МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)

Описание блочного шифра Twofish (Twofish block cipher description)

Эссе по курсу **«Защита информации»**
студента 015 группы ФРТК ***Козицкого Петра.***
Долгопрудный, 2004.

Преамбула

Симметричный блочный шифр Twofish является одной из наиболее удачных разработок компании Counterpane Systems (ныне Counterpane Internet Security, Inc.), основанной и возглавляемой Брюсом Шнайером, автором бестселлера «Прикладная криптография». Широкую известность и популярность Twofish приобрел после выхода в финал конкурса на новый американский правительственный стандарт шифрования в августе 1999 года. В создании криптоалгоритма приняли участие: сам Брюс Шнайер (Bruce Schneier), Джон Келси (John Kelsey), Даг Уайтинг (Doug Whiting), Дэвид Вагнер (David Wagner), Крис Холл (Chris Hall) и Нильс Фергюсон (Niels Ferguson).

Основные криптохарактеристики Twofish таковы:

- Длина блока шифрования 128 бит.
- Допустимые длины ключей шифрования 128, 192 и 256 бит.
- Предусмотрена возможность использования ключей шифрования с длинами, отличными от допустимых и не превосходящими 256 бит.
- Отсутствие «слабых» ключей.

В настоящее время Twofish с успехом применяется в различных программных продуктах (см. http://www.schneier.com/twofish-products.html).

Алгоритм шифрования Twofish

На рисунке 1. показана блок-схема алгоритма. Twofish использует 16-раундовую сеть Фейстеля (Feistel Network) с биективной (взаимно однозначной) функцией F и дополнительными «отбеливаниями» (Whitenings) на входе и выходе. Единственное отличие от «чистой» фейстелевской структуры заключается в наличии функциональных блоков, выполняющих циклические однобитовые сдвиги вправо и влево.

128-битовый блок P открытого текста (16 байт $p_0,\ldots, p_{15}$) разбивается на четыре 32-битовых слова P_0, P_1, P_2 и P_3 с сохранением прямого порядка байтов (Little-Endian Convention):

$$P_i = \sum_{j=0}^{3} p_{(4i+j)} \cdot 2^{8j} \qquad i = 0,\ldots, 3$$

На этапе входного «отбеливания» выполняется операция XOR между этими словами и четырьмя ключами K_0, K_1, K_2 и K_3:

$$R_{0,i} = P_i \oplus K_i \qquad i = 0,\ldots, 3$$

После этого следуют 16 раундов шифрования. В каждом раунде два «левых» слова являются входными для функций g (биты одного из входных слов сначала циклически сдвигаются на 8 позиций влево). К полученным выходным словам функций g затем применяется псевдопреобразование Адамара (PHT – Pseudo-Hadamard Transform) и добавляются два раундовых ключа K_{2r+8} и K_{2r+9}, где r – номер раунда шифрования. Далее между модифицированными таким образом «левыми» словами и двумя «правыми» словами (биты одного из которых прежде циклически сдвигаются на одну позицию влево) выполняется операция XOR, после чего циклическому сдвигу на 1 бит вправо подвергается другое из теперь уже видоизмененных «правых» слов. «Левая» и «правая» пары слов затем меняются местами для следующего раунда шифрования. Таким образом,

$$(F_{r,0}, F_{r,1}) = F(R_{r,0}, R_{r,1}, r)$$
$$R_{r+1,0} = \mathrm{ROR}\left(R_{r,2} \oplus F_{r,0}, 1\right)$$
$$R_{r+1,1} = \mathrm{ROL}\ \left(R_{r,3}, 1\right) \oplus F_{r,1}$$
$$R_{r+1,2} = R_{r,0}$$
$$R_{r+1,3} = R_{r,1}$$

где $r = 0,\ldots, 15$, а аббревиатурами ROR и ROL обозначены функции двух аргументов,

