Szyfry produktowe

Rozważmy dwa systemy kryptograficzne, w których zbiór tekstów jawnych jest równy zbiorowi kryptogramów:

$$S_1 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_1), \qquad S_2 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_2)$$

$\mathcal{K}_1$ jest przestrzenią kluczy systemu S_1, $\mathcal{K}_2$ jest przestrzenią kluczy systemu S_2.

Szyfrem produktowym nazywamy szyfr

$$S = S_1 \times S_2 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_1 \times \mathcal{K}_2).$$

W systemie produktowym klucz ma postać $K = (K_1, K_2)$, gdzie $K_1 \in \mathcal{K}_1$ jest kluczem w systemie S_1, a $K_2 \in \mathcal{K}_2$ jest kluczem w systemie S_2.

Przestrzeń kluczy systemu $S_1 \times S_2$ ma postać $\mathcal{K}_1 \times \mathcal{K}_2$.

Szyfry produktowe

Szyfrowanie tekstu jawnego x w systemie produktowym ma postać

$$e_{(K_1,K_2)} = e^2_{K_2}\big(e^1_{K_1}(x)\big),$$

gdzie $e^1_{K_1}$ jest regułą szyfrowania w systemie S_1 z kluczem K_1, analogicznie $e^2_{K_2}$.

Deszyfrowanie kryptogramu y przebiega według reguły

$$d_{(K_1,K_2)} = d^1_{K_1}\big(d^2_{K_2}(y)\big),$$

gdzie $d^1_{K_1}$ jest regułą deszyfrowania w systemie S_1 z kluczem K_1, analogicznie $d^2_{K_2}$.

Szyfry produktowe

Szyfr idempotentny:

$$S^2 = S \times S = S.$$

Dla szyfru idempotentnego zachodzi:

$$\forall x \qquad \forall K_1, K_2 \qquad \exists K_3 \qquad\qquad e_{K_2}(e_{K_1}(x)) = e_{K_3}(x).$$

[2mm]Przykłady szyfrów idempotentnych: szyfr Cezara, afiniczny, Hilla.

Rzeczywiście ...

Szyfr idempotentny inaczej nazywa się mówiąc, że szyfr tworzy grupę.

Gdy szyfr tworzy grupę to dwukrotne szyfrowanie, z dwoma różnymi kluczami, jest równoważne szyfrowaniu jednokrotnemu z jakimś innym kluczem.

Gdy szyfr tworzy grupę to dwukrotne szyfrowanie nie zwiększa rozmiaru przestrzeni klucza.

Szyfry iterowane

Szyfrowanie przebiega w N podobnych rundach.

W każdej rundzie z wykorzystaniem klucza rundy szyfrujemy rezultat poprzedniej rundy.

W pierwszej rundzie szyfrujemy tekst jawny, rezultatem ostatniej rundy jest kryptogram.

Klucze kolejnych rund, $K_1, \ldots, K_N$, otrzymujemy stosując do klucza K pewną, określoną dla danego szyfru, regułę.

Klucze kolejnych rund nazywane są też podkluczami.

Ciąg kolejnych podkluczy $K_1, \ldots, K_N$ nazywamy schematem kluczy (ang. *key schedule*).

Szyfry iterowane

Szyfrowanie ma przebieg:

$$\begin{aligned} w^0 &= x \\ w^1 &= g(w^0, K_1) \\ w^2 &= g(w^1, K_2) \\ &\vdots \\ w^{n+1} &= g(w^n, K_{n+1}) \\ &\vdots \\ w^N &= g(w^{N-1}, K_N) \\ y &= w^N \end{aligned}$$

x – tekst jawny,

$K_1, \ldots, K_N$ – klucze kolejnych rund

y – kryptogram

Szyfry iterowane

Aby możliwe było odszyfrowanie kryptogramu, funkcja g realizujaca szyfrowanie w poszczególnych rundach musi posiadać funkcję odwrotną g^{-1} spełniającą warunek

$$g^{-1}\big(g(w, K), K\big) = w.$$

Wtedy oszyfrowywanie przebiega następująco

$$\begin{aligned} w^N &= y \\ w^{N-1} &= g^{-1}(w^N, K_N) \\ &\vdots \\ w^n &= g^{-1}(w^{n+1}, K_n) \\ &\vdots \\ w^0 &= g^{-1}(w^1, K_1) \\ x &= w^0 \end{aligned}$$

Sieci podstawieniowo–przestawieniowe

Sieci podstawieniowo–przestawieniowe (SPP) są szyframi blokowymi.

