Reverse Berlekamp-Massey Decoding

Jiun-Hung Yu and Hans-Andrea Loeliger
Department of Information Technology and Electrical Engineering
ETH Zurich, Switzerland
Email: {yu, loeliger}@isi.ee.ethz.ch

***Abstract*—We propose a new algorithm for decoding Reed-Solomon codes (up to half the minimum distance) and for computing inverses in $F[x]/m(x)$. The proposed algorithm is similar in spirit and structure to the Berlekamp-Massey algorithm, but it works naturally for general $m(x)$.**

I. Introduction

In this paper, we propose a new algorithm that solves the following problem.

Partial-Inverse Problem: Let $b(x)$ and $m(x)$ be nonzero polynomials over some finite field F, with $\deg b(x) < \deg m(x)$. Find a nonzero polynomial $\Lambda(x) \in F[x]$ of the smallest degree such that

$$\deg\Big(b(x)\Lambda(x) \bmod m(x)\Big) < d \tag{1}$$

for fixed $d \in \mathbb{Z}$, $1 \leq d \leq \deg m(x)$. □

In the special case where $d = 1$ and $\gcd\big(b(x), m(x)\big) = 1$, the problem reduces to computing the inverse of $b(x)$ in $F[x]/m(x)$.

Another special case of the Partial-Inverse Problem is the standard key equation for decoding Reed-Solomon codes [1]–[5] (see Section II-B). In this case, we have $m(x) = x^{n-k}$, where n and k are the blocklength and the dimension of the code, respectively. The proposed algorithm then essentially coincides with the Berlekamp-Massey algorithm [2], [6] except that it processes the polynomial $b(x)$ (the syndrome) in the reverse order.

The Partial-Inverse Problem with general $m(x)$ arises from an alternative key equation that will be discussed in Section IV. This alternative key equation and the corresponding decoding algorithm generalize naturally to polynomial remainder codes [7], [9]–[11], which have not been amenable to Berlekamp-Massey decoding.

The Partial-Inverse Problem can also be solved by a version of the Euclidean algorithm in the style of [3], [5], [7], [8]. In fact, it has long been known that the Berlekamp-Massey algorithm and the Euclidean algorithm are related [12]–[14], and explicit translations were given in [12], [14]. In this respect, the algorithm proposed in this paper allows such a translation that is particularly transparent. However, this topic is not elaborated in the present paper due to lack of space.

The paper is structured as follows. Section II comprises a number of remarks on the Partial-Inverse Problem, including its application to the standard key equation. The new algorithm is proposed in Section III and proved in Sections V and VI. Decoding Reed-Solomon codes via the alternative key equation is described in Section IV, and the generalization of this approach to polynomial remainder codes is outlined in the appendix.

The following notation will be used. The Hamming weight of $e \in F^n$ will be denoted by $\mathrm{w_H}(e)$. The coefficient of x^ℓ of a polynomial $b(x) \in F[x]$ will be denoted b_ℓ. The leading coefficient (i.e., the coefficient of $x^{\deg b(x)}$) of a nonzero polynomial $b(x)$ will be denoted by $\operatorname{lcf} b(x)$, and we also define $\operatorname{lcf}(0) \stackrel{\triangle}{=} 0$. We will use "mod" both as in $r(x) = b(x) \bmod m(x)$ (the remainder of a division) and as in $b(x) \equiv r(x) \bmod m(x)$ (a congruence modulo $m(x)$). For $x \in \mathbb{R}$, $\lceil x \rceil$ is the smallest integer not smaller than x.

II. Remarks

A. General Remarks

We begin with a number of remarks on the Partial-Inverse Problem as stated in Section I.

