Ссылка на статью:
// Ученые записки УлГУ. Сер. Математика и информационные технологии. УлГУ. Электрон. журн. 2021, № 1, с. 87–101.
Поступила: 18.04.2021
Окончательный вариант: 18.04.2021

УДК 519.725

Об алгоритмах декодирования циклических кодов

Рацеев С.М.*, ***Иванцов А.М.***, ***Булдаковский П.А.***

*ratseevsm@mail.ru
УлГУ, Ульяновск, Россия

Циклические коды довольно широко применяются при помехоустойчивом кодировании. Такие коды используются для обнаружения и/или исправления ошибок. Среди циклических кодов особо выделяются коды Рида-Соломона. В работе рассматриваются некоторые простые алгоритмы декодирования циклических кодов и кодов Рида-Соломона.

Ключевые слова: циклический код, декодирование кода, коды Рида-Соломона.

Введение

Пусть $F = GF(q)$ — конечное поле, где q — некоторая степень простого числа. Циклическим кодом называется линейное подпространство $A \subset F^n$, выдерживающее циклический сдвиг компонент своих векторов. Для некоторых циклических кодов существуют сравнительно простые алгебраические методы кодирования и декодирования, поэтому такие коды часто используются для передачи информации в каналах связи с помехами. О сферах применения циклических (и не только) кодов можно посмотреть в работе [1].

Пусть $(x^n - 1)$ — идеал кольца многочленов $F[x]$, порожденный многочленом $x^n - 1$, $F[x]/(x^n - 1)$ — факторкольцо. Пространство $F[x]/(x^n - 1)$ распадается на q^n непересекающихся смежных классов, которое можно задать в виде линейной оболочки:

$$F[x]/(x^n - 1) = L(\overline{1}, \overline{x}, \overline{x}^2, \ldots, \overline{x}^{n-1}),$$

где $\overline{x}^i$ — смежный класс, порожденный одночленом x^i. Вместо рассуждений в терминах смежных классов, будем вести рассмотрение в терминах многочленов минимальных степеней в классах, так как имеется взаимно однозначное соответствие между многочленами степеней, меньших n, и смежными классами.

Хорошо известно, что в факторкольце $F[x]/(x^n-1)$ линейный код A является циклическим тогда и только тогда, когда A — идеал кольца $F[x]/(x^n-1)$. Идеал A кольца $F[x]/(x^n-1)$ содержит единственный унитарный многочлен $g(x)$ минимальной степени, причем все многочлены идеала A кратны многочлену $g(x)$. Многочлен $g(x)$ называется порождающим многочленом идеала. Любой многочлен $g(x)$, порождающий идеал в кольце $F[x]/(x^n-1)$, делит многочлен x^n-1. Пусть $\deg\, g(x) = r$. Тогда многочлены

$$g(x),\ xg(x), \ldots, x^{n-r-1}g(x)$$

образуют базис пространства A, порожденного многочленом $g(x)$. Поэтому циклический код A имеет следующий вид:

$$A = \{a(x)g(x) \mid \deg a(x) < n-r\}.$$

Многочлен $h(x) = (x^n-1)/g(x)$ называется проверочным многочленом кода A.

Пусть A — циклический $[n,k]$-код с проверочным многочленом $h(x) = h_0 + h_1x + \ldots + h_kx^k$. Тогда двойственный код $A^\perp$ является циклическим $[n, n-k]$-кодом с порождающим многочленом:

$$g^\perp(x) = h_0^{-1}x^k h\left(\frac{1}{x}\right) = h_0^{-1}(h_k + h_{k-1}x + \ldots + h_0x^k).$$

Самыми простыми и распространенными кодами являются коды с проверкой на четность, которые получаются следующим образом. Пусть $q = 2$. К каждому информационному вектору длины k дописывается $(k+1)$-й бит так, чтобы вес вектора был четным. Этот дополнительный бит называется битом проверки на четность. Пусть $n = k+1$. Полученный код является $[n, n-1, 2]$-линейным кодом. Он используется для обнаружения любого нечетного числа ошибок. При этом исправлять ошибки он не умеет. Данный код является циклическим кодом, при этом $g(x) = 1+x$, $h(x) = \frac{x^n-1}{1+x} = 1 + x + x^2 + \ldots + x^{n-1}$. Коды CRC (Cyclic Redundancy Code — циклические избыточные коды) являются обобщениями кодов с проверкой на четность и также применяются для обнаружения ошибок. Некоторые коды CRC прописаны в международных стандартах. В таблице ниже приводятся несколько примеров таких кодов.

CRC	$g(x)$	**Стандарт**	**Применение**
CRC-4	$x^4 + x + 1$	ITU G.704	Датчик качества передачи данных через поток
CRC-5	$x^5 + x^2 + 1$	ITU G.704	Пакеты токенов USB
CRC-7	$x^7 + x^3 + 1$	ITU-T G.707, ITU-T G.832	Системы телекоммуникации, MMC, SD
CRC-8-CCITT	$x^8 + x^2 + x + 1$	ITU-T I.432.1 (02/99)	Технологии ATM, ISDN
CRC-11	$x^{11} + x^9 + x^8 + x^7 + x^2 + 1$		Сетевые протоколы FlexRay
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$		6-разрядные коды в системах телекоммуникации
CRC-16	$x^{16} + x^{15} + x^2 + 1$	ANSI X3.28	Протокол Modbus, USB
CRC-16-CCITT	$x^{16} + x^{12} + x^5 + 1$		8-разрядные коды в Bluetooth, SD-карты
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	IEEE 802.3	Ethernet, архиваторы

На данный момент не существует ни одного эффективного алгоритма декодирования произвольного циклического кода [2]. В следующих двух параграфах приводится один из самых простых и эффективных алгоритмов декодирования некоторого подкласса циклических кодов, исправляющего ошибки, и модификация этого алгоритма, исправляющего пакеты ошибок. При этом данным алгоритмом пакеты ошибок исправляются для любого циклического кода, а разрозненные ошибки, к сожалению, только для некоторого подкласса циклических кодов.

Важным частным случаем циклических кодов являются коды Рида-Соломона, которые могут исправлять разрозненные ошибки и пакеты ошибок, поэтому широко используются в телекоммуникационных системах (например, при передаче данных по сетям WiMAX, в оптических линиях связи, в спутниковой и радиорелейной связи, в некоторых прикладных программах в области хранения данных, в частности, в RAID 6). Пусть $\alpha = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1})$, где α_i – различные элементы поля $GF(q)$, $y = (y_0, y_1, \ldots, y_{n-1})$ — ненулевые (не обязательно различные) элементы из $GF(q)$. Тогда обобщенный $[n, k, d = n - k + 1]$-код Рида–Соломона состоит из всех кодовых векторов вида

$$u = (y_0 b(\alpha_0),\ y_1 b(\alpha_1), \ldots,\ y_{n-1} b(\alpha_{n-1})),$$

где $b(x)$ — информационные многочлены над полем $GF(q)$ степени не выше $k - 1$. Если $n = q - 1$, вектор y состоит из единиц и $\alpha_i = \alpha^i$, $i = 0, 1, \ldots, n - 1$, где α — примитивный элемент поля $GF(q)$, то в этом случае получаем код Рида–Соломона (РС).

