УДК 519.719.1, 681.322

Ткаченко В. Г., Синявский О. В.
Ткаченко В. Г., Сінявський О. В.
Tkachenco V. G, Sinyavsky O. V.

ПОСТРОЕНИЕ КОРРЕКТИРУЮЩЕГО КОДА ДЛЯ КРИПТОСИСТЕМ НА ОСНОВЕ ТИПОВ МОНОТОННЫХ БУЛЕВЫХ ФУНКЦИЙ

ПОБУДОВА КОРЕКТУЮЧОГО КОДУ ДЛЯ КРИПТОСИСТЕМ НА ОСНОВІ ТИПІВ МОНОТОННИХ БУЛЬОВИХ ФУНКЦІЙ

CONSTRUCTION OF ERROR CORRECTING CODE FOR CRYPTOSYSTEMS ON THE BASIS OF TYPES MONOTONOUS BOOLEAN FUNCTIONS

Аннотация. *В статье рассмотрена возможность применения максимальных типов монотонных булевых функций (МБФ) при построении кода, исправляющего ошибки при передаче информации в криптосистемах. Выведен критерий определения максимального типа МБФ. Получены матричные уравнения для рекуррентного вычисления распределения количества типов максимального ранга. Используя эти уравнения, можно строить матрицы распределения больших типов МБФ. С помощью них можно строить корректирующий код.*

Анотація. *У статті розглянута можливість застосування максимальних типів монотонних бульових функцій (МБФ) при побудові коду, що виправляє помилки при передаванні інформації в криптосистемах. Виведено критерій визначення максимального типу МБФ. Отримані матричні рівняння для рекурентного обчислення розподілу кількості типів максимального рангу. Використовуючи ці рівняння, можна будувати матриці розподілу великих типів МБФ. За допомогою їх можна будувати коректуючий код.*

Summary. *In article possibility of application of the maximum types monotonous Boolean functions (MBF) is considered at construction of the code correcting errors at an information transfer in cryptosystems. The criterion of definition of maximum type MBF is deduced. The matrix equations for recurrent calculation of distribution of quantity of types of the maximum rank are received. Using these equations it is possible to build matrixes of distribution of big types MBF. By means of them it is possible to build error correcting code.*

В процессе передачи информации в криптосистемах по телекоммуникационным сетям связи возникает проблема исправления ошибок. Контроль целостности данных и исправление ошибок — важные задачи на многих уровнях работы с информацией. Одним из средств решения этих задач является применение помехоустойчивых кодов, лежащих в основе систем кодирования-декодирования.

К настоящему времени разработано много различных помехоустойчивых кодов для криптосистем, отличающихся друг от друга основанием, расстоянием, избыточностью, структурой, функциональным назначением, энергетической эффективностью, корреляционными свойствами, алгоритмами кодирования и декодирования, формой частотного спектра ([1] … [4]). На основе максимальных типов [5] монотонных булевых функций (МБФ) построена криптосистема [6] с кодами, использующими деревья разложения типов МБФ.

Однако код для криптосистемы, используемый в [6], не предусматривает коррекцию ошибок.

Целью настоящей работы является построение кода, исправляющего ошибки при передаче информации в криптосистемах, на основе типов МБФ.

1. Основные понятия. В этой статье используются термины, введенные в работе [7], а именно: вектор $T = (a_0, a_1, ..., a_i, ..., a_n)$ является типом МБФ, если *i*-я компонента вектора a_i равна числу минимальных входных наборов данной МБФ, лежащих на слое $n - i$ булевого куба ранга n. Число n является рангом типа T, число v ненулевых компонент - весом типа T; номер i первой слева ненулевой компоненты – левой границей типа T; номер j первой справа ненулевой компоненты - правой границей типа T; сумма m всех компонент типа T − мощностью типа. Тип T называется максимальным, если при увеличении любой его компоненты на 1 полученный вектор не будет являться типом. Обозначения $K1(n)$, $K2(n, v)$, $K3(n, i, j)$, $K4(n, v, i, j)$, $K5(n, i)$, $K6(n, j)$, $K7(n, v, i)$, $K8(n, v, j)$ используются для количества максимальных типов, имеющих указанный ранг, вес, левую и правую границы.