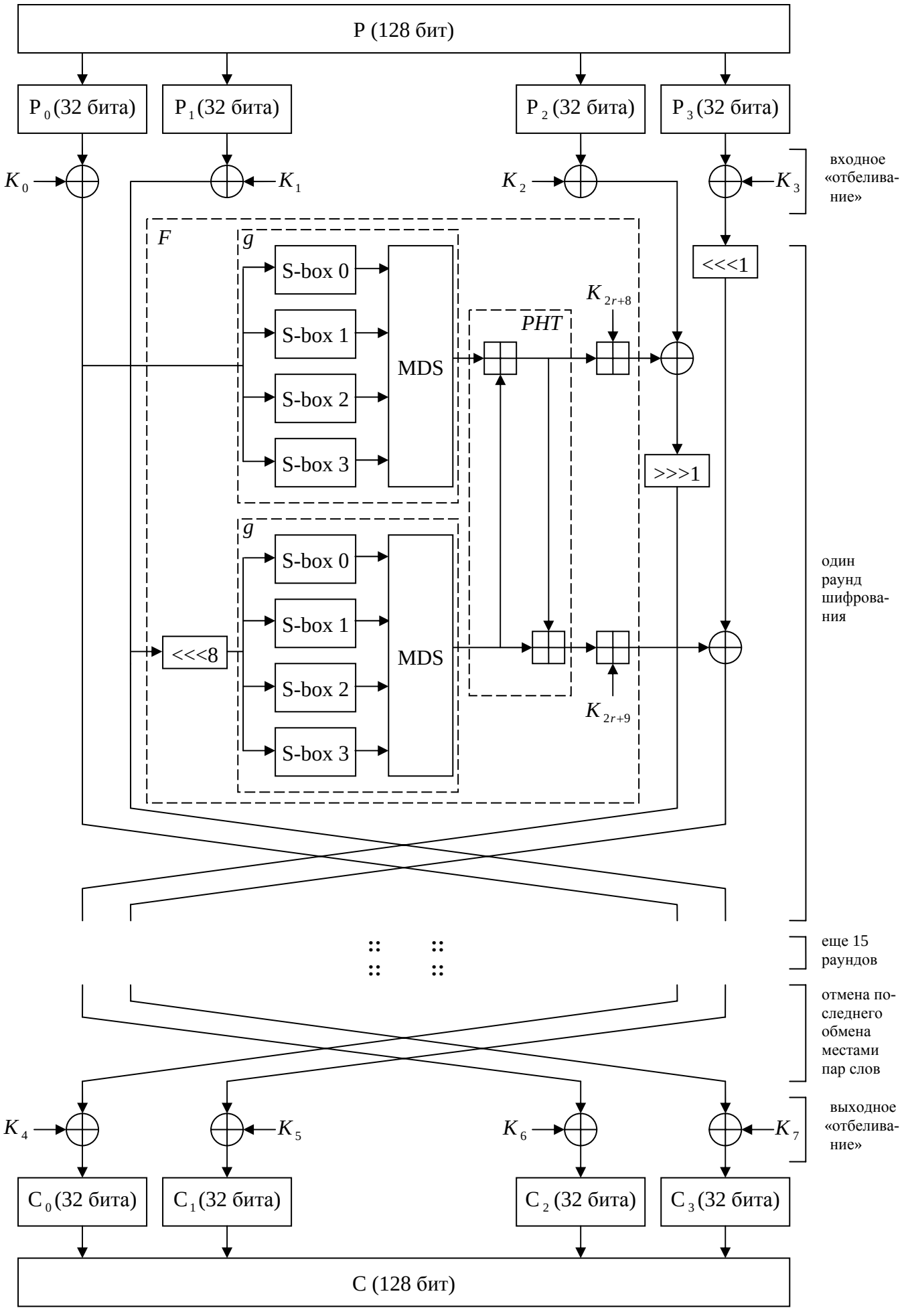

Рис. 1: блок-схема алгоритма шифрования Twofish

выполняющие побитовый циклический сдвиг первого аргумента вправо и влево соответственно на число позиций, равное второму аргументу.

