Frog [1]

Криптоалгоритм $Frog$[2] шифрует n-байтовые блоки открытых данных под управлением l-байтового секретного ключа. Параметры n и l могут варьироваться в следующих диапазонах: $8 \le n \le 128$, $5 \le l \le 125$.

Все операции в алгоритме шифрования выполняются над байтами (символ $\oplus$ обозначает операцию побитового сложения байтов по модулю 2). Блок шифруемых данных P представляется в виде n-байтового массива: $P = (p_0, \dots, p_{n-1})$. Операция шифрования выполняется под управлением внутреннего ключа K (*Internal Key*), формируемого на основе -байтового секретного ключа пользователя $UK = (uk_0, \dots, uk_{l-1})$. Внутренний ключ – это массив из восьми записей: $K_0, \dots, K_7$. Каждая запись содержит три поля:

$X = (x_0, \dots, x_{n-1})$ – массив из n байтов;

$S = (s_0, \dots, s_{255})$ – массив из 256 байтов, задающий подстановку (перестановку) на множестве байтов;

$B = (b_0, \dots, b_{n-1})$ – массив из n байтов со значениями в диапазоне от 0 до $n - 1$.

Алгоритм зашифрования состоит из восьми итераций, соответствующих восьми записям K_i, $0 \le i \le 7$, внутреннего ключа K. Одна итерация зашифрования под управлением очередной записи внутреннего ключа иллюстрируются на рис.1.

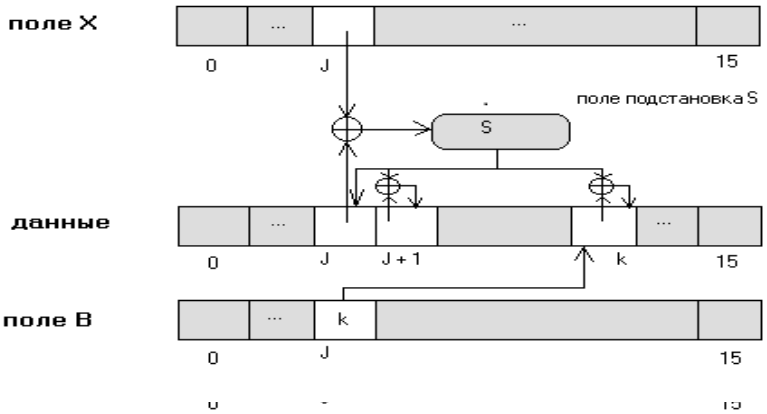

Рис. 1. *Одна итерация алгоритма* $Frog$

Алгоритм зашифрования $FrogEncrypt(P, K, C)$

Вход: $P = (p_0, \dots, p_{n-1})$ – n-байтовый блок открытых данных.

$(c_0, \dots, c_{n-1}) := (p_0, \dots, p_{n-1})$;
for $i := 0$ **to** 7 **do** {
  with ki **do** {
    for $j := 0$ **to** $n - 1$ **do** {
      $t := c_j \oplus x_j$;
      $c_j := s_t$;
      $t := (j + 1) \bmod n$;
      $c_t := c_t \oplus c_j$;
      $t := b_j$;
      $c_t := c_t \oplus c_j$
    }
  }

[1] Авторы шифра: *Dianelos Georgoudis, Damian Leroux, Billy Simon Chaves* (Коста-Рика)

[2] $Frog$ (англ.) – лягушка

}.

Выход: $C = (c_0, \dots, c_{n-1})$ – n-байтовый блок шифртекста.

При расшифровании в полях S внутреннего ключа записываются обратные подстановки S^{-1}.

Алгоритм расшифрования $FrogDecrypt(C, K, P)$

Вход: $C = (c_0, \dots, c_{n-1})$ – n-байтовый блок шифртекста.

