择函数组 S. S 的本质是数据压缩 (6 位输入压缩成 4 位输出). 其输入任意改变数位, 输出至少变换 2 位. 因为算法中使用了 16 次迭代, 使得即使是改变明文或密钥的 1 位, 密文都会发生的 32 的变化. 这种密文的每一位都与明文和密钥的每一位有着敏感而复杂的关系, 牵一发而动全身的现象称为雪崩现象.
- DES 的子密钥的产生与使用很有特色, 确保了原密钥中的各位的使用次数基本上相等; 即 56 位密钥每一位的使用次数在 $12 \sim 15$ 次之间. 使得保密性进一步提高.

(2) 弱密钥和半若密钥

- DES 的 64 位密钥 (有效位位 56 位) 太短.
- 在 16 次迭代中分别是不同的子密钥确保了 DES 强度, 但实际上却存在者一些相互重合的子密钥, 称这些密钥位弱密钥或半弱密钥.
- 称使 16 个子密钥全相同的密钥位弱密钥, 即 $k_1 = k_2 = \cdots = k_{16}$. 4 种弱密钥分别为 $01 = 00000001, 1F = 00011111, E0 = 11100000, FE = 11111110, F1 = 11110001, 0E = 00001110$
- 称使 16 个子密钥部分相同的密钥为半弱密钥.
  若存在 k 和 k' 使得 $DES_k(m) = DES_{k'}^{-1}(m), DES_k[DES_{k'}(m)] = m$, k 和 k' 构成半弱密钥.
  半弱密钥由 6 对
- 弱密钥或者半弱密钥的存在是 DES 的不足. 但由于弱密钥或半弱密钥的数量与密钥的总数相比是微不足道的, 所以对 DES 并不构太大威胁

(3) DES 存在互补对称性

- 设 $C = DES(M, K)$, 则 $\overline{C} = DES(\overline{M}, \overline{K})$, 其中 $\overline{M}, \overline{K}, \overline{C}$ 表示 M, K, C 的非.
- 互补对称性会使得 DES 在" 选择明文攻击" 下所需的工作量减半.
- 产生互补对称性的原因是 DES 中两次使用 $\oplus$ 运算

(4) DES 的安全性

- 虽然 DES 的描述相当长, 但能以软件或硬件的方式非常有效的实现. 所需完成的算术成本仅是比特串的异或. 扩展函数 E,S 盒, 置换 IP 和 IP^{-1} 以及 $K_1, K_2, \cdots, K_{16}$ 的计算都能在一个固定时间内通过查表 (以软件) 或者电路布线完成.
- 正式颁布后, 世界各国的许多公司都推出了自己实现的软硬件产品.
- DES 的密钥空间为 $2^{56} \approx 7 \times 10^6$, 对实现真正的安全而言太小. 对于" 已知明文的攻击", 已经开发出了各种专用机器用于对密钥的穷举搜索.
- 穷举搜索即给定的 64 比特的明文 M 和相应的的密文 C, 测试每一个可能的密钥 K, 直到找到密钥满足 $E(M, K) = C$ 为止, 同时注意可能有多个这样的密钥 K.
- 1977 年, DH 已经建议制造一个一秒能测试 10^6 个密钥的芯片.
- 每秒能测试 10^6 个密钥的机器大约需要一天就可以搜索整个密钥空间, 制造这样的机器需要...
- 1990 年 EA 提出差分分析法使得选择明文攻击下的复杂度下降 $O(2^{37})$
- 1993 年 RM 给出一种非常详细的密钥搜索机器的设计方案.3.5 个小时左右平均搜索时间

0.8 IDEA 密码系统

- PES: 第一版密码算法由中国薛和 james 提出, 于 1990 年公布
- IPES: 1991 年为抵抗差分分析法, 增强算法强度

- IDEA: 1992 年改名为 IDEA, 国际数据加密算法.
- IDEA 分组密码算法, 明文块和密文都是 64 位, 密钥长度为 128 比特. 其加密和解密算法相同, 但密钥各异.
- IDEA 的特点是用软件或者硬件实现都不难, 加、解密运算速度非常快.
- IDEA 的加密算法的流程图
  - 第一轮的输出四个子块: [11], [12], [13] 和 [14], 将中间的两个子块交换 (最后一轮除外) 后, 就是下一轮的输入.
  - 经过八轮运算后, 有一个最后的输出变换.