Ograniczymy się do szyfrowania ciągów bitów.

Niech l i m będą liczbami naturalnymi. Długość bloku SPP to ml.

SPP składa się z następujących operacji

$$\pi_S \colon \{0,1\}^l \to \{0,1\}^l$$

$$\pi_P \colon \{0,1,2,\ldots,lm\} \to \{0,1,2,\ldots,lm\}$$

Przekształcenie π_S nazywane jest S-blokiem (ang. *S-box, S – substitution*).
π_P jest permutacją lm bitów.

Ciąg x lm bitów uważać możemy za składający się z m podciągów l-bitowych:

$$x = x^{(1)} \,||\, x^{(2)} \,||\, \ldots \,||\, x^{(m)},$$

gdzie dla każdego $1 \leq k \leq m$

$$x^{(k)} = x_1^{(k)} x_2^{(k)} \ldots x_l^{(k)}.$$

Sieci podstawieniowo–przestawieniowe

Niech x będzie tekstem jawnym, $(K_1, \ldots, K_{N+1})$ – ciągiem podkluczy SPP. Schemat działania SPP wygląda następująco:

$$\begin{aligned}
&w^0 = x \\
&\quad u^1 = w^0 \oplus K_1 \\
&\qquad v^{1(1)} = \pi_S(u^{1(1)}), \quad v^{1(2)} = \pi_S(u^{1(2)}), \ldots, v^{1(m)} = \pi_S(u^{1(m)}) \\
&w^1 = (v^1_{\pi_P(1)}, v^1_{\pi_P(2)}, \ldots, v^1_{\pi_P(lm)}) \\
&\quad u^2 = w^1 \oplus K_2 \\
&\qquad v^{2(1)} = \pi_S(u^{2(1)}), \quad v^{2(2)} = \pi_S(u^{2(2)}), \ldots, v^{2(m)} = \pi_S(u^{2(m)}) \\
&\vdots \\
&w^{N-1} = (v^{N-1}_{\pi_P(1)}, v^{N-1}_{\pi_P(2)}, \ldots, v^{N-1}_{\pi_P(lm)}) \\
&\quad u^N = w^{N-1} \oplus K_N \\
&\qquad v^{N(1)} = \pi_S(u^{N(1)}), \quad v^{N(2)} = \pi_S(u^{N(2)}), \ldots, v^{N(m)} = \pi_S(u^{N(m)}) \\
&y = v^N \oplus K_{N+1}
\end{aligned}$$

Sieci podstawieniowo–przestawieniowe

$$\begin{aligned}
&w^0 = x \\
&\quad u^1 = w^0 \oplus K_1 \\
&\qquad v^{1(1)} = \pi_S(u^{1(1)}), \quad v^{1(2)} = \pi_S(u^{1(2)}), \ldots, v^{1(m)} = \pi_S(u^{1(m)}) \\
&w^1 = (v^1_{\pi_P(1)}, v^1_{\pi_P(2)}, \ldots, v^1_{\pi_P(lm)}) \\
&\quad u^2 = w^1 \oplus K_2 \\
&\qquad v^{2(1)} = \pi_S(u^{2(1)}), \quad v^{2(2)} = \pi_S(u^{2(2)}), \ldots, v^{2(m)} = \pi_S(u^{2(m)}) \\
&\vdots \\
&w^{N-1} = (v^{N-1}_{\pi_P(1)}, v^{N-1}_{\pi_P(2)}, \ldots, v^{N-1}_{\pi_P(lm)}) \\
&\quad u^N = w^{N-1} \oplus K_N \\
&\qquad v^{N(1)} = \pi_S(u^{N(1)}), \quad v^{N(2)} = \pi_S(u^{N(2)}), \ldots, v^{N(m)} = \pi_S(u^{N(m)}) \\
&y = v^N \oplus K_{N+1}
\end{aligned}$$

Operacje zaznaczone na czerwono (wstępna i finalna operacja XOR bloku z podkluczami) noszą nazwę *whitening*.

