ISSN 0201-744X, ISSN 0321-2211

АВТОМАТИЗАЦІЯ ТА ІНТЕЛЕКТУАЛІЗАЦІЯ ПРИЛАДОБУДУВАННЯ

УДК 681.3

НОВЫЙ АЛГОРИТМ БЛОЧНОГО ШИФРОВАНИЯ ДАННЫХ С СИММЕТРИЧНЫМ КЛЮЧОМ

Акбаров Д. Е., Умаров Ш. А.
Ферганский филиал Ташкентского университета информационных технологий, г. Фергана, Узбекистан
E-mail: sht00357@gmail.com

В данной статье, продолжая идею разработки симметричных блочных алгоритмов шифрования данных, не базированных на сети Фейстеля, предлагается новый симметричный блочный алгоритм. В предложенном алгоритме используются комбинации преобразований: побитное сложение по mod2, матричное преобразование по mod256, S-блока и таблицы сжатия, осуществляет шифрование 128 (в модификации 128+32l) битовых блоков данных с помощью 128 (в модификации 128+32l) битового ключа k, где l=1,2,…, d, $d<\infty$.

Ключевые слова: *алгоритм, шифрования, S-блок преобразований байтов, таблица сжатия, матричное преобразование.*

Введение

В 2000 году Национальным институтом стандартов и технологий США (NIST) официально объявлено принятие нового стандарта алгоритма шифрования данных AES-FIPS 197 криптографической защиты для закрытия важной информации правительственного уровня на замену существующему с 1974 года алгоритму DES, считая его устаревшим по параметрам: длине ключа, удобству реализации на современных процессорах, быстродействию и др., за исключением самого главного, – стойкости. Этот алгоритм шифрования данных AES-FIPS 197 в отличие от многих других стандартов не базируется на сети Фейстеля. В этой статье, продолжая идею разработки симметричных блочных алгоритмов шифрования данных, не базированных на сети Фейстеля, авторы предлагают новый симметричный блочный алгоритм [1, 2].

Постановка задачи

Предлагаемый алгоритм, используя комбинацию преобразований: побитное сложение по mod 2, матричное преобразование по mod 256, S-блока и таблицы сжатия, осуществляет шифрование 128-битовых блоков данных (в модификации 128+32*l*) с помощью 128-битового ключа *k* (в модификации 128+32*l*).

Решение задачи. В описании предлагаемого алгоритма использованы следующие обозначения:

– T_0 – 128-разрядные (в модификации 128+32*l*) (битовые) блоки открытых данных, где *l*=1,2,…, *d*, $d<\infty$;

– $T_{\text{ш}}$–128 (в модификации 128+32*l*) -разрядные (битовые) блоки шифрованных данных;

– $t_i - i$-бит последовательности открытых данных;

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

– $xd_{4\times 1} = (xd_1, xd_2, xd_3, xd_4)$ – 32-разрядный (четыре-восьми битный) входной вектор матричного преобразования $A_{n\times 4}$, где значения байтов xd_i на интервале $0 \le xd_i \le 255$, $i = 1,2,3,4$ и $d = 1,2,...,q$, q – некоторое ограниченное число, которое определяется ниже;

– $A_{4\times 4}$ – квадратная матрица, используемая в алгоритме шифрования, которая генерируется от ключевой последовательности по заранее определенному правилу с условием, что значение определителя $\|A_{4\times 4}\|$ - нечетное число и $\|A_{4\times 4}\| \ne 0$, очевидно, она секретная, где элементы

a_{ij} $(i = 1,2,3,4;\ j = 1,2,3,4;)$ этой матрицы выражаются одним байтом, поэтому удовлетворяют условию $0 \le a_{ij} \le 255$;

– $yd_{n\times1} = (yd_1, yd_2, ..., yd_n)$ – выходной вектор результата матричного преобразования $A_{n\times4}x_{4\times1}$ по $\mathrm{mod}\,256$, т.е. $yd_{n\times4} = A_{n\times4}xd_{4\times1}(\mathrm{mod}\,256)$, где yd_i – байты, $0 \le yd_i \le 255$, i=1,2,…,n;

– $t_{4\times4}$ (в модификации $t_{4\times l}$, где $l = 1,2,...,T$, $T < \infty$) – 128 (в модификации 128+32l) – разрядный входной вектор матричного преобразования $A_{4\times4}$, где значения байтов t_{ij} на интервале $0 \le t_{ij} \le 255$, $i = 1,2,3,4;\ j = 1,2,...,l$;

– S – блок (который генерируется от ключевой последовательности по заранее определенному правилу, следовательно, он секретный) преобразования, состоящий из 256 узлов заменит $S_0, S_2, ..., S_{255}$, имеющие один байт (8 битов) входов и выходов:

S_0	S_1	S_2	…	S_{255}

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

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

– $zd_{n\times1} = (zd_1, zd_2, ..., zd_n)$ – вектор, представляющий собой результат преобразования вектора $yd_{n\times1} = (yd_1, yd_2, ..., yd_n)$ через S – блок, т.е. $zd_{n\times1} = S(yd_{n\times1})$, где zd_i – байты, $0 \le zd_i \le 255$, i=1,2,…,n;

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

q_{00}	q_{01}	…	$q_{0,15}$
q_{10}	q_{11}	…	$q_{1,15}$
…	…	…	…
$q_{15,0}$	$q_{15,1}$	…	$q_{15,15}$

– $wd_{4\times1} = (wd_1, wd_2, wd_3, wd_4)$ - 32 -разрядный (4-байтный) вектор результат сжатия.

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

Было отмечено, что квадратная матрица $A_{4\times4}$, прямоугольная матрица $A_{n\times4}$ (где $n = 2^m$, $m = 2,...,M$, $M < \infty$), S – блок и ТС генерируются от ключевой последовательности по заранее определенным правилам. Здесь приведем правила генераций этих перечисленных преобразований и раундовых ключей.

