УДК 681.3.06 : 519.248.681

С.А. Головашич

ООО Криптомаш, Харьков

АЛГОРИТМ БЛОЧНОГО СИММЕТРИЧНОГО ШИФРОВАНИЯ «ЛАБИРИНТ»

Статья посвящена описанию спецификации преобразования нового алгоритма блочного симметричного шифрования «Лабиринт», удовлетворяющего требованиям первого (максимального) класса стойкости, в соответствии с условиями проекта NESSIE (длина блока не менее 128 бит, длина ключа не менее 256 бит). Структура алгоритма «Лабиринт» позволяет эффективно его реализовать как на 64-битных, так и на 32-битных микропроцессорах.

криптозащита, блочное симметричное шифрование, поточное шифрование

Введение

Специфика информационных потоков, циркулирующих в современных ИТ-системах, часто требует обеспечения конфиденциальности передаваемых данных, что, проще всего, достигается путём применения средств криптографической защиты информации (КЗИ). Неотъемлемым элементом подсистемы КЗИ в компьютерных системах стали блочные симметричные шифры (БСШ). Последнее десятилетие ознаменовалось эволюционной сменой стандартов в области алгоритмов КЗИ, которая не обошла и БСШ. Так, новый международный стандарт БСШ FIPS-197 [1] (победитель проекта AES [1] – алгоритм Rijndael), сегодня фактически вытеснил, широко распространённый ещё 10 лет назад, стандарт блочного шифрования DES и его 3-х каскадную версию – Triple-DES [2].

В настоящее время на Украине отсутствует национальный стандарт БСШ, и временно разрешён к применению стандарт бывшего СССР – ГОСТ 28147-89 [3], принятый в 1989 году. Однако, учитывая значительный прогресс в сфере методов и средств криптоанализа, на протяжении последних 10 – 20 лет, этот алгоритм уже переходит в класс “морально устаревших”. Косвенным подтверждением этого факта стали результаты проведенных в США и Европе конкурсов (проекты AES [4] и NESSIE [5] соответственно), которые были направлены на выбор новых криптопримитивов, удовлетворяющих возросшим требованиям безопасности. Так, в соответствии с требованиями проекта NESSIE, алгоритм ГОСТ 28147-89 [3] относиться только к третьему (наименьшему) классу стойкости. Поэтому проблема разработки отечественного стандарта БСШ, удовлетворяющего наиболее жёстким требованиям безопасности и учитывающего последние достижения в области методов криптоанализа и криптозащиты симметричных алгоритмов, является весьма актуальной для Украины. Основной целью разработки алгоритма «Лабиринт» было решение указанной проблемы.

Предлагаемый в данной работе алгоритм «Лабиринт» является результатом эволюционного развития серии алгоритмов БСШ, разработанных автором в ходе работы над диссертацией (алгоритм – «Шторм» [6]) и последующих исследований в области проектирования БСШ (алгоритм «Торнадо» [7 – 9], версии 1-3).

1. Обозначения и соглашения

Алгоритм «Лабиринт» является итеративным «инволютивным» шифром [10] и предусматривает три варианта длины блока (для применения в различных классах безопасности) и переменную длину ключа шифрования для каждого варианта длины блока.

Структура алгоритма «Лабиринт» идентична для различных длин блока, поэтому описание алгоритма приводится в общем виде для блока длиной 128n бит, где n – *коэффициент кратности длины блока*, который может принимать значения 1, 2 или 4, т.е. алгоритм поддерживает блоки длиной 128, 256 и 512 бит. Для каждой длины блока алгоритм поддерживает 3 значения длины ключа в диапазоне от 128n до 256n с шагом 64n бита (т.е. от 2 до 4 полублоков включительно).

Префикс 0x будем использовать для обозначения шестнадцатеричных чисел. Например: 0x1AF = 431.

1.1. Список символов и обозначений

$\mathbf{Z_p}$ – кольцо целых чисел по модулю p;
$\mathbf{F_p}$ – поле порядка p (запись эквивалентна GF(p));
B – векторное пространство 8-битных элементов (байтов), $\mathbf{B} = GF(2)^8 = \{0,1\}^8$;
W – векторное пространство 64-битных элементов (слов), $\mathbf{W} = \mathbf{B}^8 = GF(2)^{64} = \{0,1\}^{64} = \mathbf{Z}_{2^{64}}$;
H – векторное пространство 64n-битных элементов (полублоков), $\mathbf{H} = \mathbf{W}^n = GF(2)^{64n} = \{0,1\}^{64n}$;
T – векторное пространство 128n-битных элементов (блоков), $\mathbf{T} = \mathbf{H}^2 = GF(2)^{128n} = \{0,1\}^{128n}$;
$\oplus$ – побитовая операция «исключающее ИЛИ» (XOR, сложение по модулю 2);
$\times$ – операция умножения матрицы на вектор (элементы принадлежат полю $GF(2^8)$);
$+$ – сложение в кольце $\mathbf{Z}_{2^m}$;
$-$ – вычитание в кольце $\mathbf{Z}_{2^m}$;
$<<< l$ – циклический сдвиг на l бит влево;
$>>> l$ – циклический сдвиг на l бит вправо;
mod m – остаток от деления на m;
x ‖ y – конкатенация операндов x и y (x – старшая часть, y – младшая часть).

Для обозначения совокупностей элементов, представляющих единое целое, будем использовать два способа записи:

$(x_0,\dots,x_m)$ – способ записи множества либо последовательности элементов, для которых порядок взаимного расположения в памяти шифратора значения не имеет (с точки зрения применяемых операций);
$\langle x_m \| \dots \| x_0 \rangle$ – способ записи вектора, который отражает необходимый порядок расположения его элементов в памяти шифратора («младшие» справа).

Отдельные разряды будем обозначать b_i, байты – B_i, а слова – W_i, используя при этом Little Indian нотацию, т.е. нумерация разрядов, байтов, слов и т.д. выполняется от младших «весов» к старшим: разряд b_i имеет «вес» 2^i, а байт B_i – «вес» 2^{8i} и т.д.

Например:

$X \in \mathbf{H} = \mathbf{W}^4$ $(n = 4)$, $W_i \in \mathbf{W}$, $B_i \in \mathbf{B}$, $b_i \in \{0,1\} \Rightarrow$

$X = \langle W_3 \| W_2 \| W_1 \| W_0 \rangle = \langle B_{31} \| \dots \| B_1 \| B_0 \rangle = \langle b_{255} \| \dots \| b_1 \| b_0 \rangle$,

$W_0 = \langle B_7 \| \dots \| B_1 \| B_0 \rangle = \langle b_{63} \| \dots \| b_1 \| b_0 \rangle$,

$W_1 = \langle B_{15} \| \dots \| B_9 \| B_8 \rangle = \langle b_{127} \| \dots \| b_{65} \| b_{64} \rangle$,

$B_0 = \langle b_7 \| \dots \| b_1 \| b_0 \rangle$,

$B_{15} = \langle b_{127} \| \dots \| b_{121} \| b_{120} \rangle$;

X = 0x123456789FFEEDDCCBBAA9988776655 4433221100 $\Rightarrow$

W_0 = 0x7766554433221100,
W_1 = 0xFFEEDDCCBBAA9988,
B_0 = 0x00, B_1 = 0x11, B_{15} = 0xFF.

Дальше будет использоваться следующая нотация:

- символы операций +, –, <<<, >>>, заключённые в квадратные скобки (например [+]), будем использовать для обозначения операций выполняемых над словами (64 бита);
- верхний индекс (например Fun^n) – обозначает количество повторений (n) некоторого преобразования (Fun) для векторных аргументов (длиной n элементов);
- нижний индекс (например X_i) – обозначает номер (позицию) элемента в некоторой последовательности (векторе);
- верхний индекс в круглых скобках (например $X^{(i)}$) – обозначает номера итерации (или цикла);
- нижний индекс в угловых скобках (например $X_{<64n>}$) – обозначает разрядность вектора (X);
- символы с 1 – 3 штрихами (например, X″) – обозначают промежуточные результаты.

1.2. Основные определения

Блоком будем называть битовый кортеж длина которого соответствует разрядности входа шифратора, т.е. 128n бит.

Полублоком будем называть битовый кортеж длиной 64n битов. Блок (T) состоит из двух полублоков: левого L («старшего») и правого R («младшего»), т.е. $T = \langle L \| R \rangle$.

S-блоком будем называть группу совместно вычисляемых нелинейных булевых функций от общего ограниченного множества аргументов (8 бит), реализуемых, как правило, табличным способом.

Слоем будем называть последовательность идентичных преобразований, выполняемых «параллельно», т.е. независимо для каждого из элементов вектора-аргумента.

Числом ветвей активизации линейного преобразования будем называть минимальное суммарное количество активных S-блоков на входе и выходе этого преобразования, при условии фактической активизации входа. Для байтовых S-блоков – это минимальное количество активных байтов до и после преобразования. В этом случае, число ветвей активизации может быть определено, как минимальное суммарное количество ненулевых байтов на входе и выходе этого преобразования, при отображении отличного от нуля входного значения.

Циклической матрицей будем называть такую квадратную матрицу все строки (и столбцы) которой могут быть получены циклическим сдвигом одной строки (или столбца). Такая матрица может быть однозначно определена только первым столбцом (или строкой). Полином с коэффициентами (при формальных x), соответствующими элементам первого столбца, будем называть *полиномом порождающим* циклическую матрицу.