Sieci podstawieniowo–przestawieniowe

Przykład

Niech $m = l = 2$, $N = 4$. Definiujemy:[2mm]S-blok π_S:

z	00	01	10	11
$\pi_S(z)$	10	11	01	00

Permutacja π_P:

z	1	2	3	4
$\pi_P(z)$	4	1	2	3

Jako klucz wyjściowy wybieramy ciąg ośmiobitowy $K = (k_1, \ldots, k_8)$.[2mm]Schemat konstrukcji podkluczy wybieramy następująco:

$$K_i = (k_{2i-1}, k_{2i}, k_{2i+1}, k_{2i+2}), \quad i = 1, \ldots, 5.$$

Zatem gdy $K = 01000110$ to:$K_1 = 0100$, $K_2 = 0001$, $K_3 = 0110$, $K_4 = 1001$, $K_5 = 0100$.

Sieci podstawieniowo–przestawieniowe

SPP można łatwo zmodyfikować, na przykład:

- Można użyć różnych S-bloków do przekształcania różnych części bloku w rudzie lub w różnych rundach.
- zastępując kończącą każdą rundę permutację inną liniową operacją odwracalną

Najbardziej znane współczesne algorytmy symetryczne, DES i AES, mają postać nieco zmodyfikowanych SPP.[2mm]Dobrze skonstruowane SPP są odporne na wszystkie znane publicznie ataki kryptoanalityczne.

Sieć Feistela

Pojedyncza, runda sieci Feistela ma postać:

- Blok tekstu dzielimy na dwie połówki, lewą L_i oraz prawą R_i
- Funkcja rundy ma następującą postać:

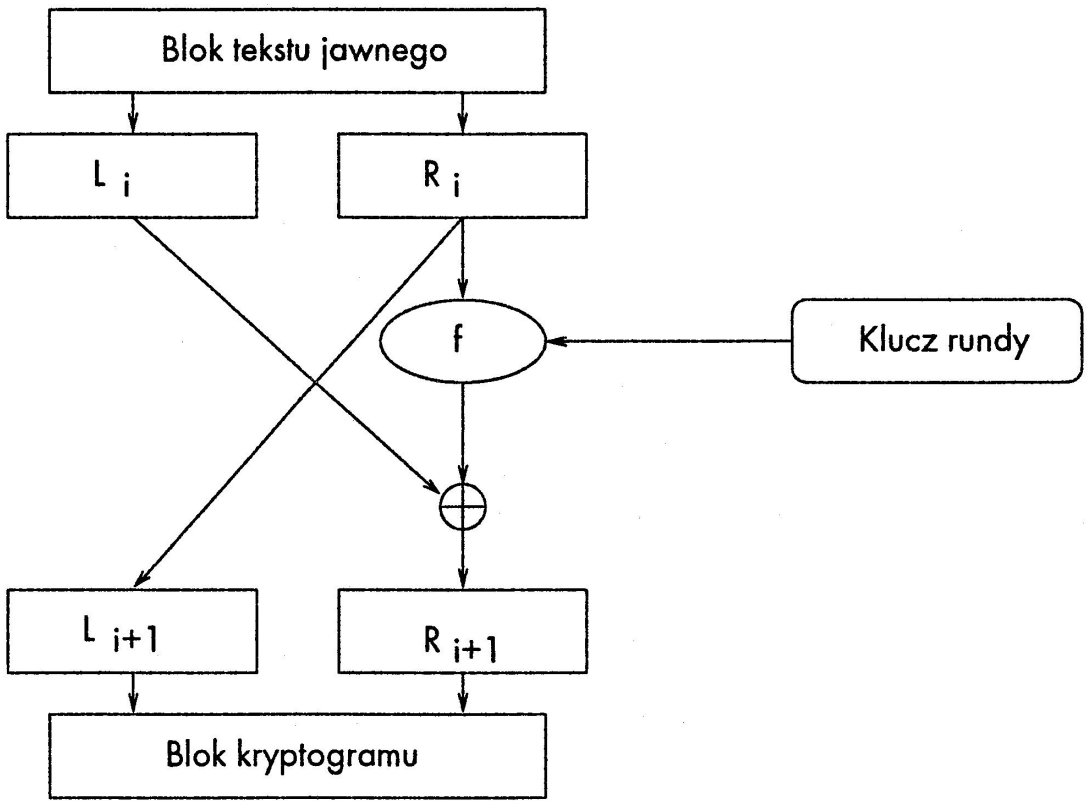
$$g(L_i, R_i, K_{i+1}) = (L_{i+1}, R_{i+1})$$

