## الگوریتم رمزنگاری DES:

ابتدا قبل از هرچیزی، متن و کلید را به بیت تبدیل می‌کنیم

L | R

M = 0000 0001 0010 0011 0100 0101 0110 0111 | 1000 1001 1010 1011 1100 1101 1110 1111 (64bit)

K = 0001 0011 0011 0100 0101 0111 0111 1001 | 1001 1011 1011 1100 1101 1111 1111 0001 (64bit)

8 16 24 32 40 48 56 64

کنار گذاشته می‌شود

مرحله‌ی 1: ایجاد 16 زیرکلید هرکدام بطول 48bit

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 |

8×7

بطور مثال: جای 57 در کلید با 0 عوض می‌شود

جای 49 با 1 عوض می‌شود و الی آخر

مقادیر 8 تغییر می‌کنند.

64بیت → PC1 → 56بیت

k → k+

56 بیت موثر

56بیت

k+

$C_0$ (28بیت) | $D_0$ (28بیت)

مرحله‌ی 2: k+ به دونیم 28 تایی شکسته می‌شود:

در این مرحله 16 زیرکلید از $C_0$ تا $C_{16}$ و از $D_0$ تا $D_{16}$ تولید می‌شود.

روش تولید؟ ← براساس جدول مقابل:

| iteration number | number of rotate left shift |
|---|---|
| 1 | 1 |
| 2 | 1 |
| 3 | 2 |
| 4 | 2 |
| 5 | 2 |
| 6 | 2 |
| 7 | 2 |
| 8 | 2 |
| 9 | 1 |
| 10 | 2 |
| 11 | 2 |
| 12 | 2 |
| 13 | 2 |
| 14 | 2 |
| 15 | 2 |
| 16 | 1 |

← تعداد بیت‌هایی که باید به چپ شیفت داده شوند.

در نهایت 16 تا $C_i D_i$ داریم هرکدام 56 بیت

Scanned by CamScanner

بعداز بدست آوردن $C_{16}$ تا $C_1$ و $D_{16}$ تا $D_1$ بایداز جدول جابجایی 2 (PC-2) استفاده کنیم:

PC2

| | | | | | |
|---|---|---|---|---|---|
| 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 |

→ برای یک 16 زیرکلید (جسباندن C ها و D ها)

یعنی $k_{16}$ تا $k_1$ ازاین جدول استفاده کنیم

↓

مثال: در $k_1$: اولین بیت معادل 14 امین بیت در $C_1D_1$ و دومین بیت معادل 17 امین بیت $C_1D_1$ می شود و الی آخر.

$8 \times 6 = 48$

↓

درنهایت 16 تا $k$ داریم هرکدام 48 بیت

مرحله سوم: کرگذاری بلوک های داده ی 64 بیتی (M) ← DES core Function

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 |

$8 \times 8 = 64$

قدم اول: اعمال جدول IP (initial Permutation) روی (M)

→ کرگذاری 64 بیت بلوک M: طور مثال:

اولین بیت IP معادل 58 بیت M می شود

دومین بیت IP معادل 50 بیت M می شود

⋮

قدم دوم: شکستن IP به دو بخش 32 تایی:

واجرای 16 راند

64 بیت

IP

32 | 32

$(L_0)$ $(R_0)$

فرمول این 16 راند:

$$L(n) = R[n-1]$$

$$R(n) = L(n-1) \text{ XOR } f(R(n-1), k(n))$$

$n \rightarrow 1$ تا 16

همه ی پارامترها مشخص اند بجز کملکرد تابع (f)



Scanned by CamScanner

**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

Scanned by CamScanner

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

Scanned by CamScanner

## مرحله پنجم

نحوه ی استفاده ازS-boxes و یا Substitution boxes :

فرض کنید عدد ۴۸ بیتی باینری زیر را داریم که می خواهیم S-Boxes را بر آن اعمال کنیم :

011101000101110101000111010000111001011010111 01

همانطور که در قسمت قبل گفته شد ، ۸ گروه ۶ بیتی از آن استخراج می شود که از B1 تا B8 را تشکیل می دهند:

011101 000101 110101 000111 101000 011100 101101 011101

محاسبات زیر را در نظر بگیرید:

B[n] => S[n][row][column]

B[1] => S[1](01, 1110) = S[1][1][14] = 3 = 0011
B[2] => S[2](01, 0010) = S[2][1][2 ] = 4 = 0100
B[3] => S[3](11, 1010) = S[3][3][10] = 14 = 1110
B[4] => S[4](01, 0011) = S[4][1][3 ] = 5 = 0101
B[5] => S[5](10, 0100) = S[5][2][4 ] = 10 = 1010
B[6] => S[6](00, 1110) = S[6][0][14] = 5 = 0101
B[7] => S[7](11, 0110) = S[7][3][6 ] = 10 = 1010

Scanned by CamScanner

B[8] => S[8](01, 1110) = S[8][1][14] = 9 = 1001

نحوه ی محاسبه :

B[n] => S[n][row][column]

n : دقیقا مشخص است یا قدیمی B

Row : از کنار هم قرار گرفتن بیت اول و بیت آخر یک گروه ۶ بیتی فوق تشکیل می شود.