Циклической матрицей с максимальным числом ветвей или просто *CMBN-матрицей (CMBN – Cyclic Maximum Branch Number)*, будем называть циклическую матрицу размерностью t × t (с элементами из поля $GF(2^m)$), для которой операция умножения образует отображение векторов с максимальным числом «ветвей активизации». Одна ветвь соответствует одному элементу во входном либо выходном векторе. Максимальное достижимое число ветвей активизации для невырожденной матрицы составляет (t + 1). Полином степени (t – 1), порождающий такую матрицу будем называть *MBN-полиномом*. Отметим, что свойства такого преобразования, по сути своей, подобно *MDS-преобразованию*, т.е. умножению на *циклический разделимый код максимального расстояния* (*циклический МДР- или MDS-код*). Однако, учитывая, что данное преобразование является биективным, и мы не вносим избыточность в результат, эти два понятия (*MBN-* и *MDS-* преобразования) мы будем разделять. Процедуру умножения вектора на CMBN-матрицу также будем называть *CMBN-кодированием*, а порождающий полином *CMBN-кодом*.

Под одним *циклом* шифрования будем понимать две итерации цепи Фейстеля. Выбор такой нотации обусловлен двумя причинами:

- управляемое ключом изменение каждого бита блока данных возможно после применения не менее чем 2 итераций цепи Фейстеля;
- сохранение «совместимости» с системой обозначений, использовавшейся в предыдущих публикациях и ряде зарубежных публикаций.

Исходным (или пользовательским) ключом будем называть битовый вектор, подаваемый на ключевой вход шифратора.

Процедурой «разворачивания ключа» будем называть преобразование, выполняемое до применения процедуры шифрования и предназначенное для формирования, на основе исходного ключа, ключевого материала, используемого процедурой шифрования. По своей длине, рабочий ключ может превосходить исходный до нескольких десятков раз.

Рабочим ключом будем называть ключевую последовательность, непосредственно используемую процедурой шифрования и полученную в результате работы процедуры разворачивания ключа».

Подключом (рабочего ключа) или *рабочим подключом* будем называть элемент рабочего ключа, используемый на одной итерации или в отдельном преобразовании (например, начальном или конечном). Рабочий подключ, используемый на одной итерации, также, будем называть *ключом итерации*.

2. Элементарные преобразования алгоритма «Лабиринт»

Все элементарные преобразования в алгоритме «Лабиринт» имеют векторную структуру, т.е. блок либо полублок данных следует интерпретировать как последовательность слов, которые, в свою очередь, могут рассматриваться как последовательность из 8 байтов.

2.1. Векторные операции над словами

Ниже приведены обозначения векторных элементарных операций, выполняемых над словами, блоками либо полублоками, т.е. первый либо оба аргумента имеют вид последовательности из 1, 2n либо n слов соответственно. Результат этих операций имеет ту же размерность, что и первый аргумент.

- $x\ [+]^t\ y$ – сложение в кольце $\mathbf{Z}_{2^{64}}$ (по модулю 2^{64}) одноимённых слов, составляющих вектора *x* и *y* (длиной по *t* слов каждый);
- $x\ [-]^t\ y$ – вычитание в кольце $\mathbf{Z}_{2^{64}}$ (по модулю 2^{64}) слов, составляющих вектор *y* из одноимённых слов, составляющих вектор *x* (длина векторов равна t слов);
- $x\ [<<<]^t\ l$ – циклический сдвиг каждого из *t* слов, составляющих вектор *x*, на *l* бит влево ($0 \le l \le 63$);
- $x\ [>>>]^t\ l$ – циклический сдвиг каждого из t слов, составляющих вектор *x*, на *l* бит вправо ($0 \le l \le 63$).

Для обозначения векторных (биективных) преобразований общего вида над словами будем использовать аналогичную нотацию:

$$Y = Fun^t(X, K).$$

Подобная запись означает, что входной (X) и выходной (Y) вектора данных состоят из t слов, а вектор управляющего сигнала K (для управляемых отображений) состоит из t последовательных кортежей одинаковой длины, и каждое слово результата Y получается в результате независимого применения преобразования Fun к соответствующему слову аргумента X, с использованием соответствующего управляющего кортежа ключа K.

2.2. Элементарные операции над байтами

Кроме перечисленных выше бинарных операций, к элементарным операциям также будем относить унарную операцию над байтами – *нелинейную подстановку* 8-битных векторов (*S-блок*). Применение этой операции к отдельному байту будем обозначать: $y = S(x)$, $x, y \in \mathbf{B}$. Для обозначения преобразования обратного к данному, будем использовать следующую запись: $x = S^{-1}(y)$.

Для обозначения векторных преобразований над байтовыми строками, будем использовать обозначения аналогичные введенным выше, например:

$$Y = [S]^{8n}(X), \quad X, Y \in \mathbf{H}.$$

3. Структура алгоритма «Лабиринт»

3.1. Преобразования зашифрования и расшифрования

Алгоритм «Лабиринт» является «инволютивным» шифром, т.е. зашифрование и расшифрование выполняются на основе одной общей процедуры, отличие заключается только в противоположной последовательности применения ключей итераций $K^{(i)}$, а также ключей начального и конечного преобразований (подключи K^{IT} и K^{FT} соответственно).

Преобразования зашифрование/расшифрования состоит из двух фаз:

1. Процедура разворачивания ключа KShed, на основе исходного ключа UK выполняет формирование подключей рабочего ключа WK. Порядок применения подключей определяется необходимым «направлением» преобразования (mode) – зашифрование (e) или расшифрование (d).

2. Процедура шифрования EF выполняет преобразование входного блока данных (P либо C), на рабочем ключе WK:

$$WK = KShed(UK, mode = e\backslash d);$$

$$C = E_{UK}(P) = EF_{WK_E}(P);$$

$$P = D_{UK}(C) = EF_{WK_D}(C), \quad P, C \in \mathbf{T};$$

$$WK_E = \left(K^{IT}, K^{(0)}, K^{(1)}, \ldots, K^{(2r-1)}, K^{FT}\right);$$

$$WK_D = \left(K^{FT}, K^{(2r-1)}, K^{(2r-2)}, \ldots, K^{(0)}, K^{IT}\right),$$

где KShed – процедура разворачивания ключа; mode – режим преобразования: зашифрование (e) / расшифрование (d); UK – исходный (пользовательский) ключ; WK_E – рабочий ключ зашифрования (прямая последовательность подключей); WK_D – рабочий ключ расшифрования (обратная последовательность подключей); $K^{(i)}$ – ключ для i-й итерации; E_{UK} – преобразование зашифрования на исходном ключе UK; D_{UK} – преобразование расшифрования на исходном ключе UK; EF_{WK} – процедура шифрования на рабочем ключе WK; P – блок «открытого» текста; C – блок криптограммы.

3.2. Процедура шифрования EF

Алгоритм «Лабиринт» является итеративным шифром, т.е. основу его процедуры шифрования составляет цикловое преобразование, которое повторяется заданное число раз (обозначим число циклов шифрования символом r). Для алгоритма «Лабиринт» каждый цикл состоит из двух абсолютно идентичных итераций, однако, учитывая, что для обновления обоих полублоков, составляющих один блок, требуется, как минимум, две итерации, в данной спецификации понятие цикла отделено от понятия итерации. Кроме повторяемого циклового преобразования процедура шифрования включает начальное (IT) и конечное (FT) преобразования. Свойство инволютивности шифра достигается за счёт применения классической конструкции полублоковой цепи Фейстеля (рис. 1).

Алгоритм «Лабиринт», независимо от длины блока и ключа использует фиксированное количество итераций шифрования – 16 (или, иначе говоря, 8 циклов, т.е. r = 8). Процедура шифрования EF состоит из трёх шагов:

1. Исходный блок данных P (длиной 128n бит) обрабатывается начальным (IT) преобразованием на ключе K^{IT}.

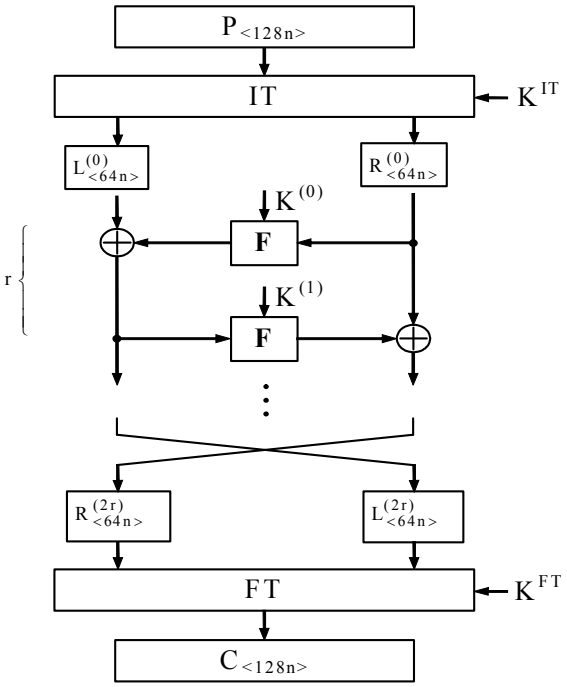

Рис. 1. Структура процедуры шифрования EF

2. Результат 1-го шага разбивается на два полублока длиной по 64n бита: левый L («старший») и правый R («младший»). Полученная пара полублоков преобразуется на 16 итерациях (8 циклах) цепи Фейстеля. Все итерации полностью идентичны и построены на базе общего нелинейного преобразования – F-функции, управляемой ключом итерации $K^{(i)}$.

