Simultaneous Partial Inverses and Decoding Interleaved Reed–Solomon Codes

Jiun-Hung Yu, *Member, IEEE*, and Hans-Andrea Loeliger, *Fellow, IEEE*

***Abstract*— This paper introduces the simultaneous partial-inverse problem (SPI) for polynomials and develops its application to decoding interleaved Reed–Solomon codes beyond half the minimum distance. While closely related both to standard key equations and to well-known Padé approximation problems, the SPI problem stands out in several respects. First, the SPI problem has a unique solution (up to a scale factor), which satisfies a natural degree bound. Second, the SPI problem can be transformed (monomialized) into an equivalent SPI problem where all moduli are monomials. Third, the SPI problem can be solved by an efficient algorithm of the Berlekamp–Massey type. Fourth, decoding interleaved Reed–Solomon codes (or subfield-evaluation codes) beyond half the minimum distance can be analyzed in terms of a partial-inverse condition for the error pattern: if that condition is satisfied, then the (true) error locator polynomial is the unique solution of a standard key equation and can be computed in many different ways, including the well-known multi-sequence Berlekamp–Massey algorithm and the SPI algorithm of this paper. Two of the best performance bounds from the literature (the Schmidt–Sidorenko–Bossert bound and the Roth–Vontobel bound) are generalized to hold for the partial-inverse condition and thus to apply to several different decoding algorithms.**

***Index Terms*— Interleaved Reed–Solomon codes, subfield-evaluation codes, simultanenous partial-inverse problem, Euclidean algorithm, multi-sequence Berlekamp–Massey algorithm, performance bounds.**

I. Introduction

THIS paper revolves around the following problem and develops its application to decoding interleaved Reed–Solomon codes beyond half the minimum distance.

Simultaneous Partial-Inverse (SPI) Problem: For $i = 1, \ldots, L$, let $b^{(i)}(x)$ and $m^{(i)}(x)$ be polynomials over some field F with $\deg m^{(i)}(x) \geq 1$ and $\deg b^{(i)}(x) < \deg m^{(i)}(x)$. For fixed $\tau^{(i)} \in \mathbb{Z}$ with $0 \leq \tau^{(i)} \leq \deg m^{(i)}(x)$, find a nonzero polynomial $\Lambda(x) \in F[x]$ of the smallest degree such that

$$\deg \left(b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x) \right) < \tau^{(i)} \tag{1}$$

for all $i \in \{1, \ldots, L\}$. □

We will see that this problem has always a unique solution (up to a scale factor) and the solution satisfies

$$\deg \Lambda(x) \leq \sum_{i=1}^{L} \left(\deg m^{(i)}(x) - \tau^{(i)} \right). \tag{2}$$

Moreover, we will see that the SPI problem for general moduli $m^{(i)}(x)$ can efficiently be transformed ("monomialized") into an equivalent SPI problem with monomial moduli $m^{(i)}(x) = x^{\nu_i}$.

The special case[1] $L = 1$ was extensively discussed in [3]. In this paper, we address the generalization from $L = 1$ to $L > 1$, which is not obvious.

For $L > 1$, the SPI problem appears to be new, but it is closely related to a number of well-researched problems in coding and computer science including "key equations" for interleaved Reed–Solomon codes [4], [5], the multi-sequence linear-feedback shift-register (MLFSR) problem of [6]–[8], and generalizations of Padé approximation problems [9]–[11]. However, none of these related problems shares all the mentioned properties (unique solution, degree bound, monomialization) of the SPI problem.

By developing the decoding of interleaved Reed–Solomon codes around the SPI problem, we generalize and harmonize a number of key ideas from the literature, as will be detailed below.

We will consider codes as follows. Let $F = F_q$ be a finite field with q elements. The codewords are $L \times n$ arrays over F such that every row is a codeword in some Reed–Solomon code over F. We will only consider column errors, and we will not distinguish between columns with a single error and columns with many errors. The Reed–Solomon codes (for each row) consist of the codewords

$$\left\{ \left(a(\beta_0), \ldots, a(\beta_{n-1}) \right) : a(x) \in F[x] \quad \text{with } \deg a(x) < k \right\}, \tag{3}$$

where $\beta_0, \ldots, \beta_{n-1}$ are n different elements of F. Note that punctured Reed–Solomon codes are included and $\beta_\ell = 0$ (for a single index ℓ) is allowed. The dimension k will be allowed to depend on the row. However, for the further discussion in this section, we will assume that all row codes have the same dimension k.

Such interleaved Reed–Solomon codes can equivalently be viewed as punctured Reed–Solomon codes over F_{q^L} simply by replacing $F[x] = F_q[x]$ in (3) by $F_{q^L}[x]$ while the evaluation points $\beta_0, \ldots, \beta_{n-1}$ remain in F_q [4], [12], [13]. Note that

[1]Except that $b(x) = 0$ was excluded in [3].

Manuscript received December 21, 2016; revised October 1, 2017 and April 14, 2018; accepted August 28, 2018. Date of publication September 5, 2018; date of current version November 20, 2018. This paper was presented in part at the 2014 Allerton Conference on Communication, Control, and Computing [1] and the 2015 IEEE International Symposium on Information Theory [2].

J.-H. Yu is with the Department of Electrical and Computer Engineering, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: yuji@nctu.edu.tw).

H.-A. Loeliger is with the Department of Information Technology and Electrical Engineering, ETH Zurich, 8092 Zürich, Switzerland (email: loeliger@isi.ee.ethz.ch).

Communicated by Y. Mao, Associate Editor for Algebraic Coding Theory.

Digital Object Identifier 10.1109/TIT.2018.2868701

symbol errors in F_{q^L} correspond to column errors in the array code.

Decoding such array codes (or subfield-evaluation codes) beyond the Guruswami-Sudan decoding radius [14] was studied in [4], [12], [13], and [15]–[20]. Following [14], some of these papers use list-decoding algorithms [13], [16] while others use unique-decoding algorithms that return at most one codeword [4], [12], [15], [17]–[20]. The best unique-decoding algorithms can now correct t errors (column errors or F_{q^L}-symbol errors) up to

$$t \leq \frac{L}{L+1}(n-k) \tag{4}$$

with high probability if q is large [4], [12], [15]. For $L \geq n-k-1$, the bound (4) becomes

$$t < n-k, \tag{5}$$

which cannot be improved. (For small L, however, improvements over (4) have been demonstrated, cf. [11] and the references therein.)

Specifically, for t errors with random error values (uniformly distributed over all nonzero columns), the best bound on the probability P_f of a decoding failure (due to Schmidt *et al.* [4]) is

$$P_f \leq \gamma \frac{q^{-L(n-k)+(L+1)t}}{q-1} \tag{6}$$

with

$$\gamma \triangleq \left(\frac{q^L - q^{-1}}{q^L - 1}\right)^t. \tag{7}$$

(Note that $\gamma > 1$, but $\gamma \approx 1$ for any t of interest.) The bound (6) implies that the decoding algorithm of [4] decodes up to (4) errors with high probability if q is large.

Another type of bound, not relying on randomness, uses the rank of the error matrix $E \in F^{L\times n}$ that corrupts the transmitted (array-) codeword [20]. The decoding algorithm by Roth and Vontobel [20] corrects any t (column) errors provided that

$$t \leq \frac{n-k+\operatorname{rank}(E)-1}{2}, \tag{8}$$

which beats the guarantee in [4] by a margin of $\operatorname{rank}(E)/2$.

Note that [4] and [20] use different decoding algorithms, and the decoding algorithm of [4] assumes cyclic Reed–Solomon codes (as row codes) where $m(x) = x^n - 1$.

The bound (8) can also be used with random error values. For $t \leq L$, $\operatorname{rank}(E)$ is then likely to equal t, in which case (8) reduces to (5); for $t = n-k-1 \leq L$, (8) (by (117)) yields the same bound as (6) with $\gamma = 1$, which agrees with the bounds in [17] and [19], where different decoding algorithms are used.

In this paper, we define a partial-inverse condition (Definition 2) for the error pattern, which is always satisfied up to half the minimum distance and almost always satisfied almost up to the full minimum distance. If that condition is satisfied, then the (true) error locator polynomial is the unique solution of a standard key equation and can thus be computed in several different ways.

Specifically, we will show that (8) guarantees the partial-inverse condition to be satisfied. For random error values (as above), the probability for this condition not to hold will be shown to be bounded by (6), with the minor improvement that (7) is replaced by $\gamma = 1$. In this way, the scope of both (6) and (8) is widened.

The primary decoding algorithms for interleaved Reed–Solomon codes are based on the MLFSR algorithm by Feng and Tzeng [7] with corrections by Schmidt *et al.* [4] and Schmidt and Sidorenko [8] (but see also [21]). The complexity of this algorithm is $O(L(n-k)^2)$ additions and/or multiplications in F. (Asymptotically faster algorithms have been proposed [22] and will be discussed below.) However, the MLFSR algorithm is restricted to monomial moduli, which arise naturally from cyclic Reed–Solomon codes.

Beyond cyclic codes, for $L = 1$, it is a classical result that decoding general Reed–Solomon codes can be reduced to a key equation with monomial modulus [28], which is amenable to the MLFSR algorithm. (However, standard accounts of that method do not allow an evaluation point β_ℓ to equal zero.) For $L > 1$, such a transformation was used in [20]. In this paper, the same effect (without any constraints) is achieved by the monomialization of SPI problems, with the additional benefit that the partial-inverse condition is preserved. This transformation can be carried out, either by the Euclidean algorithm or by the partial-inverse algorithm of [3], with complexity $O(L(n-k)^2)$.

Finally, we propose algorithms to solve the SPI problem. The basic SPI algorithm is of the Berlekamp–Massey type. In the special case where $m^{(i)}(x) = x^{\nu_i}$, it looks very much like, and has the same complexity as, the MLFSR algorithm [7], [8]. However, the two algorithms are different: for $L = 1$, the MLFSR algorithm of [7] and [8] reduces to the Berlekamp–Massey algorithm [23] while the proposed SPI algorithm (Algorithm 3 of this paper) reduces to the reverse Berlekamp–Massey algorithm of [3].

As shown in [3], the reverse Berlekamp–Massey algorithm is easily translated into two other algorithms, one of them being a variation of the Euclidean algorithm by Sugiyama *et al.* [24]. The (reverse) Berlekamp–Massey algorithm and the Euclidean algorithm may thus be viewed as two versions of the same algorithm. In this paper, we extend this to $L > 1$: by easy translations of Algorithm 3, we obtain two other algorithms (Algorithms 10 and 11), one of which is of the Euclidean type and reminiscent of [6] rather than of [7]. (Yet another, quite different, "Euclidean" algorithm was proposed in [25].) For $L > 1$, no such connection between the (different) approaches of [6] and [7] has been described in the literature. However, the (reverse) Berlekamp–Massey version for monomial (or monomialized) SPI problems stands out by having the lowest complexity.

As mentioned, asymptotically faster algorithms for the MLFSR problem have been presented in [22] for cyclic row codes and in [5] for general row codes, both achieving $O(L^3(n-k)\log^2(n-k)\log\log\ (n-k))$. (Note that the asymptotic speed-up in $n-k$ is bought with the factor L^3.) It seems likely that such asymptotically fast algorithms can also be developed for the SPI problem, but this is not addressed in the present paper. In any case, the algorithms from [5] and [22] also profit from the performance bounds and the monomialization scheme of this paper.

In summary, we demonstrate that the SPI problem allows to harmonize and to generalize a number of ideas from the literature on interleaved Reed–Solomon codes. Specifically:

1) A general SPI problem can be transformed into an equivalent SPI problem with $m^{(i)}(x) = x^{\nu_i}$.
When applied to decoding, this monomialization preserves the partial-inverse condition (with the associated guarantees). We also show how the error evaluator polynomial (which is used, e.g., in Forney's formula, cf. Section III-A.3) can be transformed accordingly.
2) We show that the SPI problem can be solved by an efficient algorithm of the Berlekamp–Massey type.
In the Appendix, we also show that this algorithm is easily translated into two other algorithms, one of which is of the Euclidean type. For the MLFSR problem with $L > 1$, no such connection between the Berlekamp–Massey approach [7] and the Euclidean approach [6] has been demonstrated. However, for $L > 1$, the (reverse) Berlekamp–Massey version has lower complexity than the other versions.
3) Using the partial-inverse condition, we prove the Schmidt–Sidorenko–Bossert bound (6) (with $\gamma = 1$) and the Roth–Vontobel bound (8) for a range of algorithms including MLFSR algorithms (such as, e.g., [4] and [5]) and the SPI decoding algorithms of this paper.

The paper is structured as follows. In Section II, we discuss the SPI problem without regard to any algorithms or applications. In particular, we prove the degree bound (2) and we discuss the monomialization of the SPI problem. In Section III, we consider the decoding of interleaved Reed–Solomon codes. The pivotal concept in this section is a partial-inverse condition for the error pattern, which guarantees that the (correct) error locator polynomial can be computed by many different (well-known and new) algorithms. In Section IV, the bounds (6) and (8) are shown to apply to the partial-inverse condition.

In Section V, we return to the problem of actually solving SPI problems, for which we propose the reverse Berlekamp–Massey algorithm. In Section VI, we adapt and apply this algorithm to decoding interleaved Reed–Solomon codes.

The proof of the reverse Berlekamp–Massey algorithm is given in Appendix A. The other two versions (including the Euclidean version) of the algorithm are given in Appendix B. Section VII concludes the paper.

The reader is assumed to be familiar with the basics of algebraic coding theory [26]–[28] as well as with the notion of a ring homomorphism and its application to $F[x]$ [29].

We will use the following notation. The coefficient of x^d of a polynomial $b(x) \in F[x]$ will be denoted by b_d, and the leading coefficient (i.e., the coefficient of $x^{\deg b(x)}$) of a nonzero polynomial $b(x)$ will be denoted by $\operatorname{lcf} b(x)$. We will use "mod" both as in $r(x) = b(x) \bmod m(x)$ (the remainder of a division) and as in $b(x) \equiv r(x) \mod m(x)$ (a congruence modulo $m(x)$). We will also use "div" for polynomial division: if

$$a(x) = q(x)m(x) + r(x) \tag{9}$$

with $\deg r(x) < \deg m(x)$, then $q(x) = a(x) \operatorname{div} m(x)$ and $r(x) = a(x) \bmod m(x)$.

II. About the Simultaneous Partial-Inverse Problem

In this section, we consider the SPI problem without regard to any algorithms or applications. The properties and facts that are proved here are mostly straightforward generalizations from the case $L = 1$ from [3], but Theorem 1 (monomialization) requires some extra work.

A. Basic Properties

The SPI problem as defined in Section I has the following properties, which will be heavily used throughout the paper.