После реализации всех 16-ти раундов шифрования последний обмен местами «левой» и «правой» пар слов отменяется, и между полученными 32-битовыми словами и ключами K_4, K_5, K_6 и K_7 выполняется операция XOR (этап выходного «отбеливания»):

$$C_i = R_{16,(i+2) \bmod 4} \oplus K_{i+4} \qquad i = 0,\ldots, 3$$

Полученные слова C_0, C_1, C_2 и C_3 затем объединяются в 128-битовый блок C (16 байт $c_0,\ldots, c_{15}$) зашифрованного текста:

$$c_i = \left\lfloor \frac{C_{\lfloor i/4 \rfloor}}{2^{8(i \bmod 4)}} \right\rfloor \bmod 2^8 \qquad i = 0,\ldots, 15$$

где $\lfloor x \rfloor$ - целая часть x.

1. Функция *F*

Функция F представляет собой функцию трех аргументов: двух входных слов R_0 и R_1 и номера раунда шифрования r, необходимого для выбора надлежащих раундовых ключей. Преобразование R_0 функцией g дает T_0. Биты R_1 сначала циклически сдвигаются влево на 8 позиций, далее результат сдвига преобразуется функцией g в T_1. Затем к T_0 и T_1, прошедшим PHT, добавляются два раундовых ключа:

$$T_0 = g(R_0)$$
$$T_1 = g(\mathrm{ROL}(R_1, 8))$$
$$F_0 = (T_0 + T_1 + K_{2r+8}) \bmod 2^{32}$$
$$F_1 = (T_0 + 2T_1 + K_{2r+9}) \bmod 2^{32}$$

где F_0 и F_1 - выходы функции F.

2. Функция *g*

Функция g составляет основу алгоритма шифрования. Входное 32-битовое слово X разбивается на четыре байта. Каждый байт проходит через соответствующий S-box, зависящий от раундового ключа. Каждый S-box описывается биективной функцией, преобразующей «входной» байт в «выходной». Четыре результирующих байта интерпретируются как вектор длины 4 над полем Галуа GF(2^8), который умножается слева на MDS-матрицу размеров 4×4 (вычисления также производятся над полем GF(2^8)). Полученный в ходе преобразований вектор рассматривается как 32-битовое слово, которое и является «выходом» функции g. Итак,

$$x_i = \left\lfloor \frac{X}{2^{8i}} \right\rfloor \bmod 2^8 \qquad i = 0,\ldots, 3$$
$$y_i = s_i[x_i] \qquad i = 0,\ldots, 3$$
$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} \cdot & \cdots & \cdot \\ \vdots & MDS & \vdots \\ \cdot & \cdots & \cdot \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$
$$Z = \sum_{i=0}^{3} z_i \cdot 2^{8i}$$

где s_i - функции S-box'ов, Z - выход функции g.

Несколько слов о соответствии между байтовыми величинами и элементами поля GF(2^8). Будем рассматривать GF(2^8) как GF(2)[x]/$v(x)$, где $v(x) = x^8 + x^6 + x^5 + x^3 + 1$ - примитивный полином степени 8 над полем GF(2). Элемент поля $a = \sum_{i=0}^{7} a_i \cdot x^i$, где $a_i \in$ GF(2), отождествляется с байтовой величиной $\sum_{i=0}^{7} a_i \cdot 2^i$. Сложение в поле GF($2^8$) эквивалентно операции XOR между двумя байтами.

MDS-матрица имеет следующий вид:

$$MDS = \begin{pmatrix} 01 & EF & 5B & 5B \\ 5B & EF & EF & 01 \\ EF & 5B & 01 & EF \\ EF & 01 & EF & 5B \end{pmatrix}$$

Элементы MDS-матрицы представляют собой байтовые величины, записанные в шестнадцатиричной системе счисления.