1. Матричное преобразование $A_{4\times4}$, которое имеется в количестве 16 элементов, при этом каждый элемент a_{ij} $(i = 1,2,3,4;\ j = 1,2,3,4;)$ содержит по 8 битов=1 байт, причем значение определителя должно удовлетворять следующим условиям: $\|A_{4\times4}\| \ne 0$ и $\|A_{4\times4}\|$ - нечетное число. Поскольку такая матрица генерируется в виде треугольной матрицы исходной ключевой последовательности $k = k_1k_2...k_{128}$ (в следующих модификациях $k = k_1k_2...k_{128}k_{129}...k_{128+32l}$), то поступаем таким образом:

1) последовательность $k = k_1k_2...k_{128}$ (в модификациях $k = k_1k_2...k_{128}k_{129}...k_{128+32l}$) группируется на байты:
$k1 = k_1...k_8$, $k2 = k_9...k_{16}$,..., $k16 = k_{121}...k_{128}$, в модификациях
$k1 = k_1...k_8$, $k2 = k_9...k_{16}$,..., $k16 = k_{121}...k_{128}$, $k17 = k_{129}...k_{136}$, $k18 = k_{137}...k_{144}$,..., $k[16+4l] =$
$= k_{(15+4l)8+1}...k_{(15+4l)8+8}$);

2) диагональные элементы: $a_{11}, a_{22}, a_{33}, a_{44}$ определим из первых четырех нечетными значениями элементов последовательности $k1, k2, ..., k16$ (в модификациях последовательности $k1, k2, ..., k16, k17, ..., k[16+4l]$);

3) в случае, когда в данной последовательности не имелись четыре нечетными значениями элементов, то исходный ключ k циклически сдвигается поочередно на λ=1, 2, …255 бит и повторяются пункты 1) и 2), пока не полностью определились четырех нечетными значениями диагональных элементов;

4) если еще не полностью определены диагональные элементы a_{11}, a_{22}, a_{33}, a_{44}, нечетными значениями, то вычисляются последовательность чисел по формуле $y = x^z \bmod 257$, где $z = const$, *передаваем ая вместе с ключом и* $0 \le x \le 256$, при этом в некоторой последовательности образуется совокупность чисел

$\{0 \le y \le 256 : y = x^z \bmod 257,$
$, z = const, 0 \le x \le 256\}$, так как, число $n = 257$ простое;

5) с первыми и последующими элементами γ множества $\{0 \le \gamma \le 255 : \gamma = y \bmod 256,\ 0 \le y \le 256\}$, значения которых нечетные, полностью определяются значения диагональных элементов $a_{11}, a_{22}, a_{33}, a_{44}$;

6) выше диагональные или ниже диагональные элементы, где не все элементы равны нулю, образуются из последовательности $k1, k2, ..., k16$ (в модификациях последовательности $k1, k2, ..., k16, k17, ..., k[16+4l]$) или их сдвигов на λ бит, а при необходимости можно воспользоваться элементами множества $\{0 \le \gamma \le 255 : \gamma = y \bmod 256,\ 0 \le y \le 256\}$, по заранее определенному правилу.

2. Матричное преобразование $A_{n\times 4}$ имеет в количестве $4n$ элементов, каждый элемент a_{ij} $(i = 1,..., n;\ j = 1,2,3,4;)$ содержит по 8 битов=1 байт.

При $m = 3,$ прямоугольная матрица преобразования $A_{8\times 4}$ имеет 32 элемента, 128 (в модификации 128 + 32*l*) - битный ключ k циклический сдвигается направо на λ бит и элементы a_{ij} $(i = 1,2,3,4;\ j = 1,2,3,4)$ образуются из 128 (в модификации 128+32*l*) битовой последовательности, полученной в результате сдвига, полагая первый байт a_{11}, следующий байт a_{12} и так далее тридцать второй байт a_{44}. Остальные шестнадцать элементы: $a_{51}, a_{52}, ..., a_{84}$ образуются из опять циклической, сдвинутой на λ бит 128-битовой последовательности. В случае *l=8*, все элементы матрицы $A_{8\times 4}$ образуются из $128+32\times 8=256$ битовой ключевой последовательности.

При $m = 4,$ прямоугольная матрица преобразования $A_{16\times 4}$ имеет 64 элементов, 128 (в модификации 128 + 32*l*) - битный ключ k циклический сдвигается направо на λ бит и элементы a_{ij} $(i = 1,2,3,4;\ j = 1,2,3,4)$ образуются из сдвинутой 128 битовой последовательности, полагая первый байт a_{11}, следующий байт a_{12} и так далее, тридцать второй байт a_{44}. Уже сдвинутый 128-битный ключ k, опять циклический, сдвигается направо на λ бит и элементы a_{ij} $(i = 5,...,8;\ j = 1,2,3,4)$ образуются из опять сдвинутой 128 - битной последовательности, полагая первый байт a_{51}, следующий байт a_{52} и так далее тридцать второй байт $a_{8,4}$. Далее, аналогично, продолжая циклические сдвиги направо на λ бит образуются последующие элементы: $a_{91}, a_{92}, ..., a_{16,4}$ матрицы $A_{16\times 4}$. В обоих случаях для l из интервала $1 \le l \le 7$ элементы матрицы образуются, используя (128 + 32*l*) битовой ключевой последовательности со сдвигом направо на λ бит.

По правилу, аналогичному приведенной процедуре, образуются элементы $A_{n\times 4}$ при любом $n = 2^m$, $m = 2,..., M$, $M < \infty$.

