LOKI 91 и 97 .

***LOKI* –91 .** Криптоалгоритм $LOKI$-91[1] шифрует 64-битовые блоки открытых данных под управлением секретного ключа такого же размера.

Блок B и ключ K, разделенные на 32-битовые половины L, R и KL, KR, проходят 16 раундов криптографического преобразования по схеме Фейстеля . Схема первых двух раундов представлена на рис. 1.

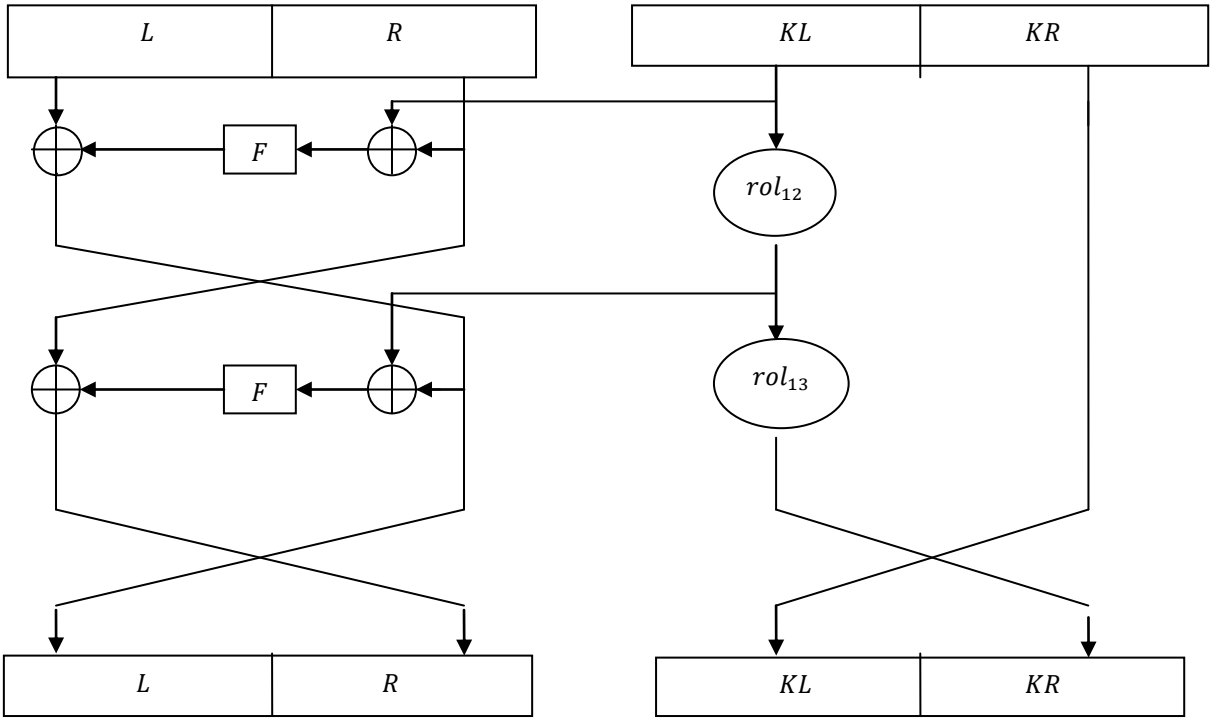

Рис. 1. Цикл (два раунда) в $LOKI$-91

Раундовая функция $F(X) = P(S(E(X)))$, от 32-битового аргумента X, возвращающая 3-битовое значение Y, является композицией следующих преобразований E, S и P.

E-преобразование. Блок $X = x_{31}x_{30} \ldots x_0$ (x_{31} – старший бит) расширяется до 48 битов, образуя четыре 12-битовых подблока X_0, X_1, X_2 и X_3 в соответствии с табл. 1.

Таблица 1

Перестановка с расширением в LOKI-91

3	2	1	0	31	30	29	28	27	26	25	24
27	26	25	24	23	22	21	20	19	18	17	16
19	18	17	16	15	14	13	12	11	10	9	8
11	10	9	8	7	6	5	4	3	2	1	0

Подблок X_3 образуют биты x_3, x_2, x_1, x_0, x_{31},…, x_{24} блока X и т.д.

