MARS

Криптоалгоритм $Mars$[1] шифрует 128-битовые блоки открытых данных под управлением секретного ключа, длина которого может составлять от 128 до 448 битов (с шагом 32).

Структура алгоритма зашифрования представлена на рис.1. Криптографическое преобразование выполняется над 128-битовым блоком данных $X = (x_0, x_1, x_2, x_3)$, представленным в виде массива из четырех 32-битовых слов, под управлением 40 32-битовых раундовых подключей $k_0, k_1, \dots, k_{39}$, формируемых на этапе предвычислений на основе секретного ключа.

В алгоритме используются операции над 32-битовыми словами (с $little\text{-}endian$-порядком байтов, т.е. младший байт расположен слева, занимая младшую адресную позицию) и следующие преобразования над 128-битовым блоком $X = (x_0, x_1, x_2, x_3)$:

Преобразование $AddKey[Q]$, выполняемое под управлением 128-битового ключа $Q = (q_0, q_1, q_2, q_3)$, где q_i – 32-битовые слова, определяется как

$AddKey[Q](X) \equiv \{ (x_0, x_1, x_2, x_3) := (x_0 \boxplus_{32} q_0, x_1 \boxplus_{32} q_1, x_2 \boxplus_{32} q_2, x_3 \boxplus_{32} q_3)\}$;

обратное преобразование имеет вид:

$SubKey[Q](X) \equiv \{ (x_0, x_1, x_2, x_3) := (x_0 \boxminus_{32} q_0, x_1 \boxminus_{32} q_1, x_2 \boxminus_{32} q_2, x_3 \boxminus_{32} q_3)\}$,

где $x \boxplus_{32} q$ и $x \boxminus_{32} q$ – соответственно сложение и вычитание 32-битовых слов (неотрицательных чисел) x и q по модулю 2^{32}.

В преобразованиях $F[S]$ и $B[S]$ (см. рис.2) используются две таблицы: $S_0[0..255]$ и $S_1[0..255]$, каждая из которых состоит из 256 32-битовых слов (таблицы объединены в одну таблицу $S[0..512]$, так что $S_0[m] = S[m]$ и $S_1[m] = S[m + 256]$ для $0 \le m \le 255$):

$F[S](X) \equiv \{$

$(a, b, c, d) := (x_0 \& 255, ror_8(x_0) \& 255, ror_{16}(x_0) \& 255, ror_{24}(x_0) \& 255)$;

(Здесь a, b, c, d – байты слова x_0, начиная с младшего)

$X := (ror_{24}(x_0), (x_1 \oplus_{32} S_0[a]) \boxplus_{32} S_1[b], x_2 \boxplus_{32} S_0[c], x_3 \oplus_{32} S_1[d])$

$\}$.

$B[S](X) \equiv \{$

$(a, b, c, d) := (x_0 \& 255, rol_8(x_0) \& 255, rol_{16}(x_0) \& 255, rol_{24}(x_0) \& 255)$;

(Здесь a, b, c, d – также байты слова x_0, начиная с младшего)

$X := (rol_{24}(x_0), x_1 \oplus_{32} S_1[a], x_2 \boxminus_{32} S_0[b], (x_3 \boxminus_{32} S_1[c]) \oplus_{32} S_0[d])$

$\}$.

Преобразования $F^{-1}[S]$ и $B^{-1}[S]$, обратные к $F[S]$ и $B[S]$, можно представить в виде:

$F^{-1}[S](X) \equiv \{$

$x_1 \leftrightarrow x_3;\ B[S](X);\ x_1 \leftrightarrow x_3$

$\}$,

$B^{-1}[S](X) \equiv \{$

$x_1 \leftrightarrow x_3;\ \ F[S](X);\ x_1 \leftrightarrow x_3$

$\}$.

$FM[S](X) \equiv \{$

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 7\ \boldsymbol{do}\ \{$

$F[S](X)$;

$\boldsymbol{if}\ i = 0 \vee i = 4\ \boldsymbol{then}\ x_0 := x_0 \boxplus_{32} x_3$;

$\boldsymbol{if}\ i = 1 \vee i = 5\ \boldsymbol{then}\ x_0 := x_0 \boxplus_{32} x_1$;

$X := (x_1, x_2, x_3, x_0)$

$\}$.

