# Коды Рида-Соломона с точки зрения обывателя.

1. Поля Галуа.

Согласно «официальному» определению, полем называется множество, на котором заданы операции сложения и умножения, причём выполняются групповые аксиомы (см. теорию групп; должно быть обеспечено выполнение переместительного закона сложения и умножения, наличие нейтральных элементов относительно сложения и умножения).

Говоря обычным языком, для любых элементов должны выполняться равенства a+b = b+a и a*b = b*a. И должен существовать такой элемент e, принадлежащий нашему множеству (полю), что для всех элементов множества a выполняется a = a+e, и такой элемент u, что a = a*u. Для обычного сложения e=0, а u=1.

Однако, умножение и сложение, которые определены для полей, могут не иметь ничего общего с обычным сложением или умножением (кроме выполнения вышеупомянутых законов).

Полями Галуа называются поля, в которых присутствует конечное число элементов. Поле Галуа с количеством элементов N обозначается GF(N).

Определим сложение, как операцию «исключающее ИЛИ» (XOR). Очевидно, что в таком случае, операция сложения является обратной самой себе. Операция умножения в двоичном виде будет выглядеть так

```
  x        0011
                    0011
            ---------------
              +     00110
                      0011
            ---------------
                      0101
```

Т.е. обычное умножение «столбиком» со сложением определённым как XOR.

Такую операцию часто представляют как умножение полиномов. $(x+1)*(x+1) = x^2+1$

Можно определить также операцию деления чисел (или полиномов) с остатком - по аналогичным правилам, например:

```
1100111100000 / 100011101
100011101
-------------
 100000110
 100011101
 --------------
     11011000
```

Число 11011000 (или полином $x^7+x^6+x^4+x^3$) является остатком от деления.

Теперь рассмотрим поле Галуа, состоящее из $2^4 = 16$ элементов. Операция сложения для этого поля определена как XOR, операция же умножения дополнена получением остатка по некоторому модулю.

Выберем в качестве модуля полином $x^4+x+1$ (т..е число 10011).

Возьмём единицу и будем последовательно умножать её на 2 и рассмотрим числа, которые будут при этом получаться: (рассмотрим двоичную форму и представление в виде полинома)

$1 = 2^0 = 0001 = x^0$

$2^0*2 = 2^1 = 0010 \bmod 10011 = 2 = x^1$
$2^1*2 = 2^2 = 0100 \bmod 10111 = 4 = x^2$
$2^2*2 = 2^3 = 1000 \bmod 10011 = 8 = x^3$
Эти три строки повторяют обычное умножение, однако дальше всё не так просто

$2^4*2 = 2^4 = 10000 \bmod 10011 = 0011 = 3 = x+1$

```
 10011
--------
 00011
```

$2^4*2 = 2^5 = 100000 \bmod 10011 = 0110 = 6 = x^2+x$

```
 10011
----------
 000110
```

и т.д.
составим таблицу умножения

| Степень | Результат | |
|---|---|---|
| 0 | 1 | 0001 |
| 1 | 2 | 0010 |
| 2 | 4 | 0100 |
| 3 | 8 | 1000 |
| 4 | 3 | 0011 |
| 5 | 6 | 0110 |
| 6 | 12 | 1100 |
| 7 | 11 | 1011 |
| 8 | 5 | 0101 |
| 9 | 10 | 1010 |
| 10 | 7 | 0111 |
| 11 | 14 | 1110 |
| 12 | 15 | 1111 |
| 13 | 13 | 1101 |
| 14 | 9 | 1001 |
| 15 | 1 | 0001 |

Таким образом, $2^{15} = 1$ и, как нетрудно догадаться, при дальнейшем умножении весь цикл повторится снова. Полученные «степени двойки» несложно умножать между собой, например, $13 * 15 = 2^{13} * 2^{12} = 2^{12+13} = 2^{25 \bmod 15} = 2^{10} = 7$
Тот же результат, разумеется, можно получить умножив 13 на 15 по описанным правилам по модулю 10011

```
        1111
        1101
        -----
        1111
       1111
      1111
     ----------
     1001011 mod 10011
     10011
     -------
        0111 = 7
```