gdzie

$$\begin{aligned} L_{i+1} &= R_i \\ R_{i+1} &= L_i \oplus f(R_i, K_{i+1}) \end{aligned}$$

$\oplus$ oznacza dodawanie modulo 2.

Sieć Feistela



[2mm]

Schemat jednej rundy sieci Feistela

Sieć Feistela

Pojedyncza runda sieci Feistela jest zawsze odwracalna, *niezależnie od postaci funkcji* f

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

Powstanie DES

- W 1973 roku amerykańskie Narodowe Biuro Standardów (*National Bureau of Standards*) ogłosiło otwarty konkurs na bezpieczny algorytm szyfrujący.
- Żadna ze zgłoszonych propozycji nie zbliżyła się do stawianych wymagań.
- W 1974 roku konkurs ogłoszono ponownie.
- Konkurs wygrała propozycja przedstawiona przez zespół IBM (algorytm LUCIFER).
- Zaproponowany algorytm został przesłany przez NBS do oceny do amerykańskiej Narodowej Agencji Bezpieczeństwa (*National Security Agency – NSA*).
- Po ocenie i poprawkach dokonanych przez NSA algorytm został w 1976 roku oficjalnie zatwierdzony jako standard szyfrowania danych (*Data Encryption Standard – DES*).
- Do dziś trwają kontrowersje jakie (i czy w ogóle) poprawki zostały dokonane przez NSA w wyjściowym algorytmie i czy w algorytm nie zostało wbudowane „boczne wejście”

DES

DES jest 16-rundową siecią Feistela o długości bloku 64 bity.[2mm]Długość klucza stosowanego w DES to 56 bitów.

Podsumowując, DES szyfruje 64 bitowy blok tekstu z użyciem klucza o długości 56 bitów i w rezultacie otrzymujemy kryptogram o długości 64 bitów.

Przed 16 rundami szyfrowania blok tekstu jawnego jest poddawany ustalonej permutacji początkowej, IP (*Initial Permutation*)

$$IP(x) = L_0 R_0$$

Po 16 rundach szyfrowania stosowana jest permutacja odwrotna IP^{-1}

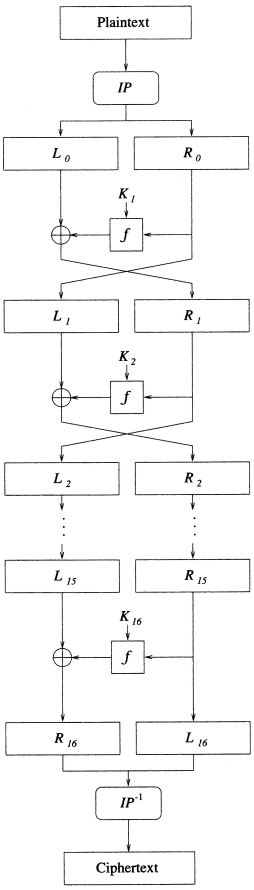
$$y = IP^{-1}(R_{16} L_{16})$$

Należy zwrócić uwagę, że przed zastosowaniem IP^{-1} połówki bloku $L_{16}R_{16}$ zamieniane są miejscami.

Permutacje IP oraz IP^{-1} nie mają znaczenia kryptograficznego.

DES

Schemat algorytmu DES



DES

Pojedyncza runda DES ma postać taką jak runda sieci Feistela, w której L_i oraz R_i są łańcuchami 32-bitowymi a funckcja f ma postać:

$$f\colon \{0,1\}^{32} \times \{0,1\}^{48} \to \{0,1\}^{32}$$

gdzie 48-bitowy łańcuch bitowy jest kluczem rundy.