Для кодов Рида–Соломона хорошо известны следующие алгоритмы декодирования [3, 4, 5, 6]: алгоритм Гао (асимптотическая сложность которого оценивается величиной $O(n(\log n)^2)$ и совпадает со сложностью лучших алгоритмов декодирования кодов РС), алгоритм Питерсона–Горенстейна–Цирлера (сложность алгоритма $O(n^3)$ [7]), алгоритм Берлекэмпа–Месси (сложность алгоритма $O(n^2)$ [7]), алгоритм Сугиямы–Касахары–Хирасавы–Намекавы (сложность алгоритма $O(n^2)$ [7]). Для обобщенных кодов Рида-Соломона и кодов Гоппы подобные алгоритмы рассматривались в работах [8, 9, 10, 11]. Актуальность алгоритмов декодирования обобщенных кодов РС состоит также том, что их можно применять для декодирования кодов Гоппы, при этом именно на основе кодов Гоппы строятся некоторые перспективные постквантовые криптосистемы [12].

Указанные алгоритмы декодирования кодов Рида–Соломона позволяют найти единственное кодовое слово u, находящееся на расстоянии не более чем $t = [(d - 1)/2]$ от полученного вектора v, так как любой шар радиуса t с центром в произвольной точке пространства $V = F^n$ содержит не более одного кодового слова. Но на практике иногда требуется исправление большего числа ошибок. В этом случае декодирование может быть неоднозначным, т.е. декодер может возвратить список некоторых кодовых слов. Для этой цели используется метод Гурусвами-Судана списочного декодирования кодов Рида-Соломона (см., напр. [13, 14, 15, 16]).

1 Декодер с вылавливанием ошибок

Определим синдромный многочлен $s(x)$ как остаток от деления принятого на приемном конце многочлена $v(x)$ на $g(x)$. Понятно, что $\deg s(x) < r$. Так как $v(x) = u(x) + e(x) =$

$a(x)g(x)+e(x)$, то многочлен $s(x)$ зависит только от многочлена ошибок $e(x)$. Проблема декодирования циклического кода сводится к поиску многочлена $e(x)$ по известному синдромному многочлену $s(x)$.

Хорошо известно, что в случае циклических кодов при выполнении условия $d \geq 2t+1$ каждому многочлену ошибок $e(x)$ веса не более t соответствует единственный синдромный многочлен. Таким образом, задача декодирования сводится к однозначному вычислению многочлена ошибок $e(x)$ минимального веса, который принадлежит одному и тому же смежному классу по модулю $g(x)$, что и синдромный многочлен $s(x)$.

Пусть $v(x) = u(x)+e(x)$ — полученный искаженный многочлен. Разделим многочлен $v(x)$ с остатком на $g(x)$:

$$v(x) = q(x)g(x) + s(x), \quad \deg s(x) < \deg g(x).$$

Запись $xv(x) \pmod{x^n-1}$ означает циклический сдвиг вектора v длины n на одну позицию вправо. Меггитом замечено, что нет необходимости вычислять синдромы для всех циклических сдвигов принятого слова. Новый синдром можно найти по уже вычисленному:

$$\big(xv(x) \pmod{x^n-1}\big) \equiv xs(x) \pmod{g(x)},$$

где $v(x) \equiv s(x) \pmod{g(x)}$. На основе данного сравнения можно построить так называемый декодер Меггита (см., напр., [17]), который требует меньше памяти для хранения синдромных многочленов, но из-за этого снижается производительность алгоритма по сравнению с алгоритмом декодирования, который использует все синдромные многочлены. Поэтому рассмотрим более интересный и производительный алгоритм декодирования циклических кодов, который отлично подходит для исправления пакетов ошибок. Данный алгоритм строится на основе следующего утверждения.

Для многочлена $f(x)$ через $wt(f(x))$ будем обозначать число ненулевых коэффициентов.

Теорема 1 ([17])**.** Пусть A — циклический $[n,k,d]$-код над полем $F = GF(q)$, $d \geq 2t+1$, с порождающим многочленом $g(x)$. Пусть $e(x)$ — многочлен ошибок веса не более t, $s(x)$ — остаток от деления $e(x)$ на $g(x)$. Обозначим через $s_i(x)$ остаток от деления $x^i s(x)$ на $g(x)$, $i = 0,1,\ldots,n$. Если для некоторого j, $0 \leq j \leq n$, вес многочлена $s_j(x)$ не превосходит t, то выполнено сравнение:

$$e(x) \equiv x^{n-j} s_j(x) \pmod{x^n-1}.$$

Учитывая теорему 1, можно построить следующий эффективный алгоритм декодирования циклического кода, который называется *декодером с вылавливанием ошибок.*

Алгоритм 1 (**декодер с вылавливанием ошибок**)**.**

1. Принятый многочлен $v(x)$ делим с остатком $s(x)$ на $g(x)$.

2. Полагаем $s_0(x) = s(x)$. Если $s_0(x) = 0$, по полагается, что ошибок не происходило. Алгоритм в этом случае возвращает $v(x)$.

3. Если $wt(s_0(x)) \leq t$, то $e(x) := s_0(x)$. Алгоритм возвращает $v(x) - e(x)$.

3. Цикл $i := 1$ до $n-1$:

   3.1. Вычисляем $s_i(x)$ как остаток от деления $xs_{i-1}(x)$ на $g(x)$.

   3.2. Если $wt(s_i(x)) \leq t$, то $e(x) := x^{n-i} s_i(x) \pmod{x^n-1}$ и алгоритм возвращает $v(x) - e(x)$.

Заметим, что данный алгоритм предлагает очень простую реализацию. Однако, он не применим для любого циклического кода, так как не для всех циклических кодов существует многочлен $s_j(x)$ веса не более t из теоремы 1.

Если на многочлены ошибок накладывать некоторые ограничения (например, все ошибки расположены «близко» друг от друга), то для таких случаев рассмотренный алгоритм декодирования вполне себе применим.

Предложение 1. Пусть A — циклический $[n, k, d]$-код, $d \geq 2t+1$, с порождающим многочленом $g(x)$. Пусть $e(x)$ — многочлен ошибок веса не более t, для которого найдется такое j, $0 \leq j \leq n-1$, для которого многочлен $x^j e(x) \pmod{x^n-1}$ имеет степень не более $n-k-1$. Тогда вес вектора $s_j(x)$ не превосходит t, причем выполнено сравнение $e(x) \equiv x^{n-j} s_j(x) \pmod{x^n-1}$.

Доказательство. Учитывая теорему 1, достаточно показать существование многочлена $s_j(x)$, вес которого не превосходит t. Учитывая неравенство Синглтона, из $n-k+1 \geq d \geq 2t+1$ получаем $\deg g(x) = n - k \geq 2t$.

Пусть $x^j e(x) \pmod{x^n - 1}$ имеет степень не более $n - k - 1$ и вес многочлена $e(x)$ не более t. Понятно, что

$$\big(x^j e(x) \pmod{x^n - 1}\big) \pmod{g(x)} = x^j e(x) \pmod{x^n - 1}.$$

Поэтому

$$\big(x^j e(x) \pmod{x^n - 1}\big) \equiv x^j s(x) \pmod{g(x)}.$$

Следовательно, вес многочлена $s_j(x) \equiv x^j s(x) \pmod{g(x)}$ не превосходит t. □

Предложение 2. Пусть в условиях теоремы 1 выполнено $t < n/k$. Тогда найдется такое целое j, $0 \leq j \leq n$, для которого многочлен $s_j(x)$ имеет вес не более t.

Доказательство. Сведем доказательство к предложению 1. Разделим n с остатком на k: $n = qk + r$, $0 \leq r < k$. Разобьем вектор e из n элементов на q подмножеств подряд идущих элементов $E_1, E_2, \ldots, E_q$, где в каждом подмножестве ровно k элементов, а в E_q $k+r$ элементов. Возможны два случая.