Proposition 1: The SPI problem has always a solution. $\square$

Proof: The polynomial $\Lambda(x) = m^{(1)}(x) \cdots m^{(L)}(x)$ satisfies $b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x) = 0$ for all i, which implies the existence of a solution for any $\tau^{(i)} \ge 0$. $\square$

Proposition 2: The solution $\Lambda(x)$ of an SPI problem is unique up to a scale factor in F. $\square$

Proof: Let $\Lambda'(x)$ and $\Lambda''(x)$ be two solutions of the problem, which implies $\deg \Lambda'(x) = \deg \Lambda''(x) \ge 0$. Define

$$r'^{(i)}(x) \triangleq b^{(i)}(x)\Lambda'(x) \bmod m^{(i)}(x) \tag{10}$$

$$r''^{(i)}(x) \triangleq b^{(i)}(x)\Lambda''(x) \bmod m^{(i)}(x) \tag{11}$$

and consider

$$\Lambda(x) \triangleq \Big(\operatorname{lcf} \Lambda''(x)\Big)\Lambda'(x) - \Big(\operatorname{lcf} \Lambda'(x)\Big)\Lambda''(x). \tag{12}$$

Then

$$r^{(i)}(x) \triangleq b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x) \tag{13}$$

$$= \Big(\operatorname{lcf} \Lambda''(x)\Big)r'^{(i)}(x) - \Big(\operatorname{lcf} \Lambda'(x)\Big)r''^{(i)}(x) \tag{14}$$

by the natural ring homomorphism $F[x] \to F[x]/m^{(i)}(x)$. Clearly, (14) implies that $\Lambda(x)$ also satisfies (1) for every $1 \le i \le L$. But (12) implies $\deg \Lambda(x) < \deg \Lambda'(x)$, which is a contradiction unless $\Lambda(x) = 0$. Thus $\Lambda(x) = 0$, which means that $\Lambda'(x)$ equals $\Lambda''(x)$ up to a scale factor. $\square$

Proposition 3 (Degree Bound): If $\Lambda(x)$ solves the SPI problem, then

$$\deg \Lambda(x) \le \sum_{i=1}^{L} \Big(\deg m^{(i)}(x) - \tau^{(i)}\Big). \tag{15}$$

$\square$

Proof: The case $\tau^{(i)} = \deg m^{(i)}(x)$ for all i is obvious. Otherwise, let $\nu_i \triangleq \deg m^{(i)}(x) - \tau^{(i)}$ and $\nu \triangleq \sum_{i=1}^{L} \nu_i$, and consider, for $i = 1, \ldots, L$, the linear mappings

$$\varphi_i : F^{\nu+1} \to F^{\nu_i} \tag{16}$$

given by

$$\begin{aligned}(\Lambda_0,\ldots,\Lambda_\nu) &\mapsto \Lambda(x) \triangleq \Lambda_0+\Lambda_1 x+\ldots+\Lambda_\nu x^\nu && (17)\\ &\mapsto r^{(i)}(x) \triangleq b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x) && (18)\\ &\mapsto (r_0^{(i)},\ldots,r^{(i)}_{\deg m^{(i)}(x)-1}) && (19)\\ &\mapsto (r^{(i)}_{\tau^{(i)}},\ldots,r^{(i)}_{\deg m^{(i)}(x)-1}). && (20)\end{aligned}$$

Note that a polynomial $\Lambda_0+\Lambda_1\, x+\ldots+\Lambda_\nu x^\nu$ satisfies (1) if and only if $(\Lambda_0,\ldots,\Lambda_\nu)\in\ker\varphi_i$. But

$$\begin{aligned}\dim\left(\bigcap_{i=1}^{L}\ker\varphi_i\right) &\geq \nu+1-\sum_{i=1}^{L}\nu_i && (21)\\ &= 1, && (22)\end{aligned}$$

i.e., $\left(\bigcap_{i=1}^{L}\ker\varphi_i\right)$ is not trivial. There thus exists a nonzero polynomial $\Lambda_0+\Lambda_1\, x+\ldots+\Lambda_\nu x^\nu$ that satisfies (1) simultaneously for $i=1,\ldots,L$. □

For occasional later use, the right-hand side of (15) will be denoted by

$$D \triangleq \sum_{i=1}^{L}\left(\deg m^{(i)}(x)-\tau^{(i)}\right). \tag{23}$$

Another obvious bound on the degree of the solution is

$$\deg\Lambda(x)\leq \deg\operatorname{lcm}\left(m^{(1)}(x),\ldots,m^{(L)}(x)\right), \tag{24}$$

where lcm denotes the common multiple of the smallest degree. In particular, we have

Proposition 4 (Monomial-SPI Degree Bound):[2] Assume $m^{(i)}(x)=x^{\nu_i}$ for $i=1,\ldots,L$. If $\Lambda(x)$ solves the SPI problem, then

$$\deg\Lambda(x)\leq \max_{i=1,\ldots,L}\nu_i. \tag{25}$$

□

The right side of (25) may be smaller or larger than (23).

B. Irrelevant Coefficients and Degree Reduction

Let $\Lambda(x)$ be the solution of a given SPI problem and let u be a (nonnegative) integer such that

$$u\geq\deg\Lambda(x). \tag{26}$$

Note that $u=D$ as in (23) qualifies by (15).

Proposition 5 (Irrelevant Coefficients): In the SPI problem, coefficients $b_\ell^{(i)}$ of $b^{(i)}(x)$ with

$$\ell<\tau^{(i)}-u \tag{27}$$

and coefficients $m_s^{(i)}$ of $m^{(i)}(x)$ with

$$s\leq\tau^{(i)}-u \tag{28}$$

have no effect on the solution $\Lambda(x)$. □

Proof: From (27) and (26), we obtain

$$\ell+\deg\Lambda(x)<\tau^{(i)}, \tag{29}$$

which proves the first claim. As for the second claim, we begin by writing

$$b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x) = b^{(i)}(x)\Lambda(x)-m^{(i)}(x)q^{(i)}(x) \tag{30}$$

for some $q^{(i)}(x)\in F[x]$ with

$$\deg q^{(i)}(x)<\deg\Lambda(x). \tag{31}$$

(If $q^{(i)}(x)\neq 0$, (31) follows from considering the leading coefficient of the right-hand side of (30) with $\deg b^{(i)}(x)<\deg m^{(i)}(x)$). From (28), (31), and (26), we then obtain

$$s+\deg q^{(i)}(x)<\tau^{(i)}. \tag{32}$$

The second claim then follows from (30) and (32). □

Irrelevant coefficients according to Proposition 5 may be set to zero without affecting the solution $\Lambda(x)$. In fact, such coefficients can be stripped off as follows.

Proposition 6 (Degree Reduction): For any $u>0$ satisfying (26), let

$$s^{(i)}\triangleq\max\{\tau^{(i)}-u,0\} \tag{33}$$

and define the polynomials $\tilde{b}^{(i)}(x)$ and $\tilde{m}^{(i)}(x)$ with

$$\tilde{b}_\ell^{(i)}\triangleq b^{(i)}_{\ell+s^{(i)}} \tag{34}$$

and

$$\tilde{m}_\ell^{(i)}\triangleq m^{(i)}_{\ell+s^{(i)}} \tag{35}$$

for $\ell\geq 0$. Then the modified SPI problem with $b^{(i)}(x)$, $m^{(i)}(x)$, and $\tau^{(i)}$ replaced by $\tilde{b}^{(i)}(x)$, $\tilde{m}^{(i)}(x)$, and $\tilde{\tau}^{(i)}\triangleq\tau^{(i)}-s^{(i)}$, respectively, has the same solution $\Lambda(x)$ as the original SPI problem. In addition, we have

$$b^{(i)}(x)\Lambda(x)\operatorname{div}m^{(i)}(x)=\tilde{b}^{(i)}(x)\Lambda(x)\operatorname{div}\tilde{m}^{(i)}(x) \tag{36}$$

□

Proof: Consider an auxiliary simultaneous partial-inverse problem with $b^{(i)}(x)$ replaced by $x^{s^{(i)}}\tilde{b}^{(i)}(x)$ and $m^{(i)}(x)$ replaced by $x^{s^{(i)}}\tilde{m}^{(i)}(x)$ (and $\tau^{(i)}$ unchanged). This auxiliary problem has the same solution as the original problem by Proposition 5. The equivalence of this auxiliary problem with the modified problem is obvious from (30). □

C. Monomialized SPI Problem

For a given SPI problem, let u be a (nonnegative) integer that satisfies (26). Moreover, let $n^{(i)}\triangleq\deg m^{(i)}(x)$.

It turns out that the given SPI problem (with general moduli $m^{(i)}(x)$) can be transformed into another SPI problem where (1) is replaced with

$$\deg\left(\tilde{b}^{(i)}(x)\Lambda(x) \bmod x^{n^{(i)}-\tau^{(i)}+u}\right)<u \tag{37}$$

with $\tilde{b}^{(i)}(x)$ as defined below. The precise statement is given as Theorem 1 below.

We will need the additional condition

$$n^{(i)}-\tau^{(i)}+u>0 \tag{38}$$

[2]Proposition 4 was pointed out by an anonymous reviewer.

for all $i \in \{1, \ldots, L\}$. Note that this condition does not entail any loss in generality: for $u > 0$, (38) is always satisfied, and by (26), $u = 0$ is admissible only if the SPI problem has the trivial solution $\Lambda(x) = 1$.

The polynomial $\tilde{b}^{(i)}(x)$ in (37) is defined as follows. Let

$$\overline{b}^{(i)}(x) \triangleq x^{n^{(i)}-1} b^{(i)}(x^{-1}). \tag{39}$$
$$\overline{m}^{(i)}(x) \triangleq x^{n^{(i)}} m^{(i)}(x^{-1}). \tag{40}$$

Moreover, let $w^{(i)}(x)$ be the inverse of

$$\overline{m}^{(i)}(x) \bmod x^{n^{(i)}-\tau^{(i)}+u} \tag{41}$$

in the ring $F[x]/x^{n^{(i)}-\tau^{(i)}+u}$; this inverse exists because $\overline{m}^{(i)}(0) \neq 0$, which implies that $\overline{m}^{(i)}(x)$ is relatively prime to $x^{n^{(i)}-\tau^{(i)}+u}$. Further, let

$$s^{(i)}(x) \triangleq \left(w^{(i)}(x)\overline{b}^{(i)}(x)\right) \bmod x^{n^{(i)}-\tau^{(i)}+u}, \tag{42}$$

and finally

$$\tilde{b}(x) \triangleq x^{n^{(i)}-\tau^{(i)}+u-1} s^{(i)}(x^{-1}). \tag{43}$$

Theorem 1 (Monomialized SPI Problem): For a given SPI problem, let u be an integer satisfying both (26) and (38). Then the modified SPI problem where $b^{(i)}(x)$, $m^{(i)}(x)$, and $\tau^{(i)}$ are replaced by $\tilde{b}^{(i)}(x)$ (as defined above), $x^{n^{(i)}-\tau^{(i)}+u}$, and u, respectively, has the same solution $\Lambda(x)$ as the original SPI problem. In addition, we have

$$b^{(i)}(x)\Lambda(x) \operatorname{div} m^{(i)}(x) = \tilde{b}^{(i)}(x)\Lambda(x) \operatorname{div} x^{n^{(i)}-\tau^{(i)}+u} \tag{44}$$

□

Note that the computation of $\tilde{b}^{(i)}(x)$ requires the computation of $w^{(i)}(x)$ (= the inverse of (41) in $F[x]/x^{n^{(i)}-\tau^{(i)}+u}$), which can be computed by the extended Euclidean algorithm or by the algorithms in [3, Sec IV] (which coincide with the SPI algorithms of Section V for $L = 1$).

Proof of Theorem 1: Consider the original SPI problem (1) and let $\Lambda(x)$ be its solution (which is unique up to a nonzero scale factor). Let

$$r^{(i)}(x) \triangleq b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x), \tag{45}$$

where $\deg r^{(i)}(x) < \tau^{(i)}$. We then write

$$r^{(i)}(x) = b^{(i)}(x)\Lambda(x) - q^{(i)}(x)m^{(i)}(x) \tag{46}$$

for some (unique) $q^{(i)}(x)$ with

$$\deg q^{(i)}(x) < \deg \Lambda(x) \le u. \tag{47}$$

Now let

$$\overline{\Lambda}(x) \triangleq x^u \Lambda(x^{-1}) \tag{48}$$
$$\overline{q}^{(i)}(x) \triangleq x^{u-1} q^{(i)}(x^{-1}) \tag{49}$$
$$\overline{r}^{(i)}(x) \triangleq x^{\tau^{(i)}-1} r^{(i)}(x^{-1}). \tag{50}$$

By substituting x^{-1} for x in (46) and multiplying both sides by $x^{n^{(i)}+u-1}$ (i.e., reversing (46)), we obtain

$$x^{n^{(i)}+u-\tau^{(i)}}\overline{r}^{(i)}(x) = \overline{b}^{(i)}(x)\overline{\Lambda}(x) - \overline{q}^{(i)}(x)\overline{m}^{(i)}(x). \tag{51}$$

We then have

$$\overline{b}^{(i)}(x)\overline{\Lambda}(x) \equiv \overline{q}^{(i)}(x)\overline{m}^{(i)}(x) \mod x^{n^{(i)}-\tau^{(i)}+u} \tag{52}$$

and thus

$$s^{(i)}(x)\overline{\Lambda}(x) \equiv \overline{q}^{(i)}(x) \mod x^{n^{(i)}-\tau^{(i)}+u} \tag{53}$$

with $s^{(i)}(x)$ defined in (42). Note that $\deg \overline{\Lambda}(x) \le u$ and $\deg \overline{q}^{(i)}(x) < u$ from (47).

We now write (53) as

$$s^{(i)}(x)\overline{\Lambda}(x) = \overline{p}^{(i)}(x)x^{n^{(i)}-\tau^{(i)}+u} + \overline{q}^{(i)}(x) \tag{54}$$

for some (unique) $\overline{p}^{(i)}(x)$ with $\deg \overline{p}^{(i)}(x) < \deg \overline{\Lambda}(x) \le u$, and let

$$p^{(i)}(x) \triangleq x^{u-1}\overline{p}^{(i)}(x^{-1}). \tag{55}$$

By substituting x^{-1} for x in (54) and multiplying both sides by $x^{n^{(i)}-\tau^{(i)}+2u-1}$, we obtain

$$\tilde{b}^{(i)}(x)\Lambda(x) = x^{n^{(i)}-\tau^{(i)}+u} q^{(i)}(x) + p^{(i)}(x), \tag{56}$$

from which we have

$$\deg\left(\tilde{b}^{(i)}(x)\Lambda(x) \bmod x^{n^{(i)}-\tau^{(i)}+u}\right) < u. \tag{57}$$

We have arrived at the modified SPI problem.

Now, let $\tilde{\Lambda}(x)$ denote the solution of the modified SPI problem, which implies

$$\deg \tilde{\Lambda}(x) \le \deg \Lambda(x). \tag{58}$$

In the following, we will show

$$\deg \tilde{\Lambda}(x) \ge \deg \Lambda(x). \tag{59}$$

When this is established, we have $\deg \tilde{\Lambda}(x) = \deg \Lambda(x)$; thus $\Lambda(x)$ solves the modified SPI problem, and (44) is obvious from (56).