Ciąg szesnastu 48-bitowych kluczy rund $K_1, \ldots, K_{16}$ jest wyprowadzany z 56-bitowego klucza K.

Każdy klucz K_i jest permutacją wybranych bitów klucza K.

Oznaczmy przez A pierwszy, 32-bitowy, argument funkcji f a przez J drugi, 48-bitowy, argument tej funkcji.

$A \equiv R_i$ – prawa połówka bloku tekstu otrzymanego po zakończeniu poprzedniej rundy,
<all>[2mm]– klucz K_{i+1} rundy.

DES

Aby wyliczyć wartość funkcji $f(A, J)$ wykonujemy następujące operacje:

- 32-bitowy łańcuch A jest rozszerzany do długości 48 bitów za pomocą ustalonej funkcji rozszerzającej E (funkcja E nazywana jest też czasem permutacją rozszerzającą). $E(A)$ składa się z 32 przepermutowanych bitów A wraz z 16 bitami z A pojawiającymi się ponownie. Dokladna postać funkcji E podamy później.
- Obliczamy $B = E(A) \oplus J$. 48-bitowy łańcuch B dzielimy na 8 łańcuchów 6-cio bitowych $B = B_1B_2B_3B_4B_5B_6B_7B_8$. B_1 składa się z bitów 1-6, B_2 z bitów 7-12, $\ldots$, B_8 z bitów 43-48.

DES

- W następnym kroku wykorzystywanych jest 8 S-bloków, oznaczanych $S_1, \ldots, S_8$. Każdy z S-bloków jest przekształceniem łańcucha 6-bitowego w łańcuch 4-bitowy:

  $$S_i \colon \{0,1\}^6 \to \{0,1\}^4.$$

  Dokładna postać S-bloków używanych w DES zostanie podana później.
  Wyliczamy:

  $$C_1 = S_1(B_1), \ldots, C_8 = S_8(B_8).$$

  Otrzymane 8 łańcuchów 4-bitowych zapisujemy w postaci jednego łańcucha:

  $$C = C_1C_2C_3C_4C_5C_6C_7C_8.$$

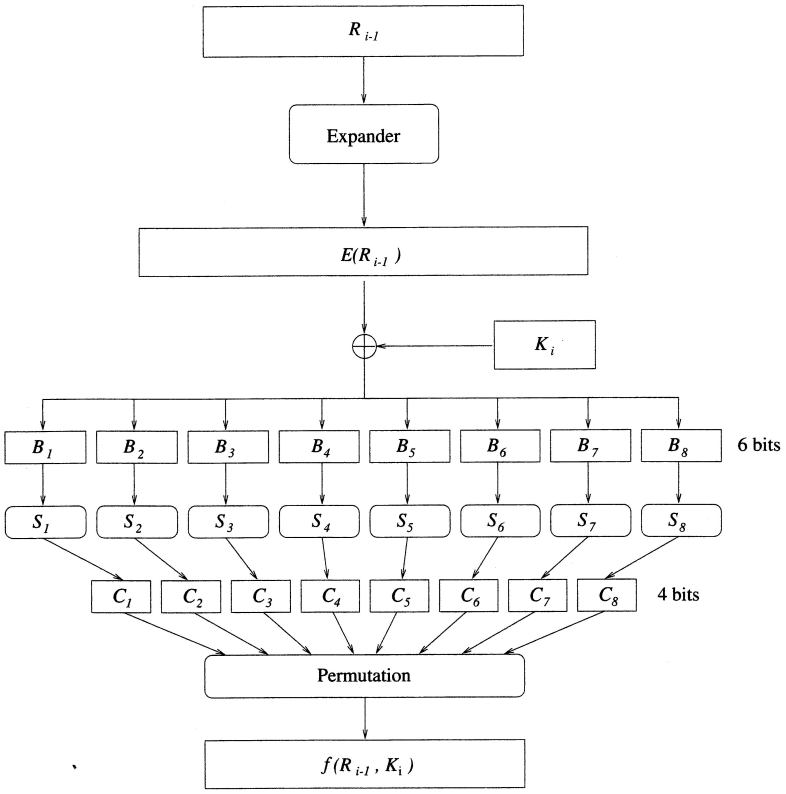
  Otrzymujemy w ten sposób 32-bitowy łańcuch C.