3. Генерация ключей

В ходе шифрования одного блока открытого текста алгоритму генерации ключей необходимо сформировать сорок 32-битовых ключей $K_0, \ldots, K_{39}$, а также четыре зависящих от вида раундового ключа S-box'а, используемых функцией *g*. Twofish принимает ключи с длинами N, равными 128, 192 и 256 бит. Ключи с длинами, отличными от указанных и меньшими 256 бит, также могут быть использованы посредством дополнения их нулями до следующей большей из упомянутых основных длин. Например, 80-битовый ключ $m_0, \ldots, m_9$ будет дополнен до 128-битового ключа таким образом: $m_0, \ldots, m_9, m_i$, где $i = 10, \ldots, 15$.

Положим $k = N / 64$ (возможные значения k равны 2, 3 и 4). Ключ M состоит из $8k$ байт $m_0, \ldots, m_{8k-1}$. Эти байты прежде всего преобразуются в $2k$ слов длиной в 32 бита каждое:

$$M_i = \sum_{j=0}^{3} m_{(4i+j)} \cdot 2^{8j} \qquad i = 0, \ldots, 2k-1$$

и, далее, в два вектора длины k:

$$M_e = (M_0, M_2, ..., M_{2k-2})$$
$$M_o = (M_1, M_3, ..., M_{2k-1})$$

Третий вектор длины k, S, также формируется из ключа следующим образом: ключ разбивается на группы по 8 байт в каждой; каждая из групп рассматривается как вектор длины 8 над полем Галуа GF(2^8), на который умножается справа матрица размеров 8×4, получаемая из RS-кода (Reed-Solomon Code). Результат умножения (размером в четыре байта) рассматривается как 32-битовое слово. Из этих слов и составляется вектор S:

$$\begin{pmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \end{pmatrix} = \begin{pmatrix} \cdot & \cdots & \cdot \\ \vdots & RS & \vdots \\ \cdot & \cdots & \cdot \end{pmatrix} \cdot \begin{pmatrix} m_{8i} \\ m_{8i+1} \\ m_{8i+2} \\ m_{8i+3} \\ m_{8i+4} \\ m_{8i+5} \\ m_{8i+6} \\ m_{8i+7} \end{pmatrix}$$

$$S_i = \sum_{j=0}^{3} s_{i,j} \cdot 2^{8j}$$

где $i = 0,\ldots, k-1$, и, наконец,

$$S = (S_{k-1},...,S_0)$$

Следует обратить внимание на обратный порядок записи слов в векторе *S*.

Для определения операции умножения RS-матрицы справа на вектор-столбец m представим $GF(2^8)$ как $GF(2)[x]/\omega(x)$, где $\omega(x) = x^8 + x^6 + x^3 + x^2 + 1$ - другой примитивный полином степени 8 над полем GF(2). Соответствие между байтовыми величинами и элементами поля $GF(2^8)$ задается подобно тому, как это было сделано в предыдущем разделе в случае с MDS-матрицей. RS-матрица записывается следующим образом:

$$RS = \begin{pmatrix} 01 & A4 & 55 & 87 & 5A & 58 & DB & 9E \\ A4 & 56 & 82 & F3 & 1E & C6 & 68 & E5 \\ 02 & A1 & FC & C1 & 47 & AE & 3D & 19 \\ A4 & 55 & 87 & 5A & 58 & DB & 9E & 03 \end{pmatrix}$$

Таким образом, векторы M_e, M_o и S составляют основу алгоритма генерации ключей.

3. 1. Функция *h*

На рисунке 2. показана блок-схема функции *h*. Она представляет собой функцию двух аргументов: 32-битового слова X и списка 32-битовых слов $L = (L_0,...,L_{k-1})$ длины *k*. Возвращаемое функцией *h* значение представляет собой 32-битовое слово. Режим работы функции *h* включает *k* этапов. На каждом этапе каждый из четырех байтов x_j слова *X* проходит через фиксированный S-box, после чего между ним и соответствующим байтом $l_{i,j}$ слова L_i из списка L выполняется операция XOR. Далее байты снова проходят через фиксированный S-box, а затем умножаются слева (как вектор) на MDS-матрицу. Таким образом,

$$l_{i,j} = \lfloor L_i / 2^{8j} \rfloor \bmod 2^8$$
$$x_j = \lfloor X / 2^{8j} \rfloor \bmod 2^8$$