3. Генерация S – блока осуществляется следующим образом. В начале процедуры генерации *S*-блока 128 (в модификации 128 + 32*l*) - битный ключ k разделяется на 16 (в модификации 16+4*l*) байты, эти байты попарно сравниваются, в результате сравнения определяются попарно различные байты в количестве не больше чем 32-х (в модификации не больше чем 32+4*l*) байтов, с которыми начинается заполнение ячеек *S*-блока. Далее, исходный ключ k, циклический, сдвигается направо на λ =1 бит, затем, полученная последовательность разделяется на байты, и они попарно сравниваются с содержащими элементами ячеек *S*-блока, в результате, выявленными попарно различными байтами продолжается заполнение последующих ячеек *S*-блока. Далее, исходный ключ k циклический сдвигается направо на λ =2 бит, опять, байты полученной последовательности попарно сравниваются с содержащими элементами ячеек *S*-блока, выявленными попарно различными байтами продолжается заполнение последующих ячеек *S*-блока, и так далее, этот процесс продолжается, со сдвигом $\lambda = 3,4,...,255,$ 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 $z = const, 0 \le x \le 256$. В результате вычисления в некоторой последовательности образуется совокупность чисел $\{0 \le y \le 256 : y = x^z \bmod 257, z = const,$ $0 \le x \le 256\}$, поскольку число $n = 257$ простое. Элементы γ множества $\{0 \le \gamma \le 255 : \gamma = y \bmod 256,\ 0 \le y \le 256\}$ сравнивается с содержащими элементами ячеек *S*-блока, выявленными попарно различными байтами продолжается заполнение последующих ячеек *S*-блока. Таким образом, полностью заполняются все ячейки *S*-блока.

4. Генерация ТС-таблицы сжатия осуществляется следующим образом. В начале процедуры генерации 128 (в модификации 128 + 32*l*) - битный ключ k разделяется на 32 (в модификации 32+8*l*) полубайты, 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 первым элементом первой строки, если они равны, то третий полубайт сравнивается вторым полубайтом первой строки, в случае, когда в качестве второго элемента первой строки был принять второй полубайт, если они неравны, то третий полубайт принимается как третий элемент первой строки, иначе третий полубайт сравнивается первым элементом второй строки, если они не раны, то третий полубайт принимается, как вторым элементом второй строки, иначе третий полубайт принимается, как первый элемент третьей строки и так далее аналогичным определяются значения элементов таблицы. При этом заполняются не больше 32-ух ячеек таблицы. Процесс заполнения остальных ячеек полубайтами продолжается с циклическим сдвигом направо на λ=1 бит исходного ключа k, затем полученная последовательность разделяется на полубайты, и аналогично к предыдущему они сравниваются с содержащимися элементами строк таблицы, в результате, выявленными попарно различными полубайтами в строке продолжается заполнения последующих ячеек строк таблицы. Далее, исходный ключ k циклический сдвигается направо на λ=2 бит, опять, полубайты полученной последовательности попарно сравниваются с содержащимися элементами ячеек строк таблицы, выявленными попарно различными в строке полубайтами продолжается заполнение последующих ячеек строк таблицы, и так далее, этот процесс продолжается, со сдвигом $\lambda = 3,4,...,127,$ 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 $z = const$, $0 \le x \le 16$. В результате вычисления в некоторой последовательности образуется совокупность чисел $\{0 \le y \le 16 : y = x^z \bmod 17,\ z = const,\ 0 \le x \le 16\}$, поскольку число $n = 17$ простое. Элементы γ множества $\{0 \le \gamma \le 15 : \gamma = y \bmod 16,\ 0 \le y \le 16\}$ сравниваются с содержащимися элементами ячеек в строке таблицы, выявленными попарно различными полубайтами, продолжается заполнение последующих ячеек строк таблицы. Таким образом, полностью заполняются все ячейки строк таблицы сжатия ТС.

Отметим, что вместо того, чтобы генерировать квадратную матрицу $A_{4\times4}$, прямоугольную матрицу $A_{n\times4}$ (где $n = 2^m$, $m = 2,...,M$, $M < \infty$), S-блок и ТС от исходного ключа их можно генерировать заранее и передать вместе с исходным ключом, при этом эти преобразования могут генерироваться не зависимо от исходного ключа и использоваться как долговременные ключи.

При шифровании открытые данные разбивают на 128 (в модификации 128+32l)-разрядные блоки. Процедура зашифрования каждого 128 (в модификации 128+32l)-разрядного блока T_0 включает 8 раунда (цикла). В ключевое запоминающее устройство вводится 128 (в модификации 128+32l) -разрядный исходный ключ k.

5. Каждый ключ $i-$раунда k_{pi} генерируется по 128 (в модификации 128+32l) -разрядному исходному ключу $k = k_1 k_2 ... k_{128}$ (в модификации $k = k_1 k_2 ... k_{128} k_{129} ... k_{128+32l}$), циклически сдвигая его налево на $i\lambda$ бит, в результате получив $k' = k_1' k_2' ... k_{128}'$ (в модификации $k' = k_1' k_2' ... k_{128}' k_{129}' ... k'_{128+32l}$) и, разделив его на 16 (в модификации 16+4l) байта $k1 = k_1' ... k_8'$, $k2 = k_9' ... k_{16}', ..., k16 = k_{121}' ... k_{128}'$ (в модификации $k1 = k_1' ... k_8'$, $k2 = k_9' ... k_{16}', ..., k16 = k_{121}' ... k_{128}'$, $k17 = k_{129}' ... k_{136}'$, $k18 = k_{137}' ... k_{144}', ..., k[16+4l] = | = k'_{(15+4l)8+1} ... k'_{(15+4l)8+8}$):