S-преобразование. Каждый из подблоков X_0, X_1, X_2, X_3 преобразуется в байты b_3, b_2, b_1, b_0 по правилу: два старших и два младших бита подблока X_i задают число $c_i \in \{0,1,\ldots,15\}$, а восемь внутренних битов – число (байт) $b_i' \in \{0,1,\ldots,255\}\}$, $i = 0,1,2,3$. Отметим попутно, что значения c_i и $b_i' \in \{$ можно вычислить, не прибегая к использованию - преобразования по формулам:

$$c_3 := integer((X \,\&\, 0x0000000c) \vee shr_{24}(X \,\&\, 0x03000000)),$$
$$c_2 := integer(shr_{24}(X \,\&\, 0x0C00000) \vee shr_{16}(X \,\&\, 0x00030000)),$$
$$c_1 := integer(shr_{16}(X \,\&\, 0x000C0000) \vee shr_{8}(X \,\&\, 0x00000300)),$$
$$c_0 := integer(shr_{8}(X \,\&\, 0x0000000C) \vee (X \,\&\, 0x00000003));$$

[1] Авторы шифра: *L.Brown, M.Kwan, J.Pieprzyk* и *J.Seberry* (Австралия)

$b_3^{'} := byte(\ (shr_{24}(X) \vee shl_{30}(X)) \& \ 0x00000FFF),$
$b_2^{'} := byte\ (shr_{18}(X)\ \&\ 0\text{x}00000FFF),$
$b_1^{'} := byte\ (shr_{10}(X)\ \&\ 0\text{x}00000FFF),$
$b_0^{'} := byte\ (shr_2(X)\ \&\ 0\text{x}00000FFF),$

где $shr_n X$ обозначает логический сдвиг 4-байтового блока X на n битов вправо (в сторону младших битов, с вытеснением n битов, освободившиеся n старших позиций автоматически заполняются битовыми нулями). Байты b_i вычисляются теперь по правилу:

$$b_i := (b_i')^{31},\ i = 0,1,2,3.$$

причем, в данном случае байты интерпретируются как элементы конечного поля $\mathbb{F}_{256} \cong \mathbb{F}_2[x]\ /\ f_{c_i}(x)$, а коэффициенты неприводимых многочленов 8-ой степени $f_0(x)$,…, $f_{15}(x)$ определяющих конкретную реализацию конечного поля, задаются табл. 2 (набор коэффициентов многочлена $f_c(x))= x^8 + g_7x^7 + g_6x^6 \ldots + g_0$ представлен в 16-ичной записи; например, $f_0(x) = 9f$ следует понимать как $x^8 + x^7 + x^4 + x^3 + x^2 + x + 1$, поскольку $(9F)_{16} = (g_7g_6g_5g_4g_3g_2g_1g_0)_2 = (10011111)_2$.

Таблица 2

Многочлены, определяющие реализации конечного поля $\mathbb{F}_{256}$ в $LOKI-91$

c	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$f_c(x))$	77	7D	87	8B	8D	9F	A3	A9	B1	BD	C3	CF	D7	DD	E7	F3

P-преобразование. 32-битовый блок $Y = b_3b_2b_1b_0$0 (b_3 – старший байт), полученный в результате S-преобразования, подвергается операции перестановки битов соответствии с табл. 3: в позицию i перемещается бит из позиции $8i\ mod\ 31$, $i \neq 0, 31$ (биты 0 и 31 остаются на месте).

Таблица 3

Перестановка P в LOKI

31	23	15	7	30	22	14	6	29	21	13	5	28	20	12	4
27	19	11	3	26	18	10	2	25	17	9	1	24	16	8	0

В алгоритме зашифрования используются 32-битовые раундовые подключи ke_0, ke_1,…, ke_{15}, генерируемые на основе секретного ключа K по правилу:

$for\ i := 0\ to\ 6\ do\ \{$
    $ke_{2i} \coloneqq KL;$
    $KL := rol_{12}(KL);$
    $ke_{2i+1} \coloneqq KL;$
    $KL := rol_{13}(KL);$
    $KL \leftrightarrow KR$
$\};$
$ke_{14} := KL;$
$ke_{15} := rol_{12}(KL).$

Алгоритм зашифрования $LOKI-91$

Вход: $\boldsymbol{B}$ – **64**-битовый блок открытых данных.

$for\ i := 0\ to\ 7\ do\ \{$
    $L := L \oplus F(R \oplus ke_{2i});$
    $R \coloneqq R \oplus F(R \oplus ke_{2i+1})$
$\};$
$L \leftrightarrow R.$

Выход: B – 64-битовый блок шифртекста.

Алгоритм симметричен, т.е. может быть использован и для расшифрования, но раундовые подключи в этом случае используются в обратном порядке: ke_{15}, ke_{14},…, ke_0. □

***LOKI*–97.** Криптоалгоритм *LOKI*-97 [2] шифрует 128-битовые блоки открытых данных под управлением 256-битового секретного ключа (более короткие – 128- и 192-битовые ключи – расширяются до длины 256 специальной процедурой).

Схема зашифрования, состоящая из 16 раундов криптографического преобразования представлена на рис. 2.

