

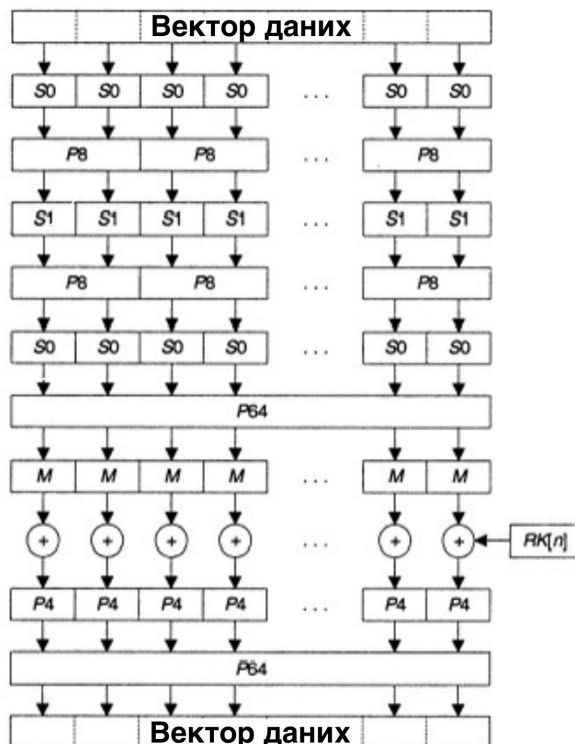
## Алгоритм ICEBERG

Алгоритм шифрування ICEBERG запропонований відносно нещодавно у 2004 р. – французьким криптологом Жилем-Франсуа Піре (Gilles-François Piret).

ICEBERG - це аббревіатура від “Involutional Cipher Efficient for Block Encryption in Reconfigurable Hardware”, “Інволюційний шифр для ефективного блокового шифрування в реконфігурованому (перебудованому) апаратному забезпеченні”. Як видно з назви, цей алгоритм оптимізовано під апаратні реалізації за допомогою програмованих логічних інтегральних схем.

### Структура алгоритму

Алгоритм ICEBERG шифрує дані 64-бітових блоків з використанням 128-бітного ключа шифрування. Оброблюваний блок даних подається у вигляді 64-бітового вектора, над яким у кожному раунді алгоритму послідовно виконуються такі операції:



1. Таблична заміна  $S_0$ . Вектор даних подається у вигляді 16 значень по 4 біти, кожне з яких заміщається згідно з таблицею:

Вхідне значення	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Вихідне значення	D	7	3	2	9	A	C	1	F	4	5	E	6	0	B	8

2. Бітова перестановка P8, що переставляє біти вектору даних за наступним правилом:

$$y_{8i+j} = x_{8i+p8(j)}, i = 0..7, j = 0..7,$$

де:

- $x_n$  та  $y_n$  - відповідно, вхідний та вихідний біти вектору даних;
- функція  $p8()$  визначена згідно таблиці:

Вхідне значення	0	1	2	3	4	5	6	7
Вихідне значення	0	1	4	5	2	3	6	7

3. Таблична заміна  $S_1$ , що працює аналогічно до заміни  $S_0$ , але згідно з іншою таблицею:

Вхідне значення	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Вихідне значення	4	A	F	C	0	D	9	B	E	6	1	7	3	5	8	2

4. Знову виконується перестановка P8, після якої повторно застосовується таблична заміна  $S_0$ .

5. Бітова перестановка P64, що переставляє біти вектору даних наступним чином:

$$y_i = x_{p64(i)}, i = 0..63,$$

де функція  $p64()$  визначена згідно таблиці:

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

Згідно таблиці вхідним значенням 0, 1, 2, ... відповідають вихідні значення 0, 12, 23 і т.д.

6. Операція множення на матрицю М. Застосовується до кожного 4-бітного фрагмента вектору даних шляхом його множення на фіксовану матрицю

0	1	1	1
1	0	1	1
1	1	0	1
1	1	1	0

Це аналогічно до застосування наступної табличної заміни:

<b>Вхідне значення</b>	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
<b>Вихідне значення</b>	0	E	D	3	B	5	6	8	7	9	A	4	C	2	1	F

7. Накладання матеріалу ключа (операція  $\delta$ ):

$$y_i = x_i \oplus RK[n]_i,$$

де  $RK[n]_i$  - і-й біт фрагмента розширеного ключа для n-го раунду (процедура розширення ключа буде описана далі).