1) полученные байты группируется на слова:

$$x1_{4\times1} = (x1_1, x1_2, x1_3, x1_4) = (k1, k2, k3, k4),$$

$$x2_{4\times1} = (x2_1, x2_2, x2_3, x2_4) = (k5, k6, k7, k8),$$

$$x3_{4\times1} = (x3_1, x3_2, x3_3, x3_4) = (k9, k10, k11, k12),$$

$$, x4_{4\times1} = (x4_1, x4_2, x4_3, x4_4) = (k13, k14, k15, k16), ...;$$

2) эти вектора $xd_{4\times1}$, где $d = 1,2,...,4+l$, преобразуются матрицей $A_{n\times4}$, т.е. $yd_{n\times1} = A_{n\times4} xd_{4\times1} (\bmod 256)$;

3) вектора $yd_{n\times1}$, $d = 1,2,...,4+l$, преобразуются через S-блока, причем по значениям $yd_1,\ldots, yd_n$ в десятичной системе исчисления $(yd_1)_{10},\ldots, (yd_n)_{10}$ определяются номера узлов преобразования в S-блоке, а результатом преобразования байтов $yd_1,\ldots, yd_n$ являются двоичные представления значения $S_{yd_1},\ldots, S_{yd_n}$, т.е. $(S_{yd_1})_2,\ldots, (S_{yd_n})_2$, соответствующих узлов $(yd_1)_{10},\ldots, (yd_n)_{10}$:

$$zd_{n\times1} = S(yd_1,..., yd_n) = (S(yd_1),..., S(yd_n)) =$$
$$= ((S_{yd_1})_2,..., (S_{yd_n})_2) = (zd_1,..., zd_n);$$

4) по таблице сжатия ТС $8\times n-$разрядный ($n-$байтный) вектора $zd_{n\times1}$, $d = 1,2,...,4+l$,

сжимается на 32-разрядный (4-байтный) вектор $w_{4\times1}=(w_1,w_2,w_3,w_4)$ следующим образом:

-каждый байт zd_i вектора $zd_{n\times1}$ разделяются на полубайты, т.е. полагается, что

$$zd_{n\times1}=(zd_1,...,zd_n)=(z^{'}d_1,...,z^{'}d_{2n})=z^{'}d_{2n\times1};$$

-по значениям полубайтов $z^{'}d_1$ и $z^{'}d_{2n}$ в десятичной системе исчислении $(z^{'}d_1)_{10}$ и $(z^{'}d_{2n})_{10}$ соответственно определяются номера строка и столбца ТС, число $q_{(z^{'}d_1)_{10}(z^{'}d_{2n})_{10}}$ (полубайт), находящееся в пересечении этих строка и столбца является результатом сжатия полубайтов $z^{'}d_1$ и $z^{'}d_{2n}$ на полубайт байт $q_{(z^{'}d_1)_{10}(z^{'}d_{2n})_{10}}$, далее, этот процесс повторяется для всех пар ($z^{'}d_2$, $z^{'}d_{2n-1}$), ($z^{'}d_3$, $z^{'}d_{2n-2}$), …, ($z^{'}d_n$, zd_{n+1}), т.е. для всех пар ($z^{'}d_i$, $z^{'}d_{2n-(i-1)}$), где i=1,…, n;

-результат сжатия, полученный на предыдущем шаге, используя ТС, аналогичным образом, подвергается к сжатию 2(m-2) раз и получается результат полного сжатия $wd_{4\times1}=(wd_1,wd_2,wd_3,wd_4)$ - 32-разрядный (4-байтный) вектор;

5) последовательность

$(w1_{4\times1},w2_{4\times1},...,w[4+l]_{4\times l})=$

$=(w1_1,w1_2,w1_3,w1_4,w2_1,w2_2,w2_3,w2_4,...$

$,w[4+l]_1,w[4+l]_2,w[4+l]_3,w[4+l]_4)=$

$=k_1(pi)\ k_2(pi)...\ k_{128}(pi),k_{129}(pi)...k_{128+32l}(pi))=k_{pi}$,

при l=0 последовательность

$$k_1(pi)\ k_2(pi)\ldots\ldots\ k_{128}(pi)=k_{pi},$$

образует раундового ключа.

Алгоритм шифрования

В начале последовательность битов блока T_0, подлежащее зашифрованию $T_0=(t_1(0),t_2(0),...,t_{128}(0))$ (в модификации $T_0=(t_1(0),t_2(0),...,t_{128}(0),...,t_{128+32l}(0))$) побитно по модулю 2 сложится 128 (в модификации 128+32l) –разрядным исходным ключом $k=k_1k_2...k_{128}$ (в модификации $k=k_1k_2...k_{128}k_{129}...k_{128+32l}$), т.е.

$$T_0\oplus k=(t_1(0),t_2(0),...,t_{128}(0))\oplus$$

$(k_1k_2...k_{128})=$

$=(t_1(0)\oplus k_1)(t_2(0)\oplus k_2)...(t_{128}(0)\oplus k_{128})=$

$$=t_1(1)t_2(1)...t_{128}(1)=T_1$$

(в модификации

$T_0\oplus k=(t_1(0),t_2(0),...,t_{32}(0),t_{33}(0),...,t_{128+32l}(0))\oplus$

$(k_1k_2...k_{128}k_{129}...k_{128+32l})=$

$=(t_1(0)\oplus k_1)(t_2(0)\oplus k_2)...(t_{128+32l}(0)\oplus k_{128+32l})=$

$=t_1(1)t_2(1)...t_{128+32l}(1)=T_1$).

