DES: Data Encryption Standard

INE5429 - Segurança em Computação

Caique Rodrigues Marques
c.r.marques@grad.ufsc.br

Nota: As imagens usadas foram retiradas do livro-texto[2].

3.2 Devido ao algoritmo de *key scheduler* apenas copiar os primeiros 8 *rounds* nos *rounds* de 9 a 16, na ordem inversa, é possível notar que o ciframento e o deciframento do cifrador de Feistel são iguais. Assim, para conseguir a mensagem m basta que o oráculo cifre c, assim o texto cifrado retornado vai ser o texto c decifrado, logo, a mensagem m.

3.7 A estrutura do DES constitui do uso da função de Feistel que é inversível, a partir disto é possível mostrar que o decifrador DES é o inverso do cifrador DES. Definindo uma função de Feistel F e a entrada composta por (L_1, R_1), o primeiro *round* do cifrador é definido como:

$$L_2 = R_1 \oplus F(L_1)$$
$$R_2 = L_1,$$

onde L_2 e R_2 compõem a saída cifrada resultante. Supondo que o cifrador possua apenas um *round*, é possível conseguir a entrada original apenas com a função F e a saída cifrada:

$$L_1 = R_2$$
$$R_1 = L_2 \oplus F(L_1).$$

Isso é possível devido a dois pontos:

(a) Parte da entrada é carregada ao fim de cada *round*;

(b) A operação de $\oplus$ entre quaisquer dois elementos garante a inversibilidade, assim,

- $A \oplus B = C$;
- $A \oplus C = B$;
- $B \oplus C = A$.

Tais propriedades são válidas porque o conjunto D, munido da operação de $\oplus$ entre dois elementos a e b quaisquer pertencentes a D, é um grupo abeliano.[1]

Portanto, a função de Feistel F é inversível.

Resta apenas as funções IP e IP^{-1}. Ambas as funções são de permutações, logo, apenas rearranja os bits e, por definição, IP^{-1} é o inverso de IP. Para ciframento:

- O texto original é entrada da função IP, resultando na saída composta por (L, R);
- A saída do item anterior é entrada da função de Feistel, resultando em (X, Y);
- Por fim, a saída do item anterior é entrada para a função IP^{-1}, resultando no texto cifrado c.

Para deciframento:

- O texto cifrado c é entrada da função IP, resultando na saída composta por (X, Y);
- A saída do item anterior é entrada da função de Feistel, resultando em (L, R);
- Por fim, a saída do item anterior é entrada para a função IP^{-1}, resultando no texto original.

Portanto, o deciframento DES é o inverso do ciframento DES.

3.12 Dadas as duas tabelas abaixo, a única coisa notável é que as tabelas são bem parecidas, a exceção está na *Permuted Choice One* que possui um bit a menos que a tabela de *Initial Permution* - ainda é possível notar alguns padrões de posicionamento entre ambas (começando pelo 57 em ambas, por exemplo), o que pode permitir alguma implementação similar da tabela IP na *PC-1*. De resto, não há mais nada de especial.

[1] Para ser classificado como um grupo abeliano é necessário satisfazer cinco axiomas.

(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) 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

3.13 Durante o deciframento, *rounds* com um 1 bit rotacionados são 2, 9 e 16, enquanto o restante (exceto o *round* 1) são rotacionados 2 bits.

Número do *round*	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits rotacionados	0	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

3.16 (a) Em questão de segurança, o permutador de 10 bits não é tão forte. Como ele apenas rearranja os 10 bits, então há 2^{10} possibilidades de chave, assim, com um computador atual é relativamente simples conseguir decifrar a entrada partindo-se da saída.

(b) Considerando a simplicidade da função citada na alternativa anterior, a função que faz deslocamento circular de 1 bit à esquerda também não adiciona muita segurança, além de ser ainda mais simples de deduzir a entrada.

3.17 Dadas as tabelas correspondentes às duas *S-boxes* do *simplified DES*:

$$S0 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{bmatrix} \end{array} \qquad S1 = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{bmatrix} \end{array}$$

Sendo `wxyz` uma entrada, os valores de `w` e `z` correspondem à linha do *S-box*, enquanto `x` e `y` correspondem à coluna. Por exemplo, sendo a entrada da *S-box* a sequência `0001`, então `wz = 01` e `xy = 00`, assim, o resultado está na linha de número um e coluna número zero da matriz $S1$, resultando em 2, que em binário é `10`, portanto, `s = 1` e `t = 0`. Montando toda essa relação em uma tabela verdade:

w	x	y	z	s	t
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	1	0
0	1	0	1	0	1
...	...	...	...	...	...
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	1	1

Com a tabela verdade acima, é possível montar um mapa de Karnaugh[1] e dele derivar as equações para `s` e `t`, porém, o resultado estará na forma normal disjuntiva. Segue as equações para as variáveis `s` e `t`:

$$s = \overline{x}\overline{y}z + \overline{w}x\overline{z} + xyz + w\overline{x}\overline{y},$$
$$t = \overline{w}y\overline{z} + \overline{w}xz + w\overline{y}\overline{z} + wyz.$$