It remains to prove (59). We begin by writing

$$\deg\left(\tilde{b}^{(i)}(x)\tilde{\Lambda}(x) \bmod x^{n^{(i)}-\tau^{(i)}+u}\right) < u \tag{60}$$

with $\deg \tilde{\Lambda}(x) \le u$ because of (58). Now, let

$$\tilde{p}^{(i)}(x) \triangleq \tilde{b}^{(i)}(x)\tilde{\Lambda}(x) \bmod x^{n^{(i)}-\tau^{(i)}+u}, \tag{61}$$

where $\deg \tilde{p}^{(i)}(x) < u$. We then write

$$\tilde{b}^{(i)}(x)\tilde{\Lambda}(x) = x^{n^{(i)}-\tau^{(i)}+u}\tilde{q}^{(i)}(x) + \tilde{p}^{(i)}(x) \tag{62}$$

for some (unique) $\tilde{q}^{(i)}(x)$ with $\deg \tilde{q}^{(i)}(x) < \deg \tilde{\Lambda}(x)$.

Further, let

$$\hat{\Lambda}(x) \triangleq x^u \tilde{\Lambda}(x^{-1}) \tag{63}$$
$$\hat{p}^{(i)}(x) \triangleq x^{u-1}\tilde{p}^{(i)}(x^{-1}) \tag{64}$$
$$\hat{q}^{(i)}(x) \triangleq x^{u-1}\tilde{q}^{(i)}(x^{-1}). \tag{65}$$

By substituting x^{-1} for x in (62) and multiplying both sides by $x^{n^{(i)}-\tau^{(i)}+2u-1}$, we obtain

$$s^{(i)}(x)\hat{\Lambda}(x) = x^{n^{(i)}-\tau^{(i)}+u}\hat{p}^{(i)}(x) + \hat{q}^{(i)}(x), \tag{66}$$

where $s^{(i)}(x)$ is obtained from (43). It follows that

$$s^{(i)}(x)\hat{\Lambda}(x) \equiv \hat{q}^{(i)}(x) \mod x^{n^{(i)}-\tau^{(i)}+u} \tag{67}$$

and therefore

$$w^{(i)}(x)\overline{b}^{(i)}(x)\hat{\Lambda}(x) \equiv \hat{q}^{(i)}(x) \mod x^{n^{(i)}-\tau^{(i)}+u}. \tag{68}$$

By multiplying both sides of (68) by $\overline{m}^{(i)}(x)$, we obtain

$$\overline{b}^{(i)}(x)\hat{\Lambda}(x) \equiv \hat{q}^{(i)}(x)\overline{m}^{(i)}(x) \mod x^{n^{(i)}-\tau^{(i)}+u}, \quad (69)$$

and therefore

$$\overline{b}^{(i)}(x)\hat{\Lambda}(x) - \hat{q}^{(i)}(x)\overline{m}^{(i)}(x) = x^{n^{(i)}-\tau^{(i)}+u}\hat{r}^{(i)}(x) \quad (70)$$

holds for some (unique) $\hat{r}^{(i)}(x)$ with $\deg \hat{r}^{(i)}(x) < \tau^{(i)}$. Let

$$\tilde{r}^{(i)}(x) \overset{\triangle}{=} x^{\tau^{(i)}-1}\hat{r}^{(i)}(x^{-1}). \quad (71)$$

By substituting x^{-1} for x in (70) and multiplying both sides by $x^{n^{(i)}+u-1}$, we obtain

$$b^{(i)}(x)\tilde{\Lambda}(x) - \tilde{q}^{(i)}(x)m^{(i)}(x) = \tilde{r}^{(i)}(x) \quad (72)$$

and therefore

$$\deg\left(b^{(i)}(x)\tilde{\Lambda}(x) \bmod m^{(i)}(x)\right) < \tau^{(i)}. \quad (73)$$

Thus $\tilde{\Lambda}(x)$ satisfies (1), and (59) follows. □

III. Utilizing the SPI Problem for Decoding Interleaved Reed–Solomon Codes

A. About Interleaved Reed–Solomon Codes

We first establish some (more or less standard) concepts and the pertinent notation.

1) Array Codes and Evaluation Isomorphism: Let $F = F_q$ be a finite field with q elements. We consider array codes as defined in Section I where codewords are $L \times n$ arrays over F such that each row is a codeword in some Reed–Solomon code as in (3) with blocklength n and dimension $k^{(i)}$, $i \in \{1, \ldots, L\}$. Let

$$m(x) \overset{\triangle}{=} \prod_{\ell=0}^{n-1}(x - \beta_\ell), \quad (74)$$

where $\deg m(x) = n$. Let ψ be the evaluation mapping

$$\psi : F[x]/m(x) \to F^n : a(x) \mapsto \left(a(\beta_0), \ldots, a(\beta_{n-1})\right), \quad (75)$$

which is a ring isomorphism. The row code (3) can then be described as

$$\{c \in F^n : \deg \psi^{-1}(c) < k^{(i)}\}. \quad (76)$$

The standard definition of Reed–Solomon codes requires, in addition, that

$$\beta_\ell = \alpha^\ell \quad \text{for } \ell = 0, \ldots, n-1, \quad (77)$$

where $\alpha \in F$ is a primitive n-th root of unity. This additional condition implies

$$m(x) = x^n - 1, \quad (78)$$

which makes the code cyclic and turns ψ into a discrete Fourier transform [26]. However, (77) and (78) will not be required below. In particular, the set $\{\beta_0, \ldots, \beta_{n-1}\}$ will be permitted to contain 0.

In general, the inverse mapping ψ^{-1} may be computed by Lagrange interpolation or according to the Chinese remainder theorem.

We are primarily interested in the special case where all row codes have the same dimension $k^{(i)} = k$. Nonetheless, we allow the general case, for which we define

$$k_{\max} \overset{\triangle}{=} \max\{k^{(i)}, 1 \le i \le L\}, \quad (79)$$

$$k_{\min} \overset{\triangle}{=} \min\{k^{(i)}, 1 \le i \le L\}, \quad (80)$$

and

$$k_{\text{avg}} \overset{\triangle}{=} \frac{1}{L}\sum_{i=1}^{L} k^{(i)}. \quad (81)$$

Note that the overall rate of the array code is k_{avg}/n. Throughout the paper, we will assume

$$k_{\max} < n. \quad (82)$$

2) Notation for Individual Rows and Error Support: Let $Y = C + E \in F^{L\times n}$ be the received word where $C \in F^{L\times n}$ is the transmitted (array-) codeword and $E \in F^{L\times n}$ is the error pattern. Further, let $y^{(i)}$ be the i-th row of the matrix Y, let $c^{(i)}$ be the i-th row of C, and let $e^{(i)}$ be the i-th row of E. We then have $y^{(i)} = c^{(i)} + e^{(i)}$, $i = 1, \ldots, L$, and therefore

$$Y^{(i)}(x) = C^{(i)}(x) + E^{(i)}(x) \quad (83)$$

where $Y^{(i)}(x) \overset{\triangle}{=} \psi^{-1}(y^{(i)})$, $C^{(i)}(x) \overset{\triangle}{=} \psi^{-1}(c^{(i)})$, and $E^{(i)}(x) \overset{\triangle}{=} \psi^{-1}(e^{(i)})$. Note that $\deg E^{(i)}(x) < \deg m(x) = n$ and $\deg C^{(i)}(x) < k^{(i)}$.

We will index the columns of codewords and error patterns beginning with zero as in $E = (e_0, \ldots, e_{n-1})$, and we define

$$U_E \overset{\triangle}{=} \left\{\ell \in \{0, \ldots, n-1\} : e_\ell \neq 0\right\}, \quad (84)$$

the index set of the nonzero columns of E. Note that $e_\ell = 0$ if and only if

$$E^{(i)}(\beta_\ell) = 0 \quad \text{for all } i \in \{1, \ldots, L\}. \quad (85)$$

We will only consider column errors, and we will not distinguish between columns with a single error and columns with many errors.

3) Error Locator Polynomial and Interpolation: We define the error locator polynomial as[3]

$$\Lambda_E(x) \overset{\triangle}{=} \prod_{\ell \in U_E}(x - \beta_\ell). \quad (86)$$

(In particular, $\Lambda_E(x) = 1$ if $E = 0$.) Note that

$$\deg \Lambda_E(x) = |U_E| = \text{number of column errors}. \quad (87)$$

If $\Lambda_E(x)$ is known and satisfies $\deg \Lambda_E \le n - k_{\max}$, the polynomial $C^{(i)}(x)$ for each $i \in \{1, \ldots, L\}$ can be recovered in many different ways (cf. the discussion in [3]), e.g., by means of

$$C^{(i)}(x) = \frac{Y^{(i)}(x)\Lambda_E(x) \bmod m(x)}{\Lambda_E(x)} \quad (88)$$

or by means of

$$C^{(i)}(x) = Y^{(i)}(x) \bmod \tilde{m}(x) \quad (89)$$

[3]In the literature, the error locator polynomial is more often defined with $(x - \beta_j)$ in (86) replaced by $(1 - x\beta_j)$.

with $\tilde{m}(x) \stackrel{\triangle}{=} m(x)/\Lambda_E(x)$ according to [3, Proposition 9]. For large L, (89) may be more attractive.

If we need to recover the actual codeword $c^{(i)}$ (rather than just the polynomial $C^{(i)}(x)$), interpolation according to (88) or (89) requires the additional computation of $\psi(C^{(i)}(x))$. Alternatively, it may be attractive to compute the error pattern $e^{(i)} = y^{(i)} - c^{(i)}$ from

$$Q^{(i)}(x) \stackrel{\triangle}{=} Y^{(i)}(x)\Lambda_E(x) \operatorname{div} m(x) \tag{90}$$

and Forney's formula

$$e_\ell^{(i)} \stackrel{\triangle}{=} \begin{cases} 0 & \text{if } \Lambda_E(\beta_\ell) \neq 0 \\ \frac{Q^{(i)}(\beta_\ell)m'(\beta_\ell)}{\Lambda'_E(\beta_\ell)} & \text{if } \Lambda_E(\beta_\ell) = 0 \end{cases} \tag{91}$$

for $\ell = 0, 1, \ldots, n-1$, where $\Lambda'_E(x)$ and $m'(x)$ denote the formal derivatives of $\Lambda_E(x)$ and $m(x)$, respectively.

4) Shiozaki–Gao Error-Locator Equation [30], [31]: From (85) and (86), we have

$$E^{(i)}(x)\Lambda_E(x) \bmod m(x) = 0 \tag{92}$$

and thus

$$Y^{(i)}(x)\Lambda_E(x) \bmod m(x) = C^{(i)}(x)\Lambda_E(x) \bmod m(x). \tag{93}$$

If $|U_E| \leq n - k_{\max}$, we obtain

$$\deg\left(Y^{(i)}(x)\Lambda_E(x) \bmod m(x)\right) < k^{(i)} + |U_E| \tag{94}$$

for all $i \in \{1, \ldots, L\}$.

B. Partial-Inverse Conditions

We now begin to develop the specific approach of this paper. We first note that any finite set of inequalities of the form (1) (indexed by $i \in \{1, \ldots, L\}$, with suitable polynomials $b^{(i)}(x)$ and $m^{(i)}(x)$, and with suitable integers $\tau^{(i)} \in \mathbb{Z}$) implicitly defines an SPI problem.

Definition 1 (SPI Solution): The solution of such an SPI problem will be called the *SPI solution* of the inequalities. □

Clearly, the SPI solution is unique, up to a nonzero scale factor in F.

For a given code as above and some arbitrary, but fixed, error pattern $E \in F^{L\times n}$, we now consider the conditions

$$\deg\left(E^{(i)}(x)\Lambda(x) \bmod m(x)\right) < k^{(i)} + |U_E|, \tag{95}$$

$i = 1, \ldots, L$. Note that $\Lambda(x) = \Lambda_E(x)$ satisfies (95) by (92).

Definition 2 (Partial-Inverse Condition): E satisfies the *partial-inverse condition* if $|U_E| \leq n - k_{\max}$ and the SPI solution of (95) is $\Lambda_E(x)$. □

We will see in Section IV that E always satisfies the partial-inverse condition if $|U_E| \leq (n-k_{\max})/2$, and E is very likely to satisfy the partial-inverse condition if $|U_E| \leq \frac{L}{L+1}(n-k_{\text{avg}})$.

We now proceed to derive several equivalent formulations of the partial-inverse condition: the received-word version (Proposition 7), the syndrome version (Theorem 2), and the monomialized version (Theorem 3).

Proposition 7 (Received-Word Partial-Inverse Condition): Let $Y = C + E \in F^{L\times n}$ where C is a codeword and $|U_E| \leq n - k_{\max}$. Then E satisfies the partial-inverse condition if and only if the SPI solution of

$$\deg\left(Y^{(i)}(x)\Lambda(x) \bmod m(x)\right) < k^{(i)} + |U_E|, \tag{96}$$

$i = 1, \ldots, L$, is $\Lambda_E(x)$. □

Proof: Consider any fixed E with $|U_E| \leq n - k_{\max}$. From (92), the solution of the SPI problem (95) has degree at most $|U_E|$. From (94), the solution of the SPI problem (96) has degree at most $|U_E|$ as well. But for any $\Lambda(x)$ with $\deg\Lambda(x) \leq |U_E|$, we have

$$\deg\left(C^{(i)}(x)\Lambda(x) \bmod m(x)\right) < k^{(i)} + |U_E| \tag{97}$$

for $i = 1, \ldots, L$. Thus $\Lambda(x)$ satisfies (95) if and only if it satisfies (96). □

The conditions (96) can be simplified as follows. For a given code and received word $Y = C + E$, let

$$S^{(i)}(x) \stackrel{\triangle}{=} Y^{(i)}_{k^{(i)}} + Y^{(i)}_{k^{(i)}+1}x + \ldots + Y^{(i)}_{n-1}x^{n-k^{(i)}-1} \tag{98}$$

$$= E^{(i)}_{k^{(i)}} + E^{(i)}_{k^{(i)}+1}x + \ldots + E^{(i)}_{n-1}x^{n-k^{(i)}-1} \tag{99}$$

(the syndrome polynomials) and

$$\tilde{m}^{(i)}(x) \stackrel{\triangle}{=} m_{k^{(i)}} + m_{k^{(i)}+1}x + \ldots + m_n x^{n-k^{(i)}}. \tag{100}$$

Theorem 2 (Syndrome-Based Partial-Inverse Condition): Let $Y = C + E \in F^{L\times n}$ where C is a codeword and $|U_E| \leq n - k_{\max}$. Then E satisfies the partial-inverse condition if and only if the SPI solution of

$$\deg\left(S^{(i)}(x)\Lambda(x) \bmod \tilde{m}^{(i)}(x)\right) < |U_E|, \tag{101}$$

$i = 1, \ldots, L$, is $\Lambda_E(x)$. Moreover, if E satisfies the partial-inverse condition, then

$$Y^{(i)}(x)\Lambda_E(x) \operatorname{div} m(x) = S^{(i)}(x)\Lambda_E(x) \operatorname{div} \tilde{m}^{(i)}(x). \tag{102}$$

□

Note that (102) is the polynomial $Q^{(i)}(x)$ in (90), which is required in Forney's formula (91).

