Journal of Siberian Federal University. Engineering & Technologies, 2016, 9(2), 214-224
~ ~ ~

УДК 681.3

Working Out the New Algorithm Enciphered the Data with a Symmetric Key

Shukhrat A. Umarov*[a] and Davlatali E. Akbarov[b]
[a]UShA, Fergana Branch of the Tashkent University Information Technologies Fergana, Uzbekistan
[b]ADE, Fergana Branch of the Tashkent University Information Technologies Fergana, Uzbekistan

Received 22.05.2015, received in revised form 18.10.2015, accepted 24.01.2016

In this article data encryption algorithm with symmetric key which uses combination of practically irreversible transformations is described. Offered algorithm based on Feistel scheme.

Keywords: algorithm, symmetric key, Feistel scheme.

Citation: Umarov S.A., Akbarov D.E. Working out the new algorithm enciphered the data with a symmetric key, J. Sib. Fed. Univ. Eng. technol., 2016, 9(2), 214-224, DOI: 10.17516/1999-494X-2016-9-2-214-224.

Разработка нового алгоритма шифрования данных с симметричным ключом

Ш.А. Умаров[а], Д.Е. Акбаров[б]
[а]УША, Ферганский филиал Ташкентского университета информационных технологий Фергана, Узбекистан
[б]АДЕ, Ферганский филиал Ташкентского университета информационных технологий Фергана, Узбекистан

В статье описывается алгоритм шифрования данных с симметричным ключом с использованием комбинации практически необратимых преобразований. Предлагаемый алгоритм основан на сети Файстеля.

Ключевые слова: алгоритм, симметричный ключ, сети Файстеля.

* Corresponding author E-mail address: sht00357@gmail.com

Введение

Предлагаемый симметричный блочный алгоритм шифрования данных основан на сети Фейстеля, который обеспечивает удобную аппаратную реализацию [1, 2].

Математическая модель такого преобразования, реализуемого сетью Файстеля в *i*-м цикле шифрования, имеет следующий вид:

$$\begin{cases} L_i = R_{i-1}, \\ R_i = L_{i-1} \oplus f_i\left(R_{i-1}, k_{pi}\right), \end{cases}$$

где $X_i = (L_{i-1}, R_{i-1})$ – входной блок *i*-го цикла, разделенный на две половины L_{i-1} и R_{i-1}, а блок $Y_i = (L_i, R_i)$ – результат зашифрования блока X_i на ключе *i*-раунда k_{pi} с помощью функции f_i.

Алгоритм шифрования реализуется несколькими итерациями преобразования сети Файстеля с использованием ключа *k*. При этом каждая *i*-итерация использует в качестве входного блока X_i результат предыдущей итерации Y_{i-1}, т.е. $X_i = Y_{i-1}$, ключ *i*-раунда k_{pi}, вычисляемый определенным образом по ключу *k*, а функция f_i зависит или не зависит от номера итерации. Когда функция f_i не зависит от номера итерации, в каждом раунде используется одна и та же функция *f* с разными ключевыми параметрами соответствующего раунда.

Ценность преобразований подобного вида заключается в том, что функция f_i может не быть обратимой функцией, но преобразование сети Файстеля обратимо. Действительно, из математической модели *i*-го цикла, приведенного выше, учитывая свойства двоичного сложения, нетрудно получить соотношения

$$\begin{cases} R_{i-1} = L_i, \\ L_{i-1} = R_i \oplus f_i\left(L_i, k_{p_i}\right), \end{cases}$$

которые дают математическую модель алгоритма расшифрования в каждой *i*-й итерации.

Вышеперечисленные блочные симметричные алгоритмы шифрования данных, основанные на сети Файстеля, отличаются друг от друга по конструкции функции *f*.

Постановка задачи

В предлагаемом алгоритме для конструкции функции *f* сети Файстеля используются следующие преобразования: битовое сложение открытого текста с раундовым ключом по mod 2; матричное преобразование по mod 256; *S*-блок; таблица сжатия. Математические модели этих перечисленных преобразований и их свойства отличаются от преобразований известных вышеперечисленных алгоритмов.

Предлагаемый алгоритм, используя комбинацию преобразований: битовое сложение по mod 2; матричное преобразование по mod 256; *S*-блока и сжатия, осуществляет шифрование 64-битовых блоков данных с помощью 256-битового ключа *k*.

Решение задачи. В приводимом описании предлагаемого алгоритма использованы следующие обозначения:

- T_0 – 64-разрядные (битовые) блоки открытых данных;
- $T_ш$ –64-разрядные (битовые) блоки шифрованных данных;
- t_i – *i*-бит-последовательности открытых данных;
- L_i и R_i – левая и правая половина 64-разрядного (битового) блока L_iR_i, где i = 0,1,2,…,8;

– $(a_1(i), a_2(i), ..., a_{32}(i))$ – биты левой части i-раунда преобразования, т.е.

$$L_i = (a_1(i), a_2(i), ..., a_{32}(i));$$

– $(b_1(i), b_2(i), ..., b_{32}(i))$ – биты правой части i-раунда преобразования, т.е.