Для удобного представления $K3(n, i, j)$ используются матрицы распределения максимальных типов ранга n [7]. В них на пересечении строки i и столбца j находится элемент $K3(n, i, j)$. Сумма элементов строки i равна $K5(n, i)$, сумма элементов столбца j равна $K6(n, j)$ и сумма элементов всей матрицы равна $K1(n)$. На рис. 1 показаны такие матрицы для рангов 5, 6 и 7.

	0	1	2	3	4	5
0	1	0	0	0	0	0
1	0	1	3	4	1	0
2	0	0	1	5	4	0
3	0	0	0	1	3	0
4	0	0	0	0	1	0
5	0	0	0	0	0	1

	0	1	2	3	4	5	6
0	1	0	0	0	0	0	0
1	0	1	4	10	9	1	0
2	0	0	1	9	24	9	0
3	0	0	0	1	9	10	0
4	0	0	0	0	1	4	0
5	0	0	0	0	0	1	0
6	0	0	0	0	0	0	1

	0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0
1	0	1	5	20	43	25	1	0
2	0	0	1	14	84	143	25	0
3	0	0	0	1	19	84	43	0
4	0	0	0	0	1	14	20	0
5	0	0	0	0	0	1	5	0
6	0	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0	1

Рисунок 1 − Матрицы распределения типов ранга 5, 6 и 7

Такие матрицы обозначим M_n, где n – ранг типа, порядок такой матрицы есть $(n+1)$. На рис. 1 представлены M_5, M_6 и M_7.

Для любых двух векторов V_1, V_2 ранга n и таких, у которых правая граница $j(V_1)$ строго меньше левой границы $i(V_2)$, определена операция сдвиг-суммы [5]:

$$V = V_1 \oplus V_2 = (a_0, a_1, ..., a_n) \oplus (b_0, b_1, ..., b_n) = (b_0, a_0 + b_1, ..., a_{n-1} + b_n, a_n) = (c_0, c_1, ..., c_{n+1})$$

Правой частью V_2 вектора V [6], у которого компонента n равна нулю, называется вектор с компонентами с нулевой по $(n-1)$, у которого компоненты с $(i+1)$ по $(n-1)$ вектора V_2 совпадают с соответствующими компонентами вектора V, а i компонента вектора V_2 больше нуля и меньше или равна i-й компоненты вектора V (при этом i будет левой границей вектора V_2). Левой частью V_1 вектора V, у которого компонента 0 равна нулю, называется вектор с компонентами с нулевой по $(n-1)$, у которого компоненты с нулевой по $(j-1)$ вектора V_1 совпадают с компонентами с первой по j вектора V, а j компонента вектора V_1 больше нуля и меньше или равна $(j+1)$ компоненты вектора V (при этом j будет правой границей вектора V_1). Максимальной правой (левой) частью вектора n ранга называется его правая (левая) часть, являющаяся типом ранга $(n-1)$ наибольшей возможной мощности. Остатком вектора V ранга $(n+1)$ после выделения правой части V_2 называется вектор $(n+1)$ ранга, получающийся вычитанием компонент с нулевой по $(n-1)$ вектора V_2 из соответствующих компонент вектора V. Если нулевая компонента остатка равна нулю, то, отбрасывая эту нулевую компоненту из остатка, получаем вектор n-го ранга D_1, который назовём дополнением правой части V_2. Дополнение D_1 является левой частью вектора V и $D_1 \oplus V_2 = V$. Остатком вектора V ранга $(n+1)$, после выделения левой части V_1, называется вектор $(n+1)$ ранга, получающийся вычитанием компонент с нулевой по $(n-1)$ вектора V_1 из компонент с 1 по n-ю вектора V. Если n-я компонента остатка равна нулю, то, отбрасывая эту n-ю компоненту из остатка, получаем вектор n-го ранга D_2, который назовём дополнением левой части V_1. Дополнение D_2 является правой частью вектора V и $V_1 \oplus D_2 = V$.