    [1] $X_1^{(9)}$ 和输出变换的第一个密钥子块 $Z_1^{(9)}$ 相乘　[2] $X_2^{(9)}$ 和输出变换的第二个密钥子块 $Z_2^{(9)}$ 相乘　[3] $X_3^{(9)}$ 和输出变换的第三个密钥子块 $Z_3^{(9)}$ 相乘　[4] $X_4^{(9)}$ 和输出变换的第四个密钥子块 $Z_4^{(9)}$ 相乘

    最后, 这四个子块连接到一起产生密文.

1. IDEA 加密时, 一共有 8 轮迭代, 每轮需要 6 个子密钥, 最后一轮 (输出变换) 需要 4 个子密钥. 所以, 在整个加密过程中需要 52 个子密钥, 每个子密钥 16 比特.

2. 设密钥 $K_1 = k_1k_2\cdots k_{128}$, 将 K_1 分为 8 段, 依此得到

$$\begin{aligned} Z_1^{(1)} &= k_1k_2\cdots k_{16}, & Z_2^{(1)} &= k_{17}k_{18}\cdots k_{32} \\ Z_3^{(1)} &= k_33k_34\cdots k_{48}, & Z_4^{(1)} &= k_{49}k_{50}\cdots k_{64} \\ Z_5^{(1)} &= k_65k_66\cdots k_{80}, & Z_6^{(1)} &= k_{81}k_{82}\cdots k_{96} \\ Z_1^{(2)} &= k_{97}k_{98}\cdots k_{112}, & Z_2^{(2)} &= k_{113}k_{114}\cdots k_{128} \end{aligned}$$

   即遣 6 个位第一轮迭代的子密钥块, 后 2 个为第二轮迭代 6 个子密钥块的前 2 个

3. 将 K_1 循环左移 25 位得到 $K_2 = k_{26}k_{27}\cdots k_{128}k_1K_2\cdots k_{24}k_{25}$, 再将

K_2 分为 8 段, 依此得到

$$Z_3^{(2)} = k_{26}k_{27}\cdots k_{41}, \qquad Z_4^{(2)} = k_{42}k_{43}\cdots k_{57}$$
$$Z_5^{(2)} = k_{58}k_{59}\cdots k_{73}, \qquad Z_6^{(2)} = k_{74}k_{75}\cdots k_{89}$$
$$Z_1^{(3)} = k_{90}k_{91}\cdots k_{105}, \qquad Z_2^{(3)} = k_{106}k_{107}\cdots k_{121}$$
$$Z_3^{(3)} = k_{122}k_{123}\cdots k_{128}k_1k_2\cdots k_9, \qquad Z_4^{(3)} = k_{10}k_{11}\cdots k_{25}$$

即前 4 个为第二轮迭代 6 子密钥块的另外 4 个, 后 4 个为第三轮迭代 6 子密钥块的前 4 个

4. 将 K_2 继续左移 25 位得到 $K_3 = k_{51}k_{52}k_{53}\cdots k_{128}k_1k_2\cdots k_{49}k_{50}$, 可依此得到 $Z_5^{(3)}, Z_6^{(3)}, Z_1^{(4)}, Z_2^{(4)}, Z_3^{(4)}, Z_4^{(4)}, Z_5^{(4)}, Z_6^{(4)}$.

5. 继续以上步骤, 直到产生 52 个子密钥为止.