$$R_i = (b_1(i), b_2(i), ..., b_{32}(i));$$

– $u_{4\times1} = (u_1, u_2, u_3, u_4)$ – 32-разрядный (четыре восьмибитных) входной вектор матричного преобразования, где значения байтов x_i на интервале $0 \le x_i \le 255$, $i = 1,2,3,4$;

– $A_{n\times4}$ – прямоугольная матрица (которая генерируется от ключевой последовательности по заранее определенному правилу, следовательно, она секретная), где $n = 2^m$, $m = 2,\ldots,M$, $M < \infty$, элементы a_{ij} $(i = 1,...,n;\ j = 1,2,3,4;)$ этой матрицы выражаются одним байтом, поэтому удовлетворяют условию $0 \le a_{ij} \le 255$;

– $y_{n\times1} = (y_1, y_2, ..., y_n)$ – выходной вектор результата матричного преобразования $A_{n\times4}u_{n\times4}$ по mod 256, т.е. $y_{n\times4} = A_{n\times4}u_{n\times4}(\text{mod } 256)$, где y_i-байты, $0 \le y_i \le 255$, i=1,2,…,n ;

– S-блок (который генерируется от ключевой последовательности по заранее определенному правилу, следовательно, он секретный) преобразования, состоящий из 256 узлов замены S_0, S_2,…,S_{255}, имеющие один байт (восемь битов) входов и выходов:

$$S_0 \qquad S_1 \qquad S_2 \qquad \ldots \qquad S_{255}$$

где $0 \le S_1, S_2, \ldots, S_{256} \le 255$ и $S_i \ne i$ и $S_i \ne S_j$ при $i \ne j$, т.е. числа S_i принимают значения в интервале $0 \le S_i \le 255$ произвольным правилом;

– $\oplus$ – операция побитового сложения векторов-блоков по mod 2 (по модулю 2);

– $z_{n\times1} = (z_1, z_2, ..., z_n)$ – вектор, представляющий собой результат преобразования вектора $y_{n\times1} = (y_1, y_2, ..., y_n)$ через S-блок, т.е. $z_{n\times1} = S(y_{n\times1})$, где z_i–байты, $0 \le z_i \le 255$, i=1,2,…,n;

– $k = k_1k_2\ldots k_8$ – 256-разрядный ключ, записанный в виде восьми 32-разрядных подключей k_i, $i = 1,\ldots,8$;

– $k = k_1(i)k_2(i)\ldots k_{32}(i)$ – 32-разрядный i-подключ;

– $k_i^1 = k_1(i)k_2(i)...k_8(i), ..., k_i^4 = k_{25}(i)k_{26}(i)...k_{32}(i)$-четыре байта 32-разрядного i-подключа k_i;

– $k_{pi} = (k_1(pi)k_2(pi)...k_{32}(pi))$ – 32-разрядные (битовые) ключи i-го раунда (цикла), где $pi = 1,\ldots,8$;

– $k_н$ – 64-разрядный начальный ключ;

– $k_к$ – 64-разрядный конечный ключ;

– f – функция шифрования;

– ТС – таблица сжатия 16x16 (секретная, передается вместе с ключом или генерируются от ключа по определенному правилу), используемая при генерации раундовых ключей, в ячейках которой в равномерном распределеним расположены числа q_{ij}, где $0 \le q_{ij} \le 15$, $i = 0,\ldots,15$, $j = 0,\ldots,15$:

$$\begin{array}{cccc} q_{00} & q_{01} & \ldots & q_{0,15} \\ q_{10} & q_{11} & \ldots & q_{1,15} \\ \ldots & \ldots & \ldots & \ldots \\ q_{15,0} & q_{15,1} & \ldots & q_{15,15} \end{array}$$

– $w_{n\times 4} = (w_1, w_2, w_3, w_4)$ – 32-разрядный (4-байтный) вектор результата сжатия.

Кроме перечисленных обозначений использованы промежуточные, которые не были приведены в данном списке.

При шифровании открытые данные разбиваются на 64-разрядные блоки. Процедура зашифрования каждого 64-разрядного блока T_0 включает 8 раундов (циклов). В ключевое запоминающее устройство вводится 256-разрядный ключ k, записанный в виде восьми 32-разрядных подключей k_i : $k = k_1k_2\ldots k_8$.

Каждый ключ i-раунда k_{pi} генерируется по 32-разрядному подключу $k_i = k_1(i)k_2(i)\ldots k_{32}(i)$, разделив его на четыре байта, т.е. $k_i = (k_i^1, k_i^2, k_i^3, k_i^4) = \left(k_1(i)k_2(i)\ldots k_8(i), \ldots, k_{25}(i)k_{26}(i)\ldots k_{32}(i)\right)$, и преобразуя эти байты через соответствующие узлы блока S, причем по значениям $k_i^1,\ldots,k_i^4$ в десятичной системе исчисления $(k_i^1)_{10},\ldots, (k_i^4)_{10}$ определяются номера узлов преобразования в S-блоке, а результатом преобразования байтов $k_i^1,\ldots, k_i^4$ являются двоичные представления значения $S_{k_i^1},\ldots, S_{k_i^4}$, т.е. $(S_{k_i^1})_2,\ldots, (S_{k_i^4})_2$, соответствующих узлов $(k_i^1)_{10},\ldots, (k_i^4)_{10}$:

$$k_{pi} = S\left(k_i^1,\ldots,k_i^4\right) = \left(S\left(k_i^1\right),\ldots,S\left(k_i^4\right)\right) = \left((S_{k_i^1})_2,\ldots,(S_{k_i^4})_2\right) = \left(k_{pi}^1,\ldots,k_{pi}^4\right) .$$

Из исходного 256-разрядного ключа k с использованием таблицы сжатия ТС генерируется 64-разрядный начальный ключ $k_н$.