3.18 Dado o texto cifrado $c =$ `10100010` e a chave $k =$ `0111111101`, iremos decifrar a mensagem c usando o *simplifified DES*. Dada a estrutura do SDES, começando pela parte de geração de chaves:

(a) **Permutação de 10 bits**: Ao aplicar a função de permutação de 10 bits na chave:
$P10(k) = 1111110011$;

(b) **Deslocamento circular de 1 bit à esquerda**: Aplicando deslocamento circular de 1 bit nas partições `11111` e `10011`:
$LS1(11111) = 11111$ e $LS1(10011) = 00111$;

(c) **Permutação de 8 bits**: A função de permutação de 8 bits nas duas partições anteriores e resultando na subchave k_1:
$P8_a(11111, 00111) = 01011111$;

(d) **Deslocamento circular de 2 bits à esquerda**: As duas partições `11111` e `10011` têm os 2 bits deslocados à esquerda:
$LS2(11111) = 11111$ e $LS2(10011) = 00111$;

(e) **Permutação de 8 bits**: Por fim, ambas as partições do item anterior são concatenadas e permutadas, resultado na subchave k_2:
$P8_b(11111, 00111) = 11111100$.

Seguindo com a estrutura do SDES:

(a) **Permutação inicial**: O texto cifrado c de 8 bits é entrada da função `IP` que realiza a permutação inicial:
$IP(c) = 00110001$;

(b) **Expansão e permutação**: Os 4 bits menos significativos do resultado da permutação de c do item anterior é entrada da função `E/P`, que realiza a expansão para 8 bits e permuta-os:
$E/P(0001) = 10000010$;

(c) **XOR**: Realizando a operação de ou-exclusivo (XOR) com o resultado do item anterior e a subchave k_2:
$10000010 \oplus k_2 = 01111110$;

(d) ***S-boxes***: Os bits mais e menos significativos do resultado da operação XOR anterior são entradas para as funções $S0$ e $S1$, respectivamente. Dos quatro bits, o quarto e o primeiro bits correspondem à linha da S-box, enquanto os dois bits do meio restantes correspondem à coluna:
$S0(01, 11) = 00$ e $S1(10, 11) = 00$;

(e) **Permutação de 4 bits**: Os quatro bits resultantes do passo anterior são concatenados e permutados:
$P4(00, 00) = 0000$;

(f) **XOR**: Por fim, há a operação de ou-exclusivo (XOR) entre os quatro bits do passo anterior e o os quatro bits mais significativos resultantes da permutação inicial do texto cifrado c:
$0011 \oplus 0000 = 0011$.

A função `Switch` recebe o resultado final da última operação XOR e os 4 bits menos significativos resultantes da permutação inicial do texto cifrado c. Nesta função os 4 bits menos significativos passam a ser os mais significativos e vice-versa:
$SW(0011, 0001) = 00010011$

Seguindo a estrutura final do SDES:

(a) **Expansão e permutação**: Os 4 bits menos significativos do resultado da função `Switch` são entradas desta função:
$E/P(0011) = 10010110$;

(b) **XOR**: Realizando a operação de ou-exclusivo (XOR) entre a subchave k_1 e o resultado do passo anterior:
$10010110 \oplus k_2 = 11001001$;

(c) ***S-Boxes***: Os bits mais e menos significativos do resultado da operação XOR anterior são entradas para as funções $S0$ e $S1$, respectivamente. Dos quatro bits, o quarto e o primeiro bits correspondem à linha da S-box, enquanto os dois bits do meio restantes correspondem à coluna:
$S0(10, 10) = 01$ e $S1(11, 00) = 10$;

(d) **Permutação de 4 bits**: Os quatro bits resultantes do passo anterior são concatenados e permutados:
$P4(0110) = 1010$

(e) **XOR**: Há a operação de ou-exclusivo (XOR) entre os quatro bits do passo anterior e o os quatro bits mais significativos resultantes da função `Switch`:
$0001 \oplus 1010 = 1011$

(f) IP^{-1}: Por fim, é realizado a inversa da permutação inicial com a concatenação do resultado anterior e com os 4 bits menos significativos resultantes da função `Switch`:
$IP^{-1}(1011, 0011) = 11101010$.

O texto decifrado é `11101010`, sendo que a cada quatro bits corresponde a uma codificação a uma letra do alfabeto, assim, `1110` corresponde à letra `O` e `1010` corresponde à letra `K`. Portanto o texto final decifrado é `OK`.

Referências

[1] Karnaugh, M. The map method for synthesis of combinational logic circuits. *Transactions of the American Institute of Electrical Engineers*, 72 (November 1953).

[2] Stallings, W. *Cryptography and Network Security: Principles and Practice*, 6th ed. Pearson, 2014.