# ПЕРСПЕКТИВНЫЕ БЛОЧНЫЕ СИММЕТРИЧНЫЕ ШИФРЫ И ИХ СВОЙСТВА

УДК 681.3.06:519.248.681

*И. Д. ГОРБЕНКО, д-р техн. наук, С. А. ГОЛОВАШИЧ, канд. техн. наук*

## АЛГОРИТМ БЛОЧНОГО СИММЕТРИЧНОГО ШИФРОВАНИЯ «ТОРНАДО». СПЕЦИФИКАЦИЯ ПРЕОБРАЗОВАНИЯ

На протяжении последнего десятилетия в нашей стране проходили процессы интенсивной интеграции в мировое информационное сообщество. Следствием этих процессов стало, с одной стороны, повсеместное внедрение средств вычислительной техники, а с другой – массовое использование сети Internet, как наиболее доступного средства коммуникаций и доступа к неограниченным информационным ресурсам. Сформировавшаяся за это время инфраструктура вычислительных систем использует вычислительную технику зарубежного производства и базируется на открытых каналах телекоммуникаций. При этом информация, обрабатываемая в этих системах, часто носит конфиденциальный характер и требует применения средств криптографической защиты. Неотъемлемым элементом криптографической защиты информации в компьютерных системах стали блочные симметричные шифры (БСШ).

В настоящее время в Украине отсутствует национальный стандарт БСШ, и временно разрешён к применению стандарт бывшего СССР ГОСТ 28147-89, принятый в 1989 году. Однако, учитывая значительный прогресс в сфере методов и средств криптоанализа, на протяжении последних 10–15 лет этот алгоритм уже переходит в класс "морально" устаревших. Косвенным подтверждением этому факту является проведение в США и Европе конкурсов (проекты AES [1] и NESSIE [2] соответственно), направленных на выбор новых криптопримитивов, удовлетворяющих возросшим требованиям безопасности. Так, в соответствии с требованиями проекта NESSIE, алгоритм ГОСТ 28147-89 относится 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 «Торнадо» было решение этой проблемы.

Предлагаемый алгоритм является итеративным «инволютивным» шифром [3] и предусматривает три варианта длины блока (для применения в различных классах безопасности) и переменную длину ключа шифрования для каждого варианта длины блока.

Структура алгоритма «Торнадо» идентична для различных длин блока, отличие заключается только в необходимом числе циклов шифрования. Поэтому описание алгоритма приводится в общем виде для блока длиной **128*n*** бит, где ***n*** – *коэффициент кратности длины блока*, который может принимать значения **1**, **2** или **4**, то есть алгоритм поддерживает блоки длиной **128**, **256** и **512** бит. Для каждой длины блока алгоритм поддерживает 4 значения длины ключа в диапазоне $256n \div 448n$ с шагом $64n$ бита (то есть значения $(4 \div 7) \times 64n$). Отметим, что в данной статье приводится версия 3.0 алгоритма «Торнадо».

## 1 Обозначения и соглашения

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($p$));

$\mathbf{B}$ – векторное пространство 8-битных элементов (байтов), $\mathbf{B} = \mathrm{GF}(2)^8 = \{0,1\}^8$;
$\mathbf{W}$ – векторное пространство 64-битных элементов (слов), $\mathbf{W} = \mathbf{B}^8 = \mathrm{GF}(2)^{64} = \{0,1\}^{64}$;
$\mathbf{H}$ – векторное пространство $64n$-битных элементов (полублоков), $\mathbf{H} = \mathbf{W}^n = \mathrm{GF}(2)^{64n} = \{0,1\}^{64n}$;
$\mathbf{T}$ – векторное пространство $128n$-битных элементов (блоков), $\mathbf{T} = \mathbf{H}^2 = \mathrm{GF}(2)^{128n} = \{0,1\}^{128n}$;
$\mathbf{P}$ – множество $2^{12}$ «равновероятных» перестановок из 8 элементов. Элементы множества кодируются 12-битными векторами, то есть $\{0,1\}^{12}$. Правило формирования множества приведено в пп. 3.7;
$\mathbf{P}'$ – подмножество перестановок множества $\mathbf{P}$ размерностью $2^8$. Элементы множества кодируются 8-битными векторами, то есть $\{0,1\}^8$. Правило формирования подмножества приведено в пп. 3.7;

$\oplus$ – побитовая операция «исключающее ИЛИ» (XOR, сложение по модулю 2);
$\wedge$ – побитовая операция логическое «И» (AND);
$\vee$ – побитовая операция логическое «ИЛИ» (OR);
$\overline{x}$ – операция побитового инвертирования («НЕ», NOT);
$\times$ – операция умножения матрицы на вектор (элементы принадлежат полю $\mathrm{GF}(2^8)$);
$+$ – сложение в кольце $\mathbf{Z}_{2^m}$;
$-$ – вычитание в кольце $\mathbf{Z}_{2^m}$;
$<< l$ – логический сдвиг на $l$ бит влево;
$>> l$ – логический сдвиг на $l$ бит вправо;
$<<< l$ – циклический сдвиг на $l$ бит влево;
$>>> l$ – циклический сдвиг на $l$ бит вправо;
$\mathrm{div}\ m$ – целая часть от деления на $m$;
$\mathrm{Mod}\ m$ – остаток от деления на $m$;
$x \| y$ – конкатенация операндов $x$ и $y$.

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

$(x_0, \ldots, x_m)$ – способ записи множества либо последовательности элементов, для которых порядок взаимного расположения в памяти шифратора значения не имеет (с точки зрения применяемых операций);
$\langle x_m \| \ldots \| x_0 \rangle$ – 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, то есть нумерация разрядов, байтов, слов и так далее выполняется от младших «весов» к старшим: разряд $b_i$ имеет «вес» $2^i$, а байт $B_i$ – «вес» $2^{8i}$ и так далее.

Например:

$X \in \mathbf{H}=\mathbf{W}^4$ $(n = 4)$, $W_i \in \mathbf{W}$, $B_i \in \mathbf{B}$, $b_i \in \mathrm{GF}(2) \Rightarrow$

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

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

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

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

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

$X$ = 0x123456789FFEEDDCCBBAA99887766554433221100 $\Rightarrow$

$W_0$ = 0x7766554433221100, $W_1$ = 0xFFEEDDCCBBAA9988,

$B_0$ = 0x00, $B_1$ = 0x11, $B_{15}$ = 0xFF.

Символы операций $\oplus$, +, –, <<, >>, <<<, >>>, заключённые в квадратные скобки (например [+]), будем использовать для обозначения операций, выполняемых над словами (64 бита).

Далее будет использоваться следующая нотация:

- верхний индекс (например, $\mathrm{Fun}^n$) – 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 номер 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, то есть $128n$ бит.

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

*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» для всех элементов вектора-аргумента.

*Циклической MDS–матрицей* будем называть квадратную матрицу размерностью 8×8 (с элементами из поля $\mathrm{GF}(2^8)$), все строки которой получены циклическим сдвигом полинома 7-й степени над полем $\mathrm{GF}(2^8)$, образующего *циклический разделимый код максимального расстояния* (*циклический МДР- или MDS-код*).

*Числом ветвей активизации* линейного преобразования будем называть минимальное суммарное количество активных S-блоков на входе и выходе этого преобразования при условии фактической активизации входа.

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

- чётные и нечётные итерации имеют незначительное отличие, в то время как пары соседних итераций полностью идентичны;
- изменение каждого бита блока данных возможно после применения не менее чем 2 итераций цепи Фейстеля.