1) The stated assumptions imply $\deg m(x) \geq 1$.
2) For $d = \deg m(x)$, the problem is solved by $\Lambda(x) = 1$. Smaller values of d will normally require a polynomial $\Lambda(x)$ of higher degree.
3) In the special case where $d = 1$, we have the following solutions. If $\gcd\big(b(x), m(x)\big) = 1$, then $b(x)$ has an inverse in $F[x]/m(x)$ and $\Lambda(x)$ is that inverse (up to a scale factor); otherwise, the solution is $\Lambda(x) = m(x)/\gcd\big(b(x), m(x)\big)$, which yields $b(x)\Lambda(x) \bmod m(x) = 0$.
4) The previous remark implies that the problem has a solution for any $d \geq 1$.
5) We will see that the solution $\Lambda(x)$ of the problem is unique up to a scale factor (Proposition 2 in Section V) and satisfies
$$\deg \Lambda(x) \leq \deg m(x) - d \tag{2}$$
(by (42) in Section VI).
6) In consequence of (2), coefficients b_ℓ of $b(x)$ with
$$\ell < 2d - \deg m(x) \tag{3}$$
and coefficients m_ℓ of $m(x)$ with
$$\ell < 2d - \deg m(x) + 1 \tag{4}$$
are irrelevant for the solution $\Lambda(x)$: these coefficients do not affect (1) since
$$b(x)\Lambda(x) \bmod m(x) = b(x)\Lambda(x) - q(x)m(x) \tag{5}$$

with $\deg q(x) < \deg \Lambda(x) \leq \deg m(x) - d$.
Such irrelevant coefficients may be set to zero without affecting the solution $\Lambda(x)$.

B. Application to the Standard Key Equation

The standard key equation for decoding Reed-Solomon codes [2]–[5], [13] is

$$S(x)\Lambda(x) \equiv \Gamma(x) \bmod x^{n-k}, \tag{6}$$

where n and k are the blocklength and the dimension of the code, respectively, and where $S(x)$ is a (given) syndrome polynomial with $\deg S(x) < n-k$. The desired solution is a pair $\Gamma(x)$ and $\Lambda(x) \neq 0$ such that $\deg \Gamma(x) < \deg \Lambda(x) \leq (n-k)/2$.

The problem of finding such a pair $\Gamma(x)$ and $\Lambda(x)$ translates into a Partial-Inverse Problem with $b(x) = S(x)$, $m(x) = x^{n-k}$, and $d = \lceil (n-k)/2 \rceil$. Because of (2), the resulting $\Lambda(x)$ satisfies $\deg \Lambda(x) \leq (n-k)/2$, and we have $\Gamma(x) = S(x)\Lambda(x) \bmod x^{n-k}$. If the number of errors does not exceed $(n-k)/2$, the condition $\deg \Gamma(x) < \deg \Lambda(x)$ will then be satisfied automatically.

III. The Algorithm

The Partial-Inverse Problem as stated in Section I can be solved by the following algorithm.

Proposed Algorithm:
Input: $b(x)$, $m(x)$, and d as in the problem statement.
Output: $\Lambda(x)$ as in the problem statement.

```
 1    if $\deg b(x) < d$ begin
 2        return $\Lambda(x) := 1$
 3    end
 4    $\Lambda^{(1)}(x) := 0,\ d_1 := \deg m(x),\ \kappa_1 := \operatorname{lcf} m(x)$
 5    $\Lambda^{(2)}(x) := 1,\ d_2 := \deg b(x),\ \kappa_2 := \operatorname{lcf} b(x)$
 6    loop begin
 7        $\Lambda^{(1)}(x) := \kappa_2\Lambda^{(1)}(x) - \kappa_1 x^{d_1-d_2}\Lambda^{(2)}(x)$
 8        $d_1 := \deg \big(b(x)\Lambda^{(1)}(x) \bmod m(x)\big)$
 9        if $d_1 < d$ begin
10            return $\Lambda(x) := \Lambda^{(1)}(x)$
11        end
12        $\kappa_1 := \operatorname{lcf} \big(b(x)\Lambda^{(1)}(x) \bmod m(x)\big)$
13        if $d_1 < d_2$ begin
14            $(\Lambda^{(1)}(x), \Lambda^{(2)}(x)) := (\Lambda^{(2)}(x), \Lambda^{(1)}(x))$
15            $(d_1, d_2) := (d_2, d_1)$
16            $(\kappa_1, \kappa_2) := (\kappa_2, \kappa_1)$
17        end
18    end                                                        □
```

Note that lines 14–16 simply swap $\Lambda^{(1)}(x)$ with $\Lambda^{(2)}(x)$, d_1 with d_2, and κ_1 with κ_2. The only actual computations are in lines 7 and 8.