1. $r = 0$, поэтому $t < q$. Это значит, что найдется хотя бы одно множество E_i, в котором одни нули. Следовательно, для некоторого циклического сдвига элементов вектора e множество E_q будет состоять из нулей, поэтому для некоторого j степень многочлена $x^j e(x) \pmod{x^n - 1}$ строго меньше $n - k$.

2. $r > 0$, поэтому $t \leq q$. Если вес вектора e меньше, чем t, то все сводится к предыдущему случаю. Поэтому пусть $wt(e) = t$. Пусть $e_{i_1}, \ldots, e_{i_t}$ — все ненулевые компоненты вектора e. Рассмотрим пары соседних элементов последовательности $e_{i_1}, \ldots, e_{i_t}$. Под расстоянием между e_i и e_j, $i < j$, будем понимать число $j - i$. Если расстояние между элементами одной из пар превышает число k, то, очевидно, что для некоторого j степень многочлена $x^j e(x) \pmod{x^n-1}$ строго меньше $n - k$. Предположим, что расстояние между соседними элементами каждой из пар не превосходит k. Тогда расстояние от e_{i_1} до e_{i_t} не превосходит $(t - 1)k$. Поэтому общее число элементов вектора e до элемента e_{i_1} и после элемента e_{i_t} не меньше чем k. Следовательно, для некоторого j степень многочлена $x^j e(x) \pmod{x^n - 1}$ строго меньше $n - k$. □

Пример 1. Рассмотрим двоичный циклический код с параметрами $n = 15$, $k = 7$, $g(x) = 1 + x^4 + x^6 + x^7 + x^8$. Пусть поле $GF(2^4)$ определяется примитивным многочленом $1 + x + x^4$ с примитивным элементом α. Так как, в частности, элементы $\alpha, \alpha^2, \alpha^3, \alpha^4$ поля $GF(2^4)$ являются корнями многочлена $g(x)$, то для кодового расстояния этого кода (который можно рассматривать как код БЧХ) выполнено неравенство $d \geq 5$, поэтому код может исправлять одну и две ошибки.

Выясним, применим ли алгоритм 1 для исправления любой одной или любых двух ошибок. Из предложения 1 следует, что алгоритм 1 исправляет все одиночные ошибки. Пусть $e(x) = x^i + x^j$ — двойная ошибка, $0 \leq i < j \leq 14$. Возможны два случая. 1) $j - i < 8 = \deg g(x)$. Тогда степень многочлена $x^{15-i}e(x) \equiv 1 + x^{j-i} \pmod{x^{15} - 1}$ меньше 8, поэтому по предложению 1 такая ошибка исправляется алгоритмом 1. 2) $j - i \geq 8$. Тогда $15 - (j - i) \leq 7$. Поэтому $x^{15-j}e(x) \equiv 1 + x^{15-(j-i)} \pmod{x^{15} - 1}$. Таким образом, алгоритм 1 для данного кода исправляет все одиночные и двойные ошибки.

Предположим, что на приемном конце получен вектор

$$v = \left(\begin{array}{cccccccccccccc} 0, & 1, & 0, & 1, & 0, & 1, & 1, & 0, & 0, & 1, & 1, & 1, & 0, & 0, & 0 \end{array} \right),$$

который соответствует многочлену:

$$v(x) = x + x^3 + x^5 + x^6 + x^9 + x^{10} + x^{11},$$

содержащему не более двух ошибок. Тогда:

$$\begin{aligned} v(x) \equiv s(x) &= s_0(x) = x + x^5 + x^6 + x^7, \\ xs(x) &\equiv s_1(x) = 1 + x^2 + x^4, \\ x^2s(x) &\equiv s_2(x) = x + x^3 + x^5, \\ x^3s(x) &\equiv s_3(x) = x^2 + x^4 + x^6, \\ x^4s(x) &\equiv s_4(x) = x^3 + x^5 + x^7, \\ x^5s(x) &\equiv s_5(x) = 1 + x^7, \end{aligned}$$

где все сравнения проводились по модулю $g(x)$. Так как вес многочлена $s_5(x)$ не превышает двух, то $j = 5$. Поэтому:

$$e(x) \equiv x^{15-5}(1 + x^7) \equiv x^2 + x^{10} \pmod{x^{15} - 1},$$

$$u(x) = v(x) - e(x) = x + x^2 + x^3 + x^5 + x^6 + x^9 + x^{11}.$$

Это значит, что исходный кодовый вектор равен:

$$u = \left(\begin{array}{cccccccccccccc} 0, & 1, & 1, & 1, & 0, & 1, & 1, & 0, & 0, & 1, & 0, & 1, & 0, & 0, & 0 \end{array} \right).$$

2 Циклические коды, исправляющие пакеты ошибок

Так как некоторые каналы связи чувствительны к пакетам из t ошибок [4], которые группируются в пределах короткого временного интервала, а не произвольно разбросанных t ошибок в сообщении, то для этих целей можно строить особые (более эффективные для данного случая) коды, обладающие большей скоростью.

Циклическим пакетом длины t называется вектор, все ненулевые компоненты которого расположены среди t последовательных (по циклу) компонент, первая и последняя из которых отличны от нуля.

Пусть $b(x)$ — некоторый многочлен степени $t-1$, $b_0 \neq 0$. Тогда многочлен $e(x) = x^i b(x) \pmod{x^n-1}$ представляет собой циклический пакет ошибок длины t, при этом x^i указывает начальную позицию пакета в $e(x)$, $b(x)$ — собственно пакет ошибок.

Учитывая сказанное в предыдущем параграфе, синдромные многочлены $s(x)$ циклического кода, исправляющего пакеты ошибок длины t, должны быть различными, т.е. при $e_1(x) \neq e_2(x)$ из

$$e_i(x) = q_i(x)g(x) + s_i(x), \quad \deg s_i(x) < \deg g(x), \quad i = 1, 2,$$

следует, что $s_1(x) \neq s_2(x)$, где $e_1(x)$ и $e_2(x)$ задают циклические пакеты длины t.

Теорема 2 ([17])**.** Пусть A — циклический $[n,k]$-код с порождающим многочленом $g(x)$ степени $r = n-k$. Тогда ни один из векторов кода A не является пакетом длины $n-k$ или менее. Поэтому код A может обнаружить любой циклический пакет ошибок длины $n-k$ и менее.

Предложение 3. Пусть A — циклический $[n,k,d]$-код с порождающим многочленом $g(x)$, способный исправлять все циклические пакеты ошибок длины не более t. Пусть $e(x)$ — циклический пакет ошибок длины не более t. Тогда в обозначениях теоремы 1 найдется такое j, $0 \leq j \leq n$, для которого $\deg s_j(x) < t$, причем выполнено сравнение $e(x) \equiv x^{n-j} s_j(x) \pmod{x^n-1}$.