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

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

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

## 2 Элементарные преобразования алгоритма «Торнадо»

Все элементарные преобразования в алгоритме «Торнадо» имеют векторную структуру, то есть 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 элементарных операций, выполняемых над словами, блоками либо полублоками, то есть 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.

$x\,[\oplus]^n\,y$ – сложение по модулю 2 двух векторов $x$ и $y$ длиной по $n$ слов;

$x\,[+]^n\,y$ – сложение по модулю $2^{64}$ одноимённых слов, составляющих векторы $x$ и $y$ (длиной по $n$ слов);

$x\,[-]^n\,y$ – вычитание по модулю $2^{64}$ слов, составляющих вектор $y$ из одноимённых слов, составляющих вектор $x$ (длина по $n$ слов);

$x\,[<<]^n\,l$ – логический сдвиг каждого из $n$ слов, составляющих вектор $x$, на $l$ бит влево ($0 \le l \le 63$);

$x\,[<<<]^n\,l$ – циклический сдвиг каждого из $n$ слов, составляющих вектор $x$, на $l$ бит влево ($0 \le l \le 63$);

$x\,[>>>]^n\,l$ – циклический сдвиг каждого из $n$ 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:

$$Y = \mathrm{Fun}^n(X, K).$$

Подобная запись означает, что входной ($X$) и выходной ($Y$) векторы данных состоят из $n$ слов, а вектор управляющего сигнала $K$ (для управляемых отображений) состоит из $n$ 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: $y = \mathrm{S}_{\nu}(x)$; $x, y \in \mathbf{B}$.

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

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

Наиболее эффективно преобразования этого класса реализуются табличным способом. В этом случае реализация S-блока может быть эффективно совмещена с реализацией последующего фиксированного преобразования, то есть оба преобразования могут быть выполнены с помощью одной общей таблицы.

В алгоритме используется три различных S-блока 8 в 8 бит (обозначены $\mathrm{S}_0$, $\mathrm{S}_1$, $\mathrm{S}_2$), которые выбираются из множества, так называемых, предельно-нелинейных биективных преобразований [4]. Они строятся на основе конструкции Ниберг-Динга, то есть являются аффинно-эквивалентными функции вычисления обратного элемента в поле $\mathrm{GF}(2^8)$:

$$\mathrm{S}(X) = \mathbf{M}_{\mathrm{y}} \times \left[ \left( \mathbf{M}_{\mathbf{x}} \times X \oplus \mathbf{V}_{\mathbf{x}} \right)^{2^8 - 2} \right]_B \oplus \mathbf{V}_{\mathrm{y}}; \quad X, \mathbf{V}_{\mathbf{x}}, \mathbf{V}_{\mathrm{y}} \in \mathbf{F}_2^8; \quad \mathbf{M}_{\mathbf{x}}, \mathbf{M}_{\mathrm{y}} \in \mathbf{GL}\left(8, \mathbf{F}_2\right),$$

где $B$ – некоторый базис над $\mathrm{GF}(2^8)$, определяемый образующим (неприводимым) полиномом;

$\mathrm{M_X}$, $\mathrm{M_Y}$ – квадратные невырожденные матрицы размерностью (8×8).

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

В данной спецификации не приводятся конкретные значения $B$, $\mathrm{M_X}$, $\mathrm{M_Y}$, $\mathrm{V_X}$, $\mathrm{V_Y}$, что позволяет использовать S-блоки в качестве долговременного секретного ключа шифрования. Применение указанной конструкции позволяет достигнуть предельно низких значений вероятностей заданного трансформирования дифференциальной разности ($p_{\mathrm{S}}^{\mathrm{DC}} = 2^{-6}$) и линейной аппроксимации ($p_{\mathrm{S}}^{\mathrm{LC}} = 2^{-6}$) S-блока, а также получить максимальную алгебраическую степень булевого полинома ($\deg_y = 7$). Кроме того, выбор входного ($\mathrm{M_X}$) и выходного ($\mathrm{M_Y}$) аффинных преобразований может осуществляться на основе дополнительных требований к S-блокам [4].

## 3 Структура алгоритма «Торнадо»

### *3.1 Преобразования зашифрования и расшифрования*

Алгоритм «Торнадо» является «инволютивным» шифром, то есть 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 $K^{\mathrm{IT}}$ и $K^{\mathrm{FT}}$ соответственно).

Преобразования зашифрование / расшифрования состоит из двух фаз:

1. Процедура разворачивания ключа KShed на основе исходного ключа *UK* (с учётом направления преобразования – зашифрование или расшифрование (e\d)) выполняет формирование рабочего ключа *WK*.
2. Процедура шифрования EF выполняет преобразование входного блока данных (*P* либо *C*) на рабочем ключе *WK*.

$$WK = \mathrm{KShed}(UK, mode = \mathrm{e}\backslash \mathrm{d}):$$
$$C = \mathrm{E}_{UK}(P) = \mathrm{EF}_{WK_{\mathrm{E}}}(P),$$
$$P = \mathrm{D}_{UK}(C) = \mathrm{EF}_{WK_{\mathrm{D}}}(C); \quad P, C \in \mathbf{T};$$
$$WK_{\mathrm{E}} = \left(K^{\mathrm{IT}}, K^{(0)}, K^{(1)}, \ldots, K^{(2r-1)}, K^{\mathrm{FT}}\right),$$
$$WK_{\mathrm{D}} = \left(K^{\mathrm{FT}}, K^{(2r-1)}, K^{(2r-2)}, \ldots, K^{(0)}, K^{\mathrm{IT}}\right),$$

где KShed – процедура разворачивания ключа;
*mode* – режим преобразования: зашифрование (e) / расшифрование (d);
$UK$ – исходный (пользовательский) ключ;
$WK_{\mathrm{E}}$ – рабочий ключ зашифрования;
$WK_{\mathrm{D}}$ – рабочий ключ расшифрования;
$K^{(i)}$ – ключ для $i$-й итерации;
$\mathrm{E}_k$ – преобразование зашифрования на исходном ключе k;
$\mathrm{D}_k$ – преобразование расшифрования на исходном ключе k;
$\mathrm{EF}_k$ – процедура шифрования на рабочем ключе k;
$P$ – блок входного («открытого») текста;
$C$ – блок выходного текста («криптограммы»).

### *3.2 Процедура шифрования* EF

Алгоритм «Торнадо» является итеративным шифром, то есть основу его процедуры шифрования составляет цикловое преобразование, которое повторяется заданное число раз (обозначим число циклов шифрования символом $r$). Кроме повторяемого циклового преобразования процедура шифрования включает начальное (IT) и конечное (FT) преобразования. Свойство инволютивности шифра достигается за счёт применения обобщённой конструкции полублоковой цепи Фейстеля. Структура процедуры шифрования EF приведена на рис. 1.