3. Два полублока L и R, полученные в результате 8-циклового итеративного преобразования, меняются местами и обрабатываются конечным (FT) преобразованием на ключе K^{FT}. Полученный после FT преобразования двоичный вектор C (длиной 128n бит) является результирующим блоком («криптограммой»).

Аналитически, процедура EF может быть представлена следующим образом:

$$C = EF_{WK}(P):$$

$$P,\ C, K^{IT}, K^{FT} \in \mathbf{T},\ L^{(i)},\ R^{(i)},\ K^{(i)} \in \mathbf{H};$$

$$WK = \left(K^{IT}, K^{(0)}, \ldots, K^{(2r-1)}, K^{FT}\right);$$

$$\left\langle L^{(0)} \,\|\, R^{(0)} \right\rangle = IT_{K^{IT}}(P)$$

$$i = \overline{0, 2r-1}: \begin{cases} R^{(i+1)} = L^{(i)} \oplus F_{K^{(i)}}\left(R^{(i)}\right) \\ L^{(i+1)} = R^{(i)} \end{cases}$$

$$C = FT_{K^{FT}}\left(\left\langle R^{(2r)} \,\|\, L^{(2r)} \right\rangle\right).$$

В приведенных выше соотношениях символом i обозначается номер итерации, а символом r – количество циклов шифрования (один цикл – две итерации).

На каждой итерации F-функция использует 64n-битный подключ $K^{(i)}$. Длина ключей начального (K^{IT}) и конечного (K^{FT}) преобразований составляет по 128n бит каждый.

3.3. Базовая F-функция

Конструкция F-функции алгоритма «Лабиринт» построена в соответствии с принципами, позволяющими обеспечить свойства рассеивания и размножения активизации [11].

F-функция состоит из четырех элементарных преобразований:

- сложение слов полублока со словами ключа итерации по модулю 2^{64};
- фиксированная байтовая перестановка P;
- фиксированная нелинейная подстановка байтов составляющих полублок;
- фиксированное преобразование линейного смешивания.

Подробно эти преобразования будут рассмотрены ниже.

F-функция использует один подключ длиной 64n бит и имеет следующее аналитическое представление:

$$Y = F_{K^{(i)}}(X) = SL^n\left(P\left(X\,[+]^n\ K^{(i)}\right)\right),\ X, Y, K^{(i)} \in \mathbf{H}.$$

В общем виде F-функция может быть представлена следующей схемой (рис. 2).

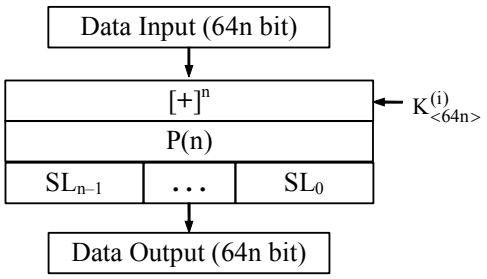

Рис. 2. F-функция общего вида (n = 1, 2 или 4)

Частный случай F-функции, когда n = 1 (т.е. длина полублока составляет 64 бита), представлен на рис. 3. В этом случае (n = 1) преобразование P_{byte}, как будет показано ниже, является тождественным, поэтому на схеме (рис. 3) оно отсутствует.

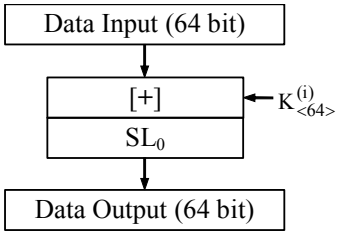

Рис. 3. 64-битная F-функция (n = 1)

При выполнении векторных операций (т.е. сложение с итеративным ключом $K^{(i)}$ и преобразование SL^n), полублок данных рассматривается как последовательность из n слов (по 64 бита), к каждому из которых независимо применяется соответствующее преобразование:

$$X, Y, K^{(i)} \in \mathbf{H},\quad W_j, W'_j, K_j^{(i)} \in \mathbf{W},\quad j = \overline{0, n-1};$$

$$X = \left\langle W_{n-1} \| \ldots \| W_1 \| W_0 \right\rangle;$$

$$Y = \left\langle W'_{n-1} \| \ldots \| W'_1 \| W'_0 \right\rangle;$$

$$K^{(i)} = \left\langle K_{n-1}^{(i)} \| \ldots \| K_1^{(i)} \| K_0^{(i)} \right\rangle;$$

$$Y = X\,[+]^n\ K^{(i)} \ \Rightarrow\ W'_j = W_j\,[+]\,K_j^{(i)};$$

$$Y = SL^n(X) \ \Rightarrow\ W'_j = SL_j(W_j).$$

3.4. Инволютивная операция линейного смешивания IMix

В составе начального (IT) и конечного (FT) преобразований используется операция *инволютивного линейного смешивания* (IMix). Данное преобразование предназначено для предотвращения возможности раздельной активизации полублоков с помощью характеристик с малым количеством активных байтов. Особенностью этого преобразования является свойство инволютивности, т.е. двухкратное применение преобразования IMix приводит к тождественному отображению аргумента.

Для описания данного преобразования воспользуемся следующими обозначениями:

$$\left\langle L' \| R' \right\rangle = IMix\left(\left\langle L \| R \right\rangle\right):\ \ \mathbf{T} \to \mathbf{T},$$

$$L, R \in \mathbf{H},\quad L_j, R_j, \Sigma_j \in \mathbf{W};$$

$$L = \left\langle L_{n-1} \| \ldots \| L_1 \| L_0 \right\rangle,\quad R = \left\langle R_{n-1} \| \ldots \| R_1 \| R_0 \right\rangle,$$

где L и R – соответственно, левый и правый полублоки.

С учётом введенных обозначений преобразование IMix может быть записано следующим образом:

$$\text{for } j = \overline{0, n-1}:$$

$$L'_j = L_j \oplus \Sigma_{(j-1) \bmod n};\quad R'_j = R_j \oplus \Sigma_{(j-1) \bmod n};$$

$$\Sigma_j = \begin{cases} (L_j \oplus R_j)\ [<<<]\ 28, & j = 0; \\ L_j \oplus R_j, & j > 0. \end{cases}$$

3.5. Начальное IT и конечное FT преобразования

Начальное IT (рис. 4) и конечное FT (рис. 5) преобразования включают сложение блока данных с рабочим подключом (длиной 128n бит) и фиксированное преобразование блока данных. Фиксированные (т.е. не зависящие от ключа) составляющие преобразований IT и FT являются взаимно обратными и включают два «слоя»:

- нелинейная подстановка («S-слой»);
- линейное смешивание («L-слой»).

«L-слой» обоих преобразований удобно рассматривать как последовательность из двух операций:

- циклический сдвиг слов левого и правого полублоков на 2 разряда влево и вправо;
- линейное смешивание полублоков IMix.

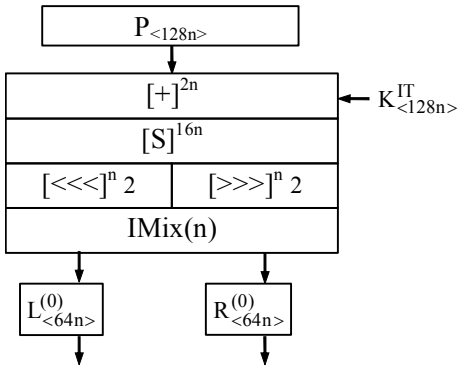

Рис. 4. Начальное IT-преобразование

Начальное преобразование (IT) может быть представлено следующей последовательностью операций:

$$\left\langle L^{(0)} \parallel R^{(0)} \right\rangle = IT_{K^{IT}}(P): \quad L, R \in \mathbf{H}, \quad P, T, K^{IT} \in \mathbf{T}:$$

1) $T = P \ [+]^{2n} \ K^{IT}$;

2) $\left\langle L' \parallel R' \right\rangle = [S]^{16n}(T)$;

3) $L'' = L' \ [<<<]^n \ 2, \quad R'' = R' \ [>>>]^n \ 2$;

4) $\left\langle L^{(0)} \parallel R^{(0)} \right\rangle = IMix\left(\left\langle L'' \parallel R'' \right\rangle\right)$,

где P – исходный блок текста на входе шифратора («открытый текст»); $L^{(0)}$, $R^{(0)}$ – соответственно, левый и правый полублоки на выходе IT-преобразования.