Исходный 256-разрядный ключ k преобразуется через S-блок, и его результат подвергается сжатию до 64-разрядного (битового) блока, который принимается как конечный ключ $k_к$.

Алгоритм шифрования осуществляется в следующих процессах. Вначале последовательность битов блока T_0, подлежащего зашифрованию, побитно суммируется начальным ключом по mod 2, т.е. $T_0 \oplus k_i = T_0^{'}$, и опять присвоив $T_0 = T_0^{'}$, T_0 разбивается на две половины по 32 бита:

$$T_0 = \left(t_1(0), t_2(0),\ldots, t_{32}(0), t_{33}(0),\ldots, t_{64}(0)\right) = (a_1(0), a_2(0),\ldots, a_{32}(0), b_1(0), b_2(0),\ldots, b_{32}(0)) = (L_0, R_0) .$$

На первом раунде вычисление значения функции f производится в следующем виде:

1. Блок R_0 побитно по модулю 2 складывается 32-разрядным ключом первого раунда $k_p1 = k_1(p1)k_2(p1)\ldots k_{32}(p1)$, т.е.

$$\begin{aligned}
&b_1(0)b_2(0)\ldots b_{32}(0) \oplus k_1(p1)k_2(p1)\ldots k_{32}(p1) = \\
&= (b_1(0) \oplus k_1(p1))(b_2(0) \oplus k_2(p1))\ldots(b_{32}(0) \oplus k_{32}(p1)) = \\
&= x_1(1)x_2(1)\ldots x_8(1)x_1(2)x_2(2)\ldots x_8(2)x_1(3)x_2(3)\ldots x_8(3)x_1(4)x_2(4)\ldots x_8(4) = \\
&= (x_1, x_2, x_3, x_4) = x_{4\times 1}.
\end{aligned}$$

2. После этого 32-битовая последовательность, разделенная на четыре подблока, по восемь бит каждый, т.е. $x_{4\times 1} = (x_1, x_2, x_3, x_4) = (X, Y, Z, W)$, соответствующие элементы которого $(x,y,z,w) = (x_l(1), x_l(2), x_l(3), x_l(4))$, $l = 1,\ldots,8$; этих подблоков подвергается логическому преобразованию:

$$F(X,Y,Z,W) = yw \oplus x\overline{w} \oplus xzw \oplus xyz = u(1);$$

$$G(X,Y,Z,W) = \overline{x} \oplus y \oplus z\overline{w} \oplus xzw \oplus xyw = u(2);$$

$$R(X,Y,Z,W) = \overline{x}z \oplus \overline{x}y\overline{z} \oplus xzw = u(3);$$

$$V(X,Y,Z,W) = \overline{w} \oplus y\overline{z} \oplus \overline{x}zw = u(4);$$

где выбранные булевы функции $F(X,Y,Z,W)$, $G(X,Y,Z,W)$, $R(X,Y,Z,W)$, $V(X,Y,Z,W)$ являются сбалансированными, регулярными и максимально нелинейными [3].

3. Результат преобразования предыдущего этапа $u_{4\times1}$ подвергается матричному преобразованию в конечном поле целых чисел по модулю 256, т.е.

$$y_{n\times1} = (A_{n\times1}u_{n\times1})\text{mod } 256.$$

4. Каждый i-байт y_i, $i = 1,...n$, $8 \times n$-разрядного (n-байтного) вектора $y_{n\times1}$ преобразуется через соответствующие узлы блока S, причем по значению i-байта $(y_1(i)y_2(i)\ldots y_8(i))_2 = y_i$ в десятичной системе исчисления $(y_1(i)y_2(i)\ldots y_8(i))_2 = (y_i)_{10}$ определяются номера узлов преобразования в S -блоке, а результатом преобразования байта является двоичное представление значения S_{y_i}, т.е. $\left(S_{y_i}\right)_2$ соответствующего узла $(y_i)_{10}$: $z_i = S(y_i) = S\left(y_1(i)y_2(i)\ldots y_8(i)\right) = \left(S_{y_i}\right)_2$, следовательно, имеем $8\times n$-разрядного (n-байтного) вектора $z_{n\times1} = S(y_{n\times1}) = (z_1,z_2,\ldots,z_n)$ как результат преобразования вектора $y_{n\times1}$ через S-блок.