$\}$;

---

[1] Авторы шифра: *Carolynn Burwick*, *Don Coppersmith*, *Edward d'Avignon*, *Rosario Gennaro*, *Shai Halevi*, *Charanjit Jutla*, *Stephen M. Matyas Jr.*, *Luke O'Connor*, *Mohammad Peyravian*, *David Safford* и *Nevenko Zunic* (США, *IBM* корпорация)

Алгоритм зашифрования *MARS*

Вход: $X = (x_0, x_1, x_2, x_3)$ – блок открытых данных – массив из 4 32-битовых слов

$Key\ Addition$:
$AddKey[k_0, k_1, k_2, k_3](X)$;

$Forward\ Mixing$:
$FM[S](X)$;

$Forward\ Keyed\ Transformation$:
$FKT[k_4, k_5, \dots, k_{19}](X)$;

$Back\ Keyed\ Transformation$:
$BKT[k_{20}, k_{21}, \dots, k_{35}](X)$;

$Back\ Mixing$:
$BM[S](X)$;

$Key\ Subtraction$:
$SubKey[k_{36}, k_{37}, k_{38}, k_{39}](X)$;

Выход: X – 128-битовый блок шифртекста

Рис. 1. *Алгоритм зашифрования MARS*

Преобразования $FM[S]$ и $BM[S]$ (см. рис.3) определяются как

$BM[S](X) \equiv \{$
    for $i := 1$ ***to*** 7 ***do*** $\{$
        if $i = 2 \vee i = 6$ ***then*** $x_0 := x_0 \boxplus_{32} x_3$;
        if $i = 3 \vee i = 7$ ***then*** $x_0 := x_0 \boxplus_{32} x_1$;
        $B[S](X)$;
        $X := (x_1, x_2, x_3, x_0)$
    $\}$
$\}$.

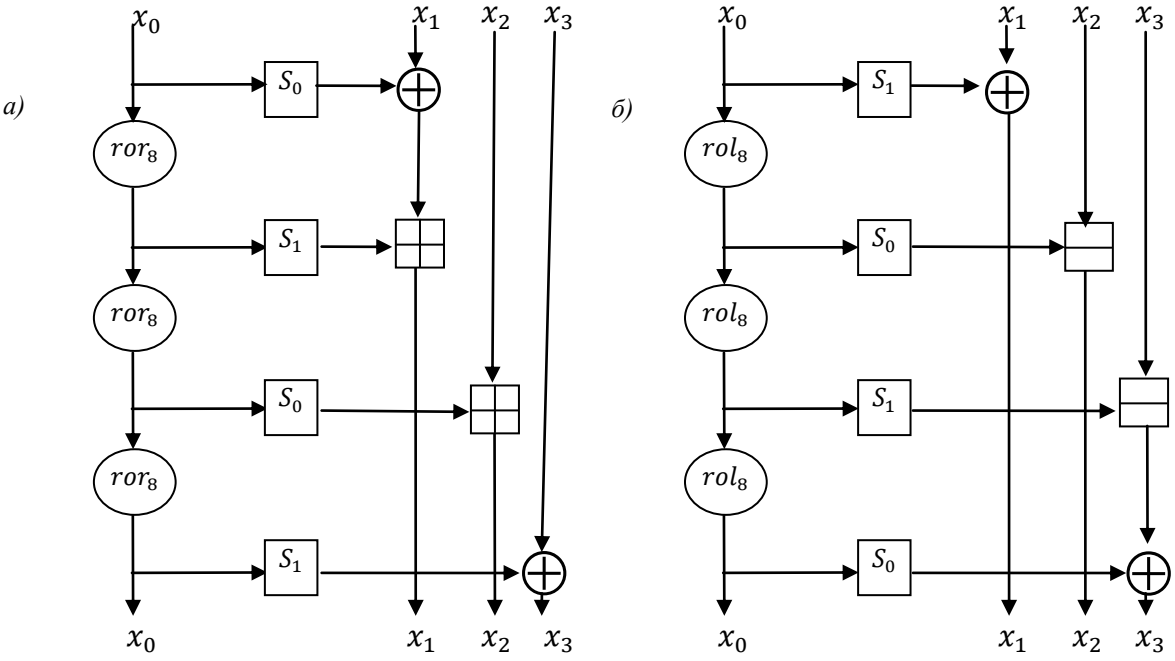


Рис. 2. *Преобразования а*) $F[S](X)$ *и б*) $B[S](X)$

Следующие преобразования являются обратными по отношению к $FM[S]$ и $BM[S]$:

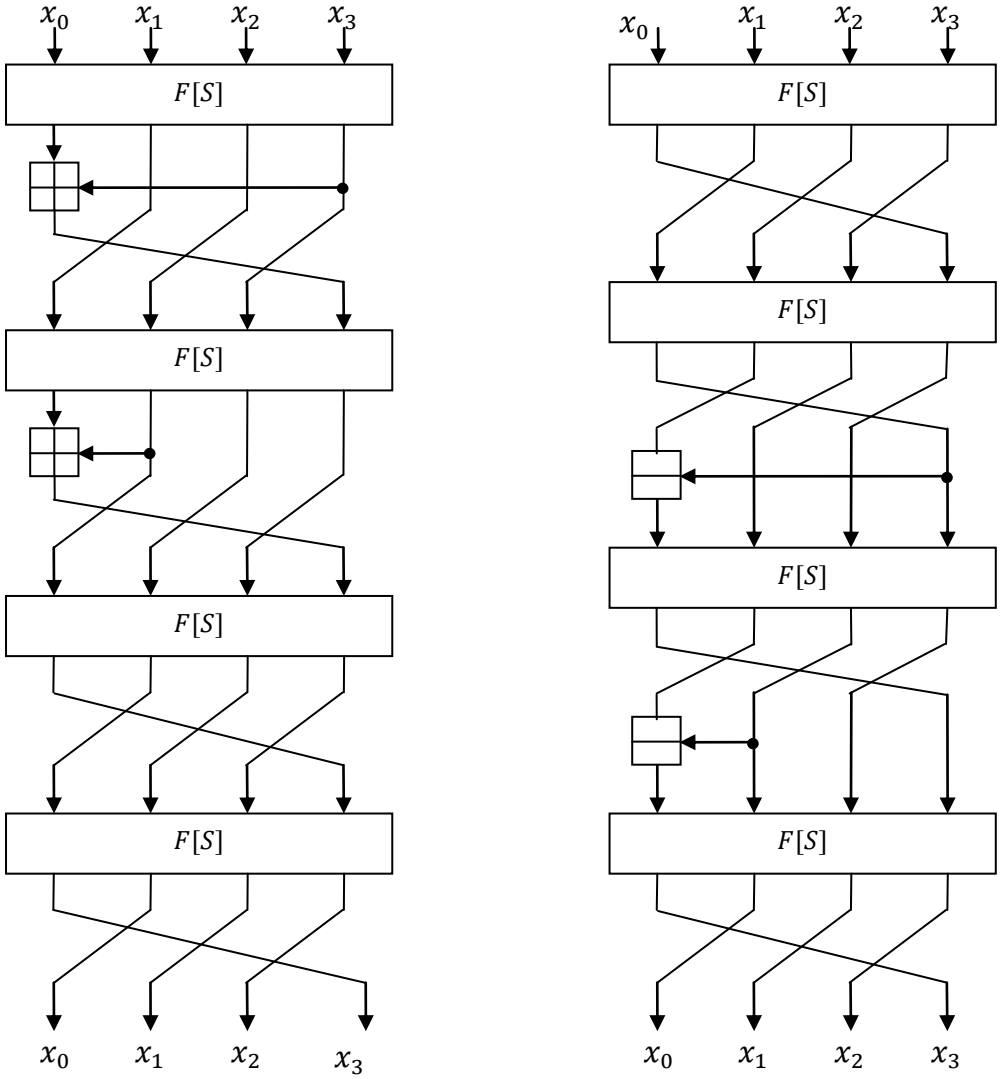
$FM^{-1}[S](X) \equiv \{$
for $i := 7$ **downto** 0 *do* $\{$
$X := (x_3, x_0, x_1, x_2);$
if $i = 1 \vee i = 5$ **then** $x_0 := x_0 \boxminus_{32} x_1;$
if $i = 0 \vee i = 4$ **then** $x_0 := x_0 \boxminus_{32} x_3;$
$F^{-1}[S](X)$
$\}$
$\}.$

$BM^{-1}[S](X) \equiv \{$
for $i := 7$ **downto** 0 **do** $\{$
$X := (x_3, x_0, x_1, x_2);\ B^{-1}[S](X);$
if $i = 3 \vee i = 7$ **then** $x_0 := x_0 \boxplus_{32} x_1;$
if $i = 2 \vee i = 6$ **then** $x_0 := x_0 \boxplus_{32} x_3$
$\}$
$\}.$

Функция $E[k, q](x_0)$ с 32-битовыми параметрами (раундовыми подключами) k, q и 32-битовым аргументом x_0, возвращающая тройку 32-битовых слов: (L, M, R), которая определена на рис. 4.