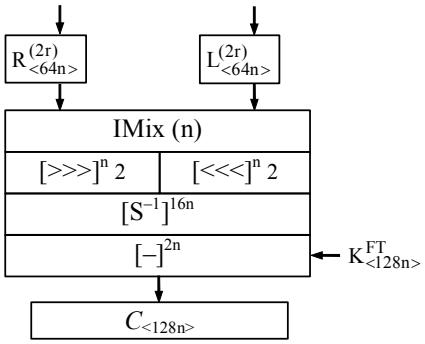

Рис. 5. Конечное FT-преобразование

Конечное преобразование (FT) может быть представлено следующей последовательностью операций:

$$C = FT_{K^{FT}}\left(R^{(2r)} \parallel L^{(2r)}\right): \quad L, R \in \mathbf{H}, \quad C, T, K^{FT} \in \mathbf{T}:$$

1) $\left\langle L' \parallel R' \right\rangle = IMix\left(\left\langle R^{(2r)} \parallel L^{(2r)} \right\rangle\right)$;

2) $L'' = L' \ [>>>]^n \ 2, \quad R'' = R' \ [<<<]^n \ 2$;

3) $T = [S^{-1}]^{16n}\left(\left\langle L'' \parallel R'' \right\rangle\right)$;

4) $C = T \ [-]^{2n} \ K^{FT}$,

где C – результирующий блок текста на выходе шифратора («криптограмма»); $L^{(2r)}$, $R^{(2r)}$ – соответственно левый и правый полублоки после последней итерации цепи Фейстеля.

Следует обратить внимание, что после выполнения последней итерации цепи Фейстеля, левый и правый полублоки должны поменяться местами, и только после этого результирующий блок поступает на вход FT-преобразования. Однако, в ряде случаев, с целью повышения эффективности реализации, указанная перестановка может быть совмещена с FT-преобразованием.

3.6. Фиксированная перестановка байтов P(n)

Вид байтовой перестановки P(n) определяется количеством 64-битных слов в полублоке (т.е. параметром n). Байтовая перестановка выполняется в соответствии со следующим соотношением:

$$P(n): \quad B'_i = B_{(9i \bmod 8n)}, \quad B_i \in \mathbf{B}, \quad i = \overline{0, 8n-1},$$

где B_j – байт-источник (где j – позиция байта в исходном полублоке); B_i'– байт-получатель (где i – позиция байта в результирующем полублоке).

На рис. 6 – 8 показаны схемы перестановки байтов для всех допустимых значений n (т.е. 1, 2 и 4). На приведенных схемах каждая ячейка соответствует одному байту, «большое» число в ячейке – исходная позиция (64-битного) слова в полублоке, а нижний индекс – исходная позиция байта внутри соответствующего (64-битного) слова.

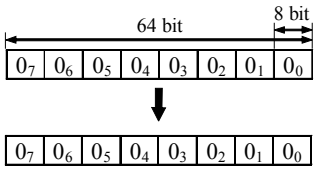

Рис. 6.Перестановка P для случая n = 1

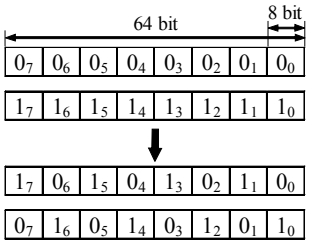

Рис. 7. Перестановка P для случая n = 2

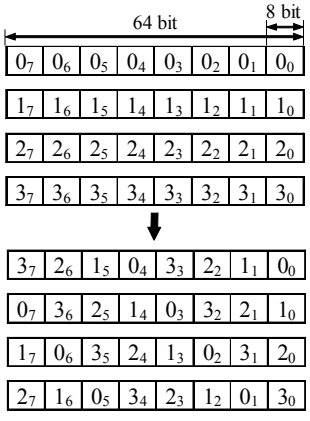

Рис. 8. Перестановка P для случая n = 4

3.7. Узел замены (S-box)

S-блок (или узел замены) определяет фиксированное биективное преобразование отдельного байта. S-блок реализуется в виде таблицы подстановки размером 256 байт.

S-блок построен в соответствии с критериями отбора и конструкцией изложенной в [12]. Функция S-блока может быть представлена в табличном виде (табл. 1). Алгебраическая структура подстановки аффинно эквивалентна конструкции Ниберг-Динга, детальное описание алгебраической конструкции S-блока будет приведено в разделе 4.2. Значение узла замены может быть заменено другим, при условии, что он будет удовлетворять требованиям, приведенным в разделе 4.2. Отметим, что значение S-блока должно выбираться совместно с параметрами MBN-преобразования.

Таблица 1

	00	01	02	03	04	05	06	07
00	2B	F8	D2	D5	35	A2	3F	C6
10	60	7E	E0	7F	2D	AC	E3	0D
20	80	63	C9	46	79	E7	89	E9
30	71	B8	B2	02	06	F2	E5	FD
40	10	B9	50	08	A1	A8	7D	40
50	6E	7B	77	54	92	58	95	C1
60	2A	CC	B4	0C	DF	A7	27	9E
70	B1	11	C8	AA	90	18	45	36
80	E6	F4	73	BE	07	6A	42	F7
90	74	87	A5	8B	1A	9C	4E	6B
A0	DA	64	7A	8F	EF	D4	93	AF
B0	C4	65	03	BF	D9	68	C5	E2
C0	A0	12	83	ED	8C	00	57	DD
D0	FA	5C	49	39	43	5E	86	0F
E0	3A	70	E4	1E	04	55	72	DE
F0	A9	F0	5F	AE	09	8A	81	53

	08	09	0A	0B	0C	0D	0E	0F
00	BB	C2	A3	F1	9F	6F	1D	E1
10	BC	9D	C0	FE	3B	D1	1B	C3
20	1C	AB	17	97	5A	20	30	EC
30	28	D3	3E	3C	D0	BA	CE	29
40	01	15	7C	78	33	69	EB	0E
50	98	EE	1F	9B	96	51	26	61
60	32	37	B3	FC	0A	AD	2C	19
70	75	94	8E	CB	16	BD	FB	48
80	41	4C	05	EA	DC	76	D8	6C
90	0B	24	91	34	4A	2E	F3	E8
A0	66	13	CF	82	59	D7	31	4F
B0	84	A6	23	99	C7	B0	5B	62
C0	22	FF	9A	A4	F5	F9	3D	6D
D0	B7	67	52	CA	14	38	DB	25
E0	56	47	CD	B6	8D	85	88	D6
F0	21	B5	F6	4B	4D	5D	44	2F

Алгоритм «Лабиринт» построен на базе одного общего S-блока, однако в составе конечного FT-преобразования используется обратный (или инверсный), S-блок (обозначается S^{-1}), который определяет преобразование обратное к «прямому» S-блоку, т.е.:

$$S: \quad \mathbf{B} \to \mathbf{B};$$

$$\forall x: \quad y = S(x) \quad \Rightarrow \quad x = S^{-1}(y), \quad x, y \in \mathbf{B}.$$

Отметим, что, обычно, преобразования этого класса реализуются табличным способом. При этом, особенно при программной реализации, преобразование на узле замены может быть эффективно совмещено с реализацией последующего фиксированного линейного преобразования табличным способом, т.е. оба преобразования могут быть выполнены с помощью одной общей таблицы.

3.8. MBN-преобразование

Это преобразование определяет биективное линейное отображение слов, составляющих полублок. MBN-преобразование может быть представлено в виде матричного умножения – квадратная матрица размерностью 8×8 байт, образованная циклическим MBN-кодом, умножается справа на вектор-столбец длиной 8 байт (соответствующий слову-аргументу). Полученный вектор-столбец (длиной 8 байт) соответствует слову-результату. Элементы матрицы (байты), и элементы векторов аргумента / результата, интерпретируются как элементы поля $GF(2^8)$, образованного выбранным неприводимым (над полем GF(2)) полиномом 8-й степени $f_{MBN}(x)$.

Учитывая циклическую структуру используемого MBN-кода, данное преобразование может быть записано следующим образом:

$$\text{CMBN}: \quad \mathbf{W} \to \mathbf{W};$$

$$Y = \text{CMBN}(X), \quad X, Y \in \mathbf{W}, \quad B_j \in \mathbf{B};$$

$$X = \langle B_7 \| \ldots \| B_1 \| B_0 \rangle;$$

$$Y = \bigoplus_{j=0}^{7} \text{LCM}\left(B_j\right) <<< (8 \times j), \quad \text{LCM}: \quad \mathbf{B} \to \mathbf{W},$$

где LCM – операция умножения одного байта на циклический MBN-код.

LCM-отображение определяется операцией умножения элемента поля $GF(2^8)$, соответствующего байту-аргументу, на фиксированный полином 7 степени с коэффициентами из $GF(2^8)$. Используемый полином формирует циклический MBN-код. С точки зрения реализации алгоритма, принципиальным является свойство цикличности порождаемого кода, так как на нём базируется структура MBN-преобразования.

Указанное выше LCM-отображение имеет следующую аналитическую запись:

$$\text{LCM}: \quad \mathbf{B} \to \mathbf{W};$$

$$Y = \text{LCM}(X), \quad X \in \mathbf{B}, \quad Y \in \mathbf{W};$$

$$g(x) = g_7 x^7 + \ldots + g_1 x + g_0;$$

$$Y = \langle g_7 X \| \ldots \| g_1 X \| g_0 X \rangle, \; X, g_i, (g_i X) \in F_{2^8}, \; g(x), Y \in F_{2^8}^8,$$

где g(x) – полином, порождающий циклический MBN-код; X – входной байт; Y – выходное слово.