5. По таблице сжатия ТС $8\times n$-разрядный (n-байтный) вектор $z_{n\times1}$ сжимается в 32-разрядный (4-байтный) вектор $w_{4\times1} = (w_1, w_2, w_3, w_4)$ следующим образом:

– каждый байт z_i вектора $z_{n\times1}$ разделяется на полубайты, т.е. полагается, что $z_{n\times1} = (z_1,\ldots,z_n) = \left(z'_1,\ldots,z'_{2n}\right) = z'_{2n\times1}$;

– по значениям полубайтов z_1 z'_1 и z'_{2n} в десятичной системе исчисления $\left(z'_1\right)_{10}$ и $\left(z'_{2n}\right)_{10}$ соответственно определяются номера строки и столбца ТС, число $q_{\left(z'_1\right)_{10}\left(z'_{2n}\right)_{10}}$ (полубайт), находящееся в пересечении этой строки и столбца, является результатом сжатия полубайтов z'_1 и z'_{2n} на полубайт байт $q_{\left(z'_1\right)_{10}\left(z'_{2n}\right)_{10}}$, далее этот процесс повторяется для всех пар (z'_2, z'_{2n-1}), (z'_3, z'_{2n-2}), …, (z'_n, z'_{n+1}), т.е. для всех пар (z'_i, $z'_{2n-(i-1)}$), где i=1,…, n;

– результат сжатия, полученный на предыдущем шаге с использованием ТС аналогичным образом, подвергается сжатию (m-2) раз и получается результат полного сжатия $w_{4\times1} = (w_1, w_2, w_3, w_4)$ – 32-разрядный (4-байтный) вектор.

6. Результат полного сжатия $w_{4\times1} = (w_1, w_2, w_3, w_4)$ – 32-разрядный (4-байтный) вектор побитно по модулю 2 сложится левой половиной L_0 блока данных T_0:

$$L_0 \oplus w_{4\times1} = t_1(0)t_2(0)\ldots t_{32}(0) \oplus w_1(1)w_2(1)\ldots w_8(1)w_1(2)\ldots w_8(2)w_1(3)\ldots w_8(3)w_1(4)\ldots w_8(4) = \\ = L_0 \oplus f\left(R_0, k_{p1}\right) = R_1,$$

где функцией $f(R_0, k_{p1})$ обозначено преобразования пунктов 1-4 блока данных R_0 на ключе k_{p1}.

7. Без каких либо изменений битов блока R_0 полагается: $L_1 = R_0$.

Вышеприведенные пункты 1-6 преобразования блока данных T_0 в целом представляют 1-раунд преобразования предлагаемого алгоритма шифрования.

Полагая, что $L_0 = L_1$, $R_0 = R_1$ и $k_{p1} = k_{p2}$, затем, аналогичным образом повторяя пункты преобразования 1-6, осуществляется 2-раунд преобразования алгоритма. Таким образом, если результат преобразования (i–1)-раунда получен, то, полагая $L_0 = L_{i-1}$, $R_0 = R_{i-1}$ и $k_{p1} = k_{pi-1}$, затем повторяя пункты преобразования 1-6, осуществляется i-раунд алгоритма шифрования. Количество раундов предлагаемого алгоритма шифрования равно 8, т.е. i=1,2,…,8.

Блок $T_к = R_8L_8$, образованный из объединения блоков L_8 и R_8, побитно суммируется по модулю 2 с конечным ключом $k_к$, т.е. $T_к \oplus k_k = T_ш$.

Алгоритм дешифрования осуществляется в следующих процессах. По алгоритму $T_к = (R_8, L_8)$ и $T_ш$-64-разрядный (битовый) блок шифрованного данного образован в виде $T_ш = T_к \oplus k_k = (R_8, L_8) \oplus k_k$. Такое создание шифрованных данных обусловлено тем, что в шифровании и расшифровании пользуются одним и тем же алгоритмом, причем при расшифровании принимаются в качестве входного блока данных $T_ш$ и начального ключа k_k, раундовые ключи используются в обратном порядке: в 1-раунде k_{p8}, во 2-раунде k_{p7}…, 8-раунде k_{p1} и в качестве конечного ключа k_i. В итоге следует открытый блок данных.

Выше было отмечено, что прямоугольное матричное преобразование $A_{n\times4}$ (где $n = 2^m$, $m = 2,\ldots,M$, $M < \infty$), S-блок и ТС – таблица сжатия генерируются от ключевой последовательности по заранее определенным правилам. Здесь приведем правила генераций.

Генерация матрицы матричного преобразования. Матрица преобразования $A_{n\times4}$ имеет nx4 элементов. Каждый элемент a_{ij} ($i = 1,\ldots,n$; $j = 1,2,3,4$) содержит по 8 битов=1 байт.

При $m = 2$ матрица преобразования квадратичная, т.е. $A_{4\times4}$, она имеет 16 элементов, содержащих по 8 битов. В этом случае 256-битный ключ k циклический сдвигается направо на λ бит (где λ – любое нечетное число из интервала от 3 до 255) и элементы a_{ij} ($i = 1,2,3,4$; $j = 1,2,3,4$) образуются из первой 128-битовой сдвинутой последовательности. Например, полагаем первый байт a_{11}, следующий байт a_{12} и так далее шестнадцатый байт a_{44}. Далее сравнением устанавливается, чтобы на каждой строке матрицы хотя бы один элемент был с нечетным значением и все элементы матрицы были с различными значениями.