Преобразования $FKT[q_4, q_5, \dots, q_{19}]$ и $BKT[q_{20}, q_{21}, \dots, q_{35}]$ с параметрами (раундовыми подключами) $q_4, q_5, \dots, q_{19}$ и $q_{20}, q_{21}, \dots, q_{35}$ определяются как (см. рис. 5).

$FKT[q_4, q_5, \dots, q_{19}](X) \equiv \{$
for $i := 0$ **to** 7 **do** $\{$
$(L, M, R) := E[q_{2i+4}, q_{2i+5}](x_0);$
$X := (rol_{13}(x_0), x_1 \boxplus_{32} L, x_2 \boxplus_{32} M, x_3 \oplus_{32} R);$
$X := (x_1, x_2, x_3, x_0)$
$\}$
$\}.$



а) *б*)

Рис. 3. *Преобразования а*) $FM[S](X)$ *и б*) $BM[S](X)$

$BKT[q20, q21, \dots, q35](X) \equiv \{$
for $i := 8$ **to** 15 **do** $\{$
$(L, M, R) := E[q_{2i+4}, q_{2i+5}](x_0);$
$X := (rol_{13}(x_0), x_1 \oplus_{32} R, x_2 \boxplus_{32} M, x_3 \boxplus_{32} L);$
$X := (x_1, x_2, x_3, x_0)$
$\}$
$\}$.

Следующие преобразования являются обратными по отношению к преобразованиям FKT и BKT (с теми же ключевыми параметрами):

$FKT^{-1}[q_{20}, q_{21}, \dots, q_{35}](X) \equiv \{$
for $i := 7$ **downto** 0 **do** $\{$
$X := (ror_{13}(x_3), x_0, x_1, x_2);$
$(L, M, R) := E[q_{2i+4}, q_{2i+5}](x_0);$
$X := (x_0, x_1 \boxminus_{32} L, x_2 \boxminus_{32} \boxminus_{32} M, x_3 \oplus R)$
$\}$
$\};$

$BKT^{-1}[q_{20}, q_{21}, \dots, q_{35}](X) \equiv \{$
for $i := 15$ **downto** 8 **do** $\{$

$X := (ror_{13}(x_3), x_0, x_1, x_2)$;
$(L, M, R) := E[q_{2i+4}, q_{2i+5}](x_0)$;
$X := (x_0, , x_1 \oplus R, x_2 \boxminus_{32} M, x_3 \boxminus_{32} L)$
}
}.

Алгоритм расшифрования *MARS*

Вход: $C = (c_0, c_1, c_2, c_3)$ – 128-битовый блок шифртекста.
$AddKey[k_{36}, k_{37}, k_{38}, k_{39}](C)$;
$BM^{-1}[S](C)$;
$BKT^{-1}[k_{20}, k_{21}, \dots, k_{35}](C)$;
$FKT^{-1}[k_4, k_5, \dots, k_{19}](C)$;
$FM^{-1}[S](C)$;
$SubKey[k_0, k_1, k_2, k_3](C)$.
Выход: C – 128-битовый блок открытых данных.

Генерация раундовых подключей

На этапе предвычислений на основе секретного ключа KS, имеющего длину $32n$ битов $(4 \le n \le 14)$, формируются раундовые подключи: $k_0, k_1, k_2, \dots, k_{39}$.

Пусть $t_0, t_1, \dots, t_{14}$, W, m, p – вспомогательные 32-битовые переменные;

$B_0 = 0\text{x}A4A8D57B$,
$B_1 = 0\text{x}5B5D193B$,
$B_2 = 0\text{x}C8A8309B$,
$B_3 = 0\text{x}73F9A978$.