Пусть в качестве кодовых последовательностей берутся все максимальные типы ранга n. Выведем критерий, определяющий, в каком случае полученная кодовая последовательность не содержит ошибок.

ЛЕММА 1. Вектор $(a_0, a_1, ..., a_n)$ является кодовой последовательностью без ошибок тогда и только тогда, когда его правая (левая) максимальная часть и её дополнение являются максимальными типами.

Доказательство. Пусть вектор является кодовой последовательностью без ошибок. Следовательно, он является максимальным типом и по теореме 2 из [6] разлагается на 2 максимальных типа меньшего ранга. Тогда его правая (левая) максимальная часть и её дополнение являются максимальными типами.

Пусть максимальная правая (левая) максимальная часть и её дополнение являются максимальными типами. Тогда, пользуясь следствием 1 теоремы 2 из [5] о соответствии допустимых пар из максимальных типов ранга $(n-1)$ максимальным типам ранга n, сдвиг-сумма этих правой (левой) части и её дополнения даст максимальный тип n-го ранга. Следовательно, этот вектор является кодовой последовательностью без ошибок. Лемма доказана.

Используя равенство, полученное в лемме 5 из [5]

$$K3(n,i,j) = \sum_{q=i}^{j} K3(n-1, i-1, q-1) \sum_{t=q}^{j} K3(n-1, t, j), \tag{1}$$

выведем матричную формулу для построения матрицы M_n ранга типа n на основе матрицы M_{n-1} ранга типа $(n-1)$.

Введём две операции. Операция «звёздочка слева» обозначает операцию добавления строки сверху и столбца слева таких: $\begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & & & \\ \dots & & & \\ 0 & & & \end{pmatrix}$, а операция «звёздочка справа» обозначает операцию добавления строки снизу и столбца справа таких: $\begin{pmatrix} & & & 0 \\ & & & \dots \\ & & & 0 \\ 0 & \dots & 0 & 1 \end{pmatrix}$. Обозначим через I_n квадратную матрицу, порядка $(n+1)$, вида: $I_n = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 0 & 1 & 1 & 1 & \dots & 1 \\ 0 & 0 & 1 & 1 & \dots & 1 \\ 0 & 0 & 0 & 1 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots & 1 \\ 0 & 0 & 0 & 0 & & 1 \end{pmatrix}$, т.е. она получается из верхней треугольной матрицы порядка n, состоящей из единиц, с помощью операции звёздочка слева. Обозначим через ** операцию транспонирования матрицы относительно побочной диагонали, т.е. $I_n{*}{*} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 & 0 \\ 0 & 1 & 1 & \dots & 1 & 0 \\ 0 & 0 & 1 & \dots & 1 & 0 \\ \dots & \dots & \dots & \dots & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$.

Обозначим через T_n верхнюю треугольную матрицу порядка $n+1$ из единиц, тогда $I_n = {*}T_{n-1}$, а $I_n{*}{*} = T_{n-1}{*}$.

ЛЕММА 2. Справедливо рекуррентное матричное уравнение:

$$M_n = {*}M_{n-1} \times (I_{n-1} \times M_{n-1}){*}, \tag{2}$$

где операция × - обычное умножение матриц.

Доказательство. Легко проверить, что элемент с индексами (q, j) во внутренней сумме произведения матриц $I_{n-1} \times M_{n-1}$ равен $\sum_{t=q}^{j} K3(n-1, t, j)$. Отсюда, элемент матрицы M_n с индексами (i, j), т.е. $K3(n, i, j)$ равен $\sum_{q=i}^{j} K3(n-1, i-1, q-1) \sum_{t=q}^{j} K3(n-1, t, j)$. Здесь во внешней сумме индексы $(i-1, j-1)$ соответствуют строке *i* и столбцу *j* матрицы M_{n-1} сдвинутыми вправо на 1 из-за операции звёздочка слева. Лемма доказана.