For $|U_E| > 0$, the proof of Theorem 2 follows from Proposition 6 with $u = |U_E|$ and $s^{(i)} = k^{(i)}$. For $|U_E| = 0$, (95), (101), and (102) are always satisfied.

The partial-inverse condition for general $m(x)$ can be reduced to a partial-inverse condition for $m(x) = x^{n-k^{(i)}}$ as follows. For a given code and received word $Y = C + E$, let

$$\deg\left(\breve{S}^{(i)}(x)\Lambda(x) \bmod x^{n-k^{(i)}}\right) < |U_E| \tag{103}$$

be the monomialization of (101) according to Theorem 1 (with $b^{(i)}(x) = S^{(i)}(x)$, $\tilde{b}^{(i)}(x) = \breve{S}^{(i)}(x)$, $u = |U_E|$, $n^{(i)} = n - k^{(i)}$ and $n^{(i)} - \tau^{(i)} + u = n - k^{(i)}$).

Note that the computation of $\breve{S}^{(i)}(x)$ from $S^{(i)}(x)$ does not depend on $|U_E|$. Note also that the condition (38) translates to (82).

Theorem 3 (Monomialized Partial-Inverse Condition): Let $Y = C + E \in F^{L\times n}$ where C is a codeword and $|U_E| \leq n - k_{\max}$. Then E satisfies the partial-inverse condition if and only if the SPI solution of (103) is $\Lambda_E(x)$. Moreover, if E satisfies the partial-inverse condition, then

$$Y^{(i)}(x)\Lambda_E(x) \operatorname{div} m(x) = \breve{S}^{(i)}(x)\Lambda_E(x) \operatorname{div} x^{n-k^{(i)}}. \tag{104}$$

□

Note that (104) is the polynomial $Q^{(i)}(x)$ in (90) and (91). The proof of Theorem 3 is immediate from Theorems 1 and 2.

The complexity of computing $\breve{S}^{(i)}(x)$ is determined by the complexity of computing the inverse (41) and the multiplication (42), both mod $x^{n-k^{(i)}}$. If the inverse is computed with the Euclidean algorithm (or with the partial-inverse algorithm of [3]), the complexity of these computations is $O\big(L(n-k^{(i)})^2\big)$ additions and/or multiplications in F. (Asymptotical speed-ups are certainly possible, but outside the scope of this paper.)

In summary, for $|U_E| \le n-k_{\max}$, the SPI solutions of (95), (96), (101), and (103) coincide. In Section IV, we will see that this SPI solution is very likely to be $\Lambda_E(x)$.

C. Computing the Error Locator Polynomial

If E satisfies the partial-inverse condition, $\Lambda_E(x)$ can be computed in many different ways. Let $S^{(i)}(x)$, $\tilde{m}^{(i)}(x)$, and $\breve{S}^{(i)}(x)$ be defined as in (98), (100), and (103), respectively.

Proposition 8 (Key Equations): If E satisfies the partial-inverse condition, then $\Lambda_E(x)$ is the unique (up to a scale factor) nonzero polynomial $\Lambda(x)$ of smallest degree such that

$$\deg\left(S^{(i)}(x)\Lambda(x) \bmod \tilde{m}^{(i)}(x)\right) < \deg\Lambda(x) \qquad (105)$$

or, equivalently, such that

$$\deg\left(\breve{S}^{(i)}(x)\Lambda(x) \bmod x^{n-k^{(i)}}\right) < \deg\Lambda(x) \qquad (106)$$

for all $i \in \{1,\ldots,L\}$. □

Proof: Assume that E satisfies the partial-inverse condition. Then $\Lambda(x) = \Lambda_E(x)$ is the SPI solution of (101) and (103). □

Note that (106) is a standard key equation, which can be derived and solved in many different ways. The point of Proposition 8 is the guarantee from the partial-inverse condition.

In order to compute $\Lambda_E(x)$ with an MLFSR algorithm as in [6]–[8], we need a slightly different formulation.

Proposition 9 (MLFSR Problems): Assume that E satisfies the partial-inverse condition. Then the smallest integer κ such that there exists a nonzero polynomial $\Lambda(x)$ with $\deg\Lambda(x) \le \kappa$ such that

$$\deg\left(S^{(i)}(x)\Lambda(x) \bmod \tilde{m}^{(i)}(x)\right) < \kappa \qquad (107)$$

or, equivalently, such that

$$\deg\left(\breve{S}^{(i)}(x)\Lambda(x) \bmod x^{n-k^{(i)}}\right) < \kappa \qquad (108)$$

for all $i \in \{1,\ldots,L\}$ is $\kappa = \deg\Lambda_E(x)$. Moreover, $\Lambda(x) = \Lambda_E(x)$ is the unique (up to a scale factor) such polynomial $\Lambda(x)$. □

Proof: Again, the proof follows from noting that $\Lambda(x) = \Lambda_E(x)$ is the SPI solution of (101) and (103). □

Note that (108) is an MLFSR problem as in [6]–[8]. The point of Proposition 9 is the guarantee from the partial-inverse condition.

Instead of using an MLFSR algorithm, we can solve either of the key equations (105) and (106) by solving SPI problems as shown in Algorithms 1 and 2, respectively (shown in framed boxes). Note that line 1 initializes τ to $\max_i \{\deg\tilde{m}^{(i)}(x)\}$. Clearly, these algorithms will always terminate, and they will return the correct error locator $\Lambda_E(x)$ if $\Lambda_E(x)$ satisfies the partial-inverse condition.

Algorithm 1: Error Location by SPI Algorithm

Input: $S^{(i)}(x)$ for $i = 1,\ldots,L$.

Output: nonzero $\Lambda(x) \in F[x]$, a candidate for the error locator $\Lambda_E(x)$ (up to a scale factor).

```
1   $\tau := n - k_{\min}$
2   loop
3       solve the SPI problem[a] with $\tau^{(i)} = \tau$,
          $m^{(i)}(x) = \tilde{m}^{(i)}(x)$, and $b^{(i)}(x) = S^{(i)}(x)$
4       if $\deg\Lambda(x) > n - k_{\max}$,
              declare "decoding failure"
5       if (105) holds, return $\Lambda(x)$
6       $\tau := \tau - 1$
7   end
```

[a]omitting indices i with $\deg\tilde{m}^{(i)}(x) < \tau$

Algorithm 2: Error Location by Monomial-SPI Alg.

Same as Algorithm 1, but with $\breve{S}^{(i)}(x)$ instead of $S^{(i)}(x)$, $x^{n-k^{(i)}}$ instead of $\tilde{m}^{(i)}(x)$, and checking (106) instead of (105).

It might seem that Algorithms 1 and 2 are inefficient. However, the SPI algorithms of this paper (cf. Section V) have this property: computing the solution for $\tau^{(i)} = \tau = t$ is effected by first computing the solution for $\tau^{(i)} = t+1$. In other words, line 3 of Algorithms 1 and 2 for $\tau = t$ is naturally implemented as continuing the computation for $\tau = t+1$. Moreover, the check of (105) (or of (106)) can be naturally integrated into these algorithms (cf. Section VI). When implemented in this way, the complexity of Algorithm 2 is $O(L(n-k_{\min})(n-k_{\max}))$ additions and/or multiplications in F, which agrees with the complexity of the MLFSR algorithm of [7] and [8]. No matter which algorithm is used to produce a candidate $\Lambda(x)$ for $\Lambda_E(x)$, it may be helpful to check the condition

$$m(x) \bmod \Lambda(x) = 0, \qquad (109)$$

which guarantees that $\Lambda(x)$ is a valid error locator polynomial (i.e., a product of different factors $(x-\beta_\ell)$, $\ell \in \{0,\ldots,n-1\}$, up to a scale factor). If this condition is not satisfied, then decoding failure should be declared.

IV. Successful-Decoding Guarantees and Probabilities

We have seen in Section III-C that decoding (by any of the mentioned methods) will certainly succeed if the error pattern E satisfies the partial-inverse condition. In this section, we analyze conditions and probabilities for this to happen. Our main results are Theorems 4 and 5 below.

A. Roth–Vontobel Bound

Theorem 4 generalizes a result from [20] (paraphrased in (8)) from the specific decoding algorithm of [20] to any decoding algorithm as in Section III-C.

Theorem 4: Let r_E be the rank of the error pattern $E \in F^{L\times n}$ as a matrix over F. If

$$|U_E| \le \frac{n - k_{\max} + r_E - 1}{2} \tag{110}$$

then E satisfies the partial-inverse condition. □

The partial-inverse condition is thus implied by (110). In particular, any nonzero error pattern E with $|U_E| \le (n - k_{\max})/2$ satisfies the partial-inverse condition.

Note that (110) implies $|U_E| < n - k_{\max}$ (since $r_E \le |U_E|$). Theorem 4 then follows immediately from the following lemma, which is inspired by [17] and [18] and, especially, [20].

Lemma 1: Let $E \in F^{L\times n}$ be an error pattern with rank r_E and

$$2|U_E| < n - k_{\max} + r_E. \tag{111}$$

Then any nonzero polynomial $\Lambda(x)$ such that

$$\deg\left(E^{(i)}(x)\Lambda(x) \bmod m(x)\right) < k^{(i)} + |U_E| \tag{112}$$

for all $i \in \{1,\ldots,L\}$ is a multiple of $\Lambda_E(x)$. □

Proof: The lemma trivially holds for $E = 0$. We now assume $U_E \ne \emptyset$. For any $s \in U_E$, let $\tilde{U}_s$ be a subset of U_E that contains s such that $|U_E| - |\tilde{U}_s| = r_E - 1$ and, in addition, the columns e_ℓ of E with $\ell \in (U_E \setminus \tilde{U}_s) \cup \{s\}$ are linearly independent. (Note that such a set $\tilde{U}_s$ exists for any $s \in U_E$.) Let $\tilde{e} \in F^n$ be a linear combination of rows of E such that $\tilde{e}_\ell = 0$ for $\ell \notin \tilde{U}_s$ and $\tilde{e}_s = 1$. (Note that such $\tilde{e}$ exists.) Finally, we define $\tilde{E}_s(x) \overset{\triangle}{=} \psi^{-1}(\tilde{e})$. We thus have $\tilde{E}_s(\beta_\ell) = 0$ for $\ell \notin \tilde{U}_s$ and $\tilde{E}_s(\beta_s) = 1$. Note also that $\tilde{E}_s(x)$ is a linear combination of $E^{(1)}(x),\ldots,E^{(L)}(x)$.

Now assume that some nonzero $\Lambda(x) \in F[x]$ satisfies (112). It follows that we also have

$$\deg\left(\tilde{E}_s(x)\Lambda(x) \bmod m(x)\right) < k_{\max} + |U_E| \tag{113}$$

for $i = 1,\ldots,L$. We then write

$$\tilde{E}_s(x)\Lambda(x) = g(x)m(x) + \tilde{E}_s(x)\Lambda(x) \bmod m(x) \tag{114}$$

according to the division theorem. But $\tilde{E}_s(x)$, and thus $\tilde{E}_s(x)\Lambda(x)$, has at least $n - |\tilde{U}_s|$ zeros in the set $\{\beta_0,\ldots,\beta_{n-1}\}$. Using $|\tilde{U}_s| = |U_E| - r_E + 1$ and (111), we obtain

$$n - |\tilde{U}_s| \ge |U_E| + k_{\max}. \tag{115}$$

Thus $\tilde{E}_s(x)\Lambda(x) \bmod m(x)$ has at least $|U_E| + k$ zeros in the set $\{\beta_0,\ldots,\beta_{n-1}\}$, which contradicts (113) unless $\tilde{E}_s(x)\Lambda(x) \bmod m(x) = 0$. Thus $\tilde{E}_s(x)\Lambda(x) \bmod m(x) = 0$ for all $s \in U_E$. It follows that $\Lambda(\beta_\ell) = 0$ for all $s \in U_E$, which implies that $\Lambda(x)$ is a multiple of $\Lambda_E(x)$. □

It is instructive to consider Theorem 4 for random errors. For any fixed U_E, assume that the nonzero columns of E are uniformly and independently distributed over all possible nonzero columns. In the special case where $|U_E| \le L$, it is then very likely that E has rank $|U_E|$ (as quantified by Proposition 10 below), in which case (110) reduces to $|U_E| < n - k_{\max}$. In other words, E is very likely to satisfy the partial-inverse condition provided that

$$|U_E| \le \min\{L, n - k_{\max} - 1\}. \tag{116}$$

If (116) holds, the probability that E does not satisfy the partial-inverse condition is bounded by

Proposition 10 (Full-Rank Probability): Assume $0 < |U_E| \le L$. If the $|U_E|$ nonzero columns of $E \in F^{L\times n}$ are uniformly and independently distributed over $F^L \setminus \{0\}$, then

$$\Pr\left(r_E \ne |U_E|\right) < \frac{q^{-L+|U_E|}}{q-1} \tag{117}$$

with $q = |F|$. □

This is certainly standard but, for the convenience of the reader, we give a (short) proof.

Proof: Assume $0 < |U_E| \le L$. It is easily seen that

$$\Pr\left(r_E = |U_E|\right) = \frac{(q^L-1)(q^L-q)\cdots(q^L-q^{|U_E|-1})}{(q^L-1)^{|U_E|}} \tag{118}$$

$$= \frac{(q^L-q)\cdots(q^L-q^{|U_E|-1})}{(q^L-1)^{|U_E|-1}} \tag{119}$$

where the numerator of (118) is the number of ways of picking $|U_E|$ linearly independent column vectors. The numerator of (119) can be written as

$$q^{L(|U_E|-1)}\left(1-q^{-(L-1)}\right)\cdots\left(1-q^{-(L-|U_E|+1)}\right) \tag{120}$$

and thus

$$\Pr\left(r_E = |U_E|\right) > \left(1-q^{-(L-1)}\right)\cdots\left(1-q^{-(L-|U_E|+1)}\right) \tag{121}$$

$$> 1 - \sum_{i=1}^{|U_E|-1} q^{-(L-i)} \tag{122}$$

$$> 1 - \frac{q^{-(L-|U_E|)}}{q-1} \tag{123}$$

□

B. Schmidt–Sidorenko–Bossert Bound

We now turn to random errors beyond the full-rank case. Theorem 5 generalizes a result from [4] (paraphrased in (6)).

Theorem 5: For $L > 1$ and any error support set U_E with $0 < |U_E| \le n - k_{\max}$, assume that the $|U_E|$ nonzero columns of $E \in F^{L\times n}$ are uniformly and independently distributed over $F^L \setminus \{0\}$. Then the probability P_{fpi} that E does not satisfy the partial-inverse condition is bounded by

$$P_{\text{fpi}} < \frac{q^{-L(n-k_{\text{avg}})+(L+1)|U_E|}}{q-1} \tag{124}$$

□

The proof is given in Section IV-C below. The right side of (124) agrees with the bound (14) of [4] except for the factor (7). (But [4] considers a specific decoding algorithm and only cyclic Reed–Solomon codes.)