При $m = 3$ матрица преобразования прямоугольная, т.е. $A_{8\times4}$ она имеет 32 элемента, 256-битный ключ k циклический сдвигается направо на λ бит и элементы a_{ij} ($i = 1,2,3,4,\ldots,8$; $j = 1,2,3,4$) образуются из 256-битовой последовательности, полученной в результате сдвига; полагаем, что первый байт a_{11}, следующий байт a_{12} и так далее тридцать второй байт a_{84}. Далее сравнением устанавливается, что на каждой строке матрицы хотя бы один элемент был с нечетным значением, кроме того, все элементы матрицы были с различными значениями.

При $m = 4$ матрица преобразования прямоугольная, т.е. $A_{16\times4}$ и имеет 64 элемента, 256-битный ключ k циклически сдвигается направо на λ бит и элементы a_{ij} ($i = 1,2,3,4,\ldots,8$; $j = 1,2,3,4$) образуются из сдвинутой 256-битовой последовательности; полагаем, что первый байт a_{11}, следующий байт a_{11} и так далее тридцать второй байт a_{84}. Далее уже сдвинутый 256-битный ключ k опять циклический сдвигается направо на λ бит и элементы a_{ij} ($i = 9,\ldots,16$; $j = 1,2,3,4$) образуются из опять сдвинутой 256-битовой последовательности; полагаем, что первый байт a_{91}, следующий байт a_{92} и так далее тридцать второй байт $a_{16,4}$. Далее сравнением элементов устанавливается, что элементы любых двух столбцов прямоугольной матрицы $A_{n\times4}$ должны быть пропорциональными и на каждой строке матрицы хотя бы один элемент имел нечетное значение, кроме того, все элементы матрицы имели различные значения.

По правилу приведенной процедуры образуются элементы $A_{n\times4}$ при достаточно большом $n = 2^m$, $m = 2,\ldots,M$, $M < \infty$.

Генерация *S*-блока. В начале процедуры генерации S-блока 256-битный ключ k разделяется на 32-байта, эти байты попарно сравниваются, в результате сравнения получим попарно различные байты в количестве не больше чем 32 байта, с которых начинается заполнение ячеек S-блока. Далее исходный 256-битный ключ k циклический сдвигается направо на λ = 1 бит, затем полученная последовательность разделяется на 32 байта, и они попарно сравни-

ваются с содержащими элементами ячеек *S*-блока, в результате выявленными попарно различными байтами продолжается заполнение последующих ячеек *S*-блока. Далее 256-битный исходный ключ k циклический сдвигается направо на $\lambda = 2$ бит, затем вновь 32 байта полученной последовательности попарно сравниваются с элементами ячеек *S*-блока, выявленными попарно различными байтами, продолжается заполнение последующих ячеек *S*-блока и так далее. Этот процесс продолжается со сдвигом $\lambda = 3,4,\ldots,255$, пока не заполнятся все ячейки *S*-блока. Если еще не заполнены все ячейки *S*-блока, то вычисляется множество чисел по формуле $y = x^z \bmod 257$, где фиксированное нечетное число $z = conct$, известное передаваемое вместе с ключом и $0 \le x \le 256$. В результате вычисления в некоторой последовательности образуется совокупность чисел $\{0 \le y \le 256 : y = x^z \bmod 257, z = conct, 0 \le x \le 256\}$, так как число $n = 257$ простое. Элементы γ множества $\{0 \le y \le 256 : y = y \bmod 256, 0 \le y \le 256\}$ сравниваются с содержащими элементами ячеек *S*-блока, выявленными попарно различными байтами, и продолжается заполнение последующих ячеек *S*-блока. Таким образом, полностью заполняются все ячейки *S*-блока.

Генерация ТС – таблицы сжатия. В начале процедуры генерации 256-битный ключ k разделяется на 64 полубайта, первый полубайт принимается как первый элемент первой строки, второй полубайт сравнивается с первым полубайтом. Если они не равны, то вторым элементом первой строки принимается второй полубайт, иначе второй полубайт принимается как первый элемент второй строки. Далее третий полубайт сравнивается с первым элементом первой строки, если они равны, то третий полубайт сравнивается со вторым полубайтом первой строки. А когда в качестве второго элемента первой строки был принят второй полубайт, если они не равны, то третий полубайт принимается как третий элемент первой строки, иначе третий полубайт сравнивается с первым элементом второй строки, если они не равны, то третий полубайт принимается как вторым элементом второй строки, иначе третий полубайт принимается как первый элемент третьей строки и так далее аналогичным образом определяются значения элементов таблицы. При этом заполняются не больше 32 ячеек таблицы. Процесс заполнения остальных ячеек полубайтами продолжается с циклическим сдвигом направо на $\lambda = 1$ бит исходного ключа k, затем полученная последовательность разделяется на полубайты и аналогично к предыдущему они сравниваются с содержащими элементами строк таблицы, в результате выявленными попарно различными полубайтами в строке продолжается заполнение последующих ячеек строк таблицы. Далее исходный ключ k циклический сдвигается направо на $\lambda = 2$ бит, опять полубайты полученной последовательности попарно сравниваются с содержащими элементами ячеек строк таблицы. Выявленными попарно различными в строке полубайтами продолжается заполнение последующих ячеек строк таблицы и так далее этот процесс продолжается со сдвигом $\lambda = 3,4,\ldots,255$, пока не заполнятся все ячейки строк таблицы. Если еще не заполнены все ячейки *S*-блока, то вычисляется множество чисел по формуле $y = x^z \bmod 17$, где $z = conct$, $0 \le x \le 16$. В результате вычисления в некоторой последовательности образуется совокупность чисел $\{0 \le y \le 16 : y = x^z \bmod 17, z = conct, 0 \le x \le 16\}$, так как число $n = 17$ простое. Элементы γ множества $\{0 \le y \le 15 : y = y \bmod 16, 0 \le y \le 16\}$ сравниваются с содержащими элементами ячеек в строке таблицы, выявленными попарно различными полубайтами, и продолжается заполнение последующих ячеек строк таблицы. Таким образом, полностью заполняются все ячейки строк таблицы сжатия ТС.