Column : مابقی بیت ها . ستون را تشکیل می دهند (یعنی از بیت های ۲ تا ۵) .

برای مثال:

011101را در نظر بگیرید

چون اولین گروه ۶ بیتی عدد ۴۸ بیتی ما را تشکیل می دهد . n=1 خواهد بود

برای تشکیل Row دو بیت اول و آخر کنار هم قرار می گیرند یعنی Row=01 .

Column تشکیل شده از بیت ۲ تا ۵ عدد فوق است یعنی Column=1110 .

نتیجه ی حاصل:

اولین سطر محاسباتی است که در بالا ذکر شد یعنی: S[n][row][column] = S[1](01, 1110)

برای محاسبه ی این موقعیت در جدول S1 . ابتدا اعداد درون پرانتزها به معادل دسیمال خود تبدیل می شوند داریم S[1][1][14]

جدول S1 هم به صورت زیر است (برای سهولت مراجعه شماره ردیف ها و ستون ها نیز نوشته شده است. صفر تا سه شماره ردیف ها هستند و صفر تا ۱۵ شماره ستون ها(رنگ آبی)) :

**S1 (ROW/Column)**

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

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

Scanned by CamScanner

سپس این موقعیت در جدول S1 پیدا می شود یعنی به جدول S1 مراجعه کرده و سپس عدد قرار گرفته در سطر ۱ و ستون ۱۴ را پیدا می کنیم این عدد معادل ۳ است. سپس آنرا به باینری تبدیل می نماییم.

بنابراین به صورت خلاصه داریم :

B[1] => S[1](01, 1110) = S[1][1][14] = 3 = 0011

به همین ترتیب برای سایر گروه های ۶ بیتی عمل می شود.

نتیجه این قسمت کنار هم قرار دادن نتایج حاصل فوق که یک عدد ۳۲ بیتی دودویی را تشکیل می دهد.

## مرحله ششم

پس از اعمال جداول جانشینی و یا همان S-Boxes بر روی Bi ها بر روی نتیجه ی حاصل یک جایگردانی دیگر صورت می گیرد.

| Permutation 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 |

یعنی به صورت خلاصه تابع f که در چند قسمت قبل راجع به آن بحث شد به صورت زیر محاسبه می شود:

f = P(S[1](B[1])...S[8](B[8]))

برای مثال :

S1(B1)S2(B2)S3(B3)S4(B4)S5(B5)S6(B6)S7(B7)S8(B8) = 0101 1100 1000 0010 1011 0101 1001 0111
==>
f = 0010 0011 0100 1010 1010 1001 1011 1011

Scanned by CamScanner

و در ادامه :

R[1] = L[0] XOR f(R[0] , K[1] )
= 1100 1100 0000 0000 1100 1100 1111 1111
XOR 0010 0011 0100 1010 1010 1001 1011 1011
= 1110 1111 0100 1010 0110 0101 0100 0100

و L1 هم که قبلا محاسبه شده بود:

L[1] = R[0] = 1111 0000 1010 1010 1111 0000 1010 1010

تا اینجا یک راند از ۱۶ راند الگوریتم پایان پذیرفت

شروع راند دوم:

مطابق فرمول کلی عنوان شده داریم :

L[2] = R[1]
R[2] = L[1] + f(R[1], K[2])

که محاسبه‌ی آن با توجه به مطالب گفته شده تاکنون ساده است این روبه تا پایان ۱۶ راند ادامه دارد.

در پایان راند ۱۶ ، حاصل کار L16 و R16 است در اینجا جای این دو بلاک معکوس می شود یعنی :

R[16]L[16]

و بر روی آن آخرین جایگردانی مطابق جدول زیر صورت می گیرد:

| 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 |

Scanned by CamScanner

برای مثال با توجه به اعداد انتخاب شده در این برنامه داریم :

L16 = 0100 0011 0100 0010 0011 0010 0011 0100
R16 = 0000 1010 0100 1100 1101 1001 1001 0101
===>
R[16]L[16] = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100
===>
IP^-1 = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101 (bin)
= 85E813540F0AB405 (hex)

و یا به صورت خلاصه :

M = 0123456789ABCDEF

کلید بکار گرفته شده برای کدگذاری:

Key = 13 34 57 79 9B BC DF F1

خروجی کدگذاری شده:

C = 85E813540F0AB405

تا اینجا بررسی الگوریتم کد کردن DES به پایان می رسد

Scanned by CamScanner