For $|U_E| \le L$, both (117) and (124) apply. In general, (124) is much stronger than (117), but the two bounds agree in the special case where $|U_E| = n - k_{\text{avg}} - 1$.

Note that (124) implies that E satisfies the partial-inverse condition (with high probability, if q is large) as long as

$$|U_E| \le \min\left\{n - k_{\max},\ \frac{L}{L+1}(n - k_{\text{avg}})\right\}. \tag{125}$$

This cannot be improved:

Proposition 11: (125) is a necessary condition for E to satisfy the partial-inverse condition. □

Proof: Assume that E satisfies the partial-inverse condition, which means that $|U_E| \le n - k_{\max}$ and $\Lambda(x) = \Lambda_E(x)$ is the SPI solution of (95). Applying the degree bound (Proposition 3) to (95) yields

$$\deg \Lambda(x) \le \sum_{i=1}^{L} \Big(n - \big(k^{(i)} + |U_E|\big)\Big) \tag{126}$$

$$= L\big(n - k_{\text{avg}} - |U_E|\big) \tag{127}$$

and thus $|U_E|(1 + L) \le L(n - k_{\text{avg}})$. □

For $k^{(1)} = \ldots = k^{(L)} = k_{\text{avg}}$, (125) reduces to

$$|U_E| \le \frac{L}{L+1}(n - k_{\text{avg}}), \tag{128}$$

which cannot be improved by allowing general $k^{(1)}, \ldots, k^{(L)}$.

C. Proof of Theorem 5

Let $L > 1$ and let U be an arbitary, but fixed, subset of $\{0, \ldots, n-1\}$ such that $0 < |U| \le n - k_{\max}$. Assume that the error pattern $E \in F^{L\times n}$ has support set $U_E = U$, and the nonzero columns of E are uniformly and independently distributed over $F^L \setminus \{0\}$.

If E does not satisfy the partial-inverse condition, then there exists a nonzero polynomial $\Lambda(x)$ with $\deg \Lambda(x) < |U_E|$ such that

$$\deg\big(E^{(i)}(x)\Lambda(x) \bmod m(x)\big) < k^{(i)} + |U_E| \tag{129}$$

for all $i = 1, \ldots, L$.

Let $\mathcal{E}_U$ be the set of all the possible error patterns $E \in F^{L\times n}$ with support set $U_E = U$. Let $\mathcal{E}_{\text{fpi}} \subset \mathcal{E}_U$ be the set of all $E \in \mathcal{E}_U$ that admit some $\Lambda(x) \in F[x]$ with $0 \le \deg \Lambda(x) < |U|$ that satisfies (129) for all $i \in \{1, \ldots, L\}$. Then,

$$P_{\text{fpi}} \le \frac{|\mathcal{E}_{\text{fpi}}|}{|\mathcal{E}_U|} = \frac{|\mathcal{E}_{\text{fpi}}|}{(q^L - 1)^{|U|}} \tag{130}$$

It thus remains to bound $|\mathcal{E}_{\text{fpi}}|$.

For $t = 0, \ldots, |U| - 1$, let $\mathcal{L}_t$ be the set of monic polynomials $\Lambda(x) \in F[x]$ with $\deg \Lambda(x) < |U|$ and with exactly t zeros in the set $\mathcal{B}_U \triangleq \{\beta_\ell : \ell \in U\}$.

Lemma 2: For any fixed $\Lambda(x) \in \mathcal{L}_t$, the number of error patterns $E \in \mathcal{E}_U$ that satisfy (129) is upper bounded by $q^{L(2|U|-(n-k_{\text{avg}})-t)}$. □

The proof will be given below. We then have

$$|\mathcal{E}_{\text{fpi}}| \le \sum_{t=0}^{|U|-1} |\mathcal{L}_t|\, q^{L(2|U|-(n-k_{\text{avg}})-t)}. \tag{131}$$

Lemma 3:

$$|\mathcal{L}_t| = \binom{|U|}{t}(q-1)^{|U|-t-1}. \tag{132}$$

□

The proof is given below. Thus (131) becomes

$$|\mathcal{E}_{\text{fpi}}| \le \sum_{t=0}^{|U|-1} \binom{|U|}{t}(q-1)^{|U|-t-1} q^{L(2|U|-(n-k_{\text{avg}})-t)} \tag{133}$$

$$= w \sum_{t=0}^{|U|-1} \binom{|U|}{t}(q-1)^{-t} q^{-Lt} \tag{134}$$

$$< w \sum_{t=0}^{|U|} \binom{|U|}{t}\Big((q-1)^{-1} q^{-L}\Big)^t \tag{135}$$

$$= w\big(1 + (q-1)^{-1} q^{-L}\big)^{|U|} \tag{136}$$

$$= \frac{q^{L(|U|-(n-k_{\text{avg}}))}}{q-1}\Big((q-1)q^L + 1\Big)^{|U|} \tag{137}$$

with

$$w \triangleq (q-1)^{|U|-1} q^{L(2|U|-(n-k_{\text{avg}}))} \tag{138}$$

in (134)–(136). From (130), we then have

$$P_{\text{fpi}} < \frac{q^{L(|U|-(n-k_{\text{avg}}))}}{q-1}\left(\frac{q^{L+1} - q^L + 1}{q^L - 1}\right)^{|U|} \tag{139}$$

$$= \frac{q^{-L(n-k_{\text{avg}}-|U|)+|U|}}{q-1}\left(\frac{q^L - (q^{L-1} - q^{-1})}{q^L - 1}\right)^{|U|} \tag{140}$$

and (124) follows if $L > 1$.

For the proof of Lemma 2, we will use the following elementary fact.

Proposition 12: The number of nonzero polynomials over F of degree at most ν and with $\mu \le \nu$ prescribed zeros in F (and allowing additional zeros in F) is $|F|^{\nu-\mu+1} - 1$. □

Proof of Lemma 2: Consider the polynomial $E^{(i)}(x) = \psi^{-1}(e^{(i)})$ where $e^{(i)}$ is a row of E, and let $\tilde{E}^{(i)}(x) \triangleq E^{(i)}(x)\Lambda(x) \bmod m(x)$. From (75), we have

$$\tilde{E}^{(i)}(\beta_\ell) = e_{i,\ell}\Lambda(\beta_\ell) \tag{141}$$

where $e_{i,\ell}$ denotes the element in row i and column ℓ of E. From (129), we have $\deg \tilde{E}^{(i)}(x) < k^{(i)} + |U|$. But (141) implies that $\tilde{E}^{(i)}(x)$ has at least $n - |U| + t$ zeros in prescribed positions: $e_{i,\ell} = 0$ for $\ell \notin U$ and $\Lambda(x)$ has t zeros in $\mathcal{B}_U = \{\beta_\ell : \ell \in U\}$. By Proposition 12, the number of such polynomials $\tilde{E}^{(i)}$ is bounded by $q^{2|U|-(n-k^{(i)})-t}$, and putting all rows together yields the lemma. □

Proof of Lemma 3: Consider nonzero polynomials $\Lambda(x) \in F[x]$ with $\deg \Lambda(x) < |U|$ and with t prescribed zeros in $\mathcal{B}_U$ $(= \{\beta_\ell : \ell \in U\})$ and no other zeros in $\mathcal{B}_U$. The number of such polynomials $\Lambda(x)$ is $(q-1)^{|U|-t}$, as is obvious from the ring isomorphism

$$F[x]/m_U(x) \to F^{|U|} : \Lambda(x) \mapsto \big(\Lambda(\beta'_1), \ldots, \Lambda(\beta'_{|U|})\big) \tag{142}$$

with $m_U(x) \triangleq \prod_{\ell\in U}(x - \beta_\ell)$ and $\{\beta'_1, \ldots, \beta'_{|U|}\} \triangleq \mathcal{B}_U$. Lemma 3 then follows from noting that it counts only monic polynomials. □

V. The Reverse–Berlekamp Massey SPI Algorithm

We now consider algorithms to solve the SPI problem. The basic algorithm (the reverse Berlekamp–Massey algorithm) is stated as Algorithm 3 in the framed box. The important

Algorithm 3: Basic SPI Algorithm
(Reverse Berlekamp–Massey algorithm)
Input: $b^{(i)}(x), m^{(i)}(x), \tau^{(i)}$ for $i = 1, \ldots, L$.
Output: $\Lambda(x)$ as in the problem statement.

```
1    for $i = 1, \ldots, L$ begin
2        $\Lambda^{(i)}(x) := 0$
3        $d^{(i)} := \deg m^{(i)}(x)$
4        $\kappa^{(i)} := \operatorname{lcf} m^{(i)}(x)$
5    end
6    $\Lambda(x) := 1$
7    $\delta := \max_{i \in \{1,\ldots,L\}} \left(\deg m^{(i)}(x) - \tau^{(i)}\right)$
8    $i := 1$
9    loop begin
10       repeat
11           if $i > 1$ begin $i := i - 1$ end
12           else begin
13               if $\delta \leq 0$ return $\Lambda(x)$
14               $i := L$
15               $\delta := \delta - 1$
16           end
17           $d := \delta + \tau^{(i)}$
18           $\kappa :=$ coefficient of $x^d$ in
                 $b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x)$
19       until $\kappa \neq 0$
         ________________________________
20       if $d < d^{(i)}$ begin
21           swap $(\Lambda(x), \Lambda^{(i)}(x))$
22           swap $(d, d^{(i)})$
23           swap $(\kappa, \kappa^{(i)})$
24           $\delta := d - \tau^{(i)}$
25       end
         ________________________________
26       $\Lambda(x) := \kappa^{(i)}\Lambda(x) - \kappa x^{d-d^{(i)}}\Lambda^{(i)}(x)$
27   end
```

See also the refinement in Algorithm 4 below.

Algorithm 4: Monomial-SPI Algorithm
In the special case $m^{(i)}(x) = x^{\nu_i}$,
line 18 of Algorithm 3 amounts to

$$18 \qquad \kappa := \begin{cases} 0, & \text{if } d \geq \nu_i \\ b_d^{(i)}\Lambda_0 + b_{d-1}^{(i)}\Lambda_1 + \ldots + b_{d-s}^{(i)}\Lambda_s, & \\ \quad \text{if } d < \nu_i, & \end{cases}$$

with $s \overset{\triangle}{=} \deg \Lambda(x)$ and $b_\ell^{(i)} \overset{\triangle}{=} 0$ for $\ell < 0$.

special case where $m^{(i)}(x) = x^{\nu_i}$ is stated as Algorithm 4. (Algorithm 4 is strikingly similar to the MLFSR algorithm of [7] and [8], but it is nonetheless a different algorithm.) Since every SPI problem can be efficiently monomialized, Algorithm 4 is the preferred SPI algorithm.

Two variations of Algorithm 3 are given in Appendix B. These variations generalize the Quotient Saving Algorithm and the Remainder Saving Algorithm (a Euclidean algorithm) of [3] to $L > 1$. However, for $L > 1$, these algorithms are less attractive than Algorithm 4.

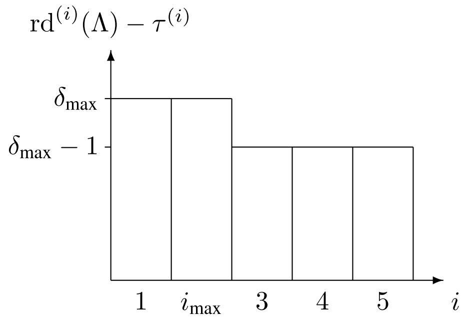

Fig. 1. Illustration of (144) and (145) for $i_{\max} = 2$.

A. Beginning to Explain the Algorithm

Lines 1–8 of Algorithm 3 are for initialization; the nontrivial part begins with line 9. Note that lines 21–23 simply swap $\Lambda(x)$ with $\Lambda^{(i)}(x)$, d with $d^{(i)}$, and κ with $\kappa^{(i)}$. The only actual computations are in lines 18 and 26.

We now begin to explain the algorithm (but the actual proof of correctness will be deferred to Appendix A). To this end, we define the following quantities. For any nonzero $\Lambda(x)$ and any $i \in \{1, 2, \ldots, L\}$, let

$$\mathrm{rd}^{(i)}(\Lambda) \overset{\triangle}{=} \deg\left(b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x)\right), \tag{143}$$

$$\delta_{\max}(\Lambda) \overset{\triangle}{=} \max_{i \in \{1,\ldots,L\}} \left(\mathrm{rd}^{(i)}(\Lambda) - \tau^{(i)}\right), \tag{144}$$

and

$$i_{\max}(\Lambda) \overset{\triangle}{=} \max \operatorname*{argmax}_{i \in \{1,\ldots,L\}} \left(\mathrm{rd}^{(i)}(\Lambda) - \tau^{(i)}\right), \tag{145}$$

the largest among the indices i that maximize $\mathrm{rd}^{(i)}(\Lambda) - \tau^{(i)}$, cf. Figure 1.

At any given time, the algorithm works on the polynomial $\Lambda(x)$. The inner **repeat** loop (lines 10–19) computes the quantities defined in (143)–(145): between lines 19 and 20, we have

$$i = i_{\max}(\Lambda), \quad \delta = \delta_{\max}(\Lambda), \; d = \mathrm{rd}^{(i)}(\Lambda), \tag{146}$$

and also

$$\kappa = \operatorname{lcf}\left(b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x)\right). \tag{147}$$

In particular, the very first execution of the **repeat** loop (with $\Lambda(x) = 1$) yields

$$i = \max \operatorname*{argmax}_{i \in \{1,\ldots,L\}} \left(\deg b^{(i)} - \tau^{(i)}\right), \tag{148}$$

$d = \deg b^{(i)}(x)$, and $\kappa = \operatorname{lcf} b^{(i)}(x)$ between lines 19 and 20.

In the special case $L = 1$, lines 11–17 (excluding line 13) amount to $d := d - 1$; in this case, the algorithm reduces to the partial-inverse algorithm of [3].

The only exit from the algorithm is line 13. Since $\delta \geq \delta_{\max}(\Lambda)$, the condition $\delta \leq 0$ guarantees that $\Lambda(x)$ satisfies (1).

The algorithm maintains the auxiliary polynomials $\Lambda^{(i)}(x)$, $i = 1, \ldots, L$, which are all initialized to $\Lambda^{(i)}(x) = 0$.

Thereafter, however, $\Lambda^{(i)}(x)$ become nonzero (after their first respective execution of lines 21–23) and satisfy

$$i_{\max}(\Lambda^{(i)}) = i. \tag{149}$$

The heart of the algorithm is line 26, which cancels the leading term in

$$b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x) \tag{150}$$

(except for the first execution for each index i, see below). Line 26 is explained by the following lemma.