The correctness of this algorithm will be proved in Section VI. In particular, we will see that the value of d_1 is reduced in every execution of line 8.

Note that lines 8 and 12 do not require the computation of the entire polynomial $b(x)\Lambda^{(1)}(x) \bmod m(x)$. Indeed, lines 8–12 can be replaced by the following loop:

Equivalent Alternative to Lines 8–12:

```
31        repeat
32            $d_1 := d_1 - 1$
33            if $d_1 < d$ begin
34                return $\Lambda(x) := \Lambda^{(1)}(x)$
35            end
36            $\kappa_1 :=$ coefficient of $x^{d_1}$ in
                      $b(x)\Lambda^{(1)}(x) \bmod m(x)$
37        until $\kappa_1 \neq 0$
```

In the special case where $m(x) = x^{\nu}$, line 36 amounts to

```
41        $\kappa_1 := b_{d_1}\Lambda_0^{(1)} + b_{d_1-1}\Lambda_1^{(1)} + \ldots + b_{d_1-\tau}\Lambda_\tau^{(1)}$
```

with $\tau \triangleq \deg \Lambda^{(1)}(x)$ and where $b_\ell \triangleq 0$ for $\ell < 0$. In the other special case where $m(x) = x^n - 1$ as in (10) below, line 36 becomes

```
51        $\kappa_1 := b_{d_1}\Lambda_0^{(1)} + b_{[d_1-1]}\Lambda_1^{(1)} + \ldots + b_{[d_1-\tau]}\Lambda_\tau^{(1)}$
```

with $b_{[\ell]} \triangleq b_{\ell \bmod n}$. In both cases, the proposed algorithm looks very much like, and is as efficient as, the Berlekamp-Massey algorithm [6].

IV. Decoding Reed-Solomon Codes via an Alternative Key Equation

Decoding Reed-Solomon codes (up to half the minimum distance) can be reduced rather directly to the Partial-Inverse Problem of Section I as follows.

Let F be a finite field, let $\beta_0, \ldots, \beta_{n-1}$ be n different elements of F, let $m(x) \triangleq \prod_{\ell=0}^{n-1}(x - \beta_\ell)$, let $F[x]/m(x)$ be the ring of polynomials modulo $m(x)$, and let ψ be the evaluation mapping

$$\psi : F[x]/m(x) \to F^n : a(x) \mapsto \big(a(\beta_0), \ldots, a(\beta_{n-1})\big), \tag{7}$$

which is a ring isomorphism. A Reed-Solomon code with blocklength n and dimension k may be defined as

$$\{c = (c_0, \ldots, c_n) \in F^n : \deg \psi^{-1}(c) < k\}, \tag{8}$$

usually with the additional condition that

$$\beta_\ell = \alpha^\ell \quad \text{for } \ell = 0, \ldots, n-1, \tag{9}$$

where $\alpha \in F$ is a primitive n-th root of unity. The condition (9) implies

$$m(x) = x^n - 1 \tag{10}$$

and turns ψ into a discrete Fourier transform [4]. However, (9) and (10) will not be required below.

Let $y = (y_0, \ldots, y_{n-1}) \in F^n$ be the received word, which we wish to decompose into

$$y = c + e \tag{11}$$

where $c \in \mathcal{C}$ is a codeword and where the Hamming weight of $e = (e_0, \ldots, e_{n-1}) \in F^n$ is as small as possible.

Let $C(x) \triangleq \psi^{-1}(c)$, and analogously $E(x) \triangleq \psi^{-1}(e)$ and $Y(x) \triangleq \psi^{-1}(y)$. Clearly, we have $\deg C(x) < k$ and $\deg E(x) < \deg m(x) = n$.