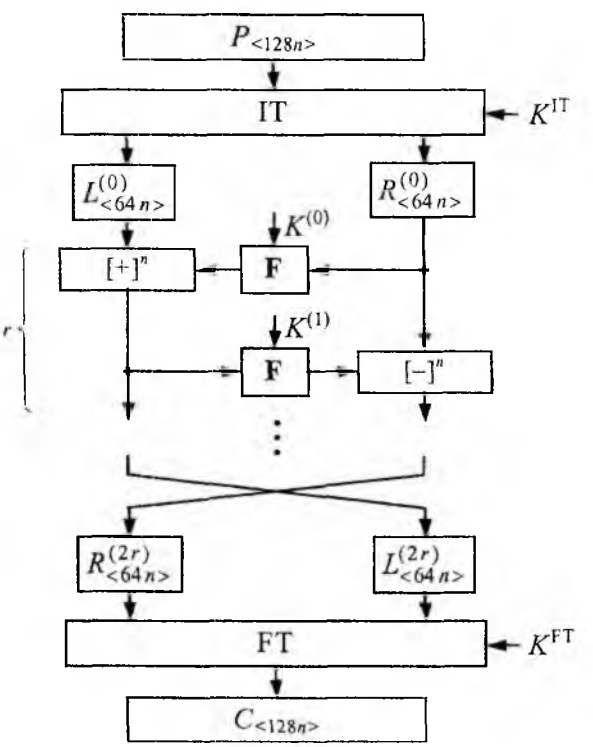


Рис. 1

Количество циклов шифрования ($r$) определяется длиной блока текста, соответствующие значения приведены в табл. 1. Выбранные значения рассчитаны в соответствии с практическим критерием оценки стойкости БСШ [5].

Таблица 1

| Длина блока $l_Б$, бит | Коэффициент $n$ | Число циклов $r$ |
|---|---|---|
| 128 | 1 | 4 |
| 256 | 2 | 6 |
| 512 | 4 | 10 |

Процедура шифрования EF состоит из трёх шагов:

1. Исходный блок данных $P$ (длиной $128n$ бит) обрабатывается начальным (IT) преобразованием на ключе $K^{IT}$.
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 $r$ циклах, каждый из которых состоит из двух итераций. Обе итерации используют общее, управляемое подключом $K^{(i)}$, нелинейное преобразование – F-функцию. Единственное отличие этих двух итераций заключается в том, что на первой итерации выход F-функции объединяется с левым полублоком операцией сложения по модулю $2^{64}$ (обозначение [+]), а на второй итерации – операцией вычитания по модулю $2^{64}$ (обозначение [–]).
3. Два полублока $L$ и $R$, полученные в результате $r$-циклового итеративного преобразования, меняются местами и обрабатываются конечным (FT) преобразованием на ключе $K^{FT}$. Полученный после преобразования двоичный вектор $C$ (длиной $128n$ бит) является результирующим блоком («криптограммой»).

Аналитически процедура EF может быть представлена следующим образом:

$$C = \mathrm{EF}_{WK}(P):$$

$$\left\langle L^{(0)} \parallel R^{(0)} \right\rangle = \mathrm{IT}\left(P, K^{\mathrm{IT}}\right);$$

$$\begin{matrix} i = 2j, \\ j = \overline{0, r-1} \end{matrix}: \quad \begin{cases} L^{(i+1)} = L^{(i)} \,[+]^n\, \mathrm{F}\left(R^{(i)}, K^{(i)}\right), & R^{(i+1)} = R^{(i)} \\ L^{(i+2)} = L^{(i+1)}, & R^{(i+2)} = R^{(i+1)} \,[-]^n\, \mathrm{F}\left(L^{(i+1)}, K^{(i+1)}\right) \end{cases};$$

$$C = \mathrm{FT}\left(\left\langle R^{(2r)} \parallel L^{(2r)} \right\rangle, K^{\mathrm{FT}}\right); \quad P, C \in \mathbf{T}; \quad L^{(i)}, R^{(i)} \in \mathbf{H}.$$

В приведенных выше соотношениях символом $i$ обозначается номер итерации, а символом $j$ – номер цикла (в обоих случаях нумерация выполняется с 0).

На каждой итерации F-функция использует $148n$-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 «Торнадо» построена в соответствии с принципами, позволяющими обеспечить свойства рассеивания и размножения активизации [6].

F-функция условно может быть разбита на четыре преобразования:

- сложение полублока с подключом по модулю 2;
- нелинейное ключезависимое «смешивающее» преобразование SCL;
- нелинейное ключезависимое «рассеивающее» преобразование SCP;
- фиксированная байтовая перестановка $P_{byte}$.

Подробно эти преобразования будут рассмотрены ниже.

F-функция имеет следующее аналитическое представление:

$$Y = \mathrm{F}\left(X, K^{(i)}\right) = \mathrm{SCP}^n\left(\mathrm{P}_{\mathrm{byte}}\left(\mathrm{SCL}^n\left(X \oplus xk_1^{(i)}, zk_2^{(i)}\right) \oplus xk_2^{(i)}\right), zk_2^{(i)}\right);$$

$$K^{(i)} = \left(xk_1^{(i)}, zk_1^{(i)}, xk_2^{(i)}, zk_2^{(i)}\right); \quad X, Y, xk_1^{(i)}, xk_2^{(i)} \in \mathbf{H}; \quad zk_1^{(i)} \in \mathbf{P}'^n; \quad zk_2^{(i)} \in \mathbf{P}^n.$$

В общем виде F-функция может быть представлена схемой, приведенной на рис. 2. На одной итерации F-функция использует подключ длиной $148n$ бит, который состоит из трёх составляющих:

- $xk_1^{(i)}$ и $xk_2^{(i)}$ – ключи сложения по модулю 2, длиной по $64n$ бита каждый;
- $zk_1^{(i)}$ и $zk_2^{(i)}$ – ключи байтовой перестановки (пространство значений $2^{8n}$ и $2^{12n}$ соответственно).

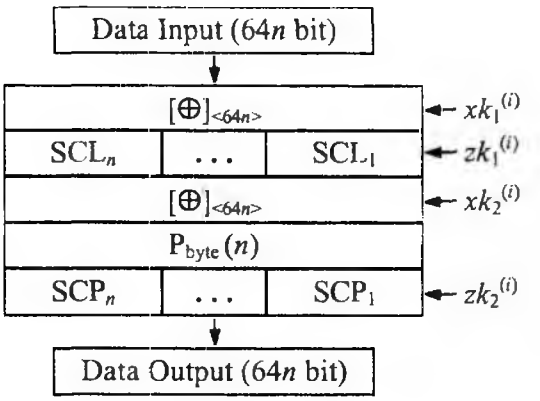


Рис. 2

Частный случай F-функции, когда $n = 1$ (то есть длина полублока составляет 64 бита), представлен на рис. 3. В этом случае ($n = 1$) преобразование $\mathrm{P}_{\mathrm{byte}}$, как будет показано ниже, является тождественным, поэтому на рисунке оно отсутствует.

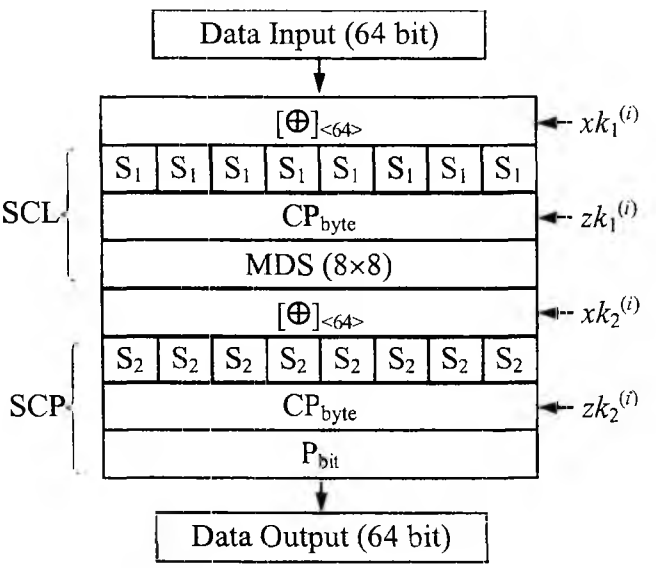


Рис. 3

При выполнении векторных преобразований $\mathrm{SLM}^n$ и $\mathrm{SCP}^n$ полублок данных рассматривается как последовательность из $n$ слов (по 64 бита), к каждому из которых независимо применяется соответствующее преобразование. Аналогичным образом подключ, используемый для «управления» векторным преобразованием, разбивается на $n$ последовательных кортежей равной длины:

$$X, Y \in \mathbf{H}; \quad zk \in \mathbf{P}^n; \quad W_j, W'_j \in \mathbf{W}; \quad zw_j \in \mathbf{P}; \quad j = \overline{0, n-1};$$
$$X = \langle W_{n-1} \| \ldots \| W_1 \| W_0 \rangle,$$
$$Y = \langle W'_{n-1} \| \ldots \| W'_1 \| W'_0 \rangle,$$
$$zk = \langle zw_{n-1} \| \ldots \| zw_1 \| zw_0 \rangle;$$
$$Y = \mathrm{SCL}^n(X, zk) \quad \Rightarrow \quad W'_j = \mathrm{SCL}(W_j, zw_j),$$
$$Y = \mathrm{SCP}^n(X, zk) \quad \Rightarrow \quad W'_j = \mathrm{SCP}(W_j, zw_j).$$

### *3.4 Операция обмена битов между двумя полублоками* $\mathbf{CX}_{\mathbf{bit}}$

В составе начального (IT) и конечного (FT) преобразований используется операция *управляемого «обмена» одноимёнными битами* ($CX_{bit}$) между левым ($L$) и правым ($R$) полублоками. Под одноимёнными понимаются разряды, расположенные в идентичных позициях полублоков-аргументов. Подмножество битов, подлежащих перестановке, определяется единичными битами управляющей маски $X$ (длиной один полублок). В алгоритме «Торнадо» используются *управляющие маски* только специального вида – маска всегда получается путём тиражирования $32n$ раза некоторого 4-битного кортежа, поэтому мы будем использовать сокращённую форму записи (например, $X = 0x55...55$ ).

Преобразование $CX_{bit}$ может быть представлено следующим образом:

$$\mathrm{CX}_{\mathrm{bit}}(\langle L \| R \rangle, X) = \langle (((L \oplus R) \wedge X) \oplus L \| (((L \oplus R) \wedge X) \oplus R \rangle; \quad L, R, X \in \mathbf{H}.$$

### *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 (то есть не зависящие от ключа) составляющие преобразований IT и FT являются взаимно обратными и включают два «слоя»:

- нелинейная подстановка («S–слой»);
- битовая перестановка («P–слой»).

«P–слой» обоих преобразований удобно рассматривать как последовательность из двух операций:

- циклический сдвиг каждого слова блока влево либо вправо на ($32n + 4$) разряда;
- операция «обмена» одноимёнными битами между полублоками ($CX_{bit}$).

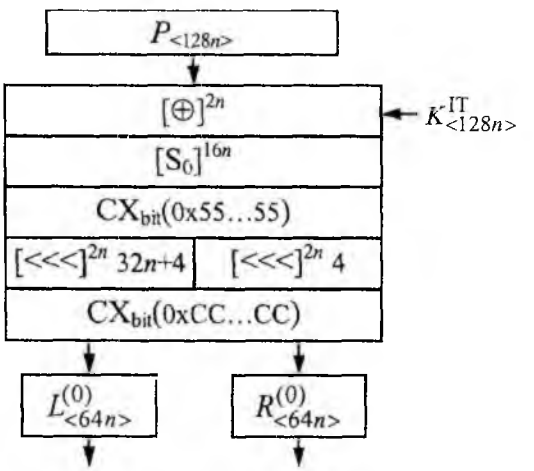


Рис. 4

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

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

$$1)\ \langle L' \| R' \rangle == [\mathrm{S}_0]^{16n}\left(P\ [\oplus]^{2n}\ K^{\mathrm{IT}}\right);$$

$$2)\ \langle L'' \| R'' \rangle = \mathrm{CX}_{\mathrm{bit}}\left(\langle L' \| R' \rangle, 0\mathrm{x}55...55\right);$$

$$3)\ \langle L''' \| R''' \rangle = \left\langle \left(L''\ [<<<]^{2n}\ (32n+4)\right) \| \left(R''\ [<<<]^{2n}\ 4\right) \right\rangle;$$

$$4)\ \left\langle L^{(0)} \| R^{(0)} \right\rangle = \mathrm{CX}_{\mathrm{bit}}\left(\langle L''' \| R''' \rangle, 0\mathrm{xCC}...\mathrm{CC}\right),$$

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

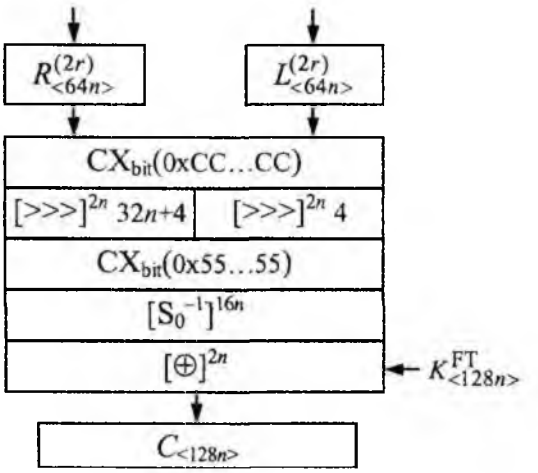


Рис. 5

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

$$C = \mathrm{FT}\left(\left\langle L^{(2r)} \| R^{(2r)} \right\rangle, K^{\mathrm{FT}}\right): \quad L, R \in \mathbf{H}; \quad C, K^{\mathrm{FT}} \in \mathbf{T};$$

$$1)\ \langle L' \| R' \rangle = \mathrm{CX}_{\mathrm{bit}}\left(\left\langle R^{(2r)} \| L^{(2r)} \right\rangle, 0\mathrm{xCC}...\mathrm{CC}\right);$$

$$2)\ \langle L'' \| R'' \rangle = \left\langle \left(L'\ [>>>]^{2n}\ (32n+4)\right) \| \left(R'\ [>>>]^{2n}\ 4\right) \right\rangle;$$

$$3)\ \langle L''' \| R''' \rangle = \mathrm{CX}_{\mathrm{bit}}\left(\langle L'' \| R'' \rangle, 0\mathrm{x}55...55\right);$$

$$4)\ C = [\mathrm{S}_0^{-1}]^{16n}\left(\langle L''' \| R''' \rangle\right)\ [\oplus]^{2n}\ K^{\mathrm{FT}},$$

где $C$ – результирующий блок текста на выходе шифратора («криптограмма»);
$L^{(2r)}$, $R^{(2r)}$ – соответственно левый и правый полублоки на входе FT-преобразования.

### *3.6 Фиксированная перестановка байтов* $\mathbf{P_{byte}(n)}$

Вид байтовой перестановки $\mathrm{P}_{\mathrm{byte}}(n)$ определяется количеством 64-битных слов в полублоке (то есть коэффициентом $n$). Байтовая перестановка выполняется в соответствии со следующим соотношением:

$$\mathrm{P}_{\mathrm{byte}}(n):\quad B'_i = B_{n\times(i \bmod 8)+(i \operatorname{div} 8)},\quad B_i \in \mathbf{B},\quad i = \overline{0, 8n-1},$$

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

На рис. 6 – 8 показаны схемы перестановки байтов ($\mathrm{P}_{\mathrm{byte}}$) для всех допустимых значений $n$ (то есть 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.