где $i = 0,\ldots, k-1$ и $j = 0,\ldots, 3$. Далее следует последовательность замен и операций XOR.

$$y_{k,j} = x_j \qquad j = 0,\ldots, 3$$

Если $k = 4$, получаем:

$$y_{3,0} = q_1[y_{4,0}] \oplus l_{3,0}$$
$$y_{3,1} = q_0[y_{4,1}] \oplus l_{3,1}$$
$$y_{3,2} = q_0[y_{4,2}] \oplus l_{3,2}$$
$$y_{3,3} = q_1[y_{4,3}] \oplus l_{3,3}$$

При $k = 3, 4$ получаем:

$$y_{2,0} = q_1[y_{3,0}] \oplus l_{2,0}$$
$$y_{2,1} = q_1[y_{3,1}] \oplus l_{2,1}$$
$$y_{2,2} = q_0[y_{3,2}] \oplus l_{2,2}$$
$$y_{2,3} = q_0[y_{3,3}] \oplus l_{2,3}$$

Во всех случаях ($k = 2, 3, 4$) получаем:

$$y_0 = q_1[q_0[q_0[y_{2,0}] \oplus l_{1,0}] \oplus l_{0,0}]$$
$$y_1 = q_0[q_0[q_1[y_{2,1}] \oplus l_{1,1}] \oplus l_{0,1}]$$
$$y_2 = q_1[q_1[q_0[y_{2,2}] \oplus l_{1,2}] \oplus l_{0,2}]$$
$$y_3 = q_0[q_1[q_1[y_{2,3}] \oplus l_{1,3}] \oplus l_{0,3}]$$

Здесь q_0 и q_1 - фиксированные перестановки 8-битовых величин, которые мы определим