Доказательство. Пусть $e(x) \equiv x^i b(x) \pmod{x^n-1}$ — некоторый циклический пакет ошибок длины не более t, $\deg b(x) < t$. Тогда $x^{n-i}e(x) \equiv b(x) \pmod{x^n-1}$. Тогда

$$\big(b(x) \equiv x^{n-i}e(x) \pmod{x^n-1}\big) \equiv x^{n-i}s(x) \pmod{g(x)}.$$

Так как $\deg b(x) < t$ и по неравенству Рейгера $2t \leq n-k$, то $\deg b(x) < n-k = \deg g(x)$. Поэтому:

$$b(x) \pmod{g(x)} = b(x).$$

Получаем $b(x) \equiv s_{n-i}(x) \pmod{x^n-1}$, где $s_{n-i}(x) \equiv x^{n-i}s(x) \pmod{g(x)}$. Таким образом, при $j = n-i$ выполнено сравнение $e(x) \equiv x^{n-j}s_j(x) \pmod{x^n-1}$. □

Из данного предложения следует, что если циклический код может исправлять пакеты ошибок длины t и менее, то можно легко привести модификацию алгоритма 1, причем модифицированный алгоритм будет всегда исправлять эти пакеты ошибок.

Алгоритм 2 (**декодер с вылавливанием пакетов ошибок**)**.**

1. Принятый многочлен $v(x)$ делим с остатком $s(x)$ на $g(x)$.

2. Полагаем $s_0(x) = s(x)$. Если $s_0(x) = 0$, по полагается, что ошибок не происходило. В этом случае алгоритм возвращает $v(x)$.

3. Если $\deg s_0(x) < t$, то $e(x) := s_0(x)$. Алгоритм возвращает $v(x) - e(x)$.

3. Цикл $i := 1$ до $n-1$:

   3.1. Вычисляем $s_i(x)$ как остаток от деления $xs_{i-1}(x)$ на $g(x)$.

   3.2. Если $\deg s_i(x) < t$, то $e(x) := x^{n-i}s_i(x) \pmod{x^n-1}$ и алгоритм возвращает $v(x) - e(x)$.

Пример 2. Рассмотрим $[7, 3]$-циклический код с порождающим многочленом $\widetilde{g}(x) = x^4 + x^3 + x^2 + 1$. Данный код исправляет все пакеты ошибок длины 2 и длины 1 [4]. Построим на его основе методом перемежения $[14, 6]$-циклический код с порождающим многочленом $g(x) = 1 + x^4 + x^6 + x^8$, который исправляет все пакеты ошибок длины 4 и менее.

Пусть на приемном конце получен вектор

$$v = (0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0),$$

в котором имеется циклический пакет ошибок длины не более 4. С помощью алгоритма 2 найдем этот пакет ошибок. Данному вектору соответствует многочлен

$$v(x) = x + x^3 + x^4 + x^5 + x^7 + x^8 + x^9 + x^{11} + x^{12}.$$

Вычисляем:

$$\begin{aligned}
&v(x) \equiv s(x) = s_0(x) = 1 + x + x^2 + x^4 + x^5,\\
&xs(x) \equiv s_1(x) = x + x^2 + x^3 + x^5 + x^6,\\
&x^2s(x) \equiv s_2(x) = x^2 + x^3 + x^4 + x^6 + x^7,\\
&x^3s(x) \equiv s_3(x) = 1 + x^3 + x^5 + x^6 + x^7,\\
&x^4s(x) \equiv s_4(x) = 1 + x + x^7,\\
&x^5s(x) \equiv s_5(x) = 1 + x + x^2 + x^4 + x^6,\\
&x^6s(x) \equiv s_6(x) = x + x^2 + x^3 + x^5 + x^7,\\
&x^7s(x) \equiv s_7(x) = 1 + x^2 + x^3,
\end{aligned}$$

где сравнения проводятся по модулю многочлена $g(x)$. Так как $\deg s_7(x) = 3$, то $j = 7$. Следовательно:

$$e(x) \equiv x^{14-7}(1 + x^2 + x^3) \equiv x^7 + x^9 + x^{10} \pmod{x^{14} - 1},$$

$$u(x) = v(x) - e(x) = x + x^3 + x^4 + x^5 + x^8 + x^{10} + x^{11} + x^{12}.$$

Если требуется построить циклический код, исправляющий пакеты ошибок длины t и менее, то для этого можно использовать коды Файра. *Кодом Файра* называется циклический код над полем $GF(q)$ с порождающим многочленом

$$g(x) = (x^{2t-1} - 1)p(x),$$

где $p(x)$ — неприводимый многочлен степени m над полем $GF(q)$, $m \geq t$, порядок корней которого равен e, причем $2t - 1$ не делится на e. Длина кода Файра равна $n = \text{НОК}(e,\, 2t - 1)$. Если $p(x)$ — примитивный многочлен, то $n = \text{НОК}(q^m - 1,\, 2t - 1)$. Хорошо известно, что код Файра исправляет все пакеты ошибок длины t и менее. Для обнаружения наличия пакета ошибок достаточно получить ненулевой синдромный многочлен $s(x)$, а для исправления пакета ошибок можно воспользоваться алгоритмом 2. Например, при $q = 2$, $m = t = 5$ код Файра, заданный порождающим многочленом $g(x) = (x^9 - 1)p(x)$, где $p(x)$ — примитивный многочлен степени 5, является $[279, 265]$-кодом, который исправляет все пакеты ошибок длины 5 и менее.

3 Алгоритм декодирования Гао для ОРС-кодов

Заметим, что в некоторых (простейших) случаях для кодов Рида-Соломона можно применять декодер с вылавливанием ошибок 1.

Пример 3. Построим поле $GF(2^3)$ на основе примитивного многочлена x^3+x+1 с примитивным элементом α. Так как $\alpha^3=\alpha+1$, то:

$$\begin{array}{rrrrrl}
\alpha^0 = & 1 & & & = & 100,\\
\alpha^1 = & & \alpha & & = & 010,\\
\alpha^2 = & & & \alpha^2 & = & 001,\\
\alpha^3 = & 1 & +\alpha & & = & 110,\\
\alpha^4 = & & \alpha & +\alpha^2 & = & 011,\\
\alpha^5 = & 1 & +\alpha & +\alpha^2 & = & 111,\\
\alpha^6 = & 1 & & +\alpha^2 & = & 101,\\
\alpha^7 = & 1 & & & = & 100.
\end{array}$$

Рассмотрим $[7,3]$-код РС над $GF(2^3)$. В этом случае $n=7$, $k=3$, $n-k=4$, $d=5$. Поэтому код исправляет до двух ошибок. Порождающий многочлен кода имеет следующий вид:

$$g(x)=(x-\alpha)(x-\alpha^2)(x-\alpha^3)(x-\alpha^4)=$$
$$=\alpha^3+\alpha x+x^2+\alpha^3x^3+x^4.$$

Так как в этом случае $t<n/k$, то по предложению 2 для данного кода применим алгоритм декодирования 1.