Важным фактом является то, что в этой таблице в колонке «результат» повторились все числа от 1 до 15. Если задуматься, то из этого следует, что операция умножения обратима: в самом деле, раз $a^{13}*a^{12} = a^{10}$, то $a^{10}/a^{12} = a^{-2+15} = a^{13}$
Таким образом, операция деления может быть выполнена так: находим в таблице «логарифм» - то есть «в какую степень нужно возвести 2 чтобы получить нужное число» - для делимого и делителя, после этого вычитаем логарифм делителя из логарифма делителя и прибавляем 15 (чтобы получить положительное число) и возводим 2 в эту степень.
Наша таблица обладает таким свойством не случайно; это обусловлено выбором основания 2 и модуля 10011. При выборе другого модуля и основания этого вполне могло и не получиться! Число 2 называется в данном случае «примитивным элементом поля». Формализуя, можно сказать что примитивный элемент поля Галуа GF(p) – это такое число a, что для любых t<p-1 и s<p-1 $a^t <> a^s$
Таким образом, построенное поле Галуа задаёт правила арифметики для чисел от 0 до 15 (т.е. для двоичных 4-разрядных чисел). В качестве сложения и вычитания используется XOR, умножение выполняется описанным выше способом и операция умножения всегда обратима, т.е. для любое число всегда без остатка делится на другое число (кроме 0: на 0 делить нельзя).
Все дальнейшие наши действия будут подразумевать применение такой арифметики над полями Галуа.
Аналогичным образом можно построить и арифметику для 256-битовых чисел – например, с помощью полинома $x^8+x^4+x^3+x^2+1$ (100011101)
Несмотря на то, что арифметика «странная», для неё можно выводить формулы, аналогичные формулам обычной арифметики. Что и будет делаться ниже.

## 2. Коды Рида – Соломона.

В кодах Рида-Соломона сообщение представляется в виде набора символов некоторого алфавита. Собственно говоря, в качестве алфавита используется то самое поле Галуа, рассмотренное в предыдущем разделе. То есть если мы хотим закодировать сообщение, представленное двоичным кодом, то мы разбиваем его (в случае, если мы используем наше поле Галуа из 16 элементов) на группы по 4 бита и дальше работаем с каждой группой как с числом из этого поля Галуа.
При построении кода Рида-Соломона задаётся пара чисел N, K, где N – общее количество символов, а K – «полезное» количество символов, остальные N-K символов представляют собой избыточный код, предназначенный для восстановления ошибок.
Такой код Галуа будет иметь так называемое «расстояние Хэмминга» D = N – K +1;
Расстояние Хэмминга является параметром кода и определяется как минимальное число различий между двумя различными кодовыми словами. В соответствии с теорией кодирования, код, имеющий расстояние Хемминга D = 2t+1, позволяет восстанавливать t ошибок. Таким образом, если в наше кодовое слово случайно внести t = (N-K)/2 ошибок (т.е. просто произвольно заменить значения t символов любыми значениями), то окажется возможным обнаружить и исправить эти ошибки.
Сообщения при кодировании Рида-Соломона представляются полиномами.
Исходное сообщение представляется как коэффициенты полинома p(x) степени K-1, имеющего (разумеется!) K коэффициентов.
Важную роль играет порождающий многочлен Рида-Соломона, g(x), который строится следующим образом:

$$g(x) = \prod_{(i = 1..D-1)} (x+a^i) = (x+a^1)(x+a^2)\ldots.(x+a^{D-1})$$

Здесь a – это примитивный член. Нетрудно понять (учтя, что операция сложения равносильна операции вычитания), что $a^1$, $a^2$.. $a^{D-1}$ - являются корнями этого многочлена.
Например, построим порождающий многочлен кода Рида-Соломона с N = 15, K = 9:

$g(x) = (x+2)(x+2^2)(x+2^3)(x+2^4)(x+2^5)(x+2^6) = x^6+7x^5+9x^4+3x^3+12x^2+10x+12.$

(Возведения в степень и умножения выполнены по правилам поля Галуа!)
Все дальнейшие примеры тоже будут использовать этот код; как нетрудно видеть, этот код будет способен исправлять до трёх ошибок: (15 – 9 ) / 2 = 3.