1. (Инициализация массива t.)
$t[0..n-1] := KS;\ t[n] := n;\ t[n+1,..,14] := 0;$
2. (Четыре итерации, в каждой из которых вычисляются по 10 раундовых подключей.)
for $j := 0$ **to** 3 **do** {
for $i := 0$ **to** 14 **do** {
$q := (i + 8) mod\ 15$;
$r := (i + 13)\ mod\ 15$;
$t_i := rol_3(t_q \oplus t_r) \oplus word_4(4 * i + j)$
};
//NB. $word_4(x)$ – 4-битовое слово со значением x.
for $l := 0$ **to** 3 **do**{
for $i := 0$ **to** 14 **do** {
$r := (i + 14) mod\ 15$;
$q := t_r\ mod\ 512$;
$t_i := rol_9(t_i \oplus S[q])$
}
};
for $i := 0$ **to** 9 **do** {
$r := i + 10 * j$;
$q := (4 * i)$ **mod** 15;
$k_r := t_q$
}
};
3. (Модификация раундовых подключей, используемых для умножения.)
for $i := 0$ **to** 15 **do** {
3.1. (Выделение двух младших битов в ключе k_{2i+5}.)

$j := k_{2i+5} \; mod \; 4$ (или $j := k_{2i+5}$ & 3);

3.2. (Два младших бита в k_{2i+1} полагаются равными 1.)

$W := k_{2i} + 5 \vee 3$;

3.3. (Выделение пяти младших битов в k_{2i+4}.)

$r := k_{2i+4} \; mod \; 32$ (или $r := k_{2i+4}$ & 31);

3.4. (Циклический сдвиг вспомогательной константы.)

$p := rol_r(B_j)$;

3.5. (Модификация раундового ключа k_{2i+5}.)

$k_{2i+5} \coloneqq W \oplus \big(p \;\&\; MASK(W)\big)$

}.

Остается определить функцию $MASK(W)$ от 32-битового аргумента $W = w_{31}w_{30} \ldots w_0$ (w_0 – младший, а w_{31} – старший биты в W), используемую в п.3.5. $MASK$ возвращает 32-битовое значение $M = m_{31}m_{30} \ldots m_0$, где $m_0 = m_{31}$; $m_i = 1$ тогда и только тогда, когда $w_{i-1} = w_i = w_{i+1}$, а w_i принадлежит серии из 10 или более одинаковых битов. Например, $MASK(0^3 1^{13} 0^{12} 101^2) = 0^4 1^{11} 0^2 1^{10} 0^5$, где α^n обозначает серию из n битов β. Значение $M = MASK(W)$ можно вычислить следующим образом:

$M := \big((not\ W) \oplus ror_1(W)\big)$& 0x$7FFFFFFF$;

$M := M \;\&\; ror_1(M) \;\&\; ror_2(M)$;

$M := M \;\&\; ror_3(M)\&\; ror_6(M)$;

if $M \neq 0$ ***then*** {

$M := rol_1(M)$;

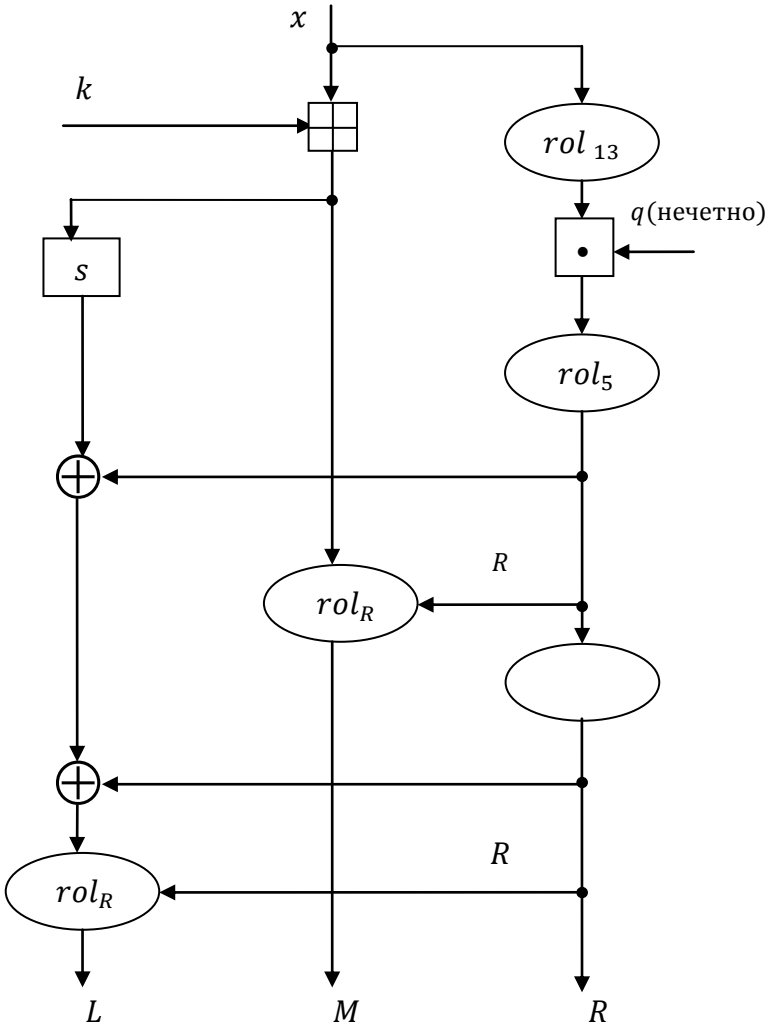
$M := M \vee rol_1(M)$;