Пусть на приемном конце принят многочлен:

$$v(x)=\alpha^5+\alpha^5x+\alpha^6x^2+x^5+\alpha^4x^6,$$

в котором не более двух ошибок (из поля $GF(2^3)$). Производим следующие вычисления:

$$\begin{gathered}
v(x)=g(x)\cdot(\alpha^4+\alpha^4x^2)+\alpha^4+\alpha x^2+\alpha^4x^3,\\
s(x)=\alpha^4+\alpha x^2+\alpha^4x^3,\\
xs(x)=g(x)\cdot\alpha^4+1+x+\alpha^4x^2+\alpha^3x^3,\\
s_1(x)=1+x+\alpha^4x^2+\alpha^3x^3,\\
x^2s(x)\equiv xs_1(x)=g(x)\cdot\alpha^3+\alpha^6+\alpha^5x+\alpha x^2+\alpha^3x^3,\\
s_2(x)=\alpha^6+\alpha^5x+\alpha x^2+\alpha^3x^3,\\
x^3s(x)\equiv xs_2(x)=g(x)\cdot\alpha^3+\alpha^6+\alpha^3x+\alpha^2x^2+\alpha^5x^3,\\
s_3(x)=\alpha^6+\alpha^3x+\alpha^2x^2+\alpha^5x^3,\\
x^4s(x)\equiv xs_3(x)=g(x)\cdot\alpha^5+\alpha+\alpha^2x^2+\alpha^4x^3,\\
s_4(x)=\alpha+\alpha^2x^2+\alpha^4x^3,\\
x^5s(x)\equiv xs_4(x)=g(x)\cdot\alpha^4+1+\alpha^6x+\alpha^4x^2+\alpha^6x^3,\\
s_5(x)=1+\alpha^6x+\alpha^4x^2+\alpha^6x^3,\\
x^6s(x)\equiv xs_5(x)=g(x)\cdot\alpha^6+\alpha^2+\alpha x^3,\\
s_6(x)=\alpha^2+\alpha x^3,
\end{gathered}$$

где все сравнения происходят по модулю $g(x)$. Так как вес многочлена $s_6(x)$ равен двум (не превосходит t), то:

$$e(x)\equiv x^{7-6}(\alpha^2+\alpha x^3)\equiv\alpha^2x+\alpha x^4\ (\mathrm{mod}\ x^7-1),$$
$$u(x)=v(x)-e(x)=\alpha^5+\alpha^3x+\alpha^6x^2+\alpha x^4+x^5+\alpha^4x^6.$$

В данном параграфе приведем (для сравнения) алгоритм декодирования Гао [3, 9] для кодов РС и ОРС-кодов. Этот алгоритм является наиболее простым и очень эффективным, но по простоте описания и реализации он заметно уступает алгоритму декодирования с вылавливанием ошибок.

Алгоритм 3 (**алгоритм декодирования Гао для кодов РС и ОРС-кодов**).
Вход: принятый вектор v.
Выход: исходный информационный вектор b, если произошло не более $[(d-1)/2]$ ошибок.
1. Интерполяция. Строится интерполяционный многочлен $f(x)$, для которого:

$$f(\alpha_i) = y_i^{-1} v_i, \quad i = 0, 1, \ldots, n-1.$$

2. Незаконченный обобщенный алгоритм Евклида. Определим многочлен

$$m(x) = (x - \alpha_0)(x - \alpha_1) \ldots (x - \alpha_{n-1}).$$

Пусть $r_{-1}(x) = m(x)$, $r_0(x) = f(x)$, $v_{-1}(x) = 0$, $v_0(x) = 1$. Производится последовательность действий обобщенного алгоритма Евклида:

$$r_{i-2}(x) = r_{i-1}(x) q_{i-1}(x) + r_i(x),$$

$$v_i(x) = v_{i-2}(x) - v_{i-1}(x) q_{i-1}(x), \quad i \geq 1,$$

до тех пор, пока не достигается такого $r_j(x)$, для которого:

$$\deg r_{j-1}(x) \geq \frac{n+k}{2}, \quad \deg r_j(x) < \frac{n+k}{2}.$$

3. Деление. Информационный многочлен равен $b(x) = \dfrac{r_j(x)}{v_j(x)}$.

В приведенном алгоритме, в частности, встречается задача нахождения интерполяционного многочлена, которая тесно связана с задачей обращения матрицы Вандермонда. Помимо классического метода Гаусса существуют алгоритмы обращения матрицы Вандермонда, которые учитывают ее структуру. В работе [18] приводятся формулы для вычисления обратной матрицы к матрице Вандермонда специального вида над конечным полем. В работе [19] приводится алгоритм со сложностью $O(n^3)$. В работах [20, 21, 22, 23, 24] приводятся алгоритмы со сложностью $O(n^2)$.

Если алгоритм 3 применять для кодов РС, то $m(x) = x^n - 1$, где $n = q - 1$. В этом случае кодирование информационных векторов происходит на основе дискретного преобразования Фурье, что является крайне эффективным преобразованием. Но зато ОРС-коды можно строить для различных значений n — длин кодовых слов. При этом скорость алгоритмов декодирования может несколько падать по сравнению с аналогичными алгоритмами для кодов РС.

Пример 4. Пусть требуется передать 10 байт информации с возможностью исправления до трех байт (скажем, на каждые 16 байт). Пусть в качестве информационного сообщения

выступает слово «correction». Каждый символ этого сообщения закодируем с помощью (расширенного) ASCII кода, т.е. каждый символ будет занимать один байт. В закодированном виде (с помощью кодировки ASCII) наше сообщение имеет вид:

$$b = (99, 111, 114, 114, 101, 99, 116, 105, 111, 110).$$

Понятно, что в двоичном канале связи каждое такое число передается в виде двоичного вектора длины 8:

$$\begin{aligned} b = \quad & (01100011, 01101111, 01110010, 01110010, 01100101, \\ & 01100011, 01110100, 01101001, 01101111, 01101110). \end{aligned}$$

Для кодирования вектора b применим обобщенный код РС над полем $GF(2^8)$ с параметрами $k = 10$, $n = 16$, $d = 7$, который может исправлять до трех ошибок. Для построения самого поля $GF(2^8)$ используем примитивный многочлен $p(x) = 1 + x^2 + x^3 + x^4 + x^8$ с примитивным элементом α:

$$\begin{array}{cccc} \alpha^0 = 10000000, & \alpha^1 = 01000000, & \alpha^2 = 00100000, & \alpha^3 = 00010000, \\ \alpha^4 = 00001000, & \alpha^5 = 00000100, & \alpha^6 = 00000010, & \alpha^7 = 00000001, \\ \dots & \dots & \dots & \dots \\ \alpha^{247} = 11000001, & \alpha^{248} = 11011000, & \alpha^{249} = 01101100, & \alpha^{250} = 00110110, \\ \alpha^{251} = 00011011, & \alpha^{252} = 10110101, & \alpha^{253} = 11100010, & \alpha^{254} = 01110001. \end{array}$$

Будем кодировать элементы поля слева направо по возрастанию степени элемента α, поэтому, например, вектор (01100011) соответствует элементу $\alpha+\alpha^2+\alpha^6+\alpha^7 = \alpha^{163} \in GF(2^8)$. Так как α является примитивным элементом поля $GF(2^8)$, то информационный вектор b над полем $GF(2^8)$ примет вид:

$$b = (\alpha^{163}, \alpha^{61}, \alpha^{155}, \alpha^{155}, \alpha^{72}, \alpha^{163}, \alpha^{10}, \alpha^{58}, \alpha^{61}, \alpha^{186}).$$

Данный вектор соответствует информационному многочлену:

$$\begin{aligned} b(x) = \quad & \alpha^{163} + \alpha^{61}x + \alpha^{155}x^2 + \alpha^{155}x^3 + \alpha^{72}x^4 + \\ & + \alpha^{163}x^5 + \alpha^{10}x^6 + \alpha^{58}x^7 + \alpha^{61}x^8 + \alpha^{186}x^9. \end{aligned}$$

Для кода $\mathrm{OPC}_{10}(\gamma, y)$ определим:

$$\gamma = (1, \alpha, \alpha^2, \dots, \alpha^{15}), \quad y = (1, 1, \dots, 1),$$

где вектор y состоит из одних единиц. Матрица Вандермонда $V = V(\gamma)$ и ее обратная имеют