表 9: IDEA 算法加密密钥子块

轮次	加密密钥子块
1	$Z_1^{(1)}, Z_2^{(1)}, Z_3^{(1)}, Z_4^{(1)}, Z_5^{(1)}, Z_6^{(1)}$
2	$Z_1^{(2)}, Z_2^{(2)}, Z_3^{(2)}, Z_4^{(2)}, Z_5^{(2)}, Z_6^{(2)}$
3	$Z_1^{(3)}, Z_2^{(3)}, Z_3^{(3)}, Z_4^{(3)}, Z_5^{(3)}, Z_6^{(3)}$
4	$Z_1^{(4)}, Z_2^{(4)}, Z_3^{(4)}, Z_4^{(4)}, Z_5^{(4)}, Z_6^{(4)}$
5	$Z_1^{(5)}, Z_2^{(5)}, Z_3^{(5)}, Z_4^{(5)}, Z_5^{(5)}, Z_6^{(5)}$
6	$Z_1^{(6)}, Z_2^{(6)}, Z_3^{(6)}, Z_4^{(6)}, Z_5^{(6)}, Z_6^{(6)}$
7	$Z_1^{(7)}, Z_2^{(7)}, Z_3^{(7)}, Z_4^{(7)}, Z_5^{(7)}, Z_6^{(7)}$
8	$Z_1^{(8)}, Z_2^{(8)}, Z_3^{(8)}, Z_4^{(8)}, Z_5^{(8)}, Z_6^{(8)}$
输出变换	$Z_1^{(9)}, Z_2^{(9)}, Z_3^{(9)}, Z_4^{(9)}$

0.9 IDEA 密码的解密运算

- IDEA 的解密过程和加密过程完全一致, 只是解密用的子密钥块不同.
- 解密子密钥块中 Z^{-1} 表示 $Z \bmod (2^{16}+1)$ 乘法的逆, 即

$$Z \odot Z^{-1} \equiv 1 \bmod (2^{16}+1)$$

, 即

$$Z + (-Z) \equiv 0 \bmod 2^{16}$$

0.10 IDEA 密码分析

- 欲对 IDEA 的密钥进行穷举搜索需进行 $2^{128}(10^{38})$ 次运算. 设计一个每秒能测试十亿个密钥的芯片, 并采用十亿片并行处理上述问题, 将花费 10^{12} 年.
- 对 IDEA 使用强力攻击的另一个困难是其加密子密钥的产生比解密子密钥的产生快的多.
- IDEA 对低于差分分析密码分析也十分有效. 一般而言, 一个密码体制的迭代次数越多, 对它进行差分密码分析就越困难. Lai 证明 IDEA 在他的 8 轮运算仅运行 4 轮后就对差分密码分析免疫了.
- IDEA 也有一组弱密钥 (十六进制): $0000, 0000, 0F00, 0000, 0000, 000F, FFFF, F000$ 在"F" 位置上的数可以是任何数.

0.11 FEAL 密码

- 1987 年日本学者清水明宏和宫口庄司在 DES 的基础上提出了一种快速数据加密算法 (Fast Data Encryption Algorithm), 简称 FEAL.
- FEAL 的算法类似于 DES, 但其每轮都比 DES 的强度大, 且因其轮次少, 因而运算速度较快.
- FEAL 和 DES 相比主要特点在于

  1. 增大了有效密钥长度;
  2. 增强了密钥的控制作用;
  3. 增大了加密函数 f 的复杂性;
  4. 减少了迭代次数

  FEAL 的整体结构

1. FEAL 的整体结构与 DES 相似. 分组长度位 64 位, 算法面向二进制设计, 加密算法是对合运算. 64 位明文在密钥的控制下, 经过初始运算、四次迭代、末尾运算, 形成 64 位密文.

   FEAL 的加密运算

- 整个加密过程分为三个阶段

(1) 64 位明文分为左右两块, $L_0'' = m_0 m_1 \cdots m_{31}, R_0'' = m_{32} m_{31} \cdots m_{63}$

$$\begin{cases} L_0' = L_0'' \oplus (K_4, K_5) \\ R_0' = R + 0'' \oplus (K_6, K_7) \\ L_0 = L_0' \\ R_0 = R_0' \oplus L_0' \end{cases}$$