Следствие. Используя верхнюю треугольную матрицу из единиц T_{n-2} размерности $(n-1)\times(n-1)$, из (2) получим

$$M_n = {}^*M_{n-1} \times ({}^* T_{n-2} \times M_{n-1})^*. \tag{3}$$

Пример 1. Получим матрицу M_6 из матрицы M_5.

$$I_5 \times M_5 = \begin{pmatrix} 1&0&0&0&0&0 \\ 0&1&1&1&1&1 \\ 0&0&1&1&1&1 \\ 0&0&0&1&1&1 \\ 0&0&0&0&1&1 \\ 0&0&0&0&0&1 \end{pmatrix} \times \begin{pmatrix} 1&0&0&0&0&0 \\ 0&1&3&4&1&0 \\ 0&0&1&5&4&0 \\ 0&0&0&1&3&0 \\ 0&0&0&0&1&0 \\ 0&0&0&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&0&0 \\ 0&1&4&10&9&1 \\ 0&0&1&6&8&1 \\ 0&0&0&1&4&1 \\ 0&0&0&0&1&1 \\ 0&0&0&0&0&1 \end{pmatrix},$$

$${}^*M_5 = \begin{pmatrix} 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0 \\ 0&0&1&3&4&1&0 \\ 0&0&0&1&5&4&0 \\ 0&0&0&0&1&3&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1 \end{pmatrix},$$

$$(I_5 \times M_5)^* = \begin{pmatrix} 1&0&0&0&0&0&0 \\ 0&1&4&10&9&1&0 \\ 0&0&1&6&8&1&0 \\ 0&0&0&1&4&1&0 \\ 0&0&0&0&1&1&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1 \end{pmatrix},$$

$$M_6 = {}^*M_5 \times (I_5 \times M_5)^* =$$

$$= \begin{pmatrix} 1&0&0&0&0&0&0 \\ 0&1&0&0&0&0&0 \\ 0&0&1&3&4&1&0 \\ 0&0&0&1&5&4&0 \\ 0&0&0&0&1&3&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1 \end{pmatrix} \times \begin{pmatrix} 1&0&0&0&0&0&0 \\ 0&1&4&10&9&1&0 \\ 0&0&1&6&8&1&0 \\ 0&0&0&1&4&1&0 \\ 0&0&0&0&1&1&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&0&0&0 \\ 0&1&4&10&9&1&0 \\ 0&0&1&9&24&9&0 \\ 0&0&0&1&9&10&0 \\ 0&0&0&0&1&4&0 \\ 0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&1 \end{pmatrix}.$$

ЛЕММА 3. Справедливо рекуррентное матричное уравнение:

$$M_n = *\left(M_{n-1} \times I_{n-1} **\right) \times M_{n-1} *. \tag{4}$$

Доказательство. Из (1), меняя порядок суммирования, имеем:

$$K3(n,i,j) = \sum_{q=i}^{j} K3(n-1,i-1,q-1) \sum_{t=q}^{j} K3(n-1,t,j) = \sum_{q=i}^{j} \sum_{t=q}^{j} K3(n-1,t,j) K3(n-1,i-1,q-1) =$$

$$\sum_{t=i}^{j} \sum_{q=i}^{t} K3(n-1,t,j) K3(n-1,i-1,q-1) = \sum_{t=i}^{j} K3(n-1,t,j) \sum_{q=i}^{t} K3(n-1,i-1,q-1).$$

Далее, действуя совершенно аналогично доказательству леммы 2, получим утверждение этой леммы. Лемма доказана.

Следствие 1. Для количества максимальных типов $K3(n,i,j)$ справедливо выражение:

$$K3(n,i,j) = \sum_{t=i}^{j} K3(n-1,t,j) \sum_{q=i}^{t} K3(n-1,i-1,q-1), \tag{5}$$

Следствие 2. Используя верхнюю треугольную матрицу из единиц T_{n-2} размерности $(n-1)\times(n-1)$, из (4), получим

$$M_n = *\left(M_{n-1} \times T_{n-2} *\right) \times M_{n-1} *. \tag{6}$$

2. Построение корректирующего кода. По определению все максимальные типы несравнимы, следовательно, если мы каждую компоненту максимального типа представим в виде двоичного числа заданной длины, то кодовое расстояние между любыми двумя типами не может быть меньше двух. Используя это свойство, можно упростить построение кодов с минимальным кодовым расстоянием больше двух, за счёт сокращения базового множества двоичных последовательностей, из которых эти коды строятся.