For any $e \in F^n$, we define the error locator polynomial

$$\Lambda_e(x) \triangleq \prod_{\substack{\ell \in \{0,\ldots,n-1\} \\ e_\ell \neq 0}} (x - \beta_\ell). \tag{12}$$

Clearly, $\deg \Lambda_e(x) = \mathrm{w_H}(e)$ and

$$E(x)\Lambda_e(x) \bmod m(x) = 0. \tag{13}$$

Theorem 1 (Alternative Key Equation). If $\mathrm{w_H}(e) \leq \frac{n-k}{2}$, then the error locator polynomial $\Lambda_e(x)$ satisfies

$$\deg\big(Y(x)\Lambda_e(x) \bmod m(x)\big) < n - \frac{n-k}{2} \tag{14}$$

Conversely, for any y and $e \in F^n$ and $t \in \mathbb{R}$ with

$$\mathrm{w_H}(e) \leq t \leq \frac{n-k}{2}, \tag{15}$$

if some nonzero $\Lambda(x) \in F[x]$ with $\deg \Lambda(x) \leq t$ satisfies

$$\deg\Big(Y(x)\Lambda(x) \bmod m(x)\Big) < n - t, \tag{16}$$

then $\Lambda(x)$ is a multiple of $\Lambda_e(x)$. □

The proof is not difficult, but omitted due to lack of space. We thus arrive at the following decoding procedure:

1) Compute $Y(x) = \psi^{-1}(y)$.
2) Run the algorithm of Section III with $b(x) = Y(x)$ and $d = \lceil \frac{n+k}{2} \rceil$. If $\mathrm{w_H}(e) \leq \frac{n-k}{2}$, then the polynomial $\Lambda(x)$ returned by the algorithm equals $\Lambda_e(x)$ up to a scale factor.
3) Complete decoding by any standard method [4] or by means of Proposition 1 below.

Note that in Step 2, because of (3), coefficients Y_ℓ of $Y(x)$ with

$$\ell < \ell_{\text{min}} \triangleq \begin{cases} k, & \text{if } n-k \text{ is even} \\ k+1, & \text{if } n-k \text{ is odd} \end{cases} \tag{17}$$

are irrelevant for finding $\Lambda(x)$ and can be set to zero. The remaining coefficients Y_ℓ are syndromes since $C_\ell = 0$ and $Y_\ell = E_\ell$ for $\ell \geq \ell_{\text{min}}$.

As mentioned, decoding can be completed by the following proposition:

Proposition 1. If $\Lambda(x)$ is a nonzero multiple of $\Lambda_e(x)$ with $\deg \Lambda(x) \leq n-k$, then

$$C(x) = \frac{Y(x)\Lambda(x) \bmod m(x)}{\Lambda(x)} \tag{18}$$

□

Proof: If $\Lambda(x)$ has the stated properties, then

$$\begin{aligned} &Y(x)\Lambda(x) \bmod m(x) \\ &\quad = C(x)\Lambda(x) \bmod m(x) + E(x)\Lambda(x) \bmod m(x) && (19) \\ &\quad = C(x)\Lambda(x), && (20) \end{aligned}$$

where the second term in (19) vanishes because of (13). □

Note that computing the numerator of (18) may be viewed as continuing the algorithm of Section III (line 36) with frozen $\Lambda^{(1)}(x) = \Lambda(x)$.

V. Key Elements of the Proof

We now turn to the proof of the algorithm proposed in Section III. In this section, we discuss some key elements of the proof; the actual proof will then be given in Section VI.

The pivotal part of the algorithm is line 7, which is explained by the following simple lemma. (The corresponding statement for the Berlekamp-Massey algorithm is the *two-wrongs-make-a-right* lemma, so called by J. L. Massey.)