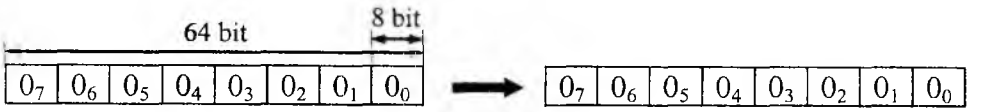


Рис. 6 (случай $n = 1$)

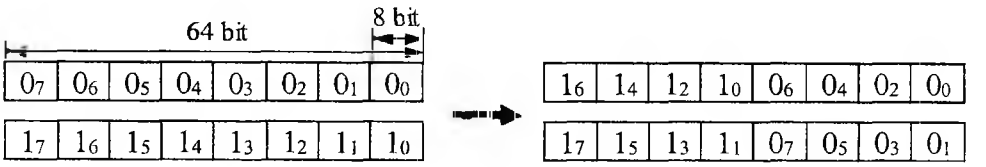


Рис. 7 (случай $n = 2$)

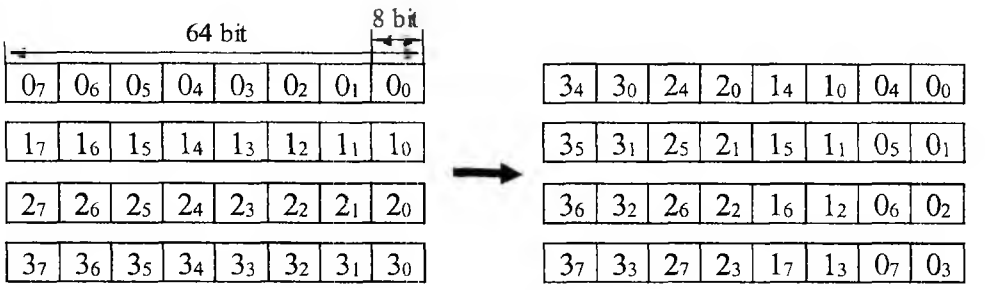


Рис. 8 (случай $n = 4$)

### *3.7 Функции $\alpha$ и $\beta$*

Как было показано выше, F-функция алгоритма «Торнадо», кроме операции "сложения по модулю 2" (для ввода циклового ключа), использует управляемую перестановку байтов внутри слова (преобразование $\mathrm{CP}_{\mathrm{byte}}$), то есть "операцию" перестановки на множестве из 8 элементов, соответствующих позициям байтов внутри слова. При этом размерность пространства допустимых перестановок составляет $2^{12}$. Перестановки, принадлежащие этому пространству, будем обозначать $\pi_z$, где $z$ – индекс или ключ, выбирающий одну из допустимых перестановок.

Ключ перестановки удобно рассматривать как три кортежа по 4 бита:

$$z = \langle k4 \| k2 \| k1 \rangle,\quad z \in \{0,1\}^{12} = \mathbf{P};\quad k1, k2, k4 \in \{0,1\}^4;$$

$$z' = \langle k2 \| k1 \rangle,\quad z' \in \{0,1\}^8 = \mathbf{P}',\quad k4 = 0.$$

Перестановка $\pi_z$ может быть представлена композицией трёх "элементарных" управляемых перестановок $\sigma_1, \sigma_2, \sigma_4$:

$$\sigma_t = \begin{pmatrix} a_7 & \dots & a_1 & a_0 \\ a'_7 & \dots & a'_1 & a'_0 \end{pmatrix}.$$

Каждая из этих "элементарных" перестановок представляет собой композицию четырёх независимых транспозиций $\tau$, управляемых одним битом ключа. Отдельную управляемую транспозицию обозначим:

$$\tau(i,j,k) = \begin{cases} a'_i \leftarrow a_i,\ a'_j \leftarrow a_j, & \text{если } k = 0 \\ a'_i \leftarrow a_j,\ a'_j \leftarrow a_i, & \text{если } k = 1 \end{cases}, \quad k \in \{0,1\},$$

где $i, j$ – позиции двух элементов, подлежащих "управляемой транспозиции";
$k$ – бит ключа, определяющий, выполнять ($k = 1$) или не выполнять ($k = 0$) транспозицию.

Каждая из перестановок $\sigma_1, \sigma_2, \sigma_4$ в качестве ключа использует соответствующий 4-битный кортеж:

$\sigma_1$: $\left\{ i = \overline{0,3}:\ \tau(2i+0,\ 2i+1,\ k1_i) \right\}$;

$\sigma_2$: $\left\{ i = \overline{0,3}:\ \tau(j+0,\ j+2,\ k2_i),\quad j = 4 \times (i \operatorname{div} 2) + (i \bmod 2) \right\}$;

$\sigma_4$: $\left\{ i = \overline{0,3}:\ \tau(i+0,\ i+4,\ k4_i) \right\}$,

где $kt_i$ – бит $i$ ключевого кортежа $kt$ (t = 1, 2, 4).

Наглядное представление перестановки $\pi_z$ приведено на рис. 9.

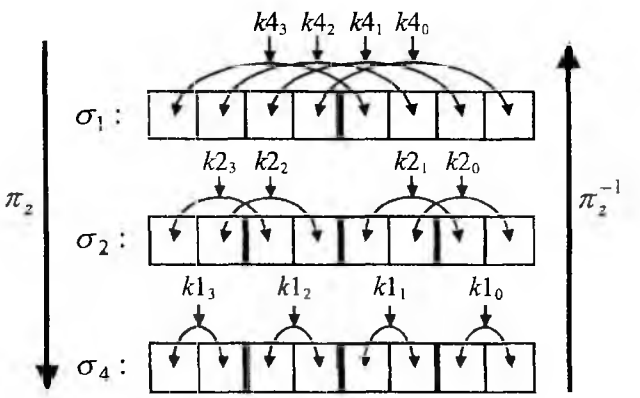


Рис. 9

Для удобства последующего описания преобразования $CP_{byte}$, на основе рассмотренных управляемых перестановок $\sigma_1, \sigma_2, \sigma_4$, определим две функции – $\alpha$ и $\beta$. Эти функции будут выполнять формирование соответственно прямой $\pi_z$ и обратной $\pi_z^{-1}$ перестановок на множестве чисел {0, 1, … , 7} по заданному 12 (или 8) –битному ключу $z$ (или $z'$).

Функция $\alpha$: $(zd_0, zd_1, \dots, zd_7) \leftarrow \alpha(z),\quad 0 \le zd_j \le 7;$

$$zd_i = \pi_z(i),\quad i = \overline{0,7};\quad \pi_z = \sigma_4 \circ \sigma_2 \circ \sigma_1,$$

где $i$ – позиция (индекс) байта-источника;
$zd_i$ – позиция (индекс) байта-получателя для "источника" $i$.

Функция $\beta$:
$$(zs_0, zs_1, \ldots, zs_7) \leftarrow \beta(z), \quad 0 \le zs_j \le 7;$$
$$zs_i = \pi_z^{-1}(i), \quad i = \overline{0,7}; \quad \pi_z^{-1} = \sigma_1 \circ \sigma_2 \circ \sigma_4,$$

где $i$ – позиция (индекс) байта-получателя;
$zs_i$ – позиция (индекс) байта-источника для "получателя" $i$.