Вместо того чтобы генерировать прямоугольную матрицу $A_{n\times 4}$ (где $n = 2^m$, $m = 2,\ldots,M$, $M < \infty$), S-блок и ТС от исходного ключа их можно генерировать независимо от исходного ключа, при этом заранее сгенерированные преобразования должны удовлетворять следующим основным требованиям.

Элементы любых двух столбцов прямоугольной матрицы $A_{n\times 4}$ должны быть пропорциональными, на каждой строке хотя бы один элемент должен быть с нечетным значением; кроме того, все элементы должны быть с различными значениями и матрица может быть открытой (или секретной, как долговременный ключ).

Расположение чисел ячеек S-блока должно быть случайным и может быть открытым (или секретным, как долговременный ключ).

Элементы каждой строки и столбца, а также диагонали таблицы ТС не должны повторяться, и она может быть открытой (или секретной, как долговременный ключ).

В настоящее время многие алгоритмы, основанные на сети Фейстеля, безупречно используются на практике, отвечая требуемой стойкости, но развитие создания высокоскоростных вычислительных средств и информационных технологий требует разработки стойких алгоритмов шифрования данных для обеспечения защиты информации в сети телекоммуникации криптографическими методами. Тот факт, что алгоритмы шифрования данных на основе сети Файстеля имеют ряд достоинств по производительности и реализации в микросхемах, наталкивает на продолжение исследований в области модификации вышеперечисленных алгоритмов.

Приводится блочная схема алгоритма шифрования для i-раунда (рис. 1).

Ниже приводится общее правило модификации сети Файстеля (рис. 2).

Процессор многих серий компьютеров выполняет арифметические операции 32-разрядными числами, представленными в двоичном коде. Создание и массовое применение процессоров 64-разрядных, 128-разрядных и еще больших позволяют осуществлять высокоскоростные вычисления. С учетом такого обстоятельства решается вопрос модификации алгоритмов шифрования данных, основанных на сети Файстеля, оставляя без изменения сущности преобразования, только увеличением длины ключа. При этом сеть Файстеля модифицируется следующим образом.

Здесь:

1. В модифицированном алгоритме длина блока открытого текста, подлежащего зашифрованию, $64 \cdot n$ *бит*, если основной алгоритм имеет блок зашифрования открытого текста длиной 64 *бит*.
2. Длина ключа $|K| \cdot n$ *бит*, если длина ключа основного алгоритма $|K|$ *бит*.
3. $K_i = K_i^1 K_i^2 \ldots K_i^n$ – подключи i-раунда, где в целом K_i – ключ i-раунда.
4. Длины L левой и R правой частей: $|L| = |R| = 32 \cdot n$ *бит*.
5. $L_{i-1}(32 \cdot n$ *бит*$)$ – левая часть i-раунда.
6. $R_{i-1}(32 \cdot n$ *бит*$)$ – правая часть i-раунда.
7. $L_{i-1}^1(32$ *бит*$)$, $L_{i-1}^2(32$ *бит*$),\ldots, L_{i-1}^n(32$ *бит*$)$ – левые 32-*бит*ные части i-раунда.
8. $R_{i-1}^1(32$ *бит*$)$, $R_{i-1}^2(32$ *бит*$),\ldots, R_{i-1}^n(32$ *бит*$)$ – правые 32-*бит*ные части i-раунда.
9. $F(R_{i-1}^1, K_i^1), F(R_{i-1}^2, K_i^2),\ldots, F(R_{i-1}^n, K_i^n)$ – соответствующие преобразования функции сети Фейстеля i-раунда.