## 3. Кодирование Рида – Соломона.

Кодирование Рида-Соломона выполняется достаточно просто. Вообще говоря, существует две разновидности кодирования: систематический и несистематический код. В несистематическом коде закодированное сообщение не содержит в явном виде исходного сообщения: закодированное сообщение получается как произведение исходного сообщения (а сообщения представляются в виде многочленов!) на порождающий многочлен g(x) : C(x) = p(x) * g(x).
Систематический код строится по-другому:
Сначала полином сдвигается на K коэффициентов влево

$$p'(x) = p(x) * x^{(N-k)}$$

а потом вычисляется его остаток от деления на порождающий полином и прибавляется к p'(x):

$$C(x) = p'(x) + p'(x) \bmod g(x)$$

Другими словами, сообщение «сдвигается» на N-K символов - так, что его полином имеет такие коэффициенты:

m8, m7, m6, m5, m4, m3, m2, m1, m0, 0,0,0,0,0,0
(m0.. m8 - символы сообщения)

Далее этот полином делится с остатком на порождающий полином g(x), в результате чего в остатке получается полином степени (N-K-1) с N-k коэффициентами.
(Поскольку полином g(x) имеет степень N-k, что следует из принципа его построения, описанного в разделе 2). Этот полином прибавляется к исходному полиному, сдвинутому на N-K символов (т.е. коэффициенты остатка как раз занимают место нулей)
Для систематического кода очевидно, что K старших коэффициентов полученного кода C(x) содержат исходное сообщение. Это удобно при декодировании, поэтому в дальнейшем будем рассматривать именно систематический вариант.
Закодированное сообщение C(x) обладает очень важным свойством: оно без остатка делится на порождающий многочлен g(x)!
Для несистематического кодирования этот факт очевиден: ведь C(x) является произведением g(x) на p(x); для систематического следует рассуждать так:
Пусть r(x) – остаток от деления p'(x) на g(x).
Тогда,

$$p'(x) = g(x)*u(x) + r(x).$$

u(x) - некий полином, в данном случае абсолютно неважно, какой.
Итак,

$$r(x) = p'(x) \bmod g(x).$$

Тогда

$$C(x) = p'(x) + p'(x) \bmod g(x) = g(x)*u(x) + r(x) + r(x).$$

Вспомним, что в арифметике поля Галуа сложения является одновременно и вычитанием – тогда r(x)+r(x) = 0!
Следовательно, C(x) = g(x)* u(x), т.е. делится на g(x) без остатка.
Таким образом,

$$C(x) \bmod g(x) = 0.$$

В случае, если закодированное сообщение будет изменено, то это равенство окажется нарушенным. Факт искажения можно рассматривать как прибавление к C(x) некоторого полинома ошибки E(x).

$$C'(x) = C(x) + E(x), \text{ тогда}$$

$$C'(x) \bmod g(x) = E(x) \bmod g(x) = e(x) <> 0.$$

(Если не считать несчастного случая, когда ошибка окажется кратной g(x))

Рассмотрим кодирование информации. Пусть наше сообщение такое:

7, 5, 10, 0, 9, 1, 1, 1, 9

Полином у нас получается такой:

$p(x) = 9x^8+1x^7+1x^6+1x^5+9x^4+0x^3+10x^2+5x+7$

Умножая на $x^6$, получаем:

$9x^{14}+x^{13}+x^{12}+x^{11}+9x^{10}+10x^8+5x^7+7x^6$

Делим на g(x):

$9x^{14}+ x^{13}+ x^{12}+ x^{11}+9 x^{10}+ 0x^9+ 10x^8+5x^7+7x^6 / x^6+7x^5+9x^4+3x^3+12x^2+10x+12.$

$9x^{14}+10 x^{13}+13 x^{12}+8 x^{11}+6 x^{10}+5 x^9+ 6 x^8$

---

$11 x^{13}+12 x^{12}+ 9 x^{11}+15 x^{10}+5x^9+12 x^8+5 x^7+7x^6$

$11 x^{13}+ 4 x^{12}+12 x^{11}+14 x^{10}+13x^9+2 x^8+13x^7$

---

и тд.

(деление выполняется как и деление обычных полиномов, однако с использованием правил операций в полях Галуа) и получаем в остатке:

$13x^5+6x^4+14x^3+15x^2+15x+3$

В итоге получается полином закодированного сообщения

$C(x)= 9x^{14}+x^{13}+x^{12}+x^{11}+9x^{10}+10x^8+5x^7+7 x^6+13x^5+6x^4+14x^3+15x^2+15x+3$

Полученное закодированное сообщение

| | |
|---|---|
| 3, 15, 15, 14, 6, 13 | 7, 5, 10, 0, 9, 1, 1, 1, 9 |
| Избыточная инф. | Полезное сообщ. |

4. Декодирование кода Рида-Соломона.