С ростом ранга количество типов МБФ сильно возрастает, поэтому используем не все, а только некоторые компоненты типа МБФ. Это соответствует одной клетке в таблице на рис. 1. Запишем каждую компоненту в двоичном виде и количество бит для каждой компоненты выберем равное числу бит, с помощью которых можно записать максимальную компоненту среди всех компонент соответствующих этому номеру. Например, возьмём тип ранга 8, левая граница равна 3, а правая граница равна 4. Используя лемму 2, построим матрицу M_8, на пересечении строки 3 и столбца 4 стоит число 34, следовательно, таких максимальных типов всего 34: (0,0,0,1,65,0,0,0,0); (0,0,0,2,61,0,0,0,0); (0,0,0,3,58,0,0,0,0); (0,0,0,4,56,0,0,0,0); (0,0,0,6,55,0,0,0,0); (0,0,0,7,51,0,0,0,0); (0,0,0,8,48,0,0,0,0); (0,0,0,9,46,0,0,0,0); (0,0,0,11,45,0,0,0,0); (0,0,0,12,42,0,0,0,0); (0,0,0,13,40,0,0,0,0); (0,0,0,15,39,0,0,0,0); (0,0,0,16,37,0,0,0,0); (0,0,0,18,36,0,0,0,0); (0,0,0,21,35,0,0,0,0); (0,0,0,22,31,0,0,0,0); (0,0,0,23,28,0,0,0,0); (0,0,0,24,26,0,0,0,0); (0,0,0,26,25,0,0,0,0); (0,0,0,27,22,0,0,0,0); (0,0,0,28,20,0,0,0,0); (0,0,0,30,19,0,0,0,0); (0,0,0,31,17,0,0,0,0); (0,0,0,33,16,0,0,0,0); (0,0,0,36,15,0,0,0,0); (0,0,0,37,12,0,0,0,0); (0,0,0,38,10,0,0,0,0); (0,0,0,40,9,0,0,0,0); (0,0,0,41,7,0,0,0,0); (0,0,0,43,6,0,0,0,0); (0,0,0,46,5,0,0,0,0); (0,0,0,47,3,0,0,0,0); (0,0,0,49,2,0,0,0,0); (0,0,0,52,1,0,0,0,0).

А сами максимальные типы ранга 8 получаем из максимальных типов ранга 7 рекуррентным способом. Например,

$$(0,0,0,1,65,0,0,0,0) = (0,0,1,30,0,0,0,0) \oplus (0,0,0,0,35,0,0,0).$$

Построим двоичное представление полученных типов. Максимальное число для третьей компоненты 52, двоичное представление: 110100 состоит из 6 бит. Значит, для записи этой компоненты используем 6 бит. Соответственно, для четвёртой компоненты максимальное число 65, которое состоит из 7 бит, значит, двоичное представление четвёртой компоненты содержит 7 бит. Из 34 максимальных типов получим такие двоичные наборы:

0000011000001	0010010101110	0101010100011	0111100010011	1010010000111
0000100111101	0010110101101	0101100011111	0111110010001	1010110000110
0000110111010	0011000101010	0101110011100	1000010010000	1011100000101
0001000111000	0011010101000	0110000011010	1001000001111	1011110000011
0001100110111	0011110100111	0110100011001	1001010001100	1100010000010
0001110110011	0100000100101	0110110010110	1001100001010	1101000000001
0010000110000	0100100100100	0111000010100	1010000001001	

Рассматривая эти двоичные наборы как код, находим, что его кодовое расстояние равно 2. Эти наборы являются максимальными типами, поэтому они не могут отличаться друг от друга только одной компонентой или совпадать. Это означает, что можно распознать только одну ошибку. Найдём спектр расстояний этого кода в процентах.

Кодовое расстояние	Частота
2	0,89%
3	4,81%
4	11,76%
5	18,18%
6	21,03%
7	23,35%
8	13,55%
9	5,17%
10	1,25%

Из таблицы видно, что количество пар наборов с кодовым расстоянием 2 составляют незначительную часть от всех пар. С ростом ранга этот процент уменьшается.