8. Бітова перестановка P4, яка застосовується до кожного 4-бітного фрагмента векторних даних:

$$y[i]_j = x[i]_{p4(j)}, i = 0..15, j = 0..3,$$

де  $y[i]$  та  $x[i]$  - відповідно, вхідне та вихідне значення і-го 4-бітного фрагмента, а функція  $p4()$  визначається згідно таблиці

<b>Вхідне значення</b>	0	1	2	3
<b>Вихідне значення</b>	1	0	3	2

9. Повторно застосовується описана вище операція P64.

Алгоритм складається із 15 раундів перетворень. Перед першим раундом виконується попереднє накладення ключа, а після заключного раунду - фінальне перетворення, що складається з наступних операцій (описаних вище):

- табличної заміни  $S_0$ ;
- бітової перестановки P8;
- табличні заміни  $S_1$ ;
- бітової перестановки P8;
- табличної заміни  $S_0$ .

При розшифруванні виконуються абсолютно ті ж операції, але фрагменти розширеного ключа використовуються у зворотному порядку (при цьому вони формуються дещо інакше, ніж при зашифруванні - див. далі).

### Процедура розширення ключа

Процедура розширення ключа складається із декількох етапів. На першому етапі з вихідного 128-бітного ключа шифрування  $K$  наступним чином обчислюється послідовність 128-бітових проміжних ключів

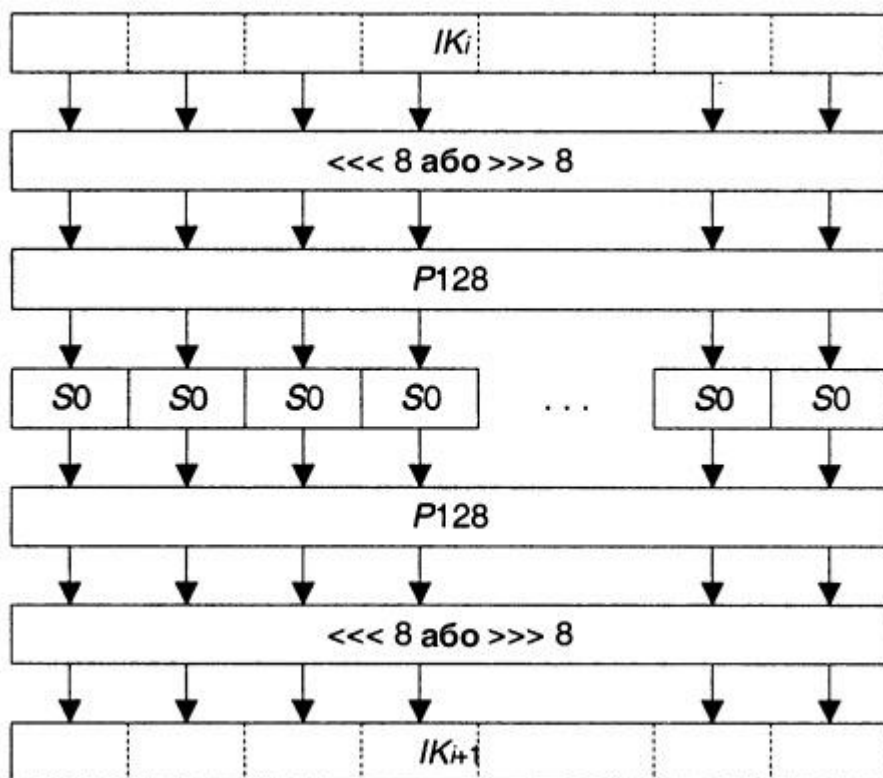
$$IK_0 \dots IK_{16} :$$

$$IK_0 = K;$$

$$IK_{i+1} = G(IK_i, C_i),$$

де  $G()$  - раунд процедури розширення ключа, а  $C_i$  - модифікуючі константи.