Декодирование кодов Рида-Соломона значительно сложнее кодирования. Очевидно, что первым шагом необходимо выполнить деление полинома на порождающий полином g(x). Если остаток равен нулю, то сообщение неискажено и декодирование (для систематического кода) тривиально: следует просто выделить из сообщения коэффициенты с N-k+1 до N-1 – это и будет основное сообщение.

В случае же присутствия ошибки (т.е. $e(x) = C(x) \bmod g(x) <>0$), то придётся выполнить следующие действия.

Декодирование основано на построении многочлена синдрома ошибки S(x) и отыскании соответствующего ему многочлена локаторов L(x).

Локаторы ошибок – это элементы поля Галуа, степень которых совпадает с позицией ошибки. Так, если искажён коэффициент при $x^4$, то локатор этой ошибки равен $a^4$, если искажён коэффициент при $x^7$ то локатор ошибки будет равен $a^7$ и т.п. (a – примитивный член, т.е. в нашем случае a=2).

Многочлен локаторов L(x) – это многочлен, корни которого обратны локаторам ошибок. Таким образом, многочлен L(x) должен иметь вид

$$L(x) = (1+xX_1)(1+xX_2)\ldots(1+xX_i),$$

где $X_1$, $X_2$, $X_i$ – локаторы ошибок. ($1+xX_i$ обращается в ноль при $x=X_i^{-1}$: $X_iX_i^{-1} = 1$, 1+1 =0.)

Ясно, что если этот многочлен будет найден, то мы легко сможем определить локаторы ошибок – для этого потребуется только определить его корни, что легко сделать обычным перебором.

Для определения этого полинома сначала получают вспомогательный полином S(x), так называемый синдром ошибки. Коэффициенты синдрома ошибки получаются подстановкой степеней примитивного члена в остаток многочлен e(x) = C(x) mod g(x), или в сам многочлен сообщения C(x).

$S_i = e(a^{i+1})$

или

$S_i = C(a^{i+1})$

Нетрудно убедиться, что если бы сообщение не было искажено, то все коэффициенты Si оказались бы равны нулю: ведь неискажённое сообщение C(x) кратно порождающему многочлену g(x), для которого числа $a^1$, $a^2$… $a^{N-K}$ являются корнями (см. принцип составления многочлена g(x)).

Между L(x) и S(x) существует соотношение

$$L(x)*S(x) = W(x) \bmod x^{N-k}$$

W(x) называется многочленом ошибок. Степень многочлена W(x) не может превышать u-1, где u – количество ошибок. А максимальное количество ошибок, которые может исправить код Рида-Соломона, это (N-K)/2.

С учётом этого обстоятельства, а также учитывая, что свободный член L(x) $L_0$=1 (ведь $L(x) = (1+xX_1)(1+xX_2)\ldots(1+xX_i)$) можно составить систему линейных уравнений.

W(x) =

$L_0S_0$

$(L_0S_1+L_1S_0)X+$

$(L_0S_2+L_1S_1+L_2S_0)X^2+$

$(L_0S_3+L_1S_2+L_2S_1+L_3S_0)X^3+$

….

$(L_0S_{N-K-1}+L_1S_{N-K-2}\ldots. L_{(N-K)/2}S_{(N-K)/2-1})X^{N-K-1}$

…

$L_{(N-K)/2}S_{N-K-1}X^{3/2(N-K)-1}$

$L_0 = 1$

Пусть t = (N-k)/2

Коэффициенты при степенях от 0 до t – 1 не равны нулю, при старших степенях должны быть нулевыми.

$L_0S_t + L_1S_{t-1} + L_2S_{t-2} + L_3S_{t-3}.. L_tS_0 = 0$

(коэфф. при X(N-K)/2)

$L_0S_{t+1} + L_1S_t + L_2S_{t-1} + L_3S_{t-2}.. L_tS_1 = 0$

$L_0S_{t+2} + L_1S_{t+1} + L_2S_t + L_3S_{t-1}.. L_tS_2 = 0$

Коэффициент $L_0$ известен, остальные необходимо найти, следовательно требуется составить t уравнений.