Lemma 4 (Remainder Decreasing Lemma): Let $\Lambda'(x)$ and $\Lambda''(x)$ be nonzero polynomials such that $i \triangleq i_{\max}(\Lambda') = i_{\max}(\Lambda'')$ and $\mathrm{rd}^{(i)}(\Lambda') \geq \mathrm{rd}^{(i)}(\Lambda'')$. Then $\delta_{\max}(\Lambda') \geq \delta_{\max}(\Lambda'')$ and the polynomial

$$\Lambda(x) \triangleq \kappa''\Lambda'(x) - \kappa' x^{d'-d''}\Lambda''(x) \tag{151}$$

with $d' \triangleq \mathrm{rd}^{(i)}(\Lambda')$, $\kappa' \triangleq \mathrm{lcf}(b^{(i)}(x)\Lambda'(x) \bmod m^{(i)}(x))$, $d'' \triangleq \mathrm{rd}^{(i)}(\Lambda'')$, and $\kappa'' \triangleq \mathrm{lcf}(b^{(i)}(x)\Lambda''(x) \bmod m^{(i)}(x))$ satisfies both

$$\mathrm{rd}^{(i)}(\Lambda) < \mathrm{rd}^{(i)}(\Lambda') \tag{152}$$

and

$$\delta_{\max}(\Lambda) \leq \delta_{\max}(\Lambda') \tag{153}$$

and either

$$i_{\max}(\Lambda) < i_{\max}(\Lambda'), \tag{154}$$

or

$$\delta_{\max}(\Lambda) < \delta_{\max}(\Lambda'). \tag{155}$$

□

Proof: First, $\delta_{\max}(\Lambda') \geq \delta_{\max}(\Lambda'')$ is obvious from the assumptions. For the rest of proof, we define for every $\ell \in \{1, \ldots, L\}$

$$r'^{(\ell)}(x) \triangleq b^{(\ell)}(x)\Lambda'(x) \bmod m^{(\ell)}(x) \tag{156}$$
$$r''^{(\ell)}(x) \triangleq b^{(\ell)}(x)\Lambda''(x) \bmod m^{(\ell)}(x) \tag{157}$$

and we obtain from (151)

$$r^{(\ell)}(x) \triangleq b^{(\ell)}(x)\Lambda(x) \bmod m^{(\ell)}(x) \tag{158}$$
$$= \kappa'' r'^{(\ell)}(x) - \kappa' x^{d'-d''} r''^{(\ell)}(x) \tag{159}$$

by the natural ring homomorphism $F[x] \to F[x]/m^{(\ell)}(x)$. Moreover, we define

$$\delta^{(\ell)}(\Lambda) \triangleq \mathrm{rd}^{(\ell)}(\Lambda) - \tau^{(\ell)} \tag{160}$$
$$\delta^{(\ell)}(\Lambda') \triangleq \mathrm{rd}^{(\ell)}(\Lambda') - \tau^{(\ell)} \tag{161}$$
$$\delta^{(\ell)}(\Lambda'') \triangleq \mathrm{rd}^{(\ell)}(\Lambda'') - \tau^{(\ell)} \tag{162}$$

(cf. Figure 1 for Λ, Λ', and Λ'', respectively). By the stated assumptions, we have for $\ell = i$

$$\delta^{(i)}(\Lambda') = d' - d'' + \delta^{(i)}(\Lambda''), \tag{163}$$

and we obtain from (159)

$$\delta^{(i)}(\Lambda) < \delta^{(i)}(\Lambda'), \tag{164}$$
$$\delta^{(\ell)}(\Lambda) \leq \delta^{(i)}(\Lambda') \text{ for } \ell < i, \tag{165}$$

and

$$\delta^{(\ell)}(\Lambda) < \delta^{(i)}(\Lambda') \quad \text{for } \ell > i. \tag{166}$$

Clearly, (164) implies (152); (164)–(166) together imply both (153) and either (154) or (155) (or both). □

It follows from (152)–(155) that the algorithm makes progress and eventually terminates.

For each index $i \in \{1, \ldots, L\}$, when line 26 is executed for the very first time, it is necessarily preceded by the swap in lines 21–23. In this case, line 26 reduces to

$$\Lambda(x) := -\left(\mathrm{lcf}\, m^{(i)}(x)\right) x^{\deg m^{(i)}(x) - \mathrm{rd}^{(i)}(\Lambda')}\Lambda'(x) \tag{167}$$

where $\Lambda'(x)$ is the value of $\Lambda(x)$ before the swap. It follows, in particular, that $\deg \Lambda(x) > \deg \Lambda'(x)$.

In any case, we always have

$$\deg\left(b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x)\right) < d \tag{168}$$

after executing line 26.

Finally, we note that every execution of the swap in lines 21–23 strictly reduces $d^{(i)}$. We also note that the execution of line 24 results in

$$\delta = \begin{cases} \delta_{\max}(\Lambda), & \text{if } \Lambda(x) \neq 0 \\ \deg m^{(i)} - \tau^{(i)}, & \text{if } \Lambda(x) = 0, \end{cases} \tag{169}$$

where the second case happens only once—the very first time— for each index $i \in \{1, \ldots, L\}$.

Theorem 6: Algorithm 3 returns the solution of the SPI problem. □

The proof will be given in Appendix A.

B. Complexity of Algorithm 3

Let

$$\hat{D} \triangleq L \max_{i \in \{1,\ldots,L\}} \left(\deg m^{(i)}(x) - \tau^{(i)}\right). \tag{170}$$

and note that D as in (23) satisfies $D \leq \hat{D}$.

Theorem 7 (Number of Iterations): The number N_{it} of executions of line 18 of Algorithm 3 is

$$N_{\mathrm{it}} = \hat{D} + L \deg \Lambda(x) \tag{171}$$
$$\leq \hat{D} + LD \tag{172}$$

where $\Lambda(x)$ is the solution of the SPI problem. □

The step from (171) to (172) is obvious from (15). The proof of (171) will be given in Appendix A-C.

In the special case addressed by Algorithm 4, with $m^{(i)}(x) = x^{\nu_i}$ for $i = 1, \ldots, L$ and $\nu_{\max} \triangleq \max_{i \in \{1,\ldots,L\}} \nu_i$, we have

Theorem 8: The complexity of Algorithm 4 is bounded by

$$O(N_{\mathrm{it}} \deg \Lambda(x)) \leq O \min\left(\hat{D}D + LD^2,\ L\nu_{\max}^2\right) \tag{173}$$

additions and multiplications in F. □

Proof: The left side of (173) is obvious from Algorithm 4. The first term on the right side follows from (172) and Proposition 3. The second term on the right side follows from (171), $\hat{D} \leq L\nu_{\max}$, and Proposition 4. □

The complexity for decoding (as in Algorithm 1) will be addressed in Section VI.

Algorithm 5: Basic SPI Error-Locating Algorithm
(an implementation of Algorithm 1)

Input: $S^{(i)}(x)$, $\tilde{m}^{(i)}(x)$, n, and $k^{(i)}$
Output: nonzero $\Lambda(x) \in F[x]$, same as Algorithm 1.
Use Algorithm 3 with $b^{(i)}(x) = S^{(i)}(x)$, $m^{(i)}(x) = \tilde{m}^{(i)}(x)$, $\tau^{(i)} = n - k_{\min}$, and with the following three modifications: first, $\tau^{(i)} = \tau$ does not depend on i (but it is decreased during the algorithm, see below); second, initialize also $d := n - k_{\min}$; third, replace line 13 of Algorithm 3 with the following lines:

```
71   if $\delta \leq 0$ begin
72       if $\deg \Lambda(x) > n - k_{\max}$ return "decod.
           failure"
73       if $d \leq \deg \Lambda(x)$ return $\Lambda(x)$
74       else begin
75           $\tau := \tau - 1$
76           $\delta := \delta + 1$
77       end
78   end
```

Algorithm 6: Monomial-SPI Error-Locating Algorithm
(an implementation of Algorithm 2)

Input: $\breve{S}^{(i)}(x)$, n, and $k^{(i)}$
Output: nonzero $\Lambda(x) \in F[x]$, same as Algorithm 2.
The algorithm is Algorithm 4 with $b^{(i)}(x) = \breve{S}^{(i)}(x)$, $m^{(i)}(x) = x^{n-k^{(i)}}$, $\tau^{(i)} = n - k_{\min}$, and with modifications as in Algorithm 5.

VI. Using the SPI Algorithm for Decoding

As described in Section III-C, the SPI algorithm can be used to compute (an estimate of) the error locator polynomial of an interleaved Reed–Solomon code as in Section III. The preferred version of such a decoding algorithm is Algorithm 8, which is the final result of this section. We get there step by step, beginning with Algorithm 5 (see the framed boxes).

Algorithm 5 implements Algorithm 1 of Section III-C using Algorithm 3, and Algorithm 6 implements Algorithm 2 using Algorithm 4.

Line 73 makes sure that Algorithm 5 stops only when (105) is satisfied for all $i \in \{1, \ldots, L\}$: when this line is executed, we always have $d = \tau$ and

$$\deg\big(b^{(i)}(x)\Lambda(x) \bmod \tilde{m}^{(i)}(x)\big) < d. \tag{174}$$

Because line 73 checks the condition $d \leq \deg \Lambda(x)$ rather than (105), Algorithm 5 may terminate later (with a smaller value of τ) than Algorithm 1. In fact, from the moment where (105) holds (and assuming that the partial-inverse condition is satisfied), Algorithm 5 continues to decrease both d and τ (without changing $\Lambda(x) = \Lambda_E(x)$) until $d = |U_E|$.

Lemma 5: Assume that E satisfies the partial-inverse condition. Then Algorithm 5 stops with $\tau = |U_E|$. Moreover, the number N_{it} of executions of line 18 of Algorithm 3 is $L(n - k_{\min})$. □

Algorithm 7: Fixed-Iterations Algorithm
Input: $S^{(i)}(x)$, $\tilde{m}^{(i)}(x)$, n, and $k^{(i)}$
Output: nonzero $\Lambda(x) \in F[x]$, same as[a] Algorithm 1.
The algorithm is Algorithm 3 with $b^{(i)}(x) = S^{(i)}(x)$, $m^{(i)}(x) = \tilde{m}^{(i)}(x)$, $\tau^{(i)} = 0$, and with the following modifications: first, there is an extra integer variable N_{it} that is initialized to zero; second, line 13 is replaced with

```
81   if $N_{\text{it}} = L(n - k_{\min})$ return $\Lambda(x)$
```

and third, the extra line

$$N_{\text{it}} := N_{\text{it}} + 1$$

is inserted between lines 18 and 19.

[a]Except that the condition $\deg \Lambda(x) \leq n - k_{\max}$ is not checked inside the algorithm, but should be added as an external check.

Algorithm 8: Monomial-SPI Fixed-Iterations Algorithm

Input: $\breve{S}^{(i)}(x)$, n, and $k^{(i)}$
Output: nonzero $\Lambda(x) \in F[x]$, same as[a] Algorithm 2.
The algorithm is Algorithm 4 with $b^{(i)}(x) = \breve{S}^{(i)}(x)$, $m^{(i)}(x) = x^{n-k^{(i)}}$, $\tau^{(i)} = 0$, and with modifications as in Algorithm 7.

[a]See the footnote in Algorithm 7.

Proof: The first claim follows from the discussion above. From Theorem 7, we have

$$N_{\text{it}} = \hat{D} + L \deg \Lambda_E(x). \tag{175}$$

But $\hat{D} = L(n - k_{\min} - \tau)$ with $\tau = |U_E|$ when the algorithm stops. Thus $N_{\text{it}} = L(n - k_{\min})$. □

An immediate consequence is

Proposition 13 (Monom.-SPI Error Locating Complexity): Assume that E satisfies the partial-inverse condition. Then the complexity of Algorithm 6 is $O\big(L(n - k_{\min})(n - k_{\max})\big)$ additions and multiplications in F. □

In Lemma 5 and Proposition 13, the conditioning on the partial-inverse condition is unsatisfactory. But Lemma 5 suggests a solution to this problem: if N_{it} exceeds $L(n - k_{\min})$, the partial-inverse condition is not satisfied and it is pointless to continue. In other words, we can use the condition $N_{\text{it}} = L(n - k_{\min})$, rather than line 73, to stop the algorithm. The resulting error-locating algorithm (as a modification of Algorithm 3) is given as Algorithm 7 (see the box). The same modification can also be applied to Algorithm 6, resulting in Algorithm 8. We then have

Proposition 14: The complexity of Algorithm 8 (for arbitrary input) is $O\big(L(n - k_{\min})(n - k_{\max})\big)$ additions and multiplications in F. □

VII. Conclusion

We have introduced the SPI problem for polynomials and used it to generalize and to harmonize a number of ideas from the literature on decoding interleaved Reed–Solomon codes beyond half the minimum distance. The SPI problem has a unique solution (up to a scale factor), which can be computed by a (new) multi-sequence reverse Berlekamp–Massey algorithm.

The SPI problem with general moduli can always (and efficiently) be reduced to an SPI problem with monomial moduli. For monomial moduli, the reverse Berlekamp–Massey algorithm looks very much like (and has the same complexity as) the multi-sequence Berlekamp–Massey algorithm of [7] and [8].

The SPI problem can be used to analyze syndrome-based decoding of interleaved Reed–Solomon codes. Specifically, we pointed out a natural partial-inverse condition for the error pattern, which is always satisfied up to half the minimum distance and very likely to be satisfied almost up to the full minimum distance. If that condition is satisfied, the (true) error locator polynomial is the unique solution of a standard key equation and can be computed in many different ways, including the algorithm of [7] and [8] and the reverse Berlekamp–Massey algorithm of this paper. Two of the best performance bounds (for two different decoding algorithms) from the literature were rederived and generalized so that they apply to the partial-inverse condition, and thus simultaneously to many different decoding algorithms.

In Appendix B, we also give two easy variations of the reverse Berlekamp–Massey algorithm, one of which is a Euclidean algorithm. However, for $L > 1$, these variations have higher complexity than the reverse Berlekamp–Massey algorithm with monomial moduli.

Appendix A
Proof of the SPI Algorithm

In this appendix, we prove Theorems 6 and 7.

A. Assertions (Properties of the Algorithm)

To prove the correctness of Algorithm 3, we augment it with some extra variables and some assertions as shown in Algorithm 9. We will prove these assertions one by one, except that the proof of Assertion (A.1) is deferred to the end of this section.

Assertion (A.2) is obvious both from the initialization and from (A.11). Assertion (A.3) is the result of the **repeat** loop, as discussed at the beginning of Section V-A.

Assertion (A.4) is obvious. Assertions (A.5)–(A.8) follow from (A.2)–(A.4), followed by the swap in lines 21–23.