Lemma 1. Let $m(x)$ be a polynomial over F with $\deg m(x) \geq 1$. For further polynomials $b(x), \Lambda^{(1)}(x), \Lambda^{(2)}(x) \in F[x]$, let

$$r^{(1)}(x) \triangleq b(x)\Lambda^{(1)}(x) \bmod m(x), \tag{21}$$

$$r^{(2)}(x) \triangleq b(x)\Lambda^{(2)}(x) \bmod m(x), \tag{22}$$

$d_1 \triangleq \deg r^{(1)}(x)$, $\kappa_1 \triangleq \operatorname{lcf} r^{(1)}(x)$, $d_2 \triangleq \deg r^{(2)}(x)$, $\kappa_2 \triangleq \operatorname{lcf} r^{(2)}(x)$, and assume $d_1 \geq d_2 \geq 0$. Then

$$\Lambda(x) \triangleq \kappa_2 \Lambda^{(1)}(x) - \kappa_1 x^{d_1-d_2}\Lambda^{(2)}(x) \tag{23}$$

satisfies

$$\deg\Big(b(x)\Lambda(x) \bmod m(x)\Big) < d_1. \tag{24}$$

□

Proof: From (23), we obtain

$$\begin{aligned} r(x) &\triangleq b(x)\Lambda(x) \bmod m(x) && (25) \\ &= \kappa_2 r^{(1)}(x) - \kappa_1 x^{d_1-d_2} r^{(2)}(x) && (26) \end{aligned}$$

by the natural ring homomorphism $F[x] \to F[x]/m(x)$. It is then obvious from (26) that $\deg r(x) < \deg r^{(1)}(x) = d_1$. □

A similar argument proves

Proposition 2 (Uniqueness of Solution). The solution $\Lambda(x)$ of the Partial-Inverse Problem of Section I is unique up to a scale factor. □

Proof: Let $\Lambda^{(1)}(x)$ and $\Lambda^{(2)}(x)$ be two solutions of the problem, which implies $\deg \Lambda^{(1)}(x) = \deg \Lambda^{(2)}(x) \geq 0$. Define $r^{(1)}(x)$ and $r^{(2)}(x)$ as in (21) and (22) and consider

$$\Lambda(x) \triangleq \Big(\operatorname{lcf} \Lambda^{(2)}(x)\Big)\Lambda^{(1)}(x) - \Big(\operatorname{lcf} \Lambda^{(1)}(x)\Big)\Lambda^{(2)}(x). \tag{27}$$

Then

$$\begin{aligned} r(x) &\triangleq b(x)\Lambda(x) \bmod m(x) && (28) \\ &= \Big(\operatorname{lcf} \Lambda^{(2)}(x)\Big) r^{(1)}(x) - \Big(\operatorname{lcf} \Lambda^{(1)}(x)\Big) r^{(2)}(x), && (29) \end{aligned}$$

which implies that $\Lambda(x)$ also satisfies (1). But (27) implies $\deg \Lambda(x) < \deg \Lambda^{(1)}(x)$, which is a contradiction unless $\Lambda(x) = 0$. Thus $\Lambda(x) = 0$, which means that $\Lambda^{(1)}(x)$ and $\Lambda^{(2)}(x)$ are equal up to a scale factor. □

Definition (Minimal Partial Inverse): For fixed nonzero $b(x)$ and $m(x) \in F[x]$ with $\deg b(x) < \deg m(x)$, a nonzero polynomial $\Lambda(x) \in F[x]$ is a *minimal partial inverse* of $b(x)$ if

$$\deg\Big(b(x)\Lambda^{(1)}(x) \bmod m(x)\Big) \leq \deg\Big(b(x)\Lambda(x) \bmod m(x)\Big) \tag{30}$$

(with $\Lambda^{(1)}(x) \neq 0$) implies $\deg \Lambda^{(1)}(x) \geq \deg \Lambda(x)$. □

The following lemma is the counterpart to Theorem 1 of [6].

Lemma 2 (Degree Change Lemma). For fixed nonzero $b(x)$ and $m(x) \in F[x]$ with $\deg b(x) < \deg m(x)$, let $\Lambda(x)$ be a minimal partial inverse of $b(x)$ and let

$$r(x) \triangleq b(x)\Lambda(x) \bmod m(x). \tag{31}$$

If

$$\deg \Lambda(x) \leq \deg m(x) - \deg r(x), \tag{32}$$

then any nonzero polynomial $\Lambda^{(1)}(x) \in F[x]$ such that

$$\deg \big(b(x)\Lambda^{(1)}(x) \bmod m(x)\big) < \deg r(x) \tag{33}$$

satisfies

$$\deg \Lambda^{(1)}(x) \geq \deg m(x) - \deg r(x). \tag{34}$$

□

The proof is given below.