$M := M \vee rol_2(M)$;

$M := M \vee rol_4(M)$;

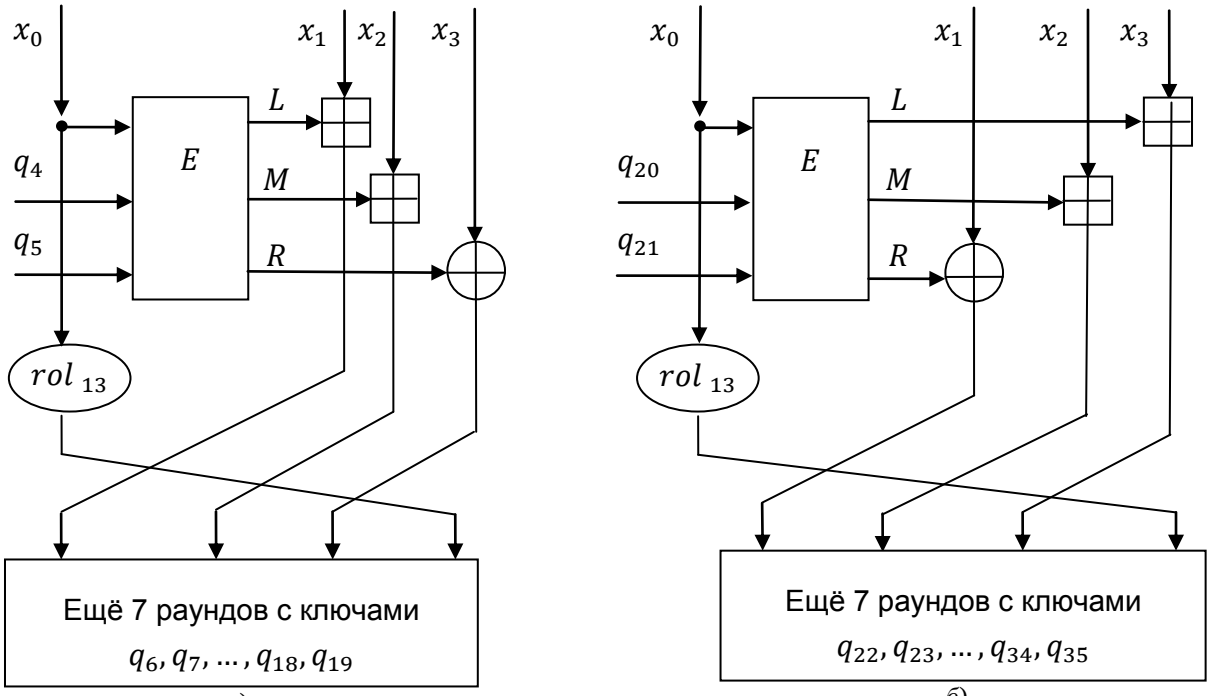
$M := M$ & 0x$FFFFFFFC$

}.



Вход: (k, q, x).
$M := x \boxplus_{32} q;$
$L := S[M \bmod 512];$
$R := rol_{13}(x) \boxdot_{32} q;$
$R := rol_5(R);$
$L := L \oplus R;$
$M := rol_R(M);$
$R := rol_5(R);$
$L := L \oplus R;$
$L := rol_R(L).$
Выход: (L, M, R).