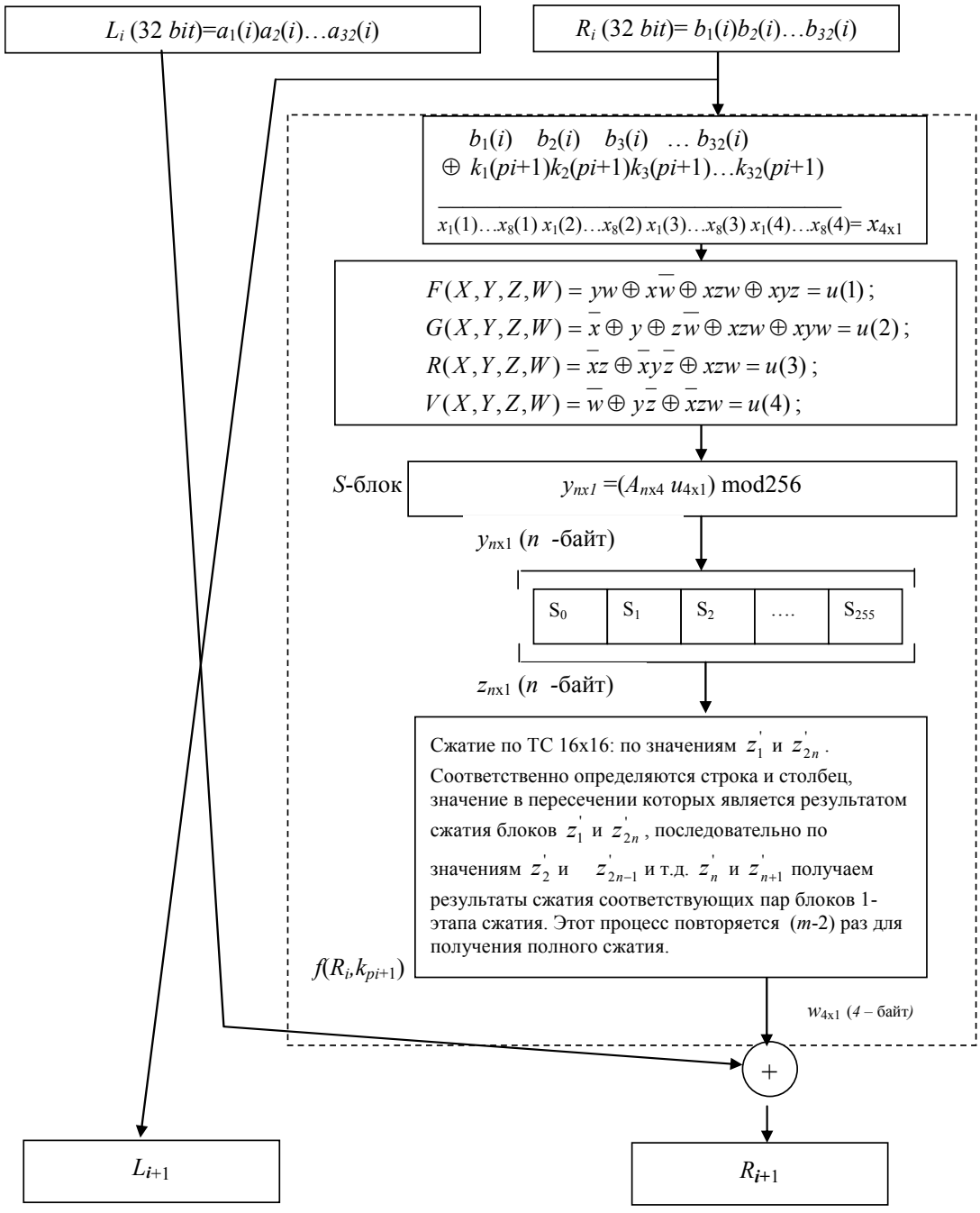

Рис. 1

Математическая модель i-раунда модифицированной сети Файстеля выражается следующим образом:

$$\begin{cases} L_i\,(32\cdot n\,бит) = R_{i-1}(32\cdot n\,бит), \\ R_i(32\cdot n\,бит) = L_{i-1}(32\cdot n\,бит) \oplus F(R_{i-1}, K_i)(32\cdot n\,бит). \end{cases} \tag{3}$$

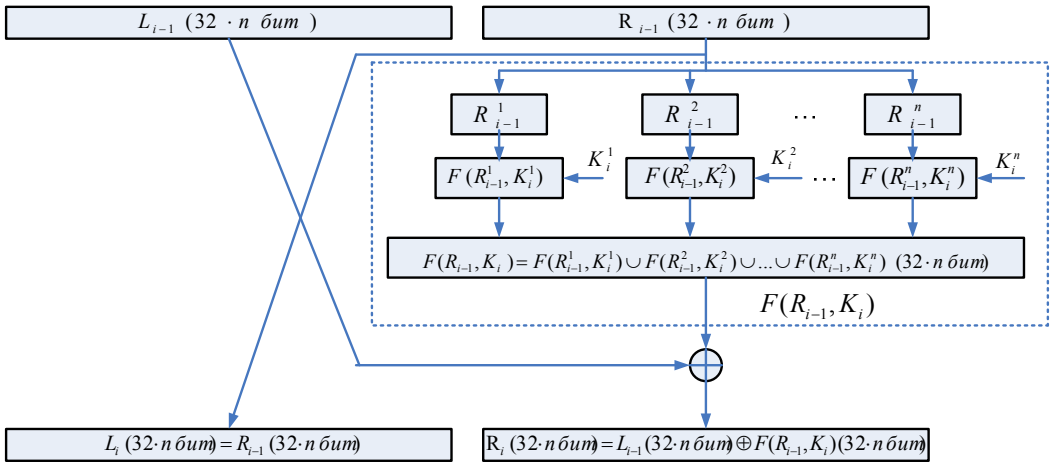

Рис. 2. *i*-раунд модифицированной сети Файстеля

Преобразования: логические функции, матричное расширение, S-блок и ТС-таблица сжатия с установленными свойствами являются новыми и используются впервые в конструкции предложенного алгоритма. Эти преобразования обладают следующими свойствами:

– логические функции являются сбалансированными, регулярными и нелинейными [4], позволяют преобразовывать блоки битов, кратные четырём;

– *S*-блок является нелинейным преобразованием, позволяет преобразовывать блоки битов, кратные восьми;

– ТС-таблица сжатия обеспечивает экспоненциальную сложность, нелинейная позволяет преобразовывать блоки битов, кратные восьми;

– побитное суммирование по mod 2 соответствующих блоков обладает свойством корреляционной иммунности, позволяет преобразовывать блоки битов любой ограниченной длины.