- W ostatnim kroku łańcuch C poddajemy permutacji P:

  $$P(C) = f(A, J).$$

DES

Schemat funkcji rundy algorytmu DES



DES – klucze rund

Startujemy z klucza o długości 64 bitów. W kluczu tym 8 bitów to bity kontroli parzystości (są to ostatnie bity kolejnych 8 bajtów, czyli bity o numerach 8, 16, . . . , 64).

- Odrzucamy bity kontroli parzystości.
- Pozostałe 56 bitów znaczących permutujemy według tabeli

Key Permutation

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

- Rezultat permutacji dzielimy na dwa łańcuchy 28-bitowe M_0N_0 (M_0 to pierwsze 28 bitów, N_0 ostatnie 28 bitów).

DES – klucze rund

- Dla rundy o numerze $1 \leq i \leq 16$ definiujemy

$$M_i = LS(M_{i-1}), \qquad N_i = LS(N_{i-1}),$$

gdzie LS_i oznacza cykliczne przesunięcie bitów w lewo o 1 lub 2 pozycje. Wartość przesunięcia w zależności od numeru rundy określa poniższa tabela

Number of Key Bits Shifted per Round																
Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Shift	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

- Jako klucz i-tej rundy wybieramy 48 bitów z 56-bitowego łańcucha $M_i N_i$. Poniższa tabelka określa, które bity należy wybrać

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

Okazuje się, że każdy bit klucza wyjściowego występuje średnio w 14 kluczach rund (na 16 wszystkich rund).

DES

Permutacja początkowa IP ma postać

Permutacja początkowa 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

Powyższy zapis permutacji należy rozumieć następująco, gdy

$$x = x_1 x_2 \dots x_{64}$$

gdzie $x_1, \dots, x_{64}$ to kolejne bity bloku x, to

$$IP(x) = x_{58} x_{50} x_{42} \dots x_{23} x_{15} x_7.$$

DES

Przekształcenie rozszerzające E ma postać:

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

Powyższy zapis przekształcenia E należy rozumieć następująco, gdy

$$x = x_1 x_2 \dots x_{32}$$

gdzie $x_1, \dots, x_{32}$ to kolejne bity bloku x, to

$$E(x) = x_{32} x_1 x_2 \dots x_{31} x_{32} x_1.$$

DES

Permutacja P ma postać

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

Powyższy zapis permutacji P należy rozumieć następująco, gdy

$$x = x_1 x_2 \dots x_{32}$$

gdzie $x_1, \dots, x_{32}$ to kolejne bity bloku x, to

$$E(x) = x_{16} x_7 x_{20} \dots x_{11} x_4 x_{25}.$$

DES

Każdy S-box jest tradycyjnie przedstawiany jako jako macierz o 4 wierszach i 16 kolumnach. Elementy tej macierzy są liczbami całkowitymi ze zbioru $\{0, 1, \dots, 15\}$.

S-box 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

Wartość $S_i(b_1 b_2 b_3 b_4 b_5 b_6)$ jest elementem tej macierzy, który znajduje się na przecięciu:
wiersza o numerze $b_1 b_6$ (w zapisie binarnym)
kolumny o numerze $b_2 b_3 b_4 b_5$ (w zapisie binarnym)

Wiersze są numerowane od 0 do 3, kolumny od 0 do 15.

Przykład: Policzyć $S_8(011001)$.

DES

S-bloki używane w DES

S-box 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-box 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

DES

S-bloki używane w DES

S-box 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-box 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

DES

S-bloki używane w DES

S-box 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-box 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

DES

S-bloki używane w DES

S-box 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-box 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

DES

S-boxy są podstawą bezpieczeństwa DES.[2mm]Kryteria wyboru S-boxów zostały opublikowane przez IBM dopiero na początku lat 90-tych XX wieku. Oto kryteria:

- Każdy S-box przekształca 6 bitów na 4. Była to maksymalna ilość jaką dało się umieścić na jednym chipie w roku 1974.
- Rezultat S-boxu nie powinien być funkcją liniową ani afiniczną bitów wejściowych.
- Każdy wiersz każdego S-boxu zawiera wszystkie liczby od 0 do 15.
- Jeżeli dwa wejściowe łańcuchy bitów w S-boxie różnia się jednym bitem to rezultaty muszą się różnić 2 bitami.
- Dla dowolnego S-boxu i ciągu wejściowego x, ciągi $S(x)$ oraz $S(x \oplus 001100)$ różnią się co najmniej dwoma bitami.