Corollary: Assume everything as in Lemma 2 including (32) and (33). If (34) is satisfied with equality, then $\Lambda^{(1)}(x)$ is also a minimal partial inverse of $b(x)$.

Proof of Lemma 2: Assume that $\Lambda^{(1)}(x)$ is a nonzero polynomial that satisfies (33), i.e., the degree of

$$r^{(1)}(x) \triangleq b(x)\Lambda^{(1)}(x) \bmod m(x) \tag{35}$$

satisfies

$$\deg r^{(1)}(x) < \deg r(x). \tag{36}$$

Multiplying (31) by $\Lambda^{(1)}(x)$ and (35) by $\Lambda(x)$ yields

$$\Lambda^{(1)}(x)r(x) \equiv \Lambda(x)r^{(1)}(x) \mod m(x). \tag{37}$$

If we assume both (32) and (contrary to (34))

$$\deg \Lambda^{(1)}(x) < \deg m(x) - \deg r(x), \tag{38}$$

then (37) reduces to

$$\Lambda^{(1)}(x)r(x) = \Lambda(x)r^{(1)}(x). \tag{39}$$

But then (36) implies $\deg \Lambda^{(1)}(x) < \deg \Lambda(x)$, which is impossible because $\Lambda(x)$ is a minimal partial inverse. Thus (32) and (38) cannot hold simultaneously. □

VI. Proof of the Proposed Algorithm

We now prove the correctness of the algorithm proposed in Section III. To this end, we restate the algorithm with added assertions as follows.

Proposed Algorithm Restated:

```
1   if $\deg b(x) < d$ begin
2       return $\Lambda(x) := 1$
3   end
4   $\Lambda^{(1)}(x) := 0,\ d_1 := \deg m(x),\ \kappa_1 := \operatorname{lcf} m(x)$
5   $\Lambda^{(2)}(x) := 1,\ d_2 := \deg b(x),\ \kappa_2 := \operatorname{lcf} b(x)$
6   loop begin
        Assertions:
        $d_1 > d_2 \geq d$                                  (A.1)
        $\deg \Lambda^{(2)}(x) = \deg m(x) - d_1$           (A.2)
                     $> \deg \Lambda^{(1)}(x)$              (A.3)
        $\Lambda^{(2)}(x)$ is a minimal partial inverse     (A.4)
7       repeat
8           $\Lambda^{(1)}(x) := \kappa_2\Lambda^{(1)}(x) - \kappa_1 x^{d_1-d_2}\Lambda^{(2)}(x)$
            Assertions:
            $\deg(b(x)\Lambda^{(1)}(x) \bmod m(x)) < d_1$   (A.5)
            $\deg \Lambda^{(1)}(x) = \deg m(x) - d_2$       (A.6)
                          $> \deg \Lambda^{(2)}(x)$         (A.7)
9           $d_1 := \deg \big(b(x)\Lambda^{(1)}(x) \bmod m(x)\big)$
10          if $d_1 < d$ begin
                Assertion:
                $\Lambda^{(1)}(x)$ is a min. partial inverse (A.8)
11              return $\Lambda(x) := \Lambda^{(1)}(x)$
12          end
13          $\kappa_1 := \operatorname{lcf} \big(b(x)\Lambda^{(1)}(x) \bmod m(x)\big)$
14      until $d_1 < d_2$
        Assertion:
        $\Lambda^{(1)}(x)$ is a minimal partial inverse     (A.9)
15      $(\Lambda^{(1)}(x), \Lambda^{(2)}(x)) := (\Lambda^{(2)}(x), \Lambda^{(1)}(x))$
16      $(d_1, d_2) := (d_2, d_1)$
17      $(\kappa_1, \kappa_2) := (\kappa_2, \kappa_1)$
18  end
```

Note the added inner **repeat** loop (lines 7–14), which does not change the algorithm but helps to state its proof.