### *3.8 Управляемая перестановка байтов* $\mathbf{CP}_{\text{byte}}$

Как было отмечено выше, преобразование $\text{CP}_{\text{byte}}$ выполняет управляемую ключом перестановку байтов внутри слова. В качестве управляющей информации данное преобразование использует 12-битный ключ перестановки $zw$. В соответствии с функциями $\alpha(zw)$ либо $\beta(zw)$ этот ключ «разворачивается» в последовательность из 8 кортежей по 3 бита, каждый из которых определяет позицию соответственно байта-получателя или байта-источника.

$$\text{CP}_{\text{byte}}: \quad \mathbf{W} \times \mathbf{P} \to \mathbf{W}$$
$$Y = \text{CP}_{\text{byte}}(X, zw),$$
$$X = \langle B_7 \| \ldots \| B_1 \| B_0 \rangle, \quad Y = \langle B'_7 \| \ldots \| B'_1 \| B'_0 \rangle,$$
$$B_j, B'_j \in \mathbf{B}; \quad X, Y \in \mathbf{W}; \quad zw \in \mathbf{P}.$$

Перестановка может быть выполнена двумя способами:

Вариант 1:
$$zw \xrightarrow{\alpha} (zd_0, zd_1, \ldots, zd_7), \quad 0 \le zd_j \le 7;$$
$$j = \overline{0,7}: \quad B'_{zd_j} = B_j.$$

Вариант 2:
$$zw \xrightarrow{\beta} (zs_0, zs_1, \ldots, zs_7), \quad 0 \le zs_j \le 7;$$
$$j = \overline{0,7}: \quad B'_j = B_{zs_j}.$$

### *3.9 «Расширяющая» битовая перестановка* $\mathbf{P}_{1\text{bit}}$

Перестановка $\text{P}_{1\text{bit}}$ является вспомогательной и приводится только для демонстрации одного из способов эффективной реализации шифра.

$$\text{P}_{1\text{bit}}: \quad \mathbf{B} \to \mathbf{W}$$
$$\langle B'_7 \| \ldots \| B'_1 \| B'_0 \rangle = \text{P}_{\text{bit}}(\langle b_7 \| \ldots \| b_1 \| b_0 \rangle),$$
$$B'_j = \langle 0 \| \ldots \| 0 \| b_j \rangle, \quad b_j \in \{0,1\}, \quad B'_j \in \mathbf{B}, \quad j = \overline{0,7}.$$

### *3.10 «Равномерная» битовая перестановка* $\mathbf{P}_{\text{bit}}$

Преобразование $\text{P}_{\text{bit}}$ определяет фиксированную перестановку битов внутри слова. Перестановка имеет следующий вид:

$$\mathrm{P_{bit}}: \quad \mathbf{W} \to \mathbf{W}$$
$$Y = \mathrm{P_{bit}}(X); \quad X, Y \in \mathbf{W};$$
$$X = \langle b_{63} \| \ldots \| b_1 \| b_0 \rangle, \quad Y = \langle b'_{63} \| \ldots \| b'_1 \| b'_0 \rangle;$$
$$b'_{8\times j+i} = b_{8\times i+j}; \quad i, j = \overline{0,7}; \quad b_t, b'_t \in \{0,1\}.$$

Эквивалентно преобразование $\mathrm{P_{bit}}$ может быть представлено с помощью «расширяющей» перестановки $\mathrm{P_{1bit}}$:

$$Y = \bigvee_{j=0}^{7} \left(\mathrm{P_{1\,bit}}(B_j) << j\right).$$

### *3.11 Умножение байта на MDS-код специального вида*

Это преобразование определяет операцию умножения элемента поля $GF(2^8)$, соответствующего байту-аргументу, на фиксированный полином 7 степени с коэффициентами из $GF(2^8)$. Этот полином должен формировать циклический код максимального расстояния (МДР или MDS -код), а также обладать рядом дополнительных свойств, приведенных ниже. Преобразование имеет следующий вид:

$$\mathrm{LCM}: \quad \mathbf{B} \to \mathbf{W}$$
$$Y = \mathrm{LCM}(x), \quad x \in \mathbf{B}, \quad Y \in \mathbf{W};$$
$$Y = \langle g_7 x \| \ldots \| g_1 x \| g_0 x \rangle; \quad x, g_i, (g_i x) \in \mathrm{F}_{2^8}; \quad Y \in \mathrm{F}_{2^8}^8;$$
$$\forall i, j, \ (i \neq j): \quad g_i \neq g_j, \quad 0 \le i, j \le 7,$$

где $g_i$ – коэффициенты полинома, образующего циклический MDS–код;
$x$ – входной байт;
$Y$ – выходное слово.

Кроме того, коэффициенты полинома $g_i$ должны быть выбраны таким образом, чтобы все булевы функции выхода преобразования были уникальны, то есть удовлетворяли следующим условиям:

$$Y = \langle y_{63} \| \ldots \| y_1 \| y_0 \rangle, \quad y_i = f_i(x), \quad y_i \in \mathrm{GF}(2);$$
$$\forall i, j, \ (i \neq j): \quad f_i(x) \neq f_j(x), \quad 0 \le i, j \le 63,$$

где $y_i$ – $i$ –й бит слова-результата $Y$;
$f_i$ – булева функция $i$ –го выхода преобразования.

Отметим, что все функции $f_i$ являются линейными, то есть $\deg(f_i) = 1$.

### *3.12 Умножение слова на MDS-матрицу*

Это преобразование определяет биективное линейное отображение слов, составляющих полублок. Преобразование имеет следующий вид:

$$\mathrm{MDS}: \quad \mathbf{W} \to \mathbf{W}$$
$$Y = \mathrm{MDS}(X); \quad X, Y \in \mathbf{W}; \quad B_j \in \mathbf{B};$$
$$X = \langle B_7 \| \ldots \| B_1 \| B_0 \rangle;$$
$$Y = \bigoplus_{j=0}^{7} \mathrm{LCM}\left(B_j\right) <<< (8\times j).$$

Это линейное преобразование может быть представлено в виде матричного умножения – квадратная матрица размерностью 8×8 байт, образованная циклическим MDS–кодом (рассмотренным в п.п. 3.11), умножается справа на вектор-столбец длиной 8 байт (соответствующий слову-аргументу). Полученный вектор-столбец (длиной 8 байт) соответствует слову–результату. Байты, составляющие матрицу и оба вектора, интерпретируются как элементы поля $GF(2^8)$, образованного *выбранным* неприводимым полиномом 8-й степени.

### *3.13 Преобразование* SCL

Преобразование SCL выполняет управляемое биективное отображение слов. В качестве управляющей информации данное преобразование использует 8-битный ключ перестановки *zw*. Преобразование состоит из следующей последовательности шагов:

$$\mathrm{SCL}: \quad \mathbf{W}\times\mathbf{P}'\to\mathbf{W}$$
$$Y=\mathrm{SCL}(X, zw);$$
$$X=\langle B_7 \| \ldots \| B_1 \| B_0\rangle; \quad B_j\in\mathbf{B}; \quad X,Y\in\mathbf{W}; \quad zw\in\mathbf{P}';$$
$$1)\ X'=\langle \mathrm{S}_2(B_7)\| \ldots \| \mathrm{S}_2(B_1)\| \mathrm{S}_2(B_0)\rangle;$$
$$1)\ X''=\mathrm{CP}_{\mathrm{byte}}(X', zw);$$
$$2)\ Y=\mathrm{MDS}(X''),$$

где $X$ – входное слово (64 бита);
$Y$ – выходное слово (64 бита).