As for (A.9), when $b^{(i)}(x)$ is visited for the very first time (i.e., the first execution of line 26 for some index i), we have $d = \deg m^{(i)}(x)$ and $\mathrm{rd}^{(i)}(\Lambda) < d$ is obvious. For all later executions of line 26, we have $d = \mathrm{rd}^{(i)}(\Lambda)$ and $d^{(i)} = \mathrm{rd}^{(i)}(\Lambda^{(i)})$ before line 26, and $\mathrm{rd}^{(i)}(\Lambda) < d$ after line 26 follows from Lemma 4.

To prove (A.10) and (A.11), we note that Line 26 changes the degree of $\Lambda(x)$ only in iterations where lines 21–24 are executed, see (176) below; every later executions of Line 26 for the fixed i does not change $\deg \Lambda(x)$ because of Lemma 4 and that $\Lambda^{(i)}(x)$ and $d^{(i)}$ remain the same during the inner repeat loop.

Algorithm 9: Annotated SPI Algorithm

```
1   for $i = 1, \ldots, L$ begin
2       $\Lambda^{(i)}(x) := 0$
3       $d^{(i)} := \deg m^{(i)}(x)$
4       $\kappa^{(i)} := \mathrm{lcf}\, m^{(i)}(x)$
5   end
6   $\Lambda(x) := 1$
7   $\delta := \max_{i\in\{1,\ldots,L\}} \big(\deg m^{(i)}(x) - \tau^{(i)}\big)$
8   $i := 1$
    Extra:
    $k := 0$                                                    (E.1)
9   loop begin
        Assertions:
        $\deg \Lambda(x) = \sum_{i=1}^{L} \big(\deg m^{(i)}(x) - d^{(i)}\big)$    (A.1)
        $\deg \Lambda(x) > \deg \Lambda^{(i)}(x), \quad i = 1, \ldots, L$       (A.2)
10      repeat
11          if $i > 1$ begin $i := i - 1$ end
12          else begin
13              if $\delta \le 0$ return $\Lambda(x)$
14              $i := L$
15              $\delta := \delta - 1$
16          end
17          $d := \delta + \tau^{(i)}$
18          $\kappa :=$ coefficient of $x^d$ in
                    $b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x)$
19      until $\kappa \neq 0$
        Assertion:
        $i = i_{\max}(\Lambda),\ \delta = \delta_{\max}(\Lambda) \ge 0$         (A.3)
20      if $d < d^{(i)}$ begin
            Assertion:
            $d^{(i)} > d = \delta + \tau^{(i)} \ge \tau^{(i)}$                 (A.4)
            Extras:
            $k := k + 1,\ i_k \triangleq i,\ \Lambda_k(x) \triangleq \Lambda(x),$
            $\Delta_k \triangleq d^{(i)} - d,\ d_k \triangleq d^{(i)}$          (E.2)
21          swap $(\Lambda(x), \Lambda^{(i)}(x))$
22          swap $(d, d^{(i)})$
23          swap $(\kappa, \kappa^{(i)})$
24          $\delta := d - \tau^{(i)}$
            Assertions:
            $d > d^{(i)} \ge \tau^{(i)}$                                     (A.5)

            $\deg \Lambda^{(i)}(x) > \deg \Lambda(x)$                         (A.6)
            $\deg \Lambda^{(i)}(x) > \deg \Lambda^{(j)}(x)$ for $j \neq i$     (A.7)
            $i_{\max}(\Lambda^{(i)}) = i,\ \delta_{\max}(\Lambda^{(i)}) \ge 0$  (A.8)
25      end
26      $\Lambda(x) := \kappa^{(i)}\Lambda(x) - \kappa x^{d-d^{(i)}}\Lambda^{(i)}(x)$
        Assertions:
        $\mathrm{rd}^{(i)}(\Lambda) < d = \delta + \tau^{(i)}$                (A.9)

        $\deg \Lambda(x) = \Delta_k + \deg \Lambda_k(x)$                      (A.10)
                $> \deg \Lambda^{(i)}(x), \quad i = 1, \ldots, L$           (A.11)
27  end
```

If lines 21–24 are executed, then line 26 changes the degree of $\Lambda(x)$ to

$$\deg \Lambda^{(i)}(x) + d - d^{(i)} = \deg \Lambda_k(x) + \Delta_k, \qquad (176)$$

which is (A.10). With (A.7), the left-hand side of (176) yields also (A.11).

It remains to prove (A.1). First, we note that (A.1) clearly holds when the **loop** is entered for the first time. But if (A.1) holds, then $\Lambda^{(i)}(x)$ in (A.6) satisfies

$$\deg \Lambda^{(i)}(x) = \sum_{j\neq i}^{L} \big(\deg m^{(j)}(x) - d^{(j)} \big) + \deg m^{(i)}(x) - d. \qquad (177)$$

It then follows from (176) that $\Lambda(x)$ after line 26 satisfies

$$\deg \Lambda^{(i)}(x) + d - d^{(i)} = \sum_{j=1}^{L} \big(\deg m^{(j)}(x) - d^{(j)} \big), \qquad (178)$$

which is (A.1). (Note that (A.1) and (A.4) together provide an alternative proof of Proposition 3.)

Finally, we note that the algorithm is guaranteed to terminate because every execution of the **repeat** loop (lines 10–19) strictly decreases $\delta_{\max}(\Lambda)$ or $i_{\max}(\Lambda)$ according to Lemma 4 and the swap in lines 21–23 strictly decreases $d^{(i)}$.

For later use, we also record the following fact from (E.2) and (A.10): Let $\Lambda_1(x), \Lambda_2(x), \ldots, \Lambda_K(x)$ be all polynomials $\Lambda_k(x)$ from (E.2) and let $\hat{\Lambda}(x)$ be the $\Lambda(x)$ returned by the algorithm. (Note that $\deg \hat{\Lambda}(x) > \deg \Lambda_K(x)$.)

Proposition 15: The polynomials $\Lambda_k(x)$ defined in (E.2) satisfy $\deg \Lambda_1(x) = 0$ (since $\Lambda_1(x) = 1$) and

$$\deg \Lambda_K(x) > \ldots > \deg \Lambda_2(x) > \deg \Lambda_1(x) \qquad (179)$$

with

$$\Delta_k = \deg \Lambda_{k+1}(x) - \deg \Lambda_k(x) \qquad (180)$$

for $k \in \{1, \ldots, K-1\}$ and

$$\Delta_K = \deg \hat{\Lambda}(x) - \deg \Lambda_K(x). \qquad (181)$$

Moreover, we have

$$\Delta_k = d_k - \deg \Big(b^{(i_k)}(x)\Lambda_k(x) \bmod m^{(i_k)}(x) \Big) \qquad (182)$$

for $k = 1, 2, \ldots, K$. □

B. Completing the Proof of Theorem 6

It is clear at this point that the algorithm terminates and the returned polynomial $\Lambda(x) = \hat{\Lambda}(x)$ satisfies (1) for all $i \in \{1, \ldots, L\}$. Below, we will show that any nonzero $\tilde{\Lambda}(x) \in F[x]$ with $\deg \tilde{\Lambda}(x) < \deg \hat{\Lambda}(x)$ cannot satisfy (1) for all i.

To this end, we need Proposition 15 and the lemma.

Lemma 6: For any nonzero $q_k(x)$ with $\deg q_k < \Delta_k$, we have

$$i_{\max}(q_k\Lambda_k) = i_{\max}(\Lambda_k) \qquad (183)$$

and

$$\begin{aligned} \mathrm{rd}^{(i_k)}(q_k\Lambda_k) &= \deg q_k + \mathrm{rd}^{(i_k)}(\Lambda_k) &(184)\\ &< d_k. &(185)\end{aligned}$$

Moreover, for any nonzero $q_k\Lambda_k$ and $q_{k'}\Lambda_{k'}$ with

$$i_{\max}(\Lambda_k) = i_{\max}(\Lambda_{k'}) \qquad (186)$$

(i.e., $i_k = i_{k'}$), we have

$$\mathrm{rd}^{(i_k)}(q_k\Lambda_k + q_{k'}\Lambda_{k'}) = \mathrm{rd}^{(i_k)}(q_k\Lambda_k) \qquad (187)$$

if $k < k'$. □

Proof: First, we establish from $\deg q_k < \Delta_k$ and (182) that

$$\deg q_k(x) + \deg \Big(b^{(i_k)}(x)\Lambda_k(x) \bmod m^{(i_k)}(x) \Big) < d_k \qquad (188)$$

for $k = 1, 2, \ldots, K$. For all $i \in \{1, \ldots, L\}$, we clearly have

$$\mathrm{rd}^{(i)}(q_k\Lambda_k) \le \deg q_k + \mathrm{rd}^{(i)}(\Lambda_k). \qquad (189)$$

For i_k, however, we have

$$\mathrm{rd}^{(i_k)}(q_k\Lambda_k) = \deg q_k + \mathrm{rd}^{(i_k)}(\Lambda_k) \qquad (190)$$

from (188) and since $d_k \le \deg m^{(i_k)}(x)$. The first claim of the lemma then follows from $i_{\max}(\Lambda_k) = i_k$; the second claim (187) is clear from

$$\mathrm{rd}^{(i_k)}(q_{k'}\Lambda_{k'}) < d_{k'} \le \mathrm{rd}^{(i_k)}(q_k\Lambda_k) < d_k \qquad (191)$$

for $k < k'$. □

Partitioning the indices $k \in \{1, \ldots, K\}$ into sets $S_1, \ldots, S_L$ such that

$$k \in S_i \iff i_{\max}(\Lambda_k) = i_k = i, \qquad (192)$$

we obtain from Lemma 6 the following corollary.

Corollary 1: For any S_i, any nonzero

$$\tilde{\Lambda}^{(i)}(x) \overset{\triangle}{=} \sum_{k\in S_i} q_k(x)\Lambda_k(x) \qquad (193)$$

with $\deg q_k < \Delta_k$ satisfies

$$i_{\max}(\tilde{\Lambda}^{(i)}) = i \qquad (194)$$

and

$$\mathrm{rd}^{(i)}(\tilde{\Lambda}^{(i)}) = \max_{k\in S_i} \big(q_k\Lambda_k \big) \qquad (195)$$

□

Finally, we note that any nonzero $\tilde{\Lambda}(x) \in F[x]$ with $\deg \tilde{\Lambda}(x) < \deg \hat{\Lambda}(x)$ can be uniquely written as

$$\tilde{\Lambda}(x) = \sum_{k=1}^{K} q_k(x)\Lambda_k(x) \qquad (196)$$

for some nonzero $q_k(x)$ with $\deg q_k(x) < \Delta_k$. It then follows from Corollary 1 that $\tilde{\Lambda}(x) = \sum_{i=1}^{L} \tilde{\Lambda}^{(i)}(x)$ cannot satisfy (1) for all i.

C. Proof of Theorem 7

For each $i \in \{1, \ldots, L\}$, let $d^{(i)}$ be as in the algorithm, and let $\tilde{d}^{(i)}$ denote the value of $d^{(i)}$ when the algorithm stops. Note that $d^{(i)}$ (for each i) is initialized to $\deg m^{(i)}(x)$ in line 3. Now let $\delta^{(i)} \overset{\triangle}{=} d^{(i)} - \tau^{(i)}$ for every $i \in \{1, \ldots, L\}$. Clearly, $\delta^{(i)}$ satisfies

$$\tilde{d}^{(i)} - \tau^{(i)} \le \delta^{(i)} \le \deg m^{(i)}(x) - \tau^{(i)}; \qquad (197)$$

moreover, every execution of the swap in lines 21–23 strictly reduces $\delta^{(i)}$. Finally, let δ be as in the algorithm, which is initialized to

$$\delta := \max_{i\in\{1,\ldots,L\}} \left(\deg m^{(i)}(x) - \tau^{(i)}\right) \tag{198}$$

(see line 7). It is obvious that the number N_{it} of executions of line 18 of Algorithm 3 (i.e., Algorithm 9) is equal to the total number of iterations of lines 10–19. These executions of line 18 (with the help of line 26) are made to make $\Lambda(x)$ satisfy (1) for all $i \in \{1,\ldots,L\}$ (which holds when $\delta \le 0$) accompanied by the reduction of $\delta^{(i)}$ from $\deg m^{(i)}(x) - \tau^{(i)}$ to $\tilde{d}^{(i)} - \tau^{(i)}$ for every i. We therefore have

$$N_{\text{it}} = \hat{D} + \sum_{i=1}^{L} n_{\text{it}}^{(i)}, \tag{199}$$

where $\hat{D}$ is defined in (170) and where $n_{\text{it}}^{(i)}$ denotes the number of executions of line 18 needed for decreasing $\delta^{(i)}$ from $\deg m^{(i)}(x) - \tau^{(i)}$ to $\tilde{d}^{(i)} - \tau^{(i)}$. The quantity $n_{\text{it}}^{(i)}$ for each $i \in \{1,\ldots,L\}$ is

$$\begin{aligned} n_{\text{it}}^{(i)} &= L \cdot \left(\deg m^{(i)}(x) - \tau^{(i)} - (\tilde{d}^{(i)} - \tau^{(i)})\right) && (200)\\ &= L \cdot \left(\deg m^{(i)}(x) - \tilde{d}^{(i)}\right) && (201)\end{aligned}$$

and thus

$$\sum_{i=1}^{L} n_{\text{it}}^{(i)} = L \cdot \sum_{i=1}^{L} \left(\deg m^{(i)}(x) - \tilde{d}^{(i)}\right). \tag{202}$$

We therefore obtain

$$N_{\text{it}} = \hat{D} + L \cdot \sum_{i=1}^{L} \left(\deg m^{(i)}(x) - \tilde{d}^{(i)}\right). \tag{203}$$

But $\sum_{i=1}^{L} \left(\deg m^{(i)}(x) - \tilde{d}^{(i)}\right) = \deg \Lambda(x)$ from (A.1) with $d^{(i)} = \tilde{d}^{(i)}$.

Appendix B
Quotient Saving Algorithm and Remainder Saving Algorithm

For $L = 1$, the reverse Berlekamp–Massey algorithm is easily translated into two other algorithms, one of which is a Euclidean algorithm [3]. In fact, it is a main point of [3] that these algorithms may be viewed as different versions of a single algorithm. We now demonstrate that this works also for $L > 1$.

However, for $L > 1$, these other algorithms are less attractive than the monomial-SPI reverse Berlekamp–Massey algorithm (Algorithm 4) as will be detailed below. However, before discounting these other algorithms from future research, it may be remembered that the complexity of the asymptotically fast MLFSR algorithms of [5] and [22] is cubic in L while the complexity of the algorithms below is only quadratic in L.

Algorithm 10: Quotient Saving SPI Algorithm

Input: $b^{(i)}(x), m^{(i)}(x), \tau^{(i)}$ for $i = 1,\ldots,L$.