Znalezienie S-boxów spełniających te kryteria zajęło w latach 70–tych kilka miesięcy pracy ówczesnych komputerów.[2mm]Tak sformułowane kryteria (oraz liczba rund algorytmu) wskazują, że twórcy algorytmu chcieli zabezpieczyć go przed pewnymi rodzajami ataków (kryptoanalizą liniową i różnicową), które publicznie stały się znane dopiero w latach 90-tych XX wieku.

DES

Bezpieczeństwo DES

- Przestrzeń kluczy DES to 2^{56}. Już w czasie, gdy DES powstawał podnoszono zarzuty, że klucz jest zbyt krótki.Algorytm LUCIFER, na podstawie którego powstał DES miał klucz długości 128 bitów, pierwotna wersja DES miała klucz długości 64 bitów. IBM oficjalnie twierdzi, że klucz w DES zredukowano do 56 bitów, żeby 8 bitów przeznaczyć na kontrolę parzystości.
- Jak dotąd najlepszą metodą ataku na DES jest wyczerpujące przeszukiwanie przestrzeni kluczy.
- Stosowane są dwie strategie
  - obliczenia rozproszone – wykorzystanie bardzo wielu komputerów połączonych przez Internet,
  - budowa specjalnych maszyn dedykowanych do przeszukiwania przestrzeni kluczy.

DES

Złamanie DES

- W 1997 roku firma RSA Data Security ogłosiła tzw. DES Challenge – konkurs na złamanie wiadomości zaszyfrowanej za pomocą DES (nagroda 10 000 $).Po 5 miesiącach od ogłoszenia zgłosił się zwycięzca – Rocke Verser.Zastosowana metoda – wyczerpujące przeszukanie przestrzeni kluczy z wykorzystaniem techniki obliczeń rozproszonych.Zwycięzca napisał program, który ochotnicy mogli uruchomić w czasie wolnym na swoich komputerach i z wykorzystaniem Internetu brać udział w przeszukiwaniu przestrzeni kluczy.Klucz został znaleziony po przeszukaniu około 1/4 przestrzeni kluczy.

DES

Złamanie DES

- W 1998 RSA Data Security ogłosiła DES Challenge II-1Zwycięzcą została firma Distributed Computing Technologies, która znalazła klucz w czasie 39 dni po przeszukaniu około 85% przestrzeni kluczy z wykorzystaniem techniki obliczeń rozproszonych.
- DES Challenge II-2 został zakończony w lipcu 1998Zwycięzcą została firma EFF (*Electronic Frontier Foundation*), która zbudowała specjalny komputer, nazywany DES-Crack, przeznaczony do łamania DES metodą wyczerpującego przeszukiwania przestrzeni kluczy. Koszt DES-Crack wyniósł około 250 000 $.DES-Crack znalazł klucz DES w czasie około 56 godzin.
- DES Challenge III zakończył się 19 stycznia 1999DES-Crack we współpracy z 100 000 komputerami (przy wykorzystaniu Internetu) znalazł klucz DES w czasie około 22 godzin.

DES

Złamanie DES

- Obecnie (2018) w sieci można znaleźć ofertę [`https://crack.sh`] złamania DES przez wyczerpujące przeszukanie całej przestrzeni kluczy.Koszt – 30 USD, czas oczekiwania na rezultat – kilka dni.Łamanie wykonuje specjalna, dedykowana maszyna realizująca sprzętowo wielokrotne równoległe szyfrowanie (deszyfrowanie) DES.Pełne przeszukanie przestrzeni klucza trwa 26 h (wg. informacji na stronie internetowej).

DES nie jest bezpieczną metodą szyfrowania.