$(p_0, \dots, p_{n-1}) := (c_0, \dots, c_{n-1})$;
for $i := 0$ **downto** 7 **do** {
  with ki **do** {
    for $j := n - 1$ **downto** 0 **do** {
      $t := b_j$;
      $p_t := p_t \oplus p_j$;
      $t := (j + 1)$ **mod** n;
      $p_t := p_t \oplus p_j$;
      $t := p_j$; $p_j := s_t \oplus x_j$
    }
  }
}.

Выход: $P = (p_0, \dots, p_{n-1})$ – n-байтовый блок открытых данных.

Генерация внутреннего ключа

Внутренний ключ K содержит в совокупности $N = 2048 + 16n$ байтов. При его формировании используются:

$W = (w_0, \dots, w_{N-1})$ – вспомогательный массив из N байтов;

$R = (r_0, \dots, r_{250})$ – массив из 251 случайных байтов, заданных табл.1. (Таблица составлена на основе первых 1255 цифр из “Миллиона случайных цифр”, опубликованных в 1955 году корпорацией $RAND$: число, образованное очередной пятеркой цифр, приведенное по модулю 256, дает очередное число таблицы.)

Таблица 1

251 *случайных байтов, используемых в* $Frog$

113	21	232	18	113	92	63	157	124	193	166	197	126	56	229	229
156	162	54	17	230	89	189	87	169	0	81	204	8	70	203	225
160	59	167	189	100	157	84	11	7	130	29	51	32	45	135	237
139	33	17	221	24	50	89	74	21	205	191	242	84	53	3	230
231	118	15	15	107	4	21	34	3	156	57	66	93	255	191	3
85	135	205	200	185	204	52	37	35	24	68	185	201	10	224	234
7	120	201	115	216	103	57	255	93	110	42	249	68	14	29	55
128	84	37	152	221	137	39	11	252	50	144	35	178	190	43	162
103	249	109	8	235	33	158	111	252	205	169	54	10	20	221	201
178	224	89	184	182	65	201	10	60	6	191	174	79	98	26	160
252	51	63	79	6	102	123	173	49	3	110	233	90	158	228	210
209	237	30	95	28	179	204	220	72	163	77	166	192	98	165	25
145	162	91	212	41	230	110	6	107	187	127	38	82	98	30	67
225	80	208	134	60	250	153	87	148	60	66	165	72	29	165	82
211	207	0	177	206	13	6	14	92	248	60	201	132	95	35	215
118	177	121	180	27	83	131	26	39	46	12					

$Buffer = (Buffer_0, \dots, Buffer_{n-1})$ – вспомогательный массив из n байтов (имеющий такую же структуру, что и блок шифруемых данных).

При формировании внутреннего ключа используется процедура $Format(W, K)$, преобразующая массив W во внутренний ключ K. Эта процедура определена ниже.

Алгоритм формирования внутреннего ключа (см. рис.2)

1. (Формирование псевдослучайного ключа W на основе секретного ключа UK и массива R).

$t := 0;$
$v := 0;$
$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ N-1\ \boldsymbol{do}\ \{$
$w_i = uk_v \oplus r_t;$
$v := (v+1)\ mod\ n;$
$t := (t+1)\ mod\ 251$
$\};$

(другими словами, массив W получается путем побитового сложения массивов UK и R, повторенных надлежащее число раз).

2. (Получение промежуточного внутреннего ключа K.)

$Format(W, K);$

3. (Формирование нового значения псевдослучайного ключа W путем хеширования промежуточного внутреннего ключа K в режиме CBC – сцепления шифрованных блоков.)