(2) 四次迭代

$$\begin{cases} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, K_{i-1}) \end{cases} \quad i = 1, 2, 3, 4$$

其中, f 是加密函数

(3) 末尾运算

$$\begin{cases} R_4' = R_4' \\ L_4' = L_4 \oplus R_4 \\ R_4'' = R_4' \oplus (K_8, K_9) \\ L_4'' = L_4' \oplus (K_{10}, K_{11}) \end{cases}$$

将 R_4'' 和 L_4'' 合并形成最后的 64 位密文

子密钥产生算法

- FEAL 的密钥为 64 位, 全部为密钥位.
- 由子密钥产生算法产生出 $K_0, K_1, \cdots, K_{11}$ 共 12 个 16 位子密钥, 用于加、解密运算. 子密钥产生算法

1. 64 位密钥首先被分成为左右两半, 左边 32 位为 A_0, 右边为 B_0.

$$A_0 = k_0 k_1 \cdots k_{31}, B_0 = k_{32} k_{33} \cdots k_{63}$$

2. 引入中间变量 D, 其初值为 $D_0 = 0$.
3. 接连进行 6 次相同的产生子密钥的变换, 每次变换后将 B_i 分成左右两半, 便得到两个子密钥

$$\left.\begin{aligned} D_i &= A_{i-1} \\ A_i &= B_{i-1} \\ B_i &= f_K(A_{i-1}, B_{i-1} \oplus D_{i-1}) = (K_{2i-2}, K_{2i-1}) \end{aligned}\right\} i = 1, 2, \cdots, 6$$

其中, f_K 为子密钥生成服务函数

0.12 服务函数

- FEAL 使用了两个服务函数: 加密函数f 和子密钥产生函数f_K. 它们为 FEAL 提供了非线性, 是确保 FEAL 安全的关键.

  1. 移位函数 S
     - 加密函数 f 和子密钥产生函数 f_K 都使用了一个辅助函数: 移位函数 S.
     - 移位函数 $S(x_1, x_2, \delta)$ 是一个字节函数, 其自变量 x_1 和函数值 x_2 均为一个字节量. 移位函数 S 用于实现字节加法 (mod 256). 循环左移运算, 其定义为:

$$S(x_1, x_2, \delta) = R(W)$$
$$W = (x_1 + x_2 + \delta)(\text{mod}256), \delta = 0\text{或 } 1$$

       其中 $R(W)$ 表示把 W 循环左移两位.

0.13 加密函数 f

- 加密函数 $f(\alpha, \beta)$ 是多字节函数, 其中自变量 α 是 4 字节变量, β 是 2 字节变量, 结果为 4 字节变量
- 若用符号 f_i 表示多字节量 f 的第 i 个字节, 则 $f(\alpha, \beta)$ 定义如下:

$$g_1 = \alpha_1 \oplus \beta_0 \oplus \alpha_0$$
$$g_2 = \alpha_2 \oplus \beta_1 \oplus \alpha_3$$
$$f_1 = S(g_1, g_2, 1)$$
$$f_2 = S(a_2, f_1, 0)$$
$$f_3 = S(a_3, f_1, 1)$$
$$f_0 = S(a_0, f_1, 0)$$

1 FEAL 解密过程

- FEAL 的加密运算为对合运算, 故其解密运算和加密运算共用一个算法, 但解密过程中的子密钥使用顺序与加密相反, 这一点与 DES 类似.
- FEAL 的解密过程如下:

$$
\begin{aligned}
&L'_4 = L''_4 \oplus (K_{10}, K_{11}) \\
&R'_4 = R''_4 \oplus (K_8, K_9) \\
&\left.\begin{aligned} R_{i-1} &= L_i \\ L_{i-1} &= R_i \oplus f(L_i, K_{i-1}) \end{aligned}\right\} \; i = 4, 3, 2, 1
\end{aligned}
$$