Таким образом, SCL преобразование может быть реализовано двумя способами:

Вариант 1:
$$zw\xrightarrow{\alpha}(zd_0, zd_1, \ldots, zd_7), \quad 0\le zd_j\le 7;$$
$$Y=\bigoplus_{j=0}^{7}\mathrm{LCM}\left(\mathrm{S}_1(B_j)\right)<<<(8\times zd_j).$$

Вариант 2:
$$zw\xrightarrow{\beta}(zs_0, zs_1, \ldots, zs_7), \quad 0\le zs_j\le 7;$$
$$Y=\bigoplus_{j=0}^{7}\mathrm{LCM}\left(\mathrm{S}_1(B_{zs_j})\right)<<<(8\times j).$$

Отметим, что применение циклической матрицы позволяет реализовать операцию умножения на матрицу в виде одной таблицы размером 256 слов (то есть 256×8 байт = 2 КБайта), которая, в свою очередь, может быть совмещена с таблицей вычисления S-блока $S_1$.

### *3.14 Преобразование* SCP

Преобразование SCP выполняет управляемое биективное отображение слов. В качестве управляющей информации данное преобразование использует 12-битный ключ перестановки *zw*.

SCP–преобразование имеет следующую структуру:

$$\mathrm{SCP}: \quad \mathbf{W}\times\mathbf{P}\to\mathbf{W}$$

$$Y=\mathrm{SCP}(X, zw);$$

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

1) $X'=\langle \mathrm{S}_2(B_7)\| \ldots \| \mathrm{S}_2(B_1)\| \mathrm{S}_2(B_0)\rangle;$

2) $X''=\mathrm{CP}_{\mathrm{byte}}(X', zw);$

3) $Y=\mathrm{P}_{\mathrm{bit}}(X'').$

Это преобразование может быть реализовано двумя способами:

Вариант 1:
$$zw\xrightarrow{\alpha}(zd_0, zd_1, \ldots, zd_7), \quad 0\le zd_j\le 7;$$

$$Y=\bigvee_{j=0}^{7}\mathrm{P}_{1\mathrm{bit}}\left(\mathrm{S}_2(B_j)\right)<< zd_j.$$

Вариант 2:
$$zw\xrightarrow{\beta}(zs_0, zs_1, \ldots, zs_7), \quad 0\le zs_j\le 7;$$

$$Y=\bigvee_{j=0}^{7}\mathrm{P}_{1\mathrm{bit}}\left(\mathrm{S}_2(B_{zs_j})\right)<< j.$$

Отметим, что применение идентичной битовой перестановки $\mathrm{P}_{1\mathrm{bit}}$ для всех байтов, составляющих слово, позволяет реализовать это преобразование в виде одной таблицы размером 256 слов (то есть 256×8 байт = 2 КБайта), которая, в свою очередь, может быть совмещена с таблицей вычисления S-блока $\mathrm{S}_2$.

### *3.15 Процедура «разворачивания ключа»*

Процедура «разворачивания ключа» алгоритма «Торнадо» построена на базе криптографического генератора псевдослучайных последовательностей (КГПСП), его структурная схема приведена на рис. 10.

КГПСП, используемый процедурой разворачивания ключа алгоритма «Торнадо», построен по схеме циклического шифрования пяти *«слов» регистра состояния генератора* в режиме «связки шифроблоков» (CBC-режим). По разрядности каждое *«слово» регистра состояния* эквивалентно полублоку (то есть 64*n* бит). Шифрование *«слов» регистра состояния* выполняется с помощью базовой F-функции, однако в качестве управляющих, используются только ключевые входы $xk_1$ и $xk_2$, а на входы $zk_1$ и $zk_2$ (ключи перестановки) всегда подаётся постоянное значение $zw^{\mathrm{fix}}=0$. То есть перестановки $\mathrm{CP}_{\mathrm{byte}}$ внутри F-функции КГПСП являются фиксированными и определяют тождественную перестановку.

Введём определение *«сырой» ключевой последовательности*, под которой будем понимать последовательность ключевых полублоков, сформированную на выходе КГПСП, до "усечения" ключей перестановки (шаги процедуры WKGen 1–7, см. ниже).

Таким образом, на каждом шаге формирования *«сырой» ключевой последовательности* текущее состояние КГПСП определяется содержимым *«пятисловного» регистра состояния* $S_{\mathrm{ks}}^{(i)}=(s_0^{(i)}, s_1^{(i)}, s_2^{(i)}, s_3^{(i)}, s_4^{(i)})$, а также *«двухсловного» управляющего ключа* $K^{\mathrm{ks}}=(xk_1^{\mathrm{ks}}, xk_2^{\mathrm{ks}})$. Разрядность каждого *«слова»* составляет 64*n* бит (полублок).

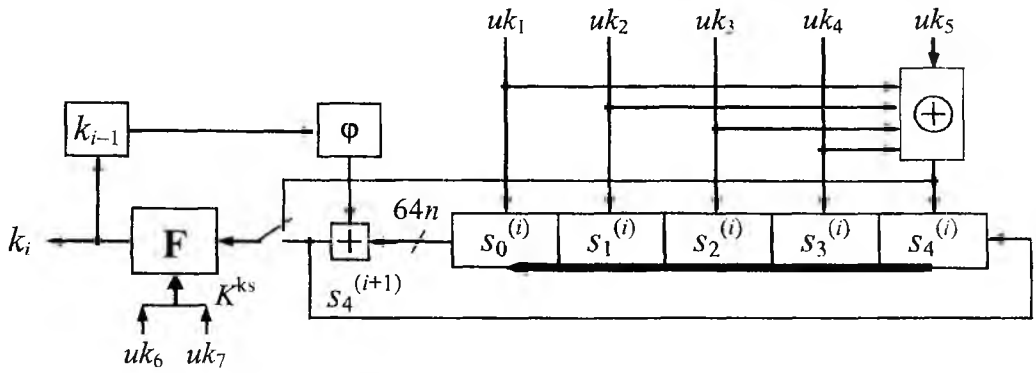


Рис. 10

Алгоритм «Торнадо» поддерживает переменную длину исходного ключа *UK*. Длина $l_{\mathrm{UK}}$ исходного ключа измеряется в полублоках и может находиться в диапазоне от 4 до 7 включительно (то есть $4 \le l_{\mathrm{UK}} \le 7$).

Для описания процедуры разворачивания ключа воспользуемся следующими обозначениями:

$$K^{(i)} = \left(xk_1^{(i)}, zk_1^{(i)}, xk_2^{(i)}, zk_2^{(i)}\right),$$

$$k_i = \left\langle kw_{i,n-1} \| \ldots \| kw_{i,1} \| kw_{i,0} \right\rangle,$$

$$zk_t^{(i)} = \left\langle zw_{t,n-1}^{(i)} \| \ldots \| zw_{t,1}^{(i)} \| zw_{t,0}^{(i)} \right\rangle;$$

$$k_i^{(i)}, xk_1^{(i)}, xk_2^{(i)} \in \mathbf{H}; \quad kw_{i,p} \in \mathbf{W};$$

$$zk_1^{(i)} \in \mathbf{P}'^n; \quad zw_{1,p}^{(i)} \in \mathbf{P}'; \quad zk_2^{(i)} \in \mathbf{P}^n; \quad zw_{2,p}^{(i)} \in \mathbf{P}.$$

где $K^{(i)}$ – рабочий подключ $i$-ой итерации;

$xk_1^{(i)}, xk_2^{(i)}$ – подключи «сложения» $i$-ой итерации;

$k_i$ – $i$-й полублок «сырого» развёрнутого ключа ($i$-й выход КГПСП);

$kw_{i,p}$ – $p$-ое (64-битное) слово $i$-го полублока «сырого» развёрнутого ключа;

$zw_{1,p}^{(i)}$ – ключ SCL перестановки для слова $p$ итерации $i$ (длина 8 бит);

$zw_{2,p}^{(i)}$ – ключ SCP перестановки для слова $p$ итерации $i$ (длина 12 бит).