Для построения корректирующих кодов разработана программа, которая выводит максимальные типы определённого ранга для заданных границ, переводит их в двоичный код и вычисляет кодовое расстояние для этих кодов. В программе используются массив $T1$, в котором содержатся максимальные типы ранга $(n-1)$, и массив $T2$, в котором содержатся максимальные типы ранга n. В обоих массивах типы упорядочены по возрастанию количества двоичных единиц, а типы с одинаковым количеством единиц упорядочены по возрастанию, как двоичные числа. Количество элементов массива $T2$ с одинаковым количеством единиц представляет собой весовой спектр рассматриваемого кода. Имеются также дополнительные массивы $E1$ и $E2$, в которых содержатся индексы первых типов в массивах $T1$ и $T2$ с заданным количеством единиц. Максимальные типы обладают тем свойством, что типы с кодовым расстоянием 2 содержат одинаковое количество единиц. Таким образом, при нахождении типов с кодовым расстоянием не менее 3, необходимо просматривать только пары типов с одинаковым количеством единиц.

Рассмотрим алгоритмы кодирования и декодирования, предназначенные для исправления одной ошибки на рассматриваемом коде с кодовым расстоянием 3. Кодирование происходит простым обращением к элементу массива $T2$ с индексом, равным блоку сообщения, т.е. элементы массива $T2$ являются передаваемыми кодовыми словами. Рассмотрим алгоритм декодирования.

АЛГОРИТМ ДЕКОДИРОВАНИЯ

Шаг 1. Находим в полученном кодовом слове S количество k двоичных единиц.

Шаг 2. Выделяем из полученной последовательности максимальную правую часть $S2$ и её дополнение $S1$ (см. лемму 1). Этим уменьшается, по крайней мере, на порядок количество перебираемых типов [6]. Сдвиг-сумма $S1 \oplus S2 = S$. Находим количество единиц $k1$ в дополнении и $k2$ − в максимальной правой части.

Шаг 3. Находим в массиве $T1$ элементы с количеством единиц $k2$ с помощью массива $E1$, т.е. индекс $E1(k2)$. Поиск слова $S2$ в массиве $E1$ проводим методом деления пополам. Сначала находим индекс $i1$ среднего элемента массива $T1$ с количеством единиц $k2$. Он равен целой части от $E1(k2) + (E1(k2+1) - E1(k2))/2$. Если элемент $T1(i1)$ с индексом $i1$ равен $S2$, то устанавливаем флажок $f2$ и завершаем шаг 3. Если $T1(i1)$ больше $S2$, то копируем значение $E1(k2)$ в $i2$ и присваиваем $i1$ значение, равное целой части от $i2 + (i1 - i2)/2$. Если $T1(i1)$ меньше $S2$, то копируем значение $E1(k2+1)$ в $i2$ и присваиваем $i1$ значение, равное целой части от $i1 + (i2 - i1)/2$. Продолжаем аналогичные действия пока либо не найден элемент $T1(i1) = S2$, либо не выполнится $|i2 - i1| = 1$. В первом случае флажок $f2$ будет установлен, а во втором нет.

Шаг 4. Находим в массиве $T1$ элементы с количеством единиц $k1$ с помощью массива $E1$, т.е. индекс $E1(k1)$. Если элемент $T1(i1)$ равен $S1$, то устанавливаем флажок $f1$. Поиск слова $S1$ в массиве $E1$ проводим методом деления пополам как в шаге 3. Если элемент $T1(i1)$ с индексом $i1$ равен $S1$, то устанавливаем флажок $f1$, а если не найден, то не устанавливаем.

Шаг 5. Если флажки $f1$ и $f2$ установлены, то принятое кодовое слово представляется в виде сдвиг-суммы максимальных типов $S1$ и $S2$ ранга $(n-1)$. Согласно лемме 1 в этом случае кодовое слово S принято без ошибок. В этом случае алгоритм заканчивается.