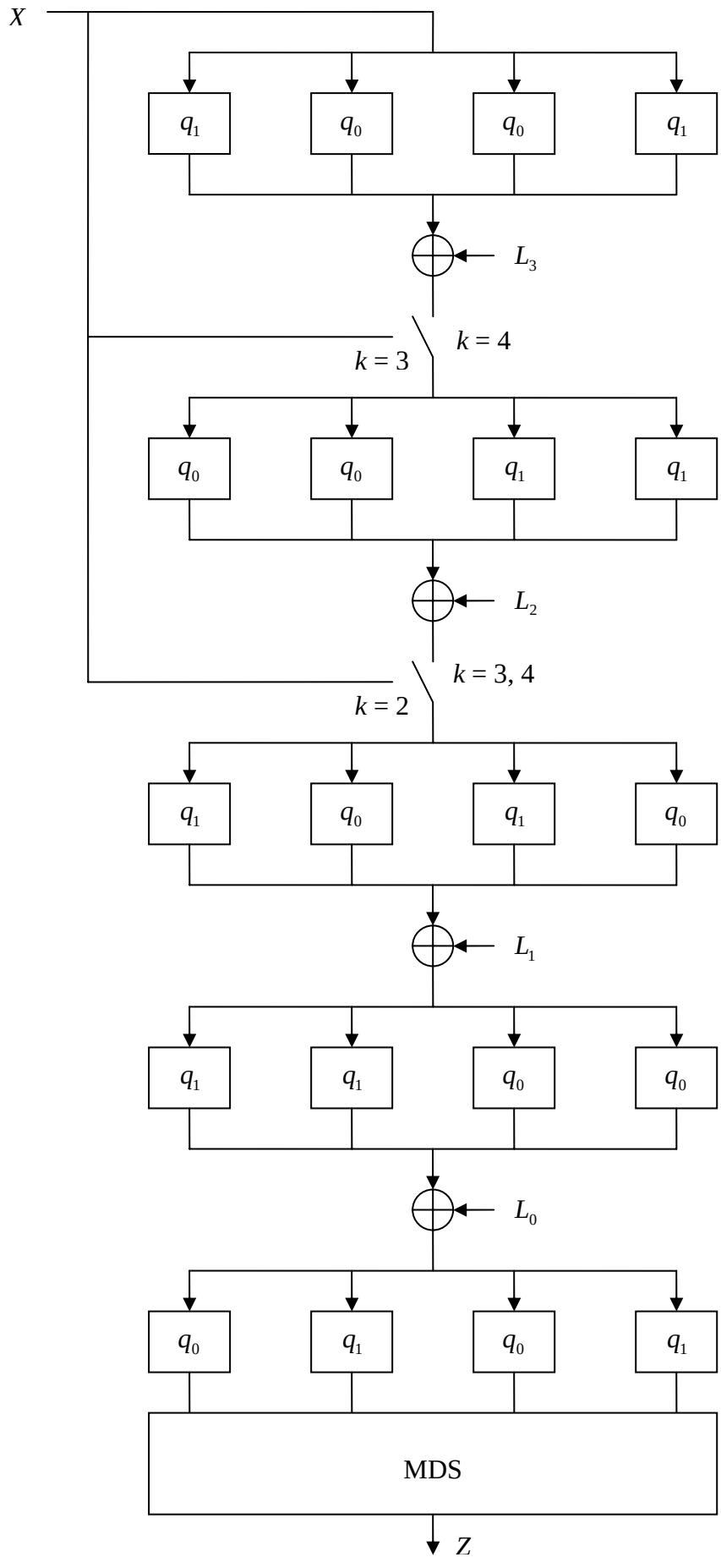

Рисунок 2: блок-схема функции *h*

позднее. Результирующий вектор, составленный из байтов y_j, j = 0,…, 3, умножается слева на MDS-матрицу точно так же, как и в случае с функцией *g*:

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} \cdot & \cdots & \cdot \\ \vdots & MDS & \vdots \\ \cdot & \cdots & \cdot \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$Z = \sum_{j=0}^{3} z_j \cdot 2^{8j}$$

Здесь *Z* – выход функции *h*.

3. 2 Зависящие от вида раундового ключа S-box'ы

Определим теперь S-box'ы в функции *g* следующим образом: $g(X) = h(X, S)$

Следовательно, для j = 0,…, 3 функция s_j зависящего от ключа S-box'а формируется преобразованием x_j в y_j внутри функции *h*, где список *L* есть вектор *S*, формируемый из ключа.

3. 3 Раундовые ключи

Раундовые ключи формируются функцией *h*:

$$\rho = 2^{24} + 2^{16} + 2^8 + 2^0$$

$$A_i = h(2i\rho, M_e)$$

$$B_i = \mathrm{ROL}(h((2i+1)\rho, M_o), 8)$$

$$K_{2i} = (A_i + B_i) \bmod 2^{32}$$

$$K_{2i+1} = \mathrm{ROL}((A_i + 2B_i) \bmod 2^{32}, 9)$$

Константа ρ «дублирует» байты: для i = 0,…, 255 32-битовое слово $i\rho$ состоит из четырех одинаковых байтов, значение (десятичное) каждого из которых равно *i*. Функция *h* оперирует двумя словами такого типа. Для A_i значения байтов равны 2*i*, а вторым аргументом берется M_e. Для B_i значения байтов равны 2*i* + 1, в качестве же второго аргумента выступает M_o, причем биты возвращаемого функцией *h* значения циклически сдвигаются на 8 позиций влево. Затем вычисленные значения A_i и B_i проходят через PHT, после чего биты одного из полученных результатов циклически сдвигаются влево на 9 позиций. В результате получаются два раундовых ключа.