Начальное состояние КГПСП определяется значением исходного ключа $UK = \left\langle uk_{l_{\mathrm{UK}}} \| \ldots \| uk_2 \| uk_1 \right\rangle$, где $uk_i$ соответствует одному полублоку.

Процедура разворачивания ключа может быть представлена следующей последовательностью шагов.

WKGen:

INPUT: $l_{\mathrm{UK}}, uk_1, uk_2, \ldots, uk_{l_{\mathrm{UK}}} \quad (4 \le l_{\mathrm{UK}} \le 7)$.

OUTPUT: $K^{(0)}, K^{(1)}, \ldots, K^{(2r-1)}, K^{\mathrm{IT}}, K^{\mathrm{FT}}$.

$k_i, s_j^{(i)}, uk_t, xk^{\mathrm{ks}}, xk^{\mathrm{fix}}, g_\varphi \in \mathbf{H}; \quad zk_t^{(i)}, zk^{\mathrm{fix}} \in \mathbf{P}^n$.

Если длина пользовательского ключа меньше 7, то выполняется его доопределение:

1) $if\,(l_{\mathrm{UK}} < 5)\ then\ uk_5 = 0;$

2) $if\,(l_{\mathrm{UK}} < 6)\ then\ uk_6 = xk_1^{\mathrm{fix}};$

3) $if\,(l_{\mathrm{UK}} < 7)\ then\ uk_7 = xk_2^{\mathrm{fix}}.$

Доопределённый пользовательский ключ используется для инициализации схемы разворачивания ключа:

4) $(s_0^{(0)}, s_1^{(0)}, s_2^{(0)}, s_3^{(0)}, s_4^{(0)}) = (uk_1, uk_2, uk_3, uk_4,\ uk_1 \oplus uk_2 \oplus uk_3 \oplus uk_4 \oplus uk_5);$

5) $K^{\mathrm{ks}} = (xk_1^{\mathrm{ks}}, zk_1^{\mathrm{ks}}, xk_2^{\mathrm{ks}}, zk_2^{\mathrm{ks}}) \ \Rightarrow\ xk_1^{\mathrm{ks}} = uk_6,\quad xk_2^{\mathrm{ks}} = uk_7,\quad zk_1^{\mathrm{ks}} = zk_2^{\mathrm{ks}} = zk^{\mathrm{fix}} = 0;$

6) $k_{-1} = \mathrm{F}_{K^{\mathrm{ks}}}\left(s_4^{(0)}\right),\quad g_\varphi = \left(k_{-1} \wedge 15\right) \vee 1.$

После выполнения фазы инициализации выполняется формирование рабочего ключа:

7) for $i = \overline{0, 5r+3}$: $k_i = \mathrm{F}_{K^{\mathrm{ks}}}\left(s_0^{(i)}\right),\quad s_4^{(i+1)} = s_0^{(i)}\,[+]^n\,\varphi(k_{i-1}),$

$$s_j^{(i+1)} = s_{j+1}^{(i)},\quad j = \overline{0, 3};\quad \varphi(k_{i-1}) = \left(k_{i-1} \wedge \overline{15}\right) \vee g_\varphi.$$

Интерпретация рабочего ключа выполняется следующим образом:

8) for $i = \overline{0, r-1}$:

$xk_1^{(2i)} = k_{5i};\quad xk_2^{(2i)} = k_{5i+1};\qquad xk_1^{(2i-1)} = k_{5i+2};\quad xk_2^{(2i+1)} = k_{5i+3};$

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

$zw_{1,p}^{(2i)} = \left(kw_{5i+4,p} >> 0\right) \wedge 0\mathrm{xFF};\qquad zw_{2,p}^{(2i)} = \left(kw_{5i+4,p} >> 16\right) \wedge 0\mathrm{xFFF};$

$zw_{1,p}^{(2i+1)} = \left(kw_{5i+4,p} >> 32\right) \wedge 0\mathrm{xFF};\qquad zw_{2,p}^{(2i+1)} = \left(kw_{5i+4,p} >> 48\right) \wedge 0\mathrm{xFFF};$

9) $K^{\mathrm{IT}} = \left\langle k_{5r+1} \,\|\, k_{5r+0} \right\rangle,\quad K^{\mathrm{FT}} = \left\langle k_{5r+3} \,\|\, k_{5r+2} \right\rangle,$

где $i$ – номер итерации;

$p$ – номер слова в полублоке (индексация с 0).

Как видно из приведенных соотношений, длина рабочего ключа определяется количеством циклов шифрования $r$ и составляет $5r + 4$ полублока. Количество циклов шифрования определяется длиной блока данных (табл. 1).

**Заключение**

Алгоритм «Торнадо» удовлетворяет требованиям первого (максимального) класса стойкости в соответствии с условиями проекта NESSIE (длина блока не менее 128 бит, длина ключа не менее 256 бит). Основной сферой применения БСШ первого класса стойкости является защита информации на уровне приложений, то есть на универсальных аппаратных платформах. Поэтому алгоритм «Торнадо» учитывает ближайшие перспективы развития массовой вычислительной техники и ориентирован, в первую очередь, на 64-битные аппаратные платформы. Однако структура алгоритма позволяет реализовать его достаточно эффективно как на 32-битных микропроцессорах, так и на 8-битных. Кроме того, алгоритм построен по Фейстель-подобной схеме, что обеспечивает инволютивность шифрующего

преобразования и возможность использования общего аппаратного решения для процедур зашифрования и расшифрования.

Конструкция алгоритма «Торнадо» учитывает известные методы криптоанализа БСШ и позволяет защититься от них. В основу алгоритма были положены хорошо известные принципы построения шифров, прошедшие апробацию временем, а также сравнительно новые конструкции [3, 4, 6]. Их совместное применение позволило создать алгоритм, для успешного криптонападения на который требуется разработка новых методов криптоанализа БСШ, если они существуют. Конструкция алгоритма вместо широко распространённого принципа «"слабой" цикловой функции, повторяемой многократно», использует принцип «"сильной" цикловой функции, повторяемой ограниченное число раз» [5]. Это позволяет повысить показатели стойкости алгоритма в соответствии с теоретическим критерием оценки стойкости без снижения показателей по практическому критерию.

Алгоритм «Торнадо» может использоваться в любом стандартном режиме применения БСШ, а также в качестве базового элемента при построении «усиленных» режимов поточного шифрования [7].

**Список литературы:** 1. *AES discussion forum*: http://aes.nist.gov 2. *NESSIE* Call for Cryptographic Primitives, Version 2.2, 8th March 2000: http://cryptonessie.org 3. *Головашич С.А.* Принцип построения инволютивных шифров // Проблемы бионики. Харьков. 2001. Вып. 54. С. 118 – 125. 4. *Головашич С.А.* Метод построения управляемых S-блоков с предельными показателями нелинейности // Радиотехника: Всеукр. межвед. науч.-техн. сб. 2001. Вып. 123. С. 215 – 221. 5. *«Supporting Document on E2»*, Nippon Telegraph and Telephone Corporation, June 14, 1998. 6. *Головашич С.А.* Метод конструирования цикловых функций БСШ // Автоматизированные системы управления и приборы автоматики. Всеукр. межвед. науч.-техн. сб. 2001. Вып. 117. С. 155 – 161. 7. *Головашич С.А.* Безопасность режимов блочного шифрования // Радиотехника: Всеукр. межвед. науч.-техн. сб. 2001. Вып. 119. С. 135 – 145.

*Харьковский национальный*
*университет радиоэлектроники*

*Поступила в редколлегию 25.06.2003*