В данной спецификации используется следующий полином, порождающий CMBN-код:

$$g(x) = 0C\,x^7 + 0F\,x^6 + 1F\,x^5 + 0A\,x^4 + 0F\,x^3 + 03\,x^2 + 01\,x + 01,$$

с коэффициентами из поля $GF(2^8)$, которое задано следующим неприводимым (над полем GF(2)) поли-

номом 8-й степени (в шестнадцатеричном представлении 0x12D):

$$f_{MBN}(x) = x^8 + x^5 + x^3 + x^2 + 1.$$

Отметим, что приведенное значение полинома g(x), порождающего циклический MBN-код (а также полинома $f_{MBN}(x)$), не является обязательным для алгоритма «Лабиринт» и может быть заменено другим. Возможность произвольного выбора S-блока и полинома порождающего MBN-код, а также «прозрачность» критериев их отбора позволяет решить проблему возможных «лазеек» (или «потайных дверей»), заложенных авторами алгоритма.

3.9. Преобразование SL

SL-преобразование определяет фиксированное биективное отображение слов. Это преобразование представляет собой объединение нелинейного преобразования байтов (S-блок), составляющих отдельное 64-битное слово, с последующим линейным «смешиванием» этих, уже преобразованных на S-блоке, байтов. Вид SL-преобразования зависит от позиции (t) слова (X), внутри полублока, к которому оно применяется.

Преобразование SL_t имеет следующее аналитическое описание:

SL_t: $\mathbf{W} \to \mathbf{W}$;

$Y = SL_t(X), \quad t = \overline{0, n-1}$;

$X = \langle B_7 \| \ldots \| B_1 \| B_0 \rangle, \quad B_j \in \mathbf{B}, \quad X, Y, xc_t \in \mathbf{W}$;

1) $X' = \langle S(B_7) \| \ldots \| S(B_1) \| S(B_0) \rangle$;

2) $Y' = MBN(X')$;

3) $Y = Y' \oplus xc_t$,

где X – входное слово (64 бита); Y – выходное слово (64 бита); t – номер (позиция внутри полублока) слова X, к которому будет применено данное преобразование; xc_t – значение XOR-константы, уникальное для каждого значения t, определяется из следующей таблицы:

t	xc_t
0	0x0
1	0x1111111111111111
2	0x2222222222222222
3	0x8888888888888888

3.10. Процедура «разворачивания ключа»

Из приведенной выше структуры шифрующего преобразования алгоритма «Лабиринт» следует, что процедура шифрования требует рабочий ключ длиной 16 × 64n + 2 × 128n бит (16 ключей итераций по 64n бит, а также ключи начального и конечного преобразований по 128n бит). Для формирования рабочего ключа указанной длины на основе исходного (пользовательского) ключа существенно меньшего объёма используется *процедура разворачивания ключа*. Процедура разворачивания ключа алгоритма «Лабиринт» поддерживает переменную длину пользовательского ключа, которая может находиться в диапазоне от 2 до 4 полублоков включительно, т.е. от 128n бит до 256n бит, с шагом 64n бит – один полублок.

Для сокращения требований к необходимому объему оперативной памяти, алгоритм использует принцип повторного использования ключевого материала на различных итерациях шифрования. Для хранения «развёрнутого» рабочего ключа необходим буфер объёмом 8 полублоков, т.е. 8 × 64n бит.

Процедура разворачивания ключа состоит из двух этапов:

- инициализация буфера рабочего ключа на основе исходного ключа пользователя;
- выборка подключей из буфера рабочего ключа.

Процедура инициализации заключается в загрузке исходного ключа в буфер рабочего ключа и дополнение этого ключа до полного заполнения буфера ключевым материалом, сформированным из исходного ключа. Для формирования «дополнительного» ключевого материала используется базовая F-функция шифратора, на базе которой построен простой криптографический генератор псевдослучайных последовательностей – исходный ключевой материал циклически зашифровывается в режиме «связки шифроблоков» (CBC-режим [13]).

Ниже приведено формальное описание процедуры разворачивания ключа (учитывая, что схематическое представление данной процедуры является не достаточно иллюстративным, ниже приводится только аналитическое описание).

Для описания процедуры разворачивания ключа воспользуемся следующими обозначениями:

$$\text{INPUT}: \quad l_{UK},\ uk_1, uk_2, \ldots, uk_{l_{UK}} \quad (2 \le l_{UK} \le 4);$$

$$\text{OUTPUT}: \quad WK = \left(K^{IT}, K^{(0)}, K^{(1)}, \ldots, K^{(2r-1)}, K^{FT}\right),$$

$$uk_j, k_j, k_{KS}, K^{(i)} \in \mathbf{H}, \qquad K^{IT}, K^{FT} \in \mathbf{T},$$

где l_{UK} – длина исходного ключа, введенного пользователем; uk_j – j-й полублок исходного ключа; k_j – j-я ячейка буфера рабочего ключа (длиной один полублок); k_{KS} – ключ, используемый F-функцией процедуры разворачивания ключа (длиной один полублок,); WK – «развёрнутый» рабочий ключ.

Процедура разворачивания ключа начинается с загрузки пользовательского ключа в шифратор, а именно в буфер рабочего ключа, одновременно выполняется инициализация ключа k_{KS}:

$k_{KS} = 0$;

for $j = \overline{0, l_{UK} - 1}$: $k_j = uk_j, \quad k_{KS} = k_{KS} \oplus uk_j$;

$k_{KS} = k_{KS}\ [<<<]^n\ 29$.

Как видно из приведенного описания, ключ k_{KS} вычисляется как сумма по модулю 2 всех полублоков пользовательского ключа. После накопления указанной суммы, выполняется циклический сдвиг на 29 разрядов каждого из n слов составляющих полублок k_{KS}. Дальнейшее заполнение буфера рабочего ключа выполняется согласно следующему рекуррентному соотношению:

for $j = \overline{l_{UK}, 7}$: $k_{KS} = \delta(k_{KS}), \quad k_j = F_{k_{KS}}(k_{j-1}) \oplus k_{j-l_{UK}}$,

где δ – функция «обновления» ключа k_{KS}, которая применяется перед каждым использованием F-функции в процедуре разворачивания ключа.

Функция δ имеет следующий вид:

$$\delta : \mathbf{H} \to \mathbf{H};$$

$$Y = \delta(X),\ X = \langle W_{n-1} \| \dots \| W_0 \rangle;$$

$$Y = \langle W'_{n-1} \| \dots \| W'_0 \rangle,\ X,\ Y \in \mathbf{H},\ W_j \in \mathbf{W};$$

$$W'_j = (W_j + 0x1084210842108421) \oplus$$

$$\oplus\ 0x4444444444444444,\ j = \overline{0,\ n-1}.$$

Полученное заполнение буфера рабочего ключа $\{k_i\}$ используется процедурой зашифрования следующим образом:

$$K^{IT} = \langle k_7 \| k_4 \rangle;$$

$$\text{for } i = \overline{0,7}: \quad K^{(i)} = k_i;$$

$$\text{for } i = \overline{8,15}: \quad K^{(i)} = k_{(3i \bmod 8)};$$

$$K^{FT} = \langle k_6 \| k_3 \rangle.$$

Для выполнения процедуры расшифрования все подключи «зашифрования» располагаются в обратном порядке, что аналитически может быть записано следующим образом:

$$K^{IT} = \langle k_6 \| k_3 \rangle;$$

$$\text{for } i = \overline{0,7}: \quad K^{(i)} = k_{(3\times(7-i) \bmod 8)};$$

$$\text{for } i = \overline{8,15}: \quad K^{(i)} = k_{7-(i \bmod 8)};$$

$$K^{FT} = \langle k_7 \| k_4 \rangle.$$

Эквивалентно последовательность выбора подключей итераций $K^{(i)}$, для преобразования зашифрования, может быть задана в табличном виде. При выполнении расшифрования эти подключи выбираются в обратном порядке.

Одним из достоинств данной схемы разворачивания ключа, и всей конструкции шифра в целом, является применимость одного и того же материала «развёрнутого» ключа, как для преобразования зашифрования, так и расшифрования.

4. Свойства преобразований алгоритма «Лабиринт»

4.1. Преобразования IT и FT

Применение начального (IT) и конечного (FT) преобразований позволяет достигнуть три цели:

- «рандомизация» входа и выхода шифратора;
- исключение возможности раздельной активизации полублоков на входе (и выходе) цепи Фейстеля, т.е. защита от NR-атак;
- увеличение количества S-блоков активизируемых на первой итерации цепи Фейстеля.

«Рандомизация» входа / выхода достигается за счёт введения ключа (по модулю 2^{64}) и последующего / предшествующего применения нелинейного преобразования байтов (S-блока).

Вторая цель достигается за счёт применения линейного комбинирующего преобразования IMix. Это преобразование является не только инволютивным, но также обладает свойством «покрытия» единственным «путём» всех байтов каждого из полублоков. За счёт этого, для активизации только одного полублока на входе цепи Фейстеля необходимо активизировать, как минимум, все байты одного из полублоков, и только в случае выполнения условия «коллизии разностей» будет активизирован только один полублок. Вероятность такого события составит $(1/255)^{8n-1}$. Если на входе шифратора активизируется менее чем 8n S-блоков, то на первой итерации цепи Фейстеля будет активизирован как минимум один S-блок. Таким образом, стратегия «отключения» первой (последней) итерации цепи Фейстеля становится не эффективной для алгоритма «Лабиринт».