3. 4 Перестановки q_0 и q_1

Перестановки q_0 и q_1 представляют собой фиксированные перестановки 8-битовых величин. Они формируются из четырех различных 4-битовых перестановок. Из входного значения *x* выходное значение *y* (*x*, *y* – байтовые величины) получается следующим образом:

$$a_0 = \lfloor x/16 \rfloor,\ b_0 = x \bmod 16$$

$$a_1 = a_0 \oplus b_0,\ b_1 = a_0 \oplus \mathrm{ROR}_4(b_0, 1) \oplus 8a_0 \bmod 16$$

$$a_2 = t_0[a_1],\ b_2 = t_1[b_1]$$

$$a_3 = a_2 \oplus b_2,\ b_3 = a_2 \oplus \mathrm{ROR}_4(b_2, 1) \oplus 8a_2 \bmod 16$$

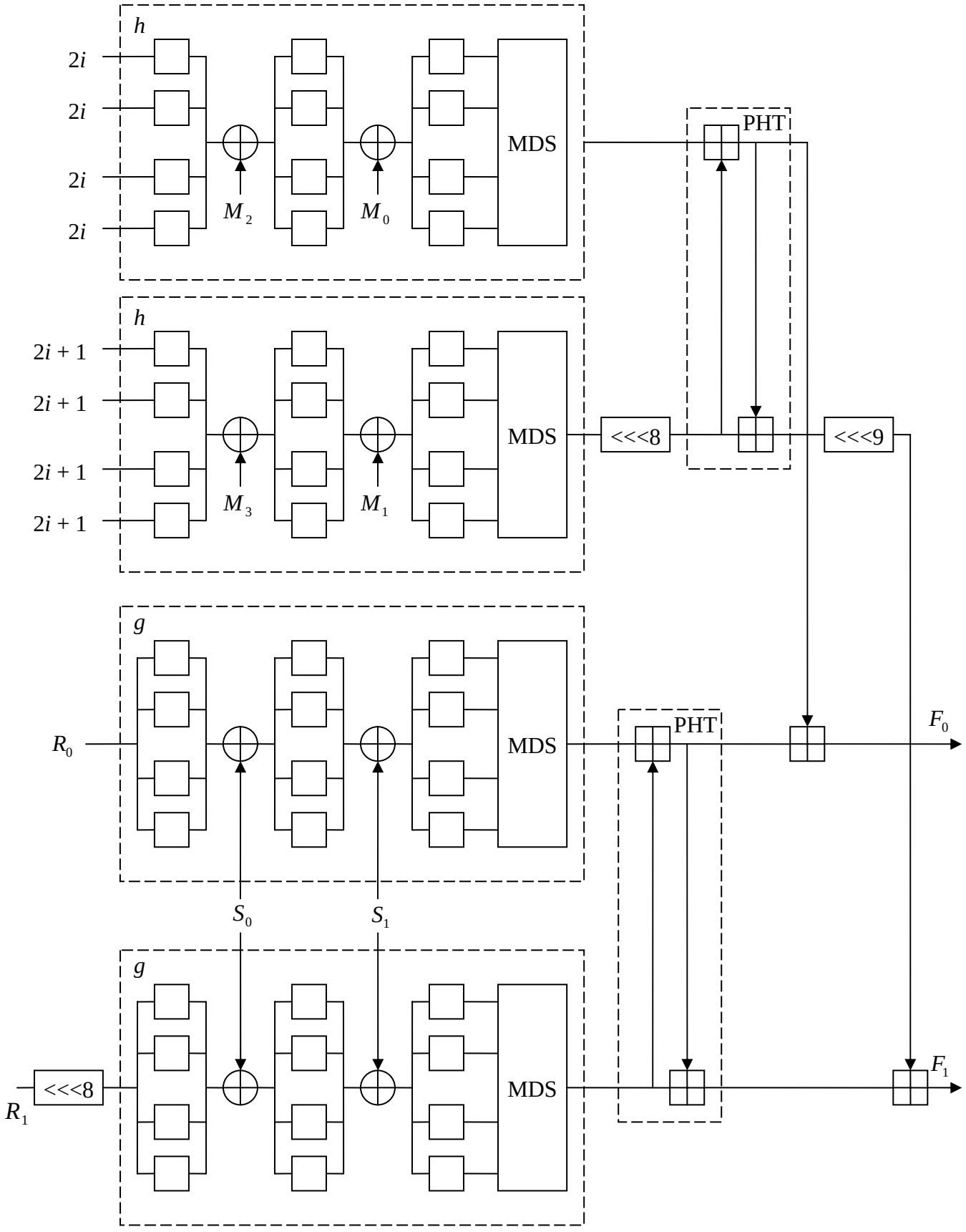

Рисунок 3: блок-схема функции *F* (при длине ключа 128 бит)