DES

Oznaczmy przez E_K^{DES} funkcję realizującą szyfrowanie za pomocą algorytmu DES z kluczem K, przez D_K^{DES} funkcję realizującą deszyfrowanie za pomocą algorytmu DES z kluczem K.[2mm]Zatem jeżeli x jest tekstem jawnym to kryptogram ma postać

$$y = E_K^{\mathsf{DES}}(x)$$

oraz zachodzi

$$x = D_K^{\mathsf{DES}}(y).$$

Można pokazać, że w ogólności jeżeli:

$$y = E_{K_2}^{\mathsf{DES}}\big(E_{K_1}^{\mathsf{DES}}(x)\big)$$

to nie istnieje taki klucz K_3, że

$$y = E_{K_3}^{\mathsf{DES}}(x).$$

DES nie tworzy grupy.

DES

Zatem podwójne zaszyfrowanie za pomocą DES nie jest równoważne pojedynczemu szyfrowaniu.[2mm]Można zatem przypuszczać, że wielokrotne szyfrowanie przy użyciu DES zwiększy bezpieczeństwo (zwiększa się rozmiar przestrzeni kluczy).[2mm]Znany jest atak kryptograficzny (*meet-in-the-middle*), który powoduje, że podwójny DES jest tylko nieznacznie bezpieczniejszy od zwykłego DES (przestrzeń kluczy po zastosowaniu tego ataku ze spodziewanego rozmiaru 2^{112} redukuje się efektywnie do 2^{57}).

Atak *meet-in-the-middle*

E_k^{A} – funkcja realizująca szyfrowanie za pomocą algorytmu A z kluczem k,

D_k^{A} – funkcja realizująca deszyfrowanie za pomocą algorytmu A z kluczem k.

Zakładamy, że A nie tworzy grupy.[2mm]Stosujemy podwójne szyfrowanie tekstu jawnego x, z dwoma różnymi kluczami k_1, k_2:

$$y = E_{k_2}^{\mathsf{A}}\big(E_{k_1}^{\mathsf{A}}(x)\big)$$

Jeżeli przestrzeń kluczy algorytmu A ma rozmiar N to można przypuszczać, że podwójne szyfrowanie zwiększy jej rozmiar do N^2.[1mm]*Atak meet-in-the-middle zmniejsza tę wartość do* $2N$.

Atak *meet-in-the-middle*

Załóżmy, że Ewa przechwytuje tekst jawny x i odpowiadający mu kryptogram y i na tej podstawie chce wyznaczyć klucze k_1 i k_2.

- Ewa wylicza i przechowuje $E_k^{\mathsf{A}}(x)$ dla wszystkich możliwych kluczy k,
- Ewa wylicza i przechowuje $D_k^{\mathsf{A}}(y)$ dla wszystkich możliwych kluczy k,
- Ewa porównuje obie listy i wybiera identyczne rezultaty; co najmniej jeden rezultat na obu listach będzie identyczny

$$z = E_{k_1}^{\mathsf{A}}(x), \qquad y = E_{k_2}^{\mathsf{A}}(z),$$
$$z = D_{k_2}^{\mathsf{A}}(y), \qquad x = D_{k_1}^{\mathsf{A}}(z).$$

- W ten sposób Ewa znajduje poszukiwane klucze k_1 i k_2.

Żeby znaleźć klucze k_1 i k_2 Ewa musi wykonać N szyfrowań i N deszyfrowań, zatem przestrzeń kluczy koniecznych do sprawdzenia ma rozmiar $2N$.

DES

Dlatego rozpowszechniony jest tak zwany Triple-DES (3DES), w którym szyfrowanie tekstu jawnego x przebiega następująco:

$$y = E_{K_3}^{\mathsf{DES}}\Big(D_{K_2}^{\mathsf{DES}}\big(E_{K_1}^{\mathsf{DES}}(x)\big)\Big).$$

Gdy wybierzemy $K_1 = K_2$ to 3DES jest równoważny zwykłemu DES. [2mm]Przestrzeń kluczy 3DES to około 2^{112}.[2mm]Inną metodą wzmocnienia DES jest tak zwany DESX, w którym używa się 3 kluczy K_1, K_2, K_3 a szyfrowanie przebiega następująco

$$y = K_3 \oplus E_{K_2}^{\mathsf{DES}}(K_1 \oplus x).$$