следующий вид:

$$V = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & \alpha & \alpha^2 & \alpha^3 & \alpha^4 & \alpha^5 & \alpha^6 & \alpha^7 & \alpha^8 & \alpha^9 & \alpha^{10} & \alpha^{11} & \alpha^{12} & \alpha^{13} & \alpha^{14} & \alpha^{15} \\
1 & \alpha^2 & \alpha^4 & \alpha^6 & \alpha^8 & \alpha^{10} & \alpha^{12} & \alpha^{14} & \alpha^{16} & \alpha^{18} & \alpha^{20} & \alpha^{22} & \alpha^{24} & \alpha^{26} & \alpha^{28} & \alpha^{30} \\
1 & \alpha^3 & \alpha^6 & \alpha^9 & \alpha^{12} & \alpha^{15} & \alpha^{18} & \alpha^{21} & \alpha^{24} & \alpha^{27} & \alpha^{30} & \alpha^{33} & \alpha^{36} & \alpha^{39} & \alpha^{42} & \alpha^{45} \\
1 & \alpha^4 & \alpha^8 & \alpha^{12} & \alpha^{16} & \alpha^{20} & \alpha^{24} & \alpha^{28} & \alpha^{32} & \alpha^{36} & \alpha^{40} & \alpha^{44} & \alpha^{48} & \alpha^{52} & \alpha^{56} & \alpha^{60} \\
1 & \alpha^5 & \alpha^{10} & \alpha^{15} & \alpha^{20} & \alpha^{25} & \alpha^{30} & \alpha^{35} & \alpha^{40} & \alpha^{45} & \alpha^{50} & \alpha^{55} & \alpha^{60} & \alpha^{65} & \alpha^{70} & \alpha^{75} \\
1 & \alpha^6 & \alpha^{12} & \alpha^{18} & \alpha^{24} & \alpha^{30} & \alpha^{36} & \alpha^{42} & \alpha^{48} & \alpha^{54} & \alpha^{60} & \alpha^{66} & \alpha^{72} & \alpha^{78} & \alpha^{84} & \alpha^{90} \\
1 & \alpha^7 & \alpha^{14} & \alpha^{21} & \alpha^{28} & \alpha^{35} & \alpha^{42} & \alpha^{49} & \alpha^{56} & \alpha^{63} & \alpha^{70} & \alpha^{77} & \alpha^{84} & \alpha^{91} & \alpha^{98} & \alpha^{105} \\
1 & \alpha^8 & \alpha^{16} & \alpha^{24} & \alpha^{32} & \alpha^{40} & \alpha^{48} & \alpha^{56} & \alpha^{64} & \alpha^{72} & \alpha^{80} & \alpha^{88} & \alpha^{96} & \alpha^{104} & \alpha^{112} & \alpha^{120} \\
1 & \alpha^9 & \alpha^{18} & \alpha^{27} & \alpha^{36} & \alpha^{45} & \alpha^{54} & \alpha^{63} & \alpha^{72} & \alpha^{81} & \alpha^{90} & \alpha^{99} & \alpha^{108} & \alpha^{117} & \alpha^{126} & \alpha^{135} \\
1 & \alpha^{10} & \alpha^{20} & \alpha^{30} & \alpha^{40} & \alpha^{50} & \alpha^{60} & \alpha^{70} & \alpha^{80} & \alpha^{90} & \alpha^{100} & \alpha^{110} & \alpha^{120} & \alpha^{130} & \alpha^{140} & \alpha^{150} \\
1 & \alpha^{11} & \alpha^{22} & \alpha^{33} & \alpha^{44} & \alpha^{55} & \alpha^{66} & \alpha^{77} & \alpha^{88} & \alpha^{99} & \alpha^{110} & \alpha^{121} & \alpha^{132} & \alpha^{143} & \alpha^{154} & \alpha^{165} \\
1 & \alpha^{12} & \alpha^{24} & \alpha^{36} & \alpha^{48} & \alpha^{60} & \alpha^{72} & \alpha^{84} & \alpha^{96} & \alpha^{108} & \alpha^{120} & \alpha^{132} & \alpha^{144} & \alpha^{156} & \alpha^{168} & \alpha^{180} \\
1 & \alpha^{13} & \alpha^{26} & \alpha^{39} & \alpha^{52} & \alpha^{65} & \alpha^{78} & \alpha^{91} & \alpha^{104} & \alpha^{117} & \alpha^{130} & \alpha^{143} & \alpha^{156} & \alpha^{169} & \alpha^{182} & \alpha^{195} \\
1 & \alpha^{14} & \alpha^{28} & \alpha^{42} & \alpha^{56} & \alpha^{70} & \alpha^{84} & \alpha^{98} & \alpha^{112} & \alpha^{126} & \alpha^{140} & \alpha^{154} & \alpha^{168} & \alpha^{182} & \alpha^{196} & \alpha^{210} \\
1 & \alpha^{15} & \alpha^{30} & \alpha^{45} & \alpha^{60} & \alpha^{75} & \alpha^{90} & \alpha^{105} & \alpha^{120} & \alpha^{135} & \alpha^{150} & \alpha^{165} & \alpha^{180} & \alpha^{195} & \alpha^{210} & \alpha^{225}
\end{pmatrix},$$