$\boldsymbol{if}\ l > n\ \boldsymbol{then}\ t := n\ \boldsymbol{else}\ t := l;$
$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ t-1\ \boldsymbol{do}\ Buffer_i := uk_i;$
$\boldsymbol{for}\ i := t\ \boldsymbol{to}\ n-1\ \boldsymbol{do}\ Buffer_i := 0;$
$Buffer_0 := Buffer_0 \oplus byte(l);$

где $byte(l)$ – байт со значением l; блок $Buffer$ – вектор инициализации в режиме шифрования CBC)

$i := 0;$
$\boldsymbol{while}\ i < n\ \boldsymbol{do}\ \{$
$FrogEncrypt(Buffer, K, Buffer);$
$\boldsymbol{for}\ j := 0\ \boldsymbol{to}\ n-1\ \boldsymbol{do}\ wi + j := Bufferj;$
$i := i + n$
$\};$

4. (Получение внутреннего ключа, используемого для шифрования данных.)

$Format(W, K)$.

Процедура $\boldsymbol{Format}(\boldsymbol{W}, \boldsymbol{K})$, преобразующая массив W во внутренний ключ K, определяется следующим образом:

1. (Массив W переписывается в K.)

$i := 0;$
$\boldsymbol{for}\ j := 0\ \boldsymbol{to}\ 7\ \boldsymbol{do}\ \{$
$\boldsymbol{with}\ K_j\ \boldsymbol{do}\ \{$
$\boldsymbol{for}\ t := 0\ \boldsymbol{to}\ n-1\ \boldsymbol{do}\ \{x_t := w_i;\ i := i+1\};$
$\boldsymbol{for}\ t := 0\ \boldsymbol{to}\ 255\ \boldsymbol{do}\ \{s_t := w_i;\ i := i+1\};;$
$\boldsymbol{for}\ t := 0\ \boldsymbol{to}\ n-1\ \boldsymbol{do}\ \{b_t := w_i;\ i := i+1\}$
}
$\};$

2. (Поля S и B в K преобразуется в подстановки (перестановки), определенные соответственно на множествах $\{0,1,\dots,255\}$ и $\{0,1,\dots,n-1\}$.)

$\boldsymbol{for}\ i := 0\ \boldsymbol{to}\ 7\ \boldsymbol{do}\ \{$
$\boldsymbol{with}\ K_i\ \boldsymbol{do}\ \{$
$\boldsymbol{for}\ j := 0\ \boldsymbol{to}\ 255\ \boldsymbol{do}\ u_j := j;$

($U = (u_0, \dots, u_{255})$ – вспомогательный 256-байтовый массив.)

$t := 0;$

$m := 256;$

$\boldsymbol{for}\ y := 0\ \boldsymbol{to}\ 255\ \boldsymbol{do}\ \{$
$t := (t + s_y)\ mod\ m;\ s_y := u_t;\ m := m - 1;$
$\boldsymbol{for}\ v := t\ \boldsymbol{to}\ m - 1\ \boldsymbol{do}\ u_v := u_v + 1$
$\};$
$\boldsymbol{for}\ j := 0\ \boldsymbol{to}\ n - 1\ \boldsymbol{do}\ u_j := j;$
$t := 0;\ m := n;$
$\boldsymbol{for}\ y := 0\ \boldsymbol{to}\ n - 1\ \boldsymbol{do}\ \{$
$t := (t + b_y)\ mod\ m;$
$b_y := u_t;$
$m := m - 1;$
$\boldsymbol{for}\ v := t\ \boldsymbol{to}\ m - 1\ \boldsymbol{do}\ u_v := u_v + 1$
$\}$
$\}$
$\};$

3. (Поля B в K подвергается дополнительному преобразованию.)

Поясним на примере. Поле

$$B = (252, 51, 63, 79, 6, 102, 123, 173, 49, 3, 110, 233, 90, 158, 228, 210)$$

на шаге 2 преобразуется в подстановку

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
b_i	12	3	11	14	6	10	1	5	8	0	7	2	13	9	15	14

Эта подстановка разлагается в произведение следующих циклов длины $4, 6, 2, 3$ и 1:

$$0 \longrightarrow 12 \longrightarrow 13 \longrightarrow 9 \rightarrow 0;\ 1 \longrightarrow 3 \longrightarrow 14 \longrightarrow 15 \longrightarrow 4 \longrightarrow 6 \longrightarrow 1;\ \ 2 \longrightarrow 11 \longrightarrow 2;$$
$$5 \longrightarrow 10 \longrightarrow 7 \longrightarrow 5;\ \ 8 \longrightarrow 8.$$

Эти циклы «склеиваются» в один цикл длины 16. Заключительное преобразование заменяет значения $b_i = (i + 1)\ mod\ n$ на $(i + 2)\ mod\ n$.)