Раундовая нелинейная функция $f(A,B)$ от 64-битовых аргументов A и B, возвращающая 64-битовое значение (блок) C, является суперпозицией функции KP, E, S_a и S_b (далее $X.L$ и $X.R$ обозначают соответственно левую (старшую) и правую (младшую) половины 64-битового блока X):

$$\mathrm{C} = f(A,B) = S_b(P(S_a(E(KP(A,B.R)))),B.L).$$

Функции KP, E, S_a и S_b определяются следующим образом.

Функция (перестановка) $KP(\mathrm{A},\mathrm{B})$ возвращает 64-битовое значение C:

$$C.L := (A.L\&\neg B.R)\vee(A.R\&B.R);$$
$$C.R := (A.R\&\neg B.R)\vee(A.L\&B.R).$$

Функция (перестановка с расширением) $E(A)$ возвращает 96-битовое значение D, формируемое следующим образом: если $A = a_{63}a_{62}\ldots a_0$ и $D = d_{95}d_{94}\ldots d_0$ (a_{63} и d_{95} – старшие биты), то $d_{95} := a_4,\ d_{94} := a_3,\ldots$ и т.д. согласно табл. 4.

Таблица 4

Перестановка с расширением P в LOKI 97

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

Функция $S_a(D)$ от 96-битового аргумента D возвращает 64-битовое значение, формируемое следующим образом. Представим блок D в виде конкатенации 13- и 11-битовых подблоков $D_7^{(13)}$, $D_6^{(11)}$, $D_5^{(13)}$, $D_4^{(11)}$, $D_3^{(13)}$, $D_2^{(11)}$, $D_1^{(13)}$, $D_0^{(11)}$ (верхний индекс обозначает разрядность подблока; $D_7^{(13)}$ образован 13 старшими битами блока D, $D_6^{(11)}$ следующими 11 битами и т.д.). Преобразуем подблоки $D_i^{(13)}$ с нечетными индексами по правилу:

$$D_i^{(13)} := \left(D_i^{(13)}\right)^3.$$

Интерпретируя 13-битовые подблоки как элементы конечного поля $\mathbb{F}_{2^{13}} \cong \mathbb{F}_2[x]/p_{13}(x)$, где $p_{13}(x) = x^{13} + x^{11} + x^8 + x^4 + 1$. Аналогично, подблоки $D_i^{(11)}$ с четным нижним индексом преобразуем по тому же правилу, интерпретируя 11-битовые подблоки как элементы конечного поля $\mathbb{F}_{2^{11}} \cong \mathbb{F}_2[x]/p_{11}(x)$, где $p_{11}(x) = x^{11} + x^9 + x^7 + x^5 + x^2 + x + 1$. Далее блоки $D_i^{(j)}$ превращается в байты b_i по правилу:

$$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 7\ \boldsymbol{do}\ b_i := byte(D_i^{(j)}\&0xFF)$$

(т.е. в каждом подблоке берутся 8 младших битов). Результатом функции $S_a(D)$ является 64-битовой блок, составленный из байтов b_7, b_6,…, b_0 (b_7 – старший байт).

Таблица 5

Перестановка P в LOKI- 97

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

[2] Авторы шифра: *Lawrie Brown* и *Josef Pieprzyk* (Австралия)

Функция $P(X)$ от 64-битового аргумента X возвращает 64-битовое значение, являющееся перестановкой битов блока X согласно табл. 5 (бит 63 перемещается в позицию 56, бит 62 – в позицию 48 и т.д.).

Функция $S_b(X,Y)$ от 64-битовых аргумента $X = x_{63}x_{62}\dots x_0$, и 32-битового аргумента $Y = y_{63}y_{62}\dots y_0$ возвращает 64-битовое значение Z, формируемое следующим образом. Определим 11- и 13-битовые блоки $D_i^{(j)}$, полагая

$$G_7^{(11)} := (31-29||63-56)^3; \quad G_6^{(11)} := (28-26||55-48)^3;$$
$$G_5^{(13)} := (25-21||47-40)^3; \quad G_4^{(13)} := (20-16||39-32)^3;$$
$$G_3^{(11)} := (15-13||31-24)^3; \quad G_2^{(11)} := (12-10||23-16)^3;$$
$$G_1^{(13)} := (9-5||15-8)^3; \quad G_0^{(13)} := (4-0||7-0)^3,$$

где сокращенная запись $(31-29\ ||\ 63-56)$ обозначает 11-битовый подблок, формируемый из битов $y_{31}, y_{30}, y_{29}, x_{63}, x_{62}, x_{61}, x_{60}, x_{59}, x_{58}, x_{57}, x_{56}$, а возведение в кубическую степень осуществляется для 11- и 13-битовых подблоков соответственно в конечных полях $\mathbb{F}_{2^{11}}$ и $\mathbb{F}_{2^{13}}$. Значение $Z = S_b(X,Y)$ формируется из байтов b_i по правилу:

$$\textbf{for}\ i := 7\ \textbf{downto}\ 0\ \textbf{do}\ b_i := byte(D_i^{(j)}\&0xFF),$$

(b_7 – старший байт).