$$V^{-1} = \begin{pmatrix}
\alpha^{252} & \alpha^{245} & \alpha^{150} & \alpha^{13} & \alpha^{28} & \alpha^{124} & \alpha^{199} & \alpha^{198} & \alpha^{190} & \alpha^{175} & \alpha^{84} & \alpha^{227} & \alpha^{196} & \alpha^{62} & \alpha^{141} & \alpha^{132} \\
\alpha^{245} & \alpha^{28} & \alpha^{206} & \alpha^{171} & \alpha^{95} & \alpha^{245} & \alpha^{184} & \alpha^{198} & \alpha^{3} & \alpha^{77} & \alpha^{62} & \alpha^{175} & \alpha^{51} & \alpha^{213} & \alpha^{25} & \alpha^{126} \\
\alpha^{150} & \alpha^{206} & \alpha^{246} & \alpha^{247} & \alpha^{81} & \alpha^{218} & \alpha^{99} & \alpha^{96} & \alpha^{219} & \alpha^{97} & \alpha^{91} & \alpha^{210} & \alpha^{14} & \alpha^{206} & \alpha^{198} & \alpha^{32} \\
\alpha^{13} & \alpha^{171} & \alpha^{247} & \alpha^{231} & \alpha^{176} & \alpha^{35} & \alpha^{173} & \alpha^{45} & \alpha^{113} & \alpha^{21} & \alpha^{243} & \alpha^{6} & \alpha^{27} & \alpha^{254} & \alpha^{21} & \alpha^{151} \\
\alpha^{28} & \alpha^{95} & \alpha^{81} & \alpha^{176} & \alpha^{253} & \alpha^{180} & \alpha^{72} & \alpha^{129} & \alpha^{27} & \alpha^{90} & \alpha^{137} & \alpha^{193} & \alpha^{246} & \alpha^{180} & \alpha^{130} & \alpha^{167} \\
\alpha^{124} & \alpha^{245} & \alpha^{218} & \alpha^{35} & \alpha^{180} & \alpha^{150} & \alpha^{147} & \alpha^{196} & \alpha^{45} & \alpha^{124} & \alpha & \alpha^{122} & \alpha^{213} & \alpha^{46} & \alpha^{2} & \alpha^{9} \\
\alpha^{199} & \alpha^{184} & \alpha^{99} & \alpha^{173} & \alpha^{72} & \alpha^{147} & \alpha^{125} & \alpha^{102} & \alpha^{51} & \alpha^{171} & \alpha^{109} & \alpha^{60} & \alpha^{231} & \alpha^{37} & \alpha^{2} & \alpha^{85} \\
\alpha^{198} & \alpha^{198} & \alpha^{96} & \alpha^{45} & \alpha^{129} & \alpha^{196} & \alpha^{102} & \alpha^{198} & \alpha^{207} & \alpha^{36} & \alpha^{15} & \alpha^{237} & \alpha^{53} & \alpha^{144} & \alpha^{168} & \alpha^{85} \\
\alpha^{190} & \alpha^{3} & \alpha^{219} & \alpha^{113} & \alpha^{27} & \alpha^{45} & \alpha^{51} & \alpha^{207} & \alpha^{183} & \alpha^{72} & \alpha^{151} & \alpha^{69} & \alpha^{225} & \alpha^{6} & \alpha^{93} & \alpha^{78} \\
\alpha^{175} & \alpha^{77} & \alpha^{97} & \alpha^{21} & \alpha^{90} & \alpha^{124} & \alpha^{171} & \alpha^{36} & \alpha^{72} & \alpha^{80} & \alpha^{87} & \alpha^{252} & \alpha^{83} & \alpha^{249} & \alpha^{64} & \alpha^{64} \\
\alpha^{84} & \alpha^{62} & \alpha^{91} & \alpha^{243} & \alpha^{137} & \alpha & \alpha^{109} & \alpha^{15} & \alpha^{151} & \alpha^{87} & \alpha^{75} & \alpha^{90} & \alpha^{185} & \alpha^{98} & \alpha^{110} & \alpha^{229} \\
\alpha^{227} & \alpha^{175} & \alpha^{210} & \alpha^{6} & \alpha^{193} & \alpha^{122} & \alpha^{60} & \alpha^{237} & \alpha^{69} & \alpha^{252} & \alpha^{90} & \alpha^{148} & \alpha^{56} & \alpha^{201} & \alpha^{200} & \alpha^{118} \\
\alpha^{196} & \alpha^{51} & \alpha^{14} & \alpha^{27} & \alpha^{246} & \alpha^{213} & \alpha^{231} & \alpha^{53} & \alpha^{225} & \alpha^{83} & \alpha^{185} & \alpha^{56} & \alpha^{96} & \alpha^{97} & \alpha^{6} & \alpha^{88} \\
\alpha^{62} & \alpha^{213} & \alpha^{206} & \alpha^{254} & \alpha^{180} & \alpha^{46} & \alpha^{37} & \alpha^{144} & \alpha^{6} & \alpha^{249} & \alpha^{98} & \alpha^{201} & \alpha^{97} & \alpha^{81} & \alpha^{26} & \alpha^{210} \\
\alpha^{141} & \alpha^{25} & \alpha^{198} & \alpha^{21} & \alpha^{130} & \alpha^{2} & \alpha^{2} & \alpha^{168} & \alpha^{93} & \alpha^{64} & \alpha^{110} & \alpha^{200} & \alpha^{6} & \alpha^{26} & \alpha^{88} & \alpha^{35} \\
\alpha^{132} & \alpha^{126} & \alpha^{32} & \alpha^{151} & \alpha^{167} & \alpha^{9} & \alpha^{85} & \alpha^{85} & \alpha^{78} & \alpha^{64} & \alpha^{229} & \alpha^{118} & \alpha^{88} & \alpha^{210} & \alpha^{35} & \alpha^{27}
\end{pmatrix}.$$

Для данного кода многочлен $m(x)$ имеет вид:

$$\begin{aligned}
m(x) = \; & (x-1)(x-\alpha)\dots(x-\alpha^{15}) = \\
= \; & \alpha^{120} + \alpha^{225}x + \alpha^{194}x^2 + \alpha^{182}x^3 + \alpha^{169}x^4 + \alpha^{147}x^5 + \\
& + \alpha^{191}x^6 + \alpha^{91}x^7 + \alpha^{3}x^8 + \alpha^{76}x^9 + \alpha^{161}x^{10} + \alpha^{102}x^{11} + \\
& + \alpha^{109}x^{12} + \alpha^{107}x^{13} + \alpha^{104}x^{14} + \alpha^{120}x^{15} + x^{16}.
\end{aligned}$$

После кодирования вектора b получаем кодовый вектор:

$$u = (b(1), b(\alpha), \dots, b(\alpha^{15})) = (b_0, b_1, \dots, b_9, 0, \dots, 0)V =$$
$$= (\alpha^{239}, \alpha^{140}, \alpha^{50}, \alpha^{33}, \alpha^{154}, \alpha^{86}, \alpha^{173}, \alpha^{145}, \alpha^{39}, \alpha^{63}, \alpha^{90}, \alpha^{251}, \alpha^{103}, \alpha^{111}, \alpha^{222}, \alpha^{118}),$$

где вектор $(b_0, b_1, \dots, b_9, 0, \dots, 0)$ получается из вектора b путем добавления нулей до длины 16. По каналу связи сообщение будет передаваться в двоичном виде:

$$\begin{aligned}
u = \; & (01101000, 00100001, 10100000, 11100100, \\
& 10011100, 10001101, 01101111, 10110010, \\
& 10101100, 10000101, 11111011, 00011011, \\
& 00010001, 01110011, 01010001, 11100011).
\end{aligned}$$

Пусть на приемном конце получен вектор

$$v = (\alpha^{239}, \alpha^{140}, \underline{\alpha}^{167}, \alpha^{33}, \underline{\alpha}^{199}, \alpha^{86}, \alpha^{173}, \underline{\alpha}^{140}, \alpha^{39}, \alpha^{63}, \alpha^{90}, \alpha^{251}, \alpha^{103}, \alpha^{111}, \alpha^{222}, \alpha^{118}),$$

т.е. ошибки произошли на позициях 2, 4 и 7 (нумеруя с нуля), но на приемном конце этот факт неизвестен.

Применим алгоритм декодирования 3.

1. Интерполяция. Вычисляем коэффициенты многочлена $f(x) = f_0 + f_1 x + \ldots + f_{15} x^{15}$:

$$(f_0, f_1, \ldots, f_{15}) = vV^{-1} =$$

$$= (\alpha^{239}, \alpha^{122}, \alpha^{40}, \alpha^{193}, \alpha^{97}, \alpha^{234}, \alpha^{75}, \alpha^{212}, \alpha^{252}, \alpha^{248}, \alpha^{179}, \alpha^{147}, \alpha^{119}, \alpha^{187}, \alpha^{140}, \alpha^{209}),$$

$$f(x) = \alpha^{239} + \alpha^{122}x + \alpha^{40}x^2 + \alpha^{193}x^3 + \alpha^{97}x^4 + \alpha^{234}x^5 + \alpha^{75}x^6 + \alpha^{212}x^7 +$$
$$+\alpha^{252}x^8 + \alpha^{248}x^9 + \alpha^{179}x^{10} + \alpha^{147}x^{11} + \alpha^{119}x^{12} + \alpha^{187}x^{13} + \alpha^{140}x^{14} + \alpha^{209}x^{15}.$$