Рис. 4. *Функция* $E[k, q](x)$



а) *б*)

Рис. 5. *Преобразования а*) *FKT и б*) *BKT*

Таблица 1

Подстановка $S[0..511]$ *в MARS*

09d0c479	28c8ffe0	84aa6c39	9dad7287	7dff9be3	d4268361	c96da1d4	7974cc93
85d0582e	2a4b5705	1ca16a62	c3bd279d	0f1f25e5	5160372f	c695c1fb	4d7ff1e4
ae5f6bf4	0d72ee46	ff23de8a	b1cf8e83	f14902e2	3e981e42	8bf53eb6	7f4bf8ac
83631f83	25970205	76afe784	3a7931d4	4f846450	5c64c3f6	210a5f18	c6986a26
28f4e826	3a60a81c	d340a664	7ea820c4	526687c5	7eddd12b	32a11d1d	9c9ef086
80f6e831	ab6f04ad	56fb9b53	8b2e095c	b68556ae	d2250b0d	294a7721	e21fb253
ae136749	e82aae86	93365104	99404a66	78a784dc	b69ba84b	04046793	23db5c1e
46cae1d6	2fe28134	5a223942	1863cd5b	c190c6e3	07dfb846	6eb88816	2d0dcc4a
a4ccae59	3798670d	cbfa9493	4f481d45	eafc8ca8	db1129d6	b0449e20	0f5407fb
6167d9a8	d1f45763	4daa96c3	3bec5958	ababa014	b6ccd201	38d6279f	02682215
8f376cd5	092c237e	bfc56593	32889d2c	854b3e95	05bb9b43	7dcd5dcd	a02e926c
fae527e5	36a1c330	3412e1ae	f257f462	3c4f1d71	30a2e809	68e5f551	9c61ba44
5ded0ab8	75ce09c8	9654f93e	698c0cca	243cb3e4	2b062b97	0f3b8d9e	00e050df
fc5d6166	e35f9288	c079550d	0591aee8	8e531e74	75fe3578	2f6d829a	f60b21ae
95e8eb8d	6699486b	901d7d9b	fd6d6e31	1090acef	e0670dd8	dab2e692	cd6d4365
e5393514	3af345f0	6241fc4d	460da3a3	7bcf3729	8bf1d1e0	14aac070	1587ed55
3afd7d3e	d2f29e01	29a9d1f6	efb10c53	cf3b870f	b414935c	664465ed	024acac7
59a744c1	1d2936a7	dc580aa6	cf574ca8	040a7a10	6cd81807	8a98be4c	accea063
c33e92b5	d1e0e03d	b322517e	2092bd13	386b2c4a	52e8dd58	58656dfb	50820371
41811896	e337ef7e	d39fb119	c97f0df6	68fea01b	a150a6e5	55258962	eb6ff41b
d7c9cd7a	a619cd9e	bcf09576	2672c073	f003fb3c	4ab7a50b	1484126a	487ba9b1
a64fc9c6	f6957d49	38b06a75	dd805fcd	63d094cf	f51c999e	1aa4d343	b8495294
ce9f8e99	bffcd770	c7c275cc	378453a7	7b21be33	397f41bd	4e94d131	92cc1f98
5915ea51	99f861b7	c9980a88	1d74fd5f	b0a495f8	614deed0	b5778eea	5941792d
fa90c1f8	33f824b4	c4965372	3ff6d550	4ca5fec0	8630e964	5b3fbbd6	7da26a48
b203231a	04297514	2d639306	2eb13149	16a45272	532459a0	8e5f4872	f966c7d9
07128dc0	0d44db62	afc8d52d	06316131	d838e7ce	1bc41d00	3a2e8c0f	ea83837e
b984737d	13ba4891	c4f8b949	a6d6acb3	a215cdce	8359838b	6bd1aa31	f579dd52
21b93f93	f5176781	187dfdde	e94aeb76	2b38fd54	431de1da	ab394825	9ad3048f
dfea32aa	659473e3	623f7863	f3346c59	ab3ab685	3346a90b	6b56443e	c6de01f8
8d421fc0	9b0ed10c	88f1a1e9	54c1f029	7dead57b	8d7ba426	4cf5178a	551a7cca
1a9a5f08	fcd651b9	25605182	e11fc6c3	b6fd9676	337b3027	b7c8eb14	9e5fd030
6b57e354	ad913cf7	7e16688d	58872a69	2c2fc7df	e389ccc6	30738df1	0824a734
e1797a8b	a4a8d57b	5b5d193b	c8a8309b	73f9a978	73398d32	0f59573e	e9df2b03
e8a5b6c8	848d0704	98df93c2	720a1dc3	684f259a	943ba848	a6370152	863b5ea3
d17b978b	6d9b58ef	0a700dd4	a73d36bf	8e6a0829	8695bc14	e35b3447	933ac568
8894b022	2f511c27	ddfbcc3c	006662b6	117c83fe	4e12b414	c2bca766	3a2fec10
f4562420	55792e2a	46f5d857	ceda25ce	c3601d3b	6c00ab46	efac9c28	b3c35047
611dfee3	257c3207	fdd58482	3b14d84f	23becb64	a075f3a3	088f8ead	07adf158
7796943c	facabf3d	c09730cd	f7679969	da44e9ed	2c854c12	35935fa3	2f057d9f
690624f8	1cb0bafd	7b0dbdc6	810f23bb	fa929a1a	6d969a17	6742979b	74ac7d05