$L_1S_{t-1} + L_2S_{t-2} + L_3S_{t-3}.. L_tS_0 = S_t$

$L_1S_t + L_2S_{t-1} + L_3S_{t-2}.. L_tS_1 = S_{t+1}$

$L_1S_{t+1} + L_2S_t + L_3S_{t-1}.. L_tS_2 = S_{t+2}$

…

$L_1S_{2t-2}+L_2S_{2t-2}+L_3S_{2t-3}\ldots +L_tS_t = S_{2t-1}$

В матричном виде

$||S_{t-1} \quad S_{t-2} \quad S_{t-3} \ldots S_0||$

$$M = \begin{Vmatrix} S_t & S_{t-1} & S_{t-2} \dots & S_1 \\ S_{t+1} & S_t & S_{t-1} \dots & S_2 \\ \dots\dots\dots & & & \\ S_{2t-2} & S_{2t-3} & S_{2t-4} & S_t \end{Vmatrix}$$

Элемент матрицы в строке r и столбце c

$M_{r,c} = S_{t-1+r-c}$

$$V = \begin{Vmatrix} S_t \\ S_{t+1} \\ S_{t+2} \\ \dots \\ S_{2t-1} \end{Vmatrix}$$

$$L = \begin{Vmatrix} L_1 \\ L_2 \\ L_3 \\ \dots \\ L_t \end{Vmatrix}$$

$ML = V, \quad => \quad L = M^{-1}V$

Например, для нашего примера – кода Рида-Соломона (15, 9) матрица M имеет вид:

S2 S1 S0
S3 S2 S1
S4 S3 S2

А вектор V
S3
S4
S5

Таким образом, вычисление полинома локаторов сводится к построению матрицы M, нахождению обратной ей и умножению на вектор V.
Обратная матрица получается так же, как и в обычной математике, например Жордановым методом.

Возможно, что матрица M окажется линейно-зависимой. Это означает, что ошибок меньше чем t, в этом случае следует повторить построение матрицы для t, уменьшенного на 1.

Найти полином L(x) можно и другими методами, например, можно применить алгоритм Евклида поиска НОД или применить метод Берлекампа-Месси, который является наиболее эффективным.

После того, как полином L(x) найден, следует найти его корни – они будут обратны к локаторам ошибок.
Далее вычисляется W(x) = L(x)*S(x), коэффициенты старшие чем N-k должны быть обнулены.

Далее следует вычислить производную L(x). Производная вычисляется следующим образом – для чётных степеней производная равна нулю, для нечётных - степени уменьшенной на 1.
$(x^2)' = 0$, $(x^3)' = x^2$

Далее вычисляются значения ошибок по формуле $Y_i = W(X_i^{-1})/L'(X_i^{-1})$
Таким образом, составляется полином ошибки. Его коэффициентами являются значения ошибок $Y_i$ стоящие в позициях, определяемых локаторами ошибок.

Ещё раз, коротко шаги декодирования.

1. Вычислить e(x) = C(x) mod g(x).
2. Если e(x) = 0 то выделить p(x) из C(x).
3. Иначе, вычислить полином синдрома $S_i = e(a^{i+1})$
4. Построить матрицу M и вычислить L(x)
5. Вычислить L'(x). $L'_i = L_{i+1}$ для чётных i и 0 для нечётных.
6. Вычислить W(x) = S(x)*L(x)
7. Получить корни L(x) – локаторы ошибок
8. Получить значения ошибок $Y_i = W(X_i^{-1})/L'(X_i^{-1})$
9. Сформировать многочлен ошибок E(X) на основе локаторов и значений ошибок и скорректировать C(x) = C(x) + E(x).