На первом раунде осуществляются следующие преобразования:

1) последовательность $T_1=t_1(1)t_2(1)...t_{128}(1)$ (в модификации $T_1=t_1(1)t_2(1)...t_{128}(1)t_{129}(1)...t_{128+32l}(1)$) группируется на байты: $t1=t_1(1)...t_8(1),\ t2=t_9(1)...t_{16}(1),...,\ t16=t_{121}(1)...t_{128}(1)$ (в модификации

$t1=t_1(1)...t_8(1),\ t2=t_9(1)...t_{16}(1),...,\ t16=t_{121}(1)...t_{128}(1)$

, $t17=t_{129}(1)...t_{136}(1),\ t18=t_{137}(1)...t_{144}(1),...,\ t[16+4l]=$

$=t_{(15+4l)8+1}(1)...t_{(15+4l)8+8}(1)$);

2) из $ti-$х образуется следующая матрица

$$X_{4\times4}=\begin{pmatrix}x_{11}&x_{12}&x_{13}&x_{14}\\x_{21}&x_{22}&x_{23}&x_{24}\\x_{31}&x_{32}&x_{33}&x_{34}\\x_{41}&x_{42}&x_{43}&x_{44}\end{pmatrix}=\begin{pmatrix}t1&t5&t9&t13\\t2&t6&t10&t14\\t3&t7&t11&t15\\t4&t8&t12&t16\end{pmatrix}$$

в модификации

$$X_{4\times4+l}=\begin{pmatrix}x_{11}&x_{12}&x_{13}&x_{14}\ldots x_{1,4+l}\\x_{21}&x_{22}&x_{23}&x_{24}\ldots x_{2,4+l}\\x_{31}&x_{32}&x_{33}&x_{34}\ldots x_{3,4+l}\\x_{41}&x_{42}&x_{43}&x_{44}\ldots x_{4,4+l}\end{pmatrix}=\begin{pmatrix}t1&t5&t9&t13\ldots t[13+4l]\\t2&t6&t10&t14\ldots t[14+4l]\\t3&t7&t11&t15\ldots t[15+4l]\\t4&t8&t12&t16\ldots t[16+4l]\end{pmatrix}$$

3) матрица $X_{4\times4}$ (в модификации матрица $X_{4\times4+l}$) преобразуется квадратной треугольной матрицей $A_{4\times4}$ в конечном поле по mod 256 , т.е. $Y_{4\times4}=A_{4\times4}X_{4\times4}(\mathrm{mod}256)$ (в модификации матрица $Y_{4\times4+l}=A_{4\times4}X_{4\times4+l}(\mathrm{mod}\,256)$);

4) элементы y_{ij} матрицы $Y_{4\times4}$ (в модификации матрицы $Y_{4\times4+l}$) преобразуются через S-блока, причем по значениям y_{ij} в десятичной системе исчисления $(y_{ij})_{10}$, определяются номера узлов преобразования в S-блоке, а результатом преобразования байтов y_{ij} являются двоичные представления значения $S_{y_{ij}}$, т.е. $(S_{y_{ij}})_2$, соответствующих узлов $(y_{ij})_{10}$: $z_{ij}=S(y_{ij})=(S_{y_{ij}})_2$;

5) последовательность битов $z_1z_2...z_{128}$ (в модификации $z_1z_2...z_{128}z_{129}...z_{128+32l}$) образованного из байтовой последовательности $z_{11}z_{21}z_{31}z_{41}z_{21}z_{22}z_{23}z_{24}...z_{44}$ (в модификации $z_{11}z_{21}z_{31}z_{41}z_{21}z_{22}z_{23}z_{24}...z_{44}z_{51}...z_{(16+4l)}$) побитно сложится по mod 2 с ключом первого раунда $k_{p1}=$ $k_1(p1)\ k_2(p1)\ldots\ldots\ k_{128}(p1)$ (в модификации $k_{p1}=k_1(p1)\ k_2(p1)...\ k_{128}(p1),k_{129}(p1)...k_{128+32l}(p1)$), т.е. $z_1z_2...z_{128}\oplus k_1(p1)\ k_2(p1)\ldots\ldots\ k_{128}(p1)=$

$(z_1\oplus k_1(p1))(z_2\oplus k_2(p1))...(z_{128}\oplus k_{128}(p1))=$

$$=t_1(2)t_2(2)...t_{128}(2)=T_2$$

(в модификации
$z_1 z_2 ... z_{128} z_{129} ... z_{128+32l} \oplus k_1(p1)\, k_2(p1) ... k_{128}(p1)$
$k_{129}(p1) ... k_{128+32l}(p1)$
$= (z_1 \oplus k_1(p1))(z_2 \oplus k_2(p1)) ... (z_{128} \oplus k_{128}(p1))$
$(z_{129} \oplus k_{129}(p1)) ... (z_{128+32l} \oplus k_{128+32l}(p1)) =$
$= t_1(2) t_2(2) ... t_{128}(2)\, t_{129}(2) ... t_{128+32l}(2) = T_2$)

Выше приведенные пункты 1)-5) преобразования блока данных T_1 в целом представляет 1-раунд преобразования предлагаемого алгоритма шифрования.

Полагая, что $T_1 = T_2$, и $k_{p1} = k_{p2}$, затем, аналогичным образом, повторяя пункты преобразования 1-5, осуществляется 2-раунд преобразования алгоритма. Таким образом, если результат преобразования $(i-1)$ – раунда получен, то полагая $T_1 = T_{i-1}$, и $k_{p1} = k_{pi-1}$, затем, повторяя пункты преобразования 1)-5) осуществляется *i*-раунд алгоритма шифрования. Количество раундов предлагаемого алгоритма шифрования равно 8, т.е. *i*=1,2,…,8.

Ниже приводится блочные схемы описания генерации ключей (рис. 1) и алгоритма шифрования для $i-$раунда (рис. 2), также общая блок-схема алгоритма расшифрования (рис. 3).

Рис. 1. Блок-схема генерации ключа $i-$раунда

Алгоритм расшифрования осуществляется следующим образом:

1. Из исходного ключа k по выше приведенным правилам генерируются: матрицы $A_{4\times4}$ и $A_{n\times4}$, S – блок, а также раундовые ключи k_{pi}, $i = 1,2,...,8$.