Увеличение количества активных S-блоков на первой итерации цепи Фейстеля достигается за счёт применения комбинации из «слоя» циклических сдвигов, выполняемых после «слоя» нелинейной подстановки и последующего применения линейного смешивающего преобразования IMix. Если на входе шифратора был активизирован только один S-блок, то после выполнения IT преобразования активными будут минимум 3 байта, а максимум 6. Активизация только двух байтов, расположенных в идентичных позициях обоих полублоков, на входе шифратора, приводит к активизации от 2 до 10 байт (число всегда чётное) на выходе IT-преобразования, которые поровну распределяются между левым и правым полублоками. При этом, вероятность получения только двух активных байт после IT-преобразования, при такой акивизации входа, составляет 2^{-10}. Эти свойства являются следствием применения циклического сдвига слов, составляющих полублоки, на 2 разряда в противоположные стороны для разных полублоков. Такой сдвиг приводит, в составе IMix преобразования, к сложению (по модулю 2) двух байтовых цепочек сдвинутых одна относительно другой на 4 разряда. Таким образом, для того чтобы два байта, активных на входе IT, после сдвига и сложения не привели к активизации двух других байтов, в результате выполнения IMix преобразования, необходимо, чтобы на выходе S-слоя они имели по 4 нулевых бита в своих выходных характеристиках (один 4 старших, другой 4 младших), а оставшиеся 4 бита у них должны совпадать – только в этом случае мы получим выполнение условия «коллизии». Вероятность такого события равна 2^{-10}.

4.2. Узел замены (S-блок)

S-блок должен выбираться из множества, так называемых, предельно-нелинейных биективных преобразований [5, 6]. Поэтому в алгоритме используется S-блок, построенный на основе конструкции Ниберг-Динга, т.е. преобразования аффинно-эквивалентного функции вычисления обратного элемента в поле $GF(2^8)$:

$$S(X) = \mathbf{M_Y} \times \left\lfloor (\mathbf{M_X} \times X \oplus \mathbf{V_Y})^E \right\rfloor_B \oplus \mathbf{V_Y},$$

$$X, \mathbf{V_X}, \mathbf{V_Y} \in \mathbf{F}_2^8,\ \mathbf{M_X}, \mathbf{M_Y} \in \mathbf{GL}(8, \mathbf{F}_2),$$

$$E = 2^8 - 1 - 2^t,\quad 0 \le t < 8,$$

где B – некоторый базис над $GF(2^8)$, определяется образующим (неприводимым) полиномом 8-й степени $f_S(x)$; E – показатель степени; $\mathbf{M_X}$, $\mathbf{M_Y}$ – квадратные невырожденные матрицы размерностью (8×8).

В указанном соотношении, для упрощения записи, вектора, участвующие в матричном умножении, рассматриваются как вектор – столбцы.

Отметим, что преобразование S(x) может быть однозначно определено с помощью меньшего количества параметров.

Применение указанной конструкции позволяет достигнуть предельно низких значений вероятностей заданного трансформирования дифференциальной разности ($p_S^{<dc>} = 2^{-6}$) и линейной аппроксимации ($p_S^{<lc>} = 2^{-6}$) S-блока, а также получить максимальную алгебраическую степень булевого полинома ($\deg_y = 7$). При этом, выбор входного ($\mathbf{M_X}$) и выходного ($\mathbf{M_Y}$) аффинных преобразований, а также векторов $\mathbf{V_X}$ и $\mathbf{V_Y}$ осуществлялся таким образом, чтобы обеспечить выполнение приведенных ниже требований [5, 6]. Значение S-блока, приведенное в данной спецификации было выбрано на основе случайного процесса, т.е. все указанные выше параметры S-блока (B, E, $\mathbf{M_X}$, $\mathbf{M_Y}$, $\mathbf{V_X}$, $\mathbf{V_Y}$) формировались на основании датчика случайных чисел, и полученное преобразование проверялось на удовлетворение дополнительным требованиям. Ниже приведен список дополнительных требований, использовавшихся для «фильтрации» S-блоков:

1) максимальное значение в таблице дифференциальных разностей для характеристик SUM→XOR должно составлять 5;

2) алгебраическая степень $\deg_x$ для всех переменных булевых функций, получаемых в результате умножения выхода S-блока на полином, порождающий MBN-код должна быть равна 7;

3) отсутствие «фиксированных точек», т.е. таких что S(x) = x.

Таким образом, можно составить общий список криптографических показателей, которые учитывались при выборе конструкции S-блока. Ниже приведено аналитическое описание указанных показателей (их более детальное описание приведено в [5, 6]):

1. Биективность.

2. Максимальная вероятность дифференциальной характеристики XOR→XOR $p_S^{<dc>}$.

3. Максимальная вероятность дифференциальной характеристики SUM→XOR $p_S^{<sdc>}$.

4. Максимальная вероятность линейной аппроксимации S-блока $p_S^{<lc>}$.

5. Максимальная вероятность отклонения «критерия распространения» $p_S^{<pc>}$.

6. Минимальная алгебраическая степень булевых полиномов S-блока $\deg_y(S)$.

7. Минимальная алгебраическая степень «присутствия» отдельных переменных булевых полиномов S-блока $\deg_x(S)$.

8. Отсутствие «фиксированных точек», т.е.: $\forall x: \; S(x) \neq x$.

Отметим, что выполнение требования 1 и получение предельно-достижимых (для биективного отображения) значений показателей 2, 4, 5, 6 обеспечивается за счёт применения конструкции Нибергт-Динга [5, 6], а достижение предельных значений для показателей 3, 7 и 8 выполнено за счёт соответствующего подбора входного и выходного аффинных преобразований. Кроме того, конструкция Нибергт-Динга обладает и другими позитивными криптографическими свойствами, в частности отсутствием «точек управляемой линейности» степени меньше 4.

Основным недостатком, этой конструкции является наличие квадратичных алгебраических соотношений, охватывающих вход и выход (выполняющихся с вероятностью 1). Однако опасность построения алгебраической атаки на шифр «Лабиринт» устраняется за счёт применения операции сложения с переносом.

S-блок, приведенный в спецификации, был сформирован на основании следующих параметров:

- неприводимый (над GF(2)) полином, образующий поле (в шестнадцатеричном представлении 0x1BD): $f_S(x) = x^8 + x^7 + x^5 + x^4 + x^3 + x^2 + 1$;
- показатель степени $E = 2^8 - 1 - 2^4 = 251$;
- матрицы «входного» и «выходного» аффинных преобразований приведены ниже («вес» двоичных разрядов возрастает «сверху вниз» и «слева направо», т.е. элемент M[0,0], находящийся в верхнем левом углу, соответствует самому младшему разряду):

$$\mathbf{M_X} = \begin{bmatrix} 10101110 \\ 01011001 \\ 01111001 \\ 00110011 \\ 00110100 \\ 01101000 \\ 01010011 \\ 01000100 \end{bmatrix}; \; \mathbf{V_X} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}; \; \mathbf{M_Y} = \begin{bmatrix} 11100010 \\ 01100011 \\ 10101001 \\ 11011101 \\ 11011001 \\ 01111001 \\ 01010011 \\ 11010111 \end{bmatrix}; \; \mathbf{V_Y} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}.$$

Ниже (табл. 1) приведено табличное описание используемого в данной спецификации узла подстановки (S-блока). Первый столбец и первая строка (выделенные жирным шрифтом) используются для указания индексов элементов подстановки. Индексы строк и столбцов таблицы, а также значения элементов таблицы указаны в шестнадцатеричной системе счисления (префикс 0x не используется для сокращения записи). «Логическое ИЛИ» индексов строки и столбца соответствует входному значению, результат подстановки которого находится на пересечении соответствующих строки и столбца. Например, результат подстановки шестнадцатеричного числа 0x53 будет равен 0x54.

Приведенное значение узла замены не является обязательным для алгоритма «Лабиринт» и приведено в данной спецификации в качестве примера. S-блок построен в соответствии с критериями отбора и конструкцией изложенной в [7]. Алгебраическая структура подстановки аффинно эквивалентна конструкции Ниберг-Динга, детальное описание алгебраической конструкции S-блока приведено выше. Значение узла замены может быть заменено другим, при условии, что он будет удовлетворять указанным выше требованиям. Отметим, что значение S-блока должно выбираться совместно с параметрами MBN-преобразования.

4.3. MBN-преобразование

Для выполнения «смешивания» байтов внутри одного слова (8 байт) используется операция умножения вектора из 8 элементов поля $GF(2^8)$, соответствующего входному слову, на порождающий многочлен CMBN-кода (*Cyclic Maximum Branch Number*) 8-й степени. Аналитически CMBN-преобразование может быть задано в следующем виде:

$$\text{CMBN:}\quad \mathbf{W} \to \mathbf{W};$$

$$b(x) = a(x)\, g(x) \bmod (x^8 + 1);$$

$$b(x) = R_{x^8+1}[a(x)\, g(x)];$$

$$a(x), b(x), g(x) \in F_{2^8}^8, \quad B_i \in F_{2^8};$$

$$a(x) = B_7 x^7 + \ldots + B_2 x^2 + B_1 x + B_0;$$

$$b(x) = B'_7 x^7 + \ldots + B'_2 x^2 + B'_1 x + B'_0;$$

$$\mathbf{B} = F_{2^8} = F_2[x] / (f_{MBN}(x));$$

$$\mathbf{W} = F_{2^8}^8 = F_{2^8}[x] / (x^8 + 1),$$

где a(x) – полиномиальное представление входного вектора; b(x) – полиномиальное представление вектора-результата; g(x) – полином образующий CMBN-код; $f_{MBN}(x)$ – примитивный полином, образующий поле $GF(2^8)$.