Throughout the algorithm (except at the very beginning, before the first execution of lines 9 and 13), d_1, d_2, κ_1, and κ_2 are defined as in Lemma 1, i.e., $d_1 = \deg r^{(1)}(x)$, $\kappa_1 = \operatorname{lcf} r^{(1)}(x)$, $d_2 = \deg r^{(2)}(x)$, and $\kappa_2 = \operatorname{lcf} r^{(2)}(x)$ for $r^{(1)}(x)$ and $r^{(2)}(x)$ as in (21) and (22).

Assertions (A.1)–(A.4) are easily verified, both from the initialization and from (A.6), (A.7), and (A.9).

As for (A.5), after the very first execution of line 8, we still have $d_1 = \deg m(x)$ (from line 4), which makes (A.5) obvious. For all later executions of line 8, (A.5) follows from Lemma 1.

As for (A.6) and (A.7), we note that line 8 changes the degree of $\Lambda^{(1)}(x)$ as follows:

- Upon entering the **repeat** loop, line 8 increases the degree of $\Lambda^{(1)}$ to

$$\deg \Lambda^{(2)}(x) + d_1 - d_2 = \deg m(x) - d_2 \tag{40}$$
$$> \deg \Lambda^{(2)}(x), \tag{41}$$

  which follows from (A.1)–(A.3).
- Subsequent executions of line 8 without leaving the **repeat** loop (i.e., without executing lines 15–17) do not change the degree of $\Lambda^{(1)}(x)$. (This follows from the fact that d_1 is smaller than in the first execution while $\Lambda^{(2)}(x)$, d_2, and $\kappa_2 \neq 0$ remain unchanged.)

Assertion (A.9) follows from the Corollary to Lemma 2 (with $\Lambda(x) = \Lambda^{(2)}(x)$ and $\deg r(x) = d_2$), which applies because $d_1 < d_2$ and (A.6). Because of (A.1), the same argument applies also to (A.8).

Finally, (A.1) and (A.6) imply that the polynomial $\Lambda(x)$ returned by the algorithm satisfies

$$\deg \Lambda(x) \leq \deg m(x) - d. \tag{42}$$

VII. Conclusion

We have proposed a new algorithm for decoding Reed-Solomon codes and polynomial remainder codes, and for computing inverses in $F[x]/m(x)$. In the special case where $m(x) = x^\nu$ or $m(x) = x^n - 1$, the proposed algorithm almost coincides with the Berlekamp-Massey algorithm, except that it processes the syndrome in reverse order.

Appendix: Extension to Polynomial Remainder Codes

Polynomial remainder codes [7], [9]–[11] are a class of codes that include Reed-Solomon codes as a special case. We briefly outline how decoding via the alternative key equation of Section IV generalizes to polynomial remainder codes, which can thus be decoded by the algorithm of Section III.

Let $m_0(x), \ldots, m_{n-1}(x) \in F[x]$ be relatively prime and let $m(x) \triangleq \prod_{\ell=0}^{n-1} m_\ell(x)$. Let $R_m \triangleq F[x]/m(x)$ denote the ring of polynomials modulo $m(x)$ and let $R_{m_\ell} \triangleq F[x]/m_\ell(x)$. The mapping (7) is generalized to the ring isomorphism

$$\begin{aligned} \psi : R_m \to R_{m_0} \times \ldots \times R_{m_{n-1}} : \\ a(x) \mapsto \psi(a) \triangleq \big(\psi_0(a), \ldots, \psi_{n-1}(a)\big) \end{aligned} \tag{43}$$

with $\psi_\ell(a) \triangleq a(x) \bmod m_\ell(x)$. Following [11], a polynomial remainder code may be defined as

$$\{c = (c_0, \ldots, c_{n-1}) \in R_{m_0} \times \ldots \times R_{m_{n-1}} : \deg \psi^{-1}(c) < K\} \tag{44}$$

where

$$K \triangleq \sum_{\ell=0}^{k-1} \deg m_\ell(x) \tag{45}$$

for some fixed k, $0 < k < n$. We also define

$$N \triangleq \deg m(x) = \sum_{\ell=0}^{n-1} \deg m_\ell(x). \tag{46}$$

As in Section IV, let $y = c + e$ be the received word with $c \in \mathcal{C}$, and let $C(x) \triangleq \psi^{-1}(c)$, $E(x) \triangleq \psi^{-1}(e)$, and $Y(x) \triangleq \psi^{-1}(y)$. Clearly, $\deg C(x) < K$ and $\deg E(x) < N$.