2. Вычисляется обратная матрица

$$A^{-1}_{4\times4} = \|A\|^{-1} \begin{vmatrix} A_{11} & A_{21} & A_{31} & A_{41} \\ A_{12} & A_{22} & A_{32} & A_{42} \\ A_{13} & A_{23} & A_{33} & A_{43} \\ A_{14} & A_{24} & A_{34} & A_{44} \end{vmatrix}$$

которая существует и она треугольная, где A_{ij} – алгебраические дополнения (миноры со знаками) соответствующих элементов a_{ij} (*i, j*=1,2,…,4).

3. На шифртекст $T_ш = T_9 = t_1(9) t_2(9) ... t_{128}(9)$ (в модификации
$T_ш = T_9 = t_1(9) t_2(9) ... t_{128}(9) t_{128}(9) ... t_{128+32l}(9)$)
побитно по mod 2 сложится раундовый ключ k_{p8}, в результате имеется $T_9 \oplus k_{p8} = Z^8_{4\times4}$.

4. Элементы $z_{11} z_{21} z_{31} z_{41} z_{21} z_{22} z_{23} z_{24} ... z_{44}$ (в модификации $z_{11} z_{21} z_{31} z_{41} z_{21} z_{22} z_{23} z_{24} ... z_{44} z_{51} ... z_{(16+4l)}$) матрицы $Z^8_{4\times4}$ подвергаются на обратное преобразование по S-блоку, т.е. в S-блоке находится ячейка значением равным со значением элемента z_{ij}, затем по порядковому номеру этой ячейки определяется значение y_{ij} матрицы $Y^8_{4\times4}$ обратного преобразования S-блока, представляя порядкового номера в двоичной системе исчислении в битах.

5. В результате преобразования матрицу $Y^8_{4\times4}$ с обратной матрицей $A^{-1}_{4\times4}$, получается матрица $X^8_{4\times4}$, последовательность битового представления элементов x_{ij} которой образует элементов (битов) блока T_8, т.е.

$$A^{-1}_{4\times4}\ Y^8_{4\times4} = A^{-1}_{4\times4} A_{4\times4} X^8_{4\times4} = X^8_{4\times4} = T_8 .$$

Здесь отметим, что приведенные пункты 3-5 преобразования блока шифртекста $T_ш = T_9$ в целом представляет 1-раунд алгоритма расшифрования.

Полагая, что $T_9 = T_8$, и $k_{p8} = k_{p7}$, затем, аналогичным образом повторяя пункты преобразования 3-5, осуществляется 2-раунд алгоритма расшифрования. Таким образом, если результат T_{9-i} *i*-раунда алгоритма расшифровавния получен, то полагая $T_9 = T_{9-i}$, и $k_{p8} = k_{8-pi}$, затем, повторяя пункты