Пример декодирования.
«Испортим» многочлен, полученный в разделе «Кодирование»
$C(x)= 9x^{14}+x^{13}+x^{12}+x^{11}+9x^{10}+10x^8+5x^7+7\ x^6+13x^5+6x^4+14x^3+15x^2+15x+3$
Путём добавления к нему полинома ошибки $E(x) = 2x^{13}+3x^{11}+7x^8$
Получаем
$C'(x)= 9x^{14}+3x^{13}+x^{12}+2x^{11}+9x^{10}+13x^8+5x^7+7\ x^6+13x^5+6x^4+14x^3+15x^2+15x+3$
То есть, полученное сообщение имеет вид:

| | |
|---|---|
| 3, 15, 15, 14, 6, 13 | 7, 5, 13, 0, 9, 2, 1, 3, 9 |
| Избыточная инф. | Полезное сообщ. |

Делим C'(x) на g(x): получаем полином ошибки e(x)
$e(x) = 6x^5+0x^4+15x^3+3x^2+10x+13$
Полином синдрома ошибки
$S(x) = 9x^5 + 3x^4+2x^3+15x^2+15x$
Матрица M:

$$M = \begin{matrix} 15 & 15 & 0 \\ 2 & 15 & 15 \\ 3 & 2 & 15 \end{matrix}$$

обратная матрица

$$M^{-1} = \begin{matrix} 7 & 10 & 10 \\ 15 & 10 & 10 \\ 6 & 15 & 7 \end{matrix}$$

Вектор V :

$$V = \begin{matrix} 2 \\ 3 \\ 9 \end{matrix}$$

$M^{-1}V = \begin{matrix} 6 \\ 5 \\ 4 \end{matrix}$

$L(x) = 4x^3 + 5x^2 + 6x + 1$
$W(x) = L(x) * S(x) = 11x^2 + 15x$

$L'(x) = 4x^2+6$

Ищем корни многочлена L(x):
$X_1^{-1} = a^7 = 1/a^8 = 11$
$X_2^{-1} = a^4 = 1/a^{11} = 3$
$X_3^{-1} = a^2 = 1/a^{13} = 4$

Вычисляем значения соответствующих значений ошибок
Y1 = 7
Y2 = 3
Y3 = 2

Получаем многочлен ошибок:
$E(x) = 7x^8+3x^{11}+2x^{13}$

Нетрудно видеть, что полином ошибки совпадает с заданным. Таким образом, остаётся скорректировать полином C'(x) и получить исходный полином C(x):
$C(x) = C'(x) + E(x) =$
$9x^{14}+3x^{13}+x^{12}+2x^{11}+9x^{10}+13x^8+5x^7+7\ x^6+13x^5+6x^4+14x^3+15x^2+15x+3 +$
$+ \quad 2x^{13} + \quad 3x^{11}+ \quad 7x^8$
$= 9x^{14}+1x^{13}+x^{12}+1x^{11}+9x^{10}+10x^8+5x^7+7\ x^6+13x^5+6x^4+14x^3+15x^2+15x+3 +$

Опять же нетрудно убедиться, что исправленный полином соответствует заданному.

Варьируя N и K, можно выбирать между надёжностью и избыточностью кода. Обычно используются коды по полю $2^8$, можно использовать полином $x^8+x^4+x^3+x^2+1$ (100011101)
Например, код (255, 223) способен успешно исправлять 16 ошибок, а (255, 239) – 8 ошибок.

RAID – массивы.
Для создания RAID – массивов используется подобная технология, однако задачу облегчает то, что что известно, какое именно устройство отказало т.е. известно место ошибки. Простейший RAID массив строится на следующем принципе.
Задаём общее количество устройств количество N и количество «полезных» устройств K, соответственно, N-K устройств хранят избыточную корректирующую информацию. Информация пишется на устройства следующим образом: берётся K блоков, на их основании вычисляется N-k корректирующих блоков и полученные N блоков пишутся на N устройств.
Для вычисления корректирующих значений выбирается матрица F из N-K строк и K столбцов, эта матрица используется для вычисления корректирующих значений:

$F = \begin{matrix} F_{11} & F_{12} & F_{13} & F_{14} \\ F_{21} & F_{22} & F_{23} & F_{24} \end{matrix}$

$F_{31}$ $F_{32}$ $F_{33}$ $F_{34}$
(для N = 7, K = 3)

Корректирующие значения вычисляются как произведение информационных блоков на эту матрицу :
C = Fd, где d – вектор записываемых блоков данных

$$D = \begin{matrix} d1 \\ d2 \\ d3 \\ d4 \end{matrix}$$

Таким образом, при умножении получается вектор контрольных сумм C

$$C = \begin{matrix} c1 \\ c2 \\ c3 \end{matrix}$$

(Разумеется, блоки разбиваются, например, на слова или байты и каждое слово контрольной суммы вычисляется отдельно!)
Предположим, происходит отказ u устройств, u<=k.
В этом случае восстановление всех информационных слов возможно следующим образом:

Строим объединённую матрицу из матрицы F и единичной матрицы:

$$F^{\wedge} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ F_{11} & F_{12} & F_{13} & F_{14} \\ F_{21} & F_{22} & F_{23} & F_{24} \\ F_{31} & F_{32} & F_{33} & F_{34} \end{matrix}$$

Умножая эту матрицу на вектор d, получаем объединённый вектор cd

$$cd = \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ c_1 \\ c_2 \\ c_3 \end{matrix}$$

$$\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ F_{11} & F_{12} & F_{13} & F_{14} \end{matrix} * \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{matrix} = \begin{matrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ c_1 \end{matrix}$$

| | | | | | | |
|---|---|---|---|---|---|---|
| $F_{21}$ | $F_{22}$ | $F_{23}$ | $F_{24}$ | | | $c_2$ |
| $F_{31}$ | $F_{32}$ | $F_{33}$ | $F_{34}$ | | | $c_3$ |

Из этой таблицы мы можем вычеркнуть N-K строк, при этом матрица станет квадратной

| | | | | | | |
|---|---|---|---|---|---|---|
| ~~1~~ | ~~0~~ | ~~0~~ | ~~0~~ | | | ~~d1~~ |
| ~~0~~ | ~~1~~ | ~~0~~ | ~~0~~ | | $d_1$ | ~~$d_2$~~ |
| 0 | 0 | 1 | 0 | | $d_2$ | $d_3$ |
| ~~0~~ | ~~0~~ | ~~0~~ | ~~1~~ | * | $d_3$ | = ~~$d_4$~~ |
| $F_{11}$ | $F_{12}$ | $F_{13}$ | $F_{14}$ | | $d_4$ | $c_1$ |
| $F_{21}$ | $F_{22}$ | $F_{23}$ | $F_{24}$ | | | $c_2$ |
| $F_{31}$ | $F_{32}$ | $F_{33}$ | $F_{34}$ | | | $c_3$ |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | | $d_1$ | $d_3$ |
| $F_{11}$ | $F_{12}$ | $F_{13}$ | $F_{14}$ | * | $d_2$ | = $c_1$ |
| $F_{21}$ | $F_{22}$ | $F_{23}$ | $F_{24}$ | | $d_3$ | $c_2$ |
| $F_{31}$ | $F_{32}$ | $F_{33}$ | $F_{34}$ | | $d_4$ | $c_3$ |

| | | | | |
|---|---|---|---|---|
| | 0 | 0 | 1 | 0 |
| | $F_{11}$ | $F_{12}$ | $F_{13}$ | $F_{14}$ |
| R= | $F_{21}$ | $F_{22}$ | $F_{23}$ | $F_{24}$ |
| | $F_{31}$ | $F_{32}$ | $F_{33}$ | $F_{34}$ |

Находя обратную матрицу, $R^{-1}$ получаем уравнение, которое позволяет восстановить вектор d :

$d = R^{-1} * c^{\wedge}$

| | |
|---|---|
| | $d_3$ |
| $c^{\wedge}$ = | $c_1$ |
| | $c_2$ |
| | $c_3$ |

Ранг матрицы F^ должен быть равен K.

Лит.
1. В.М. Охорзин, Д.С. Кукунин, М.С. Новодворский
Построение каскадных кодов на основе кодов Боуза – Чоудхури – Хоквингема
и Рида – Соломона.
Санкт-Петербургский государственный университет телекоммуникаций
им. проф. М.А. Бонч-Бруевича
http://dvo.sut.ru/libr/opds/i287ohor/index.htm
2. Крис Касперски
Могущество кодов Рида-Соломона или информация, воскресшая из пепла

Журнал «Системный администратор»
http://www.insidepro.com/kk/027/027r.shtml
3. Трифонов Петр Владимирович – Адаптированное кодирование в многочастотных системах.
dcn.infos.ru/~petert/papers/PhD.pdf
4. A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems - James S. Plank, Technical Report UT-CS-96-332, University of Tennessee, July, 1996.
http://www.cs.utk.edu/~plank/plank/papers/CS-96-332.html

5. «Элементарное руководство по по CRC – алгоритмам обнаружения ошибок» Ross N. Williams , RockSoft