$$a_4 = t_2[a_3],\ b_4 = t_3[b_3]$$
$$y = 16b_4 + a_4$$

где функция ROR_4 аналогична функции ROR с той лишь разницей, что в качестве ее первого аргумента выступает 4-битовая величина. Таким образом, сначала байт разбивается на два полубайта (a_0, b_0). Затем следует этап биективного преобразования, приводящий к двум новым полубайтам (a_1, b_1). Далее каждый из полубайтов проходит через свой 4-битовый фиксированный S-box (t_0, t_1), после чего вновь следуют этапы биективного преобразования и прохождения через надлежащие S-box'ы (t_2, t_3). Наконец, два результирующих полубайта (a_4, b_4) формируют байтовое выходное значение *y*. Для q_0 4-битовые S-box'ы имеют такой вид:

t_0 = [8 1 7 D 6 F 3 2 0 B 5 9 E C A 4]
t_1 = [E C B 8 1 2 3 5 F 4 A 6 7 0 9 D]
t_2 = [B A 5 E 6 D 9 0 C 8 F 3 2 4 7 1]
t_3 = [D 7 F 4 1 2 6 E 9 B 3 0 8 5 C A]

а для q_1 такой:

t_0 = [2 8 B D F 7 6 E 3 1 9 4 0 A C 5]
t_1 = [1 E 2 B 4 C 3 7 6 D A 5 F 9 0 8]
t_2 = [4 C 7 5 1 6 9 A 0 E D 8 2 B 3 F]
t_3 = [B 9 5 1 C 3 D E 6 4 7 F 2 0 8 A]

где каждый 4-битовый S-box представляет собой таблицу значений в шестнадцатиричной системе счисления.

4. Обзор раундовой функции

На рисунке 3. показана блок-схема функции *F* при длине ключа шифрования 128 бит (*k* = 2). Внедрение S-box'ов и алгоритма генерации раундовых ключей значительно усложняет ее вид, однако это необходимо для формирования полноценной картины, отображающей работу алгоритма.

Заключение

Несколько слов по поводу криптостойкости Twofish.

Разработчики алгоритма уже изначально были уверены в неприступности своего творения для криптоаналитиков. Во время первого раунда конкурса на новый американский правительственный стандарт шифрования Б. Шнайером был даже объявлен конкурс на лучшую криптоатаку против Twofish с призовым фондом в 10 тысяч долларов. Задача была действительно непростая: сложность алгоритма сыграла свою роль (и стала, кстати, одной из причин, по которой Twofish не стал новым стандартом шифрования США).

Тем не менее, определенные успехи в криптоанализе Twofish все же были достигнуты. Один из известных методов (Impossible Differential Attack), принадлежащий Ларсу Нудсену (Lars Knudsen), для 6-раундового Twofish (без «отбеливаний») имеет сложность 2^{128} для 128-битового ключа, 2^{160} для 192-битового ключа и 2^{192} для 256-битового ключа. Также была предложена атака на 7-раундовый Twofish (опять же без «отбеливаний») со сложностью 2^{256} для 256-битового ключа.

С «отбеливаниями» наилучшая атака «ломала» 6-раундовый Twofish со сложностью 2^{256} и только с длиной ключа, равной 256 бит.

Таким образом, значительный запас криптостойкости Twofish очевиден.

Источники:

1. Twofish: A 128-Bit Block Cipher (http://www.schneier.com/paper-twofish-paper.pdf)
2. Impossible Differentials in Twofish (http://www.schneier.com/paper-twofish-impossible.pdf)