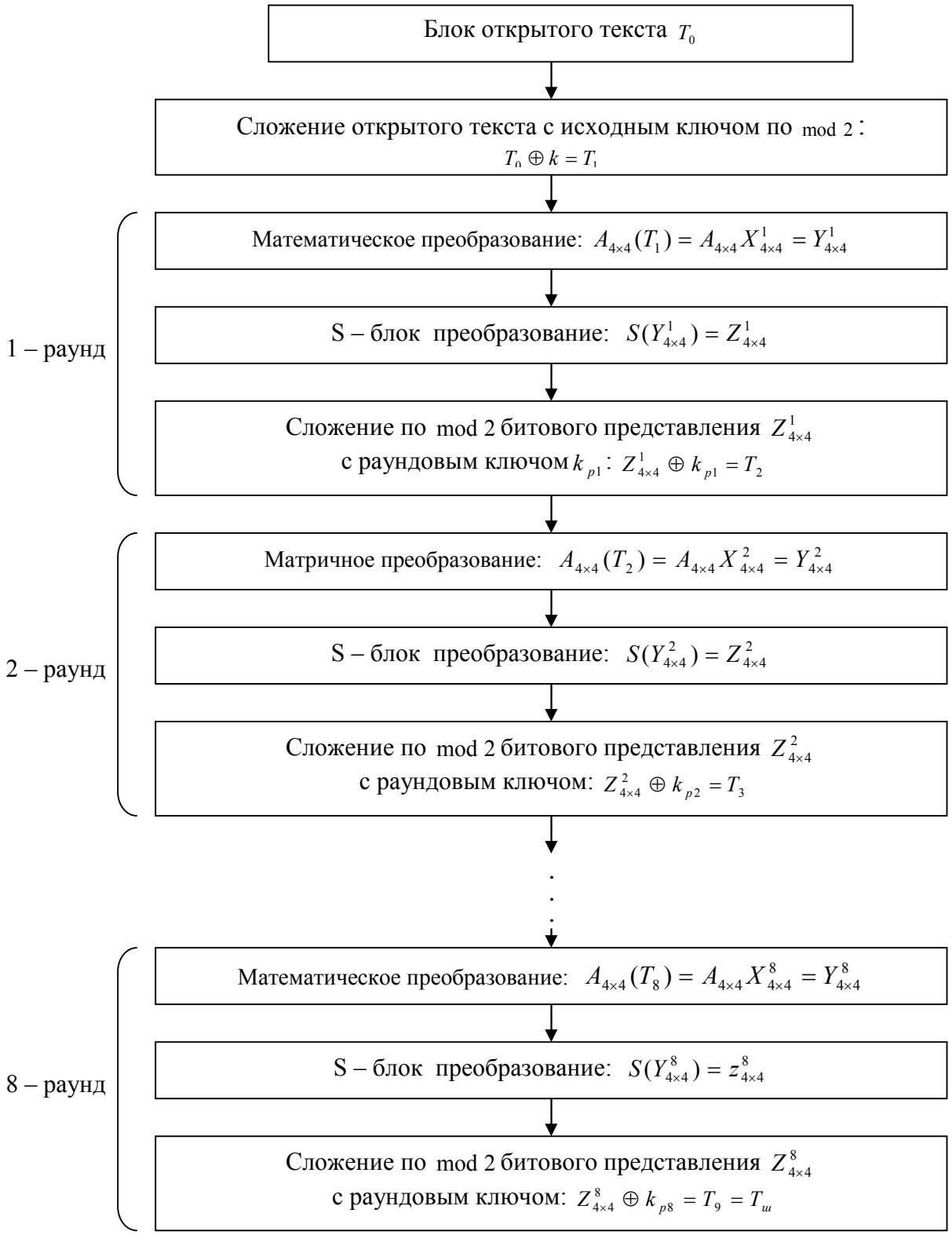

Рис. 2. Блок-схема алгоритма шифрования данных

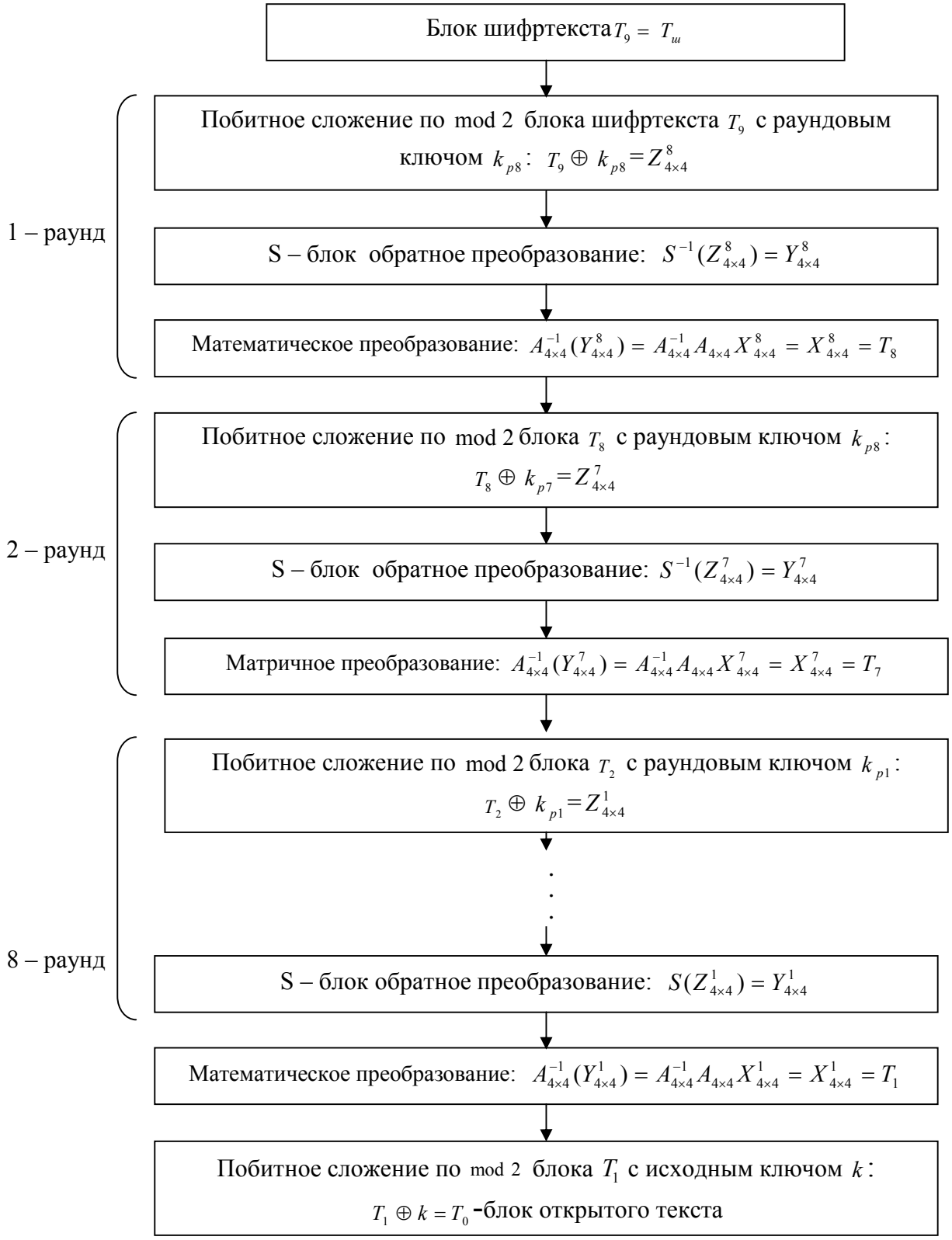

Рис. 3. Блок схема алгоритма расшифрования блока шифртекста

преобразования 3-5, осуществляется (i+1)-раунд алгоритма расшифрования, где i=0,1,2,…,7. Повторяя пункты 3-5 преобразования алгоритма расшифрования для блока шифртекста $T_ш = T_9$ при i=0,1,2,…,7, получается блок T_1.

6. На последовательность битов блока T_1 побитно сложится по mod 2 последовательность битов исходного ключа k, в результате получится блок открытого текста T_0, т.е.

$$T_1 \oplus k = T_0.$$

Пункты 1-6 в целом описывает алгоритм расшифрования.