Output: $\Lambda(x)$ as in the problem statement.

```
1   for $i = 1,\ldots,L$ begin
2       $\Lambda^{(i)}(x) := 0$
3       $d^{(i)} := \deg m^{(i)}(x)$
4       $\kappa^{(i)} := \operatorname{lcf} m^{(i)}(x)$
5       for $j = 1,\ldots,L$ begin
6           $Q^{(i,j)}(x) := 0$
7           if $i = j$ begin $Q^{(i,j)}(x) := -1$ end
8       end
9   end
10  $\Lambda(x) := 1$
11  for $i = 1,\ldots,L$ begin $Q^{(i)}(x) := 0$ end
12  $\delta := \max_{i\in\{1,\ldots,L\}} \left(\deg m^{(i)}(x) - \tau^{(i)}\right)$
13  $i := 1$
14  loop begin
15      repeat
16          if $i > 1$ begin $i := i - 1$ end
17          else begin
18              if $\delta \le 0$ return $\Lambda(x)$
19              $i := L$
20              $\delta := \delta - 1$
21          end
22          $d := \delta + \tau^{(i)}$
23          $\kappa := \sum_{\ell} b^{(i)}_{d-\ell}\Lambda_\ell - \sum_{\ell} m^{(i)}_{d-\ell} Q^{(i)}_{\ell}$
24      until $\kappa \neq 0$
        ------------------------------------------------
25      if $d < d^{(i)}$ begin
26          swap $(\Lambda(x), \Lambda^{(i)}(x))$
27          swap $(d, d^{(i)})$
28          swap $(\kappa, \kappa^{(i)})$
29          for $j = 1,\ldots,L$ swap $(Q^{(j)}(x), Q^{(i,j)}(x))$
30          $\delta := d - \tau^{(i)}$
31      end
        ------------------------------------------------
32      $\Lambda(x) := \kappa^{(i)}\Lambda(x) - \kappa x^{d-d^{(i)}}\Lambda^{(i)}(x)$
33      for $j = 1,\ldots,L$ begin
34          $Q^{(j)}(x) := \kappa^{(i)} Q^{(j)}(x) - \kappa x^{d-d^{(i)}} Q^{(i,j)}(x)$
35      end
36  end
```

A. Quotient Saving Algorithm

Algorithm 10 (see box) is a variation of Algorithm 3 that achieves a generalization of Algorithm 4 to general $m^{(i)}(x)$. To this end, we store and update the quotients $Q^{(i)}(x)$, $i = 1,\ldots,L$, defined by

$$b^{(i)}(x)\Lambda(x) = Q^{(i)}(x)m^{(i)}(x) + r^{(i)}(x) \tag{204}$$

with $r^{(i)}(x) \overset{\triangle}{=} b^{(i)}(x)\Lambda(x) \bmod m^{(i)}(x)$. The coefficient of x^d of $r^{(i)}(x)$ in line 18 of Algorithm 3 can then be computed as in line 23 of Algorithm 10.

Algorithm 11: Remainder Saving SPI Algorithm
(a Euclidean algorithm)
Input: $b^{(i)}(x), m^{(i)}(x), \tau^{(i)}$ for $i = 1, \ldots, L$.
Output: $\Lambda(x)$ as in the problem statement.

```
1    for $i = 1, \ldots, L$ begin
2        $\Lambda^{(i)}(x) := 0$
3        $d^{(i)} := \deg m^{(i)}(x)$
4        $\kappa^{(i)} := \operatorname{lcf} m^{(i)}(x)$
5        for $j = 1, \ldots, L$ begin
6            $r^{(i,j)}(x) := 0$
7            if $i = j$ begin $r^{(i,j)}(x) := m^{(i)}(x)$ end
8        end
9    end
10   $\Lambda(x) := 1$
11   for $i = 1, \ldots, L$ begin $r^{(i)}(x) := b^{(i)}(x)$ end
12   $\delta := \max_{i\in\{1,\ldots,L\}} \big(\deg m^{(i)}(x) - \tau^{(i)}\big)$
13   $i := 1$
14   loop begin
15       repeat
16           if $i > 1$ begin $i := i - 1$ end
17           else begin
18               if $\delta \leq 0$ return $\Lambda(x)$
19               $i := L$
20               $\delta := \delta - 1$
21           end
22           $d := \delta + \tau^{(i)}$
23           $\kappa :=$ coefficient of $x^d$ of $r^{(i)}(x)$
24       until $\kappa \neq 0$
         ________________________________
25       if $d < d^{(i)}$ begin
26           swap $(\Lambda(x), \Lambda^{(i)}(x))$
27           swap $(d, d^{(i)})$
28           swap $(\kappa, \kappa^{(i)})$
29           for $j = 1, \ldots, L$ swap $(r^{(j)}(x), r^{(i,j)}(x))$
30           $\delta := d - \tau^{(i)}$
31       end
         ________________________________
32       $\Lambda(x) := \kappa^{(i)}\Lambda(x) - \kappa x^{d-d^{(i)}}\Lambda^{(i)}(x)$
33       for $j = 1, \ldots, L$ begin
34           $r^{(j)}(x) := \kappa^{(i)} r^{(j)}(x) - \kappa x^{d-d^{(i)}} r^{(i,j)}(x)$
35       end
36   end
```

The quotients $Q^{(i)}(x)$ of (204) are initialized in line 11, updated in line 34, and stored (as $Q^{(i,j)}(x)$) in line 29, in parallel with $\Lambda(x)$.

All other quantities in the algorithm remain unchanged. In any case (as in (168)), we always have

$$\deg\big(b^{(i)}(x)\Lambda(x) - Q^{(i)}(x)m^{(i)}(x)\big) < d \tag{205}$$

after executing lines 33–35.

Theorem 7 still applies, with "line 18" replaced by "line 23". Due to the additional computation of lines 33–35, the complexity of Algorithm 10 is

$$O\big(N_{\text{it}} L \deg \Lambda(x)\big) \leq O\big(L(\hat{D}D + LD^2)\big). \tag{206}$$

Compared with (173), the factor L in (206) makes this algorithm less attractive for $L > 1$ than Algorithm 4.

B. Remainder Saving Algorithm

Another variation of Algorithm 3 is Algorithm 11, where we store and update the remainders $r^{(i)}(x)$ from (204). In consequence, the computation of line 18 of Algorithm 3 is unnecessary and replaced by the trivial line 23 of Algorithm 11. However, updating the remainders $r^{(i)}(x)$ requires the additional computation in lines 33–35.

Otherwise, the algorithm works exactly like Algorithms 3 and 10. In particular, we always have

$$\deg r^{(i)}(x) < d \tag{207}$$

after executing lines 33–35.

Note that this algorithm is rather a Euclidean algorithm [6], [9] than a Berlekamp–Massey algorithm (but it is a new algorithm as well).

Due to the computation of lines 33–35, the complexity of Algorithm 11 is

$$O(N_{\text{it}} L \nu_{\max}) \tag{208}$$

with

$$\nu_{\max} \triangleq \max_{i\in\{1,\ldots,L\}} \deg m^{(i)}(x). \tag{209}$$

But

$$N_{\text{it}} L \nu_{\max} \geq N_{\text{it}} \deg \Lambda(x) \tag{210}$$

by (24). It follows that the complexity of Algorithm 11 is never smaller than the complexity (173) of Algorithm 4. In particular, for monomial moduli, we have $\deg \Lambda(x) \leq \nu_{\max}$ (by Proposition 4), i.e., the left side of (210) exceeds the right side by a factor of L.

For general SPI problems, the difference between the left side and the right side of (210) may be small, in which case Algorithm 11 may be more attractive than first monomializing the SPI problem and then using Algorithm 4. Indeed, (210) is an equality if $\deg \Lambda(x) = L\nu_{\max}$, which happens in the very special case where

$$\operatorname{lcm}\big\{m^{(1)}(x), \ldots, m^{(L)}(x)\big\} = m^{(1)}(x) \cdots m^{(L)}(x), \tag{211}$$

$\deg m^{(1)}(x) = \ldots = \deg m^{(L)}(x) = \nu_{\max}$, and $\tau^{(1)} = \ldots = \tau^{(L)} = 0$. However, this case does not arise in decoding as in this paper.

ACKNOWLEDGMENT

The paper has much benefitted from the comments by the anonymous reviewers.

REFERENCES

[1] J.-H. Yu and H.-A. Loeliger, "An algorithm for simultaneous partial inverses," in *Proc. 52nd Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, USA, Sep./Oct. 2014, pp. 928–935.

[2] J.-H. Yu and H.-A. Loeliger, "Decoding of interleaved Reed–Solomon codes via simultaneous partial inverses," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Hong Kong, Jun. 2015, pp. 2396–2400.

[3] J.-H. Yu and H.-A. Loeliger, "Partial inverses mod $m(x)$ and reverse Berlekamp–Massey decoding," *IEEE Trans. Inf. Theory*, vol. 62, no. 12, pp. 6737–6756, Dec. 2016.

[4] G. Schmidt, V. R. Sidorenko, and M. Bossert, "Collaborative decoding of interleaved Reed–Solomon codes and concatenated code designs," *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 2991–3012, Jul. 2009.

[5] J. S. R. Nielsen, "Generalised multi-sequence shift-register synthesis using module minimisation," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Istanbul, Turkey, Jul. 2013, pp. 882–886.

[6] G.-L. Feng and K. K. Tzeng, "A generalized Euclidean algorithm for multisequence shift-register synthesis," *IEEE Trans. Inf. Theory*, vol. 35, no. 3, pp. 584–594, May 1989.

[7] G.-L. Feng and K. K. Tzeng, "A generalization of the Berlekamp-Massey algorithm for multisequence shift-register synthesis with applications to decoding cyclic codes," *IEEE Trans. Inf. Theory*, vol. 37, no. 5, pp. 1274–1287, Sep. 1991.

[8] G. Schmidt and V. R. Sidorenko, "Multi-sequence linear shift-register synthesis: The varying length case," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Seattle, WA, USA, Jul. 2006, pp. 1738–1742.

[9] J. V. Z. Gathen and J. Gerhard, *Modern Computer Algebra*, 3rd ed. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[10] J. R. N. Nielsen and A. Storjohann, "Algorithms for simultaneous Padé approximations," in *Proc. ACM Int. Symp. Symbolic Algebr. Comput. (ISSAC)*, Waterloo, ON, Canada, Jul. 2016, pp. 405–412.

[11] S. Puchinger and J. R. D. Nielsen, "Decoding of interleaved Reed–Solomon codes using improved power decoding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Aachen, Germany, Jun. 2017, pp. 356–360.

[12] A. Brown, L. Minder, and A. Shokrollahi, "Probabilistic decoding of interleaved RS-codes on the Q-ary symmetric channel," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Chicago, IL, USA, Jun./Jul. 2004, p. 326.

[13] F. Parvaresh, M. H. Taghavi, and A. Vardy, "On the performance of multivariate interpolation decoding of Reed–Solomon codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Seattle, WA, USA, Jul. 2006, pp. 2027–2031.

[14] V. Guruswami and M. Sudan, "Improved decoding of Reed–Solomon and algebraic-geometric codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 1755–1764, Sep. 1999.

[15] D. Bleichenbacher, A. Kiayias, and M. Yung, *Decoding of Interleaved Reed Solomon Codes over Noisy Data* (Lecture Notes in Computer Science), vol. 2719. 2003, pp. 97–108.

[16] F. Parvaresh and A. Vardy, "Multivariate interpolation decoding beyond the Guruswami-Sudan radius," in *Proc. 42nd Annu. Allerton Conf. Commun. Control, Comput.*, Urbana, IL, USA, Oct. 2004, p. 1362.

[17] J. J. Metzner and E. J. Kapturowski, "A general decoding technique applicable to replicated file disagreement location and concatenated code decoding," *IEEE Trans. Inf. Theory*, vol. 36, no. 4, pp. 911–917, Jul. 1990.

[18] C. Haslach and A. J. H. Vinck, "A decoding algorithm with restrictions for array codes," *IEEE Trans. Inf. Theory*, vol. 45, no. 7, pp. 2339–2344, Nov. 1999.

[19] H. Kurzweil, M. Seidl, and J. B. Huber, "Reduced-complexity collaborative decoding of interleaved Reed–Solomon and Gabidulin codes," in *Proc. IEEE Int. Symp. Inf. Theory*, Saint Petersburg, Russia, Jul./Aug. 2011, pp. 2557–2561.

[20] R. M. Roth and P. O. Vontobel, "Coding for combined block–symbol error correction," *IEEE Trans. Inf. Theory*, vol. 60, no. 5, pp. 2697–2713, May 2014.

[21] L.-P. Wang, Q.-L. Wang, and K.-P. Wang, "A lattice-based linear shift register synthesis for multisequences of varying length," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2008, pp. 1751–1754.

[22] V. Sidorenko and M. Bossert, "Fast skew-feedback shift-register synthesis," *Designs, Codes and Cryptography*, vol. 70. New York, NY, USA: Springer, 2014, pp. 55–67.

[23] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inf. Theory*, vol. IT-15, no. 1, pp. 122–127, Jan. 1969.

[24] Y. Sugiyama, M. Kasahara, S. Hirasawa, and T. Namekawa, "A method for solving key equation for decoding Goppa codes," *Inf. Control*, vol. 27, no. 1, pp. 87–99, 1975.

[25] M. Bossert and S. Bezzateev, "Decoding of interleaved RS codes with the Euclidean algorithm," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Toronto, ON, Canada, Jul. 2008, pp. 1803–1807.

[26] R. E. Blahut, *Algebraic Codes for Data Transmission*. Cambridge, U.K.: Cambridge Univ. Press, 2003.

[27] J. Justesen and T. Hoholdt, *A Course in Error-Correcting Codes*. Zürich, Switzerland: European Mathematical Society, 2004.

[28] R. Roth, *Introduction to Coding Theory*. Cambridge, U.K.: Cambridge Univ. Press, 2006.

[29] I. N. Herstein, *Topics Algebra*, 2nd ed. Hoboken, NJ, USA: Wiley, 1975.

[30] A. Shiozaki, "Decoding of redundant residue polynomial codes using Euclid's algorithm," *IEEE Trans. Inf. Theory*, vol. IT-34, no. 5, pp. 1351–1354, Sep. 1988.

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

Jiun-Hung Yu (S'10–M'14) was born in Nantou, Taiwan, in 1979. He received the M.S. degree in communication engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2003, and the Ph.D. degree in electrical engineering from ETH Zurich, in 2014. From 2003 to 2008, he was with Realtek Semiconductor Cooperation, Hsinchu, Taiwan. From 2008 to 2017, he was with the Signal and Information Processing Laboratory of ETH Zurich. Since 2017, he has been with National Chiao Tung University. He is currently an Assistant Professor. His research interests include communication theory, error-correcting codes, and statistical signal processing.

Hans-Andrea Loeliger (S'85–M'92–SM'03–F'04) received the Diploma in electrical engineering and the Ph.D. degree in 1992 from ETH Zurich, Switzerland. From 1992 to 1995, he was with Linköping University, Linköping Sweden. From 1995 to 2000, he was a technical consultant and coowner of a consulting company. Since 2000, he has been a Professor with the Department of Information Technology and Electrical Engineering of ETH Zurich, Switzerland. His research interests include the broad areas of signal processing, machine learning, information theory, error correcting codes, communications, and electronic circuits.