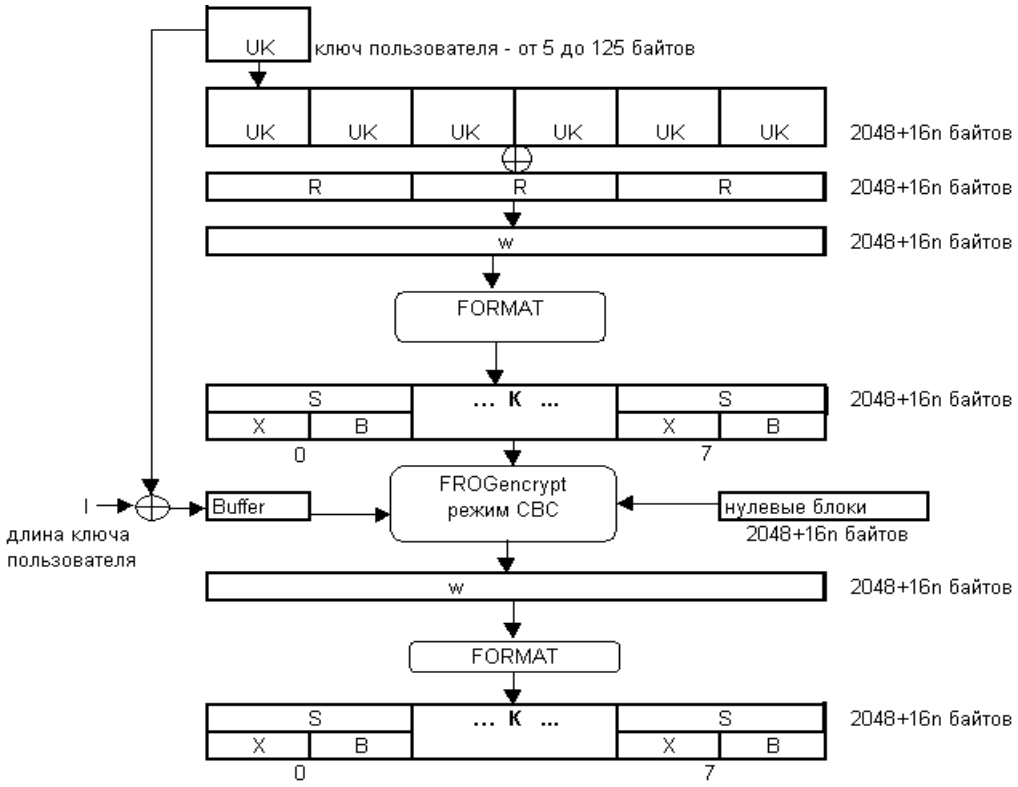

Рис. 2. *Формирование внутреннего ключа K в алгоритме* $Frog$.

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

```
    with K_i do {
        (f_0, ..., f_{n-1}) := (false, ..., false);
        j := 0;
        for t := 0 to n − 2 do {
            if b_j = 0 then {
                y := j;
                repeat y := (y + 1) mod n until not f_y;
                b_j := y;
                v := y;
                while b_v ≠ y do v := b_v;
                b_v := 0
            };
            f_j := true;
            j := b_j
        };
        for j := 0 to n − 1 do
            if b_j = (j + 1) mod n then b_j := (j + 2) mod n
    }
}.
```