010e65c4	86a3d963	f907b5a0	d0042bd3	158d7d03	287a8255	bba8366f	096edc33
21916a7b	77b56b86	951622f9	a6c5e650	8cea17d1	cd8c62bc	a3d63433	358a68fd
0f9b9d3c	d6aa295b	fe33384a	c000738e	cd67eb2f	e2eb6dc2	97338b02	06c9f246
419cf1ad	2b83c045	3723f18a	cb5b3089	160bead7	5d494656	35f8a74b	1e4e6c9e
000399bd	67466880	b4174831	acf423b2	ca815ab3	5a6395e7	302a67c5	8bdb446b
108f8fa4	10223eda	92b8b48b	7f38d0ee	ab2701d4	0262d415	af224a30	b3d88aba
f8b2c3af	daf7ef70	cc97d3b7	e9614b6c	2baebff4	70f687cf	386c9156	ce092ee5
01e87da6	6ce91e6a	bb7bcc84	c7922c20	9d3b71fd	060e41c6	d7590f15	4e03bb47
183c198e	63eeb240	2ddbf49a	6d5cba54	923750af	f9e14236	7838162b	59726c72
81b66760	bb2926c1	48a0ce0d	a6c0496d	ad43507b	718d496a	9df057af	44b1bde6
054356dc	de7ced35	d51a138b	62088cc9	35830311	c96efca2	686f86ec	8e77cb68
63e1d6b8	c80f9778	79c491fd	1b4c67f2	72698d7d	5e368c31	f7d95e2e	a1d3493f
dcd9433e	896f1552	4bc4ca7a	a6d1baf4	a5a96dcc	0bef8b46	a169fda7	74df40b7
4e208804	9a756607	038e87c8	20211e44	8b7ad4bf	c6403f35	1848e36d	80bdb038
1e62891c	643d2107	bf04d6f8	21092c8c	f644f389	0778404e	7b78adb8	a2c52d53
42157abe	a2253e2e	7bf3f4ae	80f594f9	953194e7	77eb92ed	b3816930	da8d9336
bf447469	f26d9483	ee6faed5	71371235	de425f73	b4e59f43	7dbe2d4e	2d37b185
49dc9a63	98c39d98	1301c9a2	389b1bbf	0c18588d	a421c1ba	7aa3865c	71e08558
3c5cfcaa	7d239ca4	0297d9dd	d7dc2830	4b37802b	7428ab54	aeee0347	4b3fbb85
692f2f08	134e578e	36d9e0bf	ae8b5fcf	edb93ecf	2b27248e	170eb1ef	7dc57fd6
1e760f16	b1136601	864e1b9b	d7ea7319	3ab871bd	cfa4d76f	e31bd782	0dbeb469
abb96061	5370f85d	ffb07e37	da30d0fb	ebc977b6	0b98b40f	3a4d0fe6	df4fc26b
159cf22a	c298d6e2	2b78ef6a	61a94ac0	ab561187	14eea0f0	df0d4164	19af70ee