Анализ результатов

С целью улучшения криптостойкости, для шифрования каждого следующего блока T_0 исходный ключ k циклический сдвигается влево на λ бит (где λ - любое нечетное число). В общем случае, если длина ключа равно 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: сложение открытого текста с исходным ключом и промежуточных данных шифрования с раундовыми ключами по mod 2, матричное преобразование по mod 256, S-блок преобразование и таблица сжатия, используемая в генерации раундовых ключей, 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 элементов с равновероятными распределениями в конструкции таблицы. Преобразование по S-блоку линейное, поэтому оно генерируется от ключа по заранее определенному правилу, чтобы обеспечить секретное расположение значения ячеек этого преобразования. Преобразование гаммирования битовым сложением открытого текста с исходным ключом и промежуточных данных шифрования с раундовым ключом по mod 2 линейные, но неизвестными являются данные, которые подлежат к гаммированию с неизвестным исходным и раундовым ключами. Такое обстоятельство обеспечит практической необратимости этих преобразований. Преобразование прямоугольной матрицей по mod 256 является линейным, но в конечном поле целых чисел по mod 256 четные числа не имеют обратные элементы. Кроме того, элементы этого матричного преобразования генерируется от ключевой последовательности по заранее определенному правилу. Тем самым обеспечивается практическая необратимость предлагаемого матричного преобразования. Все эти перечисленные свойства преобразований предлагаемого алгоритма шифрования данных обеспечит стойкости вместе с неизвестным ключом длиной 128 (в модификации 128+32l) бита.

Выводы

На основе идеи разработки симметричных блочных алгоритмов шифрования данных, не базированных на сети Фейстеля, предложен новый симметричный блочный алгоритм, в котором используется комбинация преобразований: побитное сложение по mod 2, матричное преобразование по mod 256, S-блока и таблицы сжатия, осуществляет шифрование 128 (в модификации 128+32l) битовых блоков данных с помощью 128 (в модификации 128+32l) битового ключа k, где l=1,2,…, d, $d<\infty$.

Генерация базовых преобразования с указанными свойствами со стороны (по усмотрению) группы пользователей является одним из основных преимуществ для прикладных задач.

Литература

1. Зензин О. С., Иванов М. А. Стандарт криптографической защиты – AES. Конечные поля / О. С. Зензин, М. А. Иванов; под ред. М. А. Иванова. – М.: КУДИЦ-ОБРАЗ, 2002. – 176 с.

2. Акбаров Д. Е. Ахборот хавфсизлигини таъминлашнинг криптографик усуллари ва уларнинг қўлланилиши – Тошкент, “Ўзбекистон маркаси”, 2009. – 432 б.

УДК 681.3

Д. Є. Акбаров, Ш. А. Умаров

Ферганська філія Ташкентського університету інформаційних технологій, м. Фергана, Узбекистан

НОВИЙ АЛГОРИТМ БЛОЧНОГО ШИФРУВАННЯ ДАНИХ З СИМЕТРИЧНИМ КЛЮЧЕМ

У цій статті, продовжуючи ідею розробки симетричних блокових алгоритмів шифрування даних, що не базовані на мережі Фейстеля, пропонується новий симетричний блоковий алгоритм.

У запропонованому алгоритмі використовуються комбінації перетворень: побітного додавання по mod2, матричне перетворення по mod256, S-блоку і таблиці стиснення, здійснює шифрування 128 (в модифікації 128 + 32l) бітових блоків даних за допомогою 128 (в модифікації 128 + 32l) бітового ключа k , де l = 1,2, ..., d, d <∞.
Ключові слова: алгоритм, шифрування, S-блок перетворень байтів, таблиця стиснення, матричне перетворення.

D. E. Akbarov, Sh. A. Umarov
Fergana branch of the Tashkent university information technologies, Fergana, Uzbekistan
NEW ALGORITHM OF BLOCK ENCRYPTION OF DATA WITH THE SYMMETRIC KEY
In this article, continuing idea of development of the symmetric block data encryption algorithms which aren't based on Feystel's network the new symmetric block algorithm is offered.
In the offered algorithm are used a combination of conversions: bit-by-bit addition on mod2, matrix transformation on mod256, the S-unit and the table of compression, realizes encoding 128 (in modification 128+32l) bit data units by means of 128 (in modification 128+32l) a bit key of k, where l=1,2,…, d, d<∞.
Keywords: algorithm, encodings, S-block conversions of bytes, table of compression, matrix transformation.

Надійшла до редакції
26 квітня 2016 року

Рецензовано
11 травня 2016 року

УДК 681.7.067

АВТОМАТИЗОВАНИЙ РОЗРАХУНОК ШИРОКОКУТНИХ ОБ'ЄКТИВІВ

Сокуренко В. М., Буйлов І. С.
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», м. Київ, Україна

E-mail: sokurenko2@meta.ua, enekotrade@gmail.com

Однією із затребуваних задач сучасної обчислювальної оптики є задача повної автоматизації процедури знаходження параметрів оптичної системи (ОС), оптимальних за різними критеріями. Широко розповсюджені методи локальної оптимізації мають суттєвий недолік, який полягає в тому, що для їх результативного застосування, як правило, потрібна якісна вихідна ОС. Поставлену задачу пошуку найменшого значення оціночної функції в заданому просторі допустимих значень параметрів потенційно дозволяють вирішити методи глобальної оптимізації (ГО). У зв'язку з цим стосовно автоматизованого розрахунку ОС різними авторами було запропоновано використовувати такі методи ГО як імітаційний відпал, генетичні алгоритми тощо. У даній роботі чисельно досліджуються можливості адаптивного методу диференційної еволюції Коші, запропонованого в 2013 році. Алгоритм цього методу відрізняється наявністю внутрішнього механізму адаптації параметрів-коефіцієнтів класичного методу диференційного еволюції, а також застосуванням розподілу Коші для генерації нових «випадкових» значень цих коефіцієнтів. Зазначений вище алгоритм був реалізований у власній комп'ютерній програмі автоматизованого розрахунку ОС довільного призначення. В роботі здійснена експериментальна перевірка ефективності реалізації вибраного методу під час розрахунку аналогів ОС, запропонованих нещодавно в патентах США.

Ключові слова: *об'єктив, оптична система, адаптивний метод диференційної еволюції Коші, автоматизований розрахунок, глобальна оптимізація.*

Вступ

На відміну від широко поширених методів локальної оптимізації, істотним недоліком яких є "застрягання" в першому ж знайденому мінімумі,