Шаг 6. Если флажок $f1$ или $f2$ не установлен, то находим в массиве $T2$ элементы с количеством единиц $(k-1)$ и $(k+1)$ с помощью массива $E2$, т.е. индексы $E2(k-1)$ и $E2(k+1)$. Последовательно заменяем в кодовом слове S бит 0 на 1, а бит 1 на 0. После замены нуля на единицу ищем полученное слово $S3$ методом деления пополам начиная с индекса $E2(k+1)$. После замены единицы на ноль ищем полученное слово $S3$ методом деления пополам начиная с индекса $E2(k-1)$. При нахождении в массиве $T1$ кодового слова $S3$ поиск прекращаем. В этом случае при передаче была одна ошибка и в действительности было передано кодовое слово $S3$, а не $S2$. Найденный индекс $S3$ равен блоку сообщения. Если в результате поиска кодовое слово $S3$ не найдено, то в принятом кодовом слове S имеется более одной ошибки.

Проведем анализ предложенных алгоритмов и сравним их с алгоритмами кодирования и декодирования для кода Хэмминга, также позволяющего исправлять одну ошибку. Алгоритм кодирования описанного кода состоит из одной операции выборки из массива, тогда как для кода Хэмминга в алгоритме кодирования требуются, по крайней мере, $2r$ операций, где r число проверочных разрядов. Алгоритм обнаружения ошибки (шаги 1-5 алгоритма декодирования) состоит из $\log_2(m_1) + \log_2(m_2)$ операций сравнения, где m_1 и m_2 – количество типов ранга $(n-1)$ с числом единиц, равным числу единиц в дополнении и максимальной правой части соответственно. В случае, если обнаружена ошибка, для декодирования дополнительно требуется $k\log_2(m_{k-1}) + (d-k)\log_2(m_{k+1})$ операций сравнения, где d – длина кода; k – количество единиц в принятом кодовом слове; m_{k-1} и m_{k+1} – количество типов ранга n с числом единиц, равным $(k-1)$ и $(k+1)$ соответственно. Числа m_{k-1} и m_{k+1} значительно больше чисел m_1 и m_2. Так при n=10 отношение m_{k-1} к m_1 достигает 1000 и с ростом ранга оно растет. Это значит, что при малом числе ошибок возможна передача кодовых слов с большей скоростью. Для кода Хэмминга, как при обнаружении ошибки, так и при декодировании требуются те же $2r$ операций, что и при кодировании. Наконец, в коде Хэмминга проверочные разряды всегда находятся на одном месте, тогда как в описываемом коде их место не определено, что обеспечивает дополнительное шифрование передаваемой информации. На основе максимальных типов МБФ можно построить много подобных кодов, что также важно для шифрования.

Допустим, принята кодовая последовательность 1010000110000. Она соответствует типу $(0,0,0,40,48,0,0,0,0)$ ранга 8. Выделяем максимальную правую часть и её дополнение: $(0,0,0,40,48,0,0,0,0) = (0,0,40,13,0,0,0,0) \oplus (0,0,0,0,35,0,0,0)$. Определяем, что дополнение не является максимальным типом, поэтому кодовое слово передано с ошибкой. Это кодовое слово содержит 4 единицы. Максимальная правая часть и её дополнение содержат по 2 единицы. Максимальная правая часть содержится в $T1$, а дополнение нет. Ищем изменённое кодовое слово $S3$ среди элементов массива $T2$ с количеством единиц, равным 3 и 5. В результате находим, что $S3$ равно 0010000110000. Его индекс равен переданному сообщению. Принятое кодовое слово S отличается от переданного слова $S3$ первым битом, который при передаче был изменён с 0 на 1.

Если изменить компоненты типа МБФ таким образом, чтобы минимальное кодовое расстояние равнялось 5, то это даст возможность обнаруживать и исправлять уже 2 ошибки.