Таким образом, использованные преобразования позволяют осуществить удобную реализацию на современных платформах вычислительных устройств.

Анализ результатов

Отметим, что с целью улучшения криптостойкости для шифрования каждого следующего блока 256-битный ключ *k* циклический сдвигается влево на λ бит (где λ – любое нечетное число из интервала от 3 до 255). В общем случае если длина ключа равна *N*, то значение длины сдвига определяется любым взаимно-простым числом λ с *N*, чтобы иметь наибольшую гамму ключа по исходному данному ключу. При этом за счет сдвига исходного ключа каждый блок данного открытого текста шифруется разными ключами.

В предлагаемом алгоритме преобразования конструкции функции *f* сети Файстеля: сложение открытого текста с раундовым ключом по mod 2, логические преобразования, матричное преобразование по mod 256, *S*-блок и таблица сжатия являются практически необратимыми. Кроме того, преобразование по таблице сжатия нелинейное, т.е. преобразование по таблице сжатия обладает свойством неоднозначности в процессах шифрования и расшифрования. Нелинейность преобразования по таблице сжатия обеспечена за счет повторения значения эле-

ментов с равновероятными распределениями в конструкции таблицы. Преобразование по S-блоку линейное, поэтому оно генерируется от ключа по заранее определенному правилу, чтобы иметь секретное расположение в значении ячеек этого преобразования. Преобразование гаммирования битовым сложением открытого текста с раундовым ключом по mod 2 линейное, но неизвестным является данное, которое подлежит к гаммированию с неизвестным раундовым ключом. Такое обстоятельство обеспечит практическую необратимость этого преобразования. Преобразование прямоугольной матрицей по mod 256 является линейным, но в конечном поле целых чисел по mod 256 четные числа не имеют обратных элементов. Кроме того, элементы этого матричного преобразования генерируются от ключевой последовательности по заранее определенному правилу. Тем самым обеспечивается практическая необратимость предлагаемого матричного преобразования. Все эти перечисленные свойства преобразований функции *f* сети Фейстеля предлагаемого алгоритма шифрования данных обеспечат стойкость вместе с неизвестным ключом длиной 256 бита.

Предложенная модификация сети Фейстеля позволяет с успехом воспользоваться существующими алгоритмами шифрования данных, основанных на сети Фейстеля, при этом удлиняя длину ключа и шифруемого блока данных.

Заключение

На основе идеи разработки симметричных блочных алгоритмов шифрования данных, базирующихся на сети Фейстеля, предложен новый симметричный блочный алгоритм, в котором используются комбинации преобразований: побитное сложение по mod 2, матричное преобразование по mod 256 *S*-блока и таблицы сжатия. Шифрование блоков данных осуществляется с помощью mod 256. Генерация базовых преобразований с указанными свойствами со стороны (по усмотрению) группы пользователей является одним из основных преимуществ в приложениях.

Список литературы

[1] Алферов А.П., Зубов А.Ю., Кузмин А.С., Черемушкин А.В. *Основы криптографии*. М.: Гелиос АРВ, 2002, 408 с. [Alferov A.P., Zubov A.Iu., Kuzmin A.S., Cheremushkin A.V. *The basics of cryptography*. Moscow, Gelios ARV, 2002, 408 p. (in Russian)]

[2] Шнайер Б. *Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си*. М.: ТРИУМФ, 2003, 816 с. [Shneier B. *Applied cryptography. Protocols, algorithms, and source code in C++* . Moscow, Triumf, 2003, 816 p. (in Russian)]

[3] Молдавян А.А., Молдавян Н.А. *Криптография от примитивов к синтезу алгоритмов*. СПб.: БХВ-Петербург, 2004, 448 с. [Moldavian A.A., Moldavian N.A. *Cryptography: from primitives to the synthesis algorithms*. St. Petersburg, 2004, 448 p. (in Russian)]

[4] Молдавян А.А., Молдавян Н.А., Гуц Н.Д., Изотов Б.В. *Криптография: скоростные шифры*. СПб.: БХВ-Петербург, 2004, 496 с. [Moldavian A.A., Moldavian N.A., Guts N.D., Izotov B.V. *Cryptography: speed ciphers*. St. Petersburg, 2004, 496 p. (in Russian)]

[5] Акбаров Д.Е. *Ахборот хавфсизлигини таъминлашнинг криптографик усуллари ва уларнинг қўлланилиши*. Тошкент, Ўзбекистон маркаси, 2009, 432 б. [Akbarov D.E. Tashkent, 2009, 432 p. (in Uzbek)]