Достоинством операции умножения вектора на циклический MBN-код является простота её реализации табличным способом:

$$TL[B_i] = \langle g_7 B_i \,\|\, \ldots \,\|\, g_2 B_i \,\|\, g_1 B_i \,\|\, g_0 B_i \rangle;$$

$$B_i, g_j \in F_{2^8};$$

$$B'_i = g_j B_i, \quad i, j = \overline{0,7};$$

$$\langle B'_7, \ldots, B'_2, B'_1, B'_0 \rangle = \bigoplus_{i=0}^{7} TL[B_i] <<< (8 \times i).$$

Полином g(x) должен удовлетворять как минимум двум требованиям:

1) порождать циклический MBN-код;

2) любые 8 подряд идущих разряда результата умножения входного байта на порождающий полином g(x) должны описываться линейно независимыми булевыми функциями.

Отметим, что применение, в качестве преобразования линейного смешивания, MBN-кода обеспечивает следующие два свойства:

- получение максимально-достижимого числа ветвей активизации в пределах одного слова, т.е. для «активного» слова длиной 8 байт имеем в сумме (до и после преобразования) минимум 9 активных байт;
- минимальное количество активных байтов после MBN–преобразования зависит только от числа активных байтов на входе, и не зависит от фактического вида шаблона активизации.

При выборе полинома g(x), кроме необходимости порождения циклического MBN-кода, к нему предъявлялся ряд дополнительных требования:

- любые 8 соседних булевых функций, получаемых в результате умножения одного байта на полином g(x), должны быть линейно независимы, т.е., иначе говоря, любые 8 соседних разрядов выхода должны обеспечивать однозначное отображение входного байта;
- битовая длина коэффициентов полинома не должна превышать 5 бит;
- среди коэффициентов полинома должно быть не менее двух единичных коэффициентов;
- полином должен содержать не менее двух пар повторяющихся коэффициентов;
- подбор коэффициентов полинома должен обеспечивать возможность минимизации количества операций умножения в поле $GF(2^8)$ при программной реализации.

Отметим, что только первое из приведенных выше дополнительных требований является криптографическим, остальные требования призваны уменьшить сложность программной реализации операции умножения байта на полином $g(x)$. Присутствие первого дополнительного требования обусловлено использованием для ввода ключа операции сложения с переносом. Кроме того, неприводимые полиномы, используемые для построения MBN-кода и S-блока различны.

4.4. SL-преобразование

Свойства SL-преобразования определяются свойствами компонентов её составляющих, т.е. S-блока, линейного CMBN-преобразования и сложения (по модулю 2) с константой:

- каждый бит слова-результата SL-преобразования зависит от всех бит слова-аргумента.
- максимальная вероятность трансформирования любого шаблона активизации, при прохождении через SL-преобразование, составит $a \times p_S$, где a – количество активных S-блоков на входе преобразования, $p_S = 2^{-6}$ - максимальная вероятность ожидаемого преобразования входной активизации в выходную на отдельном S-блоке;
- применения, уникального значения XOR-константы, для каждой позиции слова j, на выходе SL_j-преобразования, обеспечивает уникальность отображения различных слов, составляющих полублок.

Рассмотрим особенности реализации SL-преобразования.

Применение матрицы с циклической структурой позволяет реализовать операцию матричного умножения в виде одной таблицы подстановки размером 256 слов (т.е. 256×8 байт = 2 КБайта), которая, в свою очередь, может быть совмещена с таблицей вычисления S-блока.

При программной реализации, в зависимости от архитектуры целевого микропроцессора и требований к реализации, более оптимальным может оказаться реализация табличным способом не только узла подстановки (умножение на первый столбец матрицы), но и его циклических сдвигов, необходимых для умножения байтов на последующие столбцы матрицы. В этом случае, операция циклического сдвига (по байтовой границе), для выполнения MBN-преобразования не используется, однако требуется увеличение размера таблицы подстановки, пропорционально разрядности регистров ALU используемой архитектуры:

Таблица 2

Разрядность регистров, бит	8	16	32	64
Размер таблицы, КБайт	2	4	8	16

Приведенные в табл. 2 значения являются следствием свойства цикличности MBN-матрицы, и обусловлены необходимостью *вычисления* циклического сдвига только для величин сдвига не кратных длине регистра (в байтах), для остальных величин достаточно операций «выборки» / «сохранения» в память.

4.5. Байтовая перестановка P(n)

Перестановка P(n) предназначена для «равномерного» перераспределения байтов, поступающих на вход SL_j-преобразования на двух соседних итерациях. Т.е., учитывая что SL-преобразование выполняет отображение слов, перестановка байтов «равномерно» рассеивает байты между словами, составляющими полублок, либо является тождественной для случая n = 1. Т.к. SL-преобразование, кроме нелинейной подстановки байтов, выполняет линейное смешивание выходов S-блоков в пределах одного слова, применение перестановки P(n) является простейшим способом «рассеивания» активизации S-блоков между различными словами полублока. Следует отметить, что реализация P(n) перестановки не связана с дополнительными вычислительными затратами, т.к. может быть совмещена с последующим преобразованием подстановки байтов по S-блоку.

Под «равномерным» рассеиванием байтов подразумевается следующее свойство: после применения перестановки P к полублоку, каждое 64-битное слово-результат содержит одинаковое число байт из каждого слова-источника, при этом байты сохраняют свою позицию в пределах слова. Это свойство учитывалось при выборе величины циклического сдвига IMix преобразования, с целью максимизации количества слов, активизируемых на на входе F-функции первой итерации цепи Фейстеля.

4.6. F-функция

F-функция обладает следующими свойствами:

- для ввода подключа итерации $K^{(i)}$ используется операция сложения по модулю 2^{64}, которая является несовместной с операцией XOR, используемой на выходе цепи Фейстеля;
- равномерное рассеивание активизации между словами полублока, за счёт применения P-перестановки;
- обеспечение уникальности отображения различных слов, составляющих полублок, за счёт применения уникальных SL_j-преобразований для каждого слова полублока.

4.7. Процедура разворачивания ключа

Схема «разворачивания ключа» обладает следующими свойствами:

- значения набора подключей итераций для прямого и обратного преобразований полностью совпадают, что позволяет использовать общий «развёрнутый» ключ как для процедуры зашифрования, так и расшифрования;
- на основании исходного пользовательского ключа переменной длины (2 – 4 полублока) формируется 8 полублоков развёрнутого ключа, каждый из которых используется на двух итерациях цепи Фейстеля;
- для IT и FT преобразований используются подключи, не используемые на первых и последних трёх итерациях цепи Фейстеля;
- IT преобразование и последующие 4 первых итерации цепи Фейстеля используют 6 различных подключей «развёрнутого ключа»;
- FT преобразование и предшествующие 5 последних итерации цепи Фейстеля используют 7 различных подключей «развёрнутого ключа»;
- как минимум половина (максимум 3/4) подключей в буфере развёрнутого ключа формируются с помощью нелинейной F-функции из исходного ключа пользователя, что в сочетании с правилом выбора подключей итераций, позволяет исключить появление слабых и полуслабых ключей, а также защититься от «слайд-атак»;
- на ключевой вход F-функции (процедуры «разворачивания ключа») подаётся рандомизирующая последовательность, сформированная с помощью простого рекуррентного генератора (функция δ) на основе исходного пользовательского ключа, при этом, все элементы рандомизирующей последовательности уникальны, а закон формирования не может быть выражен только операцией используемой для ввода ключа в F-функцию, т.е. векторным сложением по модулю 2^{64}.

4.8. Количество циклов шифрования

Алгоритм «Лабиринт» использует 16 итераций (8 циклов) цепи Фейстеля для всех поддерживаемых

длин блока. Ниже приведено обоснование достаточности этого количества для защиты от основных криптоаналитических атак – линейного и дифференциального криптоанализа.

Известно, что для цепи Фейстеля минимальное количество активных итераций достигается когда «пассивной» является каждая третья итерация [7], т.е. в этом случае количество активных итераций может быть рассчитано из соотношения:

$$a_{r'} = \begin{cases} 2t, & r' = 3t, 3t+1; \\ 2t+1, & r' = 3t+2, \end{cases}$$

где r' – количество итераций цепи Фейстеля, используемое шифром ($r' = 2r$).

Необходимым и достаточным условием достижения указанного минимального значения активизации F-функций является выполнение следующего соотношения на всех (кроме последней) итерациях (i) цепи Фейстеля:

$$\begin{cases} \Delta F_{K^{(i)}}(\Delta X) = \Delta Y; \\ \Delta F_{K^{(i+1)}}(\Delta Y) = \Delta X, ' \end{cases} \qquad (*)$$

где ΔX – некоторая характеристика на входе F-функции, вычисленная относительно операции XOR; ΔF – оператор вычисления ожидаемой трансформации входной характеристики в выходную, при прохождении F-функции.

Таким образом, минимальное количество активных итераций для алгоритма «Лабиринт» составляет $a_{r'} = 10$.