Например: (0,0,0,32,2,0,0,0,0); (0,0,0,29,39,0,0,0,0); (0,0,0,3,64,0,0,0,0); (0,0,0,30,59,0,0,0,0); (0,0,0,18,36,0,0,0,0); (0,0,0,17,17,0,0,0,0); (0,0,0,10,9,0,0,0,0); (0,0,0,8,84,0,0,0,0); (0,0,0,5,12,0,0,0,0); (0,0,0,4,97,0,0,0,0); (0,0,0,48,76,0,0,0,0); (0,0,0,41,36,0,0,0,0); (0,0,0,38,20,0,0,0,0); (0,0,0,25,104,0,0,0,0); (0,0,0,22,88,0,0,0,0); (0,0,0,15,48,0,0,0,0); (0,0,0,59,24,0,0,0,0); (0,0,0,58,98,0,0,0,0); (0,0,0,56,23,0,0,0,0); (0,0,0,55,33,0,0,0,0); (0,0,0,53,82,0,0,0,0); (0,0,0,45,73,0,0,0,0); (0,0,0,36,47,0,0,0,0); (0,0,0,34,91,0,0,0,0); (0,0,0,22,71,0,0,0,0); (0,0,0,14,108,0,0,0,0); (0,0,0,9,115,0,0,0,0); (0,0,0,63,68,0,0,0,0); (0,0,0,60,60,0,0,0,0); (0,0,0,51,15,0,0,0,0); (0,0,0,39,106,0,0,0,0); (0,0,0,15,95,0,0,0,0).

Пусть принята кодовая последовательность P = 1010001011011. Найдём с каким из 32 наборов таблицы эта последовательность имеет минимальное кодовое расстояние. В результате получим, что минимальное кодовое расстояние, равное двум, вектор P имеет с набором 1000101011011. Далее по

таблице определяем, что ошибка произошла в третьем и пятом битах, и было закодировано исходное слово 10111.

В рассмотренном примере для построения набора кодовых последовательностей с минимальным кодовым расстоянием 5 было рассмотрено базовое множество из 34 двоичных последовательностей. Если бы выборка производилась из всех двоичных последовательностей длиной 13, пришлось бы рассматривать 8192 двоичных последовательностей. Таким образом, за счёт использования максимальных типов базовое множество двоичных последовательностей составляет 0,39% от всех двоичных последовательностей данной длины.

Учитывая, что на больших рангах среднее кодовое расстояние получается значительно больше минимального, то для различных кодовых слов возможно исправление разного количества ошибок. В нашем случае от 2 до 3.

В заключение отметим следующее. В данной работе, на основе типов МБФ, построен код, позволяющий исправлять ошибки. При этом за счёт использования максимальных типов базовое множество двоичных последовательностей, на которых строился код, было сокращено. Разработана программа, позволяющая строить код по заданной клетке матрицы распределения типов, а также моделировать обнаружение и исправление ошибок по переданному кодовому слову. В дальнейшем планируется разработать универсальный алгоритм для изменения векторов, которые позволяют находить максимально возможное кодовое расстояние для типов МБФ, что позволяет обнаруживать и исправлять большее число ошибок. Также планируется уменьшить избыточность корректирующего кода.

Литература

1. *Byers J.* A Digital Fountain Approach to Reliable Sisttibuttion of Bulk Data / J. Byers, M. Luby, M. Mitzenmacher, A. Rege // In SIGCOMM. − 1998.
2. *David J.C.* Information Theory, Inference, and Learning Algorithms / J.C. David, MacKay. − Cambridge University Press. − 2003.
3. *Luby M.* LT Codes/ M. Luby // Of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS). − 2002. − P. 271-282.
4. *Варгаузин В.* Вблизи границы Шеннона /В.Варгаузин // ТелеМультиМедиа. − 2005. − №3. − С.3-10.
5. *Ткаченко В.Г.* Перечисление типов монотонных булевых функций при синтезе цифровых схем / В.Г. Ткаченко // Наукові праці ОНАЗ ім. О.С. Попова. – Одеса, 2008. – №2. – С. 54 – 69.
6. *Ткаченко В.Г.* Деревья типов монотонных булевых функций и криптосистемы с блоками переменной длины / В.Г. Ткаченко, О.В. Синявский // Наукові праці ОНАЗ ім. О.С. Попова. – 2009. – №2. – С. 32 – 42.
7. *Ткаченко В.Г.* Классификация монотонных булевых функций при синтезе цифровых схем / В.Г. Ткаченко // Наукові праці ОНАЗ ім. О.С. Попова. – 2008. – №1. – С. 35 – 43.