For such codes, the error locator polynomial

$$\Lambda_e(x) \triangleq \prod_{\substack{\ell \in \{0, \ldots, n-1\} \\ e_\ell \neq 0}} m_\ell(x) \tag{47}$$

and the error factor polynomial [11]

$$\Lambda_f(x) \triangleq m(x) / \gcd\big(E(x), m(x)\big) \tag{48}$$

do not, in general, coincide. However, if all moduli $m_\ell(x)$ are irreducible, then $\Lambda_f(x) = \Lambda_e(x)$.

We then have the following generalization of Theorem 1:

Theorem 2. For given y and e with $\deg \Lambda_f(x) \leq t \leq \frac{N-K}{2}$, assume that some nonzero polynomial $\Lambda(x)$ with $\deg \Lambda(x) \leq t$ satisfies

$$\deg \big(Y(x)\Lambda(x) \bmod m(x)\big) < N - t. \tag{49}$$

Then $\Lambda(x)$ is a multiple of $\Lambda_f(x)$. Conversely, $\Lambda(x) = \Lambda_f(x)$ is a polynomial of the smallest degree that satisfies (49). □

It follows that the decoding procedure of Section IV works also for polynomial remainder codes, except that n, k, and $\Lambda_e(x)$ are replaced by N, K, and $\Lambda_f(x)$, respectively. Moreover, $C(x)$ can still be recovered from $\Lambda(x)$ by means of (18) [11].

References

[1] I. S. Reed and G. Solomon, "Polynominal codes over certain finite fields," *J. SIAM*, vol. 8, pp. 300–304, Oct. 1962.

[2] E. R. Berlekamp, "Algebraic Coding Theory." New York: McGraw-Hill, 1968.

[3] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding Goppa codes," *Information and Control*, vol. 27, pp. 87–99, 1975.

[4] R. E. Blahut, "Algebraic Codes for Data Transmission." Cambridge University Press, Cambridge, UK, 2003.

[5] R. M. Roth, *Introduction to Coding Theory.* New York: Cambridge University Press, 2006.

[6] J. L. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Information Theory*, vol. 15, pp. 122-127, May 1969.

[7] A. Shiozaki, "Decoding of redundant residue polynomial codes using Euclid's algorithm," *IEEE Trans. Information Theory*, vol. 34, pp. 1351–1354, Sep. 1988.

[8] S. Gao, "A new algorithm for decoding Reed-Solomon codes," in *Communications, Information and Network Security*, V. Bhargava, H. V. Poor, V. Tarokh, and S.Yoon, Eds. Norwell, MA: Kluwer, 2003, vol. 712, pp. 55-68.

[9] J. J. Stone, "Multiple-burst error correction with the Chinese Remainder Theorem," *J. SIAM*, vol. 11, pp. 74–81, Mar. 1963.

[10] J.-H. Yu and H.-A. Loeliger, "On irreducible polynomial remainder codes," *IEEE Int. Symp. on Information Theory, Saint Petersburg, Russia, July 31–Aug. 5, 2011.*

[11] J.-H. Yu and H.-A. Loeliger, "On polynomial remainder codes," http://arxiv.org/abs/1201.1812.

[12] J. L. Dornstetter, "On the equivalence between Berlekamps's and Euclids's algorithms," *IEEE Trans. Information Theory*, vol. 33, pp. 428-431, May 1987.

[13] P. Fitzpatrick, "On the key equation," *IEEE Trans. Information Theory*, vol. 41, pp. 1290-1302, Sep. 1995.

[14] A. E. Heydtmann and J. M. Jensen, "On the equivalence of the Berlekamp-Massey and Euclidean algorithms for decoding," *IEEE Trans. Information Theory*, vol. 46, pp. 2614-2624, Nov. 2000.