Вероятность заданного выполнения некоторой характеристики при прохождении через функцию шифрования с Фейстель-подобной структурой может быть оценена следующим образом:

$$p_E = p_S^{(a_{r'} \times a_S)},$$

где $a_{r'}$ – количество активных итераций шифрования; a_S – минимальное количество активных S-блоков на одной итерации шифрования, при выполнении условия (*); p_S – максимальная вероятность линейной аппроксимации (дифференциальной характеристики) для одного S-блока; p_E – вероятность заданного прохождения характеристики через функцию шифрования.

Отметим, что в силу использования в алгоритме начального (IT) и конечного (FT) преобразований, классические NR-атаки становятся не применимы к этому шифру, поэтому в приведенном выше соотношении не выполняется дополнительное уменьшение количества активных циклов. С другой стороны, так как нас интересует оценка верхней границы вероятности выполнения характеристики, в приведенном соотношении не учитывается понижение вероятности характеристики при прохождении S-блоков начального и конечного преобразований.

Для оценки минимального количества активных S-блоков a_S воспользуемся свойством «равномерности» перестановки P, а также свойством MBN-преобразования обеспечивать в пределах каждого слова минимум 9 ветвей активизации. Из указанных двух свойств следует, что на двух соседних активных итерациях активизируется не менее чем 8n S-блоков, или, иначе говоря, в среднем, не менее половины S-блоков на каждой активной итерации, т.е. не менее 4n на каждой итерации. Откуда можем получить оценку верхней границы вероятности прохождения характеристики через процедуру шифрования:

$$p_E = p_S^{(10\times 4n)} = \left(2^{-6}\right)^{40n} = 2^{-240n}.$$

Напомним, что пространство значений блоков текста составляет 2^{128n}, т.е. атаки дифференциального и линейного криптоанализа на шифр «Лабиринт» не могут быть, в принципе, построены на базе независимых одноблочных характеристик. Таким образом, алгоритм является устойчивым к атакам линейного и дифференциального криптоанализа, даже без учёта вероятностей прохождения через S-блоки начального и конечного преобразований.

Заключение

Алгоритм «Лабиринт» удовлетворяет требованиям первого (максимального) класса стойкости, в соответствии с условиями проекта NESSIE (длина блока не менее 128 бит, длина ключа не менее 256 бит). Структура алгоритма «Лабиринт» позволяет эффективно его реализовать как на 64-битных, так и на 32-битных микропроцессорах. А, учитывая последние достижения в сфере специализированных микроконтроллеров, а именно тенденцию перевода смарт-карт на 32-битные архитектуры и / или оснащение их PKI-криптоускорителем, данный алгоритм может быть эффективно реализован, в том числе, и на смарт-картах. С другой стороны, выбранная конструкция алгоритма, обеспечивает возможность эффективной аппаратной реализации, при этом возможно существенное снижение затрат энергонезависимой памяти, за счёт аппаратной реализации операции умножения в поле $GF(2^8)$, и снижение вычислительной сложности, за счёт «устранения» операций фиксированного сдвига. Кроме того, алгоритм построен на базе классической цепи Фейстеля, что обеспечивает инволютивность шифрующего преобразования и возможность использования общего аппаратного решения для процедур зашифрования и расшифрования.

Существенным достоинством предлагаемой конструкции шифра является возможность применения единожды «развёрнутого» рабочего ключа для выполнения обоих преобразований зашифрования и расшифрования, без дополнительных модификаций ключевого материала. Это оказывается весьма существенно когда на одном ключе необходимо выполнять оба преобразования и позволяет до двух раз сократить затраты оперативной памяти и вычислительные затраты на выполнение процедуры разворачивания ключа.

Конструкция циклового преобразования обеспечивает возможность увеличения длины обрабаты-

ваемого блока без необходимости увеличения количества циклов шифрования, а конструкция схемы разворачивания ключа обеспечивает независимость количества циклов шифрования от длины исходного ключа. Эти два свойства алгоритма «Лабиринт» позволяет получить практически постоянную производительность шифра при различных значениях показателей стойкости – длинах блока и ключа.

Возможность произвольного выбора параметров узла замены (S-блока) и линейного MBN-преобразования, в пределах строго определённых ограничений к их криптографическим показателям, позволяет, с одной стороны, устранить потенциальную опасность ввода «закладок» (или «потайных дверей») со стороны автора алгоритма, а с другой – обеспечивает возможность введения дополнительного секретного параметра, если параметры указанных элементов составляют долговременный секретный ключ. Такие долговременные ключи могут формироваться по специальной методике, обеспечивающей контроль, не только указанных в данной спецификации, минимальных критериев «фильтрации», но и ряда дополнительных. Формирование указанных долговременных ключей, для государственных организаций, может выполняться соответствующим уполномоченным органом и поставляться пользователям в установленном порядке.

Конструкция алгоритма «Лабиринт» учитывает известные методы криптоанализа БСШ и позволяет защититься от них. В основу алгоритма были положены хорошо изученные принципы построения шифров (ГОСТ-подобная структура и AES-подобный узел нелинейного усложнения и линейного смешивания), прошедшие апробацию временем, а также был учтён опыт использования сравнительно новых конструкций и принципов построения БСШ [10, 11, 12]. Их совместное применение позволило создать алгоритм, объединивший в себе достоинства каждого из «прародительских» алгоритмов, и при этом избавиться от присущих им недостатков. Поэтому, для успешного криптонападения на полученный алгоритм требуется поиск принципиально новых методов криптоанализа БСШ. С другой стороны, накопленный опыт и математический аппарат исследования стойкости алгоритмов ГОСТ 28147-89 [3] и AES [1] может быть применён для первичной оценки безопасности алгоритма «Лабиринт», что существенно упрощает эту задачу. Конструкция алгоритма построена в соответствии с принципом «”слабой” цикловой функции, повторяемой многократно», который, по мнению автора, продемонстрировал большую эффективность, по сравнению с принципом «”сильной” цикловой функции, повторяемой ограниченное число раз» [14].

Алгоритм «Лабиринт» может использоваться в любом из пяти стандартных режимов применения БСШ, а также в «усиленных» режимах поточного шифрования, предложенных автором [15]. Именно в «усиленных» режимах применения БСШ, сегодня, может быть востребовано использование больших длин блока (256 или даже 512 бит).

Список литературы

1. Federal Information Processing Standards Publication 197 (FIPS PUB 197). Specification for the Advanced Encryption Standard (AES). // Department of Commerce, National Institute of Standards and Technology, Information Technology Laboratory. November 26, 2001.

2. Federal Information Processing Standards Publication 46-3 (FIPS PUB 46-3). Specification for the Data Encryption Standard (DES). // U.S. Department of Commerce / National Institute of Standards and Technology. October 25, 1999.

3. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. – М.: Госстандарт СССР.

4. AES discussion forum [Электрон. ресурс]. – Режим доступа: http://aes.nist.gov.

5. NESSIE Call for Cryptographic Primitives, Version 2.2, 8th March 2000 [Электрон. ресурс]. – Режим доступа: http://cryptonessie.org/.

6. Головашич С.А. Методы построения высокостойких блочных симметричных шифров и схем их применения. // Дисс. канд. техн. наук. – Х.: Харьковский национальный университет радиоэлектроники, 2001. – 180 с.

7. Горбенко И.Д., Головашич С.А. Алгоритм блочного симметричного шифрования «Торнадо». Спецификация преобразования // Радиотехника: Всеукр. межвед. научн.-техн. сб. – Х.: ХНУРЭ, 2003. – Вып. 134. – С. 62-80.

8. Долгов В.И., Головашич С.А., Руженцев В.И. Криптостойкость шифра «Торнадо». // Радиотехника: Всеукр. межвед. научн.-техн. сб. – Х.: ХНУРЭ, 2003. – Вып. 134. – С. 81-88.

9. Головашич С.А., Лепеха А.И. Статистический анализ БСШ «Торнадо» // Радиотехника: Всеукр. межвед. научн.-техн. сб. – Х.: ХНУРЭ, 2003. – Вып. 134. – С. 89-96.

10. Головашич С.А. Принцип построения инволютивных шифров // Проблемы бионики. – 2001. – Вып. 54. – С. 118-125.

11. Головашич С.А. Метод конструирования цикловых функций БСШ // Автоматизированные системы управления и приборы автоматики: Всеукр. межвед. научн.-техн. сб. – Х.: ХНУРЭ, 2001. – Вып. 117. – С. 155-161.

12. Головашич С.А. Метод построения управляемых S-блоков с предельными показателями нелинейности // Радиотехника: Всеукр. межвед. научн.-техн. сб. – Х.: ХНУРЭ, 2001. – Вып. 123. – С. 215-221.

13. NIST Special Publication 800-38A 2001 Edition – Recommendation for Block Cipher Modes of Operation. Methods and Techniques // Computer Security Division Information Technology Laboratory National Institute of Standards and Technology, Gaithersburg, MD 20899-8930. December 2001.

14. «Supporting Document on E2», Nippon Telegraph and Telephone Corporation, June 14, 1998.

15. Головашич С.А. Безопасность режимов блочного шифрования // Радиотехника: Всеукр. межвед. научн.-техн. сб. – Х.: ХНУРЭ, 2001. – Вып. 119. – С. 135-145.

Поступила в редколлегию 19.04.2007

Рецензент: д-р. техн. наук, проф. О.Г. Руденко, Харьковский национальный университет радиоэлектроники, Харьков.