2. Незаконченный обобщенный алгоритм Евклида. Полагаем $r_{-1}(x) = m(x)$, $r_0(x) = f(x)$, $v_{-1}(x) = 0$, $v_0(x) = 1$. Применяем обобщенный алгоритма Евклида:

$$\begin{aligned}
r_{-1}(x) &= r_0(x)q_0(x) + r_1(x),\\
q_0(x) &= \alpha^{196} + \alpha^{46}x,\\
r_1(x) &= \alpha^{252} + \alpha^{169}x + \alpha^{50}x^2 + \alpha^{223}x^3 + \alpha^{110}x^4 + \alpha^{192}x^5 + \alpha^{199}x^6 + \alpha^{253}x^7 + \alpha^{193}x^8 +\\
&\quad +\alpha^{114}x^9 + \alpha^{90}x^{10} + \alpha^{137}x^{11} + \alpha^{8}x^{12} + \alpha^{115}x^{13} + \alpha^{193}x^{14},\\
v_1(x) &= v_{-1}(x) - v_0(x)q_0(x) = \alpha^{196} + \alpha^{46}x,\\
r_0(x) &= r_1(x)q_1(x) + r_2(x),\\
q_1(x) &= \alpha^{58} + \alpha^{16}x,\\
r_2(x) &= \alpha^{93} + \alpha^{234}x + \alpha^{197}x^2 + \alpha^{167}x^3 + \alpha^{221}x^4 + \alpha^{174}x^5 + \alpha^{19}x^6 + \alpha^{236}x^7 + \alpha^{126}x^8 +\\
&\quad +\alpha^{67}x^9 + \alpha^{185}x^{10} + \alpha^{212}x^{11} + \alpha^{30}x^{12} + \alpha^{188}x^{13},\\
v_2(x) &= v_0(x) - v_1(x)q_1(x) = \alpha^{24}x + \alpha^{10}x + \alpha^{62}x^2,\\
r_1(x) &= r_2(x)q_2(x) + r_3(x),\\
q_2(x) &= \alpha^{15} + \alpha^{5}x,\\
r_3(x) &= \alpha^{243} + \alpha^{227}x + \alpha^{40}x^2 + \alpha^{248}x^3 + \alpha^{170}x^4 + \alpha^{32}x^5 + \alpha^{238}x^6 + \alpha^{4}x^7 + \alpha^{143}x^8 +\\
&\quad +\alpha^{86}x^9 + \alpha^{207}x^{10} + \alpha^{55}x^{11} + \alpha^{253}x^{12},\\
v_3(x) &= v_1(x) - v_2(x)q_2(x) = \alpha^{80} + \alpha^{220}x + \alpha^{105}x^2 + \alpha^{67}x^3,
\end{aligned}$$

который останавливается на третьем шаге, так как $(n+k)/2 = 13$, $\deg r_2(x) = 13$, $\deg r_3(x) = 12$.

3. Деление. Исходный информационный многочлен равен:

$$b(x) = \frac{r_3(x)}{v_3(x)} =$$

$$= \alpha^{163} + \alpha^{61}x + \alpha^{155}x^2 + \alpha^{155}x^3 + \alpha^{72}x^4 + \alpha^{163}x^5 + \alpha^{10}x^6 + \alpha^{58}x^7 + \alpha^{61}x^8 + \alpha^{186}x^9.$$

Сопоставив обратно каждому элементу вектора b символ, получаем исходное слово «correction».

Список литературы

[1] Костюков А.С., Башкиров А.В., Никитин Л.Н., Бобылкин И.С., Макаров О.Ю. Помехоустойчивое кодирование в современных форматах связи // Вестник Воронежского государственного технического университета. 2019. Т. 15, № 2. С. 132–138.

[2] Tinnirello C. Cyclic Codes: Low-Weight Codewords and Locators. PhD thesis, University of Trento, 2016.

[3] Gao S. A new algorithm for decoding Reed–Solomon codes // Communications, Information and Network Security – Norwell, MA: Kluwer, 2003. Vol. 712. P. 55–68.

[4] Блейхут Р. Теория и практика кодов, контролирующих ошибки / Пер. с англ. М.: Мир, 1986. 576 с.

[5] Huffman W. C., Pless V. Fundamentals of Error-Correcting Codes. – Cambridge University Press, 2003. 646 p.

[6] Elia M., Viterbo E., Bertinetti G. Decoding of binary separable Goppa codes using Berlekamp–Massey algorithm // Electronics Letters, 1999. Vol. 35. № 20. P. 1720–1721.

[7] Yongge Wang. Decoding Generalized Reed–Solomon Codes and Its Application to RLCE Encryption Scheme // ArXiv abs/1702.07737 (2017): n. pag.

[8] Рацеев С.М. Об алгоритмах декодирования кодов Гоппы // Челяб. физ.-матем. журн. 2020. Т. 5, № 3. С. 327–341.

[9] Рацеев С.М., Череватенко О.И. О простом алгоритме декодировании кодов БЧХ, кодов Рида-Соломона и кодов Гоппы // Вестник СибГУТИ. 2020. № 3. С. 3–14.

[10] Рацеев С.М., Череватенко О.И. Об алгоритмах декодирования обобщенных кодов Рида-Соломона // Системы и средства информатики. 2020. Т. 30, № 4. С. 83–94.

[11] Рацеев С.М., Череватенко О.И. Об алгоритмах декодирования обобщенных кодов Рида-Соломона на случай ошибок и стираний // Вестник Самарского университета. Естественнонаучная серия. 2020. Т. 26, № 3. С. 17–29.

[12] Status Report on the First Round of the NIST Post-Quantum Cryptography Standardization Process: Internal Report 8240. – National Institute of Standards and Technology, January, 2019. 27 p. https://doi.org/10.6028/NIST.IR.8240

[13] Трифонов П.В. Методы построения и декодирования многочленных кодов. дис. . . . доктора технических наук: 05.13.17: защищена 01.10.2018. С.-Пб. 2018.

[14] Guruswami V., Sudan M. Improved decoding of Reed-Solomon and algebraic-geometric codes // IEEE Transactions on Information Theory. 1999. September. Vol. 45, no. 6. P. 1757–1767.

[15] McEliece R.J. The Guruswami - Sudan Decoding Algorithm for Reed - Solomon Codes. IPN Progress Report 42–153. May 15, 2003.

[16] Madhu Sudan. Algorithmic Introduction to Coding Theory: Lecture Notes. 2001. Available from http://theory.csail.mit.edu/ madhu/FT01/

[17] Питерсон У., Уэлдон Э. Коды, исправляющие ошибки. М.: Мир, 1976. – 596 с.

[18] Althaus H., Leake R. Inverse of a finite-field Vandermonde matrix (Corresp.) // IEEE Transactions on Information Theory. 1969. Vol. 15, № 1. P. 173.

[19] Клейбанов С.Б., Норкин К.Б., Привальский В.Б. Обращение матрицы Вандермонда // Автоматика и телемеханика. 1977. № 4. С. 176–177.

[20] Björck Å, Pereyra V. Solution of Vandermonde systems of equations // Mathematics of Computation. 1970. Vol. 24, № 112. P. 893–903.

[21] Gohberg I., Olshevsky V. The fast generalized Parker-Traub algorithm for inversion of Vandermonde and related matrices // Journal of Complexity. 1997. Vol. 13, № 2. P. 208–234.

[22] Parker F.D. Inverses of Vandermonde matrices // The American Mathematical Monthly. 1964. Vol. 71, № 4. P. 410–411.

[23] Traub J.F. Associated polynomials and uniform methods for the solution of linear problems // Siam Review. 1966. Vol. 8, № 3. P. 277–301.

[24] Yan S., Yang A. Explicit algorithm to the inverse of Vandermonde matrix // 2009 International Conference on Test and Measurement. Hong Kong, 2009. P. 176–179.