У кожному раунді виконується наступна послідовність дій:



1. Операція  $\tau$  здійснює циклічний зсув вхідних даних в залежності від значення  $C_i$ :

- при  $C_i = 0$  виконується обертання на 8 бітів ліворуч;
- при  $C_i = 1$  - циклічний зсув на 8 бітів праворуч.

Константи  $C_i$  визначено як 0 для першої половини раундів, тобто для  $i=0\dots7$ , для решти раундів  $C_i = 1$ .

2. Бітова перестановка  $P_{128}$  здійснює перестановку за наступним законом:

$$y_i = x_{p_{128}(i)}, i = 0\dots127,$$

де функція  $p_{128}()$  визначена згідно таблиці:

76	110	83	127	67	114	92	97	98	65	121	106	78	112	91	82
71	101	89	126	72	107	81	118	90	124	73	88	64	104	100	85
109	87	75	113	120	66	103	115	122	108	95	69	74	116	80	102
84	96	125	68	93	105	119	79	123	86	70	117	111	77	99	94
28	9	37	4	51	43	58	16	20	26	44	34	0	61	12	55
46	22	15	2	48	31	57	33	27	18	24	14	6	52	63	42
49	7	8	62	30	17	47	38	29	53	11	21	41	32	1	60
13	35	5	39	45	59	23	54	36	10	40	56	25	50	19	3

Вхідні значення 0, 1, 2, ... відповідають вихідні значення 76, 110, 83 і т.д.

- До 4-бітних фрагментів оброблюваних даних застосовується описана вище таблична заміна  $S_0$ .
- Повторно застосовується перестановка P128, після якої повторно виконується операція  $\tau$ .

На наступному етапі формується 64-бітові підключі  $IK'_0 \dots IK'_{16}$ , які просто “набираються” зі значень непарних бітів відповідних проміжних ключів  $IK_0 \dots IK_{16}$ .

На заключному етапі процедури розширення ключа обчислюється раундові ключі RK[n]. Для цього кожен 4-бітний фрагмент кожного підключа  $IK'_0 \dots IK'_{16}$  паралельно обробляється операцією  $\phi$ , яка визначена наступним чином:

$$y_0 = ((x_0 \oplus x_1 \oplus x_2) \& b) | ((x_0 \oplus x_1) \& (\sim b));$$

$$y_1 = ((x_1 \oplus x_2) \& b) | (x_1 \& (\sim b));$$

$$y_2 = ((x_2 \oplus x_3 \oplus x_0) \& b) | ((x_2 \oplus x_3) \& (\sim b));$$

$$y_3 = ((x_3 \oplus x_0) \& b) | (x_3 \& (\sim b)),$$

де:

- $y_i$  та  $x_i$  -  $i$ -ті біти, відповідно, вхідного і вихідного значення;
- $b$  - модифікуючий біт, його призначення буде пояснено далі;
- $\sim$  - логічна операція заперечення;
- $\&$  - логічна операція “і”;
- $|$  - логічна операція “або”.

Модифікуючий біт  $b$  керує формуванням різних значень підключів для зашифрування та розшифрування:

- при зашифруванні для операції попереднього накладання ключа і 15 раундів перетворень відповідні підключі ( $RK[0]$  і  $RK[1] \dots RK[15]$ ) формуються зі значенням  $b = 1$ ; підключ  $RK[16]$ , що використовується у фінальному перетворенні, формується зі значенням  $b = 0$ ;
- при розшифруванні для попереднього накладання ключа і 15 раундів перетворень використовуються, відповідно, фрагменти  $RK[16]$  і  $RK[15] \dots RK[1]$ , які формуються з модифікуючим бітом  $b=0$ ; підключ  $RK[0]$ , що використовується у фінальному перетворенні, формується зі значенням  $b = 1$ .

## Криптоаналіз алгоритму

Алгоритм ICEBERG з'явився відносно нещодавно і, мабуть, не викликав широкого інтересу з боку криптологів - будь-які роботи, присвячені криптоаналізу даного алгоритму, не набули широкої популярності.