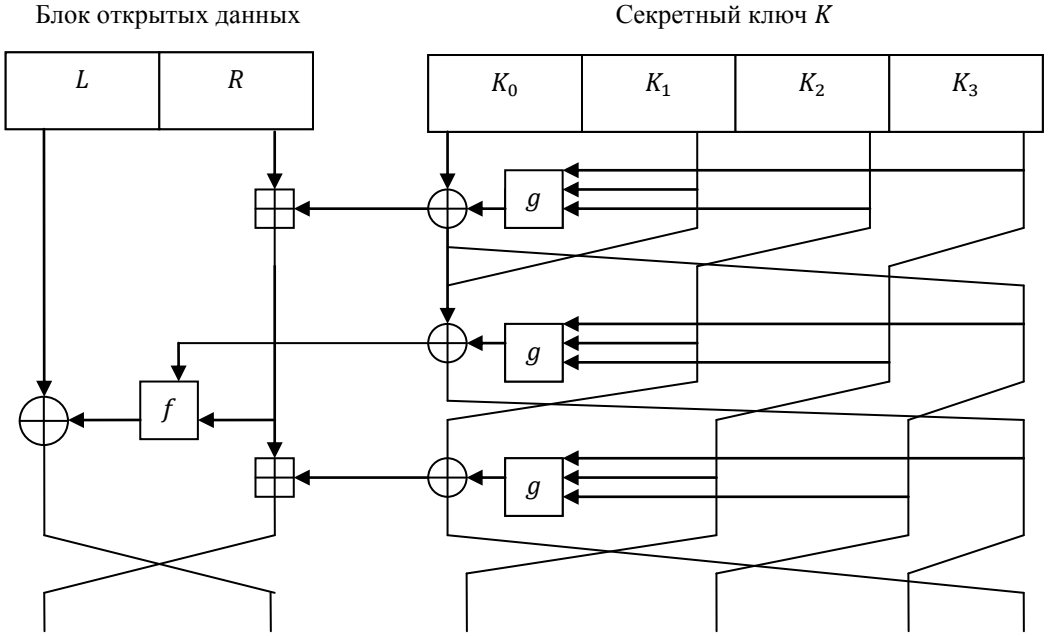

Рис. 2. *Схема зашифрования LOKI-97*

Генерация раундовых подключей

256-битовый секретный ключ K представляется в виде массива (K_3, K_2, K_1, K_0) из четырех 64-битовых подблоков. Для 192-битового ключа $K = (K_3, K_2, K_1)$ полагаем $K_0 = f(K_3, K_2)$, а для 128-битового ключа $K = (K_3, K_2)$ полагаем $K_1 = f(K_2, K_3)$, $K_0 = f(K_3, K_2)$. В алгоритме зашифровании используются 48 64-битовых раундовых подключей ke_i, $0 \le i \le 47$, которые генерируются по правилу:

$\textbf{for}\ i := 0\ \textbf{to}\ 47\ \textbf{do}\ \{$
$\quad ke_i := K_3 \oplus g_i(K_0, K_2, K_1);$
$\quad K_3 := K_2;$
$\quad K_2 := K_1;$
$\quad K_1 := K_0;$
$\quad K_0 := ke_i$
$\}.$

Функция g_i определена как

$$g_i(A, B, C) = f(A \boxplus B \boxplus (\delta \boxdot (i+1)), C),$$

где $\delta = 0\text{x}9e3779b97f4a7c15$ – 16-ичное представления целой части $(\sqrt{5}-1)2^{63}$, символы $\boxplus$ и $\boxdot$ обозначают соответственно сложение и умножение по модулю 2^{64}.

Алгоритм зашифрования *LOKI 97*

Вход: $P = L||R$ – 128-битовый блок открытых данных, представленный в виде конкатенации 64-битовых подблоков L и R.

for $i := 0$ ***to*** 7 ***do*** {
$R := R \boxplus ke_{6i}$;
$L := L \oplus f(R, ke_{6i+1})$;
$R := R \boxplus ke_{6i+2}$;
$L := L \boxplus ke_{6i+3}$;
$R := R \oplus f(L, ke_{6i+4})$;
$L := L \boxplus ke_{6i+5}$
};
$L \leftrightarrow R$.

Выход: $C = R||L$ – 128-битовый блок шифртекста.

Алгоритм расшифрования аналогичен алгоритму зашифрования: различие лишь в том, что операция $\boxplus$ заменяется на $\boxminus$ (вычитание по модулю 2^{64}), а последовательность раундовых подключей используется в обратном порядке.